Bu kitaba sığmayan daha neler var!

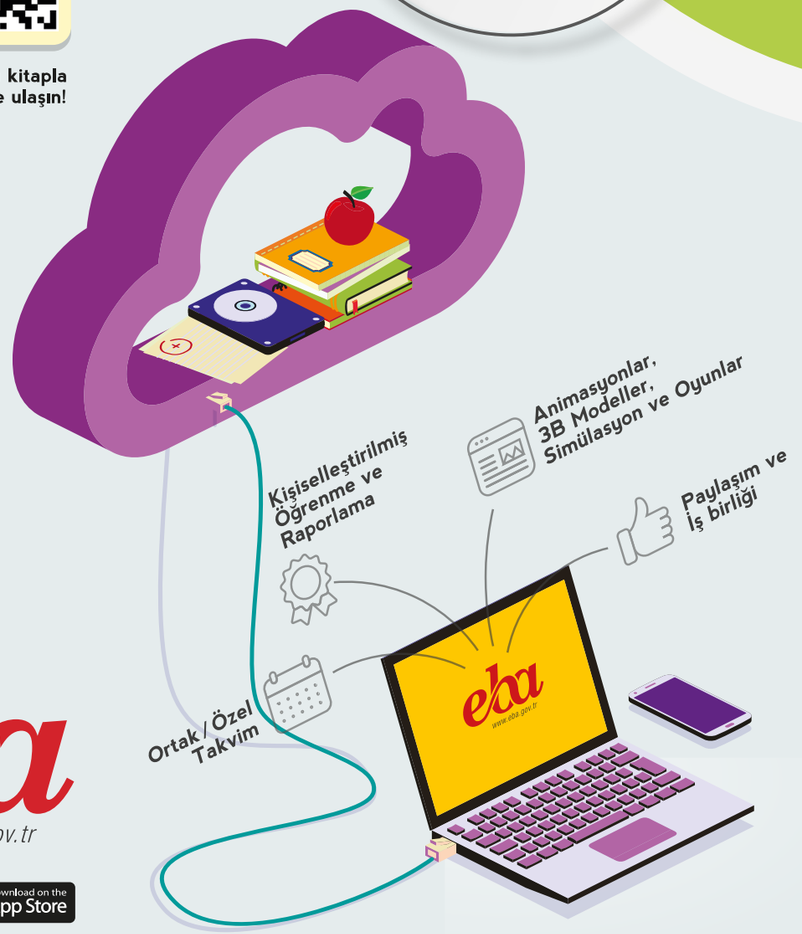
Karekodu okutun, bu kitapla ilgili EBA içeriklerine ulaşın!

ÖDS

ÖĞRENCİ/ÖĞRETMEN DESTEK SİSTEMİ

https://ods.eba.gov.tr

- Konu Anlatımlı Ders Videoları
- Soru Çözüm Videoları
- Ders Anlatım Videoları
- Çoktan Seçmeli Sorular



eba
www.eba.gov.tr

Google play | App Store

BU DERS KİTABI MİLLÎ EĞİTİM BAKANLIĞINCA ÜCRETSİZ OLARAK VERİLMİŞTİR.
PARA İLE SATILAMAZ.

ISBN: 978-975-11-7117-7

Bandrol Uygulamasına İlişkin Usul ve Esaslar Hakkında Yönetmelik'in 5'inci Maddesinin İkinci Fıkrası Çerçevesinde Bandrol Taşıması Zorunlu Değildir.

BİLİŞİM TEKNOLOJİLERİ ALANI

WEB TABANLI UYGULAMA GELİŞTİRME

11

DERS MATERYALİ

MESLEKİ VE TEKNİK ANADOLU LİSESİ

BİLİŞİM TEKNOLOJİLERİ ALANI

11 DERS MATERYALİ

WEB TABANLI UYGULAMA GELİŞTİRME

MESLEKİ VE TEKNİK ANADOLU LİSESİ

BİLİŞİM TEKNOLOJİLERİ ALANI

WEB TABANLI UYGULAMA GELİŞTİRME 11

DERS MATERYALİ

YAZARLAR

Abdullah HOCAOĞLU
Bülent ALTINBAŞ
Devrim ALTINKURT
Hakan BABAÇ
Mustafa NACAR
Özgü ASKER

MİLLÎ EĞİTİM BAKANLIĞI YAYINLARI 8337
YARDIMCI VE KAYNAK KİTAPLAR DİZİSİ 2229

HAZIRLAYANLAR

Dil Uzmanı
Erman Erşan YORGANCILAR

Program Geliştirme Uzmanı
Ergül SIRKINTI

Ölçme ve Değerlendirme Uzmanı
Filiz İSNAÇ

Rehberlik Uzmanı
Gülşen YALIN

Görsel Tasarım Uzmanı
Sermin FIRAT SOYDAN

ISBN: 978-975-11-7117-7

Millî Eğitim Bakanlığının 24.12.2020 gün ve 18433886 sayılı oluru ile Meslekî ve Teknik Eğitim Genel Müdürlüğünce ders materyali olarak hazırlanmıştır.

İSTİKLÂL MARŞI

Korkma, sönmez bu şafaklarda yüzen al sancak;
Sönmeden yurdumun üstünde tüten en son ocak.
O benim milletimin yıldızıdır, parlayacak;
O benimdir, o benim milletimindir ancak.

Çatma, kurban olayım, çehreni ey nazlı hilâl!
Kahraman ırkıma bir gül! Ne bu şiddet, bu celâl?
Sana olmaz dökülen kanlarımız sonra helâl.
Hakkıdır Hakk'a tapan milletimin istiklâl.

Ben ezelden beridir hür yaşadım, hür yaşarım.
Hangi çılgın bana zincir vuracakmış? Şaşarım!
Kükremiş sel gibiyim, bendimi çiğner, aşarım.
Yırtarım dağları, enginlere sığmam, taşarım.

Garbın âfâkını sarmışsa çelik zırhlı duvar,
Benim iman dolu göğsüm gibi serhaddim var.
Ulusun, korkma! Nasıl böyle bir imanı boğar,
Medeniyyet dediğin tek dişi kalmış canavar?

Arkadaş, yurduma alçakları uğratma sakın;
Siper et gövdeni, dursun bu hayâsızca akın.
Doğacaktır sana va'dettiği günler Hakk'ın;
Kim bilir, belki yarın, belki yarından da yakın.

Bastığın yerleri toprak diyerek geçme, tanı:
Düşün altındaki binlerce kefensiz yatanı.
Sen şehit oğlusun, incitme, yazıktır, atanı:
Verme, dünyaları alsan da bu cennet vatanı.

Kim bu cennet vatanın uğruna olmaz ki feda?
Şüheda fışkıracak toprağı sıksan, şüheda!
Cânı, cânânı, bütün varımı alsın da Huda,
Etmesin tek vatanımdan beni dünyada cüda.

Ruhumun senden İlâhî, şudur ancak emeli:
Değmesin mabedimin göğsüne nâmahrem eli.
Bu ezanlar -ki şehadetleri dinin temeli-
Ebedî yurdumun üstünde benim inlemeli.

O zaman vecd ile bin secde eder -varsa- taşım,
Her cerîhamdan İlâhî, boşanıp kanlı yaşım,
Fışkırır ruh-ı mücerret gibi yerden na'şım;
O zaman yükselerek arşa değer belki başım.

Dalgalan sen de şafaklar gibi ey şanlı hilâl!
Olsun artık dökülen kanlarımın hepsi helâl.
Ebediyyen sana yok, ırkıma yok izmihlâl;
Hakkıdır hür yaşamış bayrağımın hürriyyet;
Hakkıdır Hakk'a tapan milletimin istiklâl!

Mehmet Âkif Ersoy

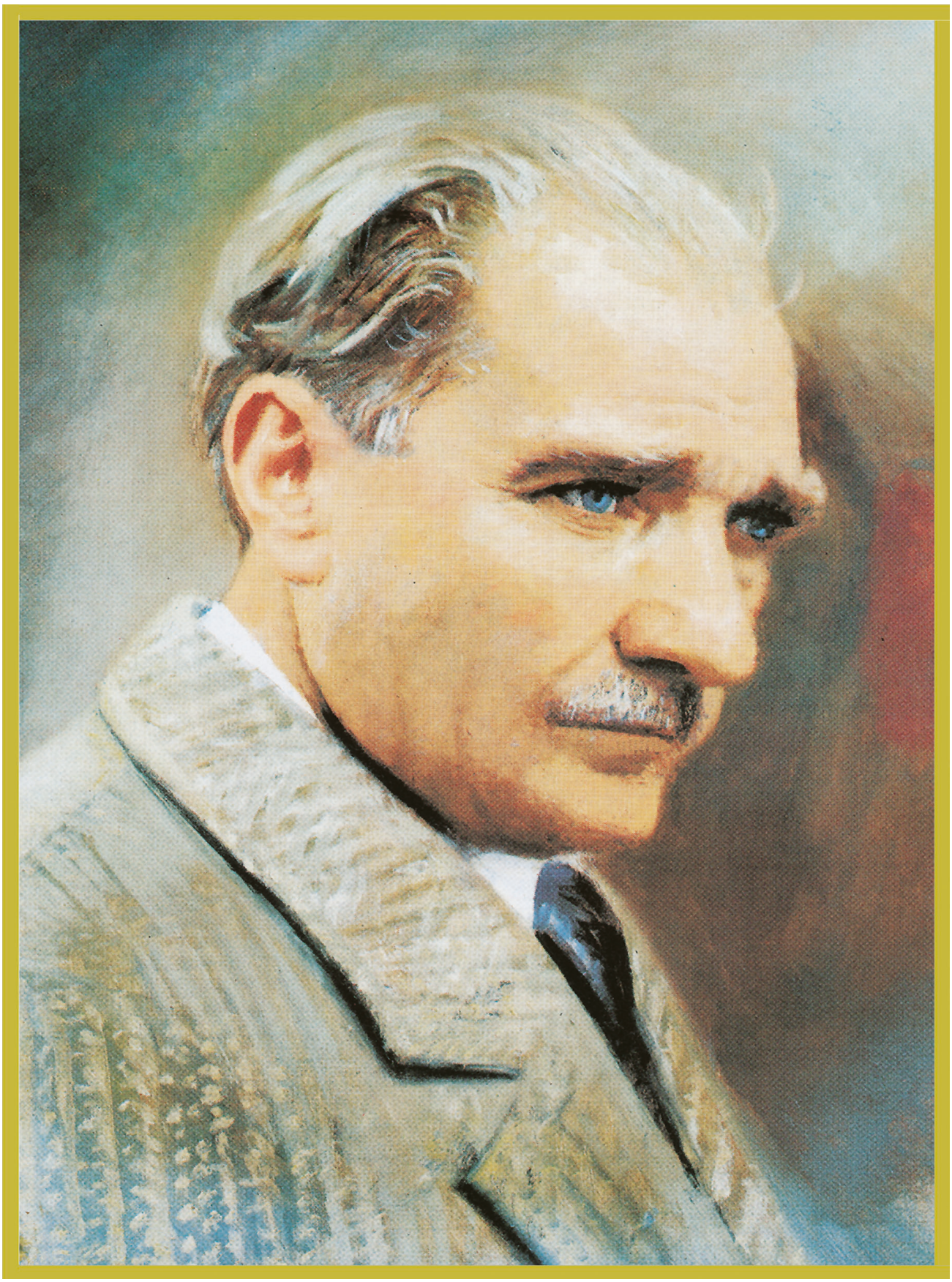
GENÇLİĞE HİTABE

Ey Türk gençliği! Birinci vazifen, Türk istiklâlini, Türk Cumhuriyetini, ilelebet muhafaza ve müdafaa etmektir.

Mevcudiyetinin ve istikbalinin yegâne temeli budur. Bu temel, senin en kıymetli hazinendir. İstikbalde dahi, seni bu hazineden mahrum etmek isteyecek dâhilî ve hâricî bedhahların olacaktır. Bir gün, istiklâl ve cumhuriyeti müdafaa mecburiyetine düşersen, vazifeye atılmak için, içinde bulunacağın vaziyetin imkân ve şeraitini düşünmeyeceksin! Bu imkân ve şerait, çok namüsait bir mahiyette tezahür edebilir. İstiklâl ve cumhuriyetine kastedecek düşmanlar, bütün dünyada emsali görülmemiş bir galibiyetin mümessili olabilirler. Cebren ve hile ile aziz vatanın bütün kaleleri zapt edilmiş, bütün tersanelerine girilmiş, bütün orduları dağıtılmış ve memleketin her köşesi bilfiil işgal edilmiş olabilir. Bütün bu şeraitten daha elîm ve daha vahim olmak üzere, memleketin dâhilinde iktidara sahip olanlar gaflet ve dalâlet ve hattâ hıyanet içinde bulunabilirler. Hattâ bu iktidar sahipleri şahsî menfaatlerini, müstevlîlerin siyasî emelleriyle tevhit edebilirler. Millet, fakr u zaruret içinde harap ve bîtap düşmüş olabilir.

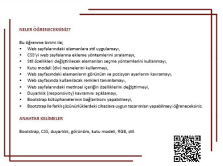
Ey Türk istikbalinin evlâdı! İşte, bu ahval ve şerait içinde dahi vazifen, Türk istiklâl ve cumhuriyetini kurtarmaktır. Muhtaç olduğun kudret, damarlarındaki asil kanda mevcuttur.

Mustafa Kemal Atatürk

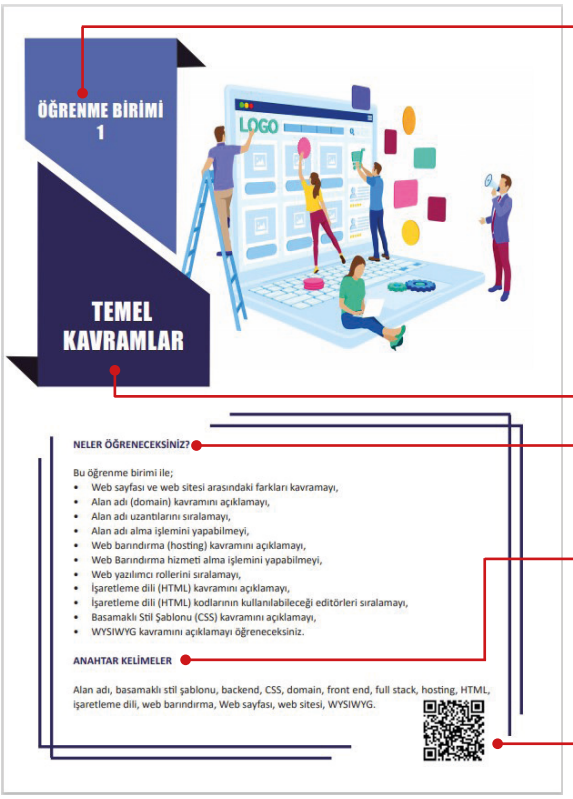


MUSTAFA KEMAL ATATÜRK

İÇİNDEKİLER



DERS MATERYALİNİN TANITIMI



Öğrenme biriminin numarasını gösterir.

Öğrenme biriminin adını gösterir.

Öğrenme biriminde öğrenilecek konuları gösterir.

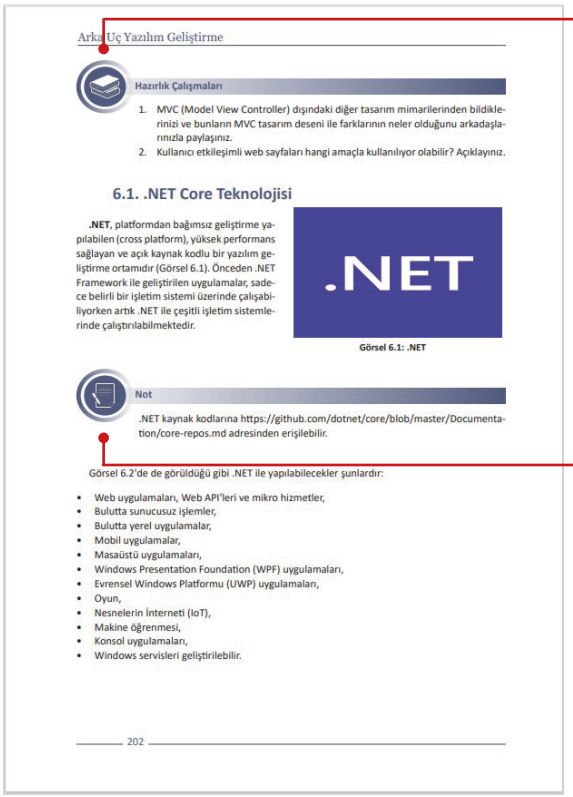
Öğrenme birimindeki önemli kavramları gösterir.

Karekod okuyucu ile taratarak resim, video, soru ve çözümleri vb. ilave kaynaklara ulaşabileceğimiz karekod bölümüdür. Detaylı bilgi için http://kitap.eba.govtr/karekod

Konu başlığını gösterir.

Konu anlatımını gösterir.

Alt konu başlığını gösterir.



Arka Uç Yazılım Geliştirme

Hazırlık Çalışmaları

1. MVC (Model View Controller) dışındaki diğer tasarım mimarilerinden bildiklerinizi ve bunların MVC tasarım deseni ile farklarının neler olduğunu arkadaşlarınızla paylaşınız.
2. Kullanıcı etkileşimli web sayfaları hangi amaçla kullanılıyor olabilir? Açıklayınız.

6.1. .NET Core Teknolojisi

.NET, platformdan bağımsız geliştirme yapılabilen (cross platform), yüksek performans sağlayan ve açık kaynak kodlu bir yazılım geliştirme ortamıdır (Görsel 6.1). Önceden .NET Framework ile geliştirilen uygulamalar, sadece belirli bir işletim sistemi üzerinde çalışabiliyorken artık .NET ile çeşitli işletim sistemlerinde çalıştırılabilmektedir.

.NET

Görsel 6.1: .NET

Not

.NET kaynak kodlarına https://github.com/dotnet/core/blob/master/Documentation/core-repos.md adresinden erişilebilir.

Görsel 6.2'de de görüldüğü gibi .NET ile yapılabilecekler şunlardır:

- Web uygulamaları, Web API'leri ve mikro hizmetler,
- Bulutta sunucusuz işlemler,
- Bulutta yerel uygulamalar,
- Mobil uygulamalar,
- Masaüstü uygulamaları,
- Windows Presentation Foundation (WPF) uygulamaları,
- Evrensel Windows Platformu (UWP) uygulamaları,
- Oyun,
- Nesnelerin İnterneti (IoT),
- Makine öğrenmesi,
- Konsol uygulamaları,
- Windows servisleri geliştirilebilir.

202

Derse başlamadan yapılacak olan hazırlıkları gösterir.

Konuyla ilgili önemli bölümleri gösterir.

Araştırılması gereken konuları gösterir.

HTML5

Araştırma

HTML belgesi oluşturmak için kullanılan ücretli ve ücretsiz editörlerin neler olduğunu araştırınız. Araştırma sonucunda elde ettiğiniz bilgileri sınıfta paylaşınız.

Metin düzenleme programı ile yazılan kodları HTML belgesi olarak kaydetmek için **Dosya** menüsünden **Farklı Kaydet** komutu tıklanır. Açılan pencereden sırasıyla **Dosya adı** ve **Kayıt türü** seçilir (Görsel 3.7).

Görsel 3.7: HTML belgesini kaydetme

HTML belgelerine isim verilirken dikkat edilmesi gereken kurallar şunlardır:

- Dosya adlarında büyük harf yerine küçük harf kullanmak ve bağlantı (köprü) oluşturmak daha kolaydır.
- Türkçe karakter (ç, ğ, ı, ö, ü) kullanılmamalıdır. Yerel (local) sunucuda Türkçe karakter kullanmak HTML belgesinin çalışmasına engel teşkil etmese de internet ortamında sunucuya yüklendiğinde dosyalar görüntülenmez. Bu durum dosyalar ile düzgün bağlantı kurulmasına engel olur.
- Sayfa içeriğine uygun isim verilmelidir. Hakkımda bilgilerinin yer aldığı HTML belgesine "dosya01.html" gibi isim vermek yerine "hakkimda.html" gibi içerikle ilgili isim vermek bağlantı linklerini yazmayı ve bağlantı kurmayı kolaylaştırır.
- Dosya adlarında "boşluk" karakteri kullanılmamalıdır. Dosya adında boşluk karakteri kullanıldığında dosya ile oluşturulan bağlantılar çalışmaz.
- Dosya isimlerinde birden fazla kelime kullanıldığında aralarında "_ (alt çizgi)" yerine "- (tire)" kullanılmalıdır. İkisi de kullanılabilir fakat arama motorları tire karakterini daha kolay yorumlar.

Dosya isimlerinin yukarıdaki kurallara dikkat edilerek oluşturulması, arama motorlarında web sitesinin doğru indekslenmesini sağlar.

3.1.2.1. DOCTYPE

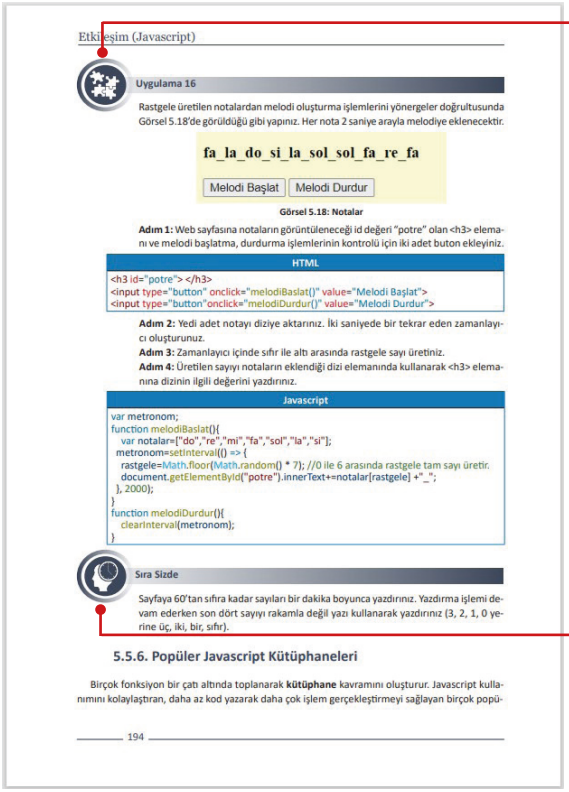
HTML5'te ilk göze çarpan yenilik DOCTYPE bildirimidir.

DOCTYPE [Belge tipi (Document type)], tarayıcıya (browser) dokümanın tipini işaret eder. DOCTYPE etiket değildir. DOCTYPE deyiminden sonra yazılan ifade ile tarayıcı, dokümanın hangi tipte olduğunu anlar. HTML belgesi için DOCTYPE ifadesinde "html" kullanılır (Görsel 3.8). Web sayfaları için <html> etiketinden önce yazılır. DOCTYPE bildirimi

<!DOCTYPE html>

Görsel 3.8: DOCTYPE kullanımı

46



Etkileşim (Javascript)

Uygulama 16

Rastgele üretilen notalardan melodi oluşturma işlemlerini yönergeler doğrultusunda Görsel 5.18'de görüldüğü gibi yapınız. Her nota 2 saniye arayla melodiye eklenecektir.

fa_la_do_si_la_sol_sol_fa_re_fa

Melodi Başlat | Melodi Durdur

Görsel 5.18: Notalar

Adım 1: Web sayfasına notaların görüntüleneceği id değeri "potre" olan <h3> elemanı ve melodi başlatma, durdurma işlemlerinin kontrolü için iki adet buton ekleyiniz.

HTML

```
<h3 id="potre"> </h3>
<input type="button" onclick="melodiBaslat()" value="Melodi Başlat">
<input type="button"onclick="melodiDurdur()" value="Melodi Durdur">
```

Adım 2: Yedi adet notayı diziye aktarınız. İki saniyede bir tekrar eden zamanlayıcı oluşturunuz.
Adım 3: Zamanlayıcı içinde sıfır ile altı arasında rastgele sayı üretiniz.
Adım 4: Üretilen sayıyı notaların eklendiği dizi elemanında kullanarak <h3> elemanına dizinin ilgili değerini yazdırınız.

Javascript

```
var metronom;
function melodiBaslat(){
  var notalar=["do","re","mi","fa","sol","la","si"];
  metronom=setInterval(() => {
   rastgele=Math.floor(Math.random() * 7); //0 ile 6 arasında rastgele tam sayı üretir.
   document.getElementById("potre").innerText+=notalar[rastgele] +"_";
  }, 2000);
}
function melodiDurdur(){
   clearInterval(metronom);
}
```

Sıra Sizde

Sayfaya 60'tan sıfıra kadar sayıları bir dakika boyunca yazdırınız. Yazdırma işlemi devam ederken son dört sayıyı rakamla değil yazı kullanarak yazdırınız (3, 2, 1, 0 yerine üç, iki, bir, sıfır).

5.5.6. Popüler Javascript Kütüphaneleri

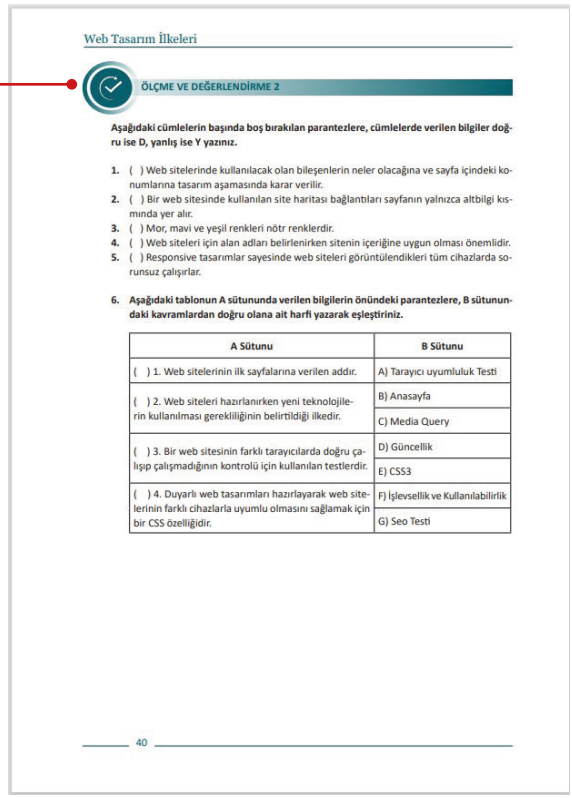
Birçok fonksiyon bir çatı altında toplanarak **kütüphane** kavramını oluşturur. Javascript kullanımını kolaylaştıran, daha az kod yazarak daha çok işlem gerçekleştirmeyi sağlayan birçok popü-

194

Öğrencilerin edindiği bilgiyi kullanmasını sağlayacak çalışmaları gösterir.

Konuyu pekiştirmek için öğrencilerin yapması gereken etkinlikleri gösterir.

Öğrenme biriminin anlaşılma derecesini gösterir.



Web Tasarım İlkeleri

ÖLÇME VE DEĞERLENDİRME 2

Aşağıdaki cümlelerin başında boş bırakılan parantezlere, cümlelerde verilen bilgiler doğru ise D, yanlış ise Y yazınız.

1. ( ) Web sitelerinde kullanılacak olan bileşenlerin neler olacağına ve sayfa içindeki konumlarına tasarım aşamasında karar verilir.
2. ( ) Bir web sitesinde kullanılan site haritası bağlantıları sayfanın yalnızca altbilgi kısmında yer alır.
3. ( ) Mor, mavi ve yeşil renkleri nötr renklerdir.
4. ( ) Web siteleri için alan adları belirlenirken sitenin içeriğine uygun olması önemlidir.
5. ( ) Responsive tasarımlar sayesinde web siteleri görüntülendikleri tüm cihazlarda sorunsuz çalışırlar.

6. **Aşağıdaki tablonun A sütununda verilen bilgilerin önündeki parantezlere, B sütunundaki kavramlardan doğru olana ait harfi yazarak eşleştiriniz.**

A Sütunu	B Sütunu
( ) 1. Web sitelerinin ilk sayfalarına verilen addır.	A) Tarayıcı uyumluluk Testi
( ) 2. Web siteleri hazırlanırken yeni teknolojilerin kullanılması gerekliliğinin belirtildiği ilkedir.	B) Anasayfa
	C) Media Query
( ) 3. Bir web sitesinin farklı tarayıcılarda doğru çalışıp çalışmadığının kontrolü için kullanılan testlerdir.	D) Güncellik
	E) CSS3
( ) 4. Duyarlı web tasarımları hazırlayarak web sitelerinin farklı cihazlarla uyumlu olmasını sağlamak için bir CSS özelliğidir.	F) İşlevsellik ve Kullanılabilirlik
	G) Seo Testi

40

1. ÖĞRENME BİRİMİ

TEMEL KAVRAMLAR

NELER ÖĞRENECEKSİNİZ?

Bu öğrenme birimi ile;

- Web sayfası ve web sitesi arasındaki farkları kavramayı,
- Alan adı (domain) kavramını açıklamayı,
- Alan adı uzantılarını sıralamayı,
- Alan adı alma işlemini yapabilmeyi,
- Web barındırma (hosting) kavramını açıklamayı,
- Web Barındırma hizmeti alma işlemini yapabilmeyi,
- Web yazılımcı rollerini sıralamayı,
- İşaretleme dili (HTML) kavramını açıklamayı,
- İşaretleme dili (HTML) kodlarının kullanılabileceği editörleri sıralamayı,
- Basamaklı Stil Şablonu (CSS) kavramını açıklamayı,
- WYSIWYG kavramını açıklamayı öğreneceksiniz.

ANAHTAR KELİMELER

Alan adı, basamaklı stil şablonu, backend, CSS, domain, front end, full stack, hosting, HTML, işaretleme dili, web barındırma, Web sayfası, web sitesi, WYSIWYG.

Hazırlık Çalışmaları

1. Bir web sitesi kurmak isteseydiniz izleyeceğiniz aşamalar neler olurdu? Düşüncelerinizi arkadaşlarınızla paylaşınız.

2. Web sitesi hazırlayacak olsaydınız hangi araçları (editörleri) kullanırdınız? Arkadaşlarınızla birlikte belirlediğiniz araçları listeleyiniz.

1.1. Web Sayfası Yayınlamada Temel Kavramlar

Web sayfasının yayınlanabilmesi için sayfa içeriğinin hazırlanmasının yanında, temel bileşenlerin de tanımlanması, bir araya getirilmesi gerekir.

1.1.1. Web Sayfası

Internet [World Wide Web (www)] için hazırlanan ve web tarayıcıları aracılığıyla görüntülenebilen belgeye **web sayfası** adı verilir. Standart bir web sayfası, farklı web sayfalarına bağlantıları içermekte olup farklı teknolojilerle hazırlanabilmektedir.

1.1.2. Web Sitesi

Web sitesi, web üzerinde yer alan sayfaları ziyaretçilerin kullanımına sunan sayfalar bütünüdür.

1.1.3. Alan Adı (Domain)

IP (Internet Protokol) adresi şeklinde ifade edilen, bilgisayarların birbirleri ile iletişim kurmasını sağlayan, numerik sisteminin daha kolaylaştırılmış ve rahatça girilebilmesi için kelimelerle ifade edilen hâlidir (Görsel 1.1). "Web sitesinin, internet dünyası içindeki kimliği" olarak da tanımlanır.

www.meb.gov.tr

www.alanismi.alanturu.ulkekodu

Görsel 1.1: Alan adı kodlama yapısı

1.1.4. Alan Adı Uzantıları

En yaygın alan adı uzantıları aşağıdaki gibidir (Görsel 1.2).

gov (government)	:	Devlet kurumları
edu (education)	:	Eğitim kurumları
k12 (kindergarten 12)	:	Temel eğitim ve ortaöğretim kurumları
org (organization)	:	Ticari olmayan kuruluşlar
com (company)	:	Ticari kuruluşlar
mil (military)	:	Askerî kurumlar

Görsel 1.2: Alan adı uzantıları

net (network)	**:**	Servis sunucular
ac (academic)	**:**	Akademik kuruluşlar
int (international)	**:**	Uluslararası kuruluşlar
info (information)	**:**	Bilgi içerikli web siteleri
biz (business)	**:**	Ticari kuruluşlar

k12 uzantısı; okul öncesi ile üniversite öncesi arasında hizmet veren eğitim kurumları (okul öncesi, temel eğitim ve ortaöğretim kurumları) tarafından kullanılmaktadır. edu uzantısı ise; ağırlıklı olarak üniversiteler tarafından kullanılmaktadır.

Sıra Sizde

Yaygın olarak kullanılan alan adı uzantılarından istediğiniz beş tanesini araştırarak özelliklerini inceleyiniz.

1.1.5. Alan Adı Alma İşlemi

Web sitesi için belirlenecek isim (alan adı), kullanıcıların kolay hatırlayacağı ve sitenin yayın amacına uygun olacak şekilde belirlenmelidir. Aşağıdaki adımlar takip edilerek alan adı alma işlemi tamamlanabilir.

• Güvenilir bir alan adı yetkilisi (örneğin; nic.tr) seçilir.

• Alan adı sorgulama aracı üzerinden istenen alan adının durumu sorgulanır.

• Sorgulanan alan adı daha önce kullanılmamış ve müsait durumda ise doğrudan seçim yapılarak kayıt süreci başlatılır. Eğer sorgulanan alan adının kullanımda olduğu (yani daha önceden kaydedildiği) görülürse benzer şekilde kullanılabilecek en iyi seçeneklerden biri tercih edilebilir.

• Kayıt sürecinin sonunda siparişe ilişkin tanımlamalar (alan adı için kullanım süresi, satın alma gerçekleştirilecekse ödeme tanımlamaları) yapılır.

• Alınan alan adı onaylanarak işlem tamamlanır.

Not

Ücretli yapılacak iş ve işlemler; büyüklerinizin (öğretmeninizin veya velilerinizin) gözetiminde yapılmalıdır.

Sıra Sizde

Belirlediğiniz bir alan adı için müsaitlik durumunu araştırarak kayıt işlemine ilişkin adımları uygulayınız.

1.1.6. Hosting (Web Sitesi Barındırma)

Hazırlanan web sitelerinin belirlenen alan adlarına göre internet ortamında yayınlanmasını sağlayan hizmet türüne **hosting** denir. Web sitesine ait içeriğin tutulduğu (barındırıldığı) alanla ilgi-

li hizmet desteğini kapsar. Bu hizmet, hosting firmaları tarafından belirli süreliğine ve farklı paket özellikleriyle sağlanır (Görsel 1.3).

Hosting paketleri; web sitesinin yer alacağı fiziksel sunucu için bellek, işlemci ve disk alanı gibi kaynakların boyutuna, hızına ve sağlanacak diğer desteklere göre değişkenlik gösterir. Bu durum maliyete doğrudan etki eder.

Server (sunucu) bilgisayarlar, hosting amacıyla kullanılan gelişmiş donanım özelliklerine sahip olan ve birçok kullanıcıya aynı anda hizmet veren bilgisayarlardır.

Web sitesi barındırma hizmeti; hizmet sağlayıcı tarafından veri merkezi ve sunucu desteği ile birlikte sunulan alan ve bağlantı hizmetinin yanı sıra e-posta barındırma hizmeti, gelişmiş güvenlik özellikleri, farklı diğer özellikleri de içerir.

Görsel 1.3: Hosting (web sitesi barındırma) anlatımı

1. Uygulama

Alan adının aktif şekilde çalıştırılması amacıyla web barındırma hizmeti alarak alan adı ile eşleştirilmesi için gerekli işlemleri, yönergeler doğrultusunda gerçekleştiriniz.

1. Adım: Güvenilir bir hosting satış (hizmet) yetkilisi seçiniz.
2. Adım: Web barındırma hizmeti ile ilgili bölümde sunulan paketler arasından ihtiyaca uygun bir hosting paketi seçiniz.
3 .Adım: Hosting hizmetiyle ilişkilendirilecek (eşleştirilecek) alan adını tanımlayınız veya gerekli DNS tanımlamalarını yapınız. Domain ve hosting, aynı hizmet yetkilisi firmadan sağlanacaksa doğrudan alan adı üzerinden eşleşme yapılabilir. İki hizmet ayrı firmalardan sağlanacaksa alan adı için kullanılan yönetim panelinden kullanılacak hostinge ait **ad sunucusu (name server)** tanımlamasını yapınız.
4. Adım: Hosting kayıt işleminin son adımına geçilerek siparişe ilişkin tanımlamaları (hosting hizmet süresi, satın alma gerçekleştirilecekse ödeme tanımlamaları) yapınız ve işlemi tamamlayınız.

1.2. Web Yazılımcısı Rolleri

Web sitesinin kullanıcılar için etkin bir yapıda olması görünen yüzü oluşturan "tasarım" ve bu yüzün arka planını oluşturan "geliştirme" işlevlerinin bütünleştirilmesi ile sağlanır. Web yazılımcı rolleri, tasarım ve geliştirme süreçlerinin istenen şekilde yürütülmesi için önemlidir.

1.2.1. Ön Uç (Frontend) Yazılımcı Rolü

Görsel 1.4: Ön uç yazılımcı

Ön uç, web sitesinin ziyaretçi tarafından görülen ve üzerinden etkileşime geçilen arayüz kısmına verilen addır. **Ön uç yazılımcı**, bu arayüzün geliştirilmesini sağlar (Görsel 1.4).

Ön uç geliştirme süreci içinde, ağırlıklı olarak web arayüzünün kullanıcıya yansıyan kısmına odaklanan HTML, CSS ve JavaScript dilleri kullanılır.

1.2.2. Arka Uç (Backend) Yazılımcı Rolü

Görsel 1.5 Arka uç yazılımcı

Arka uç web sitesinin ziyaretçi tarafından görülmeyen sunucu etkileşimi, ön uç veri alışverişi (veri tabanı ile kayıt / güncelleme / silme etkileşimi) gibi işlemlerin gerçekleştirildiği; sitenin sağlıklı çalışmasını sağlayan arka plan kısmına verilen addır. Arka uç yazılımcı, bu arayüzün geliştirilmesini sağlar (Görsel 1.5). Örneğin kullanıcı, internet üzerinden alışveriş yapılan bir web sitesine girerek alışveriş sürecini yürütürken ön uç ile etkileşime girmiş olur. Kullanıcı tarafından, arayüz üzerinden web sitesine girilen bilgiler, arka uç işleyiş içinde alınarak sunucu üzerinde bulunan veri tabanına kaydedilir. Tüm bu işleyişin sağlıklı bir şekilde gerçekleştirilmesi arka uç yazılımcı tarafından sağlanır.

Steve Jobs'un "Tasarım, bir şeyin yalnızca nasıl göründüğü ve nasıl hissettirdiği ile ilgili değildir. Tasarım, bir şeyin nasıl çalıştığıyla da alakalıdır." sözü arka uç yazılımcıların rollerinin önemine güzel bir örnektir.

İşin mutfağı olarak tanımlanabilecek arka uç (backend) programlama dilleri arasında; Java, PHP, ASP.NET, Python, Node.JS öne çıkmaktadır.

Java: Dünyadaki en popüler programlama dilleri arasında gösterilen; mobil ve masaüstü uygulamalarda yaygın olarak kullanılan, büyük miktarlarda kodların kolay okunduğu, farklı kütüphanelere sahip bir dildir.

PHP: Sunucu taraflı çalışan, dinamik içerik açık kaynak kodlu ve web barındırma hizmeti maliyetleri düşük, geniş kullanım ağına sahip bir programlama dilidir.

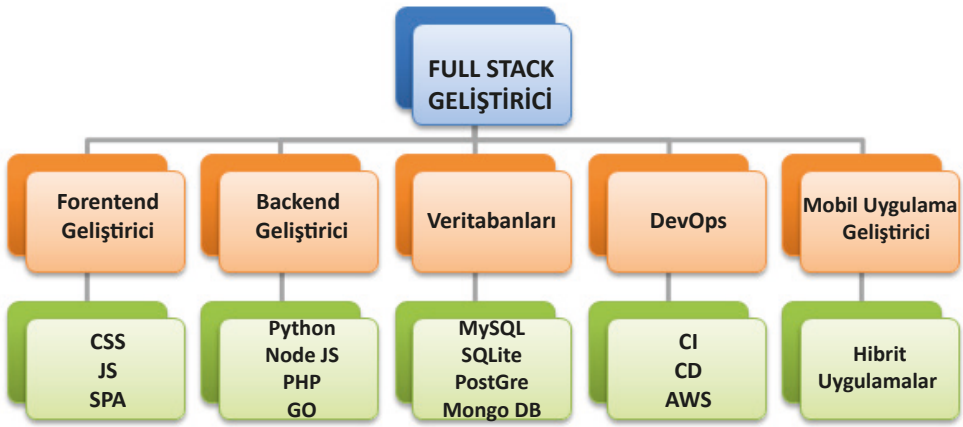
ASP.NET: Temelde C# programlama dilini kullanan güçlü bir çatıdır. Özellikle **katmanlı mimari** olarak da adlandırılan MVC (Model-View-Controller) mimari yaklaşımı sayesinde, arka uç (backend) ön uç (frontend) etkileşimini kolaylaştırır.

Python: "Makine Öğrenmesi" alanında öne çıkan, kod yazma düzeninin basit ve kolay anlaşılabilir olması sebebiyle kullanımı yaygınlaşan, popüler programlama dillerindendir.

Node.JS: JavaScript kodlarını sunucu taraflı olarak çalıştıran bir web programlama dilidir. Giriş/Çıkış (I/O) ve ağ işlemlerini bekleme yapmaksızın (asenkron olarak) çalıştıran Node.js, zaman ve kaynak kullanımı alanlarında öne çıkar.

1.2.3. Full Stack Yazılımcı Rolü

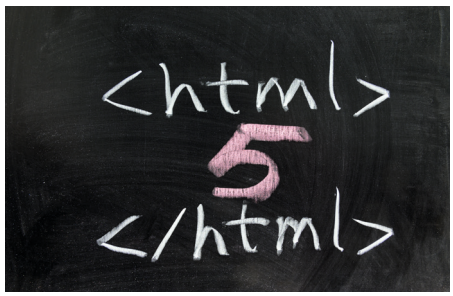
Full stack yazılımcılar, web uygulamaları veya yazılımı geliştirirken; bu yazılımın arka uç ve ön uç kısımlarını geliştirme kabiliyetine sahip kişilerdir (Şema 1.1).



Şema 1.1: Full stack yazılımcı rolü

1.3. İşaretleme Dili (HTML)

İşaretleme dili, metinler için yapılandırma veya biçimlendirme açıklamalarını içeren "etiket" (tag) adı verilen elemanlarla tanımlanan yapay bir dildir. Bilişim alanındaki en bilinen örneği HTML olan işaretleme dilinin diğer örnekleri arasında TeX, LaTeX, XML, SGML, JSON yer alır (Görsel 1.6).



Görsel 1.6: HTML etiketi

Web'in en temel yapı taşı olarak kabul edilen **HTML (HyperText Markup Language)**, web sayfaları hazırlamak için kullanılan bir işaretleme dilidir.

HyperText (bağlantı metni), web sayfalarını tek bir web sitesi içinde veya web siteleri arasında birbirine bağlayan bağlantıları işaret eder.

HTML; bir web tarayıcısında görüntülenmek üzere tanımlanan metin, resim ve diğer içerikler için "işaretleme"yi (markup) kullanmaktadır.

Tasarımcılara sayfalar ve uygulamalar için yapı profilleri, bağlantılar, blok alıntılar, paragraflar ve başlıklar oluşturmada kolaylık sağlayan HTML için **web sitesinin iskeleti** denebilir.

• HTML komutları, herhangi bir metin düzenleme editöründe yazılabileceği gibi çeşitli web tasarım editörleriyle de oluşturulabilir.

• HTML komutları etiketlerden (tag) oluşur.

• HTML etiketleri yazılırken Türkçe karakterler (ç, ğ, ı, ö, ş, ü) kullanılmamalıdır.

• İstisnai durumlar dışında, açılan bir etiket kapatılmalıdır. İlk açılan etiket en son kapatılır ve etiketi kapatma sırasında **"/" (eğik çizgi)** işareti unutulmamalıdır.

1.3.1. HTML Editörleri

HTML kodları, Not Defteri uygulaması dâhil, herhangi bir metin editöründe yazılabilir. Klasik metin düzenleme editörleri, kodlama sırasında görsel / yapısal düzenleme araçları sunmamaktadır. Bu da kod yazımını zorlaştırır.

HTML editörleri, kod yazarken söz dizimlerinin vurgulanmasını, sık kullanılan HTML yapılarının kodlama içine kolayca yerleştirilmesini sağlayan aynı zamanda otomatik tamamlama desteği de sunan düzenleme araçlarıdır.

Sıra Sizde

Web sayfası hazırlamada kullanılan HTML editörlerini araştırarak konuyla ilgili bir sunum hazırlayınız ve sınıfınızda paylaşınız.

1.4. CSS [Cascaded Style Sheet(Basamaklı Stil Şablonu)]

Görsel 1.7: CSS kodlama örneği

Basamaklı Stil Şablonu (CSS), kendine özgü kuralları ile web sitelerinin görsel olarak biçimlendirilmesini sağlayan bir tanımlama dilidir. Web siteleri üzerinde geniş bir görsel denetim imkânı sunan CSS, HTML ve JavaScript ile birlikte temel web teknolojileri arasında yer alır (Görsel 1.7).

CSS ile web sayfalarının yerleşim ve renk düzeninin yanı sıra yazı tipleri, başlık biçimleri ve diğer görsel unsurlarıyla ilgili etkili, fonksiyonel kontroller sağlanabilir. Etkili bir CSS kodlaması, web sitelerinin kullanılabilirliğine de doğrudan etki eder.

1.5. WYSIWYG [What You See Is What You Get (Editörler)]

WYSIWYG (What You See Is What You Get), "Ne görüyorsan onu alırsın." demektir. WYSIWYG editörler, kullanıcının sayfa oluştururken sonuçta göreceğine çok benzer bir ekranı görmesini sağlayan bir arayüzden oluşur. WYSIWYG, kod yazmak zorunda kalmadan bir sayfanın hazırlanmasına olanak verir.

WYSIWYG editörlerinin kullanımı hiç HTML bilgisi gerektirmez, hiçbir kodlama tecrübesi olmayan kullanıcıların, web sayfası kodlama süreçlerine uyumunu kolaylaştırır.

Sıra Sizde

Oluşturacağınız küçük gruplarla yaygın olarak kullanılan WYSIWYG editörlerini araştırıp avantaj ve dezavantajlarını anlatan bir afiş hazırlayınız. Hazırladığınız afişleri arkadaşlarınızla paylaştıktan sonra sınıf panosunda sergileyiniz

ÖLÇME VE DEĞERLENDİRME

A. Aşağıdaki tabloda A sütununda bilgiler, B sütununda kavramlar verilmiştir.

1. Buna göre A sütununda verilen bilgilerin önündeki parantezlere, B sütunundaki kavramlardan doğru olana ait harfi yazarak eşleştiriniz.

A Sütunu	B Sütunu
( ) . com	A) Askerî kurumlar
( ) . mil	B) Ticari kuruluşlar
( ) . edu	C) Uluslararası kuruluşlar
( ) . int	D) Eğitim kurumları
( ) . gov	E) Devlet kurumları

B. Aşağıdaki cümlelerde boş bırakılan yerlere doğru sözcükleri yazınız.

2. Web sayfalarının yerleşim ve renk düzeninin yanı sıra yazı tipleri, başlık biçimleri ve diğer görsel unsurlarıyla ilgili etkili ve fonksiyonel kontroller …….. ile sağlanabilir.

3. Web sitesinin ziyaretçi tarafından görülmeyen, sunucu etkileşiminin ve veri alışverişi gibi işlemlerin gerçekleştirildiği, sitenin sağlıklı çalışmasını sağlayan kısmına …….. adı verilir.

C. Aşağıdaki soruları dikkatlice okuyunuz ve doğru seçeneği işaretleyiniz.

4. "www.meb.gov.tr" alan adı tanımlamasında yer alan "meb" ifadesi; alan adı kodlama yapısının hangi parçasına karşılık gelmektedir?

A) Alan Türü B) Alan İsmi C) Ülke Kodu D) Sunucu Türü E) Sunucu Adı

5. Aşağıdakilerden hangisi ön uç geliştirme sürecinde ağırlıklı olarak kullanılan teknolojilerdendir?

A) CSS B) C C) PHP D) Python E) asp

2.
ÖĞRENME BİRİMİ

WEB TASARIM İLKELERİ

NELER ÖĞRENECEKSİNİZ?

Bu öğrenme birimi ile;

- Web tasarım ilkelerinin neler olduğunu açıklamayı,
- Web tasarımı sırasında içerik ve tasarım ilişkisini kurmayı,
- Web sitesinde bulunan bileşenlerin sayfadaki konumunu belirlemeyi,
- Dijital ortamda ve el ile web sayfası taslağı oluşturmayı,
- Web tasarım ilkelerine ve tipografik kurallara uygun web sayfası taslağı hazırlamayı öğreneceksiniz.

ANAHTAR KELİMELER

Biçimsellik, güncellik, içerik, işlevsellik, kullanılabilirlik, media query, mobil uyumluluk, responsive, SEO, tarayıcı uyumluluğu, tasarım, tipografi.

Hazırlık Çalışmaları

1. Aynı özelliklere sahip iki işletme tanıtım amaçlı Web sitesi tasarımı yaparak yayınlamışlardır. Web sitelerinden birisinin ziyaretçileri oldukça fazla diğerinin ki ise çok azdır. Bu durum hangi nedenlerden kaynaklanmış olabilir?

2. Bir web sitesini ziyaret ettiğinizde gözünüze ilk çarpan olumlu ve olumsuz özellikler nelerdir?

2.1. İçerik

Kurum, işletme veya bireyleri internet ortamında temsil eden web siteleri hazırlanırken dikkat edilmesi gereken birtakım ilkeler vardır (Görsel 2.1). Tasarım sürecinde bu ilkelerin göz önünde bulundurulması, hazırlanacak olan web sitesinin kalitesi, profesyonelliği ve site ziyaretçileri tarafından kullanışlılığı açısından oldukça önemlidir.

Görsel 2.1: Web tasarım ilkeleri

Web tasarım ilkelerinin başında, web sitesinin temeli olan içerik oluşturma gelmektedir (Görsel 2.2). Web sitesinin en önemli amacı içeriği tüm dünyadaki ziyaretçiler ile paylaşmak olduğundan, tasarıma başlandığı andan itibaren yapılması gereken ilk iş içerik planlaması olmalıdır. Web sitesi içeriği (web site content); görsel, metinsel ve işitsel materyallerden oluşur. İçeriklerin birbirine uygun grafik ve metinler ile harmanlanması sitenin kalitesi açısından önemlidir.

Görsel 2.2: İçerik (Content)

İçerikler hedef kitleye hitap edecek şekilde hazırlanmalı, gereğinden fazla öge kullanılmamasına dikkat edilmelidir. Web sitesi oluşturulurken kullanılan dil de çok önemlidir. İçerikteki yazılar anlaşılır; net ve doğal bir dil kullanılarak oluşturulmalıdır.

İçerik; özgün, güncel, bilgilendirici, dikkat çekici ve SEO [Search Engine Optimization (Arama Motoru Optimizasyonu)] uyumlu olmalıdır. İçeriğin konu ile ilgili anahtar kelimeler barındırması gerekir.

İçerik hazırlama esnasında kullanılan metin, resim ve videolar gruplara ayrılmalıdır. Metin, resim ve videoların içerik hazırlanırken gruplandırılması karmaşayı önler.

İçerikte kullanılacak metinler kadar tercih edilen görseller de önemlidir. Metin ve grafiklerdeki renkler ile fontlar (yazı tipleri) dikkatli seçilmeli ve içerik sade bir şekilde tasarlanmalıdır. Seçilen yazı tipi tüm dijital cihazlarda ve farklı tarayıcılarda doğru çalışmalı ve kullanıcı dostu olmalıdır.

Seçilen grafikler ve videolar sayfanın yüklenme hızını olumsuz etkilemeyecek formatlarda olmalıdır.

Not

Arama Motoru Optimizasyonu (SEO), bir sitenin arama motorlarında daha iyi sıralamalar elde etmesi, daha iyi performans göstererek daha fazla nitelikli ziyaretçiye ulaşabilmesi amacıyla yapılan çalışmalardır.

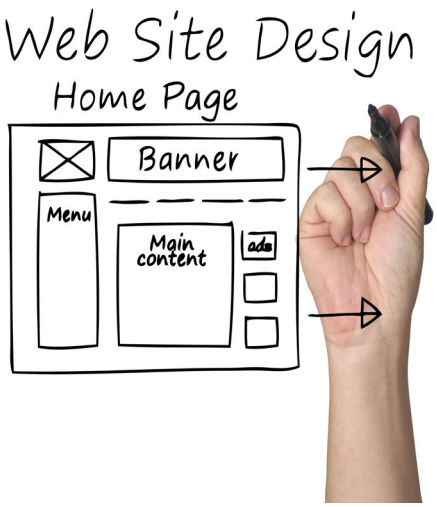
2.2. Tasarım

Web tasarım ilkelerinden biri olan tasarım (design), web sitesindeki içeriklerin yerleşim planlamasının yapıldığı (içeriğin sayfanın neresinde ve nasıl konumlandığı) kısımdır (Görsel 2.3).

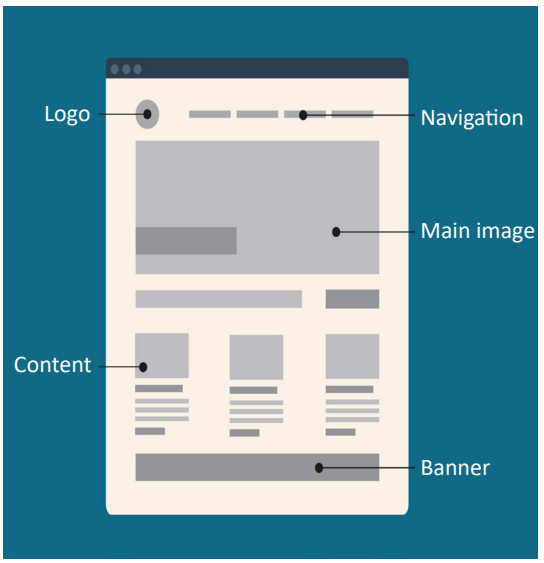
Görsel 2.3: Site yerleşim planı

Site yerleşim planı, siteyi ziyaret eden kişilerin güvenini kazanmak için önemlidir. Gelişigüzel yapılmış bir yerleşim planı ziyaretçilerin siteyi bir daha ziyaret etmemelerine sebep olabilir. Bu nedenle sitede kullanılacak bileşenlerin konumlarının doğru planlanmasına dikkat etmek gerekir.

Tasarım aşamasında görünümün nasıl olacağı, içeriğin nerede gösterileceği, logo, görseller ve menülerin konumu, renk ve tipografi düzeninin nasıl olacağı, başlık ve taban görünümü gibi düzenlemelerin tümüne karar verilir. Web sitesi tasarımına başlamadan önce hazırlanmak istenen sitenin taslak çalışması (site bileşenlerinin sayfadaki konumu) kâğıt üzerinde el ile (Görsel 2.4) veya dijital ortamda (Görsel 2.5) hazırlanır. Böylece hazırlanmış arayüz sayesinde hem sitenin taslağı oluşmuş hem de eklenmemiş bir bileşenden dolayı yaşanacak zaman kaybının önüne geçilmiş olur.

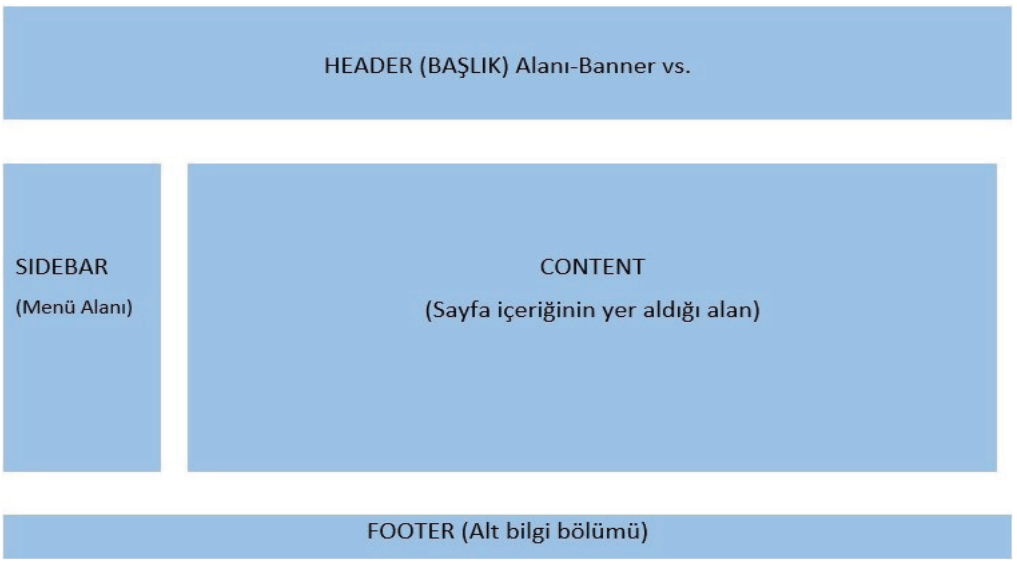


Görsel 2.4: El ile hazırlanmış bir web sayfası taslak çalışması



Görsel 2.5. Dijital ortamda hazırlanmış bir web sayfası taslak çalışması

Web sitesi tasarımı sırasında sitede yer alan bileşenler tasarımcının planlamasına ve bakış açısına göre farklılıklar gösterebilir ancak temel özelliklerin dışına fazla çıkılmaması kullanıcılar açısından tutarlılık ve kullanım rahatlığı sağlayacaktır. Sitede yer alan bileşenlerin (menüler, logo, içerik, reklam vs.) konumu ziyaretçilerin alışkanlıkları doğrultusunda olmalıdır (Görsel 2.6, Görsel 2.7).



Görsel 2.6: Anasayfa tasarım örneği

Not

Banner'lar, üzerinde yazı ve grafiklerin bulunduğu, reklam ve tanıtım amacı ile kullanılan görsel bileşenlerdir. Sayfanın reklam içerik kısmında yer alabileceği gibi, web sitelerinde logo ile birlikte tasarlanıp sayfanın başlık kısmında da yer alabilirler.

Görsel 2.7: Anasayfa tasarım örneği

Web sitesi tasarımında kullanılan en önemli bileşenlerden biri menülerdir. Anasayfa (web sitesi yüklendiğinde karşımıza çıkan ilk sayfa) dışında yer alan sayfalara köprüler (link) aracılığı ile erişebilmekteyiz. Menüler kullanıcıların aradıkları bilgilere erişebilmek için kullanılan bağlantıların gruplandırıldığı alanlardır. Menüler web sayfasının alt bilgi bölümü (footer), üst bilgi bölümü (header) veya sağ/sol yanında (sidebar) konumlandırılabilir. En yaygın kullanımı sol yan ve üst bölümde yer alan tasarımdır. Web sitelerinin üstbilgi kısmında genellikle menüler, banner, site başlığı, sosyal medya bağlantı simgeleri ve logo gibi bileşenler bulunur. **Menü** kısmında sitede kullanılan menüler, **content** kısmında sitenin içerik alanı, **alt bilgi** kısmında ise iletişim adresleri, telif bilgileri, iletişim ve site haritası bağlantıları gibi seçenekler yer alır.

Not

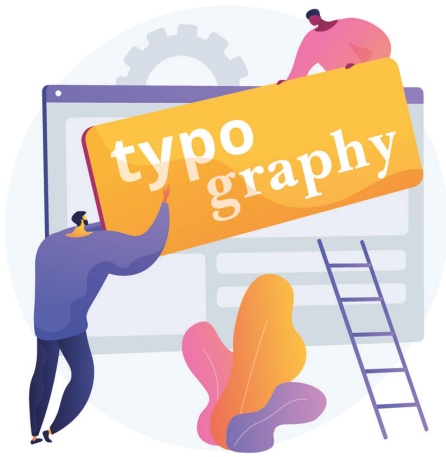
İletişim bölümü sitenin alt bölümünde olabileceği gibi sitenin üst kısmında veya menüler arasında da yer alabilmektedir. Önemli olan bu bölümün sitede mutlaka olması ve rahatça erişilebilecek bir konumda yer almasıdır.

2.3. Biçimsellik

Web tasarımı yaparken; kullanılan renk düzeni, okunabilirlik, içerik ve tasarım ilişkisini doğru kurmak gerekir.

2.3.1. Tipografi

Web sitesi içeriğinde yer alan bileşenlerin birbirleri ile uyumlu olmasını ve web sitesinin düzenli görünmesini sağlayan tasarımın en önemli unsurlarındandır. Doğru renk, font ve bileşen seçimi ile çok ilgi çekici bir web sitesi hazırlanabileceği gibi kötü bir tipografi de ziyaretçilerin ilgisinin azalmasına sebep olabilir. Tipografi, iyi hazırlanmış bir içeriği doğru şekilde sunmak için gereklidir. Web sitesindeki tüm bileşenlerin birbiri ile uyum içinde olması ve kaliteli bir tipografiye sahip olması oldukça önemlidir. Tipografi, tasarımı destekleyen bir unsurdur (Görsel 2.8).



Görsel 2.8: Tipografi

2.3.2. İçerik-Tasarım İlişkisi

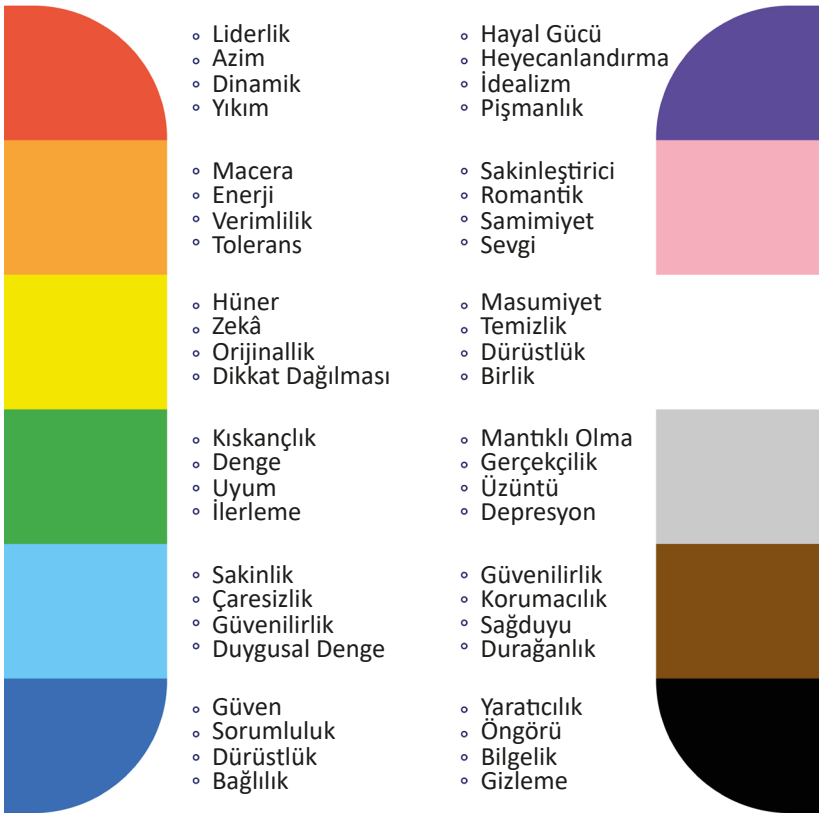
Web sayfası içeriğe uygun tasarlanmalıdır. Başta sitenin logosu olmak üzere sitenin başlığı, kullanılacak görseller, renkler ve diğer tüm bileşenler ihtiyaca uygun, birbirleri ile uyumlu, sade, dikkat çekici ve içeriği doğru yansıtacak şekilde kullanılmalıdır. İçerikle alakası olmayan başlıklar, bağlantılar, reklam alanları, görsel veya hareketli nesnelere yer verilmemelidir. Uzun bir içerik basit bir görselle ifade edilebiliyorsa tercih görselden yana yapılmalı, konu görselle anlatılmalıdır.

2.3.3. Renk Düzeni ve Okunabilirlik

Sitedeki tasarım ve içerik ilişkisi açısından önemli bir nokta da sitede kullanılacak renklerin ve okunabilirliğin doğru belirlenmesidir. Sitede kullanılan logo, zemin, yazı ve diğer bileşenlerin renklerinin birbirleri ve içerik ile uyumlu, düzenli olması gerekir. Renk seçimi isteğe bağlı olmakla birlikte içeriği yansıtacak ve kullanıcıyı rahatsız etmeyecek tonlarda olmalıdır.

Renkler insanda farklı anlamlar ve psikolojik etkiler uyandırabilmektedir (Görsel 2.9). **Sıcak renkler** kırmızı, turuncu ve sarı gibi renklerdir. **Soğuk renkler** ise mor, mavi ve yeşil tonlarından oluşan renklerdir (Görsel 2.10). Daha fazla dikkat çeken sıcak renkler yakınlık, neşe gibi duygular uyandırırken; soğuk renkler uzaklık duygusu uyandırır. Bu nedenle renk seçimi bilinçli bir şekilde yapılmalıdır.

Renklerin Psikolojisi

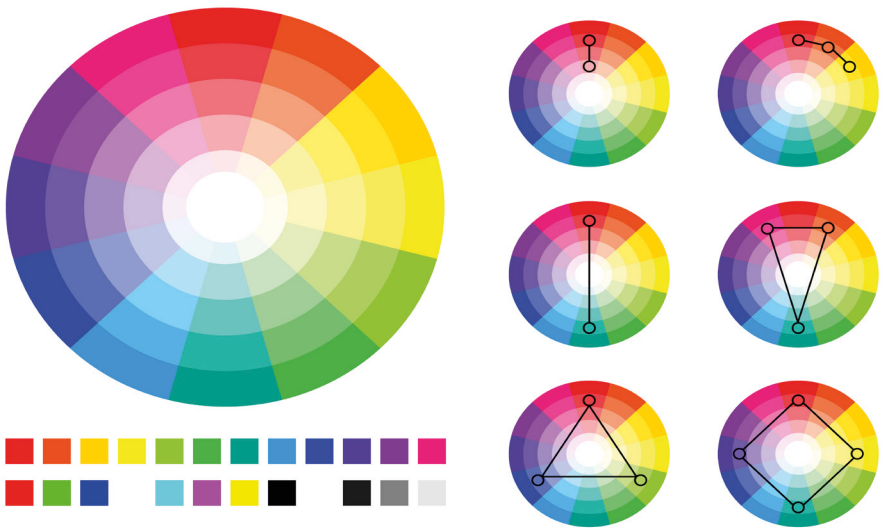


Görsel 2.9: Renklerin psikolojik etkileri

Görsel 2.10: Sıcak ve soğuk renkler

Bir nesneyi diğerinden ayırt edebilmek için kontrast (karşıtlık,zıtlık) önemlidir. Seçilen yazıların okunaklı olması için metin ve zemin renkleri arasında kontrast olması gerekir. Renk seçimi yapılırken birbirleri ile zıt renklerin seçilmesi (koyu renk zemin açık renk yazı veya açık renk zemin koyu renk yazı) okunabilirliği kolaylaştıracaktır. Arka plan renklerinin nötr renklerden (siyah, beyaz, gri) seçilmesi tercih edilmelidir.

Renk seçimi yaparken kullanılacak farklı yöntemler vardır. Bazen sadece doğaya bakmak da renk seçimi konusunda yardımcı olabilir. Renk şemalarına bakarak da renk uyumları yakalanabilir (Görsel 2.11). Aynı rengin farklı tonlarını kullanmak, birbiri ile uyumlu renkleri kullanmak bunlardan sadece ikisidir. Renk çemberinde bir rengin tam karşısındaki renk onu tamamlayıcı renktir. Renk şeması içinde bir üçgen oluşturarak üç farklı renkle kombinasyon yapılabilir. Birbirine paralel iki veya üç renk seçilerek de uyum yakalanabilir. Eşkenar üçgen renk kombinasyonunu kullanarak bir ana, iki tamamlayıcı renk seçilebilir. Bir ana, iki tamamlayıcı ve bir vurgulayıcı olmak üzere dört farklı renkle de uyum yakalanabilir.



Görsel 2.11: Renk kombinasyonları

Not

Web sayfaları tasarlanırken renk dengesini sağlamada yardım alınabilecek internet siteleri arama motorlarından kolaylıkla bulunabilir.

Sıra Sizde

Arama motoruna color scheme (renk şeması) yazarak arama yapınız. Bulduğunuz sonuçlardaki siteler arasından tasarlayacağınız site için uygun olabilecek renk kombinasyonlarını inceleyiniz.

Sitenin metinsel içeriğini oluştururken yazıyı düz bir şekilde yazmak, ziyaretçilerin yazıyı okurken sıkılmasına sebep olabilir. Farklı fontlar, renkler, yazı büyüklüğü ve kontrast sayesinde sitenin daha dikkat çekici olması sağlanabilir.

Bazı yazı tipleri diğer yazı tiplerine göre daha fazla okunabilirlik sağlar. **Sans Serif (tırnaksız)** yazı tipleri (Görsel 2.13), **Serif (tırnaklı)** yazı tiplerine (Görsel 2.12) göre dijital ekranlardan daha kolay okunmaktadır. Bu nedenle tırnaklı yazı tiplerinin kullanımından kaçınılmalıdır.

Görsel 2.12: Serif yazı tipi örnekleri

Görsel 2.13: Sans Serif yazı tipi örnekleri

Font boyutunun doğru belirlenmesi tipolojide önemli bir faktördür. Tek bir font boyutu yerine iki veya üç farklı font boyutu kullanılmalıdır. 12x-14x büyüklüğündeki fontlar tercih edilmelidir. Başlık, alt başlıklar ve metinlerde kullanılacak yazı tipi ve yazı tipi boyutları, belli bir hiyerarşik düzende olmalıdır.

Satır uzunluğu, satır ve paragraf arası boşluklar da düzenli ve uyumlu olmalıdır. Satırlar veya harfler arasındaki boşlukların gereğinden az veya fazla olmamasına özen gösterilmelidir. Yazılar sola veya iki yana yaslı olarak yazılmalı, sadece başlıklarda ortaya hizalı yazılar tercih edilmelidir.

Tipografide doğru font seçimi, metinlerdeki boşluklar, büyük ve küçük harflerin yerinde ve doğru kullanılması da oldukça önemlidir. Font seçimi yapılırken kullanılacak fontların metnin vereceği mesajın önüne geçmemesine dikkat edilmelidir (Görsel 2.14).

Görsel 2.14: Metinlerde okunabilirlik

Hizalamada sola ya da her iki yana yaslama tercih edilebilir. İçeriğin uzun olduğu durumlarda ortalı hizalama kullanılmamasına dikkat edilmelidir.

Araştırma

Arama motoruna "tipografik web siteleri" yazarak çeşitli örnekleri inceleyip, beğendiğiniz iki tanesini sınıfta arkadaşlarınızla paylaşınız.

1. Uygulama

Görsel 2.15 ’de tablo şeklinde verilen okul web sitesi ana sayfa örneğini yönergeler doğrultusunda düzenleyiniz.

<table>
<tr><td>ATATÜRK RESMİ VE TÜRK BAYRAĞI</td><td>OKUL ADI</td><td>MEB LOGOSU</td></tr>
<tr><td>Ana sayfa</td><td colspan="2" rowspan="5">İLETİŞİM BİLGİLERİ
.....
.....
.....
.....
.....</td></tr>
<tr><td>Okulumuz</td></tr>
<tr><td>Kadromuz</td></tr>
<tr><td>Galeri</td></tr>
<tr><td>İletişim</td></tr>
</table>

Görsel 2.15: Okul web sitesi ana sayfa örneği

1. Adım: Görsel 2.15’ deki tabloyu kelime işlemci programını kullanarak hazırlayınız.
2. Adım: İnternetten indirdiğiniz Atatürk resmi ve Türk Bayrağı fotoğrafını istenilen alana ekleyiniz.
3. Adım: Millî Eğitim Bakanlığı logosunu bilgisayarınıza indirip ilgili alana ekleyiniz.
4. Adım: Sayfanın zemin ve yazı rengini tipografi kurallarına uygun olarak seçip sayfada uygulayınız.
5. Adım: Bilgisayarınızda “Okul Sitem” adında bir klasör oluşturunuz.
6. Adım: Sayfayı “index” ismi ile “web sayfası” kayıt türünü seçerek klasöre kaydediniz.
7. Adım: Kaydetmiş olduğunuz “index” dosyasının üzerine farenin sol tuşu ile çift tıklayarak sayfanın tarayıcıda görüntülenmesini sağlayınız.

2.4. İşlevsellik ve Kullanılabilirlik

İçerik ve tasarımdan sonra gelen en önemli web tasarım ilkesi işlevsellik ve kullanılabilirliktir. İşlevsellik ilkesi ile web sitesindeki sayfalara erişim kolaylığı sağlanırken, kullanılabilirlik sayesinde de sitenin kullanıcı dostu olması sağlanır.

Ziyaretçilerin rahat ve kaliteli zaman geçirebilmeleri ve siteyi kolay kullanabilmeleri amacıyla hazırlanan kullanıcı dostu sitelerin tasarımı sırasında dikkat edilmesi gereken hususlar şunlardır:

- Kullanıcıların ihtiyaçları ön planda tutulmalı,
- Sayfalar yüklenirken hız sorunu olmamalı,
- Yazılar rahat okunmalı ve doğru yazı tipleri kullanılmalı,
- Kullanılan görseller ve videolar yeterli çözünürlükte ve özgün olmalı,
- İçeriğin fazla olduğu sitelere site içi arama motoru eklenmeli,
- Site haritası bulundurulmalı,
- İletişim bölümüne yer verilmeli,
- Farklı ekran boyutlarında ve tarayıcılarda sorunsuz çalışacak şekilde tasarlanmalı,
- Mobil uyumlu olmalıdır.

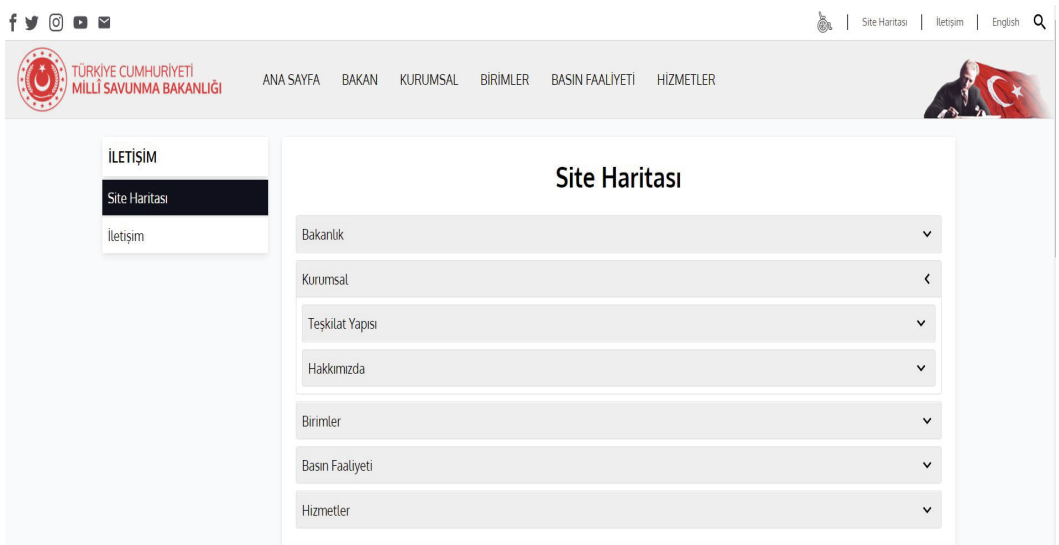
Web sitesi tasarımı yapılırken, özel gereksinimli bireyler başta olmak üzere yaşlı ve çocuklar ile yabancılar da düşünülerek ortaya çıkmış olan Evrensel Tasarım İlkeleri de gözetilmelidir.

Araştırma

Evrensel tasarım ilkelerinin neler olduğunu araştırınız. Web sitesi tasarımı sırasında bu ilkelerden nasıl yararlanılabileceğini araştırarak sınıfta arkadaşlarınızla paylaşınız.

Web sitesinin kolay yüklenmesi, kullanıcının siteye erişim sağlandıktan sonra da sayfalar arasında rahatlıkla dolaşabilmesi, bu sırada sayfalar arası kopukluk yaşanmaması, aradığı içeriğe rahat ulaşabilmesi sitenin işlevsel bir site olduğunu gösteren hususlardan bazılarıdır.

Hazırlanacak olan web sitesinde kullanıcı, sitenin anasayfasından iç sayfalara, sitenin iç sayfalarından da anasayfa veya diğer sayfalara rahatça erişebilmelidir. Bazı sitelerde bir sayfadan diğer sayfalara erişmek için çok fazla tıklama olabilmekte bu da işlevsellik açısından sorun yaratmaktadır. Bu sorunu çözmek için tüm bağlantıların bir liste şeklinde yer aldığı "site haritaları" kullanılır (Görsel 2.16).



Görsel 2.16: Site haritası örneği

2.5. Güncellik

Web sitesi hazırlanırken dikkat edilmesi gereken unsurlardan bir tanesi de güncelliktir. Sitede içerikle ilgili tüm bilgiler güncel tutulmalıdır. Haber, etkinlik, duyuru gibi eski tarihli bilgiler arşiv şeklinde ayrı bir bölümde tutulmalıdır. Güncel içerikli web sitelerinin arama motoru sonuç sıralamasında üst sıralarda yer alacağı unutulmamalıdır.

Görsel 2.17: Web teknoloji kullanımı

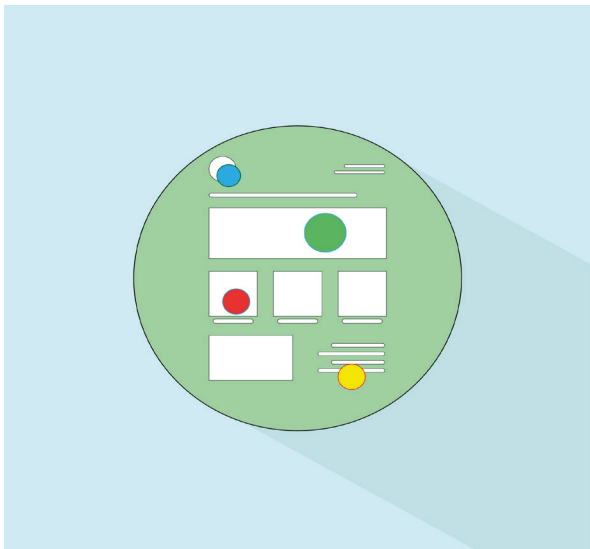
Web sitesi hazırlanırken yeni teknolojilerin kullanılması web sitesinin güncel platformlarda hatasız çalışması açısından önemlidir. Kullanılan kodlama teknolojisi ile site güncel ve sağlıklı çalışır hâle getirilebilir.

Web teknolojileri ve kodlama dillerinde HTML5, CSS3 gibi yeni teknolojilerin kullanılması farklı platformlarda da (akıllı telefon, tablet vb.) içeriği sorunsuz çalıştıracaktır (Görsel 2.17). Duyarlı (responsive) tasarımlar ile web siteleri görüntülendiği ekran boyutuyla uyumlu hâle gelecektir. Responsive tasarımlar yaparken CSS çatılarını (framework) kullanmak kodlama aşamasında kodlayıcının işini kolaylaştıracaktır. Sadece birkaç satır kod ekleyerek hem her cihaza uygun tasarımlar yapabilecek hem de çalışmanın birçok tarayıcıda sorunsuz çalışması sağlanacaktır.

Araştırma

Arkadaşlarınızla küçük gruplar oluşturunuz. Yaygın olarak kullanılan CSS frameworklerini araştırıp avantaj ve dezavantajlarını anlatan bir sunum hazırlayınız.

Yeni teknoloji ile hazırlanan web sitelerine, sitenin kullanıcı üzerindeki etkililiğini ölçmeye yarayan araçlar da eklenebilir. Isı haritaları ve dönüşüm izleme gibi araçlar sayesinde web sitesinde ziyaretçilerin sayfadaki hareketleri (en çok nelere tıkladığı, en çok hangi ürünle ilgilendiği, hangi sayfada ne kadar süre geçirdiği gibi) analiz edilir ve analiz sonucunda web sitesinin güçlü ve zayıf yönleri de tespit edilerek sitenin daha güncel kalması sağlanır (Görsel 2.18).



Görsel 2.18: Isı haritası

2.6. Uygunluk ve Güvenilirlik

Bir web sitesi için alan adı seçerken, seçilecek olan alan adının öncelikle içeriği yansıtmasına özen gösterilmelidir. Alan adının sitenin genel içeriği ile uyumlu olması arama motorlarında bulunabilme şansını da arttıracaktır. Alan adının, hatırlanmasının ve telaffuzunun kolay olacak şekilde dikkat çekici ve kısa olmasına dikkat edilmelidir.

Hazırlanan web sitesinin ziyaretçilere güven vermesi oldukça önemlidir. İçerikler özgün (başka bir yerden alıntı olmayan görsel-işitsel materyal ve içerikler), SEO kaygısı taşımayan, kullanıcı odaklı yazılan güncel bilgilerden oluşmalıdır.

Yazım yanlışları ve anlatım bozuklukları olan bir site güven vermeyeceğinden sitede çalışmayan sayfalar, hatalı yazılmış metinler, sıklıkla açılan reklam sayfaları olmamasına dikkat edilmeli ve dil bilgisi kurallarına uyulmalıdır.

Tüm bu kurallar çerçevesinde hazırlanmış internet siteleri, ziyaretçilerin sitede daha fazla kalmasını sağlayacak ve site arama motorları tarafından da daha değerli hâle gelecektir.

2.7. Uyumluluk

Bir web sitesinin görüntülendiği tüm platformlarda sorunsuz çalışması için tasarımcıların uyumluluk ile ilgili birtakım hususlara dikkat etmesi gerekir.

2.7.1. Tarayıcı Uyumluluğu

Günümüzde internet ortamına erişim masaüstü bilgisayar, laptop, tablet veya akıllı telefonlar kullanılarak yapılmaktadır. Çeşitli amaçlar için hazırlanmış olan web sitelerini ziyaret edebilmek için **tarayıcı** adı verilen programlar kullanılır. Web sitelerinin veya web uygulamalarının mobil ortamlar ve farklı tarayıcılar ile uyumlu olması, hazırlanan içeriğin farklı tarayıcılarda farklı görüntü vermemeleri açısından önemlidir. Bir tarayıcının destekleyip diğerinin desteklemediği bir özellik ile karşı karşıya kalındığında ciddi görünüm bozuklukları meydana gelebilmektedir.

Web siteleri tek bir tarayıcı üzerinden test edildiğinde, farklı tarayıcılar tarafından desteklenmeyen komut ve eklentilerle karşılaşılabilir. Bu komut ve eklentilerden dolayı da bir tarayıcıda çok iyi görünen web sitesi farklı tarayıcı veya aynı tarayıcının farklı versiyonlarında kötü görünecektir.

Sitenin tüm platformlarda doğru çalışıp çalışmadığının kontrolü için **çapraz tarayıcı** testi **(cross browser testing)** adı verilen tarayıcı uyumluluk testleri kullanılır. Bu testler sayesinde web siteleri tüm tarayıcılar üzerinde aşama aşama test edilip henüz tasarım aşamasındayken bile nasıl göründüğü incelenir. Böylelikle farklı tarayıcı, cihaz ve platformları kullanan tüm kullanıcılar için tutarlı bir web deneyimi elde edilmiş olur.

Araştırma

Çapraz tarama testi yapmak için kullanılabilecek araçları araştırıp sınıfta arkadaşlarınızla paylaşınız.

2.7.2. Duyarlı (Responsive) Tasarım

Hazırlanacak olan web sitelerinin masaüstü bilgisayarların yanı sıra tablet ve mobil cihazlarla da uyumlu olması siteye daha çok erişim sağlanması açısından önemlidir. Bu uyumluluk **duyarlı (responsive) tasarım** olarak adlandırılır (Görsel 2.19).

Görsel 2.19: Responsive tasarım

Duyarlı tasarım ile web sitesi içeriğinin, siteyi görüntülemek için kullanılan tüm cihazlarda en iyi ve en kolay şekilde kullanılabilmesi sağlanır. Duyarlı olarak hazırlanmış web sitesi veya mobil uygulamalarda, ekran boyutu ve çözünürlüğü, kullanılan cihazın ekran boyutu ve çözünürlüğüne uygun olarak otomatik algılanacağından, içeriğin sağlıklı bir şeklide görüntülenmesi sağlanır. Yazıların, resimlerin, menülerin kayması engellenip, içeriğe ulaşmak için kaydırma veya yakınlaştırma sorunu ortadan kalkacak ve tüm elemanlar kullanılan ekranların genişliğine göre yeniden şekillenerek içeriğin ekrana tam oturması sağlanacaktır.

Duyarlı web tasarımları hazırlayarak web sitelerinin farklı cihazlarla uyumlu olmasını sağlamak için bir CSS3 özelliği olan CSS Medya sorguları (CSS Media Query) kullanılır. Duyarlı web tasarımın yapı taşı olan Medya query özelliği sayesinde, yüksek piksele göre ayarlanmış olan bir web sitesi belli bir pikselin altına düştüğünde de orijinal görüntüyü korur.

Araştırma

Mobil uygulama testleri için kullanılabilecek araçlardan birkaçını araştırıp sınıfta arkadaşlarınızla paylaşınız.

Tasarımcıların dikkat etmesi gereken bir diğer nokta ise sitelerin arama motoru uyumluluğudur. Arama motorları da duyarlı tasarımı dikkate almaktadır. Duyarlı tasarıma sahip olan siteler arama sonuçlarında daha ön sıralarda yer alacaktır (2.20).

Görsel 2.20: SEO

Web sitesi veya web uygulaması geliştirmek için kullanılan bazı teknolojiler her işletim sistemi ile uyumlu olmayabilir. Hazırlanan çalışmanın farklı işletim sistemi ile uyumluluk testinin de yapılması gerekir. Hazırlanan sitenin mobil cihazlarla uyumluluğunun kontrolü için farklı işletim sistemine sahip telefonlarda birebir test edilmesi de önemlidir.

Web sayfasından çıktı almak gerekebilir. Bu tür durumlarda sağlıklı çıktılar alabilmek için sayfanın yazdırma seçenekleri ile uyumluluğu kontrol edilmelidir. Web sayfasından deneme amaçlı bir çıktı alınmalı ve sayfanın yazı tipi, hizalama, kenar boşlukları, kâğıt boyutu gibi özelliklerin uygun olup olmadığı kontrol edilmelidir.

ÖLÇME VE DEĞERLENDİRME

Aşağıdaki cümlelerin başında boş bırakılan parantezlere, cümlelerde verilen bilgiler doğru ise D, yanlış ise Y yazınız.

1. ( ) Web sitelerinde kullanılacak olan bileşenlerin neler olacağına ve sayfa içindeki konumlarına tasarım aşamasında karar verilir.

2. ( ) Bir web sitesinde kullanılan site haritası bağlantıları sayfanın yalnızca altbilgi kısmında yer alır.

3. ( ) Mor, mavi ve yeşil renkleri nötr renklerdir.

4. ( ) Web siteleri için alan adları belirlenirken sitenin içeriğine uygun olması önemlidir.

5. ( ) Responsive tasarımlar sayesinde web siteleri görüntülendikleri tüm cihazlarda sorun suz çalışırlar.

6. Aşağıdaki tablonun A sütununda verilen bilgilerin önündeki parantezlere, B sütunundaki kavramlardan doğru olana ait harfi yazarak eşleştiriniz.

A Sütunu	B Sütunu
( ) 1. Web sitelerinin ilk sayfalarına verilen addır.	A) Tarayıcı uyumluluk Testi
( ) 2. Web siteleri hazırlanırken yeni teknolojilerin kullanılması gerekliliğinin belirtildiği ilkedir.	B) Anasayfa
	C) Media Query
( ) 3. Bir web sitesinin farklı tarayıcılarda doğru çalışıp çalışmadığının kontrolü için kullanılan testlerdir.	D) Güncellik
	E) CSS3
( ) 4. Duyarlı web tasarımları hazırlayarak web sitelerinin farklı cihazlarla uyumlu olmasını sağlamak için bir CSS özelliğidir.	F) İşlevsellik ve Kullanılabilirlik
	G) Seo Testi

3. ÖĞRENME BİRİMİ

HTML5

NELER ÖĞRENECEKSİNİZ?

Bu öğrenme birimi ile;

- Etiket (tag) kullanımını kavramayı,
- Temel HTML etiketlerini kullanmayı,
- HTML5 belge yapısını oluşturmayı,
- H1-H6 başlık elemanlarını ve özelliklerini kullanmayı,
- HTML belgesindeki metin biçimlendirmeyi,
- HTML belgesinde listeleme etiketlerini kullanmayı,
- HTML belgesine tablo eklemeyi,
- Anlamlı ve anlamsız etiket kavramlarını açıklamayı,
- Div ve span kullanarak HTML tasarım şablonu oluşturmayı,
- HTML5 tasarım şablonu kullanmayı,
- HTML5 belgesine resim, ses ve video eklemeyi,
- Bağlantı elemanlarını kullanarak HTML belgeleri arasında bağlantı kurmayı,
- Form elemanlarını kullanarak form oluşturmayı,
- HTML etiketleriyle HTML belgesi oluşturmayı öğreneceksiniz.

ANAHTAR KELİMELER

Anlamlı etiket, anlamsız etiket, bağlantı (köprü), biçim, body, div, doctype, etiket, form, head, HTML, HTML5, lang, liste, meta, resim elemanı, ses elemanı, span, tablo, tasarım şablonu, title, UTF-8, video elemanı.

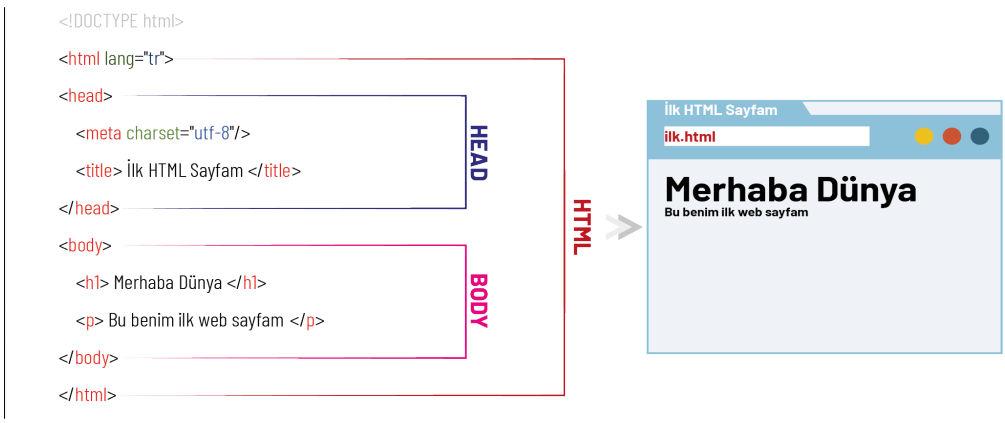
Hazırlık Çalışmaları

1. Okulunuz ile ilgili bir web sayfası hazırlama görevi size verilmiş olsaydı sayfanızda öncelikli olarak nelere yer verirdiniz?

2. Web sayfalarının hazırlanmasında sizce kodlama bilgisi mi yoksa tasarım bilgisi mi gereklidir? Neden? Bildiklerinizi arkadaşlarınızla paylaşınız.

3.1. HTML5 Belge Yapısı

HTML belgesi bir web sitesinin temel yapısını oluşturur ve genel olarak iki kısımdan oluşur: baş (head) kısmı ve gövde (body) kısmı (Görsel 3.1). Bir ev yapılmadan önce temel atılır ve daha sonra temel üzerine ev inşa edilir. Kaba inşaatı bitmiş ev, HTML olarak düşünülebilir. Sadece HTML kodları ile bir web sitesi oluşturmak mümkündür fakat bu kodlar kaba kodlardır. Web sitesinin görsel açıdan zenginleştirilmesi ve esneklik kazanabilmesi için CSS kullanmak gerekmektedir. HTML, dinamik bir dil değildir. HTML ile web sitelerine resim ve video gibi nesneler eklenebilir, yazılar yazılabilir, köprüler ile sayfaların birbirine bağlanması işlemi gerçekleştirilir.



Görsel 3.1: HTML belge yapısı

3.1.1. HTML Hakkında

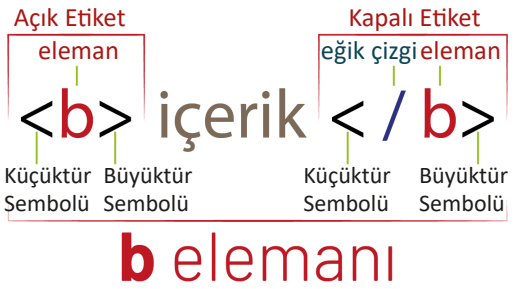
HTML, "Hyper Text Markup Language" kelimelerinin baş harflerinden oluşur ve "Zengin Metin İşaretleme Dili" anlamına gelir. İnternet üzerindeki tüm sayfaların kaynağı HTML'dir. HTML, bir programlama dili değildir. Web sayfaları oluşturmak için standart metin işaretleme dilidir ve web sayfasının yapısını tanımlar. HTML dili; web sayfasında bulunan metin, resim, video, ses gibi içeriklerin sayfadaki yerleşimleri ve ziyaretçiye gösterilmesini sağlayan komutlara sahiptir. HTML belgesi **<html>** etiketi ile başlar ve **</html>** ile biter. HTML etiketleri, tarayıcıya (browser) içeriğin nasıl görüntüleneceğini söyler. HTML etiketleri; "Bu bir başlıktır.", "Bu bir resimdir.", "Bu bir paragraftır." şeklinde içerik parçalarını etiketler.

HTML, büyük ve küçük harfe duyarlı değildir (case-insensitive). HTML etiketleri, büyük ve küçük harf fark etmeden yazılabilir fakat HTML5 standardına göre HTML komutları küçük harfle yazılır.

HTML sayfaları oluşturmak için herhangi bir metin düzenleyici kullanılabilir. HTML kod yazımını kolaylaştıran birçok program mevcuttur. HTML dosyalarının uzantıları **.htm** veya **.html'** dir. İçinde kod olmayan bir dosya **.html** uzantısı ile kaydedildiğinde o dosya bir HTML belgesi olur. HTML belgelerine web sayfası veya web dokümanı da denir.

3.1.1.1. HTML Etiketi

Etiket (tag), HTML belgesinde kullanılan komutlara verilen addır. HTML etiketleri belirli kurallara göre bir araya gelerek HTML belgelerini veya web sayfalarını oluşturur. HTML komutları temel olarak bir açılış etiketi, bir kapanış etiketi ve bu etiketler arasında bulunan içerikten oluşur. Etiketler, küçüktür sembolü "<" ile başlar, büyüktür sembolü ">" ile biter (Görsel 3.2). HTML elemanları için açık etiket ve kapalı etiketler mevcuttur. Etiketi kapatmak için eğik çizgi [slash (slaş)] kullanılır. Bazı etiketlerin kapama etiketi yoktur. <meta> etiketi, kapama etiketi kullanılmayan etiketlere örnektir.



Görsel 3.2: HTML etiket yapısı

HTML etiketleri, kendine has özelliklere (parametre) sahip olabilir. HTML etiketlerinin aldıkları özellikler, açık etiket içerisine elemandan sonra eklenir (Görsel 3.3). Eleman, birden fazla özelliğe sahip ise art arda yazılır ve sıralaması önemli değildir. Özelliklerin değeri çift tırnak veya tek tırnak içerisinde belirtilir. Kapalı etiketlerde özellik yazılmaz.

Görsel 3.3: HTML etiketinde özellik

HTML elemanlarına uygulanabilecek dört temel ortak özellik vardır:

1. **id** : HTML etiketine benzersiz kimlik sağlar.
2. **title :** İpucu görüntülenmesini sağlar.
3. **class:** HTML etiketini stil sayfası aracılığıyla birleştirir.
4. **style:** Etikete özel stil tanımlar.

Aşağıdaki örnek kodda HTML elemanlarına uygulanabilecek dört temel ortak özellik kullanılmıştır.

```
<p id="parag1">HTML dosyasındaki birinci paragraf </p>
<p id="parag2">HTML dosyasındaki ikinci paragraf </p>
<h3 title="Açıklamanın açıklaması">Açıklama</h3>
<p class="sinif1 sinif2">Örnek paragraf</p>
<p style="font-family:calibri;">Örnek paragraf stili</p>
```

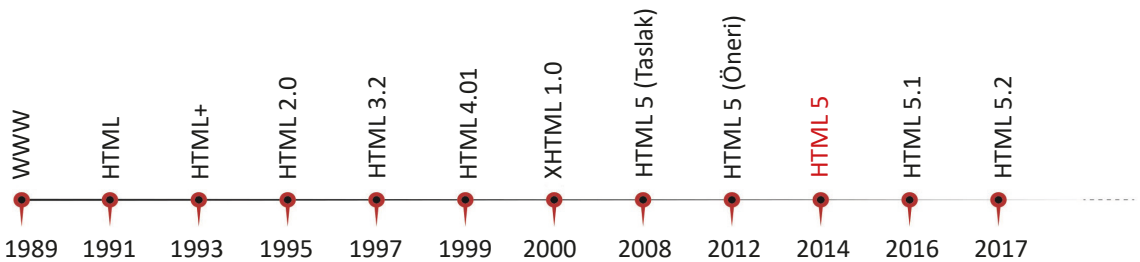
3.1.1.2. HTML5'in Tanımı ve İşlevleri

HTML5, herhangi bir ölçüye bağlı kalmadan her ekranda web sayfalarının düzgün görüntülenmesini sağlayan web duyarlı bir teknolojidir. **HTML5**, mevcut HTML4 etiketlerine eklenen yeni bir HTML etiketler kümesidir. Eski HTML etiketlerinin bazıları geçerliliğini yitirmiş, bazılarının kullanımları daha anlaşılır hâle getirilmiştir. Yeni eklenen HTML etiketleri ile HTML'nin yapabilecekleri artmış ve daha düzenli bir yapıya kavuşmuştur. HTML versiyonları arasında ilk defa özel bir logoya sahip olan versiyon, **HTML5** versiyonudur (Görsel 3.4).

HTML birçok platformda çalışmaktadır. HTML güncellemesi bu platformların da güncellenmesini gerektir. Bundan dolayı HTML çok sık ve köklü bir değişiklik yapılarak güncellenmez (Görsel 3.5).

HTML5=HTML + CSS3 + JavaScript APIs.

Görsel 3.4: HTML5 logosu



Görsel 3.5: Geçmişten günümüze HTML versiyonları

3.1.2. HTML5 Temel Etiketleri ve Belge Yapısı

HTML komutları, tek satırda veya birden fazla satırda alt alta yazılabilir. Okunabilirliği artırmak için iç içe ve girintili yazılır.

Aşağıda HTML etiketlerinde tek satırda yazılarak oluşturulmuş HTML belgesi yapısı vardır.

```
<!-- Tek Satırda oluşturulmuş HTML Belgesi -->
<!DOCTYPE html>
<html><head></head><body></body></html>
```

Aşağıda HTML etiketlerinde alt alta yazılarak oluşturulmuş HTML belgesi yapısı vardır.

```
<!-- Satır Satır alt alta oluşturulmuş HTML Belgesi -->
<!DOCTYPE html>
<html>
<head>
</head>
<body>
</body>
</html>
```

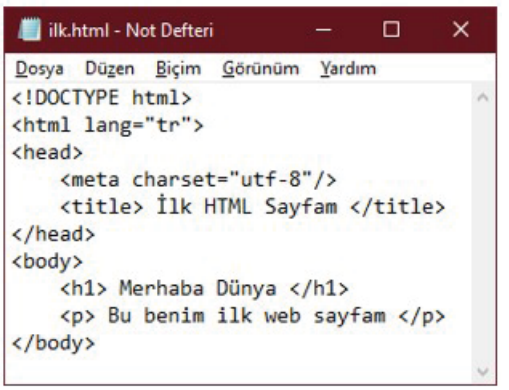
Tek satır veya birden fazla satırda alt alta yazılmış olan HTML belgesi kodları, tarayıcı tarafından aynı şekilde yorumlanır. Programlama yapan açısından ise okunabilirliği değişir.

Aşağıdaki HTML etiketlerinde iç içe girintili yazılarak oluşturulmuş HTML belgesi yapısı vardır.

```
<!DOCTYPE html>
<HTML lang="tr">
<head>
 <meta charset="utf-8" />
 <title>MERHABA HTML DÜNYASI</title>
</head>
<body>
        <!—Merhaba HTML Dünyası-->
        <p> Merhaba HTML Dünyası</p>
</body>
</html>
```

Gelişmiş web tasarım editörleri ile anlamlı HTML kodları daha hızlı ve düzenli bir şekilde yazılır. HTML kod yapısı genel standartlara uygun ve okunabilir olmalıdır.

HTML kodları tek satırda yazıldığında tarayıcı düzgün bir şekilde yorumlar fakat okunabilirliği düşüktür ve standartlara uygun değildir. Kod yazımında hata yapıldığında tespit etmek zordur. Farklı kişiler tarafından kod incelendiğinde anlaşılması güçtür. HTML5 belgesi oluşturmak için gelişmiş özelliklere sahip ücretsiz veya ücretli HTML editörleri yerine, basit metin düzenleme programları da kullanılabilir (Görsel 3.6).

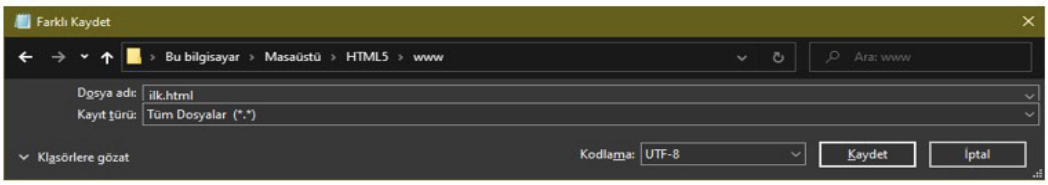


Görsel 3.6: Metin düzenleme programı ile HTML

Araştırma

HTML belgesi oluşturmak için kullanılan ücretli ve ücretsiz editörlerin neler olduğunu araştırınız. Araştırma sonucunda elde ettiğiniz bilgileri sınıfta paylaşınız.

Metin düzenleme programı ile yazılan kodları HTML belgesi olarak kaydetmek için **Dosya** menüsünden **Farklı Kaydet** komutu tıklanır. Açılan pencereden sırasıyla **Dosya adı** ve **Kayıt türü** seçilir (Görsel 3.7).



Görsel 3.7: HTML belgesini kaydetme

HTML belgelerine isim verilirken dikkat edilmesi gereken kurallar şunlardır:

• Dosya adlarında büyük harf yerine küçük harf kullanmak ve bağlantı (köprü) oluşturmak daha kolaydır.

• Türkçe karakter (ç, ğ, ı, ö, ü) kullanılmamalıdır. Yerel (local) sunucuda Türkçe karakter kullanmak HTML belgesinin çalışmasına engel teşkil etmese de internet ortamında sunucuya yüklendiğinde dosyalar görüntülenmez. Bu durum dosyalar ile düzgün bağlantı kurulmasına engel olur.

• Sayfa içeriğine uygun isim verilmelidir. Hakkımda bilgilerinin yer aldığı HTML belgesine "dosya01.html" gibi isim vermek yerine "hakkimda.html" gibi içerikle ilgili isim vermek bağlantı linklerini yazmayı ve bağlantı kurmayı kolaylaştırır.

• Dosya adlarında "boşluk" karakteri kullanılmamalıdır. Dosya adında boşluk karakteri kullanıldığında dosya ile oluşturulan bağlantılar çalışmaz.

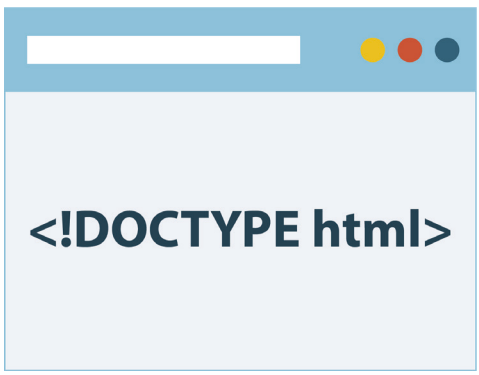
• Dosya isimlerinde birden fazla kelime kullanıldığında aralarında "_ (alt çizgi)" yerine "- (tire)" kullanılmalıdır. İkisi de kullanılabilir fakat arama motorları tire karakterini daha kolay yorumlar.

Dosya isimlerinin yukarıdaki kurallara dikkat edilerek oluşturulması, arama motorlarında web sitesinin doğru indekslenmesini sağlar.

3.1.2.1. DOCTYPE

HTML5'te ilk göze çarpan yenilik DOCTYPE bildirimidir.

DOCTYPE [Belge tipi (Document type)], tarayıcıya (browser) dokümanın tipini işaret eder. DOCTYPE etiket değildir. DOCTYPE deyiminden sonra yazılan ifade ile tarayıcı, dokümanın hangi tipte olduğunu anlar. HTML belgesi için DOCTYPE ifadesinde "html" kullanılır (Görsel 3.8). Web sayfaları için <html> etiketinden önce yazılır. DOCTYPE bildirimi



Görsel 3.8: DOCTYPE kullanımı

yazılmadığında; tarayıcı Quirks Mode'da çalışır, web sayfasının DOCTYPE kullanılmadan önceki bir versiyonu olduğunu düşünür ve bazı etiketler çalışmaz, HTML doğrulayıcı kullanılmaz, stil sayfaları düzgün görüntülenmez.

3.1.2.2. <html> Etiketi

HTML kodları HTML etiketi arasına yazılır. **<html>** etiketi ile başlar **</html>** kapatma etiketi ile sonlandırılır. Sayfanın başlangıcını ve sonunu belirtir. Aşağıda HTML5 iskeleti görülmektedir.

```
<!DOCTYPE html>
<html>
 <head>
   <!— meta bilgileri burada yer alır -->
 </head>
 <body>
   <!— sayfa içeriği burada yer alır -->
 </body>
</html>
```

Tüm meta verileri ve sayfa içeriği HTML etiketi arasında yer alır. HTML5 iskeletindeki etiketlerin sıralaması değiştirilemez.

Sıra Sizde

Metin editörü kullanarak HTML5 iskeletini oluşturunuz. Oluşturduğunuz belgeyi "htmletiketi.html" olarak kaydediniz ve tarayıcıda görüntüleyiniz. Tarayıcıda değişiklik olup olmadığını gözlemleyiniz.

1. Uygulama

HTML iskeletini, metin düzenleyici kullanarak yönergeler doğrultusunda gerçekleştiriniz.

1. Adım: DOCTYPE bildirimini yazınız.

```
<!DOCTYPE html>
```

2. Adım: <html> elemanını oluşturunuz.

```
<html lang="tr">

</html>
```

3. Adım: <head> ve <body> elemanlarını oluşturunuz.

4. Adım: TAB tuşunu kullanarak yazdığınız HTML kodları girintili hâle getiriniz.

```
<head>
 <meta charset="utf-8"/>
 <title> İlk HTML Sayfam </title>
</head>
<body>
 <h1> Merhaba Dünya </h1>
 <p> Bu benim ilk web sayfam </p>
</body>
```

5. Adım: Oluşturduğunuz HTML iskeletini "ilk.html" adında HTML belgesi olarak kaydediniz ve tarayıcıda görüntüleyiniz.

```
<!DOCTYPE html>
<html lang="tr">
 <head>
  <meta charset="utf-8"/>
  <title> İlk HTML Sayfam </title>
 </head>
 <body>
  <h1> Merhaba Dünya </h1>
  <p> Bu benim ilk web sayfam </p>
 </body>
</html>
```

Not

Lang özelliği, HTML5'te kullanılan birincil dili belirtir; <html> etiketine ait bir özelliktir, ayrı bir komut değildir. HTML5'te birincil dil HTML etiketi içerisinde belirtilmelidir. HTML5'ten önceki versiyonlarda dil özelliği DOCTYPE bildirimi içerisinde tanımlanır. HTML etiketine ait lang özelliğinde kullanılan ülke kodları, ISO-639-1 standartlarında ikili olarak belirlenmiştir. Türkçe için lang özelliğinin değeri "tr"dir.

Araştırma

Farklı diller için lang parametresi ile kullanılan ISO-693-1 tablosunu inceleyerek elde ettiğiniz bilgileri arkadaşlarınız ile paylaşınız.

3.1.2.3. <head> Etiketi

<head> etiketi, belgenin başlık kısmını tanımlar. **<head>** etiketi **<html>** etiketinin içinde yer alır. **<head>** açma ve kapama etiketleri arasında site başlığını bildirmek için yazılan **<title>** etiketi dışındaki hiçbir içerik kullanıcı tarafından görülmez.

Arama motorlarına bilgi vermek (SEO çalışması) için **<head>** etiketi arasındaki bilgiler kullanılır. Bu etiket içerisinde sunucu ve arama motorlarına yönelik bildirimler (meta bilgileri), title (başlık), stiller vb. bulunur.

2. Uygulama

HTML kaynak kodunu görüntüleme işlemini yönergeler doğrultusunda gerçekleştiriniz.

1. Adım: https://mtegm.meb.gov.tr adresini tarayıcıda açınız.
2. Adım: Açılan sayfanın boş bir yerinde sağ tıklayınız.
3. Adım: Açılan hızlı menüden "Sayfa Kaynağını Görüntüle" seçeneğini seçiniz.
4. Adım: Açılan HTML kodlarını inceleyiniz.
5. Adım: <head> açık etiketini bulunuz.
6. Adım: </head> kapalı etiketini bulunuz.
7. Adım: HTML belgesi iskeletinde yer alan temel HTML etiketlerini bu sayfa içinde bulunuz.
8. Adım: Bulduğunuz temel etiketleri (html, head, title, body) yukarıdan aşağıya sırayla not alınız.
9. Adım: Bulduğunuz temel etiketlerin sırasını, arkadaşlarınızla karşılaştırınız.

3.1.2.4. <meta> Etiketi

<meta> etiketleri, HTML belgesi hakkında bilgiler içerir; meta veriler, sayfada gösterilmez. Meta verileri tarayıcılar, arama motorları ve diğer web hizmetleri tarafından kullanılır. **<meta>** etiketleri **<head>** etiketi arasına yazılır. **<meta>** etiketinin kapama etiketi bulunmaz Meta verilerinde; sayfa karakter kodlaması, web sayfası hakkında açıklama, arama motorları için anahtar kelime tanımlama, web sayfası yazarı, web sayfası yönlendirme vb. bilgiler bulunur.

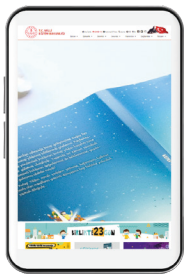
```
<head>
        <meta charset="UTF-8">
        <meta name="description" content="Web Tasarımı ve Programlama">
        <meta name="keywords" content="HTML, CSS, JavaScript">
        <meta name="author" content="İKRA">
        <meta http-equiv ="refresh" content="4; url = 'https://mtegm.meb.gov.tr/' ">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
```

Sıra Sizde

Meta bilgilerini HTML belgesine ekleyip kaydediniz ve tarayıcıda görüntüleyiniz. Sayfada değişiklik olup olmadığını gözlemleyiniz.

Hazırlanan web sayfalarının görüntülendiği tüm cihazlarda düzgün görünmesi için <meta> verileri ile ayarlama işlemine ihtiyaç vardır.

HTML belgesinde viewport tanımlaması yapılmadığında web sitesinin görüntülendiği cihazdaki görüntüsü Görsel 3.9.a'daki gibi olur. Viewport tanımlaması yapıldığında ise Görsel 3.9.b'deki gibi web sayfası cihazda düzgün görüntülenecektir.



Görsel 3.9.a): Viewport tanımlaması yok

Görsel 3.9.b): Viewport tanımlaması var

3. Uygulama

Web sayfası görüntü alanını tüm cihazlarda düzgün görüntülemek için meta verileri ile ayarlama işlemini yönergeler doğrultusunda gerçekleştiriniz.

1. Adım: HTML belgesi temel yapısını oluşturunuz.

2. Adım: Aşağıdaki meta verisini <head> açma ve kapama etiketleri arasına yazınız.

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

3.1.2.5. <style> Etiketi

<style> etiketi ile sayfada kullanılan stil işlemlerinin tanımlaması yapılır. Örneğin; arka plan rengi, yazı rengi ya da nesnelerin hizalanması vb.

```
<head>
  <style> Buraya CSS kodlarınız gelir </style>
</head>
```

3.1.2.6. <title> Etiketi

<title> etiketi içeriği, tarayıcının başlık çubuğunda veya sayfa sekmesinde görüntülenir. **<title>** etiketi <head> etiketi arasında yazılır.

```
<head>
  <meta charset="utf-8"/>
  <title> WEB TABANLI UYGULAMA GELİŞTİRME </title>
</head>
```

SEO açısından sayfa başlığı ve <title> etiketi önemlidir. Arama motorları, web sitesindeki her sayfanın başlık içeriğini veri tabanlarına kaydeder. <title> etiketi içinde geçen bir kelime arandığında aranan kelimeye sahip tüm sayfalar listelenir, sayfanın arama listesinin kaçıncı sırasında olacağını arama motoru algoritması belirler. Başlık, içeriği tanımlayıcı, kısa ve her HTML belgesinde farklı olmalıdır. <title> tagı, sitenin künyesi gibidir. HTML belgesi içeriği ile <title> içeriği uyumlu olmalıdır, arama motorları bu uyumsuzluğu tespit ettiğinde sayfa puanı (page rank) düşer.

4. Uygulama

<title> elemanı kullanarak ANASAYFA, HAKKIMDA, İLETİŞİM başlıklı HTML sayfaları oluşturma işlemini yönergeler doğrultusunda gerçekleştiriniz.

1. Adım: HTML belgesi temel yapısını oluşturunuz.
2. Adım: <head> elemanı içinde <title> ANASAYFA </title> HTML kodunu ekleyiniz.
3. Adım: Oluşturduğunuz HTML belgesini "anasayfa.html" olarak kaydediniz.

```
<!DOCTYPE html>
<html>
<head>
 <meta charset="utf-8"/>
 <title> ANASAYFA </title>
</head>
<body>
   <!--Burada anasayfa içeriği yer alacak-->
</body>
</html>
```

anasayfa.html

4. Adım: "anasayfa.html" HTML belgesinde Head elemanı içindeki <title> ANASAYFA </title> HTML kodunu <title> HAKKIMDA </title> olarak güncelleyiniz.
5. Adım: Bu HTML belgesini "hakkimda.html" olarak "Farklı" kaydediniz.

```
<!DOCTYPE html>
<html>
<head>
 <meta charset="utf-8"/>
 <title> HAKKIMDA </title>
</head>
<body>
   <!--Burada hakkımda içeriği yer alacak-->
</body>
</html>
```

hakkimda.html

6. Adım: "anasayfa.html" HTML belgesinde Head elemanı içine <title> ANASAYFA </title> kodunu <title> İLETİŞİM </title> olarak güncelleyiniz.
7. Adım: Bu HTML belgesini "iletisim.html" olarak "Farklı" kaydediniz.

```
<!DOCTYPE html>
<html>
<head>
 <meta charset="utf-8"/>
 <title> İLETİŞİM </title>
</head>
<body>
   <!-- Burada iletişim içeriği yer alacak-->
</body>
</html>
```

iletisim.html

8. Adım: Oluşturduğunuz sayfaları tarayıcıda görüntüleyiniz. Tarayıcı sekmelerinde görüntülenen başlıkları gözlemleyiniz.

Not

HTML belgeleri isimlendirilirken Türkçe karakterler (ç, Ç, ğ, Ğ, ı, İ, ş, Ş, ö, Ö, ü, Ü) kullanılmaz.

3.1.2.7. <link> Etiketi

Link etiketi, HTML belgesinin başka kaynaklarla olan ilişkisini tanımlar. Simge (icon), stil (style), betik (script) dosyaları ile bağlantı için kullanılır. Bağlantı etiketleri <head> alanında yazılır.

rel özelliği, geçerli sayfa ile bağlantı verilmiş sayfa arasındaki ilişkiyi belirtir. Aşağıda "css" uzantılı bir stil dosyası ve "icon.png" isimli bir simge dosyası ile bağlantı tanımlanmıştır. Link etiketi anlamsız etiketlerden biridir, sadece özellikleri içerir. Genellikle stil sayfalarının belgeye bağlanması için kullanılır.

sizes özelliği HTML5 ile gelen özelliklerden biridir, bağlantı verilen simge için yükseklik ve genişlik değerleri x veya X ile ayrılır

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" href="stil.css"/>
</head>
...
```

Aşağıdaki HTML kodunda link etiketi kullanarak ikon dosyası eklenmiştir.

```
<!DOCTYPE html>
<html>
<head>
  <link rel="icon" href="icon.png" sizes="32x32"/>
</head>
...
```

3.1.2.8. <script> Etiketi

<script> etiketi istemci tarafında çalıştırılacak bir script tanımlar. JavaScript gibi. <script> etiketi içerisinde script ifadeleri olabileceği gibi src özelliği ile harici bir script dosyası çağırılabilir. JavaScript ile HTML belgelerinizi statik bir yapıdan dinamik bir yapıya geçirebilirsiniz. Değişen içerikler, form kontrolleri vb.

Aşağıdaki örnekte "Merhaba Dünya" yazan script kodu eklenmiştir.

```
<!DOCTYPE html>
<html>
  <head>
        <title>Script Kullanımı</title>
  </head>
  <body>
```

Aşağıdaki HTML kodunda link etiketi kullanarak sitil dosyası eklenmiştir.

```
        Bu HTML belgesi script kontrolü için oluşturulmuştur. </br>
        <script>
                document.write("Merhaba Dünya");
        </script>
        <noscript>Tarayıcınız JavaScript desteklemiyor. </noscript>
  </body>
</html>
```

Sıra Sizde

Yukarıdaki script kodlarını kullanarak HTML belgesi oluşturunuz. Farklı web sayfası kodlarını inceleyip <script> kullanılıp kullanılmadığını tespit ediniz.

3.1.2.9. <!-- yorum --> Etiketi

Kod yazılırken kodların yanına bu kodun ne olduğunu ya da neden kullanıldığını hatırlamak için yorumlar eklenir. Kodlar üzerinde birden fazla kişi çalıştığında önemli olduğu düşünülen, unutulabilecek kodlar var ise açıklama yazılabilir. Bunun için **<!-- açıklama -->** şeklinde yazılan yorum etiketi kullanılır. Yorum etiketleri yazmak kodların anlaşılmasını kolaylaştırır. Yorum etiketleri, HTML belgesinin tüm bölümlerinde kullanılabilir. Yorum etiketlerindeki açıklamalar, tarayıcı ekranında görünmez; kodu inceleyen kişiler tarafından okunabilir. Açıklama satırları bir veya birkaç kelime olabileceği gibi sayfalarca da olabilir. Yorum etiketlerinin diğer HTML etiketleri gibi özellikleri bulunmaz.

Aşağıdaki örnekte açıklama satırı kullanılmıştır.

```
<!-- Bu bir açıklama satırı -->
```

Aşağıdaki örnekte yorum satırı kullanılmıştır.

```
<!-- Değerler, toplumdaki bireylerin korumaları veya göz ardı etmeleri neticesinde ya zamanla kaybolur veya nesilden nesile aktarılarak yıllarca devam ettirilebilir. Birçok toplum tarafından kabul gören ve yıllarca sürdürülen genel (evrensel) değerler incelendiğinde ise karşımıza, liderlik, doğruluk, ahlak, adalet, sorumluluk ve yardımseverlik gibi bireyi ideal insan olma çizgisine yönlendiren değerler çıkmaktadır. Kişiler, dâhil oldukları grup, toplum ve kültür değerlerini genellikle benimseyerek, bunları kendi muhakeme ve seçimlerinde birer ölçüt olarak kullanırlar (Dilmaç, 2009, s.29).

Toplumlar, birçok yönden birbirlerinden farklılıklar göstermişler, ideal insan oluşturma yönündeki değerleri daima önemsemiş ve her dönem yaşatmaya çalışmışlardır. Günümüz toplumlarında, bu tarz üstün değerlerin yaşatılması ve yeni nesillere aktarılması görevini üstlenen kurumlar arasında okullar da yer alır (Belet, 2008, s.2). Bilişsel hedeflerinin yanında vatansever olmak, saygılı olmak, dürüst olmak ve adil olmak gibi birçok duyuşsal hedefi olan okulların görevlerinden biri, okul programında açık olarak belirtilen veya belirtilmeyen değerleri öğretmek, öğrencileri belirlenen kurallar doğrultusunda disipline etmek, onların ahlaki gelişimlerine katkıda bulunmak ve karakterlerini olumlu yönde etkilemektir (Akbaş, 2008, s.9) -->
```

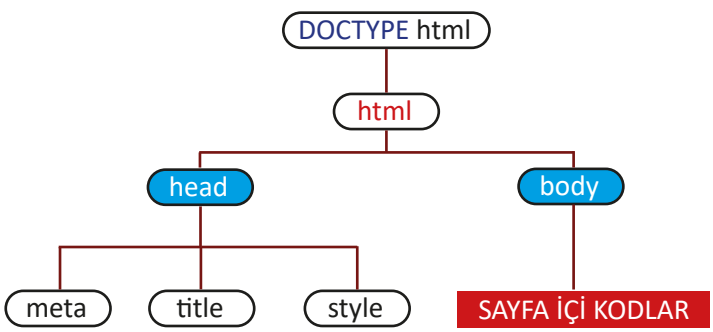
3.1.2.10. <body> Etiketi

<body> etiketi belgenin bedenini tanımlar ve başlık, paragraf, görüntü, köprü, tablo, liste gibi tüm görünür içerik için bir kap oluşturur. **<body>** HTML belgesinin tüm içeriğinin yer aldığı bölümdür. Bu bölümde yer alan içeriğin tümü tarayıcıda görüntülenir.

<body> etiketi de **<html>** ve **<head>** etiketleri gibi tüm belge içeriğinde sadece bir kez kullanılır. HTML belgesi şematik yapısı aşağıdaki gibidir.

```
<!DOCTYPE html>
<html>
  <head>
        <meta charset="utf-8">
        <title>Html yapısı</title>
        <style>
              <!-- Sitil kodları -->
        </style>
  </head>
  <body>
        <!-- Sayfa içi kodları -->
  </body>
</html>
```

<body> etiketi bir kutu gibidir, bu kutuya web sayfasında bulunmasını istediğimiz her eleman düzgün bir şekilde atılır. Kutuda bulunan her elemanı tarayıcı ekranı görüntüleyecektir (Şema 3.1).



Şema 3.1: <body> etiketinin HTML hiyerarşisindeki konumu

3.1.3. HTML5 Anlamsal Etiketler

Anlamsal (semantik) etiket, bir web belgesinin genel düzenini oluştururken belgeyi daha anlamlı hâle getirmek için tasarlanmış etiketler kümesidir. HTML5'ten önceki sürümlerde anlamsal yetersizlik vardır. Tasarımcılar, bu sorunu class ve id kullanarak çözmeye çalışmıştır fakat bu çözüm, kodları daha karışık hâle getirmiştir. HTML5'te anlamsal etiketler (semantic elements) kullanılarak bu sorun çözülmüştür. HTML5 ile menü, içerik, sayfa alt bilgisi (footer) vb. kısımlar için anlamsal etiketler kullanılmaktadır. Anlamsal bir etiket hem tarayıcı hem de geliştirici için açıkça anlaşılır.

Anlamsal olmayan etiklere örnek:

 ve içeriği hakkında hiçbir bilgi vermez. HTML belgesinde bu etiketlere id özelliğini tanımlamadan bir anlam yüklemek zordur.

Anlamsal etiketlere örnek:

<form>, <table> ve <article> kutu modeline göre içeriğinde neyi barındıracağını açıkça tanımlar. HTML belgesinde bu etiketler içeriğinde neleri barındıracağı hakkında fikir verir.

DOCTYPE bildirimine göre HTML5 belgesinde kullanılabilecek etiketler ve açıklamaları Tablo 3.1'de verilmiştir.

Tablo 3.1: DOCTYPE Bildirimine Göre HTML5 Anlamsal Etiketler ve Bu Etiketlerin Açıklamaları

Etiket	Açıklama
<canvas>	Sayfada bir tuval alanı oluşturur. Tuvale çizim javascript ile yapılabilir.
<audio>	Sayfaya ses oynatıcı bir modül ekler.
<video>	Video oynatıcı bir modül ekler.
<progress>	İşlem süreci göstergesi ekler.
<caption>	Başlık olarak düşünülen metinleri düzenler.
<header>	Sitenin başlık ve açıklama içeriğini içine alır.
<nav>	Menüleri ve bir takım zaruri işlevleri içine alır.
<footer>	Sitelerin en alt kısmını içine alır.
<action>	Sitelerin ana içerik kısmını içine alır.
<aside>	Ana içerikte yazılan kısım <aside> ile ayrılır.
<article>	Makale, deneme tarzı yazılar <article> ile yazılır.
<embed>	Dışarıdan eklenen bileşenler (component) için kullanılır (swf uzantılı dosyalar vb.).
<details>	Detay bilgisi içerir.
<summary>	Yazının başlığını belirler.
<time>	Tarih ve saat verilerini kapsar.
<mark>	Yazı içerisinde özellikle üstünde durulan kelimeleri belirler.
<figcaption>	<figure> elementinin başlığını belirler.
<figure>	Çeşitli medya içeriği gruplarını belirler.
<hgroup>	Başlık grubunu belirler. H1, H2 gibi başlık elementleri burada tanımlanabilir.
datalist>	Düzenlenebilir elemanlara otomatik tamamlama özelliği verilmesini sağlar.

3.2. Başlık Elemanları

Sayfa gövdesinde yer alan görsel ögeler genel olarak iki kategoride sınıflandırılabilir:

- Blok HTML etiketleri (p, div, başlık elemanları, listeler vb.)
- Satır içi HTML etiketleri (a, span, em vb.)

Blok HTML etiketleri; sayfada konumlandıklarında sağ ve sol taraflarında diğer ögelerin konumlanmasına izin vermezler. HTML başlık etiketleri de blok HTML etiketidir. Başlık etiketlerinden sonra eklenen ögeler bir alt satırdan devam eder. CSS kullanarak blok etiketlerinin genişlik, yükseklik, kenar boşluğu, dolgu ve kenar gibi görsel özellikleri değiştirilebilir fakat satır içi HTML etiketlerde bu değişiklikler yapılmaz.

3.2.1. HTML <h1> - <h6> Başlık Etiketleri

Başlık etiketleri; <h1>, <h2>, <h3>, <h4>, <h5>, <h6>'dır. Başlık kelimesinin İngilizce karşılığı olan heading kelimesinin baş harfi kullanılarak oluşturulmuş anlamsal etiketlerdir. <h1> ... <h6> etiketleri HTML başlıkları tanımlamak için kullanılır. Başlık etiketleri arasında kullanılan metin altında ve üstünde otomatik oluşan boş satırlar ile diğer metinlerden ayrılır ve tek satırda gösterilir. <h1> en büyük başlığı, <h6> en küçük başlığı ifade etmek için kullanılır. Başlık etiketleri aynı rakama sahip kapama etiketi ile kullanılır.

5. Uygulama

Başlık etiketleriyle içerik oluşturma işlemini yönergeler doğrultusunda gerçekleştiriniz.

1. Adım: Temel HTML yapısını oluşturunuz.
2. Adım: <body> etiketi içine aşağıdaki başlık etiketlerini yazınız.
3. Adım: HTML belgesini kaydediniz ve tarayıcı ekranında görüntüleyiniz.

```
<h1>Türkiye</h1>
<h2>Türkiye</h2>
<h3>Türkiye</h3>
<h4>Türkiye</h4>
<h5>Türkiye</h5>
<h6>Türkiye</h6>
```

Türkiye

Türkiye

Türkiye

Türkiye

Türkiye

Türkiye

Görsel 3.10: <hx> kullanımı

Not

Görsel 3.10'da ekran çıktısı verilen yukarıdaki örnekte <h1> en büyük başlığı, <h6> ise en küçük başlığı temsil eder.

3.2.2. Başlık Etiketlerinin Kullanım Sebepleri

Başlık etiketleri, görsel açıdan web içeriği ve web makalelerini daha okunaklı ve anlaşılır hâle getirir. HTML5 ile gelen HTML5 anlamsal etiketlerini kullanmak arama motorları açısından büyük önem taşır. Oluşturulan içerikleri HTML anlamsal etiketleri içinde sunmak, web sitesini indekslemeye gelen arama motorlarına içerik ile alakalı daha anlamlı bilgiler sunar.

<h1> etiketi de anlamsal etikettir. <h1> etiketi ile oluşturulan başlığın nasıl göründüğünden çok hangi anlamda bilgi sunduğu önemlidir. <h1> etiketi kullanılarak oluşturulan başlığın büyüklüğü farklı yöntemler kullanılarak da oluşturulabilir fakat arama motorları görselliği yorumlayamadığı için HTML anlamsal kodları ile sunulmalıdır.

3.2.3. Başlık Kullanımı

h başlıklar, h1'den h6'ya kadar olsa bile herhangi bir içerikte h4'ten sonraki başlıklar genellikle kullanılmaz. Gereğinden fazla alt başlık kullanımı web okuyucularının dikkatini dağıtır, bir metnin başlıklara bölünmesindeki temel amaç olan kullanıcı deneyimini olumsuz etkiler.

Görsel olarak güzel olduğu düşünülerek h1 başlığından sonra direkt h3 başlık oluşturulmamalı, bunun yerine h başlıkları anlamsal olarak kullanılmalıdır. Herhangi bir metne biçimsel özelliklerini kazandırmak için CSS kullanılmalıdır.

<h1> etiketi içine yazılan başlık, HTML sayfasının en önemli özet bilgisi olarak algılanır ve sayfada bir tane kullanılmalıdır. Alt başlıklar bu başlığı takip edecek şekilde hiyerarşik olarak kullanılır.

Sıra Sizde

Başlık etiketlerini kullanarak bir HTML belgesi oluşturunuz. Kullandığınız başlık etiketlerini arkadaşlarınızın kullandıklarıyla karşılaştırınız.

3.3. Paragraflar ve Metin Biçimlendirme

HTML sayfalarında paragraf oluşturmak için <p> etiketi kullanılır. <p> etiketi, blok HTML etiketidir. Blok ögeler iç içe kullanılmaz fakat satır içi HTML etiketleri diğer satır içi etiketler ile yan yana kullanılabilir.

6. Uygulama

Paragraflar ve metin biçimlendirme işlemini yönergeler doğrultusunda gerçekleştiriniz.

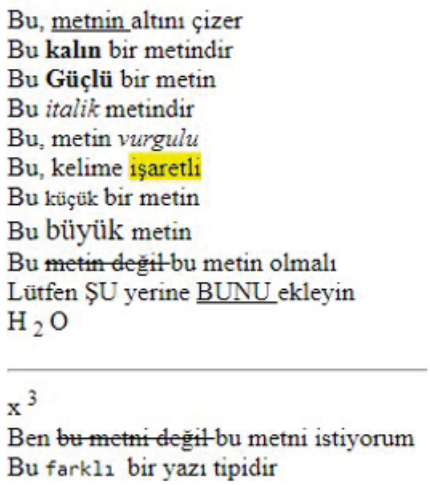
1. Adım: Temel HTML yapısını metin düzenleme programında oluşturunuz.
2. Adım: Metni <body> etiketi içine yazınız.

```
<p>
                Bu, metnin altını çizer
                Bu kalın bir metindir
                Bu Güçlü bir metin
                Bu italik metindir
                Bu, metin vurgulu
                Bu, kelime işaretli
                Bu küçük bir metin
                Bu büyük metin
                Bu metin değil bu metin olmalı
                Lütfen ŞU yerine BUNU ekleyin
                H 2 O
                x 3
                Ben bu metni değil bu metni istiyorum
                Bu farklı bir yazı tipidir
</p>
```

3. Adım: HTML belgesini kaydediniz ve tarayıcıda görüntüleyiniz.

4. Adım: Aşağıdaki metin biçimlendirme komutlarını yazınız ve tarayıcıda görüntüleyiniz.

```
<!DOCTYPE html>
<html>
 <head>
        <title>Metin Biçimlendirme Uygulaması</title>
 </head>
 <body>
        <p>
                Bu, <u> metnin </u> altını çizer <br/>
                Bu <b> kalın </b> bir metindir <br/>
                Bu <strong> Güçlü </strong> bir metin <br/>
                Bu <i> italik </i> metindir <br/>
                Bu, metin <em> vurgulu </em> <br/>
                Bu, kelime <mark> işaretli </mark> <br/>
                Bu <small> küçük </small> bir metin <br/>
                Bu <big> büyük </big> metin <br/>
                Bu <del> metin değil </del> bu metin olmalı <br/>
                Lütfen ŞU yerine <ins> BUNU </ins> ekleyin <br/>
                H <sub> 2 </sub> O <br/>
                <hr/>
                x <sup> 3 </sup> <br/>
                Ben <strike> bu metni değil </strike> bu metni istiyorum <br/>
                Bu <tt> farklı </tt> bir yazı tipidir
        </p>
 </body>
</html>
```



Görsel 3.11: Metin biçimlendirmesi

Not

Görsel 3.11' de metin biçimlendirme etiketleri kullanılarak oluşturulan HTML belgesinde, sık kullanılan HTML etiketlerinin nasıl yazıldığı ve tarayıcıda nasıl görüntülendiği gösterilmiştir. Bu örnekte kullanılan <b>, <u>, <i> anlamsal olmayan etiketlerdir.

3.3.1. Etiketi

** etiketi**, paragraf tanımlamak için kullanılır. kapama etiketi ile kullanılır. … etiketleri arasında kalan metin paragraf blokunu oluşturur. Art arda bloku kullanıldığında tarayıcılarda varsayılan olarak üstte ve altta kenar boşluğu olan bir paragraf stili uygulanır. Günümüzdeki tarayıcılar, paragraf blokları yanlış açılıp kapatıldığında da paragrafı düzgün yorumlar fakat HTML kod blok sayısı arttıkça kodun okunabilirliği azalır, HTML kodlarındaki hataların tespiti zorlaşır (Tablo 3.2).

Tablo 3.2: Etiketi Kullanımı

Doğru Kullanım	Yanlış Kullanım
Bu birinci paragraf Bu ikinci paragraf	Bu birinci paragraf Bu ikinci paragraf

3.3.2. <b> ve Etiketi

Html belgesi içindeki yazıları belirgin yapmak için **<b>** veya **** etiketleri kullanılır. etiketi, anlamsal etikettir. Görsel açıdan <b> ve etiketleri arasında bir fark yoktur. Mantıksal olarak daha önemli içerikler için <b> etiketi yerine **** kullanılır. **** etiketi, arama motorlarına bu metnin önemli olduğunu bildirir.

Sıra Sizde

Görsel 3.12'de tarayıcıdaki görüntüsü verilen HTML belgesini oluşturunuz. Aşağıda kutucuk içinde verilen HTML komutları arasındaki farkları arkadaşlarınızla tartışınız.

```
<h2>BİZDEN HABERLER</h2>
<p>MESLEKİ EĞİTİMDE <strong> 1000 OKUL İYİLEŞTİRME PROJESİ </strong> KAPSAMINDAKİ <mark> TÜM OKULLARIN </mark> YÖNETİCİLERİNE <b> BİLGİLENDİRME TOPLANTILARI </b> YAPILDI</p>
```

BİZDEN HABERLER

MESLEKİ EĞİTİMDE **1000 OKUL İYİLEŞTİRME PROJESİ** KAPSAMINDAKİ TÜM OKULLARIN YÖNETİCİLERİNE **BİLGİLENDİRME TOPLANTILARI** YAPILDI

Görsel 3.12: <b> ve etiketi kullanımı

3.3.3. <i> ve Etiketleri

HTML belgesindeki yazıları eğik (italic) yazmak için **<i>** veya **** etiketleri kullanılır. etiketi, anlamsal etikettir. Görsel açıdan <i> ve etiketleri arasında bir fark yoktur. Mantıksal olarak daha önemli içerikler için <i> etiketi yerine **** kullanılır.

Sıra Sizde

Görsel 3.13'te tarayıcıdaki görüntüsü verilen HTML belgesini oluşturunuz. Bu HTML komutları arasındaki farkları arkadaşlarınızla tartışınız.

```
<h2>Bilişim Terimleri </h2> <b>text-formatting language:</b> <em> metin biçimlendirme dili,</em> [07.01.28], Metnin ne şekilde biçimlendirileceğini belirtmek için tasarlanmış problem yönelimli dil. ÖRNEKLER: <i> HTML, nroff </i>
```

Bilişim Terimleri

text-formatting language: *metin biçimlendirme dili,* [07.01.28], Metnin ne şekilde biçimlendirileceğini belirtmek için tasarlanmış problem yönelimli dil. ÖRNEKLER: *HTML, nroff*

Görsel 3.13 <i> ve etiketi kullanımı

3.3.4. <u> ve <ins> Etiketleri

HTML belgesindeki yazıların altını çizmek için **<u>** veya **<ins>** etiketleri kullanılır. <ins> etiketi, anlamsal etikettir. Görsel açıdan <u> ve <ins> etiketleri arasında bir fark yoktur. <u> etiketi, HTML5 tarafından desteklenmemektedir.

Sıra Sizde

Görsel 3.14'te tarayıcıdaki görüntüsü verilen HTML belgesini oluşturunuz. Bu HTML komutları arasındaki farkları arkadaşlarınızla tartışınız.

```
<ins>   İlim ilim bilmektir   </ins></br>
        İlim kendin bilmektir  </br>
        Sen kendin bilmezsin   </br>
<u>     Ya nice okumaktır      </u></br>
<i><b>  Yunus Emre             </b></i>
```

İlim ilim bilmektir
İlim kendin bilmektir
Sen kendin bilmezsin
Ya nice okumaktır
Yunus Emre

Görsel 3.14: <u> ve <ins > etiketi kullanımı

3.3.5. <small> Etiketi

HTML belgesi içinde daha küçük paragraf metni yazmak için kullanılır.

Sıra Sizde

Aşağıdaki kodları kullanarak HTML belgesi oluşturup tarayıcıda görüntüleyiniz.

```
Bu metin normal büyüklükte yazılmıştır. </br>
<small> Bu metin small etiketi içinde yazılmış küçük bir metindir. </small>
```

3.3.6. <pre> Etiketi

<pre> etiketi, standart metin uzunluğu tanımlar. Metin, sabit genişlikli bir yazı tipinde görüntülenir ve hem boşlukları hem de satır sonlarını korur. Metin, tam olarak HTML kaynak kodunda yazıldığı gibi görüntülenir.

Sıra Sizde

Aşağıdaki kodları kullanarak HTML belgesi oluşturup tarayıcıda görüntüleyiniz.

```
<p><strong>İSTİKLAL MARŞI</strong></p>

<pre>   Korkma, sönmez bu şafaklarda yüzen al sancak;
       Sönmeden yurdumun üstünde tüten en son ocak.
       O benim milletimin yıldızıdır, parlayacak;
       O benimdir, o benim milletimindir ancak.

                                  Çatma, kurban olayım çehreni ey nazlı hilâl!
                                  Kahraman ırkıma bir gül… ne bu şiddet bu celâl?
                                  Sana olmaz dökülen kanlarımız sonra helâl,
                                  Hakkıdır, Hakk'a tapan, milletimin istiklâl.</pre>
```

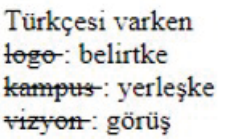
3.3.7. <del> Etiketi

HTML belgesindeki yazıların üstünü çizmek için **<del>** etiketi kullanılır.

Sıra Sizde

Görsel 3.15'te tarayıcıdaki görüntüsü verilen HTML belgesini oluşturunuz.

```
Türkçesi varken </br>
<del> logo </del> : belirtke </br>
<del> kampus </del> : yerleşke </br>
<del> vizyon </del> : görüş </br>
```



Görsel 3.15: <del> etiketi kullanımı

3.3.8. <sup> ve <sub> Etiketleri

HTML belgesi içinde metin karakterlerini üste almak için <sup>, alta almak için <sub> etiketi kullanılır.

Sıra Sizde

Aşağıdaki HTML komutlarını kullanarak HTML belgesini oluşturunuz ve tarayıcıda görüntüleyiniz.

```
H<sub>2</sub>O bileşiği hayatın, C<sub>6</sub>H<sub>12</sub>O<sub>6</sub> bileşiği ise vücudun temel ihtiyaç maddelerinden biridir. </br>
<hr>
(x+y)<sup>2</sup>=x<sub>2</sub>+2xy+y<sub>2</sub>
```

7. Uygulama

Atatürk'ün Gençliğe Hitabesi'ni paragraf ve metin biçimlendirme HTML komutlarını kullanarak yönergeler doğrultusunda gerçekleştiriniz.

1. Adım: Temel HTML yapısını metin düzenleme programında oluşturunuz.
2. Adım: Atatürk'ün Gençliğe Hitabesini <body> etiketi içine yazınız.
3. Adım: HTML belgesi olarak kaydettikten sonra tarayıcıda görüntüleyiniz.
4. Adım: Paragraf ve metin biçimlendirme etiketlerini kullanarak biçimlendiriniz.
5. Adım: HTML belgesini kaydedip tarayıcıda tekrar görüntüleyiniz.

```
<p><strong>ATATÜRK'ÜN GENÇLİĞE HİTABESİ</strong></p><p><b> Ey Türk gençliği! </b><ins>Birinci vazifen; </ins> <br> Türk istiklalini, <br> Türk cumhuriyetini, <u> ilelebet muhafaza ve müdafaa etmektir. </u></p><p><mark>Mevcudiyetinin ve istikbalinin yegâne temeli budur. </mark> Bu temel, senin en kıymetli hazinendir. İstikbalde dahi seni bu hazineden mahrum etmek isteyecek dâhilî ve haricî bedhahların olacaktır. Bir gün, istiklal ve cumhuriyeti müdafaa mecburiyetine düşersen, vazifeye atılmak için içinde bulunacağın vaziyetin imkân ve şeraitini düşünmeyeceksin. Bu imkân ve şerait, çok namüsait bir mahiyette tezahür edebilir. İstiklal ve cumhuriyetine kastedecek düşmanlar, bütün dünyada emsali görülmemiş bir galibiyetin mümessili olabilirler. Cebren ve hile ile aziz vatanın
```

```
bütün kaleleri zapt edilmiş, bütün tersanelerine girilmiş, bütün orduları dağıtılmış ve memleketin her köşesi bilfiil işgal edilmiş olabilir. Bütün bu şeraitten daha elim ve daha vahim olmak üzere, memleketin dâhilinde iktidara sahip olanlar, gaflet ve dalalet ve hatta hıyanet içinde bulunabilirler. Hatta bu iktidar sahipleri, şahsi menfaatlerini müstevlilerin siyasi emelleriyle tevhit edebilirler. Millet, fakruzaruret içinde harap ve bitap düşmüş olabilir. </p> <p> <b> Ey Türk istikbalinin evladı! </b> <em> İşte, bu ahval ve şerait içinde dahi vazifen, Türk istiklal ve cumhuriyetini kurtarmaktır. Muhtaç olduğun kudret, damarlarındaki asil kanda mevcuttur. </em> </p><p><b>Mustafa Kemal Atatürk<b></p><small> 20 Ekim 1927 </small>
```

3.3.9. Listeleme Etiketleri

HTML web sayfalarında kullanılan üç tür HTML listeleme etiketi (Tablo 3.3) şunlardır:

Tablo 3.3: Listeleme Etiketleri

Etiket	Açıklama
** **	Listeleri madde işaretleri ve daireler aracılığıyla listelemek için kullanılır.
** **	Listeleri numaralandırmak için kullanılır.
<dl>	Sözlük düzenleme şeklinde listelemek için kullanılır.

3.3.9.1. Sıralı Liste Etiketleri

Sıralı listeler **<ol>** etiketi ile oluşturulur. Liste ögeleri ise **<li>** etiketi kullanılarak sıralanır. Sıralı listelerin sırasız listelerden farkı numaraya sahip olmalarıdır.

Sıra Sizde

Sıralı liste etiketi kullanılarak oluşturulmuş aşağıdaki HTML listeleme komutlarını kullanarak HTML belgesi oluşturunuz ve tarayıcıda görüntüleyiniz (Görsel 3.16).

```
<ol>
  <li>Vatanseverlik</li>
  <li>Yardımlaşma</li>
  <li>Sorumluluk</li>
  <li>Merhamet</li>
  <li>Dürüstlük</li>
  <li>Hoşgörü</li>
  <li>Saygı</li>
  <li>Sevgi</li>
</ol>
```

- Vatanseverlik
- Yardımlaşma
- Sorumluluk
- Merhamet
- Dürüstlük
- Hoşgörü
- Saygı
- Sevgi

Görsel 3.16: Sıralı liste

3.3.9.2. Sırasız Liste Etiketleri

Sırasız listeler **<ul>** etiketi ile oluşturulur. Liste ögeleri ise **<li>** etiketi kullanılarak sıralanır.

Sıra Sizde

Sırasız liste etiketi kullanılarak oluşturulmuş aşağıdaki HTML listeleme komutlarını kullanarak HTML belgesi oluşturunuz ve tarayıcıda görüntüleyiniz (Görsel 3.17).

```
<ul>
 <li>Vatanseverlik</li>
 <li>Yardımlaşma</li>
 <li>Sorumluluk</li>
 <li>Merhamet</li>
 <li>Dürüstlük</li>
 <li>Hoşgörü</li>
 <li>Saygı</li>
 <li>Sevgi</li>
</ul>
```

1. Vatanseverlik
2. Yardımlaşma
3. Sorumluluk
4. Merhamet
5. Dürüstlük
6. Hoşgörü
7. Saygı
8. Sevgi

Görsel 3.17: Sırasız liste

3.3.9.3. Tanım Listesi Etkileri

Tanım listeleri **<dl>** etiketi ile oluşturulur. Liste ögeleri ise **<dt>** etiketi kullanılarak sıralanır. Diğer listeleme etiketlerinden farklı olarak **<dd>** etiketi kullanılarak liste ögelerinin açıklamaları yapılır.

Sıra Sizde

Tanım listesi etiketleri kullanılarak oluşturulmuş aşağıdaki HTML listeleme komutlarını kullanarak HTML belgesi oluşturunuz ve tarayıcıda görüntüleyiniz.

```
<dl>
 <dt>Ahlak</dt>
 <dd>Bir toplumun en üst genel standartlarını içerir. </dd>
 <dt>Etik</dt>
 <dd>Toplumdaki belirli bir mesleğin, işin, kurumun veya grubun standartlarıdır. </dd>
 <dt>Meslek Ahlakı</dt>
 <dd> Mesleğin genel ilkelerini ve standartlarını ortaya koyan ve mesleği yapan bireylere yol gösteren ilkeler bütünüdür.</dd>
</dl>
```

3.3.9.4. Listeleme TYPE Özelliği

Type özelliği HTML5 listeleme özelliklerinin değiştirilmesine olanak tanır. CSS kullanılarak veya TYPE özelliği kullanılarak listeleme özelliklerinde değişiklik yapılır (Tablo 3.4 ve Tablo 3.5). TYPE özelliği belirtilmediğinde varsayılan değer kullanılır.

Tablo 3.4: UL TYPE Kullanımı

UL TYPE Kullanımı	Açıklama
<ul type="square">	İçi dolu kare kullanarak madde işareti oluşturur.
<ul type="disc">	İçi dolu daire kullanarak madde işareti oluşturur (Varsayılan değer).
<ul type="circle">	Boş daire kullanarak madde işareti oluşturur.

Tablo 3.5: OL TYPE Kullanımı

OL TYPE Kullanımı	Açıklama
<ol type="1">	Sıralı liste türü (Varsayılan değer)
<ol type="i">	Küçük harfli sayılar ile listeleme (roma rakamı)
<ol type="I">	Büyük harfli sayılar ile listeleme (roma rakamı)
<ol type="a">	Küçük harfler ile listeleme
<ol type="A">	Büyük harfler kullanarak listeleme

Sıralı listelerde başlangıç değeri **start** özelliği ile değiştirilir. Kaçıncı değerden başlatılacağı start özelliğine rakam yazılarak belirlenir (Tablo 3.6).

Tablo 3.6: Sıralı Listelerde Başlangıç Değeri Belirleme

OL TYPE START Kullanımı	Açıklama
<ol type="a" start="4">	Başlangıç değeri **d** olarak belirlenmiştir.
<ol type="A" start="5">	Başlangıç değeri **E** olarak belirlenmiştir.
<ol reversed>	Sıralamayı tersine çevirir.

Sıra Sizde

Sıralı listelerde başlangıç değeri belirlenerek oluşturulmuş aşağıdaki HTML listeleme komutlarını kullanarak Görsel 3.18'deki HTML belgesini oluşturunuz ve tarayıcıda görüntüleyiniz.

```
<ol type="a" start="5">
  <li>HTML5</li>
  <li>CSS</li>
  <li>JavaScript</li>
  <li>ASP.Net Core</li>
</ol>
```

```
<ol type="i" start="2">
  <li>HTML5</li>
  <li>CSS</li>
  <li>JavaScript</li>
  <li>ASP.Net Core</li>
</ol>
```

```
<ol reversed>
  <li>HTML5</li>
  <li>CSS</li>
  <li>JavaScript</li>
  <li>ASP.Net Core</li>
</ol>
```

e. HTML5
f. CSS
g. JavaScript
h. ASP.Net Core

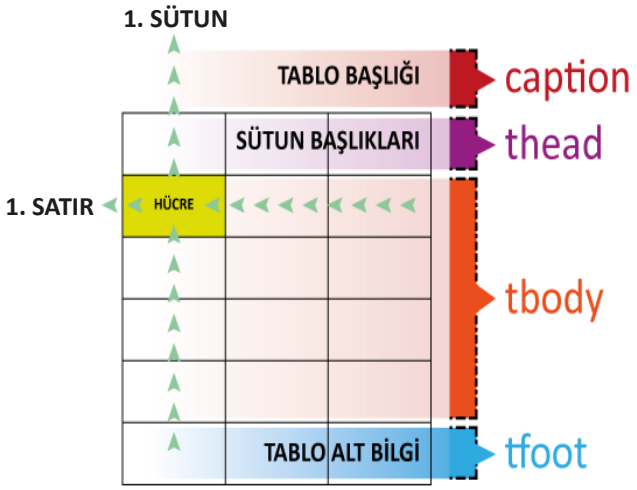
ii. HTML5
iii. CSS
iv. JavaScript
v. ASP.Net Core

4. HTML5
3. CSS
2. JavaScript
1. ASP.Net Core

Görsel 3.18: <ol> etiketi TYPE özelliği

3.3.10. Tablolar

Satır ve sütunlar şeklinde düzenlenebilecek ögeleri web sayfasında kullanmak için **HTML tabloları** kullanılır. Tablolar içerisinde metinler, listeler, resimler, videolar bağlantılar, tablo içinde tablolar vb. veriler kullanılır. Tablolardaki satır ve sütunların kesişim noktalarına **hücre** adı verilir. Veriler hücrelere yerleştirilir. HTML5 kullanılmaya başlanmadan önce tablolar yerleşim elemanları olarak kullanılırdı. HTML5 ile tablosuz tasarım kullanılmaktadır. Web sayfası içeriklerinde verilerin bir arada ve düzenli tutulması için tablolar kullanılır (Görsel 3.19).



Görsel 3.19: HTML5 tablo yapısı

3.3.10.1. <table> Etiketi

Tablolar, web sayfası içinde <table> etiketi kullanılarak oluşturulur. <table> etiketi, </table> kapama etiketi ile birlikte kullanılır. HTML5 ile tablo oluşturmak için gerekli temel ve gelişmiş özelliklere ait etiketler Tablo 3.7'de verilmiştir.

Tablo 3.7: Tablo Oluşturmak İçin Kullanılan HTML Komutları

Etiket	Açıklama
<table>	HTML sayfalarında tablo oluşturur. Tablo ögeleri bu etiketler arasında yer alır.
<tr>	Tablo satırlarını oluşturur.
<td>	Tablo sütunlarını oluşturur. Hücrelere veri eklemek için kullanılır.
<th>	Tablo sütun başlıklarını oluşturur. Varsayılan yazı tipi kalındır.
<caption>	Tablo kenarlıkları dışında başlık oluşturur.
<col>	Tablo sütunlarını biçimlendirir.
<colgroup>	Hücre gruplarını temsil eder.
<thead>	<th> dâhil tüm tablo başlık bölümünü kapsar.
<tbody>	Tablonun <tr>, <td> asıl içeriklerinin şekillendiği bölümdür.
<tfoot>	Tablonun alt kısmında yer alan son satırıdır.
<colspan>	Hücreleri sütun olarak birleştirir.
<rowspan>	Hücreleri satır olarak birleştirir.

3.3.10.2. Border Özelliği

Border özelliği, tabloya çerçeve kalınlığı uygular <table> etiketi ile border özelliği tanımlanmamış ise tablo çerçevesi görüntülenmez. Tablo çerçevesinin görüntülenmesi için border değeri en az bir (1) olmalıdır (Tablo 3.8).

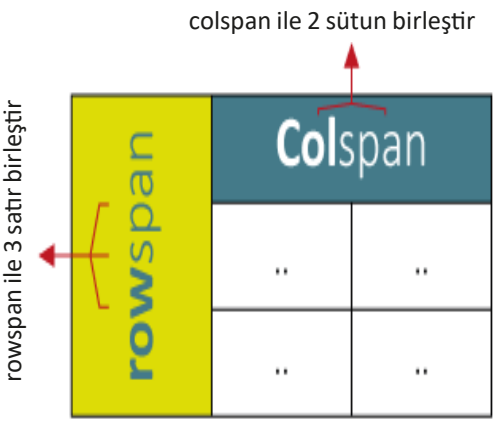
Tablo 3.8: Border Özelliğinin Kullanımı

TABLE BORDER Kullanımı	Açıklama
<table border="1">	Çerçeve kalınlığı 1 olan tablo tanımlama.
<table border="0">	Çerçevesiz tablo tanımlama.
<table>	Çerçevesiz tablo tanımlama.

3.3.10.3. Rowspan ve Colspan Özellikleri

Web tablolarında satır ve sütunlar içinde birleştirme, ayırma vb. işlemleri uygulamak için özellikler vardır. **Rowspan** özelliği satırları birleştirmek için, **colspan** özelliği sütunları birleştirmek için kullanılır (Görsel 3.20).

```
<table border="1" cellpadding="5">
 <tr>
  <th rowspan="3"> .. </th>
  <th colspan="2"> .. </th>
 </tr>
 <tr>
  <td> .. </td>
  <td> .. </td>
 </tr>
 <tr>
  <td> .. </td>
  <td> .. </td>
 </tr>
</table>
```



Görsel 3.20: Hücre birleştirme işlemleri

Sıra Sizde

Aşağıdaki tabloyu <body> etiketi içine rowspan özelliğini kullanarak oluşturunuz ve tarayıcıda Görsel 3.21'deki gibi görüntüleyiniz.

```
<table border="1" cellpadding="5">
 <tr>
  <th> Hücre 1 </th>
  <th rowspan="2"> Hücre 2</th>
  <th> Hücre 3 </th>
 </tr>
 <tr>
  <td> Hücre 4 </td>
  <td> Hücre 5 </td>
 </tr>
</table>
```

Hücre 1	Hücre 2	Hücre 3
Hücre 4		Hücre 5

Görsel 3.21: rowspan kullanımı

Sıra Sizde

Aşağıdaki tabloyu <body> etiketi içine colspan özelliğini kullanarak oluşturunuz ve tarayıcıda Görsel 3.22'deki gibi görüntüleyiniz.

```
<table border="1" cellpadding="5">
 <tr>
  <th> Hücre 1 </th>
 <th colspan="2"> Hücre 2</th>
 </tr>
 <tr>
  <td> Hücre 4 </td>
  <td> Hücre 5 </td>
  <td> Hücre 6 </td>
 </tr>
</table>
```

Hücre 1	Hücre 2	
Hücre 4	Hücre 5	Hücre 6

Görsel 3.22: colspan kullanımı

8. Uygulama

Anlamsal HTML tablo komutlarını kullanarak tablo oluşturma işlemini yönergeler doğrultusunda gerçekleştiriniz.

1. Adım: Temel HTML yapısını metin düzenleme programında oluşturunuz.
2. Adım: <table> blokunu <body> etiketi içine yazınız.
3. Adım: <caption> .. </caption> elemanını oluşturunuz.
4. Adım: <thead> .. </thead> elemanını oluşturunuz.
5. Adım: <tbody> .. </tbody> elemanını oluşturunuz.
6. Adım: <tfoot> .. </tfoot> elemanını oluşturunuz.
7. Adım: Oluşturduğunuz elemanların içine <th>, <tr>, <td> etiketlerini kullanarak tablonun satır ve sütunlarını oluşturunuz.
Adım 8: HTML belgesini "tablo.html" olarak kaydediniz ve tarayıcıda Görsel 3.23'teki gibi görüntüleyiniz.

```
<table  border="1" cellpadding="5">
 <caption>Yazılı Notları</caption>
  <thead>
   <tr>
    <th>İsim</th>
    <th>Not</th>
   </tr>
  </thead>
  <tbody>
   <tr>
    <td>Halide Edip</td>
    <td>100</td>
   </tr>
   <tr>
    <td>Sabiha</td>
    <td>90</td>
   </tr>
   <tr>
    <td>Gökçen</td>
    <td>80</td>
   </tr>
  </tbody>
```

Yazılı Notları

İsim	Not
Halide Edip	100
Sabiha	90
Gökçen	80
Ortalama	95

Görsel 3.23: Anlamsal HTML tablo komutları

```
  <tfoot>
   <tr>
    <td>Ortalama</td>
    <td>95</td>
   </tr>
  </tfoot>
</table>
```

Sıra Sizde

Sütun başlıkları haftanın günleri, satır başlıkları ders saatleri olan kendi ders programı tablonuzu HTML5 etiketlerini kullanarak oluşturunuz.

3.3.11. Metin Biçimlendirmesinde Kullanılan Diğer Etiketler

HTML5'te metin biçimlendirme (renk, yazı karakteri, yazı boyutu değiştirme, hizalama vb.) işlemleri CSS ile yapılır fakat bazı temel metin biçimlendirme (kalınlaştırma, altını çizme, eğik yazma vb.) işlemleri için ise özel etiketler kullanılmaktadır. Metin hizalama işlemleri için de CSS kullanılabileceği gibi tablolar ve listeleme komutları ile temel seviyede hizalama işlemleri gerçekleştirilir. Bu işlemler dışında metin biçimlendirmesi için kullanılan bazı etiketler Uygulama 9'da verilmiştir.

9. Uygulama

Metin biçimlendirmesinde kullanılan diğer etiketleri yönergeler doğrultusunda gerçekleştiriniz.

1. Adım: Temel HTML yapısını metin düzenleme programında oluşturunuz.

2. Adım: Aşağıdaki diğer metin biçimlendirme etiketlerini <body> etiketi içine yazınız.

```
<p><abbr title= "Hyper Text Markup Language">HTML</abbr></p>
<p><bdo dir="ltr">Bu metin soldan sağa doğru. </bdo></p>
<p><bdo dir="rtl">Bu metin sağdan sola doğru. </bdo></p>
<p></p><cite> Web Tabanlı Uygulama Geliştirme</cite> Komisyon, 2021.</p>
<p></p>HTML <code>table</code> etiketi tablo oluşturur.
<p><dfn>HTML</dfn> standart metin işaretleme dilidir. </p>
<p><dfn title="Hyper Text Markup Language">HTML</dfn> standart metin işaretleme
dilidir. </p>
<p><dfn><abbr title="HyperText Markup Language">HTML</abbr></dfn>standart me-
tin işaretleme dilidir. </p>
<p><dfn id="htmlreferans">HTML</dfn> standart metin işaretleme dilidir. </p>
<p>Kopyalama klavye kısa yolu : <kbd>Ctrl</kbd>+<kbd>C</kbd></p>
<p>Bakınız, şair vatanı ne güzel tarif ediyor:</br><q>Bayrakları bayrak yapan üstündeki
kandır.</br>Toprak eğer uğrunda ölen varsa vatandır. </q></p>
<p><samp>Örnek bilgisayar program çıktısı</samp></p>
<p>İlk ders <time>09:00</time>'da başlıyor. </p>
<p>Matematik Kuralı:</p>
<var>x</var>(<var>y</var>+<var>z</var>)=<var>xy</var>+<var>xz</var></p>
<p>
  <details>
    <summary>Göster / Gizle</summary>
    <p>Başlık tıklandığında içeriğe ulaşabilirsiniz</p>
  </details>  </p>
```

3. Adım: Her paragrafta oluşturduğunuz HTML belgesini tarayıcıda görüntüleyiniz.

3.4. Yerleşim Elemanları

HTML etiketlerinin etiket türüne bağlı olarak varsayılan iki tip görüntüleme değeri vardır. Bunlar, blok düzeyinde etiketler ve satır içi etiketlerdir.

Blok Düzeyinde Etiketler

Blok düzeyinde etiketler, her zaman yeni bir satırdan başlar. Blok etiketler, mevcut tam genişliği kaplar. Blok düzeyinde etiketlerin bir alt ve bir üst bir kenarı vardır. Blok etiketler, kapama etiketleri ile kullanılır. ** ** etiketi blok düzeyinde bir etikettir.

Satır İçi Etiketler

Satır içi etiketler yeni bir satırdan başlamaz, yalnızca gerektiği kadar genişlik kaplar. Satır içi etiketlerin blok düzeyi etiketlerinde olan alt ve üst kenar bulunmaz. **** etiketi, satır içi etikettir.

3.4.1. Etiketi

** etiketi,** yerleşim elemanı olarak kullanılan ve diğer HTML etiketlerini içine alabilen bir kutu olarak kullanılır. etiketi ile style, class, id özellikleri kullanılır. Bu özellikleri kullanmak zorunlu değildir. HTML belgesindeki verilerin konumlarını (yukarı, aşağı, sağa ve sola) CSS yardımı ile hizalar. CSS ile kullanıldığında etiketi, içerik bloklarına stil uygulamak için kullanılır. Metinler, resimler ve videolar div içine yerleştirilir. etiketi tablolara benzerlik gösterir fakat tablolar katı, esnek olmayan ve ızgara şeklindedir. etiketleri (Görsel 3.24) ile oluşturulan CSS tabanlı tasarımlar ise daha esnek, akışkan ve genişletilebilir yapıdadır.

10. Uygulama

HTML5 belge şablonunu etiketini kullanarak yönergeler doğrultusunda gerçekleştiriniz.

1. Adım: Temel HTML yapısını metin düzenleme programında oluşturunuz.
2. Adım: etiketinin class özelliğini kullanarak header, nav, section, aside ve footer özelliğine sahip div bloklarını oluşturunuz.
3. Adım: etiketinin class özelliği section olan div bloku içine sırasıyla header2, article, ve footer2 özelliğine sahip div bloklarını oluşturunuz.
4. Adım: blokları ile oluşturduğunuz HTML belgesini kaydedip tarayıcıda görüntüleyiniz.

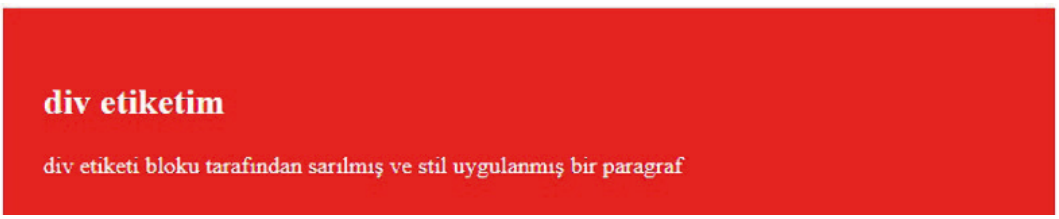
```
<html>
<head>
        <title>DIV ile HTML Şablonu</title>
</head>
<body>
```

```
<div class="header"> </div>

<div class="nav"> </div>
<div class="section">
        <div class="header2"> </div>
        <div class="article"> </div>
        <div class="footer2"> </div>
</div>

<div class="aside"> </div>
<div class="footer"></div>

</body>
</html>
```

Sıra Sizde

 etiketini kullanarak farklı renklerde div blokları oluşturunuz.

```
<div style="background-color:red;color:white;padding:30px;">
 <h2>div etiketim</h2>
 <p>div etiketi bloku tarafından sarılmış ve stil uygulanmış bir paragraf</p>
</div>
```



Görsel 3.24: etiketi

Not

 yapısı, oluşturulması planlanan site taslağına göre şekillendirilir.

Anlamsal etiketler ile yukarıdaki açma ve kapatma karışıklığı çözülür. Bu şablon HTML belgesinin iskeletini oluşturur. HTML şablonu CSS kullanarak şekillendirilir.

3.4.2. Etiketi

 etiketi, metin veya HTML belgesinin bir bölümünü işaretlemek için kullanılan satır içi kapsayıcıdır. Yapısal ve görsel olarak bir katman oluşturur. etiketi ile style, class, id özellikleri kullanılır. Bu özellikleri kullanmak zorunlu değildir. CSS ile kullanıldığında etiketi paragraf

metinlerine stil uygulamak için kullanılır. etiketi, etiketinden farklı olarak satır içi eleman olduğundan paragraf içinde kullanıldığında ilgili paragrafı keserek alt satıra geçmez (Görsel 3.25).

```
<p> Bu bir paragraf <span style="color:green;font-weight:bold;"> bu metne span etiketi kullanılarak yeşil renk ve kalın font stil uygulandı</span> paragrafın devamı <span style="color:blue;font-weight:bold;"> bu metne span etiketi kullanılarak mavi renk ve kalın font stil uygulandı</span> paragrafın sonu</p>
```

Bu bir paragraf **bu metne span etiketi kullanılarak yeşil renk ve kalın font stil uygulandı** paragrafın devamı **bu metne span etiketi kullanılarak mavi renk ve kalın font stil uygulandı** paragrafın sonu

Görsel 3.25: etiketi

Sıra Sizde

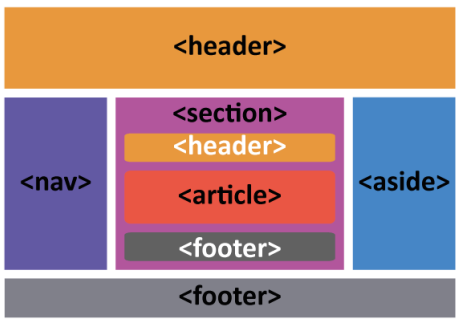
Görsel 3.24'te kullanılan HTML belgesinde etiketi yerine yazarak, Görsel 3.25'te kullanılan HTML belgesinde etiketi yerine de yazarak farklı HMTL belgesi olarak kaydediniz. HTML belgelerindeki değişiklikleri gözlemleyiniz.

3.4.3. HTML5 Tasarım Şablonu

HTML5 ile gelen en önemli özellik, anlamsal etiketlerdir (Tablo 3.9). Anlamsal etiketler kullanılarak HTML5 belge şablonu oluşturulur (Görsel 3.26). HTML5'in yapısı ile kodlamanın devamıdır.

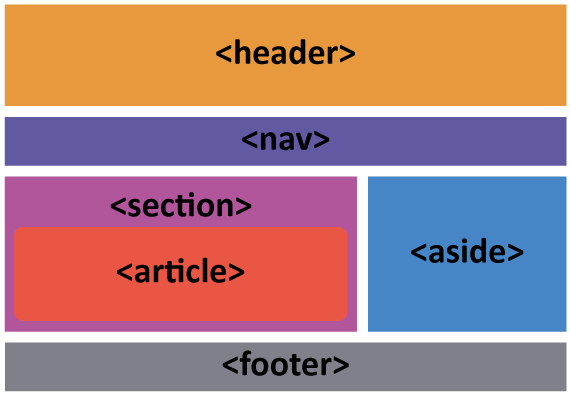
Tablo 3.9: HTML5'te Web Belgesinin Ana Bölümlerini Tanımlamak İçin Kullanılabilecek Anlamsal Etiketler

Etiket	Açıklama
<header>	HTML belgesi veya bir bölüm için bir başlık tanımlar.
<nav>	HTML belgesindeki menü bağlantısını tanımlar.
<section>	HTML belgesindeki bir bölümü tanımlar.
<article>	HTML belgesinde bağımsız, kendi kendine yeten bir içerik tanımlar.
<aside>	HTML belge içeriğinin yanı sıra içeriği tanımlayan ek içerik tanımlar.
<footer>	HTML belgesi veya bir bölüm için bir altbilgi tanımlar.
<details>	Kullanıcının talebi üzerine açıp kapatabileceği ek ayrıntıları tanımlar.
<summary>	<details> etiket bloku için bir başlık tanımlar.



Görsel 3.26: HTML5 belge şablonu

HTML5 tasarım şablonu ve anlamsal etiketlerin HTML belgesindeki konumları Görsel 3.27'de görülmektedir. Site tasarımına uygun olarak anlamsal etiketler yer değiştirebilir.



Görsel 3.27: Anlamsal etiketler ve konumları

3.4.3.1. <header> Etiketi

<header> etiketi, HTML belgesi şablonunun üst kısmını oluşturan yerleşim etiketidir. <header> bloku bir veya daha fazla <h1>-<h6> başlık elemanı, simge veya logo ile yazar bilgilerini içerir. <header> bloku sadece sayfa başlığı bölümünde kullanılmaz, makalelerdeki başlıkları belirlemek için de kullanılır.

```
<header>
        <h1>İlk Kişisel Web Sayfam</h1>
        <p>Header Yazarı : Selim</p>
</header>
```

3.4.3.2. <nav> Etiketi

<nav> etiketi, HTML belgesinde gezinme bağlantılarının tanımı için kullanılır. </nav> kapama etiketi ile kullanılır. Navigasyon kelimesinin kısaltmasıdır. <nav> bloku sadece sayfalar arası gezinme bağlantı blokudur. HTML belgesindeki tüm bağlantılar <nav> etiket bloku içinde yer almaz. Özel gereksinimli kullanıcılar için ekran okuyucu gibi tarayıcılar bu blokta yer alan içeriğin ihmal edilip edilmeyeceğine karar verir.

```
<nav>
        <a href="/html/">HTML</a> |
        <a href="/css/">CSS</a> |
        <a href="/js/">JavaScript</a> |
        <a href="/dotnet/">.NET Core</a>
</nav>
```

3.4.3.3. <section> Etiketi

<section> etiketi, HTML belgesinde bölüm tanımlar. **</section>** kapama etiketi ile kullanılır. <section> blokunda başlık ve bir makale yer alabilir (Görsel 3.28).

```
<section>
        <h2>HTML5</h2>
        <p>HTML5, HTML standardının en son sürümüdür.</p>
</section>

<section>
        <h2>CSS</h2>
        <p>CSS, Cascading Style Sheets (Basamaklı Biçim Sayfaları) ifadesinden oluşturulmuş
bir kısaltmadır. </p>
</section>
```

Görsel 3.28: <section> etiketi kullanımı

3.4.3.4. <article> Etiketi

<article> etiketi, web sayfasında kullanılacak haber, makale, forum içeriği, blog içeriği, müzik vb. içerikleri tanımlamak ve yerleştirmek için kullanılır. </article> kapama etiketi ile kullanılır.

```
<article>
        <h1>article kullanımı</h1>
        <p>article etiketi, web sayfasında kullanılacak haber, makale vb. metinleri tanımlamak
ve yerleştirmek için kullanılır. </p>
</article>
```

3.4.3.5. <footer> Etiketi

<footer> etiketi, HTML belgesinin veya bir bölümün alt bilgisini tanımlayan bloktur. <footer> bloku yazar bilgisi, telif hakkı bilgisi, iletişim bilgisi, site haritası üst bağlantılara dön, ilgili belgeler vb. bilgilerini içerir. Bir HTML belgesinde bir veya birden fazla <footer> bloku kullanılabilir.

```
<footer>
        <p>Yazar : Azra <a href="mailto:yazar@meb.gov.tr">e-Posta</a></p>
</footer>
```

3.4.3.6. <aside> Etiketi

<aside> etiketi, ana içerikten farklı olarak yer alması istenen içeriğin yerleştirileceği alanı tanımlar. HTML belgesinde sidebar, kenar çubuğudur. Web sayfasında sağ veya solda yer alır. Bazı web sayfalarında reklam alanı olarak kullanılmaktadır. İçerikten ayrı kullanılmak istenen reklam, bilgi notu gibi ögeler sidebar alanında yani <aside> blokunda yer alır.

```
<h1>EBA'da neler var?</h1>
<p>Eğitim-öğretim sürecinde bilişim teknolojisi donanımları vasıtasıyla etkin materyaller kullanmak amacıyla YEĞİTEK tarafından tasarlanan .. </p>
<aside>
        <h4>EBA Dükkan</h4>
        <p>EBA tarafından geliştirilen android tabanlı içerik, uygulama ve oyunlara EBA Dükkândan erişebilirsiniz.
        </p>
</aside>
```

3.4.3.7. <figure> ve <figcaption> Etiketi

HTML belgesinde resim, fotoğraf, şekil vb. ögeleri işaretlemek için **<figure>** bloku kullanılır. **<figcaption>** etiketi figure blokunda kullanılan öge için alt yazı tanımlar (Görsel 3.29).

```
<figure>
        <img src="resim.jpg">
        <figcaption>Şekil.1-Figure Uygulaması</figcaption>
</figure>
```

Görsel 3.29: <figcaption> etiketi kullanımı

3.4.3.8. <hgroup> Etiketi

<hgroup> etiketi, belgenin yapısı gereği birden çok başlığı bir başlık gibi gruplamak için kullanılır.

```
<header>
        <hgroup>
                <h1>İlk Kişisel Web Sayfa Başlığım</h1>
                <h2>Kişisel Web Sayfası Alt Başlığım </h2>
        </hgroup>
</header>
```

HTML5'te, web belgesinin ana bölümlerini tanımlamak görsel olarak değil anlamsal bir değere sahiptir. Arama motorları siteyi indeksleme için geldiğinde HTML belgesinde anlamsal etiketlerin içeriklerini kullanır. HTML5 ana anlamsal etiketler ile web sayfasının iskeleti oluşturulur, görsel olarak sitenin geliştirilmesi CSS aracılığıyla sağlanır.

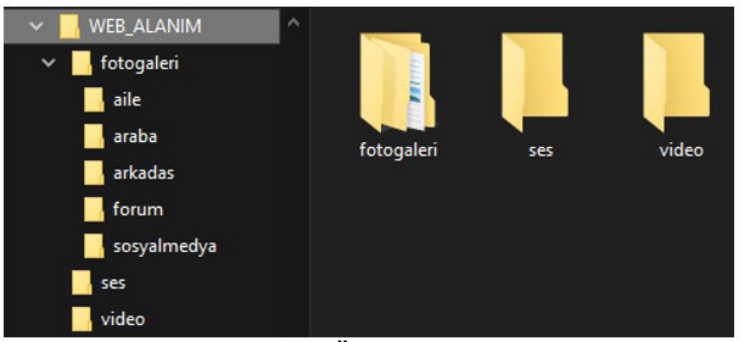
Sıra Sizde

Anlamsal etiketler kullanarak aşağıda kodları verilen HTML5 belgesini oluşturunuz ve tarayıcıda görüntüleyiniz.

```
<!DOCTYPE html>
<html>
 <head>
        <meta charset="UTF-8">
        <title>İlk HTML5 Belgesi ve Anlamsal Etiketler</title>
 </head>
<body>

        <nav>
                <h2>MENÜ</h2>
                <ul>
                        <li><a href="index.html">anasayfa</a></li>
                        <li><a href="hakkimda.html">hakkımda</a></li>
                        <li><a href="iletisim.html">iletişim</a></li>
                </ul>
        </nav>

 <p>Web Sayfası İçeriği
        <article>
                <section>
                        Okuldan Haberler
                </section>
        </article>
        <article>
                <section>
                        Sağlık Haberleri
                </section>
        </article>

        <figure>
                <img src="logo.png" alt="logom" />
                <figcaption>
                        <a href="url">Benim Logom</a>, daha fazla bilgi için
                </figcaption>
        </figure>

        <footer>
                <h3 id="yazar">HTML5 Yazarı : Zehra</h3>
        </footer>
 </p>
</body>
</html>
```

3.5. Medya (Resim, Video ve Ses) Elemanları

Web sayfaları sadece yazılardan oluşmaz. Bir web sayfasını etkileyici hâle getirmek için fotoğraf, resim, video ve ses gibi donatılar kullanılır. Kullanılacak her içerik elemanı, barındırma alanında bir alan kaplar. Web sayfasında kullanılacak resim, video ve ses gibi dosyaların bulunacağı klasör yapısı, kodlamaya geçmeden önce oluşturulur (Görsel 3.30). HTML belgesi içeriğinde kullanılacak donatılar URL yazarak belgeye eklenir. HTML belgesi içeriğinde kullanılacak resim, video, ses gibi dosyalar kategorilerden oluşuyor ise bu kategoriler için alt klasörler oluşturulması, web barındırma alanının düzenli olmasını sağlar. Klasör isimlerinde Türkçe karakter kullanılmaz.

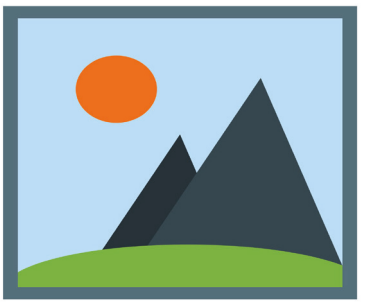


Görsel 3.30: Örnek klasör yapısı

3.5.1. HTML5 <img> Kullanımı

HTML belgesine resim (görsel) eklemek için **<img> etiketi** kullanılır. "img", "görsel" anlamında kullanılan "image" kelimesinin kısaltmasıdır. HTML belgesine eklenecek görsel gif, jpg ve png formatında olmalıdır. <img> </img> şeklinde kullanım doğru değildir, <img> etiketi boş etikettir. <img> etiketinin kapama etiketi bulunmaz bunun yerine etiketin sonunda / (eğik çizgi) kullanılır. Resim eklemek için <img> etiketinin src özelliği kullanılır. **src özelliği** HTML belgesine eklenecek resmin yolunu gösterir (Görsel 3.31).

```
<img src="proje.jpg" />
```



Görsel 3.31: <img> kullanımı

Web alanında resim dosyaları HTML belgesi ile aynı klasörde veya alt klasörlerde yer alır. Resim dosyasını sadece dosya adı ile çağırmak için HTML belgesi ile resim dosyası aynı klasör içinde yer almalıdır. Alt veya üst klasörde bulunan bir resme ulaşmak için resim yolu doğru yazılmalıdır.

```
<img src="resimgaleri/arkadas/01.jpg" />
```

Görsel 3.32: src kullanımı

Alt klasörlere ulaşmak için dosya yolu ve dosya adı yazılır (Görsel 3.32). Üst klasördeki bir dosyaya ulaşmak için ../ kullanılır. Ulaşılacak üst klasör sayısı kadar ../ ifadesi tekrar edilir (Tablo 3.10).

Tablo 3.10: <img> Etiketi src Özelliği Kullanımı

<img> src kullanımı	HTML belgesi ile "resim.jpg" dosyası
<img src="resim.jpg">	Aynı klasörde
<img src="galeri/resim.jpg">	Bir alt klasörde
<img src="galeri/arkadas/resim.jpg">	İki alt klasörde
<img src="../resim.jpg">	Bir üst klasörde
<img src="../../resim.jpg">	İki üst klasörde

3.5.1.1. <img> Etiketinin Özellikleri

<img> etiketi ile kullanılabilecek özellikler Tablo 3.11'de verilmiştir.

Tablo 3.11: <img> Etiketinin Özellikleri

Özellik	Alacağı Değer	Açıklama
height	piksel	Resmin yüksekliğini belirtir.
ismap	ismap	Tıklanabilir alanlara sahip bir resim olduğunu belirtir.
src	URL	Resmin URL'sini ifade eder.
title	metin	Fare işaretçisi resmin üzerine geldiğinde bir ipucu görüntüler.
usemap	harita ismi	Tıklanabilir alanlara sahip resim haritası oluşturur.
width	piksel	Resmin genişliğini belirtir.
sizes	piksel	Görüntünün medya ortamına göre olması gereken boyutlarını tanımlamamızı sağlar.

Sıra Sizde

Farklı klasörlerde bulunan görseller ile <img> etiketinin özelliklerini kullanarak HTML belgesi oluşturunuz ve tarayıcıda görüntüleyiniz.

Görsel 3.33: Border kullanımı

```
<img src="resim.jpg" width="400" height="200" border="6" title="Öğrenci"/>
```

Görsel 3.33'te CSS kullanılmadan <img> etiketi özellikleri ile yükseklik, genişlik, çerçeve kalınlığı ve alternatif metin özellikleri uygulanmıştır. Özellikler yazılırken aralarında en az bir boşluk bırakılır. <img> ögesine CSS aracılığıyla stil uygulanır. CSS kullanılarak kenarlık eklenir, resim opaklığı ayarlanır, köşeler yuvarlatılır, filtre uygulanır.

Aynı görüntü için farklı kaynakların bir listesini tanımlayarak tarayıcının, kullanılacak en iyi olanı seçmesi **srcset özelliği** kullanılarak sağlanır. Aşağıdaki uygulamada görüntü alanına uygun varsayılan görseli yükleyecek tarayıcının karar vermesini kolaylaştırmak için srcset özelliğinde görseller genişlikleri ile birlikte kullanılmıştır.

Sıra Sizde

srcset ve boyut özelliklerini kullanarak görsel ekleyiniz, oluşturduğunuz HTML belgesini tarayıcıda görüntüleyiniz.

```
<!—srcset ve boyut özelliklerinin kullanımı -->

<img src="/resimler/ogrenci.jpg"
title="Eğlenerek Öğrenme Etkinliği Görsel"
srcset=" /resimler/ogrenci-360.jpg 360w,
         /resimler/ogrenci-720.jpg 720w,
         /resimler/ogrenci-1440.jpg 1440w"
sizes="(min-width: 800px) 720px, 360px"/>
```

Eğlenerek Öğrenme Etkinliği Görsel

Görsel 3.34: Görsel tarayıcı tarafından görüntülenmiyor

Tarayıcı, görseli yüklerken veya görseli bulamadığında <img> etiketinin title özelliğindeki açıklama, tarayıcıda görüntülenir (Görsel 3.34).

3.5.2. HTML5 <video> Kullanımı

<video> etiketi, HTML belgesine video eklemek için kullanılır. Eski sürümlerde video eklemek için eklenti yüklenmesi gerekir. HTML5 ile video eklemek, resim eklemek kadar basittir. <video> etiketi, </video> kapama etiketi ile birlikte kullanılır. Kullanıcının video oynatabilmesi, duraklatabilmesi gibi temel denetimler aşağıdaki HTML kodu ile yapılır.

```
<video src="video.mp4" controls="controls"> </video>
```

3.5.2.1. <video> Etiketi

<video> etiketi, videonun nasıl oynatılacağını belirten özelliklere sahiptir.

```
<video width="600" height="500" controls="controls">
        <source src="film.mp4" type="video/mp4">
        <source src="film.ogg" type="video/ogg">
        <source src="film.webm" type="video/webm">
Tarayıcınız video ögesini desteklemiyor.
</video>
```

Source etiketine **type** özelliğini eklemek web site hızını arttırır. Type özelliği belirtilmediğinde tarayıcı oynatabileceği videoyu buluncaya kadar her video dosyasını yükler.

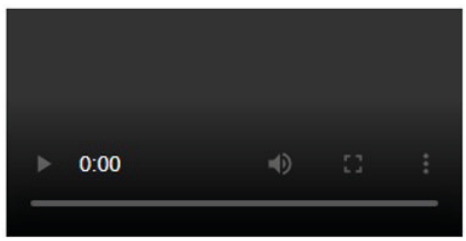
Sıra Sizde

İçeriğinde video bulunan web sitelerinin kaynak kodlarını inceleyerek video ekleme yöntemlerini ve kullanılan HTML etiketi özelliklerini arkadaşlarınız ile paylaşınız.

3.5.2.2. Controls Özelliği

<video controls> özelliği videonun oynatılabilmesi için gereken oynat (play), durdur (pause), ses ayarı (volume) ve tam ekran (full screen) ögelerini içeren medya oynatıcı aracını oluşturur. Kaynak dosya belirtilmez ise sadece medya oynatıcı oluşturulur ve sesiz modda görünür (Görsel 3.35). Kaynak belirtildiğinde tam ekran modu da aktif olur.

```
<video controls>
        <source src=" " type="video/mp4">
Tarayıcınız video ögesini desteklemiyor.
</video>
```



Görsel 3.35: HTML5 medya oynatıcı temel ögeler

3.5.2.3. Src Özelliği

Src özelliği, açılan HTML belgesinde video dosyasının sunucudaki (localhost veya web) kaynağını ifade eder. Src özelliği belirtilmediğinde sadece video oynatıcı oluşturulur.

```
<video controls>
        <source src="video.mp4" type="video/mp4">
</video>
```

3.5.2.4. Height ve Witdh Özellikleri

Yükseklik (height) ve genişlik (width) özelliklerini piksel (pixel) olarak ayarlamak için kullanılır. Height ve width özellikleri belirtilmediğinde medya oynatıcı, standart medya aracı boyutunda oluşturulur.

```
<video controls width="300px" height="200px">
        <source src="video.mp4" type="video/mp4">
</video>
```

3.5.2.5. Autoplay Özelliği

Autoplay özelliği, HTML belgesi aktif olduğunda video dosyası medya oynatıcısı tarafından otomatik olarak oynatılmaya başlanır. Mobil tarayıcılarda bu özellik pasif olarak gelir.

```
<video controls autoplay="autoplay">
        <source src="video.mp4" type="video/mp4">
</video>
```

3.5.2.6. Loop Özelliği

Loop özelliği, video dosyası medya aracından oynat butonu ile oynatılmaya başlar. Kullanıcı tarafından durdurulmadığı sürece video dosyası oynatılmaya devam eder.

```
<video controls loop>
        <source src="video.mp4" type="video/mp4">
</video>
```

3.5.2.7. Muted Özelliği

Muted özelliği ile açılan HTML belgesinde video dosyasının medya oynatıcıdaki varsayılan durumu sessiz (muted) olarak ayarlanır.

```
<video controls muted>
        <source src="video.mp4" type="video/mp4">
</video>
```

3.5.2.8. Poster Özelliği

Poster özelliği, video oynatıcı arka planında (background) **src** özelliği ile belirtilen video görseli dışında bir görsel görüntülenmesini sağlar (Görsel 3.36). Poster özelliği kullanılmaz ise medya oynatıcıda video dosyasının ilk karesi arka planı oluşturur.

```
<video controls poster="arkaplan.jpg" >
   <source src="video.mp4" type="video/mp4">
</video>
```

Görsel 3.36: Poster özelliği kullanımı

3.5.2.9. Preload Özelliği

Preload özelliği ile açılan HTML belgesinde video dosyasının önyüklemesinin yapılıp yapılmayacağı kararı verilir. Önyükleme yapılacak ise preload=auto, yapılamayacak ise preload=none olarak bildirilir.

```
<video controls preload="none | auto | metadata">
            <source src="video.mp4" type="video/mp4">
</video>
```

HTML5 <video> ögesine CSS aracılığıyla stil uygulama olanağı sağlar. CSS kullanarak kenarlık eklenir, video opaklığı ayarlanır, filtre uygulanır ve videoda 3D dönüşüm yapılır. HTML5 ayrıca JavaSript tarafından video oynatmayı kontrol etmek için kullanılabilecek yöntemler, özellikler ve olaylar sağlar.

11. Uygulama

<video> etiketini kullanarak web sayfasına video ekleme işlemini yönergeler doğrultusunda gerçekleştiriniz.

1. Adım: Temel HTML yapısını metin düzenleme programında oluşturunuz.
2. Adım: <body> etiketi içine <video> blokunu ekleyiniz.
3. Adım: "http://meb.ai/KR71t6" URL adresini kullanarak HTML belgenize video ekleyiniz.

```
<source src=" http://meb.ai/KR71t6" type="video/mp4">
```

4. Adım: HTML belgenizi tarayıcıda görüntüleyiniz.
5. Adım: Videoyu oynatınız, video görüntülenmiyorsa kodlarınızı gözden geçiriniz.
6. Adım: <video> blokunun altına yeni bir <video> bloku oluşturunuz.
7. Adım: "http://meb.ai/UeSk1O" URL adresini kullanarak HTML belgenize video ekleyiniz.

```
<source src="http://meb.ai/UeSk1O" type="video/mp4">
```

Adım 8: Adım 3 ve 4 basamaklarını farklı videolar için tekrarlayınız.

3.5.3. HTML5 <audio> Kullanımı

<audio> etiketi, HTML belgesine ses oynatıcı (audio player) eklemek için kullanılır (Görsel 3.37). Audio, İngilizcede ses anlamına gelir. Eski sürümlerde ses eklemek için eklenti yüklemek gerekir. HTML5 ile ses eklemek, resim eklemek kadar basittir. <audio> etiketi, </audio> kapama etiketi ile birlikte kullanılır. Kullanıcının müzik dosyasını oynatabilmesi, duraklatabilmesi gibi temel denetimler <audio> HTML bloku ile yapılır.

Görsel 3.37: Ses oynatıcısı

3.5.3.1. <audio> Etiketi

<audio> etiketi, ses oynatıcının nasıl oynatılacağını belirten özelliklere sahiptir.

```
<audio controls="controls">
        <source src="ukulele.mp3" type="video/mpeg">
        <source src="ukulele.ogg" type="video/ogg">
Tarayıcınız ses (audio) ögesini desteklemiyor.
</audio>
```

3.5.3.2. Src Özelliği

Src özelliği, açılan HTML belgesinde ses dosyasının sunucudaki (localhost veya web) kaynağını ifade eder. Src özelliği belirtilmediğinde sadece ses oynatıcı oluşturulur.

```
<audio controls>
        <source src="audio.mp3" type="audio/mp3">
</audio>
```

3.5.3.3. Autoplay Özelliği

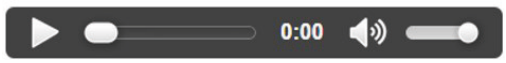
Autoplay özelliği, HTML belgesi aktif olduğunda ses dosyası, ses oynatıcısı tarafından otomatik olarak oynatılmaya başlanır. Mobil tarayıcılarda bu özellik pasif olarak gelir. Web sayfaları açılır açılmaz ses dosyalarının kullanıcı isteği dışında oynatılması rahatsız edici bir özelliktir.

```
<audio controls autoplay="autoplay">
        <source src="audio.mp3" type="audio/mp3">
</audio>
```

3.5.3.4. Controls Özelliği

Controls özelliği, ses dosyasının oynatılabilmesi için gereken "oynat, durdur ve ses ayarı" ögelerini içeren ses oynatıcı aracını oluşturur. Kaynak dosya belirtilmez ise sadece ses oynatıcı oluşturulur ve sesiz modda görünür (Görsel 3.38).

```
<audio controls>
        <source src="audio.mp3" type="audio/mpeg">
Tarayıcınız ses ögesini desteklemiyor.
</audio>
```



Görsel 3.38: HTML5 ses oynatıcısı temel ögeler

3.5.3.5. Loop Özelliği

Loop özelliği ile ses dosyası medya aracından oynat butonuna basarak oynatılmaya başlar. Kullanıcı tarafından durdurulmadığı sürece ses dosyası oynatılmaya devam eder.

```
<audio controls loop>
        <source src="audio.mp3" type="audio/mp3">
</audio>
```

3.5.3.6. Muted Özelliği

Muted özelliği ile açılan HTML belgesinde ses dosyasının medya oynatıcıdaki varsayılan durumu sessiz (muted) olarak ayarlanır.

```
<audio controls muted>
        <source src="audio.mp3" type="audio/mp3">
</audio>
```

3.5.3.7. Preload Özelliği

preload özelliği ile açılan HTML belgesinde ses dosyasının önyüklemesinin yapılıp yapılmayacağı kararı verilir. Önyükleme yapılacak ise preload=auto, yapılamayacak ise preload=none olarak bildirilir.

```
<audio controls preload="none | auto | metadata">
        <source src="audio.mp3" type="audio/mp3">
</audio>
```

HTML5; JavaSript tarafından ses dosyası oynatmayı kontrol etmek için kullanılabilecek yöntemler, özellikler ve olaylar sağlar.

12. Uygulama

<audio> etiketini kullanarak web sayfasına ses ekleme işlemini yönergeler doğrultusunda gerçekleştiriniz.

1. Adım: Temel HTML yapısını metin düzenleme programında oluşturunuz.
2. Adım: <body> etiketi içine <audio> blokunu ekleyiniz.
3. Adım: "http://meb.ai/fN7MRY" URL adresini kullanarak HTML belgenize ses ekleyiniz.

```
<source src=" http://meb.ai/fN7MRY" type="audio/mp3">
```

4. Adım: HTML belgenizi tarayıcıda görüntüleyiniz.
5. Adım: Videoyu oynatınız, ses görüntülenmiyorsa kodlarınızı gözden geçiriniz.
6. Adım: <audio> blokunun altına yeni bir <audio> bloku oluşturunuz.
7. Adım: "http://meb.ai/U4S81U" URL adresini kullanarak HTML belgenize ses ekleyiniz.

```
<source src=" http://meb.ai/U4S81U" type="audio/mp3">
```

Adım 8: 3 ve 4. adımları farklı sesler için tekrarlayınız.

Sıra Sizde

İçeriğinde video bulunan web sitelerinin kaynak kodlarını inceleyerek video ekleme yöntemlerini ve kullanılan HTML etiketi özelliklerini arkadaşlarınız ile paylaşınız.

3.6. Bağlantı Elemanları

HTML'nin önemli özelliklerinden biri, metin ya da resim üzerinden başka bir belgeye **bağlantı** (köprü) kurabilmesidir. HTML bağlantıları **link** ya da **köprü** olarak da ifade edilir. Bir bağlantıya tıklandığında başka bir belgeye veya aynı belge içindeki farklı bir bölüme geçilir. Fare imleci, bağlantının üzerine geldiğinde küçük bir ele dönüşür.

3.6.1. HTML Bağlantı Söz Dizimi

<a> etiketi, HTML belgelerinde bağlantı oluşturur. A harfi İngilizcedeki "anchor" kelimesinin kısaltmasıdır. Anchor, Türkçede gemilerde kullanılan "çıpa" anlamındadır (Görsel 3.39). HTML belgesi açısından ise "üzerine tıklanabilen metin" anlamına gelir. Bağlantı (köprü) etiketi <a> şeklindedir, href özelliği ile kullanılır. <a> etiketi bir adres belirterek başka web adreslerine ya da aynı sayfanın farklı bölümlerine köprü oluşturmak veya resimlere ve e-posta adresine link vermek için kullanılır. <a> etiketinin kullanımı aşağıda gösterilmiştir.

Görsel 3.39: <a> etiketi anlamı, çıpa

```
<a href="URL" target="hedef">Bağlantı Metni</a>
```

```
<a href="https://mtegm.meb.gov.tr" target="_blank">mtegm.meb.gov.tr</a>
```

Bağlantılar tüm tarayıcılarda varsayılan olarak şu şekilde görünür:
- Ziyaret edilmemiş bir bağlantının altı çizilidir ve mavidir.
- Ziyaret edilen bir bağlantının altı çizili ve mordur.
- Etkin bir bağlantının altı çizilidir ve kırmızıdır.

Bağlantılardan farklı görünüm elde etmek için CSS kullanılır.

3.6.1.1. URL Yapısı

URL, adres bilgisidir. Web tarayıcıya işaret edilen belgenin nerede olduğunu gösterir. URL'ler, dosyaların web ya da yerel (local) disk üzerinde yerini işaret eder. Bilgi işlem terimlerinden biri olan URL'nin açılımı Uniform Resource Locator'dır.
URL'nin genel yapısı şu üç kısımdan oluşur:
- Erişilecek kaynak tipi (www, ftp vb.)
- Sunucu adresi
- Dosya adı

KaynakTipi://host.saha [:port]/yol/DosyaAdı

3.6.1.2. Href Özelliği

<a>… bloku içine metin (kelime, cümle, kişi, kurum vb.), paragraf, başlık, resim, ses veya video eklenebilir. <a> etiketi içinde href özelliği ile hedef belirtilir.

13. Uygulama

Href özelliğinin kullanımı ile ilgili yöntemleri yönergeler doğrultusunda gerçekleştiriniz.

1. Adım: Temel HTML belgesi yapısını metin editörü kullanarak oluşturunuz.
2. Adım: <body> blokunda hedefsiz bağlantı oluşturunuz.

```
<a href="#"> Hedefsiz bağlantı </a>
```

3. Adım: Aynı sayfada "#cipa" adında etiket bağlantısı oluşturunuz.

```
<a href="#cipa"> Aynı sayfada bir etiket </a>
```

4. Adım: Aynı sitede sayfa bağlantısı oluşturunuz.

```
<a href="aynisite-sayfa.html"> Aynı sitede sayfa bağlantısı </a>
```

5. Adım: Aynı sitede dosya bağlantısı oluşturunuz.

```
<a href="aynisite-dosya.pdf"> Aynı sitede dosya bağlantısı </a>
```

6. Adım: Farklı bir siteye bağlantı oluşturunuz.

```
<a href="https://mtegm.meb.gov.tr"> Başka bir site bağlantısı </a>
```

7. Adım: Farklı sitede sayfa bağlantısı oluşturunuz.

```
<a href=" https://mtegm.meb.gov.tr/www/sss.php"> Başka sitede sayfa bağlantısı </a>
```

8. Adım: Farklı sitede dosya bağlantısı oluşturunuz.

```
<a href="http://kitap.eba.gov.tr/panel/dosyalar/upload/1375/0/U_0_18_09_2020_14_59_03_220.pdf"> Başka sitede dosya bağlantısı </a>
```

9. Adım: Oluşturduğunuz HTML belgesini tarayıcıda açıp bağlantıları kontrol ediniz.

Köprü bağlantıları için metin, resim, ses ve video kullanılabilir. Tablo 3.12'de bağlantı metinleri kullanımı verilmiştir.

Tablo 3.12: Bağlantı Metinleri Kullanımı

Bağlantı Metni	Örnek
Metin	tıklayın
Kurum Adı	Mesleki ve Teknik Eğitim
Başlık	CANLI DERS
Cümle	Kitabı görmek için tıklayınız
Resim	<img src="asistan.gif"></img>
Ses	<audio src="mars.mp3"></audio>
Video	<video src="ebavideo.ogg"></video>

3.6.1.3. Target Özelliği

Target özelliği sayesinde HTML bağlantısının nerede açılacağı belirlenir. "_blank" değeri bağlantıyı yeni bir pencere ya da sekmede açar. "_self" değeri bağlantıyı aynı çerçevede açar (varsayılan değer). "_parent" değeri bağlantıyı bir üst çerçevede açar. "_top" değeri bağlantıyı en üst çerçevede açar.

3.6.2. Sayfa İçi Bağlantılar

Çıpa olarak da adlandırılan sayfa içi bağlantı oluşturma, bir web sayfasında bağlantı verilen yere tıklandığında aynı sayfa içinde başka bir bölüme yönlendirilmeyi sağlar.

14. Uygulama

<a> etiketinin name özelliği ile sayfa içi bağlantılarını yönergeler doğrultusunda oluşturunuz.

1. Adım: Temel HTML belgesi yapısını metin editörü kullanarak oluşturunuz.
2. Adım: <body> etiketi içine <nav> blokunu oluşturup sayfa içi bağlantılarını oluşturunuz.

```
<nav>
        <a name="icindekiler"><h2>İÇİNDEKİLER</h2></a>
        <ul>
          <li><a href="#Bolum1">HTML5</a>
          <li><a href="#Bolum2">CSS</a>
```

```
            <li><a href="#Bolum3”>JavaScript</a>
            <li><a href="#Bolum4">ASP.Net Core</a>
          </ul>
          <p>Açıklama</br>.....</br>.....</br></p>
</nav>
```

3. Adım: <article> bloklarını kullanarak web sayfası içeriğini oluşturunuz.

```
          <p>Web Sayfası İçeriği </p>
<article>
 <section>
          <a name="Bolum1">HTML5/a> <a href="#icindekiler">İçindekiler</a>
          <p>Açıklama</br>.....</br>.....</br></p>
  </section>
</article>
<article>
 <section>
          <a name="Bolum2">CSS Bölümü</a> <a href="#icindekiler">İçindekiler</a>
          <p>Açıklama</br>.....</br>.....</br></p>
 </section>
</article>
<article>
 <section>
          <a name="Bolum3">JavaScript Bölümü</a> <a href="#icindekiler">İçindekiler</
a>
          <p>Açıklama</br>.....</br>.....</br></p>
 </section>
</article>
<article>
 <section>
          <a name="Bolum4">ASP.Net Core Bölümü</a> <a href="#icindekiler">İçindeki-
ler</a>
          <p>Açıklama</br>.....</br>.....</br></p>
 </section>
</article>
```

4. Adım: <nav> blokunda belirlediğiniz bağlantı isimlerini <article> bloklarında kullanınız.

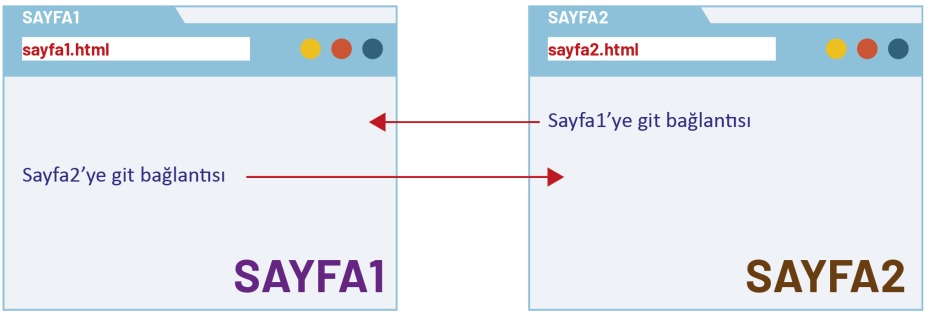
5. Adım: <figure> ve <footer> blokunu oluşturup bir logo yerleştiriniz ve bağlantı oluşturunuz.

```
<figure>
 <img src="logo.png" alt="logom" />
          <figcaption>
                    <a href="url">Benim Logom</a>, daha fazla bilgi için
          </figcaption>
</figure>
<footer>
          <h3 id="yazar">HTML5 Yazarı</h3>
</footer>
```

6. Adım: HTML belgesini kaydedip tarayıcıda görüntüleyiniz.

3.6.3. Sayfalar Arası Bağlantılar

Aynı sitedeki başka bir sayfaya gitmek için bağlantılar oluşturulabilir (Görsel 3.40). Sayfalar arasında bağlantı oluşturmak gezinmeyi kolaylaştırır.



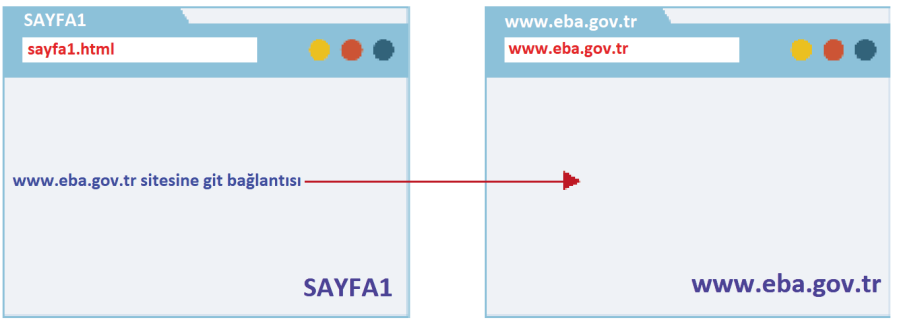

```
<a href="sayfa2.html">SAYFA2</a>
<a href="klasör\sayfa2.html">SAYFA2</a>
```

```
<a href="sayfa1.html">SAYFA1</a>
<a href="klasör\sayfa1.html">SAYFA1</a>
```

Görsel 3.40: Sayfalar arası bağlantı

3.6.4. Site Dışına Bağlantı

HTML belgesinde, farklı bir siteye gidilmesi için site dışı bağlantılar oluşturulabilir. Site dışı bağlantılar, kullanıcıyı farklı bir siteye yönlendirir (Görsel 3.41). Kullanıcı bağlantının bulunduğu siteden ayrılmış olur. Site dışı verilen linklerin site içi linklerden tek farkı, hedef değerinde gidilmesi istenen sitenin adının yazılı olmasıdır.




```
<a href="https://www.eba.gov.tr">SİTE</a>
<a href="https://www.eba.gov.tr/index.html">SİTE</a>
<a href="https://www.eba.gov.tr klasör/sayfa2.html">SİTE</a>
```

Görsel 3.41: Site dışına bağlantı

3.6.5. Diğer Bağlantı Çeşitleri

HTML belgesinde <a> etiketi kullanarak görüntü, e-posta, dosya vb. köprü bağlantıları da oluşturulabilir.

3.6.5.1. Görüntüyü Bağlantı Olarak Kullanma

Görüntü üzerinden köprü oluşturmak için aşağıdaki HTML kodu yazılır.

```
<a href="URL">
        <img src="kopru.jpg">
</a>
```

3.6.5.2. E-posta Bağlantısı Oluşturma

E-posta bağlantısı oluşturmak için **mailto** ifadesi kullanılır. E-posta bağlantısı üzerinden e-posta göndermek için e-posta istemcisi, bilgisayarda kurulu ve ayarlanmış olmalıdır.

```
<a href="mailto:info@site.com.tr">e-posta gönder</a>
```

3.6.5.3. Dosya Bağlantısı Oluşturma

Tarayıcılar; HTML belgeleri gibi pdf, gif, jpg vb. dosya türlerini de görüntüleyebilir. Bu tür dosyaları barındıran bir sitede dosyaya ulaşmak için dosya yolları kullanılır. Bu dosya yollarına bağlantı tanımlanarak dosyaların indirilmesi sağlanır.

Bağlantı oluşturulan HTML belgesi ile dosya aynı klasörde ise aşağıdaki HTML kodu yazılır.

```
<a href="dosya.pdf">Dosyayı İndir</a>
```

Bağlantı oluşturulan HTML belgesi ile dosya farklı bir sitede ise aşağıdaki HTML kodu yazılır.

```
<a href="http://www.eba.gov.tr/dosya.pdf">Dosyayı İndir</a>
```

Dosya bağlantıları, dosyayı sadece görüntülemek için veya indirme bağlantılarını kullanarak indirmek için kullanılır.

15. Uygulama

HTML bağlantıları oluşturma işlemini yönergeler doğrultusunda gerçekleştiriniz.

1. Adım: Temel HTML belgesi yapısını metin editörü kullanarak oluşturunuz.

2. Adım: <body> etiketi içine aşağıdaki gibi bağlantı kodları oluşturunuz.

```
<a href="index.html">anasayfa</a> |
<a href="hakkimda.html">hakkımda</a>|
<a href="galeri.html">galeri</a> |
<a href="urunler.html">ürünler</a>|
<a href="iletisim.html">iletişim</a>|
<a href="mailto:mailadresim@benimwebsayfam.com.tr">Eposta Gönder</a>
```

3. Adım: Oluşturduğunuz HTML belgesini “index.html” adında kaydediniz.

4. Adım: “index.html” dosyasının kopyasını alıp “hakkimda.html” adında kaydediniz.

5. Adım: “index.html” dosyasının kopyasını alıp “galeri.html” adında kaydediniz.

6. Adım: "index.html" dosyasının kopyasını alıp "urunler.html" adında kaydediniz.
7. Adım: "index.html" dosyasının kopyasını alıp "iletisim.html" adında kaydediniz.
8. Adım: Oluşturduğunuz HTML belgelerine açıklayıcı bir başlık bloku ekleyiniz.

```
<h1> Bu ANASAYFA </h1>
```

Adım 9: "index.html" HTML belgesini tarayıcıda açınız ve bağlantıları test ediniz.

Not

Bu adımları tekrar ederek farklı HTML sayfaları ve HTML bağlantıları oluşturabilirsiniz.

3.7. Form Elemanları

Formlar, web sitesi ziyaretçilerinden bilgi almayı sağlar. İnternet ortamını dergi, gazete, televizyon vb. ortamlardan ayıran en önemli özelliklerden biri formlardır. Formlar aracılığı ile geribildirim, yorum, öneri, şikâyet veya sipariş işlemleri çok kısa sürede yapılır.

3.7.1. Formun Tanımı

Form, kullanıcının veri girişi yapmasını ve sunucuya göndermesini sağlayan yapılardır. Form kullanılarak dinamik HTML belgeleri oluşturulur. Formlar aracılığıyla siteyi ziyaret eden kullanıcılar siteye üye olabilir, e-posta gönderebilir, herhangi bir ürün satın alabilir.

3.7.2. Form Oluşturmak

Form oluşturmak için üç temel form elemanı kullanılır: <form>, <label> ve <input> etiketi.

<form> etiketi, form oluşturmak için kullanılır. </form> kapama etiketi ile kullanılır. <form> bloku, diğer tüm form kontrollerini oluşturan etiketleri içine alarak kullanıcının veri girişi yapmasını sağlayan bir yapı tanımlar (Görsel 3.42).

Input, "girdi" anlamına gelir ve kullanıcıdan alınan verileri sunucuya göndermek için kullanılır.

Label, "etiket" anlamına gelir ve form elemanlarının başlıkları, bu eleman içine yazılır. <label> etiketi </label> kapama etiketi ile kullanılır.

```
<form name="giris" action="giris.html" method="get" >
  <label>Ad:</label> <input type="text"/> <br/><br/>
  <label>Soyad: </label> <input type="text"/> <br/><br/>
  <input type="submit" value="Gönder"/>
</form>
```

Görsel 3.42: Form oluşturma

Tablo 3.13'te <form> etiketi ile kullanılan özellikler ve görevleri verilmiştir.

Tablo 3.13: <form> Etiketi ile Kullanılan Özellikler ve Görevleri

Özellik	Görevi
name	Forma isim vermek için kullanılır.
action="dosya adı"	Formdaki verilerin gönderileceği dosya yolu (URL) belirlenir.
method="get | post"	Form veri gönderme yöntemleri; get ile veriler tarayıcının adres çubuğu üzerinden gönderilir, gönderilen veriler test edilebilir. **post** yöntemi (varsayılan) ile veriler HTML mesajı olarak gönderilir. **get** yöntemine göre daha güvenilirdir.

3.7.3. Form Elemanları

Form elemanları <form> ... </form> etiketleri arasında yazılır. <form> etiketleri dışında yazılan form elemanı, görsel olarak oluşur fakat etkin değildir. Form özelliği kullanılarak <form> bloku dışında oluşturulan veri giriş elemanı o forma dâhil edilir.

3.7.3.1. <input> Etiketi

<input> etiketi, genel amaçlı kullanılan form elemanıdır. <input> etiketi içinde kullanılan type özelliğinin alacağı değere göre form elemanını oluşturur. <input> etiketinin kapama etiketi bulunmaz.

Type Özelliğinin Alabileceği Standart Veri Tipleri

Text, tek satırlık yazı yazılabilecek alan ekler.

```
Ad:<input type="text" name="isim"/><br/>
Soyad:<input type="text"  name="soyisim"/>
```

Password, parola yazmak için kullanılır. Metin kutusuna yazılan karakterler daire karakteri veya yıldız **(*)** karakteri ile gösterilir.

```
Kullanıcı:<input type="text" name="kullanici"/><br/>
Şifre: <input type="password" name="sifre"/>
```

Checkbox, onay kutusu oluşturmak için kullanılır. Çoklu seçim yapmaya izin verir.

```
<input type="checkbox" name="kutu1" checked="on"/>HTML<br/>
<input type="checkbox" name="kutu2"/>CSS<br/>
<input type="checkbox" name="kutu3"/>Java Script
```

Radio, radyo düğmeleri oluşturmak için kullanılır. Onay kutularından farklı olarak çoklu seçim yapılamaz. Name özelliğinin değeri, diğer seçenekler için aynı olmalıdır.

```
<input type="radio" name="secim"/>Evet<br/>
<input type="radio" name="secim"/>Hayır
```

File, dosya seçim elemanı eklemek için kullanılır. Dosya yükleme işleminde "get metodu" kullanılmaz ve sadece HTML kodları ile dosya yükleme işlemi gerçekleşmez. Dosya yükleme elemanı sadece dosyanın seçilmesini sağlar.

```
<input type="file" name="dosya"/>
```

Submit, formdaki verileri kabul eder ve yollar. **Reset,** formdaki verileri temizler.

```
<form action="uye.html" method="post">
        <label>Ad Soyad:</label><input type="text" name="ad" autofocus="autofo-
cus"><br/><br/>
        <label>e-Posta:</label><input type= ><br/><br/>
        <label>Şifre:</label><input type="text" name="eposta"><br/><br/>
        <input type="radio" name="cinsiyet" value="erkek" checked="on"> Erkek <br/><br/>
        <input type="radio" name="cinsiyet" value="kadin"> Kadın<br/><br/>
        <input type="submit" value="Gönder"> <input type="reset" value="Temizle">
</form>
```

Sıra Sizde

Yukarıdaki <form> blokuna benzer formlar oluşturup arkadaşlarınızın oluşturdukları ile karşılaştırınız.

16. Uygulama

HTML5 ile form oluşturmak için kullanılan diğer veri tiplerini yönergeler doğrultusunda gerçekleştiriniz.

1. Adım: Temel HTML kod yapısını metin editörü kullanarak oluşturunuz.
2. Adım: <body> etiketi içine <form> bloku oluşturunuz.
3. Adım: <form> bloku içine aşağıdaki <input> değerlerini ekleyiniz.

```
<p>Renk seçin: <input type="color" name="renk"></p>
<p>Doğum tarihi: <input type="date" name="dogumtarihi"></p>
<p>Doğum günü:<input type="datetime" name="dogumgunu"></p>
<p>Ay ve Yıl: <<input type="month" name="ay"></p>
<p>Hafta ve Yıl: <input type="week" name="hafta"></p>
<p>Randevu saati: <input type="time" name="randevusaat"></p>
<p>Randevu tarihi ve saati: <input type="datetime-local" name="randevutarih"></p>
<p>E-posta: <input type="email" name="eposta"></p>
<p>Web sayfanız: <input type="url" name="weburl"></p>
<p>Sayı (1-9): <input type="sumber" name="sec" min="1" max="9"></p>
<p>Parola: <input type="password" name="sifre"></p>
<p>Aralık (1-9): <input type="range" name="secim" min="1" max="9"></p>
<p>Arama: <input type="search" name="arama"></p>
<p>GSM: <input type="tel" name="gsmno"></p>
```

4. Adım: Oluşturduğunuz HTML belgesini kaydedip tarayıcıda görüntüleyiniz.
5. Adım: Tarayıcıda oluşan form bileşenlerini kontrol ediniz.

Not

Form nesnelerinde kullanılan name özelliği, tarayıcıda forma girilen veriler üzerinde interaktif işlemler yapmak için değişken olarak kullanılacaktır.

Sıra Sizde

HTML form bileşenlerini kullanarak randevu formu oluşturunuz. Oluşturduğunuz formu arkadaşlarınızın oluşturdukları ile karşılaştırınız.

3.7.3.2. <button> Etiketi

<button> etiketi, buton oluşturmak için kullanılan anlamsal etikettir. <input> etiketi ile buton oluşturma yazım şekliyle aynıdır. <input> etiketinden farkı, <button> etiketinin </button> kapama etiketi ile kullanılmasıdır. Bu durum içerik alanında başka etiketlerin yer almasına olanak tanır.

```
<form action="uye.html" method="post">
        <label>Ad Soyad:</label><input type="text" name="ad"><br/><br/>
        <label>e-Posta:</label><input type="text" name="eposta"><br/><br/>
        <label>Şifre:</label><input type="text" name="eposta"><br/><br/>
        <input type="radio" name="cinsiyet" value="erkek" checked="on"> Erkek<br/><br/>
        <input type="radio" name="cinsiyet" value="kadin"> Kadın<br/><br/>
        <input type="hidden" name="ipAdresi" value="212.175.132.130”/>
        <button type="submit" value="Gönder">
                <img src="gonder.jpg" width="16" height="16">
        </button>
        <button type="reset" value="Temizle">
                <img src="temizle.jpg" width="16" height="16">
        </button>
</form>
```

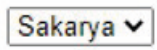
Sıra Sizde

<button> etiketini kullanarak 3.7.3.2’ deki HTML form blokuna benzer üyelik formu oluşturup tarayıcıda görüntüleyiniz.

3.7.3.3. <select>, <option>, <optgroup> Seçim Listesi Etiketleri

Seçenek sayısının çok fazla olduğu durumlar için **<select> etiketi** ile seçim listesi oluşturulur. **<option> etiketi**, seçim listesi ögelerini belirtir (Görsel 3.43). **<optgroup> etiketi,** liste elemanlarını gruplandırır. <select> etiketi bir veya birden fazla <option> etiketi içerir. <optgroup> etiketlerini gruplandırır. <select> etiketi bir veya birden fazla <option> etiketi içerir. <optgroup> etiketleri iç içe kullanılmaz. **Multiple** özelliği ile çok seçimli, **size** özelliği ile kaç seçenek görüneceği belirlenerek sabit veya aşağıya açılan listeler oluşturulur.

```
<select name="iller" size="1">
        <option value="plaka54">Sakarya</option>
        <option value="plaka06">Ankara</option>
        <option value="plaka34">İstanbul</option>
</select>
```



Görsel 3.43: Seçim listesi

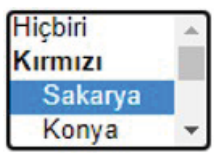
Seçim listesinde tek seçenek görünüyor ise **drop-down list** (aşağıya açılan liste), listede birden fazla seçenek görünüyor ise **list box** (sabit liste) adını alır (Görsel 3.44).

```
<select name="aylar" multiple>
        <option value="1">Ocak</option>
        <option value="2">Şubat</option>
        <option value="3">Mart</option>
        <option value="4">Nisan</option>
        <option value="5">Mayıs</option>
        <option value="6">Haziran</option>
</select>
```

Görsel 3.44: Multiple liste

Seçim listeleri **<optgroup> etiketi** ile gruplandırılır. <optgroup> etiketi ile oluşturulan grup isimleri seçilmez. </optgroup> kapama etiketi ile kullanılır (Görsel 3.45).

```
<select name="riskharitasi" multiple>
        <option selected value="none">Hiçbiri</option>
        <optgroup label="Kırmızı">
                <option value="k1">Sakarya</option>
                <option value="k2">Konya</option>
                <option value="k3">Balıkesir</option>
        </optgroup>
        <optgroup label="Sarı">
                <option value="s1">Eskişehir</option>
                <option value="s2">Kırşehir</option>
                <option value="s3">Yozgat</option>
        </optgroup>
        <optgroup label="Turuncu">
                <option value="t1">Artvin</option>
                <option value="t2">Çanakkale</option>
                <option value="t3">Kütahya</option>
        </optgroup>
```



Görsel 3.45: <option> kullanımı

```
        <optgroup label="Mavi">
                <option value="m1">>Uşak</option>
                <option value="m2">Batman</option>
                <option value="m3">Ağrı</option>
        </optgroup>
</select>
```

Sayfa açıldığında listedeki ilk seçenek seçili olarak gelir, farklı bir seçeneğin görünmesi için **selected** özelliği kullanılır (Görsel 3.45).

3.7.3.4. <fieldset> ve <legend> Etiketleri

<fields> etiketi, form elemanlarını bir arada toplar ve bunların etrafında bir çerçeve oluşturur. Oluşan çerçeve, tarayıcının genişliğine veya bulunduğu blokun genişliğine göre alanı kaplar. Bu etiketin başlık kısmını **<legend> etiketi** oluşturur (Görsel 3.46).

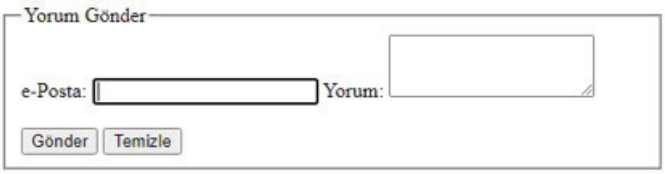
Görsel 3.46: <fieldset> ve <legend> kullanımı

```
<form>
<fieldset>
        <legend>Üyelik Bilgileri</legend>
                <label>Ad Soyad : </label>
                <input type="text" name="adSoyad" required="required"> <br/><br/>
                <label>e-Posta : </label>
                <input type="email"  name="ePosta" required="required"> <br/><br/>
                <label>Doğum Tarihi : </label>
                <input type="date" name="dogumTarihi" required="required"> <br/><br/>
                <input type="submit" value="Gönder"/>
                <input type="reset" value="Temizle"/>
</fieldset>
</form>
```

3.7.3.5. <textarea> Etiketi

<textarea> etiketi, formda yazı alanı oluşturmak için kullanılır. Genellikle **rows** ve **cols** özellikleri ile kullanılır. **Rows** yüksekliği, **cols** ise genişliği ifade etmek için kullanılır (Görsel 3.47).

Tabindex özelliği, form elemanları arasında klavyedeki **Tab** tuşu ile dolaşılacak sıranın belirlenmesine imkân sağlar.



Görsel 3.47: Rows, cols ve tabindex özellikleri

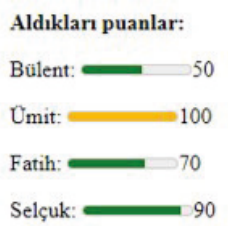
```
<form>
<fieldset>
        <legend>Yorum Gönder</legend>
                <label>e-Posta:</label>
                <input type="text" tabindex="1"/>
                <label>Yorum:</label>
                <textarea rows="3" cols="20" tabindex="2"></textarea> <br/><br/>
                <input type="submit"value="Gönder"/>
                <input type="reset" value="Temizle"/>
</fieldset>
</form>
```

Sıra Sizde

Fieldset özelliğini kullanarak mesajlaşma formu oluşturup tarayıcıda görüntüleyiniz.

3.7.3.6. <meter> Etiketi

<meter> etiketi, bilinen bir dizi, veya kesirli değer içinde bir skaler ölçüm tanımlar. Bu aynı zamanda bir gösterge olarak da bilinir. Ölçüm için <meter> etiketi kullanılır (Görsel 3.48).



Görsel 3.48: <meter> etiketi

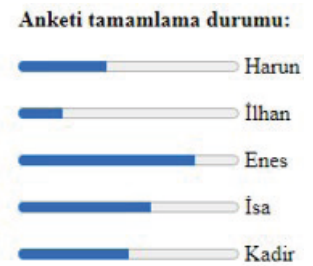
```
<p><strong>Aldıkları puanlar:</strong></p>
<p><label for="bulent">Bülent:</label>
<meter id="bulent" min="0" low="40" high="90" max="100" value="55"></meter>50</p>
<p><label for="umit">Ümit:</label>
<meter id="umit" min="0" low="40" high="90" max="100" value="100"></meter>100</p>
<p><label for="fatih">Fatih:</label>
<meter id="fatih"min="0" low="40" high="90" max="100" value="70"></meter>70</p>
<p><label for="selcuk">Selçuk:</label>
<meter id="selcuk" min="0" low="40" high="90" max="100" value="90"></meter>90</p>
```

Sıra Sizde

<meter> etiketi kullanarak derslerde aldığınız puanların HTML uygulamasını oluşturunuz.

3.7.3.7. <progress> Etiketi

<progress> etiketi, görevin ilerleme durumunu tanımlar (Görsel 3.49). Bir görevin ilerleme durumunu görüntülemek için JavaScript ve <progress> etiketi birlikte kullanılır. <progress> etiketi ölçüm için kullanılmaz.



Görsel 3.49: <progress> etiketi

```
<p><strong>Anketi tamamlama durumu:</strong></p>
<p><progress id="harun" max="100" value="40"></progress>
<label for="harun">Harun </label></p>
<p><progress id="ilhan" max="100" value="20"></progress>
<label for="ilhan">İlhan</label></p>
<p><progress id="enes" max="100" value="80"></progress>
<label for="enes">Enes</label></p>
<p><progress id="isa" max="100" value="60"></progress>
<label for="isa">İsa</label></p>
<p></p><progress id="kadir" max="100" value="50"></progress>
<label for="kadir">Kadir</label></p>
```

Sıra Sizde

<progress> etiketi kullanarak sınıfınıza ait sınıf başkanı seçim anket sonuçlarını gösteren HTML uygulaması oluşturunuz.

3.7.4. Veri Giriş (Input) Özellikleri

HTML5, form oluştururken formun genelinde veya form elemanı özellikleri kullanılarak form elemanı için veri girişlerini kontrol etmemize olanak tanır.

3.7.4.1. Form Özellikleri

HTML5, form oluştururken formun genelinde otomatik tamamlama yapılmasına ve veri giriş elemanı değerinin veri tipi uyumunun kontrol edilmesine olanak tanır.

Autocomplete özelliği, form elemanlarına daha önce giriş yapılmış ise veri girişi sırasında otomatik tamamlamak için kullanılır. Bu özellik "on" veya "off" değeri alır.

Novalidate özelliği, veri girişi sırasında veri giriş elemanı değerinin veri tipi ile uyumunu kontrol etmemesi için kullanılır.

Sıra Sizde

Aşağıdaki HTML kodlarını inceleyiniz, form özelliklerine dikkat ederek formu oluşturup tarayıcıda görüntüleyiniz.

```
<form action="islemyap.html" method="get" autocomplete="on" novalidate >
        <label for="adsoyad">Ad Soyad:</label>
        <input type="text" id="adsoyad" name="adsoyad"></br></br>
        <label for="eposta">e-Posta:</label>
        <input type="text" id="eposta" name="eposta"></br></br>
        <input type="submit">
</form>
```

3.7.4.2. Veri Giriş Özellikleri

HTML5, tarayıcı ekranında forma veri girişi sırasında verilerin form elemanı özellikleri ile kontrol edilmesine olanak tanır.

Placeholder, bilgi alanıdır, kullanıcı bir değer girmeden önce giriş alanında görüntülenir.

Autocomplete, form genelinde otomatik tamamlama açık ya da kapalı olduğunda giriş elemanları için otomatik tamamlama kapalıdır. Giriş elemanında bu özelliği kapatmak ya da açmak için autocomplete özelliği giriş elemanında kullanılır.

Autofocus, input etiketi içinde autofocus özelliği varsa sayfa yüklendiğinde imleç, bu giriş elemanına odaklanır. Sadece bir elemanda kullanılmalıdır.

List, önceden tanımlanmış datalist kullanmayı sağlar.

Min ve **Max,** veri giriş elemanının alacağı en büyük ve en küçük değerleri belirler. Sayı ve tarih için kullanılır.

Pattern; metin, tarih, arama, URL, telefon, e-posta ve şifre giriş türlerinin belirli bir formatta yazılıp yazılmadığını kontrol eder.

Required, gerekli alandır, required ifade eklenen veri giriş elemanının doldurulması zorunludur. Metin, arama, URL, telefon, e-posta, şifre, tarih seçiciler, sayı, onay kutusu, radyo ve dosya veri tipleri ile kullanılabilir.

Step, veri alanına girilebilecek sayı aralığını belirtir. Sayı, tarih, ay, hafta ve saat veri tipleri ile kullanılabilir.

Multiple, eklendiği veri tipine birden çok değer girilmesini sağlar.

Sıra Sizde

Aşağıdaki HTML kodlarını inceleyiniz, veri giriş özelliklerine dikkat ederek form oluşturup tarayıcıda görüntüleyiniz.

```
<form action="islem1.html" autocomplete="on">
<input formmethod="get" placeholder="Form Metodu get|post">
<input formnovalidate="formnovalidate" placeholder="Veri Tip Uyumu Kontrolsüz"></br>
<input type="email" id="eposta" name="eposta" autocomplete="on" placeholder="O-
tomatik açık">
<input type="date" id="tarihmin" name="tarihmin" min="2021-05-01">
<inputtype="number" id="aralik" name="aralik" min="1" max="5" placeholder="max">
<input type="password" pattern=".{6,}" title="Altı veya daha fazla" placeholder="daha
fazla">
<input type="password" pattern="(?=.*\d)(?=.*[a-z])(?=.*[A-Z]).{8,}" placeholder = "Şifre">
<input type="email" pattern="[a-z0-9._%+-]+@[a-z0-9.-]+\.[a-z]{2,}$" placeholder ="Epos-
ta">
<input type="url" pattern="https?://.+" placeholder="http:// dâhil edin">
<input type="number" step="2" placeholder="Çift Sayı">
<input list="hazirliste"><datalist id="hazirliste">
   <option value="Öğretmen">
   <option value="Öğrenci">
   <option value="Veli">
</datalist>
<input type="submit" value="Gönder">
<input type="submit" formaction="islem2.html" value="Başka Gönder">
</form>
```

HTML5 kodları ile oluşturulan formlar görsel açıdan çok etkileyici değildir. Form elemanlarını hizalamak için HTML tabloları kullanılır. Görsel açıdan güzel formlar oluşturmak için ise CSS kullanılır.

Not

Form doğrulama işlemi, bir web sitesinin güvenliği ve kullanılabilirliği için hayati öneme sahiptir. Doğrulama işlemi, giriş değerinin sunucuya gönderilmeden önce doğru biçimde olup olmadığının tarayıcıda kontrol edilmesidir. Örneğin, herhangi bir e-posta adresi için bir giriş alanı varsa değer kesinlikle geçerli bir e-posta adresi içermelidir. E-posta; bir harf veya sayı ile başlamalı, ardından @ (kuyruklu a) simgesi gelmeli ve bir alan adıyla bitmelidir.
HTML5, <input> elemanında pattern (desen) özelliği tanımlayarak, JavaScript'e başvurmadan temel veri doğrulama işlemi yapılmasına izin verir.

Araştırma

Pattern özelliğini kullanarak kullanıcı adının yalnızca küçük harflerden oluşması ve kullanıcı adı uzunluğunun 10 karakterden fazla olmamasını sağlayan deseni oluşturup tarayıcıda görüntüleyiniz. Oluşturduğunuz desene uymayan girişler yapıldığında çıkan uyarıyı nasıl değerlendireceğinizi araştırınız.

ÖLÇME VE DEĞERLENDİRME

A. Aşağıda verilen cümlelerin başındaki boşluğa cümle doğru ise "D" yanlış ise "Y" yazınız.

1. ( ) HTML belgesi dosya uzantısı ".html"dir.
2. ( ) <meta> etiketinin </meta> kapama etiketi mevcuttur.
3. ( ) Arama motorlarına bilgi vermek için <head> blokundaki bilgiler kullanılır.
4. ( ) HTML büyük ve küçük harfe duyarlıdır.
5. ( ) Anlamsal etiketler hem geliştirici hem de tarayıcılar tarafından açıkça anlaşılırdır.

B. Aşağıda verilen soruların doğru cevabını işaretleyiniz.

6. HTML kısaltmasının açılımı aşağıdakilerden hangisidir?

A) Hacker Tool Media Language
B) Hello Text Markup Language
C) Home Tool Markup Language
D) Hyper Text Markup Language
E) Hyperlink Text Media Language

7. <head> blokunda yazılan etiketlerden tarayıcı ekranında içeriği görüntülenen etiket aşağıdakilerden hangisidir?

A) <base> B) <link> C) <meta> D) <style> E) <title>

8. Aşağıdakilerden hangisi HTML5'te kullanılan medya etiketi değildir?

A) <audio> B) <music> C) <source> D) <track> E) <video>

9. Aşağıdakilerden hangisi HTML5 tasarım şablonunda kullanılan anlamsal etiketlerde değildir?

A)   B) <footer> C) <header> D) <nav> E) <section>

10. Aşağıdakilerden hangisi <input> etiketi type özelliği ile kullanılan HTML form elemanı değildir?

A) checkbox B) image C) radio D) text E) value

4.
ÖĞRENME BİRİMİ

BASAMAKLI STİL ŞABLONU
(Cascading Style Sheets)

NELER ÖĞRENECEKSİNİZ?

Bu öğrenme birimi ile;
- Web sayfalarındaki elamanlara stil uygulamayı,
- CSS'yi web sayfalarına ekleme yöntemlerini sıralamayı,
- Stil özellikleri değiştirilecek elemanları seçme yöntemlerini kullanmayı,
- Kutu modeli (div) nesnelerini kullanmayı,
- Web sayfasındaki elamanların görünüm ve pozisyon ayarlarını kavramayı,
- Web sayfasında kullanılacak renkleri tanımlamayı,
- Web sayfalarındaki metinsel içeriğin özelliklerini değiştirmeyi,
- Duyarlılık (responsivity) kavramını açıklamayı,
- Bootstrap kütüphanelerinin bağlantısını yapabilmeyi,
- Bootstrap ile farklı çözünürlüklerdeki cihazlara uygun tasarımları yapabilmeyi öğreneceksiniz.

ANAHTAR KELİMELER

Bootstrap, CSS, duyarlılık, görünüm, kutu modeli, RGB, stil.

Hazırlık Çalışmaları

1. Css stil sayfalarının hangi amaçla kullanıldığını düşünüyorsunuz? Açıklayınız.
2. Günümüzde farklı ekran çözünürlüklerine sahip cihazlara uygun tasarımlar nasıl yapılmaktadır? Bildiklerinizi arkadaşlarınızla paylaşınız.

4.1. CSS Ekleme Yöntemleri

HTML bir web sitesinin vücudunu oluşturur. CSS ise o vücudun üzerindeki elbiseleri, aksesuarları ve makyajı oluşturur. HTML kodları kullanılarak hazırlanan web sitelerinde, sayfa tasarımını özelleştirmek için birtakım kodlar bulunmaktadır. HTML kodları günümüzde kullanılan web sitelerini tasarlamak için tek başına yetersizdir. Sayfa tasarımını kolayca özelleştirmek, görsel açıdan zenginleştirmek ve farklı ekran çözünürlüklerine sahip cihazlara uygun hâle getirmek için web sayfaları hazırlanırken CSS kullanmak gerekmektedir (Görsel 4.1).

Görsel 4.1: CSS ekleme

Bir elemanı görsel olarak özelleştirmek için o elemanın değiştirilecek özelliğine tasarım için uygun değer atanır. Görsel 4.2'de görülen CSS kodunda elemanın yazı rengi beyaz, genişliği 1024 piksel olarak değiştirilmiştir.

Görsel 4.2: CSS kod yapısı

CSS'yi web sayfalarına eklemek için farklı yöntemler kullanılmaktadır.

4.1.1. Satır İçi CSS Ekleme

Doğrudan HTML etiketinin içine CSS ekleme yöntemidir. HTML etiketinin "style" özelliğine değer olarak CSS kodları girilir. Sadece CSS kodunun eklendiği HTML etiketinde stil değişiklikleri görülür.

```
<h1 style="background-color: lightblue;">CSS öğreniyorum</h1>
```

Yukarıdaki örnek kodun çıktısı Görsel 4.3'teki gibidir.

Görsel 4.3: Satır içi CSS

1. Uygulama

HTML etiketlerine satır içi CSS ekleme yöntemi kullanarak stil verme işlemini Görsel 4.4'de görüldüğü şekilde yönergeler doğrultusunda gerçekleştiriniz.

Görsel 4.4: Renkli satırlar

1. Adım: Paragraf elemanının içine İstiklal Marşı'nın ilk kıtasını yazınız.

2. Adım: Paragraf elemanının arka plan rengini kırmızı ve yazı rengini beyaz yapınız.

```
<p style="color:white; background-color:red;">
        Korkma, sönmez bu şafaklarda yüzen al sancak;<br>
        Sönmeden yurdumun üstünde tüten en son ocak.<br>
        O benim milletimin yıldızıdır, parlayacak;<br>
        O benimdir, o benim milletimindir ancak.<br>
</p>
```

Not

Bir elemanın arka plan rengi background-color özelliğine renk değeri atanarak değiştirilmektedir.

Sıra Sizde

İlk satırda yazı boyutu 20px ve arka plan rengi kırmızı, ikinci satırda yazı boyutu 15px ve arka plan rengi yeşil, üçüncü satırda yazı boyutu 10px ve arka plan rengi mavi olan web sayfasını Görsel 4.4'te görüldüğü gibi tasarlayınız.

4.1.2. Sayfa İçi CSS Ekleme

HTML kodlarının <head> </head> bölümüne <style> </style> etiketi içinde CSS ekleme yöntemidir. Yapılan stil değişiklikleri sadece tek sayfa için geçerlidir.

```
<html>
  <head>
    <style>
        p{
          background-color: red;
          color:white;
          font-size:20px;
          }
    </style>
  </head>
 <body>
    <p>Bu paragrafın arka plan rengi kırmızı, yazı rengi beyaz, yazı
      boyutu 20 pikseldir.
    </p>
 </body>
</html>
```

Örnek kodun çıktısı Görsel 4.5'teki gibidir. Sayfada başka <p> etiketleri kullanıldığında, kullanılan <p> etiketlerinin de özellikleri değişecektir.

Bu paragrafın arka plan rengi kırmızı, yazı rengi beyaz, yazı boyutu 20 pikseldir.

Görsel 4.5: Dâhilî CSS

Sıra Sizde

Dâhilî CSS ekleyerek aşağıdaki kodlarda bulunan <img> etiketlerinin tümünün genişliğini 120 piksel ve yüksekliğini 80 piksel yapınız.

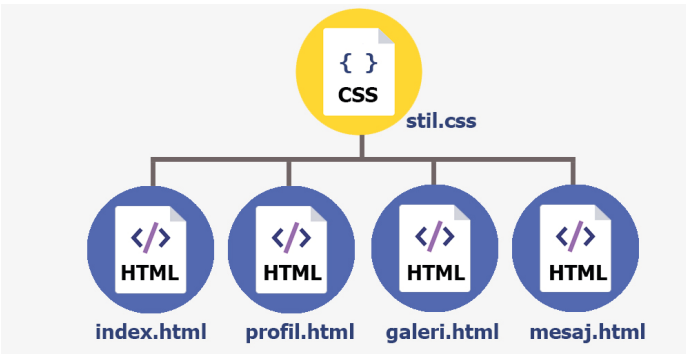
```
<body>
 <img src="galeri1.JPG">
 <img src="galeri2.JPG">
 <img src="galeri3.JPG">
 <img src="galeri4.JPG">
 <img src="galeri5.JPG">
<body>
```

Sıra Sizde

Üç adet "HTML" sayfası oluşturunuz ve HTML sayfalarının arka plan rengini kırmızı yapınız.

4.1.3. Sayfa Dışı (Haricî) CSS Ekleme

Web sayfasına uzantısı "css" olan bir stil dosyasını bağlayarak CSS ekleme yöntemidir. Web tasarımcıların en çok tercih ettiği CSS ekleme yöntemidir. Tüm web sayfalarındaki stil özelliklerini tek bir merkezden kontrol etmek için kullanılır. Örneğin, Şema 4.1'de görüldüğü gibi dört farklı web sayfasından oluşan kişisel bir web sitesi oluşturmak istenildiğinde tüm sayfaların yazı tiplerini, resim boyutlarını, arka plan renklerini vs. değiştirmek, web sitesinin tasarımını güncellemek, diğer yöntemlerle oldukça karmaşık olacak ve zaman alacaktır.



Şema 4.1: Sayfa dışı CSS ekleme

Sayfa dışı CSS ekleme yöntemi ile sadece aşağıdaki kodu web sayfalarının <head> bölümüne yazarak stil.css dosyasını bağlamak yeterli olacaktır.

```
<link rel="stylesheet" href="stil.css" />
```

Örnek kodda sayfa dışı “css” uzantılı bir dosya bağlantısı yapılmıştır.

<link> Etiketi: Web sayfasının başka dosyalarla bağlantı kurması için kullanılır.

rel Özelliği: Bağlantı kurulacak sayfa ile <link> etiketinin kullanıldığı sayfa arasında nasıl bir ilişki bulunduğunu belirtir. Yukarıdaki örnek kodda iki sayfa arasındaki bağlantı ilişkisinin stil sayfası şeklinde olacağı belirtilmiştir.

href Özelliği: Bağlantı kurulacak sayfanın dosya yolunu belirtir.

Not

Aynı web sayfasına birden fazla “css” uzantılı dosya bağlanabilir. Örneğin: stil.css haricinde stil2.css, stil3.css, ozel.css vb. dosyalar <link> etiketi ile eklenebilir.

Sıra Sizde

Bir adet haricî CSS sayfası ve üç adet “HTML” sayfası oluşturunuz. HTML sayfalarına haricî CSS sayfasını bağlayınız. Haricî CSS sayfasından tüm HTML sayfalarının arka plan rengini kırmızı yapınız. Aynı işlemlerin dâhilî CSS ile yapılmasının avantajları ve dezavantajlarını arkadaşlarınızla tartışınız.

2. Uygulama

Haricî CSS ekleme yöntemini kullanarak web sayfasında stil değiştirme işlemini yönergeler doğrultusunda gerçekleştiriniz.

1. Adım: KutuAyarlari.css ve ResimAyarlari.css isimlerinde iki adet haricî CSS sayfası ve index.html isminde bir adet “HTML” sayfası oluşturunuz.
2. Adım: index.html sayfasına iki CSS dosyasını da bağlayınız.
3. Adım: index.html sayfasına aşağıdaki kodda olduğu gibi kutular ve metinler ekleyiniz.

```
<div>
Web sitenize <b>harici CSS</b> sayfaları ekleyerek sayfalarınızı
tasarlamak çok kolaylaşmaktadır.
</div>
```

4. Adım: KutuAyarlari.css dosyasında, aşağıdaki gibi sadece kutulara ait stil özelliklerini değiştirmek için CSS kodları ekleyiniz.

```
div{font-size:34px; color:white; background-color:purple; }
```

5. Adım: ResimAyarlari.css dosyasında, aşağıdaki gibi sadece resimlere ait stil özelliklerini değiştirmek için CSS kodları ekleyiniz ve verilen adımlara göre web sayfalarını tasarlayınız.

```
img{ width: 200px; height: 200px; }
```

4.1.4. Çok Kullanılan CSS Kodları

Bazı CSS kodları kategoriler hâlinde kısa açıklamalarıyla birlikte Tablo 4.1, Tablo 4.2, Tablo 4.3, Tablo 4.4'te verilmiştir. Kodların detaylı kullanımlarına kitabı ilerleyen bölümlerinde değinilecektir.

Tablo 4.1: Yazı Stil Özellikleri (Tipografi)

CSS KODU	AÇIKLAMA	KULLANIM ŞEKLİ
font-size	Yazı boyutu	font-size: 16px; font-size: 0.5em;
font-style	Yazı eğiklik tipi	font-style:italic;
font-weight	Yazı kalınlığı	font-weight:bold;
font-family	Yazı font tipi	font-family:Verdana;
color	Yazı rengi	color:blue;
letter-spacing	Harfler arasındaki mesafe	letter-spacing: 0.5em;
word-spacing	Kelimler arasındaki mesafe	word-spacing: 0.5em;
word-wrap	Belirli bir alana sığmayan yazıların bir alt satıra geçirilmesi	word-wrap:break-word;
line-height	Satır yüksekliği	line-height:12px;
text-transform	Yazıların harflerini büyütme, küçültme	text-transform:uppercase;
text-decoration	Yazı çizgileri (alt, üst, orta)	text-decoration:underline;
text-align	Yazıları yaslama	text-align:center;
text-indent	Paragrafların başlangıç girintisi	text-indent:10px;
text-shadow	Yazı gölgesi	text-shadow:2px gray;
text-overflow	Belirli alana sığmayan yazının nasıl görünümü	text-overflow:ellipsis;

Tablo 4.2: Kutu Stil Özellikleri

CSS KODU	AÇIKLAMA	KULLANIM ŞEKLİ
width	Genişlik	Width:300px;
height	Yükseklik	Height:100px
maxwidth, maxheight	Genişliğin ve yüksekliğin alabileceği en üst değeri	maxwidth:400px; maxheight:250px;
minwidth, minheight	Genişliğin ve yüksekliğin alabileceği en alt değeri	minwidth:10px; minheight:10px;
margin	Kenarlık dışı boşluk	margin:15px; margin-left:10px;
padding	Kenarlık içi boşluk	padding:10px; padding-bottom:5px;
border-color	Kenarlık rengi	Border-color:red;
border-style	Kenarlık doku tipi	Border-style:solid;
border-width	Kenarlık kalınlığı	Border-width:3px;
border-radius	Kenarlığın köşe yuvarlaklığı	Border-radius:10px;

Tablo 4.3: Arka Plan Stil Özellikleri

CSS KODU	AÇIKLAMA	KULLANIM ŞEKLİ
background-color	Arka plan rengi	background-color: purple;
background-image	Arka plan resmi	background-image:url(resim.jpg);
background-position	Arka planın boyutu	background-size:300px 100px;
background-position	Arka planın konumunun başlangıç noktası	background-position-center center;
background-repeat	Arka plan resminin tekrarı	background-repeat:no-repeat;
background-attachment	Arka plan resmi ile zemin ilişkisi	background-attachment:fixed;

Tablo 4.4: Önemli Diğer CSS Kodları

CSS KODU	AÇIKLAMA	KULLANIM ŞEKLİ
display	Görünüm ayarları	display:inline;
position	Konumlandırma ayarları	position:absolute; left:100px; top:200px;
float	Kaydırarak yerleşim	float:left; float:right:
clear	Kaydırma işlemi etkilerini temizleme	clear:left, clear:both;
z-index	Katman sıralaması	z-index:2;
opacity	Transparan görünüm	opacity:0.4;
box-shadow	Gölgelendirme	box-shadow: 3px gray;
cursor	Fare simgesi	cursor:hand;
overflow	Belirli alana sığmayan içeriğin görünümü	Overflow:scroll;
list-style-type	Listeleme elemanları işaretleri	list-styl-type:square;
filter	Resim efekti	filter:blur(5px);
scroll-behavior	Sayfa içi yönlendirme efekti	html{scroll-behavior: smooth;}
transition	Geçiş efektleri (Örneğin; fare ile bir elemanın üstüne gelince renginin yarım saniye boyunca yavaşça değişmesi)	transition-property: color; transition-duration:0.5s; transition -delay:0.5s;
@media	Çözünürlük sorgusu	@media screen and (min-width:600px){ }

4.1.5. Seçiciler (Selectors)

Sayfa içi veya sayfa dışı CSS ekleme yöntemlerinde stil özellikleri değiştirilmek istenilen elemanların seçiciler (selectors) kullanılarak belirtilmesi gerekmektedir. Seçiciler, eşleşen HTML elemanlarını bulur ve stil değiştirme işlemini yapar. Birçok seçici vardır. Bunlardan en çok kullanılanlar Tablo 4.5'te gösterilmektedir.

Tablo 4.5: Seçiciler

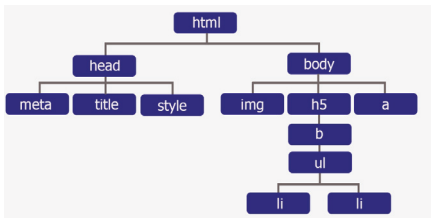
SEÇİCİ TÜRÜ	KULLANIMI
Etiket (Tag) Seçiciler	Etiket_Adı
ID (Kimlik) Seçiciler	#Kimlik_Adı
Class (Sınıf) Seçiciler	.Sınıf_Adı
Çoklu (Multiple) Seçiciler	A,B
Çocuk (Child) Seçiciler	A>B
Torun (Descendant) Seçiciler	A B
Sözde (Pseudo) Sınıf Seçiciler	A:hover
Sözde (Pseudo) Eleman Seçiciler	A:first-line

Not

Yukarıdaki tabloda A ve B harfleri herhangi bir etiket, kimlik veya sınıf ismini temsil etmektedir. Seçiciler konusunun daha iyi anlaşılabilmesi için HTML sayfalarının hiyerarşik yapısının nasıl olduğunun bilinmesi gerekir.

```
<html>
   <head>
      <meta charset="utf-8">
      <title>Html yapısı</title>
      <style>
       /* CSS kodları */
      </style>
   </head>
   <body>
      <img src="kodlama.jpg">
      <h5>
        <b>
         <ul>
            <li>Nesne Tabanlı Programlama</li>
            <li>WEB TABANLI UYGULAMA GELİŞTİRME</li>
         </ul>
        </b> </h5>
      <a href="index.html"><b>Anasayfa</b></a>
   </body>
</html>
```

Yukarıdaki örnek HTML kodlarının şematik görünümü Şema 4.2'deki gibidir. HTML hiyerarşik yapısına eklenen veya çıkarılan her HTML kodu, bu şemada bir değişiklik meydana getirir



Şema 4.2: Html hiyerarşik yapısı

Hiyerarşik yapının en üst noktasında <html> etiketi bulunmaktadır. Sayfada görüntülenecek elemanlar <body> etiketi altında bulunmaktadır. Yeni bir eleman eklendiğinde body etiketi altında yeni bir dal oluşacaktır. Şema 4.2'de görülen <img>, <h5> ve <a> etiketleri <body> etiketi tarafından kapsanır fakat kendi aralarında aynı seviyedeki elemanlardır. etiketleri de kendi aralarında aynı seviyedir fakat sırasıyla <ul>, <b>,<h5>,<body> ve en üstte <html> etiketi tarafından kapsanmaktadır.

Sıra Sizde

Aşağıdaki HTML kodlarının şematik görünümünü çiziniz. Kendi çiziminizi arkadaşlarınızın çizimleriyle karşılaştırınız.

```
<html>
   <head>
      <meta charset="utf-8">
      <title>Duyarlı Web Sitesi</title>
      <style>
       /* CSS kodları */
      </style>
   </head>
   <body>
      <div><h3>Responsive(Duyarlı) Web Sitesi Çalışma Ortamları</h3></div>
      <div>
       <table>
          <tr>
             <td><b>Telefon</b></td>
             <td><b>Tablet</b></td>
          </tr>
          <tr>
             <td><b>Bilgisayar</b></td>
             <td><b>Televizyon</b></td>
          </tr>
       </table>
      </div>
   </body>
</html>
```

4.1.5.1. Etiket Seçiciler

Etiket seçiciler, belirtilen etiket ismiyle eşleşen sayfadaki tüm elemanların stilini değiştirmek için kullanılır.

```
p{
   color:orange;
   font-size:12px;
 }
```

Yukarıdaki örnek kodda sayfadaki tüm <p> etiketleri seçilir ve hepsinin yazı rengi turuncu, yazı boyutu 12 piksel yapılır.

3. Uygulama

Etiket seçici kullanarak HTML elemanlarına stil verme işlemlerini Görsel 4.6'da görüldüğü şekilde yönergeler doğrultusunda gerçekleştiriniz.

Görsel 4.6: Etiket seçiciler

1. Adım: Web sayfasının arka plan rengini "skyblue" yapınız.

2. Adım: Sayfaya bir adet sıralı liste ve iki adet resim elemanı ekleyiniz.

3. Adım: Sıralı listenin maddelerine sırasıyla "HTML, CSS, JAVASCRIPT, ASP.NET CORE" yazınız.

4. Adım: Sıralı liste maddelerinin stil özelliklerini; yazı rengi siyah, yazı boyutu 16 piksel, yazıların altı çizili olacak şekilde ayarlayınız.

5. Adım: Resimlerin genişliğini 220 piksel ve yüksekliğini 130 piksel olarak ayarlayınız.

HTML

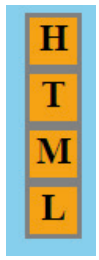
```
<body>
    <ol type="1">
        <li>HTML</li>
        <li>CSS</li>
        <li>JAVASCRIPT</li>
        <li>ASP .NET CORE</li>
    </ol>
    <img src="resim1.jpg">
    <img src="resim2.jpg">
</body>
```

CSS

```
body{
    background-color:skyblue;
    }
  li{
    color:black;
    font-size: 16px;
    text-decoration: underline;
    }
 img{
    width:220px;
    height:130px;
    }
```

4. Uygulama

Etiket seçici kullanarak HTML elemanlarına stil verme işlemlerini Görsel 4.7'de görüldüğü şekilde yönergeler doğrultusunda gerçekleştiriniz.



Görsel 4.7: HTML harfleri

1. Adım: Web sayfasının arka plan rengini **skyblue** yapınız.
2. Adım: Sayfaya dört adet etiketi ekleyiniz.
3. Adım: Her etiketinin içine etiketi ekleyiniz.
4. Adım: etiketlerinin içine sırasıyla H, T, M, L harflerini yazınız.
5. Adım: etiketlerinin stil özelliklerini; arka plan rengi turuncu, şekli kare, yazıları ortalanmış, çerçeve kenarlık kalınlığı 2 piksel, çerçeve kenarlık dokusu solid, çerçeve kenarlık rengi gri olacak şekilde ayarlayınız.
6. Adım: Stil özelliklerini ayarlarken etiket seçici kullanınız ve web sayfasını Görsel 4.7'de görüldüğü gibi tasarlayınız.

HTML	CSS

<body>
 H
 T
 M
 L
</body>
	body{ background-color:skyblue; }
	div{
	border:2px gray;
	width: 20px;
	height: 20px;
	background-color: orange;
	text-align: center;
	}

Sıra Sizde

Bir adet haricî CSS sayfası ve iki adet HTML sayfası oluşturunuz. HTML sayfalarına haricî CSS sayfasını bağlayınız. HTML sayfalarının tasarımlarını kendiniz belirleyiniz (doğa, spor, sanat, ticaret vb.) ve belirlediğiniz tasarıma uygun şekilde elemanlar ekleyiniz. HTML sayfalarındaki tüm elemanların stil özelliklerini etiket seçici kullanarak ayarlayınız.

4.1.5.2. Kimlik (ID) Seçiciler

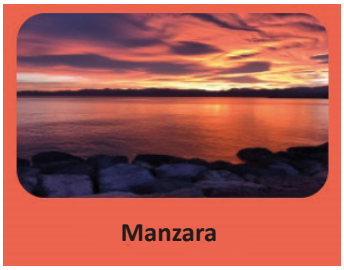
ID seçiciler, sayfadaki elemanlar arasından sadece ID özelliğinin değeri ile eşleşen elemanı seçer. Aynı ID değeri birden fazla elemana verilmemelidir. ID seçici # (diyez) işareti ile kullanılır.

```
#resim{
    width:150px;
    height:200px;
  }
```

Yukarıdaki örnek kodda ID özelliği **resim** olan nesne seçilir ve genişliği 150 piksel, yüksekliği 200 piksel yapılır.

5. Uygulama

Kimlik (ID) seçici kullanarak HTML elemanlarına stil verme işlemlerini Görsel 4.8'de görüldüğü şekilde yönergeler doğrultusunda gerçekleştiriniz.



Görsel 4.8: ID seçiciler

1. Adım: Web sayfasının arka plan rengini turuncu yapınız.

2. Adım: ID özelliğinin değeri "resim" olan bir <img> etiketi ekleyiniz ve <img> etiketine resim bağlayınız.

3. Adım: ID özelliğinin değeri "baslik" olan bir <h3> etiketi ekleyiniz ve içine "Manzara" yazınız.

4. Adım: <img> etiketinin stil özelliklerini; genişlik 220 piksel, yükseklik 130 piksel, kenarları 20 piksel yuvarlanmış olacak şekilde ayarlayınız.

5. Adım: <h3> etiketinin stil özelliğini, soldan çerçeve dışı boşluğu 65 piksel olacak şekilde ayarlayınız.

HTML

```
<body>
  <img src="manzara.jpg" id="resim">
  <h3 id="baslik">Manzara</h3>
</body>
```

CSS

```
body{
    background-color:orange;
    }
 #resim{
   width:220px;
   height:130px;
   border-radius: 20px;
    }
#baslik{
   margin-left:65px;
    }
```

6. Uygulama

ID seçici kullanarak HTML elemanlarına stil verme işlemini Görsel 4.9'da görüldüğü şekilde yönergeler doğrultusunda gerçekleştiriniz.

Görsel 4.9: CSS harfleri

1. Adım: Web sayfasına 3 adet etiketi ekleyiniz.
2. Adım: Her etiketinin içine etiketi ekleyiniz.
3. Adım: etiketlerinin içine sırasıyla C, S, S harflerini yazınız.
4. Adım: etiketlerinin ortak stil özelliklerini; genişlik ve yükseklik 35 piksel, yazı boyutu 30 piksel, yazıları ortalanmış olarak ayarlayınız.
5. Adım: İlk kutunun arka plan rengini "tomato", ikinci kutunun arka plan renginin "teal", son kutunun arka plan rengini "purple" olarak ayarlayınız.

HTML	CSS

<body>
 C
 S
 S
</body>
	div{
	width:35px; height:35px;
	font-size:30px; text-align: center;
	}
	#kutu1{ background-color: tomato; }
	#kutu2{ background-color: teal; }
	#kutu3{ background-color: purple; }

Sıra Sizde

Kendi isminizin harflerini Uygulama 6'daki gibi tüm kutular farklı renkte olacak şekilde ID seçiciler kullanarak bir web sayfası tasarlayınız.

4.1.5.3. Sınıf (Class) Seçiciler

Sınıf (Class), sayfadaki elemanlar arasından class özelliğinin değeri ile eşleşen tüm elemanları seçer. Birden fazla elemanın class değeri aynı olabilir. Class seçici **"." (nokta)** işareti ile kullanılır.

```
.icerik{
      font-size:10px;
      color:blue;
     }
```

Yukarıdaki örnek kodda sayfadaki class özelliği "içerik" olan tüm elemanlar seçilir ve yazı boyutu 10 piksel, yazı rengi mavi olarak ayarlanır.

7. Uygulama

Sınıf (Class) seçici kullanarak HTML elemanlarına stil verme işlemini Görsel 4.10'da görüldüğü şekilde yönergeler doğrultusunda gerçekleştiriniz.

Görsel 4.10: Class seçiciler

1. Adım: Web sayfasına dört adet resim ekleyiniz.

2. Adım: Tüm resimlerin class özelliklerinin değerini "galeri" yapınız.
3. Adım: Web sayfasının arka plan rengini "lightcyan" yapınız**.**
4. Adım: Tüm resimlerin stil özelliklerini; genişlik 160 piksel, yükseklik 90 piksel, çerçeve rengi 2 piksel, çerçeve tipi solid, çerçeve rengi koyu turuncu, çerçeve kenarları 10 piksel yuvarlanmış olarak ayarlayınız.

HTML	CSS

<body>
<img src="uzay.jpg" class="galeri">
<img src="kumsal.jpg" class="galeri">
<img src="gunBatimi.jpg" class="galeri">
<img src="yildizlar.jpg" class="galeri">
</body>
	body{background-color:lightcyan;}
	.galeri{
	width: 160px;
	height: 90px;
	border-color: 2px;
	border-style: solid;
	border-color:darkorange;
	border-radius: 10px;
	}

8. Uygulama

Sınıf (Class) seçici ve Kimlik (ID) seçici kullanarak HTML elemanlarına stil verme işlemini Görsel 4.11'de görüldüğü şekilde yönergeler doğrultusunda gerçekleştiriniz.

Görsel 4.11: Javascript harfleri

1. Adım: Web sayfasının arka plan rengini turuncu yapınız.
2. Adım: Web sayfasına yedi adet etiketi ekleyiniz.
3. Adım: Her etiketinin içine etiketi ekleyiniz.
4. Adım: etiketlerinin içine sırasıyla J, S, C, R, I, P, T harflerini yazınız.
5. Adım: etiketlerinin ortak stil özelliğini yazı boyutu 30 piksel olarak sınıf seçiciler kullanarak ayarlayınız.
6. Adım: Diğer etiketlerinden farklı olarak sadece "J" harfinin bulunduğu etiketinin stil özelliklerini yazı boyutu 40 piksel ve yazı rengi beyaz olacak şekilde ID seçici kullanarak ayarlayınız.

HTML	CSS

<body>
 	
J
 	
S
C
R
I
P
T
</body>
	body{background-color:orange;}
	.harf{
	font-size:30px;
	}
	#ozelHarf{
	font-size:40px;
	color:white;
	}

Sıra Sizde

Kendi adınızın sadece baş harfini ve soyadınızın tüm harflerini içeren uygulama 8'deki gibi görünüme sahip bir web sayfasını class seçiciler kullanarak tasarlayınız.

4.1.5.4. Çoklu (Multiple) Seçiciler

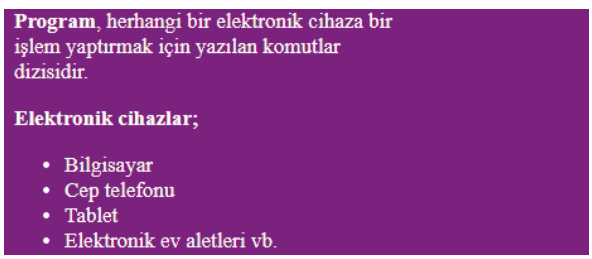
Çoklu (Multiple) seçiciler, birden fazla seçiciye aynı stil özelliklerini uygulamak için kullanılan yöntemdir. Seçiciler arasında **","** **(virgül)** işareti kullanılır.

```
.icerik,p{
        color:gray;
       }
```

Yukarıdaki örnek kodda sayfadaki class özelliği içerik olan elemanlar ve <p> etiketine sahip tüm elemanlar seçilir ve yazı renkleri gri yapılır.

9. Uygulama

Çoklu seçiciler kullanarak HTML elemanlarına stil verme işlemini Görsel 4.12'de görüldüğü şekilde yönergeler doğrultusunda gerçekleştiriniz.



Görsel 4.12: Çoklu seçiciler

1. Adım: Stil.css adında bir haricî CSS dosyası ve index.html adında bir HTML dosyası oluşturunuz ve Stil.css dosyasını index.HTML dosyasına bağlayınız.

2. Adım: index.html dosyasının <body> etiketi içine aşağıdaki kodu yazınız.

```
<p><b>Program</b>, herhangi bir elektronik cihaza bir işlem yaptırmak için
  yazılan komutlar dizisidir.</p>
 <b>Elektronik cihazlar;</b>
 <ul>
    <li>Bilgisayar</li>
    <li>Cep telefonu</li>
    <li>Tablet</li>
    <li>Elektronik ev aletleri vb.</li>
 </ul>
```

3. Adım: Web sayfasının arka plan rengini mor yapınız.

4. Adım: <p>, <b>, <ul> etiketlerinin ortak stil özellikleri; yazı rengi beyaz, yazı boyutu 16 piksel, genişlikleri 300 piksel olacak şekilde çoklu seçiciler kullanarak ayarlayınız.

CSS

```
body{
    background-color:purple;
    }
p,b,ul{ color:white; font-size:16px; width: 300px; }
```

Sıra Sizde

Oluşturduğunuz web sayfasına beş farklı resim ekleyiniz. İlk eklenen resim etiketine ID seçici değeri, sonraki eklenen üç resim etiketine class seçici değerleri veriniz. Son eklenen resim etiketine herhangi bir seçici değeri vermeyiniz. Resimlerin ortak stil özelliklerini; genişlik 100 piksel, yükseklik 100 piksel, kenarları 5 piksel yuvarlanmış olacak şekilde çoklu seçiciler kullanarak ayarlayınız.

4.1.5.5. Çocuk (Child) Seçiciler

Çocuk (Child) seçiciler, iç içe eklenmiş elemanlar arasından seçim yapmak için kullanılır. Belirtilen elemanın bir alt seviyesinde bulunan tüm elemanlar (çocuk) arasından eşleşenleri seçer.

```
p>b{
  font-size:20px;
 }
```

Yukarıdaki örnek kodda <p> etiketinin kapsadığı ilk alt elemanlar arasındaki tüm <b> etiketleri seçilir ve yazı boyutu 20 piksel olarak ayarlanır.

10. Uygulama

Çocuk seçiciler kullanarak HTML elemanlarına stil verme işlemini Görsel 4.13'de görüldüğü şekilde yönergeler doğrultusunda gerçekleştiriniz.

Görsel 4.13: Çocuk seçiciler

1. Adım: HTML dosyasının <body> etiketi içine aşağıdaki kodu yazınız.

```
<div>
  <b>KULLANICI PROFİLİ</b>
  <hr>
  <img src="profilBayan.png">
  <b>Zeynep</b>
  <p>
    <b>HTML, CSS kodlayabiliyor.</b>
  </p>
 </div>
```

2. Adım: etiketinin stil özelliklerini; arka plan rengi darkcyan, çerçeve kalınlığı 2 piksel, çerçeve dokusu solid, genişliği 400 piksel olacak şekilde etiket seçici kullanarak ayarlayınız.

3. Adım: <img> etiketinin stil özelliklerini, genişlik ve yükseklik değerleri 100 piksel

4. Adım: İçinde "KULLANICI PROFİLİ" ve "Zeynep" yazılarının olduğu <b> etiketlerinin yazı renklerini çocuk seçiciler kullanarak beyaz olarak Görsel 4.13'te görüldüğü gibi ayarlayınız.

CSS

```
  div{ background-color:darkcyan;  border:2px solid; width:400px ;  }
div>img{ width: 100px;  height:100px; }
 div>b{ color:white; }
```

Sıra Sizde

Aşağıdaki kodlar çalıştığında "Başlık-1" ve "Başlık-2" yazılarındaki stil değişimlerinin neler olacağını yazınız.

HTML

```
<div>
    <h3>Başlık-1</h3>
    <span>
      <h3>Başlık-2</h3>
    </span>
 </div>
```

CSS

```
div>h3{
   color:purple;
   background-color: orange;
  }
```

4.1.5.6. Torun (Descendant) Seçiciler

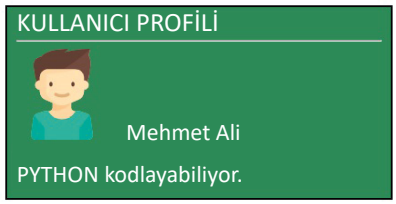
Torun (Descendant) seçiciler, çocuk seçicilerde olduğu gibi iç içe eklenmiş elemanlar arasından seçim yapmak için kullanılır. Bir elemanın kapsadığı tüm alt elemanlarda (çocuk, torun) eşleşenleri seçer. Çocuk seçici sadece bir alt elemandaki eşleşmeleri seçerken torun seçici, tüm alt elamanlardaki eşleşmeleri seçer.

```
p b{
  font-size:20px;
 }
```

Yukarıdaki örnek kodda <p> etiketinin kapsadığı tüm alt elemanlardaki <b> etiketleri seçilir ve yazı boyutu 20 piksel olarak ayarlanır.

11. Uygulama

Torun seçiciler ve çocuk seçiciler kullanarak HTML elemanlarına stil verme işlemini Görsel 4.14'de görüldüğü şekilde yönergeler doğrultusunda gerçekleştiriniz.



Görsel 4.14: Torun seçiciler

1. Adım: HTML dosyasının <body> etiketi içine aşağıdaki kodu yazınız.

```
<div>
    <b>KULLANICI PROFİLİ</b>
    <hr>
    <img src="profilBay.png">
    <b>Mehmet Ali</b>
    <p>
      <b>Python kodlayabiliyor.</b>
    </p>
 </div>
```

2. Adım: etiketinin stil özelliklerini; arka plan rengi **seagreen**, çerçeve kalınlığı **2 piksel**, çerçeve dokusu solid, genişliği **400 piksel** olacak şekilde etiket seçici kullanarak ayarlayınız.

3. Adım: <img> etiketinin stil özelliklerini **genişlik** ve **yükseklik değerleri 100 piksel** olacak şekilde çocuk seçiciler kullanarak ayarlayınız.

4. Adım: Tüm <b> etiketlerinin yazı renklerini torun seçiciler kullanarak beyaz olarak Görsel 4.14'de görüldüğü gibi ayarlayınız.

CSS

```
  div{ background-color:seagreen;  border:2px solid;   width:400px ; }
div>img{ width: 100px; height:100px; }
 div b{  color:white; }
```

Sıra Sizde

Aşağıdaki kodlar çalıştığında "Başlık-1" ve "Başlık-2" yazılarındaki stil değişimlerinin neler olacağını yazınız.

HTML

```
<div class="cerceve">
  <h3>Başlık-1</h3>
  <span>
    <h3>Başlık-2</h3>
  </span>
</div>
```

CSS

```
.cerceve h3{
    color:purple;
    background-color: orange;
   }
```

4.1.5.7. Sözde (Pseudo) Sınıf Seçiciler

Sözde (Pseudo) sınıf seçiciler, aslında sınıf olmayan fakat elemanların özel durumlarını tanımlamak için kullanılan seçicilerdir. Sözde sınıflar bir eleman için farklı durumlarda farklı stiller uygulanmasını sağlar. Sözde sınıf seçici etiket isminden hemen sonra **":" (iki nokta)** işareti yazılarak kullanılır.

```
a:link {
    color:orange;
    }
```

Örnek kodda henüz tıklanmamış <a> etiketleri seçilir ve yazı renkleri turuncu olarak ayarlanır. Birçok sözde sınıf seçici bulunmaktadır (Tablo 4.6).

Tablo 4.6: Sözde Sınıf Seçiciler

Sözde Sınıf Seçici	AÇIKLAMA	KULLANIM ŞEKLİ
link	Tıklanmamış link özellikleri belirleyen seçici	a:link{ color:red;}
visited	Ziyaret edilmiş link özellikleri belirleyen seçici	a:visited{opacity:0.5;}
active	Link'in veya başka bir elemanın tıklanma anındaki özellikleri belirleyen seçici (Fare sol tuşu basılı olma durumu)	img:active{ font-size:16px; color:blue;}
hover	Link'in veya başka bir elemanın fare ile üzerine gelindiğindeki özellikleri belirleyen seçici	div:hover{ width:200px; height:200px;}
focus	Odaklanılan elemanın özelliklerini belirleyen seçici (Genellikle input elemanlarında kullanılır.)	İnput:focus{ background-color:lighthgray}
first-child	Bir elemanın kapsadığı ilk alt elemanın özelliklerini belirleyen seçici	p:first-child{ background-color:orange;}
last-child	Bir elemanın kapsadığı sonuncu alt elemanın özelliklerini belirleyen seçici	p:last-child{ background-color:skyblue;}

Checked, disabled, enabled, empty, target ve daha birçok sözde sınıf seçici bulunmaktadır.

12. Uygulama

Sözde sınıf seçiciler ve çocuk seçiciler kullanarak HTML elemanlarına stil verme işlemini Görsel 4.15'te görüldüğü şekilde yönergeler doğrultusunda gerçekleştiriniz.

Görsel 4.15: Sözde sınıf seçiciler

1. Adım: HTML dosyasının <body> etiketi içine aşağıdaki kodu yazınız.

```
<ul>
 <li><a href="#">Etiket </a></li>
 <li><a href="#">Özellik</a></li>
 <li><a href="#">Değer</a></li>
 <li><a href="#">Fonksiyon</a></li>
</ul>
```

2. Adım: Sayfa arka plan rengini **indigo** yapınız.

3. Adım: Linklerin henüz tıklanmamışken yazı renklerini turuncu ve yazı boyutlarını **22 piksel** yapınız.

4. Adım: Linklerin tıklandıktan sonraki yazı renklerini **gri** yapınız.

5. Adım: Fare ile linklerin üzerine gelindiğinde arka plan rengini **beyaz** yapınız.

6. Adım: Linklerin üzerine fare ile basılı tutulduğunda yazı boyutunu **30 piksel** yapınız.

7. Adım: Web sayfasının linklerin tıklanıp tıklanmama durumlarına göre farklı görünümleri Görsel 4.15'te görülmektedir. Tüm adımları takip ederek web sayfasını tasarlayınız.

CSS

```
   body{ background-color: indigo; }
 a:link {color:orange; font-size:22px;}
a:visited{color:gray;}
 a:hover{background-color:white; cursor:hand; }
a:active{font-size: 30px;}
```

Sıra Sizde

Aşağıdaki kodlara göre <input> nesneleri içine bilgi girişi yapıldığında neler olacağını yazınız.

HTML

```
<body>
  <form>
    <input type="text" id="mail">
    <input type="password" id="sifre">
  </form>
</body>
```

CSS

```
#mail:focus{
        background-color: tomato;
       }
#sifre:focus{
        background-color: tomato;
        }
```

4.1.5.8. Sözde Eleman Seçiciler

Elemanların belli kısımlarını seçmek için kullanılan seçicilerdir. Sözde elemanlar, var olan bir elemanı alt kısımlara böler. Bölünen kısım tam anlamıyla bir eleman değildir, sadece elemanın bir bölümüdür. Sözde eleman seçici etiket isminden hemen sonra **"::" (iki tane iki nokta üst üste işareti)** yazılarak kullanılır.

```
p::first-line{
        font-weight: bold;
        }
```

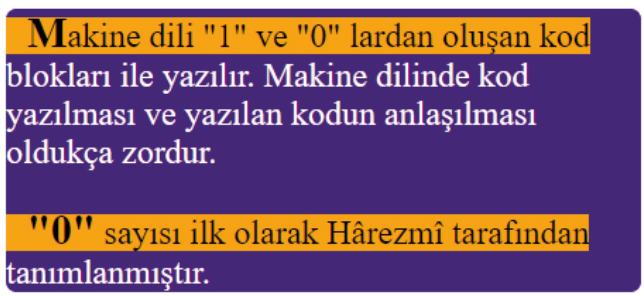
Yukarıdaki örnek kodda <p> etiketlerinin ekranda görülen sadece ilk satırı seçilir ve yazıları kalın olarak ayarlanır. Birçok sözde eleman seçici bulunmaktadır (Tablo 4.7).

Tablo 4.7: Sözde Eleman Seçiciler

Sözde Eleman Seçici	Açıklama	Kullanım Şekli
first-line	Elemanın ilk satırının özelliklerini belirleyen seçicidir.	div::first-line{color:white}
first-letter	Elemanın ilk harfinin özelliklerini belirleyen seçicidir.	div::first-letter{color:red}
before	Elemanın öncesine içerik ve içeriğe ait çeşitli özellikler eklemek için kullanılan seçicidir.	p::before{content: 'içerik'; color:red; }
after	Elemanın sonrasına içerik ve içeriğe ait çeşitli özellikler eklemek için kullanılan seçicidir.	p::after{ content: 'içerik'; color:blue; }
selection	Elemanın seçilen bölümlerine özellikler eklemek için kullanılan seçicidir.	div::selection{color:red;}

13. Uygulama

Sözde eleman seçiciler kullanarak HTML elemanlarına stil verme işlemini Görsel 4.16'da görüldüğü şekilde yönergeler doğrultusunda gerçekleştiriniz.



Görsel 4.16: Sözde eleman seçiciler

1. Adım: HTML dosyasının <body> etiketi içine aşağıdaki kodu yazınız.

```
<div>
  <p>Makine dili "1" ve "0" lardan oluşan kod blokları ile yazılır.
    Makine dilinde kod yazılması ve yazılan kodun anlaşılması oldukça
    zordur.</p>
   <p>"0" sayısı ilk olarak Hârezmî tarafından tanımlanmıştır.</p>
</div>
```

2. Adım: etiketinin stil özelliklerini; arka plan rengini **indigo**, genişlik **250 piksel**, yazı rengi beyaz, kenarları **5 piksel, yuvarlanmış** olarak ayarlayınız.

3. Adım: Sayfadaki <p> etiketlerinin ilk harfinin stil özelliklerini; yazı boyutu 20 piksel, yazıları kalın ve sol çerçeve dışı boşluğu 10 piksel olarak ayarlayınız.
4. Adım: Sayfadaki <p> etiketlerinin ilk satırının stil özelliklerini; arka plan rengi turuncu ve yazı rengi siyah olarak ayarlayınız.
5. Adım: Tüm adımları takip ederek Görsel 4.16'daki görüneme sahip bir web sayfasını tasarlayınız.

CSS

```
div{ width: 250px; background-color:indigo;
    color:white; border-radius: 5px;
   }
p::first-letter{font-size:20px; font-weight: bold; margin-left:10px; }
p::first-line{ background-color: orange; color:black; }
```

Sıra Sizde

Aşağıdaki kodlar çalıştığında sayfada nasıl bir görüntü oluşacağını arkadaşlarınızla paylaşınız.

HTML

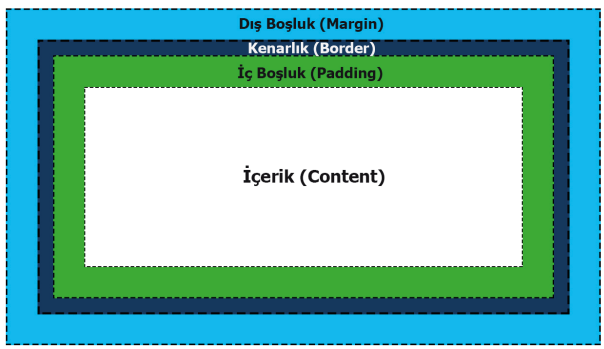
```
<body>
  <h5 id="icerik_ekleme">
  <---Before-After--->
 </h5>
</body>
```

CSS

```
#icerik_ekleme::before{
        content:'(önceki)';
        background-color:red;
        color:White;}
 #icerik_ekleme::after{
        content:"(sonraki)";
        background-color:blue;
        color:White;}
```

4.2. Kutu Modeli Özellikleri ve Çalışma Prensipleri

HTML'de <body> etiketi içerisine eklenen her eleman, bir kutu şeklinde sayfaya yerleşmektedir. Kutuların her birinin margin (dış boşluk), border (kenarlık), padding (iç boşluk), content (içerik) bölümleri bulunur (Görsel 4.17). Bir sayfadaki en büyük kapsayıcı kutu <body> elemanıdır. Genelde kutu denilince akla ve etiketleri gelmektedir. Fakat <a>, <b>, <i>, <li>, <img> ve diğer tüm elemanlar da kutu şeklinde sayfaya yerleşmektedir.



Görsel 4.17: Kutu modeli

4.2.1. CSS Ölçü Birimleri

Boyutlandırma işlemleri yaparken **px**, **em**, % ölçü birimleri kullanılabilir.

• **px (Piksel):** Sabit ölçü birimidir. Piksel cinsinde ölçü vermek için kullanılır.

• **% (Yüzde):** Göreceli ölçü birimidir. Elemanların kendisini kapsayan elemana göre ölçü biriminin değeri değişmektedir. Örneğin; elemanı tarafından kapsanan <p> elemanı için width:50%; font-size:33% şeklinde yüzde ölçü birimi ile stil ayarları yapıldığında elemanın genişlik değerinin yüzde 50'si ve font boyutu değerinin yüzde 33'ü <p> elemanın yeni değerleri olacaktır.

```
div{ width:300px; font-size:20px; }
  p{ width:50%;   font-size:33%   }
```

Yukarıdaki örnek koda göre <p> elemanın genişlik değeri 150 piksel, font boyutu 10 piksel olacaktır.

• **em:** Göreceli ölçü birimidir. % (yüzde) ölçü birimiyle arasındaki fark, elemanların kendisini kapsayan elemanın yazı tipi boyutuna göre ölçü biriminin değişmesidir. Örneğin; <body> elemanının varsayılan yazı boyutu 16px'tir, elemanın kapsadığı diğer elemanlar için 0.5em=8px, 1em=16px, 2em=32px vb. şekilde olacaktır. <body> elemanının yazı boyutu değiştirildiğinde em değerleri de değişecektir.

```
body{ font-size:32px;}
     p{ width:5em; font-size:0.25em;}
```

Yukarıdaki örnek koda göre <p> elemanın genişlik değeri 160 piksel, yazı boyutu 8 piksel olacaktır.

• **vw (Ekran Genişliği), vh (Ekran Yüksekliği):** Göreceli ölçü birimidir. Web sayfasının yansıtıldığı ekranın genişlik ve yükseklik değerlerine göre ölçü birimi değişmektedir. Örneğin, web sayfası tam ekran görüntüleniyorsa sayfa içerisinde bir elemana width:10vw; height:20vh; kodu eklenmişse elemanın genişliği sayfanın o anki genişliğinin onda biri ise o anki yüksekliğinin beşte biri kadar olacaktır. Sayfa görünümü küçültülüp büyütüldüğünde vw, vh değerleri de değişecektir.

HTML	CSS

<body>
Kutu 1
Kutu 2
<u>Kutu 3</u>
 Kutu 4
</body>
	body{
	background-color:royalblue;
	}
	b,i,u,div{
	border:solid red 2px;
	}

Görsel 4.18: Kutular

Yukarıdaki örnek kodlar çalıştırıldığında Görsel 4.18'de görüldüğü gibi elemanların kutu şeklindeki sınırları belli olmaktadır.

4.2.2. Dış Boşluk (Margin)

Dış boşluk (Margin), kutunun diğer kutularla veya sayfanın kenarlarıyla arasındaki boşluğu

belirleyen CSS kodudur. Kutu dışı boşluk da denir. Beş farklı şekilde kullanılabilir (Tablo 4.8).

Tablo 4.8: Dış Boşluk Çeşitleri

CSS Kodu	Açıklama	Kullanımı
margin-top	Üst taraftan kutu dışı boşluk verir.	margin-top: 10px;
margin-right	Sağ taraftan kutu dışı boşluk verir.	margin-right: 25px;
margin-bottom	Alt taraftan kutu dışı boşluk verir.	margin-bottom: 15px;
margin-left	Sol taraftan kutu dışı boşluk verir.	margin-left: 5px;
margin	Tüm yönlerden kutu dışı boşluk verir.	margin: 10px; Not: Her yönden 10 piksel kutu dışı boşluk verir) margin:10px 25px 15px 5px; Not: Sırasıyla üst, sağ, alt, sol taraflardan kutu dışı boşluk verir.)

4.2.3. Kenarlık (Border)

Kenarlık (Border), kutu kenarlarının genişlik, renk, doku, köşe yuvarlaklığı özelliklerini belirleyen CSS kodudur. Kenarlık, kutunun dış boşluğu ile iç boşluğu arasındaki sınır çizgilerinin oluşturduğu bölümdür. Beş farklı şekilde kullanılabilir (Tablo 4.9).

Tablo 4.9: Kenarlık Çeşitleri

CSS Kodu	Açıklama	Kullanımı
border	Kenarlığın sırasıyla width, style, color özelliklerini tek kod ile belirler.	border: 3px solid blue;
border-width	Kenarlığın kalınlığını belirler.	border-width: 3px; border-top-width: 0px; border-right-width:3px border-bottom-width: 0px; border-left-width: 3px;
border-style	Kenarlığın doku tipini belirler. solid: Düz çizgi dotted: Noktalı double: Çift çizgi dashed: Kesik çizgi	border-style: solid; border-top-style: dotted; border-right-style: double border-bottom-style: dashed; border-left-style: solid;
border-color	Kenarlığın rengini belirler.	border-color: blue; border-top-color: red; border-right-color: orange border-bottom-color: white; border-left-color: black;
border-radius	Kenarlığın köşe yuvarlaklığını belirler. Kenarlığın köşeleri üst-sol, üst-sağ, alt-sağ, alt-sol şeklinde ifade edilebilir.	border-radius: 2px; border-top-left-radius: 3px; border-top-right-radius: 4px; border-bottom-right-radius: 5px; border-bottom-left-radius: 6px; border-radius: 3px 4px 5px 6px;

4.2.4. İç Boşluk (Padding)

İç boşluk (Padding), kutunun içerik ve kenarlık bölümleri arasındaki boşluğu belirleyen CSS kodudur. Kutu içi boşlukta denir. Beş farklı şekilde kullanılabilir (Tablo 4.10).

Tablo 4.10: İç Boşluk Çeşitleri

CSS Kodu	Açıklama	Kullanımı
padding-top	Üst taraftan kutu içi boşluk verir.	padding-top: 10px;
padding-right	Sağ taraftan kutu içi boşluk verir.	padding-right: 25px;
padding-bottom	Alt taraftan kutu içi boşluk verir.	padding-bottom: 15px;
padding-left	Sol taraftan kutu içi boşluk verir.	padding-left: 5px;
padding	Tüm yönlerden kutu içi boşluk verir.	padding: 10px; Not: Her yönden 10 piksel kutu dışı boşluk verir.) padding:10px 25px 15px 5px; Not: Sırasıyla üst, sağ, alt, sol taraflardan kutu dışı boşluk verir.

4.2.5. İçerik (Content)

İçerik (Content), kutuda metinsel ifadelerin, resimlerin, videoların vb. görsel öğelerin gösterildiği bölümdür.

```
<a href="#">Burası a etiketinin içeriğidir.</a>
```

```
<b>Burası b etiketinin içeriğidir</b>
```

```
<div>
  <b>Burası hem div etiketinin hem b etiketinin içeriğidir</b>
  <img src="icerik.jpg">
  <h1>Burası hem div etiketinin hem h1 etiketinin içeriğidir.</h1>
</div>
```

4.2.6. Görünüm Ayarları

Web sayfasını oluşturan elemanlar sayfaya yerleşimlerine göre ikiye ayrılır.

Satır İçi Seviyesi Elemanlar: Satıra yan yana eklenir. Aynı satırda önünde veya sonunda eleman bulunabilir. Genişlik, yükseklik, metin hizalama vb. kodlar, satır içi seviyedeki bir nesne için kullanılamaz.

Blok Seviyesi Elemanlar: Sayfaya alt alta eklenir, aynı satırda yanında başka bir eleman bulunmaz. Genişlik, yükseklik, metin hizalama vb. kodlar blok seviyedeki bir nesne için kullanılabilir.

Örneğin; <b>, <i>, <u>, , <img> gibi elemanlar, varsayılan olarak seviyesi satır içi seviyedeyken <h1>, <p>, , <li> gibi elemanlar, varsayılan olarak blok seviyesindedir. Elemanların sayfaya yerleşim seviyeleri ve daha birçok görünüm özellikleri display kodu ile değiştirilebilir.

Display: Genel olarak elemanların yerleşim seviyesini, sayfada görünüp görünmeyeceğini belirlemek için kullanılan stil kodudur (Tablo 4.11).

Tablo 4.11: Görünüm Ayarları

CSS Kodu ve Kullanımı	Açıklama
display:none	Elemanın sayfada görünmez olmasını sağlar. Eleman sayfada herhangi bir yer kaplamaz.
display:inline	Elemanın satır içi seviyesinde davranmasını sağlar.
display:block	Elemanın blok seviyesinde davranmasını sağlar.
display:inline-block	Elemanın hem satır içi hem de blok seviyesinde davranmasını sağlar. Karma bir yerleşim seviyesidir. Elemanlar satır seviyesindeki gibi yan yana yerleşir. Blok seviyesindeki gibi genişlik, yükseklik, metin hizalama vb. kodlar kullanılabilir.

14. Uygulama

HTML elemanlarının görünüm ayarlarını değiştirme işlemini Görsel 4.19'da görüldüğü şekilde yönergeler doğrultusunda gerçekleştiriniz.

Görsel 4.19: Görünüm seviyeleri

1. Adım: Web sayfasına **iki adet elemanı** ekleyiniz.
2. Adım: Sayfanın arka plan rengini **mor** yapınız.
3. Adım: elemanlarının stil özellikleri; arka plan rengi **turuncu,** genişlik **100 piksel**, yükseklik **100 piksel** olarak ayarlayınız ve sayfayı görüntüleyiniz.
4. Adım: elemanının stil kodlarına **display:inline** kodunu ekleyiniz ve tekrar sayfayı çalıştırınız.

HTML	CSS

kutu1
 kutu2
	div{
	background-color:orange;
	width: 100px;
	height: 100px;
	display: inline;
	}

Not

128. sayfada yer alan örnek kodda elemanların display özelliği inline olarak değiştirildiği için Görsel 4.19'da görüldüğü gibi genişlik ve yükseklik özellikleri geçersiz kalır.

Float: Bir elemanın, sayfanın normal akışı dışında sola veya sağa kaydırılarak konumlandırılması için kullanılan stil komutudur. Sayfanın normal yerleşim akışı değişir fakat diğer elemanlar sola veya sağa kaydırılmış nesnenin etrafını sardıktan sonra sayfa, normal yerleşim akışına devam eder. Float kullanım şekilleri Tablo 4.12'de görüldüğü gibidir.

Tablo 4.12: Float Kullanım Şekilleri

CSS Kodu ve Kullanımı	Açıklama
float:left	Elemanı sayfa akışını bozarak sola kaydırır.
float:right	Elemanı sayfa akışını bozarak sağa kaydırır.
float:none	Varsayılan olarak her elemanın floatelemanın float özelliği değeri "none"'dır. Herhangi bir sağa, sola kaydırma işlemi yapılmayacağını belirtir.

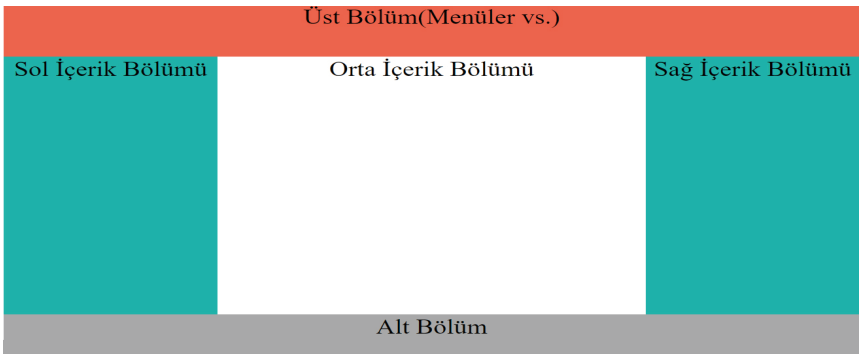
Clear: Float özelliği kullanılmış bir nesneden sonra clear özelliği kullanılmış bir nesne ekleyerek sayfanın normal yerleşim akışına devam etmesi sağlanır. Clear kullanım şekilleri Tablo 4.13'te görüldüğü gibidir.

Tablo 4.13: Clear Kullanım Şekilleri

CSS Kodu ve Kullanımı	Açıklama
clear:left	float:left komutunun etkisini kaldırır.
clear:right	float:right komutunun etkisini kaldırır.
clear:both	Hem float:left hem de float:right komutunun etkilerini kaldırır.

15. Uygulama

 elemanlarının **float** ve **clear** özelliklerini kullanarak basit bir sayfa tasarımını Görsel 4.20'de görüldüğü şekilde yönergeler doğrultusunda gerçekleştiriniz.



Görsel 4.20: Sayfa tasarımı

1. Adım: index.html sayfasına kimlik (ID) özellikleri; banner, left, center, right, footer olan beş tane elemanı ekleyiniz ve elemanlarının hangi bölüme ait olduğu bilgisini yazınız ve tüm elemanların yazılarını ortalı yapınız.

```
<div id="Banner">Üst Bölüm(Menüler vs.)</div>
<div id="Left">Sol İçerik Bölümü</div>
<div id="Center">Orta İçerik Bölümü</div>
<div id="Right">Sağ İçerik Bölümü</div>
<div id="Footer">Alt Bölüm</div>
```

2. Adım: Banner id'li elemanın genişliğini %100, yüksekliğini 40 pixel, arka plan rengini tomato yapınız.

3. Adım: Left ve Right id'li elemanın genişliğini %25, yüksekliğini %80, arka plan rengini lightseagreen, float özelliğini left yapınız.

4. Adım: Center id'li elemanın genişliğini %50, yüksekliğini %80, float özelliğini left yapınız.

5. Adım: Footer id'li elemanın genişliğini %100, yüksekliğini 40 pixel, arka plan rengini seagreen, clear özelliğini both yapınız.

```
    body{ text-align: center; }
#banner{ background-color: tomato; width: 100%; height: 40px; }
    #left{ background-color: lightseagreen; width: 25%; height: 80%; float:left; }
 #center{ width: 50%; height: 80%; float: left; }
  #right{ background-color: lightseagreen; width: 25%; height: 80%; float: left; }
 #footer{ background-color: darkgray; width: 100%; height: 40px; clear: both; }
```

16. Uygulama

Bir mesajlaşma uygulaması için gelen mesaj kutusu tasarımını Görsel 4.21'de görüldüğü şekilde yönergeler doğrultusunda gerçekleştiriniz.

Görsel 4.21: Gelen mesaj kutusu

1. Adım: index.html, mesajlar1.html, mesajlar2.html, mesajlar3.html, mesajlar4.html, stil.css isimli dosyalar oluşturunuz.

2. Adım: index.html sayfasına stil.css dosyasını bağlayınız.

3. Adım: Kişilerin resimleri için "resimler" isimli bir klasör oluşturunuz ve içine dört adet resim ekleyiniz.

4. Adım: stil.css dosyasına aşağıdaki kodları ekleyiniz.

CSS

```
body{ background-color:lightyellow;}
a{text-decoration: none;}
.mesajKutusu{ clear: both;margin-top: 10px;background-color:lightsteelblue;
          width: 250px; height: 60px; border-radius: 15px; }
.mesajKutusu:hover{background-color:lavender;  cursor:hand; }
.mesajKutusu img{
       width: 50px; height: 50px; border-radius: 35px; float: left; margin:5px;}
 span{font-size: 18px; margin-left: 20px; margin-top:20px; }
.adSoyad{padding-top:10px;color:black}
.tarih{padding-top:5px; font-size:10px; color:dimgray }
```

5. Adım: index.html sayfasının <body> elemanı içine aşağıdaki kodları ekleyiniz ve eksik olan kodları tamamlayınız.

HTML

```
 <div class="mesajKutusu">
    <a href="mesajlar1.html">
     <img src="resimler/kisi1.jpg">
     <div class="adSoyad"> Harun(Mühendis)</div>
     <div class="tarih"> bugün 22:00</div>
    </a>
 </div>
<!-- Diğer 3 kişi için mesaj kutusu kodları -->
```

Z-index: Sayfaya eklenen elemanlar konumlandırma, yerleşim ayarları ile üst üste gelebilirler. Bu gibi durumlarda hangi elemanın üst katmanda hangi elemanın alt katmanda gösterileceğini ayarlamak için z-index komutu kullanılır. En az iki elemanın üst üste geldiği durumlarda geçerlidir. Z-index değeri varsayılan olarak tüm elemanlarda "0"dır. Z-index değeri en büyük olan en üste görünür.

```
#resim1{ z-index:-1; }
```

Yukarıdaki örnek koda göre **resim1 id**'li eleman, üst üste geldiği elemanlardan daha altta görülecektir.

```
#resim1{ z-index:1;  }
```

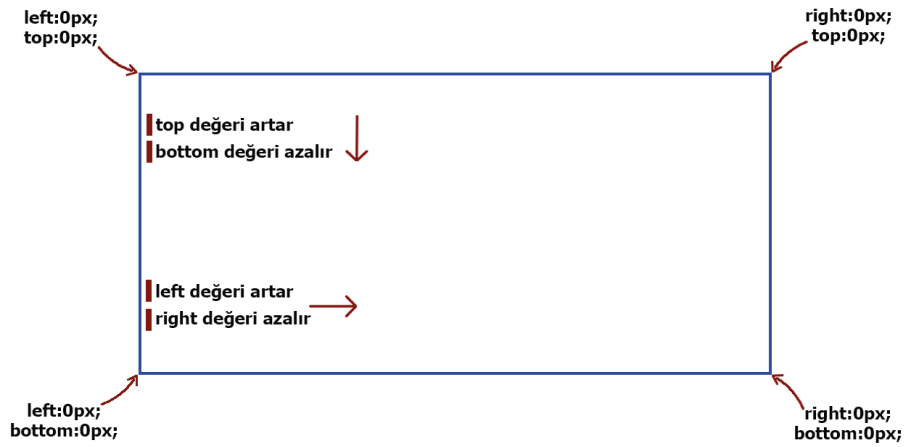
Yukarıdaki örnek koda göre **resim1 id**'li eleman, üst üste geldiği elemanlardan daha üstte görülecektir.

```
#arkaPlan{ z-index:1; }
   #resim{ z-index:2; }
     #yazi{ z-index:3; }
```

Yukarıdaki örnek koda göre **arkaPlan id**'li eleman, **resim id**'li elemandan daha altta kalır; **resim id**'li eleman da **yazi id**'li elemandan daha altta kalır.

4.2.7. Pozisyon Ayarları

Web sayfasını oluşturan elemanların konumlarının nasıl olacağını belirlemek için **konumlandırma (position)** komutu kullanılır. Konumlandırma işlemlerinde koordinat belirtmek için **üst (top), sağ (right), alt (bottom), sol (left)** özellikleri kullanılır. Kutu modeli konusunda bahsedildiği gibi web sayfasındaki elemanlar, kutu şeklinde sayfaya yerleşmektedir. Her kutunun köşeleri bulunmaktadır. Bir kutunun köşelerinin koordinatları Görsel 4.22'de görüldüğü gibidir. Elemanlar sol üst köşeleri belirtilen koordinat noktasına yerleştirilerek konumlandırılır.



Görsel 4.22: Pozisyon ayarları

Statik (Static) Pozisyon: Elemanlar sayfanın normal akışına göre yerini alır. Tüm HTML elemanlarının varsayılan pozisyon özelliği statictir. Top, right, bottom, left özellikleri bu konumlandırma komutuyla birlikte çalışmaz.

```
#Kutu{   position:static;   left:100px;   top:100px;  }
```

Kutu id'li elemanın pozisyon özelliği **static** olduğu için belirtilen konuma yerleştirilmez.

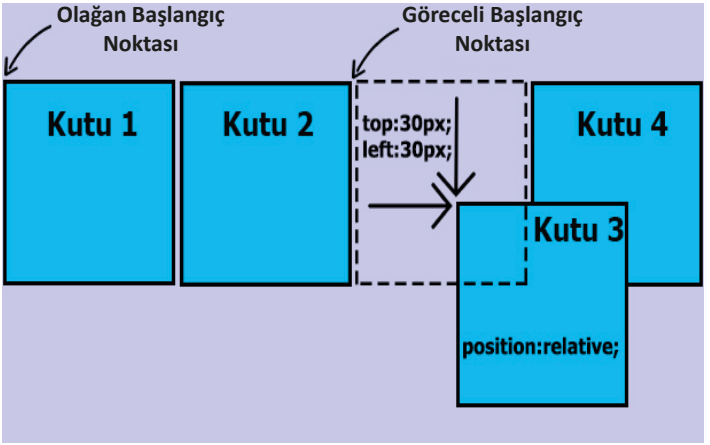
Göreceli / Bağlı Pozisyon: Her web sayfasının elemanların yerleşimi için referans aldığı olağan başlangıç noktası sayfanın sol üst köşe noktasıdır (left:0px; top:0px;).

```
#Kutu{   position:relative;   left:100px;   top:100px;  }
```

Göreceli pozisyona sahip bir elemanın sayfadaki diğer elemanların konumlarına göre başlangıç noktası değişir.

17. Uygulama

Göreceli pozisyon ile eleman konumlandırma işlemlerini Görsel 4.23'te görüldüğü şekilde yönergeler doğrultusunda gerçekleştiriniz.



Görsel 4.23: Göreceli pozisyon ayarı

1. Adım: elemanları kullanarak genişliği ve yüksekliği 50 piksel olan dört adet kutu oluşturunuz.

2. Adım: elemanlarının tümünün display özelliğini; **inline-block**, arka plan rengi **lightskyblue**, kenarlık kalınlığını **2 piksel**, kenarlık dokusunu **solid**, kenarlık rengini **siyah** olarak ayarlayınız.

3. Adım: Kutu 3'ün "position" özelliğini göreceli, "left" özelliğini 30 piksel ve "top" özelliğini 30 piksel olarak ayarlayınız.

HTML

```
<div class="kutuSablonu">Kutu 1</div>
<div class="kutuSablonu">Kutu 2</div>
<div class="kutuSablonu goreceliKutu">
 Kutu 3</div>
<div class="kutuSablonu">Kutu 4</div>
```

CSS

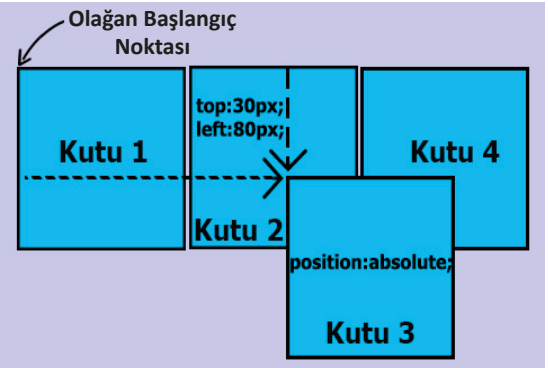
```
.kutuSablonu{
   display: inline-block;
   width: 50px; height: 50px;
   background-color: lightskyblue;
   border: 2px solid black;
}
.goreceliKutu{
   position: relative; left:30px;  top:30px;
}
```

Mutlak Pozisyon: Mutlak pozisyonla konumlandırılacak elemanlar, sayfadaki diğer elemanların yerleşimini etkilemez ve Görsel 4.23'te görüldüğü gibi onların yerleşimlerinden de etkilenmez. Mutlak pozisyonla konumlandırılacak elemanlar, göreceli pozisyona sahip bir elaman tarafından kapsanırsa olağan başlangıç noktasına göre değil, kendisini kapsayan elemanın göreceli başlangıç noktasına göre konumlanır.

```
#Kutu{   position:absolute;   left:100px;   top:100px;  }
```

Sıra Sizde

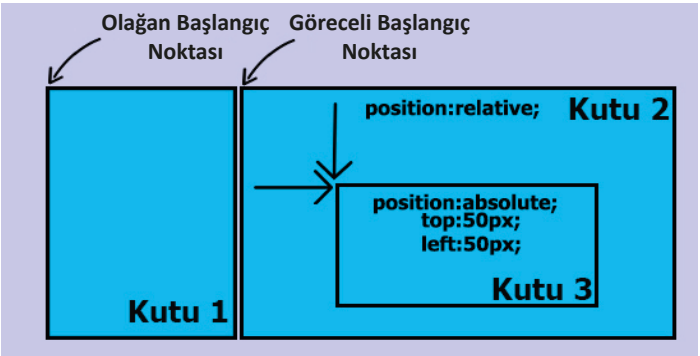
Uygulama 17'deki "goreceliKutu" sınıfının özellikleri üzerinde değişiklik yaparak Görsel 4.24'de görüldüğü gibi "Kutu 3"ü mutlak pozisyon ayarları ile konumlandırınız.



Görsel 4.24: Mutlak pozisyon ayarı-1

18. Uygulama

Göreceli pozisyona sahip bir elamanın içindeki mutlak pozisyona sahip başka bir elemanı konumlandırma işlemlerini Görsel 4.25'te görüldüğü şekilde yönergeler doğrultusunda gerçekleştiriniz.



Görsel 4.25: Mutlak pozisyon ayarı-2

1. Adım: Satır içi görünüme sahip üç adet elemanı oluşturunuz ve elemanları için arka plan rengi ve kenarlık ayarlayınız.

2. Adım: elemanlarına sırasıyla kutu1, kutu2, kutu3 isimlerinde sınıflar ekleyiniz.

3. Adım: kutu1 sınıfı için stil özelliklerini; genişlik 100 piksel, yükseklik 150 piksel olarak ayarlayınız.

4. Adım: kutu2 sınıfı için stil özelliklerini; genişlik 200 piksel, yükseklik 150 piksel, pozisyonu göreceli ayarlayınız.

5. Adım: kutu3 sınıfı için stil özelliklerini; genişlik 120 piksel, yükseklik 70 piksel, pozisyonu mutlak, left özelliği 50 piksel, top özelliği 50 piksel olacak şekilde ayarlayınız.

HTML	CSS

Kutu 1
 Kutu 2
 Kutu 3
 	
	div{
	background-color: lightskyblue;
	display: inline-block; border: 2px solid black;
	}
	.kutu1{ width: 100px; height: 150px; }
	.kutu2{ width: 200px; height: 150px; position: relative; }
	.kutu3{ width: 120px; height: 70px; position: absolute;
	left: 50px; top: 50px;}

Sabit Pozisyon: Sabit pozisyona sahip bir eleman olağan başlangıç noktasına göre konumlandırılır. Sayfa web tarayıcısında aşağıya veya yukarıya kaydırılsa bile sabit pozisyona sahip eleman konumlandığı yerde asılı bir şekilde durur. Bu özellik genellikle menü, uyarı yazısı, reklam gibi ekranda sabit bir yerde durması istenen elemanlar için kullanılır.

```
#Kutu{  position:fixed;  left:100px;  top:100px; }
```

19. Uygulama

Sabit pozisyon ile web sitesi tasarımına aşağı kaydırma tuşu görseli ekleme işlemlerini Görsel 4.26'da görüldüğü şekilde yönergeler doğrultusunda gerçekleştiriniz.

Görsel 4.26: Kaydırma çubuğu

1. Adım: Float konusundaki Uygulama 15'in kaynak kodlarını açınız.

2. Adım: HTML kodlarına <img> elemanıyla aşağı ok işareti resmi ekleyiniz.

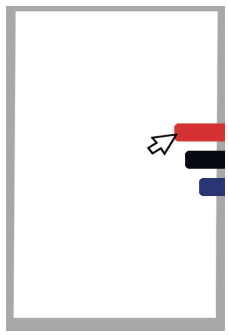
```
<img id="kaydirmaCubugu" src="asagiOk.png">
```

3. Adım: Eklediğiniz <img> elemanını sabit pozisyon ile ekranın sağ alt bölümüne konumlandırınız.

```
#kaydirmaCubugu
{
position: absolute; bottom:60px; right: 20px; width:30px;
}
```

Sıra Sizde

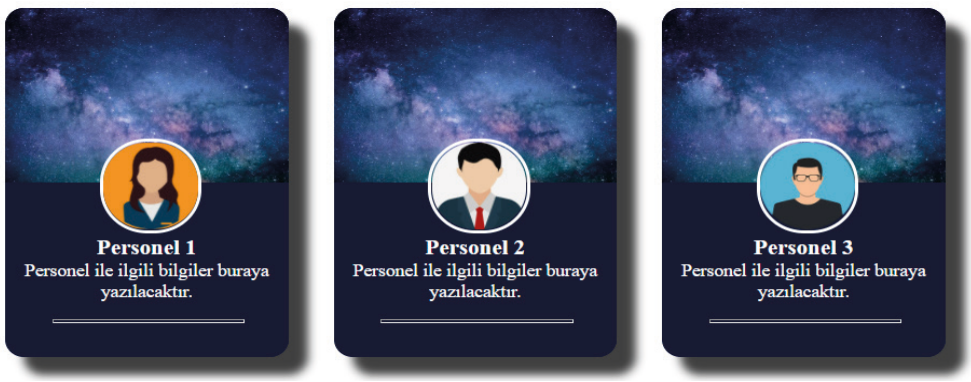
Web sayfasına Görsel 4.27'de görüldüğü gibi farklı boylarda olacak şekilde, kırmızı, siyah ve mavi renklerde sosyal medya bağlantısı için kullanılacak sabit pozisyonlu div elemanları ekleyiniz ve link bağlantılarını yapınız.



Görsel 4.27: Sosyal medya butonları

20. Uygulama

CSS ile özel kart tasarımı yapma işlemlerini Görsel 4.28'de görüldüğü şekilde yönergeler doğrultusunda gerçekleştiriniz.



Görsel 4.28: Özel kart tasarımı

1. Adım: Index.html isimli dosya oluşturunuz ve oluşturduğunuz dosyaya Stil.css dosyasını bağlayınız.

2. Adım: Proje klasörünüze arka plan için bir adet, personel profilleri için üç adet resim ekleyiniz.

3. Adım: <body> elemanı içine arka plan resmi, profil resmi, bilgilendirme yazıları eklemek için aşağıdaki kodları yazınız.

```
<div class="kart">
  <div class="profilSablon"> <img src="personel1.jpg" class="profilResim"> </div>
  <p><b>Personel 1</b><br> Personel ile ilgili bilgiler buraya yazılacaktır.</p><hr>
</div>
 <!-- Diğer personellerin kartları için gerekli kodları buraya yazınız. -->
```

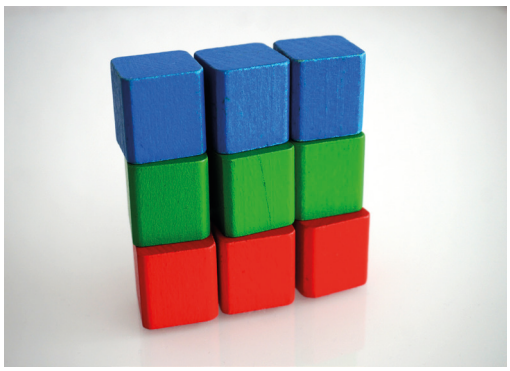
4. Adım: Stil.css oluşturunuz ve dosya içine aşağıdaki kodları ekleyiniz.

```
.kart{
   width: 150px; height:200px; background-color: #021037;
   color:white; border-radius: 10px; text-align: center;
   display: inline-block;  margin-right:20px;
   box-shadow: 10px 10px 5px 0px rgba(0,0,0,0.75);
    }
.profilSablon{
   position: relative; width: 150px; height: 100px;
   background: url(yildizlar.jpg);
   background-size: 150px 100px; background-position: center;
   border-top-left-radius: 10px;
   border-top-right-radius: 10px;
  }
.profilResim{
   position: absolute;  left:50px;   top:75px;
   width: 50px; height: 50px; border-radius: 25px;
   border:2px solid white;   z-index: 2;
         }
 .kart p{ font-size: 10px;  margin-top: 30px; }
 .kart b{ font-size: 12px; }
.kart hr{ width: 100px; margin-left:25px; }
```

4.3. Renk Kullanımı ve Tipografi

Web sitelerinde tasarım için renk seçimi önemlidir. Renkler sitenin içeriğiyle uyumlu olmalıdır. Web sayfasındaki görüntülenen elemanlar için arka plan, çerçeve veya gölge rengi verilebilir. Ayrıca birçok elemana yazı rengi de verilebilir.

Dijital ortamlarda üç ana renk vardır. Bunlar; kırmızı (red), yeşil (green), mavidir (blue) (Görsel 4.29).



Görsel 4.29: Dijital ortamda kullanılan ana renkler

4.3.1. Klasik Renk Tanımlama Çeşitleri

Kullanılacak renkleri belirlemek için renk isimleri, renk fonksiyonları, renk kodları kullanılabilir. Renk fonksiyonları ve renk kodlarıyla 16 milyonun üzerinde farklı tonlarda renk tanımlanabilir.

Renk İsimleri: Renkler CSS tarafından önceden tanımlanmış isimleriyle kullanılabilir. Renk isimleri için genellikle renklerin İngilizce karşılıkları kullanılmaktadır. Ayrıca bir rengin açık tonları için "light", koyu tonları için "dark" sözcüğü kullanılmaktadır.

RGB Fonksiyonu: Kırmızı, yeşil ve mavi renklerinin her birisi için 0-255 arasında değer belirlenip RGB fonksiyonuna yazılmasıyla istenilen renk elde edilir.

RGB(0-255 arası değer , 0-255 arası değer , 0-255 arası değer
RGB(0,0,255))

RGBA Fonksiyonu: RGB fonksiyonundan farklı olarak RGBA fonksiyonuna bir de alpha (saydamlık) değeri girilir. Saydamlık değeri en az "0" en fazla "1" değerini alır. Örneğin, renklerin yarı saydam görünmesi isteniyorsa alpha değeri "0.5" olmalıdır.

RGB(0-255 arası değer , 0-255 arası değer , 0-255 arası değer , 0-1 arası değer
RGB(0,0,255,0.5))

Renk Kodları: #(diyez) işaretinden sonra kırmızı, yeşil, mavi renklerin her birisi için iki basamaklı olacak şekilde onaltılık (Hexadecimal) sayı tabanında değer girilerek renk elde edilir. Renkler için onaltılık sayı tabanında verilebilecek en küçük değer "00", en büyük değer "FF" değeridir (Tablo 4.14).

00 00 00 ⟶ **Siyah renk** FF FF FF ⟶ **Siyah renk**

Tablo 4.14: Renk Çeşitleri

Renk isimleri	RGB Fonksiyonu	Renk Kodları	Ekran Görüntüsü
red	rgb(255,0,0)	#FF0000	Kırmızı
green	rgb(0,255,0)	#00FF00	Yeşil
blue	rgb(0,0,255)	#0000FF	Mavi
pink	rgb(255,192,203)	#FFC0CB	Pembe
yellow	rgb(255,255,0)	#FFFF00	Sarı
purple	rgb(128,0,128)	#800080	Mor
black	rgb(0,0,0)	#000000	Siyah
lightblue	rgb(173,216,230)	#ADD8E6	Açık Mavi
darkorange	rgb(255,140,0)	#FF8C00	Koyu Turuncu

4.3.2. Geçişli Renk Tanımlama

Faklı renk tonlarının doğrusal veya radyal bir şekilde birbirine karıştığı renkler geçişli renklerdir. Geçişli renkler elemanların "background" stil özelliğine uygulanır.

Doğrusal Renk Geçişi: Renk tonlarının belirtilen yönlere (soldan sağa, yukarıdan aşağıya vb.) göre doğrusal bir şekilde geçiş yapması için "linear-gradient" komutu kullanılır. En az iki renk arasında geçiş yapılır.

```
background:linear-gradient(red,white);
background:linear-gradient(red,white,blue);
background:linear-gradient(red,white,blue,black);
```

En basit kullanımı sadece renklerin belirlendiği yukarıdaki örnek kodlarda olduğu gibidir. Hiçbir yön verilmediği için yukarıdan aşağıya doğrusal bir geçiş gerçekleşir.

```
background:linear-gradient(to top,red,white);              /* Aşağıdan yukarıya geçiş */
background:linear-gradient(to right,red,white);            /* Soldan sağa geçiş */
background:linear-gradient(to bottom,red,white);           /* Yukarıdan aşağıya geçiş */
background:linear-gradient(to left,red,white);             /* Sağdan sola geçiş */
background:linear-gradient(to top right,red,white);        /*Sağ üst çapraza  geçiş */
background:linear-gradient(to top left,red,white);         /* Sol üst çapraza geçiş */
background:linear-gradient(to bottom right,red,white); /* Sağ alt çapraza geçiş */
background:linear-gradient(to bottom left,red,white);    /* Sol alt çapraza geçiş */
```

Renkler arası geçişler, yukarıdaki örnek kodlarda görüldüğü gibi geçiş yönü belirterek de yapılabilir. Renk sayısı ikiden fazla olabilir.

```
background:linear-gradient(0deg,red,white);
background:linear-gradient(90deg,red,white);
background:linear-gradient(180deg,red,white);
```

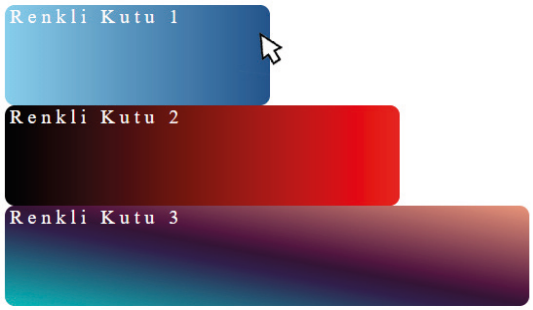
Renkler arası geçişlerin yönü, yukarıdaki örnek kodlarda görüldüğü gibi “0-180” arasında herhangi bir açı değeri ile de belirlenebilir.

Not

Web sayfasına en uygun renk geçiş kodları, çevrim içi olarak hizmet veren web sitelerinden kolayca üretilebilir.

21. Uygulama

CSS ile klasik ve geçişli renk tanımlayarak kutu renklendirme işlemlerini Görsel 4.30’da görüldüğü şekilde yönergeler doğrultusunda gerçekleştiriniz (Kutuların üzerine fare ile gelindiğinde kutunun genişliğini 0,5 saniyede animasyonlu bir şekilde 1000 piksel yapınız.).



Görsel 4.30: Renkli kutular

1. Adım: Index.html isimli dosya oluşturunuz ve oluşturduğunuz dosyaya Stil.css dosyasını bağlayınız.

2. Adım: Üç adet elemanı ekleyiniz ve içlerine "Renkli Kutu" yazınız.

```
<div class="gecisli1">    Renkli Kutu 1   </div>
<div class="gecisli2">    Renkli Kutu 2   </div>
<div class="gecisli3">    Renkli Kutu 3   </div>
```

3. Adım: elemanlarının arka plan renkleri için Stil.css dosyasına aşağıdaki kodları ekleyiniz.

```
.gecisli1{
  width: 20%; background: linear-gradient(to right, skyblue , #23538A);}
.gecisli2{
  width: 30%; background: linear-gradient(to left, red,black);}
.gecisli3{
  width: 40%;
  background: linear-gradient(to right top, #0cbaba, #380036, darksalmon); }
```

4. Adım: elemanlarının genel özellikleri; yükseklik 100 piksel, yazı rengi beyaz, yazı boyutu 20 piksel, kenar yuvarlaklığı 10 piksel olacak şekilde ayarlayınız.

```
div{ height: 100px; color:white; font-size:20px; border-radius: 10px; }
```

5. Adım: Kutuların üzerine fare ile gelindiğinde 0.5 saniyede kutu genişliğini 1000 piksel olacak şekilde ayarlayınız.

```
div:hover{
   width: 1000px; cursor:hand;
   transition-property: width; transition-duration:0.5s; transition-delay:0.1s;}
```

4.3.3. Tipografi

Tipografi, web sayfalarında, kitaplarda, gazetelerde, dergilerde, televizyon programlarında ve daha birçok alanda metinsel ifadelerin belirli bir formatta sunulması için kullanılan bir kavramdır. Tasarımlarda metinsel ifadeler için bir format belirlemek kullanıcı, okuyucu veya izleyicileri etkilemek için oldukça önemlidir. **Yazı rengi, yazı tipi, yazı boyutu, yazı vurgusu, yazı stili, yazı gölgesi, yazı hizası, yazı dekorasyonu, satır yüksekliği, harf aralıkları, büyük veya küçük harfe dönüşüm, madde işaretleri** gibi özellikler ile özel bir yazı formatı (tipografi) oluşturulabilir. CSS kodları arasında tipografiyi etkileyecek birçok kod bulunmaktadır (Tablo 4.15).

Not

Margin, padding, background, border gibi özellikler de dolaylı da olsa tipografiyi etkiler.

Tablo 4.15: Tipografi Kodları

CSS Kodu ve Kullanımı	Açıklama
color: blue	Yazının rengini "mavi" yapar.
font-family: arial	Yazı tipini "arial" yapar.
font-size: 16px	Yazı boyutunu "16 piksel" yapar.
font-weight: bold	Yazıyı kalınlaştırır.
font-style: italic	Yazıyı eğik hâle getirir (None değeri de alabilir.).
text-shadow: 51px 71px 31px yellow	5 piksel yatay eksende, 7 piksel dikey eksende gölgeyi konumlandırır. Gölgenin bulanıklığı 3 pikseldir ve gölge sarı renklidir.
text-align: left	Yazıyı ve diğer elemanları sola yaslar (Center ve right vb. değerler de alabilir.).
text-decoration: none	Yazının çizgisini kaldırır (Underline değeri de alır).
line-height: 1.5em	Satır yüksekliğini ayarlar (Piksel cinsinden de değer alabilir.).
letter-spacing: 0.25em	Harfler arasındaki mesafeyi ayarlar (Piksel cinsinden de değer alabilir.).
text-transform: uppercase	Yazıların harflerini tamamını büyütmek, küçültmek veya sadece baş harflerini büyütmek için kullanılır (Lowercase, capitalize değerlerinin alabilir.).
list-style-type:square	Listeleme elemanlarının işaretlerini ayarlar (Circle, upper-roman, lower-alpha değerleri de alabilir.).

22. Uygulama

Yayımlanacak makaleyi tipografi kodları kullanarak biçimlendirme işlemlerini Görsel 4.31'de görüldüğü şekilde yönergeler doğrultusunda gerçekleştiriniz.

Görsel 4.31: Makale

1. Adım: Index.html isimli dosya oluşturunuz ve oluşturduğunuz dosyaya Stil.css dosyasını bağlayınız.

2. Adım: <body> elemanı içine makale başlığı, makale yazarı, makale içeriği ve makale yazılma tarihi eklemek için aşağıdaki kodları yazınız.

```
div class="makale">
   <h2>Bugün siz de <span> yazılıma başlayarak hayatınızı değiştirin.</span></h2>
   <div class="yazar_tarih">Konu Yazarı: Zeynep Sare</div>
   <div  class="icerik">
      <p>Günümüzde yazılım, tüm dünyayı yakından ilgilendiren  bir kavramdır.</p>
      <p>Hiç düşündünüz mü, "hayatımızdan yazılımı çıkarsaydık ne olurdu?".
         Sanırım hareket kabiliyetimiz kısıtlanır, yaşam kalitemiz düşer,
          iletişim kurmada problemler yaşar
         ve daha birçok konuda sıkıntılı bir hayatımız olurdu.</p>
      <div class="yazar_tarih
      ">Oluşturma: 25 Şubat 2021 Perşembe</div>
   </div>
</div>
```

3. Adım: Stil.css oluşturunuz ve dosya içine aşağıdaki kodları ekleyiniz.

```
.makale { width: 618px; margin:auto}
h2 { font-size: 2.5em; font-family: Georgia; text-shadow: 1px 1px 1px gainsboro;
  letter-spacing: 0.1em; color: rgb(231, 62, 46);  }
h2 span { display: block; margin-top: 0.5em; font-family: Verdana;
  font-size: 0.6em; font-weight: normal; letter-spacing: 0em; text-shadow: none; }
.yazar_tarih { font-family: Georgia; color: dimgray; font-size: 0.85em;
  font-style: italic; letter-spacing: 0.25em; padding-bottom: 0.5em;
  border-bottom: 1px solid gainsboro;  }
.yazar_tarih span { text-transform: capitalize; font-style: normal; color: #999; }
.icerik p { font-family: Verdana;   line-height: 1.5em; color: black }
.icerik p:first-child { font-size: 1.25em; font-family: Georgia;
  font-style: italic;  letter-spacing: 0.1em; }
.icerik p:first-child:first-line { font-weight: bold; }
```

4.4. Duyarlılık (Responsivity)

Günümüz web sitelerinde en çok aranan özellik web sitesinin duyarlı bir tasarımı olmasıdır. **Duyarlılık;** tasarımın bilgisayar, tablet, akıllı telefon, akıllı televizyonlar ve hatta akıllı saatlerin ekran çözünürlüklerine uygun olmasıdır (Görsel 4.32). Örneğin; bir bilgisayar ekranı aracılığıyla görüntülenen web sitesinde, yan yana genişliği 250 piksel olan beş adet resim rahatlıkla görüntülenebilirken aynı web sitesi cep telefonu ekranında görüntülendiğinde resimler ancak alt alta rahatlıkla görüntülenebilir. Buradaki önemli nokta, kullanıcıların web sitesinin içeriğini rahatlıkla görüntüleyebilmeleridir. Media sorgulama kodları ile hangi ekran çözünürlüğüne sahip cihazdan web sitesine girildiği öğrenilir ve o ekran çözünürlüğüne uygun CSS kodları devreye girer.

Görsel 4.32: Duyarlı web sitesi

4.4.1. Medya Sorgusu

Media sorgusu için @media komutu şart ifadeleri ile birlikte kullanılır. Birden fazla şart ifadesi yazılırken aralarına and (ve) operatörü eklenir. @media komutu çoğunlukla "screen" şart ifadesi ile birlikte kullanılır. Bilgisayar, tablet, akıllı telefon gibi cihazlar, "screen" şart ifadesiyle sorgulanır. Diğer şart ifadeleri için **width, height, min-width, min-height, max-width, min-height, device-width, min-device-width, min-device-height, max-device-width, min-device-height** özellikleri kullanılabilir. Genişlik, yükseklik şart ifadeleri **parantez içinde yazılır.**

```
@media screen and (min-width:600px)
 {
  body { background-color:#e76f51; }
 }
```

"(min-width: 600px)" şart ifadesi web sayfasının gösterildiği **tarayıcı genişliği** en az 600 piksel ise stil kodlarının çalışacağını belirtir. Örneğin bir bilgisayar tarayıcısının genişliği 768 piksel ise web sitesi için yukarıdaki stil kodları **çalışacaktır**.

```
@media screen and (min-device-width: 800px)
  {
   body { background-color: #e76f51; }
  }
```

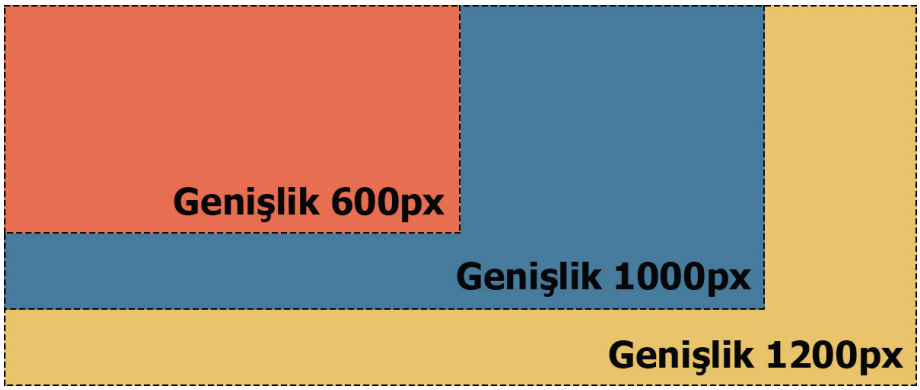
"(min-device-width: 800px)" şart ifadesi ise web sayfasının gösterildiği cihazın **ekran genişliği** en az 800 piksel ise stil kodlarının çalışacağını belirtir. Örneğin bir telefonun ekran genişliği 500 piksel ise web sitesi için yukarıdaki stil kodları **çalışmayacaktır**.

```
@media screen and (min-height: 100px) and (max-height:800px)
                  and (min-width: 100px)  and (max-width:1024px)
  {
   body { background-color: #e76f51;  }
  }
```

Yukarıdaki örnek kodlarda olduğu gibi ikiden fazla şart ifadesi kullanarak sadece belirli bir aralıktaki genişlik ve yükseklik değerlerine göre çalışan stil kodları yazılabilir.

23. Uygulama

Medya sorgulama kodları kullanarak farklı ekran boyutlarına göre web sayfasına stil verme işlemlerini Görsel 4.33'te görüldüğü şekilde yönergeler doğrultusunda gerçekleştiriniz.



Görsel 4.33: Duyarlı arka plan renkleri

1. Adım: Responsive.css ve index.html isimlerine sahip iki dosya oluşturunuz.
2. Adım: index.html dosyasına Responsive.css dosyasını bağlayınız ve arka plan rengini #a8dadc yapınız.
3. Adım: Arka plan rengini; tarayıcı genişliği en az 1200 piksel ise #e9c46a, en az 800 piksel ise #457b9d, en az 600 piksel ise #e76f51 yapınız.

CSS

```
body {  background-color: #a8dadc; color:white; }
@media screen and (min-width: 600px)
      {
       body{ background-color: #e76f51; }
      }
@media screen and (min-width: 800px)
      {
       body{ background-color: #457b9d; }
      }
@media screen and (min-width: 1200px)
      {
       body{ background-color: #e9c46a; }
      }
```

24. Uygulama

Medya sorgulama kodları kullanarak farklı ekran boyutlarına göre web sayfasına stil verme işlemlerini yönergeler doğrultusunda gerçekleştiriniz.(Görsel 4.34)

Görsel 4.34: Duyarlı reklam görseli

1. Adım: Çözünürlüğü 215x120 piksel ve çözünürlüğü 750x240 piksel olan 2 adet reklam görselini Görsel 4.36'daki gibi tasarlayınız.

2. Adım: HTML sayfasına etiketi içine reklamlara ait iki adet resim ekleyiniz.

3. Adım: elemanı için "class seçici" resim elemanları için "id seçici" ekleyiz.

4. Adım: Web sayfasının görüntülendiği cihazın ekran genişliği en az 215 piksel ise küçük boyuttaki reklamı gösteriniz, ekran genişliği en az 750 piksel ise büyük boyuttaki reklamı gösteriniz.

HTML

```
<div class="reklamAlani">
  <img src="reklam1.jpg" id="kucukReklam">
  <img src="reklam2.jpg" id="buyukReklam">
</div>
```

CSS

```
div{text-align: center;}
@media screen and (min-device-width:215px)
{
  #kucukReklam{display: inline;}
  #buyukReklam{display:none;}
}
@media screen and (min-device-width:750px)
{
   #buyukReklam{display:inline;}
   #kucukReklam{display: none;}
}
```

4.4.2. Popüler CSS Frameworkleri

CSS ile tasarım yaparken kişiselleştirilmiş sınıflar, sınıfların birleşiminden oluşan kütüphaneler veya daha kapsamlı bir yapı olan frameworkler tanımlanabilir. Tüm bu kişiselleştirilmiş yapılar, web site tasarımlarında tekrar tekrar kullanılabilir. Ayrıca başka programcılar tarafından tanımlanmış yapılar da kullanılabilir.

Tasarımı kolayca tüm cihazlara uygun hâle getirmek ve görsel açıdan zenginleştirmek için hazırlanmış birçok popüler CSS frameworkü bulunmaktadır. Bu frameworklerin yapısında çok kullanışlı sınıflar bulunmaktadır ve bu frameworklerin çoğu, açık kaynak kodludur ve ücretsizdir. Bootstrap, Foundation, Matearialize, Ui Kit gibi CSS frameworkleri bulunmaktadır.

4.4.3. Bootstrap Framework

Bootstrap, çok tercih edilen CSS frameworklerindendir. Bootstrap ile ilgili içeriklere kendi web adresinden ücretsiz olarak ulaşılabilir. Bootstrap son sürümü beta (test) aşamasında olduğu için kitap içeriğinde Bootstrap 4 işlenecektir (Görsel 4.35).

Görsel 4.35: Bootstrap logo

Bootstrap Framework özellikleri;

- Kolay ve anlaşılabilir kod yapısı,
- İçeriği ile ilgili kaynak, doküman ve örneklere ulaşma kolaylığı,
- Neredeyse tüm tarayıcılarla uyumlu çalışması,
- Farklı ekran çözünürlüklerine uygun tasarım yapma olanağı,
- Grid, tipografi, slider gibi işlevsel birçok yapı ve sınıfı kullanma imkânı,
- Birçok eleman için Javascript ile görsel efekt desteği,
- Açık kaynaklı olduğu için içindeki kaynak kodlarına müdahale edebilme olanağı,
- Her geçen gün güncellenen ve geliştirilen bir framework olmasıdır.

Bootstap Framework'ü web sitelerinde kullanmak için gerekli bağlantıların yapılması gerekmektedir. Bağlantı linklerine Bootstrap Framework web adresinden ulaşılabilir.

25. Uygulama

Haricî CSS sayfası ve haricî Javascript sayfası bağlantı yöntemlerini kullanarak Bootstrap Framework'ü web sayfasında kullanabilmek için gereken işlemleri yönergeler doğrultusunda gerçekleştiriniz.

1. Adım: Bootstrap Framework web adresine giriniz.
2. Adım: Web sitesinin sağ üst köşesindeki **Download** butonuna tıklayınız.
3. Adım: Bootstrap versiyonunu **v4.6** olarak seçiniz.
4. Adım: Açılan sayfada CSS başlığı altındaki **<link>** kodunu kopyalayınız ve HTML sayfanızdaki **<head>** etiketi içine yapıştırınız. Aynı sayfadaki **Bundle** başlığı altındaki **<script>** kodlarını kopyalayınız ve HTML sayfanızdaki body bölümünün kapatma etiketinden (</body>) hemen önceki satıra yapıştırınız.

```
<head>
   <!--  Diğer kodlar  -->
   <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/css/
bootstrap.min.css">
</head>
<body>
   <!--  Diğer kodlar  -->
  <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist
/js/bootstrap.bundle.min.js"></script>
</body>
```

5. Adım: HTML sayfanızdaki <body> etiketi içine aşağıdaki kodları yapıştırınız ve web sitesini çalıştırınız.

```
<div class="alert alert-success">
  <strong>Başardınız!</strong> Artık Bootstrap 4 Framework kullanıyorsunuz.
</div>
```

Not

Uygulamadaki kod yapısı tüm HTML sayfalarında şablon olarak kullanılabilir. Böylece tüm sayfalarda Bootstrap Framework özelliklerinden faydalanılabilir.

4.4.3.1. Konteyner Çeşitleri

Ekranın genişliğine duyarlı kapsayıcı bir div oluşturmak için "container" sınıfı kullanılır. İki farklı kullanım şekli bulunmaktadır.

Bootstrap Framework'ü ile ekran çözünürlükleri beş faklı boyutta tanımlanır. Bu özellik ile tüm çözünürlüklerdeki cihazlar için tasarım yapılabilir (Tablo 4.16).

Tablo 4.16: Ekran Boyutları

Genişlik	Boyut
<576 piksel	En küçük
>=576 piksel	Küçük
>=768 piksel	Orta
>=992 piksel	Geniş
>=1200 piksel	En geniş

Sabit Konteyner: Ekranın tüm genişliğini kaplamayan kenarlardan boşlukları olan konteyner çeşididir (Görsel 4.36). Farklı iki ekran boyutu arasında konteyner genişliği sabit kalır. Kenar boşlukları değişir.

```
<div class="container">
  <h3>Konteyner çeşitleri</h3>
  <p>Ekranın tüm genişliği kaplanmadı.</p>
</div>
```

Fluid Konteyner: Farklı çözünürlüklerde ekranın tüm genişliğinin kaplandığı konteyner çeşididir (Görsel 4.36).

```
<div class="container-fluid">
  <h1>Konteyner çeşitleri</h1>
  <p>Ekranın tüm genişliği kaplandı.</p>
</div>
```

Görsel 4.36: Container sınıfı

4.4.3.2. Renk Sınıfı

Bootstrap Framework ile hızlıca yazı ve arka plan renklerini değiştirmek için "text" ve "bg" sınıfları bulunmaktadır. Bu sınıflar için tanımlanmış birçok renk bulunmaktadır (Tablo 4.17).

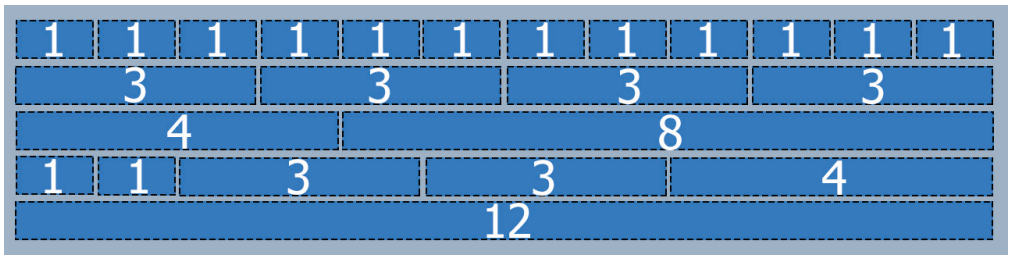
Tablo 4.17: Renk Sınıfları

Yazı Renkleri	Arka Plan Renkleri	Renk Görünümleri
text-muted		Bu yazı mat renktedir.
text-primary	bg-primary	
text-success	bg-success	
text-info	bg-info	
text-warning	bg-warning	
text-danger	bg-danger	
text-secondary	bg-secondary	
text-dark	bg-dark	
text-white	bg-white	
text-light	bg-light	

4.4.3.3. Grid Yapısı

Bootstrap grid yapısı ile ekran çözünürlüğüne göre boyutları yeniden düzenlenen sütunlar kullanılır (Tablo 4.18). Bu yapıya göre ekran 12 farklı sütuna ayrılmış kabul edilir. Bu sütunları bazen

birleşik bazen ayrı kullanmak mümkündür. Sütun grupları kapsayıcı bir elemanı içinde kullanılır. Kapsayıcı elemanının içine istenildiği kadar sütun konulabilir fakat bir sıradaki toplam sütun sayısı on ikiyi (12'yi) geçtiğinde sonraki sütun bir alt satıra yerleşir (Görsel 4.37). Ekranı sütunlara ayırırken içeriğin okunabilir olmasına dikkat edilmelidir. Örneğin cep telefonu ekranı 6 sütuna ayrılırsa içerik okunmaz hâle gelebilir. Genellikle cep telefonu ekranları en çok iki sütuna ayrılmaktadır.



Görsel 4.37: Grid yapısı taslak görünüm

Tablo 4.18: Grid Yapısı

Genişlik	Boyut	Sınıf Adı
<576 piksel	En küçük	col-
>=576 piksel	Küçük	col-sm-
>=768 piksel	Orta	col-md-
>=992 piksel	Geniş	col-lg-
>=1200 piksel	En Geniş	col-xl-

Web sitesi tasarımına göre aynı elemanı için bir veya daha fazla "col" sınıfı tanımlanabilir. İyi bir tasarım için tüm ekran boyutlarına özel "col" sınıfı tanımlanmalıdır.

```
<div class="row">
  <div class="col bg-info">    Kutu 1 </div>
  <div class="col bg-danger">  Kutu 2 </div>
  <div class="col bg-dark">    Kutu 3 </div>
  <div class="col bg-warning"> Kutu 4 </div>
</div>
```

Not

Ekranı bölümlere ayırmak amacıyla tanımlanan "col" sınıflarından en geniş çözünürlük hangi sınıfa ait ise "col" sınıfı tanımlanmayan diğer çözünürlükler için ekran, o sınıfa göre ayrılır. Yukarıdaki örnekte tüm ekran boyutları için elemanlarının dördü de yan yana dizilecektir.

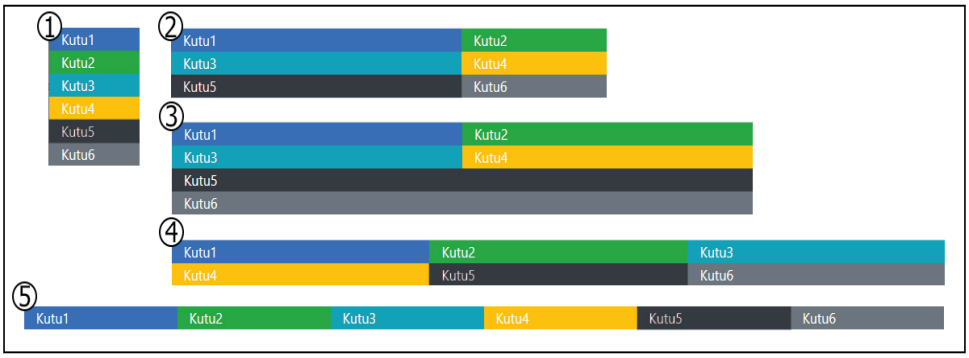
```
<div class="row">
  <div class="col-sm-6 col-md-3 bg-info">    Kutu 1 </div>
  <div class="col-sm-6 col-md-3 bg-danger">  Kutu 2 </div>
</div>
```

Not

Ekran geniş ve en geniş boyutları için “col-lg”, “col-xl” sınıfı tanımlanmadığı için geniş ve en geniş boyutlarda “col-md” sınıfına göre ekran ayrılacaktır.

26. Uygulama

Bootstrap grid yapısını kullanarak ekranı Görsel 4.38’de görüldüğü şekilde bölümlere ayırma işlemlerini yönergeler doğrultusunda gerçekleştiriniz.



Görsel 4.38: Grid yapısı gerçek görünüm

1. Adım: HTML sayfasında Bootstrap Framework kullanımı için gerekli bağlantıları yapınız.

2. Adım: HTML sayfasında etiketi açınız ve içine aşağıdaki kodları yapıştırınız.

```
<div class="row">
 <div class="col-12 col-sm-8 col-md-6 col-lg-4 col-xl-2  bg-primary">Kutu1</div>
 <div class="col-12 col-sm-4 col-md-6 col-lg-4 col-xl-2  bg-success">Kutu2</div>
 <div class="col-12 col-sm-8 col-md-6 col-lg-4 col-xl-2  bg-info">Kutu3</div>
 <div class="col-12 col-sm-4 col-md-6 col-lg-4 col-xl-2  bg-warning">Kutu4</div>
 <div class="col-12 col-sm-8 col-md-6 col-lg-4 col-xl-2  bg-dark">Kutu5</div>
 <div class="col-12 col-sm-4 col-md-6 col-lg-4 col-xl-2  bg-secondary">Kutu6</div>
</div>
```

3. Adım: Web sayfasını çalıştırınız ve tarayıcının genişliğini değiştirerek web sayfasının Görsel 4.38’deki gibi farklı boyutlardaki tasarımlarını görüntüleyiniz.

4. Adım: Hangi tasarımın hangi çözünürlüklerde görüntülendiğini maddeler hâlinde yazınız.

1. Genişliği 576 pikselden az çözünürlüklerdeki görünüm
2. Genişliği 576 piksel ile 768 piksel arası çözünürlüklerdeki görünüm
3. Genişliği 768 piksel ile 992 piksel arası çözünürlüklerdeki görünüm
4. Genişliği 992 piksel ile 1200 piksel arası çözünürlüklerdeki görünüm
5. Genişliği 1200 pikselden büyük çözünürlüklerdeki görünüm

Sıra Sizde

Aşağıdaki kodları HTML sayfasında içine yazınız ve farklı ekran boyutları için ayrı ayrı resimlerin nasıl görüntüleneceğini yazınız (Örneğin en küçük boyut için alt alta dört resim görüntülenir.).

```
<div class="row”>
  <div class=" col-12 col-sm-6 col-md-4  col-lg-4 col-xl-3">
    <img src="kare.png" style="width:100px; height:100px;"></div>
  <div class="col-12 col-sm-6 col-md-4  col-lg-4 col-xl-3">
    <img src="ucgen.png" style="width:100px; height:100px;"></div>
  <div class=" col-12 col-sm-6 col-md-4  col-lg-4 col-xl-3">
    <img src="daire.png" style="width:100px; height:100px;"></div>
  <div class="col-12 col-sm-6 col-md-4  col-lg-4 col-xl-3">
    <img src="dikdortgen.png" style="width:100px; height:100px;"></div>
</div>
```

4.4.3.4. Tipografi Yapısı

Bootstrap Framework’ün tipografiyi etkileyen sınıfları ve özel elemanları bulunmaktadır. Varsayılan olarak yazı boyutları 16 piksel, yazı tipi “Helvatica veya Arial”, satır genişliği 1.5em olarak ayarlanmıştır.

<small> Elemanı: HTML başlık elemanları içinde kullanılır. Başlığa göre daha açık yazı renginde ve yazı boyutu başlıktan daha küçük olacak şekilde metin biçimlendirmesi yapar.

<blockquote> Elemanı: Farklı kaynaklardan alıntı yapılan metinleri biçimlendirir.

<dl> Elemanı: HTML <ul> etiketi gibi liste oluşturmak için kullanılır. Fakat <dl> elemanı içine başlık elemanı için <dt> ve başlığın içeriği için <dd> elemanları kullanılır.

<kbd> Elemanı: Metin içerisinde dikkat çekilmek istenilen içerik <kbd> elamanı biçimlendirilebilir.

<pre> Elemanı: HTML kodları arasına yazılan metinler kod yazımı esnasında belli bir düzenle alt alta da yazılsa kodlar çalıştığında satıra sığana kadar metinler yan yana görüntülenecektir. <pre> elamanı içine yazılan tüm metinler, kodlama esnasındaki düzenleri ile görüntülenir.

27. Uygulama

Bootstrap Framework’e özel tipografi elemanlarını kullanarak metin biçimlendirme işlemlerini yönergeler doğrultusunda gerçekleştiriniz.

1. Adım: HTML sayfası oluşturunuz ve Bootstrap Framework bağlantı kodlarını ekleyiniz.

2. Adım: <body> elemanı içine ekranı tüm çözünürlüklerde tek parçaya ayıracak aşağıdaki kodları ekleyiniz.

```
<div class="container">
   <div  class="row bg-light">
  <div class="col-12">
     <!-- Kodlarınız -->
    </div>
  </div>
</div>
```

3. Adım: Aşağıdaki numaralandırılmış kodları sırasıyla Adım 2'de bulunan <!-- Kodlarınız --> ifadesi yerine yazınız ve web sitesini çalıştırınız.

```
1.<h3>Büyük Başlık <small> Küçük Başlık</small></h3>
2.<blockquote class="blockquote">
  <p>HTML bir web sitesinin vücudunu oluşturur.CSS ise o vücudun üzerindeki elbiseleri, aksesuarları ve makyajı oluşturur.</p>
  <footer class="blockquote-footer">11.Sınıf Web Tabanlı Uygulama Geliştirme Kitabı</footer>
  </blockquote>
3.<dl>
   <dt>HTML, CSS</dt>
    <dd>-Ön Uç(Frontend) dillerdir.</dd>
   <dt>ASP, PHP</dt>
    <dd>-Arka Uç(Backend) dillerdir</dd>
   </dl>
4.<p>Yazdığınız kodları sık sık <kbd>ctrl + s</kbd> ile kaydetmelisiniz.</p>
5.<pre> Bu etiket kullanmadan yazılan
        metin içerikleri satır sonuna kadar
        yan yana yazılır.</pre>
```

4.4.3.5. Tablo Sınıfı

Bootstrap "table" sınıfı ile görsel açıdan zenginleştirilmiş tablolar oluşturulabilir (Tablo 4.19).

Tablo 4.19: Tablo Sınıfı

Sınıf Adı	Açıklama	Kullanımı
table	Table sınıfı özelliklerini kullanmak için tek başına veya diğer table alt sınıflarıyla birlikte kullanılabilir.	<table class="table">
table-sm	Satır yüksekliği azaltılmış bir tablo için kullanılır.	<table class="table table-sm">
table-bordered	Tablonun tüm hücrelerine kenarlık ekler.	<table class="table table-bordered">
table-borderless	Tablodaki tüm kenarlıkları kaldırır.	<table class="table table-borderless">
table-hover	Tablo satırları üzerine fare ile gelindiğinde satıra arka plan rengi verir.	<table class="table table-hover">
table- striped	Tablo satırlarının arka plan renklerini bir satır arayla farklı renk tonuyla gösterir.	<table class="table table-striped">
table-dark ve diğer renk sınıfları	Tablonun satırlarının veya sütunlarının arka plan rengini istenilen renk sınıfına göre ayarlar.	<table class=”table table-dark”> <table class=”table table-primary”> <tr class=”table table-info”> <td class=”table table-danger”>
thead-dark	Tablo başlık bölümünün arka plan rengini değiştirir.	<thead class=”thead-dark”>

Not

Table sınıfı kullanılırken birden fazla sınıfı birlikte kullanmak mümkündür.

28. Uygulama

Bootstrap "table" sınıfı kullanarak tablo biçimlendirme işlemlerini Görsel 4.39'da görüldüğü şekilde yönergeler doğrultusunda gerçekleştiriniz.

Ad	Soyad	Şehir
Melih	BOZKURT	Yozgat
Hüma	AVCI	Bursa
Ayşegül	KAYA	Tokat
Beyza	ASLANHAN	Bayburt

Görsel 4.39: Tablo yapısı

1. Adım: HTML sayfası oluşturunuz ve Bootstrap Framework bağlantı kodlarını ekleyiniz.

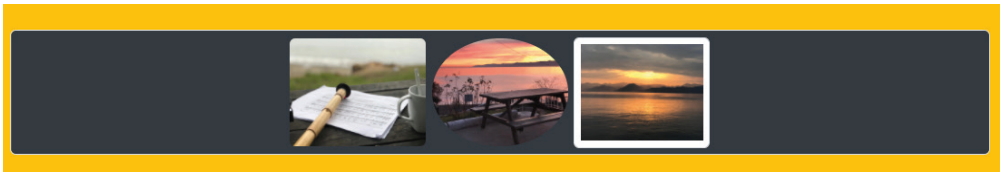
2. Adım: Ad, Soyad, Şehir başlıkları bulunan üç sütun ve beş satırlık bir tablo oluşturunuz.

3. Adım : Tablonun satırları üzerine fare ile gelindiğinde satırın arka plan rengini değiştiriniz. Tablonun arka plan rengini primary yapınız. Tablonun her tarafına kenarlık ekleyiniz. Tablodaki satır yüksekliklerini azaltınız. Başlık satırının arka plan rengini dark yapınız ve tablonun iki farklı hücresinin arka plan rengini değiştiriniz.

```
<table class="table table-hover table-primary table-striped table-bordered table-sm">
   <thead class="thead-dark">
     <tr>  <th>Ad</th> <th>Soyad</th> <th>Şehir</th> </tr>
   </thead>
   <tbody>
    <tr> <td>Melih</td> <td>BOZKURT</td> <td>Yozgat</td> </tr>
    <tr> <td class="table-danger">Hüma</td> <td>AVCI</td> <td>Bursa</td> </tr>
    <tr class="table-light"> <td>Ayşegül</td> <td>KAYA</td> <td>Tokat</td> </tr>
    <tr> <td class="table-warning">Beyza</td> <td>ASLANHAN</td> <td>Bayburt</td>
</tr>
  </tbody>
</table>
```

4.4.3.6. Kenarlık Sınıfları

Kenarlık sınıfları eklendiğinde elemanların görünümleri Görsel 4.40'da görüldüğü gibi üç farklı şekilde değişmektedir. Kenarlık sınıfları <img>, ve daha birçok elemana eklenmektedir. Kenarlık çeşitleri ile ilgili önemli bilgiler Tablo 4.20'de verilmiştir.



Görsel 4.40: Kenarlık çeşitleri

Tablo 4.20: Kenarlık Çeşitleri

Sınıf Adı	Açıklama	Kullanımı
rounded	Eklendiği elemanın köşelerini yuvarlar.	<img src=”resim1.jpg” class=”rounded”> <div class=”col-sm-2 img-rounded”>
rounded-circle	Eklendiği elemanın en boy oranına göre tam daire veya elips şeklinde bir görünüme sahip olmasını sağlar.	<img src=”resim1.jpg” class=”rounded-circle”> <div class=”col-lg-6 rounded-circle “>
img-thumbnail	Eklendiği elemanın hem çerçeveli hem de köşeleri yuvarlanmış olmasını sağlar.	<img src=”resim1.jpg” class=”img-thumbnail”> <div class=”col-md-6 img-humbnail”>

4.4.3.7. Jumbotron Sınıfı

Web sitesi içinde vurgu yapılacak özel bir içeriği yansıtmak için kullanılabilecek özel bir kutu oluşturur. Kutunun rengi varsayılan olarak gridir. Jumbotron kutusu içinde neredeyse tüm HTML elemanları kullanılabilir (Görsel 4.41).

Görsel 4.41: Jumbotron sınıfı

```
<div class="jumbotron ">
  <img src="KahveBilgisayar.jpg" style="width:100px;" class="img-thumbnail">
    <h1>Yazılım</h1>
    <p>Hayallerinizdeki uygulamaları geliştirmek için sadece bilgisayar ve kahveye ihtiyacınız var.</p>
</div>
```

4.4.3.8. Uyarı Mesajı Sınıfı

Alert sınıfı ile primary, success, info, warning, danger, secondary, dark, light arka plan renklerine sahip, dikkat çekici mesaj kutuları oluşturulabilir (Görsel 4.42). En temel kullanım şekli aşağıdaki örnek kodlarda görülmektedir.

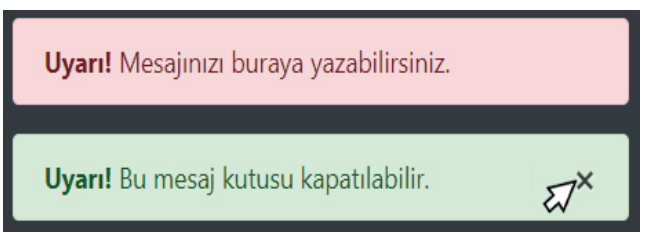
```
<div class="alert alert-danger">
  <strong>Uyarı! </strong> Mesajınızı buraya yazabilirsiniz.
</div>
```

Alert-dismisible sınıfı ile mesaj kutusu kapatılabilir bir kutuya dönüştürülür. Aşağıdaki örnek kodda görüldüğü gibi kapatma tuşu için bir <button> elemanı eklenmelidir.

```
<div class="alert alert-success alert-dismissible">
   <button type="button" class="close" data-dismiss="alert">&times;</button>
   <strong>Uyarı!</strong> Bu mesaj kutusu kapatılabilir.
</div>
```

Not

<button> elemanının içine yazılan "×" ifadesi yerine "kapat" veya başka ifadeler de yazılabilir.



Görsel 4.42: Uyarı mesajı

4.4.3.9. Buton Sınıfı

<button> , <input> ve <a> elemanları "btn" sınıfı eklenilerek özelleştirilmiş görünümlere sahip butonlar hâline dönüştürülür. Tüm buton çeşitleri aşağıdaki örnek kodlarda, butonların görünümleri de Görsel 4.43'te verilmiştir.

```
<button type="button" class="btn btn-primary">Buton elemanı</button>
<input type="button" class="btn btn-info" value="Input elemanı">
<a href="#" class="btn btn-warning">Link elemanı</a>
<button type="button" class="btn btn-outline-light">Çerçeveli</button>
<button type="button" class="btn btn-primary btn-lg">Geniş</button>
<button type="button" class="btn btn-primary">Normal</button>
<button type="button" class="btn btn-primary btn-sm">Küçük</button>
<button type="button" class="btn btn-success active">Aktif Buton</button>
<button type="button" class="btn btn-success" disabled>Pasif Buton</button>
<button type="button" class="btn btn-info btn-block">Blok Buton</button>
```

Not

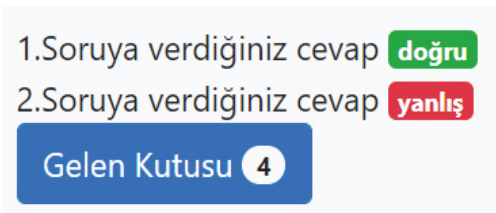
Primary, success, info, warning, danger, secondary, dark, light arka plan renklerine sahip butonlar tasarlanabilir.

Görsel 4.43: Buton çeşitleri

4.4.3.10. Badge Sınıfı

Web sayfasında dikkat çekmesi istenilen içeriği çeşitli renklerde işaretlemek için "badge" sınıfı kullanılır (Görsel 4.44).

```
1.Soruya verdiğiniz cevap <span class="badge badge-success">Doğru</span> <br>
2.Soruya verdiğiniz cevap <span class="badge badge-danger">Yanlış</span> <br>
<button type="button" class="btn btn-primary">
  Gelen Kutusu <span class="badge badge-light badge-pill">4</span>
</button>
```



Görsel 4.44: Badge çeşitleri

4.4.3.11. Kart Sınıfı

Kart sınıfı ile resim, başlık, içerik ve istenilen elemanların eklenebildiği özel kutular tasarlanabilir (Görsel 4.45).

```
<div class="card" style="width:300px">
  <img class="card-img-top"  src="profilYeni.jpg">
  <div class="card-body">
    <h4 class="card-title">Kart Başlığı</h4>
    <p class="card-text">Kart için gerekli içerik buraya yazılmaktadır. </p>
    <a href="#" class="btn btn-primary">Detaylar</a>
   </div>
</div>
```

Görsel 4.45: Kart sınıfı

Sıra Sizde

Altı adet kart tasarlayarak Türkiye'nin altı farklı iline ait turistlik yerlerin tanıtımını yapınız (Tüm kartlar; ekranın **en küçük** boyutunda alt alta, **küçük** ve **orta** boyutlarında dört kart yan yana, **geniş** ve **en geniş** boyutlarında altı kart yan yana olacaktır.).

4.4.3.12. Açılır Kapanır Kutu Yapısı

Gizlenmek veya gösterilmek istenilen elemana "collapse" sınıfı atanır. Collapse sınıfı ile içeriğin belirlenen kısmı kutu içine gizlenebilir veya kutudan dışarı çıkarılabilir. <a>, <button> elemanlarının "data-toggle" özelliği "collapse" yapılır. <a> etiketinin "href" özelliği gizlenmek veya gösterilmek istenilen elemanın "id" değeriyle aynı olmalıdır.

```
<a href="#demo" data-toggle="collapse">Detaylı bilgi için tıklayınız.</a>
  <div id="demo" class="collapse">
    Bilgisayar kasası içinde; anakart, işlemci, ram, harddisk, fan  ve anakart üzerine monte edilmiş ekran kartı, ethernet kartı, ses kartı  bulunmaktadır. İstenilirse haricî ekran kartı, ethernet kartı ve ses kartı takılabilmektedir. </div>
```

```
<button class="btn btn-info" data-toggle="collapse" data-target="#demo"> Detaylı bilgi için tıklayınız. </button>
```

<button> elemanı " data-target" özelliği gizlenmek veya gösterilmek istenilen elemanın "id" değeriyle aynı olmalıdır.

4.4.3.13. Menü Çubukları

Gezinti Menüsü: nav sınıfı özellikleri ile gezinme çubukları veya açılır listeler tasarlanabilir.

Aşağıda basit bir gezinme menüsü ve gezinme menüsü içinde açılır liste kodları görülmektedir (Görsel 4.46).

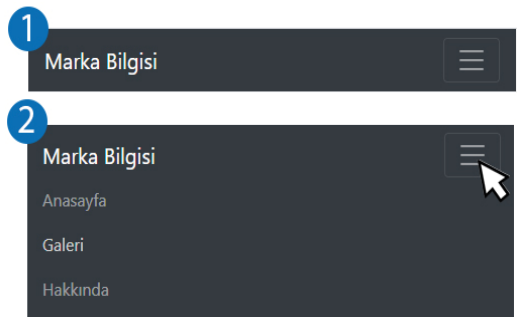
```
<ul class="nav nav-tabs bg-light">
 <li class="nav-item"> <a class="nav-link" href="#">AnaSayfa</a> </li>
 <li class="nav-item"> <a class="nav-link" href="#">Makaleler</a> </li>
  <li class="nav-item dropdown">
   <a class="nav-link dropdown-toggle" data-toggle="dropdown" href="#">Galeri</a>
    <div class="dropdown-menu">
      <a class="dropdown-item" href="#">Profil Resimleri</a>
      <a class="dropdown-item"  href="#">Mobil Resimler</a>
      <a class="dropdown-item"  href="#">Tüm Resimler</a>
    </div>
 </li>
 <li class="nav-item"> <a class="nav-link" href="#">Hakkında</a> </li>
 <li class="nav-item"> <a class="nav-link disabled" href="#">İletişim</a> </li>
</ul>
```

Görsel 4.46: Duyarlı menü

Sıra Sizde

Belirlediğiniz bir konuda (araçlar, kitaplar vs.) içinde en az bir tane açılır liste olan gezinti menüsü tasarlayınız.

Menü Çubuğu: Navbar sınıfı kullanılarak ekranın üst veya alt kısmına konumlandırılmış, duyarlı menü çubukları tasarlanabilir (Görsel 4.47).



Görsel 4.47: Duyarlı menü çubuğu

Görsel 4.47'de görülen duyarlı çubuğun kodları aşağıda verilmiştir.

```
<nav class="navbar navbar-expand-md bg-dark navbar-dark fixed-top">
  <a class="navbar-brand" href="#">Marka Bilgisi</a>   <!—Logo veya Marka Bilgisi -->
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#me-
nu">
    <span class="navbar-toggler-icon"></span> <!-- Duyarlı menü butonu -->
  </button>
  <div class="collapse navbar-collapse" id="menu"> <!-- Menü Linkleri -->
    <ul class="navbar-nav">
      <li class="nav-item"> <a class="nav-link" href="#">Anasayfa</a> </li>
      <li class="nav-item"> <a class="nav-link" href="#">Galeri</a> </li>
      <li class="nav-item"> <a class="nav-link" href="#">Hakkında</a> </li>
    </ul>
  </div>
</nav>
```

Not

İstenilirse menü linkleri bölümüne açılır liste de eklenebilir.

Sıra Sizde

Belirlediğiniz bir konuda (araçlar, kitaplar vs.) bir menü çubuğu tasarlayınız.

4.4.3.14. Form Sınıfı

Bootstrap Framework ile HTML5'te bulunan tüm form elemanları biçimlendirilebilir. Form elemanlarının biçimlendirme işlemleri "form-group, form-control, form-check" sınıfları ile yapılmaktadır.

<input> Elemanı: Type özelliği; text, password, datetime, datetime-local, date, month, time, week, number, email, url, search, tel, ve color için biçimlendirme yapılabilir.

```
<form action="/index.html">
  <div class="form-group">
    <label >Kullanıcı Adı:</label> <input type="text" class="form-control" >
  </div>
  <div class="form-group">
    <label >Şifre:</label> <input type="password" class="form-control">
  </div>
</form>
```

<textarea> Elemanı

```
<div class="form-group">
  <label">Açıklama:</label> <textarea class="form-control" rows="3"> </textarea>
</div>
```

<select> Elemanı

```
<div class="form-group">
  <label>Öğrenim Durumu:</label>
  <select class="form-control">
    <option>İlkokul</option>  <option>Ortaokul</option>  <option>Lise</option>
  </select>
</div>
```

<checkbox> Elemanı

```
<div class="form-check-inline">
   <label class="form-check-label">
   <input type="checkbox" class="form-check-input">Web tasarım yapabiliyorum. </label>
</div>
<div class="form-check-inline">
    <label class="form-check-label">
    <input type="checkbox" class="form-check-input">Mobil uygulama yapabiliyorum. </
label>
</div>
```

<radio> Elemanı

```
<div class="form-check">
    <label class="form-check-label">
    <input type="radio" class="form-check-input"> name="secim">Mezun </label>
</div>
<div class="form-check">
     <label class="form-check-label">
     <input type="radio" class="form-check-input"> name="secim">Mezun değil </label>
</div>
```

Sıra Sizde

Kullanıcıların; isim, soy isim, mail, adres, telefon, öğrenim durumu bilgilerini girebilecekleri bir iletişim formu tasarlayınız.

4.4.3.15. Slider (Resim Galerisi) Sınıfı

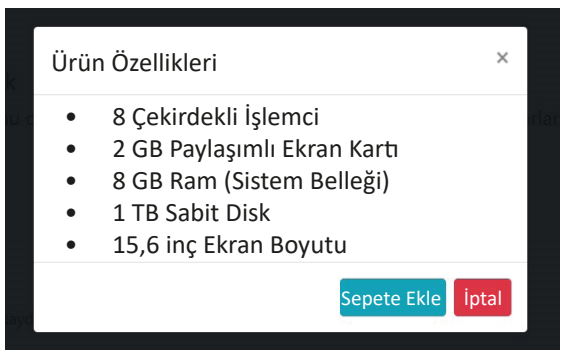
Bootstrap Framework ile gelişmiş özellikleri olan ve çok kullanışlı resim galerisi yapılmaktadır. Galerinin düzgün çalışması için kullanılacak resimlerin boyutlarının birbiriyle uyumlu olması gerekir. Resim galerisinin etkileşimli üç farklı resim değiştirme noktası bulunmaktadır. Ayrıca her resim için başlık ve açıklama metni için bir bölüm de bulunmaktadır (Görsel 4.48).

Görsel 4.48: Resim galerisi

```
<div id="galeri" class="carousel slide" data-ride="carousel">
<!-- <ul> elemanı resim geçişlerini yönetmek için çizgi şeklinde işaretçilerin eklenildiği
 bölümdür. <li> elemanları ile çizgiler eklenir. Her resim için bir işaretçi eklenmelidir.-->
  <ul class="carousel-indicators">
   <li data-target="#galeri" data-slide-to="0" class="active"></li>
   <li data-target="#galeri" data-slide-to="1"></li>
   <!-- Galerideki resim sayısına göre <li> etiketi eklenmelidir.-->
  </ul>
 <div class="carousel-inner"><!-- Galeri resimlerinin ve metinlerinin eklendiği bölümdür. -->
 <div class="carousel-item active"><!--İlk gösterilecek resim  için active sınıfı eklenir.-->
    <img src="SahilYolu.jpg"  width="1100" height="500">
    <div class="carousel-caption">
     <h3>Türkiye</h3> <p>Karadeniz Bölgesi, Ordu Sahil Yolu</p>
    </div>
 </div>
 <div class="carousel-item">
    <img src="SaatKulesi.jpg"  width="1100" height="500">
    <div class="carousel-caption">
     <h3>Türkiye</h3> <p>Ege Bölgesi, İzmir Saat Kulesi </p>
   </div>
 </div>
    <-- Diğer eklenecek resimlerin kodları buraya yazılabilir.-->
</div>
  <a class="carousel-control-prev" href="#galeri" data-slide="prev">
   <span class="carousel-control-prev-icon"></span> <!-- Önceki resmi gösterme butonu -->
  </a>
  <a class="carousel-control-next" href="#galeri" data-slide="next">
   <span class="carousel-control-next-icon"></span> <!-- Sonraki resmi gösterme butonu -->
  </a>
</div>
```

4.4.3.16. Modal Sınıfı

Modal sınıfı ile geçerli sayfa üzerinde bir iletişim kutusu açılır. Bu iletişim kutusu ile bilgilendirici içerikler, kullanıcının bilgi girişi yapabileceği kutular sunulabilir (Görsel 4.49). Örneğin bir e-ticaret sitesinin ürün ile ilgili detayları modal sınıfı kullanılarak verilebilir veya kullanıcının müşteri memnuniyetini ölçecek bir anket sorusu da modal sınıfı kullanılarak sorulabilir.



Görsel 4.49: Modal sınıfı

```
 <button type="button" class="btn btn-primary" data-toggle="modal" data-target="#Detay-
lar">
   Ürün Detayları
</button><!-- <button> elemanının data-target değeri  Modal'ın id değerine uygun olmalıdır.
-->
   <div class="modal" id="Detaylar">
    <div class="modal-dialog">
     <div class="modal-content">
       <!-- Modal Başlığı -->
       <div class="modal-header">
        <h4 class="modal-title">Ürün Özellikleri</h4>
        <button type="button" class="close" data-dismiss="modal">&times;</button>
       </div>
       <!-- Modal İçeriği -->
       <div class="modal-body">
        <ul>
         <li>8 Çekirdekli İşlemci</li>
         <li>2 GB Paylaşımlı Ekran Kartı</li>
         <li>8 GB Ram(Sistem Belleği)</li>
         <li>1 TB Sabit Disk</li>
         <li>15,6 inç Ekran Boyutu</li>
        </ul>
       </div>
       <!-- Modal Alt Bölümü -->
       <div class="modal-footer">
        <button  type="button" class="btn btn-info"> Sepete Ekle</button>
        <button  type="button" class="btn btn-danger" data-dismiss="modal">İptal</button>
       </div>
     </div>
    </div>
   </div>
```

Not

data-dismiss özelliği modalı kapatmak için kullanılmıştır.

4.4.3.17. İkon (İcon) Sınıfı

Web sitesine ikonlar eklemek için öncelikle HTML sayfasının <head> bölümüne aşağıdaki kodu eklemek gerekmektedir.

```
<link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.7.0/css/all.css" >
```

HTML sayfasının <body> bölümünde "fa" sınıfı kullanarak ikon (icon) eklenebilir. Onlarca kategoride yüzlerce ikon bulunmaktadır. İkonların kodlarına ve görsellerine internet üzerinden ulaşılabilir. Aşağıda ikon sınıfı kullanımı için örnek kodlar verilmiştir. İkonlar Görsel 4.50'deki gibi görülmektedir.

```
<i class="fa fa-home"></i>
<i class="fa fa-phone"></i>
<i class="fa fa-star"></i>
<i class="fa fa-cloud"></i>
<i class="fa fa-heart"></i>
<i class="fa fa-car"></i>
<i class="fa fa-file"></i>
<i class="fa fa-bars"></i>
```

Görsel 4.50: İkon sınıfı

Sıra Sizde

Kendi ilgi alanınıza göre bir konu belirleyiniz. Belirlediğiniz konuya uygun en az yedi sayfadan oluşan tüm çözünürlüklere uyumlu bir web sitesi tasarlayınız ve tasarladığınız web sitesinin tanıtım sunumunu yapınız.

ÖLÇME VE DEĞERLENDİRME

A. Aşağıdaki cümlelerde boş bırakılan yerlere doğru sözcükleri yazınız.

1. Elemanın hem satır içi hem de blok seviyesinde davranmasını display:..... kodu ile ayarlanır.

2. Bir elemanın sayfadaki diğer elemanların konumlarına göre başlangıç noktası değiştirmesi position:..... kodu ile ayarlanır.

3. Üst üste gelen elemanlardan hangisinin en üstte görüneceği ..... kodu ile ayarlanır.

4. Renk kodlarında # işaretinden sonra yazılan ilk iki basamak ..... rengi temsil eder.

5. RGB fonksiyonunda her parametre en çok ..... değeri alır.

B. Aşağıdaki soruları cevaplayınız.

6. Kutu modelini oluşturan dört özellik nedir? Yazınız.

7. CSS ekleme yöntemleri nelerdir?

8. Çocuk seçici ile torun seçici arasındaki fark nedir?

9. CSS ölçü birimlerinden göreceli olanlar hangileridir?

10. Bootstrap kenarlık sınıfı kullanarak bir resme üç farklı kenarlık ekleyen uygulamayı yapınız.

5.
ÖĞRENME BİRİMİ

ETKİLEŞİM (JAVASCRİPT)

NELER ÖĞRENECEKSİNİZ?

Bu öğrenme birimi ile;

- Değişken kavramını ve veri tiplerini kavramayı,
- Veri türlerini sıralamayı,
- Operatör kullanım yerlerini açıklamayı,
- Kontrol yapılarını kullanarak etkileşimli sayfa geliştirmeyi,
- Döngülerin kullanımı ve özelliklerini açıklamayı,
- Dizilerin kullanımını kavramayı,
- Fonksiyon türlerini kullanarak etkileşimli sayfa geliştirmeyi,
- Popüler Javascript kütüphanelerini listelemeyi,
- jQuery kütüphanelerinin bağlantısını yapabilmeyi,
- jQuery ile animasyonlu sayfa tasarımları yapmayı öğreneceksiniz.

ANAHTAR KELİMELER

Değişkenler, diziler, döngüler, fonksiyonlar, jQuery, kontrol yapıları, kütüphaneler, olay.

Hazırlık Çalışmaları

1. Kullandığınız web sitelerinin kullanıcı ile etkileşim sağlayan bölümleri nelerdir? Açıklayınız.

2. Popüler Javascript kütüphanelerinin kod yazımında ne gibi avantajlar sağladığını düşünüyorsunuz? Düşüncelerinizi arkadaşlarınızla paylaşınız.

5.1. Javascript Kod Yapısı ve Değişkenler

HTML ve CSS ile kullanıcıyla neredeyse hiç etkileşimi olmayan (statik) web sayfaları düzenlenmektedir. Javascript kullanılarak çok daha etkileşimli, kullanıcı dostu, farklı durumlarda farklı görünüme sahip dinamik web sayfaları tasarlanabilir. Örneğin; butona tıklandığında içine bilgi girişi yapılmamış bir <input> elemanının, arka plan rengi kırmızı olarak değiştirilip ekrana "Bilgi girişi yapılmadı." şeklinde uyarı mesajı verdirilebilir veya anlık olarak gelen mesajların görüntülendiği bir mesajlaşma paneli oluşturulabilir.

Görsel 5.1: Javascript kodlar

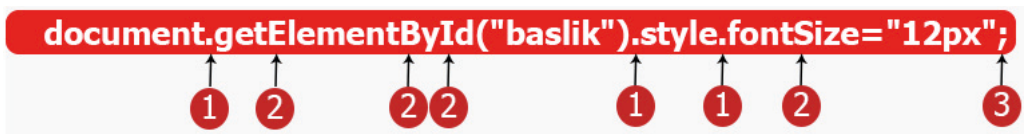
Javascript kendi başına bir programlama dili değildir fakat bir programlama dilini oluşturan değişkenler, karar yapıları, döngüler, fonksiyonlar gibi temel bileşenleri yapısında barındırır (Görsel 5.1). Günümüzde Javascript dili ile etkileşimli web sayfalarının yanı sıra çeşitli frameworkler kullanılarak mobil uygulamalar da geliştirilebilmektedir.

Javascript kodları yazılırken genel olarak üç şeye dikkat edilmelidir (Görsel 5.2):

1. Sınıf, fonksiyon, özellik gibi yapılar **"." (nokta)** kullanılarak yazılmalıdır.

2. Kod içinde geçen ifadeler birden fazla kelimeden oluşuyorsa birinci kelimeden sonraki tüm kelimelerin baş harfleri büyük yazılmalıdır. **Örneğin;** Javascript ile stil değiştirmek için CSS kodları kullanılır. Fakat birden fazla kelimeden oluşan CSS kodu kullanılacaksa CSS kodundaki **"–" (kısa çizgi)** işareti kullanılmaz ve birinci kelimeden sonraki tüm kelimelerin baş harfleri büyük yazılır.

3. Her Javascript kodu **";" (noktalı virgül)** ile bitmelidir.



Görsel 5.2: Javascript kod yapısı

Javascript kodlarının web sayfasında çalışması için sayfa içi veya sayfa dışı (haricî) Javascript ekleme yöntemleri kullanılır.

Sayfa İçi Javascript Ekleme: HTML sayfasında herhangi bir yere <script> etiketi içine Javascript kodları yazılır.

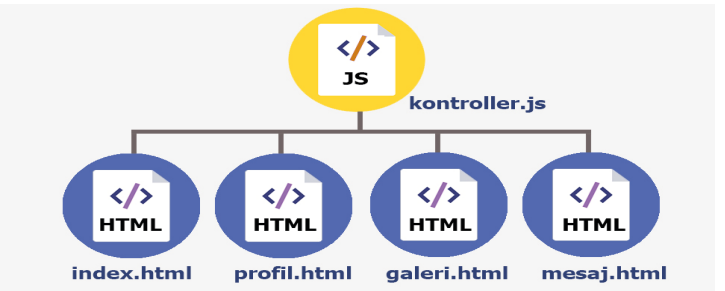
```
<script>
alert("Merhaba Javascript");
</script>
```

Not

"alert" komutu ekrana mesaj verdirmek için kullanılır.

Sayfa Dışı (Haricî) Javascript Ekleme: Haricî bir "js" uzantılı dosya içine Javascript kodları yazılabilir. Sayfa dışı Javascript kodlarını çalıştırmak için HTML sayfasında <head> etiketi içine aşağıdaki örnek kodda görüldüğü gibi bağlantı kodu yazılmalıdır (Görsel 5.3).

```
<script type="text/javascript" src="JscriptDosyasi.js"></script>
```



Görsel 5.3: Sayfa dışı Javascript ekleme

Sıra Sizde

"alarm.js" adında bir Javascript dosyası oluşturunuz ve gerekli bağlantıyı yaptıktan sonra ekrana "Haricî Javascript kodu çalışıyor." mesajını verdiriniz.

Javascript Genel Çalışma Yapısı: Web sayfasındaki HTML elemanı için tanımlanmış bir olay (fare ile tıklanma vb.) gerçekleştiğinde bir Javascript fonksiyonu çağrılır. Fonksiyon içeriğinde; değişkenler, kontrol yapıları, döngüler, zamanlayıcılar, seçiciler kullanılabilir. Seçiciler ile elemanların stil özellikleri, HTML özellikleri, metinsel içerikleri, değerleri değiştirilebilir (Görsel 5.4).

Görsel 5.4: Javascript genel çalışma yapısı

5.1.1. Seçiciler

CSS'de olduğu gibi Javascript ile istenilen elemanların özelliklerinde seçiciler kullanılarak birçok değişiklik (manipülasyon) yapılabilir. Seçiciler kullanılarak elemanlar üzerinde değişiklik yapılabilecek bazı alanlar Tablo 5.1'de verilmiştir.

Tablo 5.1: Seçicilerin Bazı Değişiklik Alanları

JAVASCRIPT KODU	AÇIKLAMA
style	Elemanların tüm stil özellikleri değiştirilebilir.
innerHTML	Elemanların içeriğindeki HTML kodları ve metinsel ifadeler hem döndürebilir hem de değiştirebilir.
innerText	Elemanların içeriğindeki metinsel ifadeler hem döndürebilir hem de değiştirebilir.
value	Bilgi girişi yapılabilen form elemanlarının (input, select vb.) değerleri (value) hem döndürebilir hem de değiştirebilir.

Seçiciler kullanılmadan elemanların herhangi bir özelliğinde değişiklik yapılamaz. Javascript dilinde dört adet seçici bulunur.

ID (Kimlik) Seçiciler: Web sayfasında bulunan elemanlar içinden "id" değeri eşleşen elemanı seçer. Aynı id değerine sahip birden fazla eleman varsa sadece ilk eşleşen eleman üzerinde değişiklik yapılır.

```
document.getElementById("baslik").style.fontSize="52px";
document.getElementById("icerik").innerHTML="<b>Bu yazı kalın bir yazıdır.</b>";
document.getElementById("paragraf").innerText="Paragrafa yeni cümleler eklenebilir.";
document.getElementById("adSoyad").value="Mustafa ÖZER";
```

Not

Yukarıdaki örnek koda göre seçici tarafından ilk tespit edilen elemanların özellikleri değiştirilmiştir.

ClassName (Sınıf) Seçiciler: Web sayfasında bulunan elemanlar içinden, "class" değeri eşleşen elemanların hepsini seçer. Seçilen elemanların tümü sırasıyla bir diziye aktarılır. Dizi elemanları **[ ] (köşeli ayraç)** ile belirtilir. İlk elemanın dizideki numarası "0"dır.

HTML

```
<body style="background-color: orange;">
   <span class="harfler">J</span>
   <span class="harfler">S</span>
   <span class="harfler">cript</span>
</body>
```

JS

```
document.getElementsByClassName("harfler")[0].style.fontSize="22px";
document.getElementsByClassName("harfler")[1].style.fontSize="22px";
document.getElementsByClassName("harfler")[0].style.color="white";
document.getElementsByClassName("harfler")[1].style.color="white";
```

Not

ClassName (Sınıf) Seçiciler konusunda verilen örnek koda göre üç adet HTML elemanı, seçici tarafından tespit edilip bir diziye aktarılmıştır. Tespit edilen elemanlardan ilk ikisinin yazı tipi 22 piksel ve yazı rengi beyaz olarak değiştirilmiştir. Kodların ekran görüntüsü Görsel 5.5'teki gibidir.

Görsel 5.5: Sınıf seçiciler

Name (İsim) Seçiciler: Web sayfasında bulunan elemanlar içinden "name" değeri eşleşen elemanların hepsini seçer. Seçilen elemanların tümü sırasıyla bir diziye aktarılır.

```
document.getElementsByName("kullaniciAdi")[0].style.backgroundColor="red";
```

Not

Yukarıdaki örnek koda göre seçici tarafından seçilip bir diziye aktarılan elemanlardan birincisinin arka plan rengi kırmızı olarak değiştirilmiştir.

TagName (Etiket) Seçiciler: Web sayfasında bulunan elemanlar içinden "etiket" adı eşleşen elemanların hepsini seçer. Seçilen elemanların tümü sırasıyla bir diziye aktarılır. Eşleşme sağlanan ilk elemanın dizi numarası "0"dır.

```
document.getElementsByTagName("span")[2].style.borderBottomStyle="solid";
document.getElementsByTagName("span")[2].style.borderBottomSize="2px";
```

Not

Yukarıdaki örnek koda göre seçici tarafından seçilip bir diziye aktarılan elemanlardan üçüncüsünün alt kenarlık stili değeri solid, alt kenarlık kalınlığı 2 piksel olarak değiştirilmiştir.

5.1.2. Değişkenler ve Veri Tipleri

Değişkenler, tüm programlama dillerinde en temel kavramlardan biridir. Değişkenler; kodların çalışması sırasında metinsel, sayısal, mantıksal vb. tiplerdeki verileri hafızada saklayan, ihtiyaca göre tekrar tekrar kullanma olanağı sağlayan veri tutuculardır. Kodlar çalıştığı (web sayfası görüntülenirken) sürece değişkenler RAM bellekte tutulur, kodların çalışması durdurulduğunda RAM bellekten silinir.

Javascript'te değişkenler, "var" anahtar kelimesi kullanılarak tanımlanır. Birçok programlama dilinde değişkenlerin veri tipleri belirtilirken Javascript'te değişkenlerin tiplerini özel olarak belirtmek gerekmez.

```
var sayi=52;      //sayısal(number) veri tipi
var oran=1.618;   //sayısal(float) veri tipi
var sehir="Ordu"; //metinsel(string) veri tipi
var tercih=true;  //mantıksal(boolean) veri tipi
var personel={    //nesne(object) veri tipi
    ad:"Sinan",
    boy:1.87,
    kilo:92,
    meslek:"Şef(Aşçı)"
  }
```

Yukarıdaki örnek kodlarda görüldüğü gibi değişkenlerin aldığı değere göre tipi otomatik tanımlanmıştır. Bu yönüyle Javascript kod yazımı diğer dillere göre avantajlıdır.

Sayısal Veri Tipi: Tam sayıları veya ondalık sayıları temsil eden veri tipidir. Sayılar **çift tırnak (“)** veya **tek tırnak (‘)** kullanılmadan yazılmalıdır.

```
var a = 5.5;        //sayısal(float) veri tipi
var b = 16;         //sayısal(number) veri tipi
var toplam = a + b; //sayısal(float) veri tipi
```

Metinsel Veri Tipi: Metinsel ifadeleri temsil eden veri tipidir. Metinler **çift tırnak (“)** veya **tek tırnak (‘)** kullanılarak yazılmalıdır.

```
var secenek="A";
var yil="2021";
var soru='Javascript ile bir elemanın stil özellikleri değiştirilebilir.';
```

Not

Tırnak içinde girilen değer bir sayı da olsa metinsel ifade olarak değerlendirilir.

1. Uygulama

Metinsel ifadeler ile toplama yaptırma işlemini yönergeler doğrultusunda gerçekleştiriniz.

1. Adım: Değerleri sırasıyla “1, 98, 8” olan üç farklı değişken tanımlayınız.
2. Adım: Toplam isimli bir değişkene değer olarak tanımlamış olduğunuz değişkenlerin toplamını atayınız.
3. Adım: Toplam değerini mesaj olarak verdiriniz.

```
var a = "1";
var b = "98";
var c = "8";
var Toplam = a + b + c;
alert(Toplam);
```

Not

Ekranda 1988 yazılı bir mesaj kutusu görüntülenecektir. Buna metinsel toplama işlemi denir.

Mantıksal Veri Tipi: Mantıksal ifadeleri temsil eden veri tipidir. True ve false olmak üzere iki farklı değer alır.

```
var soru='Javascript ile bir elemanın stil özellikleri değiştirilebilir.';
var cevap=true;
```

Nesne (Object) Veri Tipi: Birçok veri ikilisinin (değişken ve değer) tek çatı altında toplanmasına olanak sağlayan veri tipidir. Değişkenler aynı veya farklı veri tiplerinde olabilir. Her veri ikilisi arasına **","** **(virgül)** konulmalıdır. Nesnelere **veri koleksiyonu** da denilebilir.

```
var ev={
    kat:8,
    oda:3,
    kimden:"emlakçı",
    kira:850,
    aidat:90,
    bina_yasi:6,
}
```

Not

Yukarıdaki örnek kod ile nesne veri tipine sahip "ev" isimli değişkenin altı farklı özelliği, koleksiyon hâlinde tutulmaktadır. "ev" değişkenin herhangi bir özelliğine ulaşmak için değişken adından sonra **nokta (.)** koyup özellik adı yazılmalıdır (Örnek: "ev. kimden").

Değişken Tanımlama Kuralları: Değişken tanımlarken aşağıdaki kurallara dikkat edilmelidir.

- Tüm tanımlanan değişkenlerin isimleri benzersiz olmalıdır.
- Değişken isimleri harf, alt çizgi (_) veya dolar işareti ($) ile başlamalıdır.
- Değişken isimleri sayı ile başlayamaz.
- Değişken isimlerinde alt çizgi ve dolar işareti hariç özel karakterler kullanılamaz.
- Değişken isimleri büyük ve küçük harflere duyarlıdır (**Sayi** ile **sayi** değişkenleri farklıdır.).
- Javascript'e ait kod ifadeleri değişken ismi olamaz (var, if, else vb.).

Sıra Sizde

Değişken tanımlama kurallarına uyarak tüm değişken türlerinden ikişer adet değişken tanımlayıp değişkenlere değer atayınız.

5.1.3. Operatörlerin Kullanımı ve Kullanım Yerleri

Operatörler, kod yazımı sırasında; aritmetiksel işlemler, değer atama işlemleri, karşılaştırma işlemleri, mantıksal işlemler için kullanılan özel karakterlerdir.

Aritmetiksel operatörler, matematiksel işlemlerde kullanılan özel karakterlerdir (Tablo 5.2).

Tablo 5.2: Aritmetiksel Operatörler

Operatör Adı	Sembolü	Örnek	
Toplama	+	8+2	a+b
Çıkarma	-	8-2	a-b
Çarpma	*	8*2	a*b
Bölme	/	8/2	a/b
Mod alma	%	8%2	a%b
Arttırma	++		a++
Azaltma	--		b--

Atama operatörleri, değer aktarma işlemlerinde kullanılan özel karakterlerdir (Tablo 5.3).

Tablo 5.3: Atama Operatörleri

Operatör Adı	Sembolü	Örnek	Açıklama
Eşittir	=	A =5+2	Sayıların toplamını A değişkenine aktarır.
Artı Eşittir	+=	A+=5	A değişkeninin mevcut değerine 5 daha ekler.
Eksi Eşittir	-=	A-=5	A değişkeninin mevcut değerinden 5 çıkarır.
Çarpı Eşittir	*=	A*=5	A değişkeninin mevcut değerini 5 ile çarpar.
Bölü Eşittir	/=	A/=5	A değişkeninin mevcut değerini 5'e böler.
Üs Alma	**=	A**=5	A değişkeninin mevcut değerinin 5. kuvvetini alır.

Karşılaştırma operatörleri, karar ifadelerinde iki değeri karşılaştırmak için kullanılan özel karakterlerdir (Tablo 5.4).

Tablo 5.4: Karşılaştırma Operatörleri

Operatör Adı	Sembolü	Örnek	Açıklama
Küçükse	<	A<B	A değişkeni B değişkeninden küçükse
Büyükse	>	A>B	A değişkeni B değişkeninden büyükse
Eşitse	==	A==B	A değişkeni B değişkenine eşitse
Küçük veya Eşitse	<=	A<=B	A değişkeni B değişkeninden küçük veya B değişkeninin değerine eşitse
Büyük veya Eşitse	>=	A>=B	A değişkeni B değişkeninden büyük veya B değişkeninin değerine eşitse
Eşit Değilse	!=	A!=B	A değişkeni B değişkeninden farklıysa

Mantıksal operatörler, birden fazla karar ifadesinin birlikte kullanılması gereken durumlarda ve karar ifadesinin sağlanma durumunun tersine çevrilmesi gerektiği durumlarda kullanılan özel karakterlerdir (Tablo 5.5).

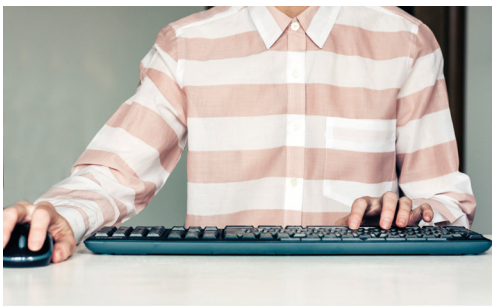
Tablo 5.5: Mantıksal Operatörler

Operatör Adı	Sembolü	Örnek	Açıklama
Ve	&&	(A<B) && (A<100)	A değişkeni B değişkeninden küçükse ve A değişkeni 100 sayısından küçükse şart sağlanır.
Veya	||	(A=="Ali") || (A=="Hakan")	A değişkeni "Ali" değerine eşitse veya A değişkeni "Hakan" değerine eşitse şart sağlanır.
Değil	!	!(Cevap=="E")	Cevap değişkeninin değeri "E" değerine eşit değilse şart sağlanır.

5.2. Olaylar ve Fonksiyonlar

Olaylar (Events): Her elemanın web sayfası üzerinde bir görevi bulunur. Bazı elemanlar sadece kod çalışması sonucunda oluşan bilgiyi ve görseli yansıtmak için kullanılır. Bazı elemanlar ise belli durumlarda (üzerine tıklandığında, bir tuşa basıldığında, eleman tarayıcıya yüklendiğinde vs.) kod parçacıklarını çalıştırmak için kullanılır (Görsel 5.6). Elemanlar, kullanıcı ile etkileşimi olaylar sayesinde sağlar. Günlük hayatta; sosyal medya uygulamalarında, web sitelerinde, oyunlarda ve daha birçok alanda elemanlar için tanımlanmış olay fonksiyonları, çalışarak kullanıcıyla etkileşimi sağlar. Olay fonksiyonları tanımlanarak elemanların hangi durumda, nasıl kodlar çalıştırabileceği belirlenir. Örneğin; butona tıklandığında şifre kontrolünün yapılması, klavyeden sağ ok tuşuna basıldığında bir sonraki resmin gösterilmesi, resim elemanının üzerine çift tıklama yapıldığında resme ait bilgilerin gösterilmesi vb.

Olaylar gerçekleşince çalışması istenen kodlar doğrudan HTML etiketi içine yazılabilir veya çalışması istenen kodları temsilen bir fonksiyon çağrılabilir. Bu yönüyle olaylar ve fonksiyonlar birbirlerini tamamlayan yapılardır.



Görsel 5.6: Fare (mouse), klavye olayları

```
<button type="button" onclick="darkMode()">Gece Modu</button>
<button type="button" onclick="lightMode()">Gündüz Modu</button>
```

Olay gerçekleştiğinde çalışması istenilen kodlar birkaç satırdan fazla ise ilgili kodlar aşağıdaki örnek kodlarda görüldüğü gibi fonksiyon yapısı içine yazılmalıdır.

```
function darkMode(){
   document.getElementsByTagName("body")[0].style.color="white";
   document.getElementsByTagName("body")[0].style.backgroundColor="black";
}
function lightMode(){
   document.getElementsByTagName("body")[0].style.color="black";
   document.getElementsByTagName("body")[0].style.backgroundColor="white";
}
```

Tanımlanabilecek birçok Javacript olayı bulunmaktadır. Bunlardan bazıları Tablo 5.6'da sıralanmaktadır.

Tablo 5.6: Javascript Olayları

OLAY	AÇIKLAMA
onLoad	Belirtilen elemanın tarayıcıya yüklenmesiyle gerçekleşen olaydır.
onClick	Kullanıcının bir elemana tıklamasıyla gerçekleşen olaydır.
onDblClick	Kullanıcının bir elemana çift tıklamasıyla gerçekleşen olaydır.
onFocus	Kullanıcının bir elemanı seçtiğinde (odaklandığında) gerçekleşen olaydır.
onBlur	Kullanıcının seçtiği elemandan ayrıldığında gerçekleşen olaydır.
onMouseOver	Kullanıcının bir elemanın üzerinde fare ile gezdiğinde gerçekleşen olaydır.
onMouseOut	Kullanıcının fare ile üzerinde gezdiği elemandan ayrıldığında gerçekleşen olaydır.
onMouseDown	Kullanıcının bir elemanın üzerine fare ile bastığı zaman gerçekleşen olaydır.
onMouseUp	Kullanıcının fare ile bir elemanın üzerine basmayı bıraktığında gerçekleşen olaydır.
onKeyDown	Metin girişi yapılabilen elemanlara bilgi girişi yaparken gerçekleşen olaydır.

Fonksiyonlar: Kod karmaşıklıklarını, kod tekrarlarını ortadan kaldıran belirli işlemleri yapmak için tanımlanan kod bloklarıdır. Bir fonksiyon tanımlamak için "function" anahtar sözcüğü, fonksiyon adı, parametre girişi için **() (parantez)** ardından yapılacak işlemlerin kodları için **{ } (süslü parantez)** gerekmektedir.

```
function fonksiyonAdi(){
  //kodlar
}
```

Not

Yukarıdaki örnek kodda bir fonksiyonun en temel şekilde yapısı gösterilmiştir.

```
fonksiyonAdi();
```

Tanımlanan fonksiyonlar, yukarıdaki örnek kodda görüldüğü gibi adları ile çağrılır ve içeriğindeki işlemleri gerçekleştirir. Bir fonksiyon istenildiği kadar çağrılabilir. Fonksiyonlar genellikle Javascript olayları ile birlikte kullanılır.

5.2.1. Parametresiz (Basit) Fonksiyonlar

Kullanımı en kolay fonksiyon çeşididir. Bu tür fonksiyonlarda yapacağı işlemleri etkileyecek herhangi bir parametre girişi bulunmaz.

HTML

```
<input type="button"  onclick="boyutlandir()">
<img src="gunBatimi.jpg" id="resim">
```

Javascript

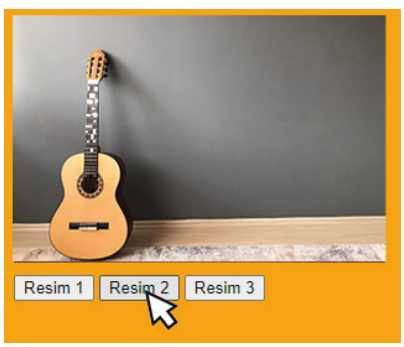
```
function boyutlandir(){
   document.getElementById("resim").style.width="150px";
   document.getElementById("resim").style.height="150px";
}
```

Not

Yukarıdaki HTML kodunda, butona tıklandığında resim elemanın boyutu, fonksiyon çağrılarak değişmektedir. Yapılan işlem, sadece bir elemanın özelliğini değiştirmek olduğu için fonksiyonda parametreye veya geriye değer döndürmeye ihtiyaç duyulmamıştır.

2. Uygulama

Parametresiz fonksiyonlar kullanarak Görsel 5.7'deki gibi resim galerisi oluşturmak için yönergeler doğrultusunda işlemleri gerçekleştiriniz.



Görsel 5.7: Resim galerisi

1. Adım: Uygulama klasörünüzde birer adet HTML, Javascript ve CSS dosyası oluşturarak HTML sayfasına gerekli bağlantı kodlarını yazınız.

2. Adım: Galeri için kullanılacak resimler için Uygulama klasörünüzde "Resimler" isimli bir klasör oluşturunuz ve içine üç adet resim koyunuz.

3. Adım: HTML dosyasının <body> etiketi içine aşağıdaki kodu yazınız.

HTML

```
<div id="galeri">
    <img src="Resimler/gunBatimi.jpg" id="resim1">
    <img src="Resimler/kahveBilgisayar.jpg" id="resim2" >
    <img src="Resimler/kumsal.jpg" id="resim3">
</div>
<div>
   <input type="button" onmousemove="resmiGoster1()" value="Resim 1">
   <input type="button" onmousemove="resmiGoster2()" value="Resim 2">
   <input type="button" onmousemove="resmiGoster3()" value="Resim 3">
</div>
```

4. Adım: CSS dosyasına aşağıdaki kodu yazınız.

CSS

```
    body{background-color: orange;}
  #galeri{ position: relative;  width:300px;  height: 200px;   margin-bottom:10px; }
      img{ width:300px; height: 200px; position: absolute; left:0px; right: 0px;  }
```

5. Adım: Javascript dosyasına aşağıdaki kodu yazınız.

Javascript

```
function resmiGoster1(){
   document.getElementById("resim1").style.zIndex=2;
   document.getElementById("resim2").style.zIndex=1;
   document.getElementById("resim3").style.zIndex=1;
}
function resmiGoster2(){
   document.getElementById("resim1").style.zIndex=1;
   document.getElementById("resim2").style.zIndex=2;
   document.getElementById("resim3").style.zIndex=1;
}
function resmiGoster3(){
   document.getElementById("resim1").style.zIndex=1;
   document.getElementById("resim2").style.zIndex=1;
   document.getElementById("resim3").style.zIndex=2;
}
```

5.2.2. Parametreli Fonksiyonlar

Parametreli fonksiyonlarda, yapılacak işlemleri etkileyecek parametre girişi veya girişleri bulunur.

```
function yeniBoyut(a){
   document.getElementById(a).style.width="150px";
   document.getElementById(a).style.height="150px";
}
```

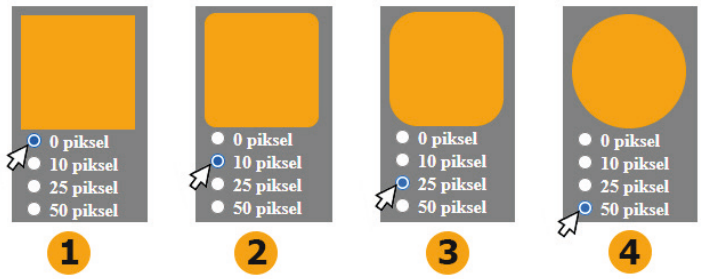
Yukarıdaki örnek kodda "a" parametresinin aldığı değere göre ID (kimlik) seçicinin seçeceği eleman değişecektir.

```
yeniBoyut("resim");
```

Fonksiyon, yukarıdaki örnek kodda görüldüğü gibi parametre değeri girilerek çağrılır.

3. Uygulama

Parametreli fonksiyonlar kullanarak Görsel 5.8'deki gibi dört farklı boyutta çerçeve kenarlarını yuvarlama işlemlerini yönergeler doğrultusunda gerçekleştiriniz.



Görsel 5.8: Kenar yuvarlama

1. Adım: Uygulama klasörünüzde birer adet HTML, Javascript ve CSS dosyası oluşturarak HTML sayfasına gerekli bağlantı kodlarını yazınız.

2. Adım: Sayfanın arka plan rengini gri yapınız, sayfaya genişliği ve yüksekliği 100 piksel olan turuncu bir kutu oluşturunuz.

HTML

```
<div  id="cerceve"> </div>
<form>
   <input type="radio" name="secim" onClick="kenarYuvarla('0px')"  value="0">
   <label for="male">0 piksel</label><br>
   <input type="radio" name="secim" onClick="kenarYuvarla('10px')" value="10">
   <label for="male">10 piksel</label><br>
   <input type="radio" name="secim" onClick="kenarYuvarla('25px')" value="25">
   <label for="male">25 piksel</label><br>
   <input type="radio" name="secim" onClick="kenarYuvarla('50px')" value="50">
   <label for="male">50 piksel</label><br>
</form>
```

CSS

```
 body{background-color: gray; color:white; font-weight: bold;}
#cerceve{background-color: orange; width: 100px; height: 100px;}
```

3. Adım: Form elemanı içine dört adet **radio buton** elemanı ekleyiniz ve tüm butonların tıklanma olayına "kenarYuvarla (derece)" parametreli fonksiyonunu atayınız.

Javascript

```
function kenarYuvarla(derece){
   document.getElementById("cerceve").style.borderRadius=derece;
}
```

4. Uygulama

Parametreli fonksiyonlar kullanarak girilen sayının karesini alma işlemlerini yönergeler doğrultusunda gerçekleştiriniz.

1. Adım: <form> elemanı içerisine sayı girişi için bir giriş elemanı ve kare alma işleminin sonucunu göstermek için "id" değeri "sonuç" olan bir paragraf elemanı ekleyiniz.
2. Adım: Giriş elemanının "onkeypress" olayına parametreli bir fonksiyon tanımlayınız ve fonksiyon parametresi ile giriş elemanının bilgilerini gönderiniz.

```
<form>
<input type="text" placeholder="Sayı giriniz." value="0"  onkeypress="kareAl(this)">
<p id="sonuc">0</p>
</form>
```

3. Adım 3: Fonksiyona parametre olarak gelen verideki "value" değerini kullanarak kare alma işlemini yapınız ve sonucu paragraf elemanına içerik olarak ekleyiniz.

Not

"kareAl" fonksiyonunda parametre olarak gönderilen "this" ifadesi hangi eleman için kullanılırsa o elemanın tüm bilgilerine ulaşma imkânı sağlar.

```
function kareAl(a){
   sayi=a.value;
   sonuc=sayi*sayi;
   document.getElementById("sonuc").innerHTML=sonuc;
}
```

5.2.3. Return Komutu

Parametreli veya parametresiz fonksiyonların tümünde yapılacak işlemlerin sonunda geriye bir değer döndürmek için "return" komutu kullanılır.

```
function megaBayt(a){
   var sonuc=0;
   sonuc = a * 1024;
   return sonuc;
   }
```

Yukarıdaki örnek kodda "a" parametresi değerine göre "sonuc" değişkeninin değeri değişecektir. Dolayısıyla fonksiyondan geriye döndürülen değer de değişecektir.

```
var x=megaBayt(100);
alert(x);
```

Fonksiyona parametre olarak "100" değeri girildiğinde fonksiyon geriye 102400 değeri gönderecektir. Bu değer bir değişkene aktarılarak veya doğrudan kullanılabilir.

```
function birlestir(ad,soyad){
   var adSoyad=ad+" "+soyad;
   return adSoyad;
}
```

Örnek kodda “ad” ve “soyad” parametrelerinin değerine göre “adSoyad” değişkeninin değeri değişecektir.

```
var y=birlestir("Zübeyde","Bektaş");
alert(y);
```

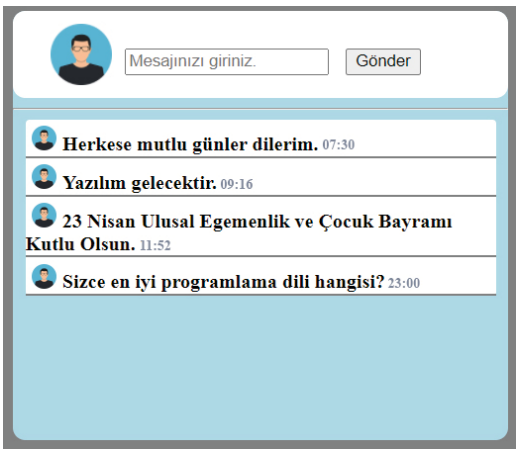
Fonksiyona parametre olarak “Zübeyde” ve “Bektaş” değerleri girildiğinde fonksiyon geriye “Zübeyde Bektaş” değerini gönderecektir.

Sıra Sizde

Üç parametreli ve geriye değer döndüren bir fonksiyon oluşturunuz. Parametrelerden gelen değerleri birbiri ile çarptırınız geriye dönen sonucu ekrana mesaj olarak verdiriniz.

5. Uygulama

Giriş elemanına yazılan mesajları Görsel 5.9’daki gibi anlık olarak ekrana listelemek için gereken işlemleri yönergeler doğrultusunda gerçekleştiriniz.



Görsel 5.9: Mesajlaşma arayüzü

1. Adım: Uygulama klasörünüzde birer adet HTML, Javascript ve CSS dosyası oluşturarak HTML sayfasına gerekli bağlantı kodlarını yazınız.

2. Adım: HTML dosyasının <body> etiketi içine aşağıdaki kodları yazınız.

HTML

```
<div id="genelCerceve">
   <div id="mesajGirisi">
      <img src="profil.png">
      <input type="text" id="mesaj"  placeholder="Mesajınızı giriniz.">
      <input type="button" value="Gönder" onclick="mesajGonder()">
   </div>
   <hr>
   <div id="mesajKutulari">
      <!-- Buraya innerHTML komutu ile kodlar eklenecek  -->
   </div>
</div>
```

3. Adım: CSS dosyası içine aşağıdaki kodları yazınız.

CSS

```
body{background-color: gray; font-weight: bold;}
#genelCerceve{background-color:lightblue; border-radius: 10px;
          width: 400px; height: 500px; }
#mesajGirisi{background-color: white;  border-radius: 10px;
          width: 400px; height: 70px;  }
#mesajGirisi > img{ width: 50px; height: 50px; float: left;
               margin-left:30px; margin-top:10px; }
#mesajGirisi>input{ margin-top:30px; margin-left:10px;  }
#mesajKutulari{ background-color: white; border-radius: 3px;
           width: 380px; height: auto; margin-left:10px; }
.gidenMesaj{ border-bottom:2px gray solid; }
.gidenMesaj > img{ width: 20px; height: 20px; margin:5px 5px 0px 5px; }
#tarih{ color:gray; font-size:10px;  }
```

4. Adım: Javascript dosyası içine aşağıdaki kodları yazınız.

Javascript

```
function mesajGonder(){
   var mesaj=document.getElementById("mesaj").value;
   var d = new Date();
   var saat=d.getHours();
   var dakika=d.getMinutes();
   document.getElementById("mesajKutulari").innerHTML+=
       "<div class='gidenMesaj'> <img src='profil.png'>"
        + mesaj +
       "<span id='tarih'>"+saat+":"+dakika+"</span></div>";
   document.getElementById("mesaj").value="";
}
```

5.3. Kontrol Yapıları

Kontrol yapıları; tüm programlama dilleri için vazgeçilmezdir. Belli bir koşula veya koşullar dizisine bağlı bir şekilde programın nasıl bir yol ile ilerleyeceğini belirleyen yapılardır (Görsel 5.10).

Görsel 5.10: Kontrol yapıları

Programın yol ayrımı noktalarını, kontrol yapıları temsil eder. Bu yapı sayesinde web sayfaları da daha dinamik ve etkileşimli hâle getirilebilir. Örneğin kullanıcıya çevrimiçi bir test yapabilir ve kullanıcının verdiği cevaplara göre testin sonucu ekrana yansıtılabilir. Karar ifadelerinde Tablo 5.4'te verilen karşılaştırma operatörleri ve Tablo 5.5'te verilen mantıksal operatörler kullanılmaktadır.

5.3.1. if Yapısı

if, kelime olarak **eğer** anlamına gelir. Sadece şart sağlandığında çalışması istenen kodlar için kullanılır. Şart ifadesi sağlandığında **true**, sağlanmadığında **false** değeri oluşur. if (true) olduğunda if yapısına bağlı kodlar çalışır, if (false) olduğunda kodlar çalışmaz.

```
if("şart ifadesi")
{
  //Şart ifadesi sağlandığında çalıştırılacak kodlar
}
```

6. Uygulama

Kullanıcının web sayfasına girdiği zamana göre reklam gösterimini ayarlama işlemlerini yönergeler doğrultusunda gerçekleştiriniz.

1. Adım: Web sayfasına elemanı içinde reklam ürününe ait bir resim ekleyiniz.
2. Adım: Sayfanın yüklenmesi olayına reklamı kaldırma işlemlerini kontrol etmek için bir fonksiyon tanımlayınız.

HTML

```
<body onload="reklamKaldir()">
   <div id="reklam">
      <img src="urunTanitim.jpg">
   </div>
<!-- Diğer HTML kodları -->
</body>
```

3. Adım: Kullanıcının sayfaya giriş yaptığı saati kontrol ediniz eğer giriş yapılan saatten 15 dakika geçtiyse reklamı kaldırınız.

Javascript

```
function reklamKaldir(){
   var d = new Date();
   var dakika=d.getMinutes();
   if(dakika >= 15){
      document.getElementById("reklam").style.display="none";
   }
}
```

Çoklu Şart İfadeleri: Bazı durumlarda birden fazla şart ifadesinin aynı anda kontrol edilmesi gerekebilir. Bu durumlarda şart ifadeleri arasına **&& (ve)**, **||(veya)** mantıksal operatörleri konulabilir.

```
var katNo, daireNo;
katNo=document.getElementById("katNo").value;
daireNo=document.getElementById("daireNo").value;
if(katNo==8 && daireNo==32){
   //kodlar
}
```

Girilen adres bilgilerinin her ikisi de doğruysa kodlar çalışacaktır.

```
var ay;
ay=document.getElementById("ay").value;
if(ay=="Haziran" || ay=="Temmuz" || ay=="Ağustos"){
   //kodlar
}
```

Girilen ay bilgisi **Haziran, Temmuz, Ağustos** aylarından herhangi birine eşitse kodlar çalışacaktır.

Sıra Sizde

Giriş elemanlarını kullanarak kullanıcı adı ve şifre bilgilerini alınız eğer kullanıcı adı "Javascript" ve şifre "12345" ise ekrana "Girişiniz Onaylandı." değilse "Girişiniz Onaylanmadı." şeklinde mesaj verdiriniz.

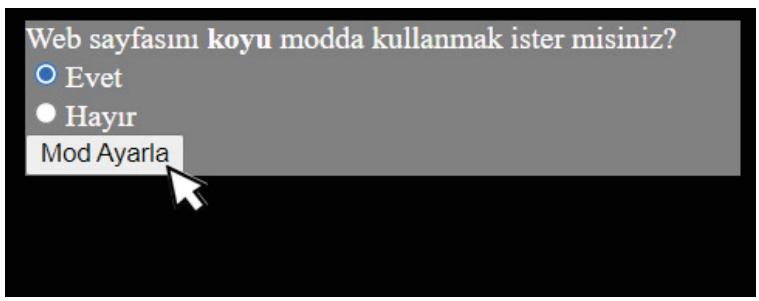
5.3.2. if-else Yapısı

Else, kelime olarak **değilse, aksi durumda** anlamına gelmektedir. Şart ifadesi sağlandığında **if** kod bloku içindeki kodlar çalışır. Şart ifadesi sağlanmadığında ise **else** kod bloku içindeki kodlar çalışır.

```
if("şart ifadesi")
{
  //Şart ifadesi sağlandığında çalıştırılacak kodlar
}
else
{
  //Şart ifadesi sağlanmadığında çalıştırılacak kodlar
}
```

7. Uygulama

Kullanıcının seçimine göre görüntü modunu ayarlama işlemlerini yönergeler doğrultusunda Görsel 5.11'de görüldüğü gibi yapınız.



Görsel 5.11: Görüntü modu

1. Adım: elemanı içine "Web sayfasını koyu modda kullanmak ister misiniz?" metni, **id** değerleri "evet" ve "hayır" olan iki adet radio buton ve bir adet buton ekleyiniz.

2. Adım: Butonun tıklanma olayına görüntü modunu değiştirmek için fonksiyon ekleyiniz.

HTML

```
<div  style="background-color: gray;">
 <form>
  Web sayfasını <b>koyu</b> modda kullanmak ister misiniz?<br>
  <input type="radio" name="mod" id="evet"  >Evet<br>
  <input type="radio" name="mod" id="hayır" >Hayır<br>
  <input type="button" value="Mod Ayarla" onclick="modDegis()">
 </form>
</div>
```

3. Adım: Eğer "id" değeri evet olan radio buton işaretli ise sayfanın arka plan rengini siyah yazı rengini beyaz yapınız. Eğer radio buton işaretli değilse sayfanın arka plan rengini beyaz yazı rengini siyah yapınız.

Javascript

```
function modDegis(){
  if(document.getElementById("evet").checked==true){
    document.getElementsByTagName("body")[0].style.backgroundColor="black";
    document.getElementsByTagName("body")[0].style.color="white"
  }
  else{
   document.getElementsByTagName("body")[0].style.backgroundColor="white";
   document.getElementsByTagName("body")[0].style.color="black"
  }
}
```

5.3.3. else if Yapısı

Şartın sağlanmadığı durumlarda **else if** kullanılarak yeni bir şart ifadesi daha yazılabilir. **else if** ifadesinden sonra, tekrar **else if** ifadesi veya sadece **else** ifadesi kullanılabilir.

```
if("şart ifadesi")
{
  //Şart ifadesi sağlandığında çalıştırılacak kodlar
}
else if("şart ifadesi")
{
  //else if şart ifadesi sağlandığında çalıştırılacak kodlar
}
```

8. Uygulama

Girilen net sayısına göre test sonucunu ekrana yazdırma işlemlerini yönergeler doğrultusunda gerçekleştiriniz.

1. Adım: id değeri "netler" olan giriş elemanı, id değeri "ekran" olan paragraf elemanı ve value değeri "Kontrol Et" olan buton elemanı ekleyiniz.

2. Adım: Buton elemanının tıklanma olayına fonksiyon tanımlayınız.

HTML

```
<input type="text" id="netler">
<input type="button"  value="Kontrol Et" onclick="kontrol()">
<p id="ekran">Test sonucu burada gösterilecek.</p>
```

3. Adım: Girilen net sayısı "0"dan az ise "Geçersiz sayı!", "0-10" arasında ise "Net sayınız ortalamanın altında", "10-25" arasında ise "Net sayınız ortalamaya yakın", "25-41" arasında ise "Net sayınız ortalamanın üstünde" mesajlarını paragraf elemanı içine yazdırınız.

Javascript

```
function kontrol(){
   var netSayisi=document.getElementById("netler").value;
   if(netSayisi<0){
      document.getElementById("ekran").innerText="Geçersiz sayı!";
    }
    else if(netSayisi<10){
     document.getElementById("ekran").innerText="Net sayınız ortalamanın altında";
    }
    else if(netSayisi<25){
     document.getElementById("ekran").innerText="Net sayınız ortalamaya yakın";
    }
    else if(netSayisi<=40){
     document.getElementById("ekran").innerText="Net sayınız ortalamanın üstünde";
    }
    else{
     document.getElementById("ekran").innerText="Geçersiz sayı!";
     }
}
```

5.3.4. Switch-Case

Switch-case, bir ifadenin aldığı değere bağlı olarak program için birçok farklı çalışma yolu belirleyen komuttur. Switch ifadesindeki değer, hangi durumun değeri ile eşleşiyorsa o duruma ait kodlar çalışır. Hiçbir durum ile eşleşme olmazsa **default** ifadesinde belirtilen kodlar çalışır.

"break;" komutu, kodlar çalıştıktan sonra **switch-case** ifadesinden çıkmayı sağlar. Switch ifadesi içine yazılan değerin veri türü ile case ifadelerindeki değerin veri türü aynı olmalıdır.

```
switch(değer) {
 case 1:
  //kodlar
 break;
 case 2:
  //kodlar
 break;
 default:
  // kodlar
}
```

```
switch(değer) {
 case "sarı":
  //kodlar
 break;
 case "kırmızı":
  //kodlar
 break;
 default:
  // kodlar
}
```

```
switch(değer) {
 case " A ":
  //kodlar
 break;
 case " B ":
  //kodlar
 break;
 default:
  // kodlar
}
```

9. Uygulama

Sistemden alınan tarih bilgisine göre haftanın hangi gününde bulunuluyorsa o günün ismini mesaj olarak gösterme işlemlerini yönergeler doğrultusunda gerçekleştiriniz.

1. Adım: Web sayfasına bir adet buton elemanı ekleyiniz ve buton elemanının tıklanma olayına fonksiyon tanımlayınız.

```
<input type="button" onclick="hangiGun()" value="Kontrol Et">
```

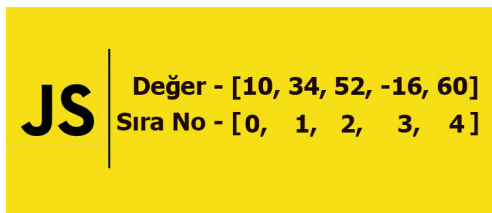
2. Adım: Fonksiyon içinde sistem tarihinden gün bilgisini alınız ve değişkene aktarınız.
3. Adım: Gün bilgisi; 0 ise "Pazar", 1 ise "Pazartesi", 2 ise "Salı", 3 ise "Çarşamba", 4 ise "Perşembe", 5 ise "Cuma", 6 ise "Cumartesi" mesajlarını verdiriniz.

Javascript

```
function hangiGun(){
  var tarih = new Date();
  var gun = tarih.getDay();
  switch(gun) {
    case 0:
      alert("Pazar");
    break;
    case 1:
      alert("Pazartesi");
    break;
    case 2:
      alert("Salı");
    break;
    case 3:
      alert("Çarşamba");
    break;
    case 4:
      alert("Perşembe");
    break;
    case 5:
      alert("Cuma");
    break;
    case 6:
      alert("Cumartesi");
  }
}
```

5.4. Diziler

Tek bir değişken adı kullanarak birçok değeri tutmak için kullanılan yapıya dizi denir. Bu yapı hem kod yazımını kolaylaştırır hem de verileri organize bir şekilde kullanmayı sağlar (Görsel 5.12).



Görsel 5.12: Diziler

```
sehirler="İzmir";
sehirler1="İstanbul";
sehirler2="Ankara";
sehirler3="Bursa";
sehirler4="Ordu";
```

Örneğin, dizileri kullanmadan şehir isimlerini hafızada tutmak istendiğinde her eleman için farklı bir değişken ismi yazmak gerekir. Bu tür bir kullanım ile hem değişkenleri tanımlamak zordur hem de tanımlanmış değişkenlerden değer okumak zordur.

Dizi Tanımlama: Diziler **içi boş köşeli ayraç ([ ])** ile tanımlanır.

```
var sehirler =[];
var arabalar =[];
var ogrenciler =[];
```

Dizi Elemanlarına Değer Atama: Dizi elemanlarına değer atama işlemleri, dizi tanımlama işlemi sırasında veya sonrasında toplu olarak yapılabilir.

```
var sehirler = ["İzmir", "İstanbul", "Ankara", "Bursa", "Ordu"]; //tanımlanırken
sehirler = ["İzmir", "İstanbul", "Ankara", "Bursa", "Ordu"]; //tanımlandıktan sonra
```

Dizi yapısı içindeki elemanlar aynı değişken ismine bağlı sıra numarası ile kullanılarak içine değer atanabilir. Sıra numaraları **[ ] (köşeli ayraç)** içine yazılır. İlk sıra numarası "0"dır.

```
sehirler[0]="İzmir";
sehirler[1]="İstanbul";
```

Dizi Elemanlarından Değer Okuma: Dizi yapısı içindeki elemanlar, aynı değişken ismine bağlı sıra numarası ile kullanılarak içindeki değer okunabilir.

```
document.getElementById("ekran").innerHTML=sehirler[0];
alert("Web sayfasına "+ sehirler[4] +" ilinden bağlanıyorsunuz");
```

Not

id seçiciler hariç tüm seçiciler, elemanlar üzerindeki değişiklikleri diziler ile yapar.

```
document.getElementsByClassName("sinifAdi")[4].innerText="Eklenecek Metin";
document.getElementsByName("elemanAdi")[2].style.fontSize="16px";
document.getElementsByTagName("div")[0].style.color="white";
```

Diziye Yeni Eleman Ekleme: "push (eklenecek değer)" fonksiyonu ile dizinin sonuna yeni bir eleman eklenir.

```
var urunler=[];
urunler.push(document.getElementById("urun").value);
```

Dizi Eleman Sayısı Bulma: "length()" fonksiyonu bir dizinin kaç elemana sahip olduğu bilgisini verir.

```
urunler.length();
```

Dizi Elemanlarını Sıralama: "sort()" fonksiyonu ile bir dizideki tüm elemanlar A'dan Z'ye alfabetik olarak sıralanır.

```
urunler.sort();
```

"reverse" fonksiyonu kullanılarak dizideki elemanları tersten sıralama işlemi yapılır.

```
urunler.reverse();
```

Diziden Eleman Çıkarma: "pop()" fonksiyonu ile dizinin sonundaki eleman kaldırılır.

```
urunler.pop();
```

Dizide Arama Yapma: "indexOf (aranacak değer)" fonksiyonu ile dizi içerisinde arama işlemi yapılır. Aranan değer hangi dizi elemanında bulunduysa o dizi elemanının sıra numara bilgisini verir.

```
urunler. indexOf(document.getElementById("arananUrun").value);
```

10. Uygulama

Ders not bilgilerini dizilerde tutarak not ortalaması bulma işlemlerinin Javascript kodlarını yönergeler doğrultusunda yazınız.

1. Adım: Üç farklı ders için gerekli not bilgilerini dizilere aktarınız.

```
var matematik=[60,90,100];
var tarih=[80,80];
var fizik=[100,90,75];
```

2. Adım: Her dersin ayrı ayrı ortalamasını alınız ve oluşan ortalamaları bir diziye aktarınız.

```
ort1=(matematik[0]+matematik[1]+matematik[2])/3;
ort2=(tarih[0]+tarih[1])/2;
ort3=(fizik[0]+fizik[1]+fizik[2])/3;
ortalamalar.push(ort1);
ortalamalar.push(ort2);
ortalamalar.push(ort3);
```

3. Adım: Tüm ortalamaların aktarıldığı dizinin ortalamasını mesaj olarak verdiriniz.

```
var genelOrtalama=0;
genelOrtalama=(ortalamalar[0]+ortalamalar[1]+ortalamalar[2])/3;
alert(genelOrtalama);
```

11. Uygulama

Girilen İngilizce kelimenin Türkçe çevirisini bulma işlemlerini yönergeler doğrultusunda Görsel 5.13'te görüldüğü gibi yapınız.

Görsel 5.13: Dijital sözlük

1. Adım: Web sayfasının arka plan rengini turuncu yapınız.

2. Adım: elemanı içine sırasıyla İngilizce kelimenin girileceği bir giriş elemanı, tıklanma olayında kelimeyi çevirme fonksiyonunu çağıran buton elemanı ve kelimenin Türkçe karşılığının gösterileceği id değeri "turkce" olan <h3> elemanı ekleyiniz.

HTML

```
<body style="background-color: orange;">
<div>
<input type="text" id="arananKelime" placeholder="İngilizce Kelime">
<input type="button" value="Çevir" onclick="cevir()">
<h3 id="turkce" >Kelimenin Türkçe Karşılığı</h3>
</div>
</body>
```

3. Adım: İngilizce ve Türkçe karşılıklarını, fonksiyonu tanımlamadan önce dizilere ekleyiniz.

4. Adım: Çevirisi yapılmak istenilen kelimeyi İngilizce kelimelerin bulunduğu dizide aratıp bulunuz.

5. Adım: Eğer İngilizce kelimelerin içinde aranılan kelime bulunursa <h3> elemanına aranılan kelimenin Türkçe karşılığını yazdırınız.

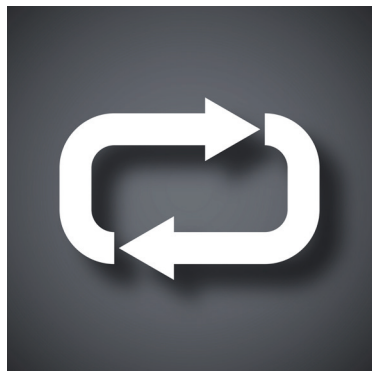
Javascript

```
var ingilizce=["apple","bottle","computer","book","mouse","wall","pen","paper","table"];
var turkce=["elma","şişe","bilgisayar","kitap","fare","duvar","kalem","kağıt","masa"];
   function cevir(){
 var siraNo;
 siraNo=ingilizce.indexOf(document.getElementById("arananKelime").value);
    if(siraNo>=0){
  document.getElementById("turkce").innerText=turkce[siraNo];
    }
```

5.5. Döngüler Zamanlayıcılar ve Popüler Javascript Kütüphaneleri

Programda belirli kodların tekrar tekrar çalıştırılmasını sağlayan yapılara **döngü** denir (Görsel 5.14). Program gereği aynı kodlar iki kez çalıştırılabileceği gibi 200 kez hatta 2000 kez de çalıştırılabilir. Böyle durumlarda program yazmak zorlaşacak, zaman alacak ve programın kod yapısı karmaşıklaşacaktır.

Örneğin; 500 adet ürün arasından, girilen barkod numarasına ait ürünün bilgilerini getirmek için 500 adet if komutu kullanmak yerine, döngü ifadesi içinde sadece bir adet if komutu kullanılabilir.



Görsel 5.14: Döngü

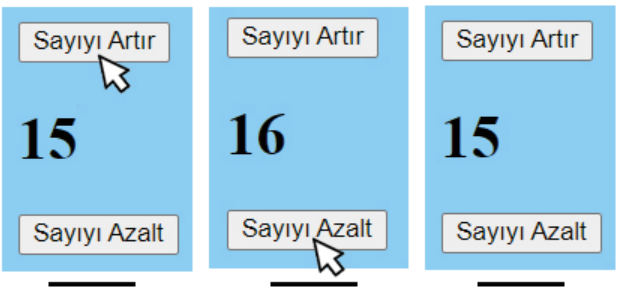
5.5.1. Sayaçlar

Bir değişkene bağlı değeri farklı aralıklarla arttırmak, azaltmak, katlamak veya bölmek gerekebilir. Böyle durumlarda sayaç adı verilen değişkene değer atama yöntemleri kullanılır. Tablo 5.2'de verilen artı eşittir, eksi eşittir, çarpı eşittir, bölü eşittir atama operatörleri ile sayaçlar oluşturulur.

Sayaç birer birer artacaksa pratik bir şekilde **degiskenAdi++**; kullanılabilir. Birer birer azalacaksa pratik bir şekilde **degiskenAdi--**; kullanılabilir.

12. Uygulama

Sayaçlar kullanarak bir sayının değerini arttırma ve azaltma işlemlerini yönergeler doğrultusunda Görsel 5.15'te görüldüğü gibi yapınız.



Görsel 5.15: Sayaçlar

1. Adım: Web sayfasına arka plan rengi açık mavi, çerçeve içi boşluğu 10 piksel olan bir elemanı ekleyiniz.

2. Adım: elemanının içine iki adet buton ve id değeri "sayac" başlık elemanı ekleyiniz.

3. Adım: buton elemanlarının tıklanma olaylarına sayıyı arttırmak ve azaltmak için fonksiyon tanımlayınız.

HTML

```
<div style="background-color: lightskyblue; padding: 10px;">
   <input type="button" onclick="arttir()" value="Sayıyı Artır">
   <h1 id="sayac">0</h1>
   <input type="button" onclick="azalt()" value="Sayıyı Azalt">
</div>
```

4. Adım: Id değeri "sayac" olan elemanın içindeki sayıyı sayaçlar ile azaltıp arttırma işlemlerini yapınız.

Javascript

```
function arttir(){
document.getElementById("sayac").innerText++;
}
function azalt(){
document.getElementById("sayac").innerText--;
}
```

5.5.2. For Döngüsü

Kodların tekrar sayısı belli olduğunda genellikle **for döngüsü** kullanılır. Döngü için tanımlanan şart ifadesinin her sağlanışında döngüdeki kodlar tekrar çalışır. For döngüsünün kaç kez çalışacağını belirlemek oldukça basittir.

13. Uygulama

Dizi içinde tanımlanmış renkler ile sekiz adet kutuyu rastgele boyama işlemini yönergeler doğrultusunda Görsel 5.16'da görüldüğü gibi yapınız.

Görsel 5.16: Renkli kutular

1. Adım: HTML sayfasına içlerinde sırasıyla birden sekize kadar rakam bulunan sekiz adet elemanı ve tıklanma olayında renkleri değiştirmek için fonksiyon tanımlı olan bir buton elemanı yerleştiriniz.

HTML

```
<div>1</div> <div>2</div> <div>3</div> <div>4</div>
<div>5</div> <div>6</div> <div>7</div> <div>8</div>
<input type="button" onclick="renkDegistir()" value="Renkleri Değiştir">
```

2. Adım: Kutuların stil özelliklerinin genişliğini 20 piksel, yüksekliğini 20 piksel, kenarlık kalınlığını 2 piksel, kenarlık stilini solid, kenarlık rengini gri, çerçeve dışı boşluğunu 2 piksel, yazıları ortalı, kaydırılarak yerleşim özelliğini sol (left) olacak şekilde ayarlayınız.

CSS

```
div{
 width:20px; height:20px; border:2px solid gray;
 margin:2px; text-align:center; float:left; }
```

3. Adım: Sekiz adet renk ismini dizide tanımlayınız ve bu renkler arasında rastgele seçim yaparak kutuların arka plan renklerini değiştiriniz.

Javascript

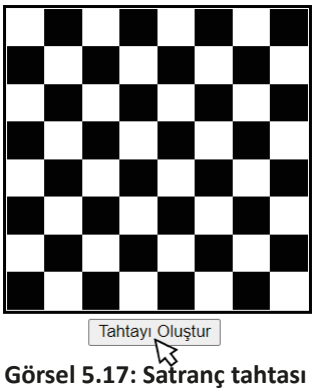
```
function renkDegistir(){
   var renkler=["red","green","blue","yellow","pink","orange","gray","white"];
   var rastgele;
   var i;
   for(i=0; i<8; i++){
   rastgele=Math.floor(Math.random() * 8); //0 ile 7 arasında rastgele tam sayı üretir.
  document.getElementsByTagName("div")[i].style.backgroundColor=renkler[rastgele];
   }
}
```

Sıra Sizde

Bir dizi oluşturarak diziye 100 adet rastgele sayı aktarınız. Dizideki elemanları büyükten küçüğe sıralayıp elemanı içinde ekrana yazdırınız.

14. Uygulama

İç içe for döngüsü kullanarak satranç tahtası oluşturma işlemlerini yönergeler doğrultusunda Görsel 5.17'de görüldüğü gibi yapınız.



Görsel 5.17: Satranç tahtası

1. Adım: Oluşturulacak satranç tahtasının çerçevesi için bir adet elemanı ve tahta oluşturmak için hazırlanacak fonksiyonu çağırmak üzere bir adet buton elemanı ekleyiniz.

HTML

```
<div id="tahtaCerceve"> </div>
<input type="button" onclick="satranc()" value="Tahtayı Oluştur">
```

2. Adım: Tahta çerçevesinin genişliğini 240 piksel, yüksekliğini 240 piksel kenarlık kalınlığını 3 piksel, kenarlık stilini solid, kenarlık rengini siyah yapınız.

3. Adım: Tahtanın siyah karelerinin; genişliğini 30 piksel, yüksekliğini 30 piksel ve float özelliğini left, arka plan rengini black yapınız.

4. Adım: Tahtanın beyaz karelerinin; genişliğini 30 piksel, yüksekliğini 30 piksel ve float özelliğini left, arka plan rengini white yapınız.

CSS

```
#tahtaCerceve{ width: 240px; height: 240px;  border:3px solid black;   }
    .kareBeyaz{ width: 30px; height:30px;  background-color: white; float: left; }
     .kareSiyah{ width: 30px; height:30px;  background-color: black;  float: left; }
```

5. Adım: Satranç tahtasını oluşturma fonksiyonu için aşağıdaki kodları <script> etiketleri arasına yazınız.

Javascript

```
function satranc(){
   var i,j;
   for(i=0;i<8;i++){
      for(j=0;j<8;j++){
         if((j+i)%2==0){
    document.getElementById("tahtaCerceve").innerHTML+="<div class='kareBeyaz'>
</div>";
         }
         else{
    document.getElementById("tahtaCerceve").innerHTML+="<div class='kareSiyah'> </
div>";
         }
      }
   }
}
```

5.5.3. While Döngüsü

Döngü için tanımlanan şart ifadesi sağlanıyorsa döngü çalışmaya başlar. Şart ifadesi sağlandığı sürece while döngüsü çalışmaya devam eder.

5.5.4. Do-While Döngüsü

Döngü için tanımlanan şart ifadesi sağlanmasa da do-while döngüsü en az bir kez çalışır. Çünkü while döngüsünde şart ifadesi döngünün başlangıcındayken do-while döngüsünde şart ifadesi döngünün sonundadır. Do-while döngüsü de diğer döngüler gibi şart sağlandığı sürece çalışmaya devam eder.

15. Uygulama

While ve do-while döngülerini kullanarak ekrana sıfırdan ona kadar ve ondan sıfıra kadar olan sayıları yazdıran Javascript kodunu yazma işlemini yönergeler doğrultusunda gerçekleştiriniz.

1. Adım: Sayıların sayfada görüntülenmesi için id değerleri “ekran1” ve “ekran2” olan iki adet elemanı ekleyiniz.

HTML

```
<div id="ekran1"> </div>
<div id="ekran2"> </div>
```

2. Adım: Başlangıç değeri “0” olan **sayi1** adında bir değişken oluşturunuz ve sayi1 değişkeninin değerini while döngüsü ile “10”a kadar arttırıp oluşan değeri, id değeri “ekran1” olan elemanına yazdırınız.

3. Adım: Başlangıç değeri "10" olan **sayi2** adında bir değişken oluşturunuz ve sayi2 değişkeninin değerini do-while döngüsü ile "0" a kadar azaltıp oluşan değeri, id değeri "ekran1" olan elemanına yazdırınız.

Javascript

```
var sayi1=0;
while(sayi1<=10){
 document.getElementById("ekran1").innerHTML+=sayi1;
sayi1++;
}
var sayi2=10;
do{
   document.getElementById("ekran2").innerHTML+=sayi2;
sayi2--;
}while(sayi2>=0)
```

5.5.5. Zamanlayıcılar

İstenilen kodları belirli zaman aralıklarında çalıştırmak için **zamanlayıcılar** kullanılır. Örneğin, zamanlayıcı kullanılarak ekrana, dakikada bir uyarı mesajı verdirebilir veya kullanıcı, web sayfasını görüntüledikten 10 saniye sonra reklam kutusu görünür hâle getirilebilir.

İki tür zamanlayıcı bulunur.

Tek Seferlik Zamanlayıcı (setTimeout): Ayarlanan zaman aralığı geldiğinde içindeki kodları sadece bir kez çalıştıran ve durduran zamanlayıcıdır.

```
setTimeout(() => {
   document.getElementById("reklam").style.display="none";
}, 5000);
```

Not

Yukarıdaki zamanlayıcı kodu çalıştırıldıktan beş saniye sonra id değeri reklam olan eleman gizlenecektir.

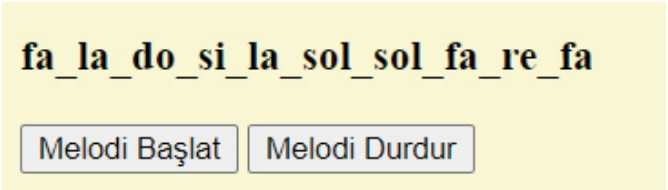
Tekrar Eden Zamanlayıcı (setInterval): Ayarlanan zaman aralığı geldiğinde içindeki kodları çalıştıran ve çalışmasını devam ettiren zamanlayıcıdır. Bu zamanlayıcı durdurulmadığı sürece sürekli çalışır.

```
var sayac=10;
var a=setInterval(function (){
   sayac=sayac-1;
   document.getElementById("ekran").innerHTML=sayac;
   if (sayac==0){
              clearInterval(a);
             }
                    },3000);
```

Yukarıdaki örnek kodda "sayac" değişkeninin değeri sıfır olana kadar her üç saniyede bir, id değeri "ekran" olan elemana metin olarak "sayac" değişkeninin değeri yazdırılmaktadır.

16. Uygulama

Rastgele üretilen notalardan melodi oluşturma işlemlerini yönergeler doğrultusunda Görsel 5.18'de görüldüğü gibi yapınız. Her nota 2 saniye arayla melodiye eklenecektir.



Görsel 5.18: Notalar

1. Adım: Web sayfasına notaların görüntüleneceği id değeri "potre" olan <h3> elemanı ve melodi başlatma, durdurma işlemlerinin kontrolü için iki adet buton ekleyiniz.

HTML

```
<h3 id="potre"> </h3>
<input type="button" onclick="melodiBaslat()" value="Melodi Başlat">
<input type="button"onclick="melodiDurdur()" value="Melodi Durdur">
```

2. Adım: Yedi adet notayı diziye aktarınız. İki saniyede bir tekrar eden zamanlayıcı oluşturunuz.

3. Adım: Zamanlayıcı içinde sıfır ile altı arasında rastgele sayı üretiniz.

4. Adım: Üretilen sayıyı notaların eklendiği dizi elemanında kullanarak <h3> elemanına dizinin ilgili değerini yazdırınız.

Javascript

```
var metronom;
function melodiBaslat(){
   var notalar=["do","re","mi","fa","sol","la","si"];
  metronom=setInterval(() => {
   rastgele=Math.floor(Math.random() * 7); //0 ile 6 arasında rastgele tam sayı üretir.
   document.getElementById("potre").innerText+=notalar[rastgele] +"_";
  }, 2000);
}
function melodiDurdur(){
   clearInterval(metronom);
}
```

Sıra Sizde

Sayfaya 60'tan sıfıra kadar sayıları bir dakika boyunca yazdırınız. Yazdırma işlemi devam ederken son dört sayıyı rakamla değil yazı kullanarak yazdırınız (3, 2, 1, 0 yerine üç, iki, bir, sıfır).

5.5.6. Popüler Javascript Kütüphaneleri

Birçok fonksiyon bir çatı altında toplanarak **kütüphane** kavramını oluşturur. Javascript kullanımını kolaylaştıran, daha az kod yazarak daha çok işlem gerçekleştirmeyi sağlayan birçok popü-

ler Javascript kütüphanesi bulunur. Bu kütüphanelerin yapısında çok kullanışlı fonksiyonlar bulunur ve bu kütüphanelerin çoğu açık kaynak kodlu ve ücretsizdir. jQuery, React, Angular, Vue gibi Javascript kütüphaneleri bulunur.

5.5.7. jQuery Kütüphanesi

jQuery kütüphanesi ile Javascript ayrı ayrı diller olarak düşünülmemelidir. jQuery kütüphanesi, standart Javascript kod yazımına farklı bir boyut kazandırmış Javascript komutlarından oluşan bir kütüphanedir. Standart Javascript kodlarının hepsi jQuery kodları ile birlikte kullanılabilir. jQuery, çok tercih edilen Javascript kütüphanelerindendir. jQuery ile ilgili içeriklere kendi web adresinden ücretsiz olarak ulaşılabilir (Görsel 5.19).

Görsel 5.19: jQuery logo

jQuery kütüphanesinin özellikleri;

- Kolay ve anlaşılabilir kod yapısı,
- Olay ve olaya ait fonksiyonları pratik olarak tanımlama kolaylığı,
- Web sayfasındaki elemanların stil ve içerik özelliklerine müdahale kolaylığı,
- İçeriği ile ilgili kaynak, doküman ve örneklere ulaşma kolaylığı,
- Neredeyse tüm tarayıcılarla uyumlu çalışması,
- Çeşitli efekt ve animasyon desteği,
- Açık kaynaklı olduğu için içindeki kaynak kodlarına müdahale edebilme olanağı,
- Her geçen gün güncellenen ve geliştirilen bir kütüphane olması şeklinde sıralanabilir.

jQuery kütüphanesini web sitelerinde kullanmak için gerekli bağlantıların yapılması gerekir. Bağlantı linklerine jQuery kütüphanesi web adresinden ulaşılabilir.

```
<script src="https://code.jquery.com/jquery-3.6.0.js"></script>
```

Yukarıdaki bağlantı linki, HTML <head> etiketleri arasına yazıldığında jQuery kütüphanesinin ilgili versiyonu kullanılabilir.

5.5.7.1. jQuery Kütüphanesi Seçici Kullanımı

jQuery kütüphanesinde kimlik, sınıf, etiket seçiciler için kullanılan

```
document.getElementById("icerik")
document.getElementsByClassName("icerik")
document.getElementsByTagName("p")
```

şeklindeki kodlar aşağıdaki gibi daha kısa bir yazım şekli ile kullanılır.

```
$("#icerik")
$(".icerik")
$("p")
```

Not

Her seçici kullanımında, seçilen elamanlar için aynı kod blokunda stil değişikliği yapılabilir, çeşitli görsel efektler eklenebilir ve olay fonksiyonu oluşturulabilir.

5.5.7.2. jQuery Kütüphanesi Bazı Temel Fonksiyonlar

Sayfa Hazır Fonksiyonu (ready): Ready fonksiyonu içine yazılan diğer kodlar, web sayfası tüm yüklemelerini gerçekleştirip hazır olduğunda çalışacaktır.

```
<script>
  $(document).ready(function(){
    //kodlar
  });
</script>
```

Yukarıdaki fonksiyon neredeyse tüm jQuery kütüphanesi kodlarını kapsayan ana fonksiyondur.

Gizle Göster (hide-show) Fonksiyonu: Seçili elemanları istenilen zaman aralığında gizleyen veya elemanlar gizliyse istenilen zaman aralığında elemanları tekrar gösteren görsel efekt fonksiyonudur. Eleman gizlendiğinde sayfadaki yerine başka eleman yerleşir.

```
$(document).ready(function(){
    $(".resim").hide(3000); //3 saniye
    $(".resim").show(1000); //1 saniye
});
```

Solup Açılma (fadeOut-fadeIn) Fonksiyonu: Seçili elemanların görünürlüğünü istenilen zaman aralığında geçişli bir şekilde değiştiren efekt fonksiyonudur. Eleman gizlendiğinde sayfadaki yerine başka eleman yerleşmez.

```
$(document).ready(function(){
    $("div").fadeOut(1000); //1 saniye
    $("div").fadeIn(5000);  //5 saniye
});
```

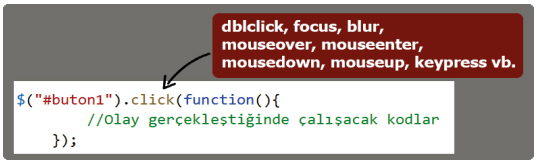
Aşağı Yukarı Kaydırma (slideDown-slideUp) Fonksiyonu: Seçili elemanların istenilen zaman aralığında alttan üste doğru görünürlüğünü ortadan kaldıran veya tam tersi şekilde görünürlüğünü üstten alta doğru ortaya çıkaran efekt fonksiyonudur. Eleman gizlendiğinde sayfadaki yerine başka eleman yerleşir.

```
$(document).ready(function(){
    $("#icerik").slideDown(2000); //2 saniye
    $("#icerik").slideUp(500);    //0,5 saniye
});
```

Animasyon (animate) Fonksiyonu: Seçili elemanların istenilen zaman aralığında belirtilen özelliklerini, geçişli bir şekilde değiştiren efekt fonksiyonudur.

```
$(document).ready(function(){
   $("#kutu").animate({left:'250px',height:'150px',width:'150px'}, 1000);
});
```

Olay Fonksiyonları: Standart olarak Javascript olayları HTML etiketi içerisinde tanımlanır ve olay gerçekleştiğinde bir fonksiyon çağırır. jQuery kütüphanesi ile elemana ait olay ve olay gerçekleşeceğinde çalışması istenilen kodlar, aynı kod bloğuna yazılır.



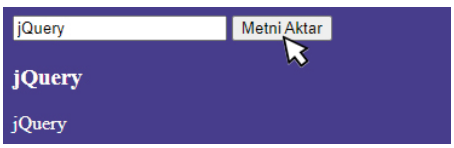
Görsel 5.20: Olay fonksiyonları

Görsel 5.20'deki örnek koda göre id değeri "buton1" olan elemana tıklanma olayı gerçekleştiğinde istenilen kodlar çalışacaktır.

Değer Okuma ve Yazma Fonksiyonları: jQuery kütüphanesinde standart Javascript kodlarında bulunan **innerHTML** özelliği yerine **html()** fonksiyonu, **innerText** özelliği yerine **text()** fonksiyonu, **value** özelliği yerine **val()** fonksiyonu kullanılır.

17. Uygulama

jQuery kütüphanesi değer okuma yazma fonksiyonları ile metin aktarma işlemlerini yönergeler doğrultusunda Görsel 5.21'de görüldüğü gibi yapınız.



Görsel 5.21: Değer okuma ve yazma

1. Adım: HTML sayfasına haricî jQuery kütüphanesi bağlantısını yapınız.

2. Adım: Sayfa arka planını mor, yazı rengini beyaz yapınız.

3. Adım: Id değeri "aciklama" olan metin giriş elemanı, id değeri "buton1" olan buton elemanı, id değerleri "icerik1" ve "icerik2" olan paragraf elemanları ekleyiniz.

HTML

```
<head>
   <scriptsrc="https://code.jquery.com/jquery-3.6.0.js"></script>
</head>
<body style="background-color:purple; color:white">
   <input type="text" placeholder="Açıklama Yazınız" id="aciklama">
   <input type="button" id="buton1" value="Metni Aktar">
   <p id="icerik1"></p>
   <p id="icerik2"></p>
</body>
```

4. Adım: jQuery kütüphanesi "ready" fonksiyonu içinde, buton elemanının tıklanma olay fonksiyonunu oluşturunuz. Metin giriş kutusundaki değeri, paragraf elemanına <h3> başlık elemanı ile ve başlıksız bir şekilde ekleyiniz.

Javascript

```
$(document).ready(function(){
   $("#buton1").click(function(){
      var a=$("#aciklama").val();
      $ ( "#icerik1").html("<h3>"+a+"</h3>");
      $ ( "#icerik2").text(a);
   });
});
```

İçerik Ekleme ve Çıkarma Fonksiyonları: Seçili elemanlar üzerinde içerik ekleme ve çıkarma işlemleri için hazırlanmış fonksiyonlar Tablo 5.7'de verilmiştir. Eklenecek içerikler, herhangi bir metin veya herhangi bir elemanın etiketi de olabilir.

Tablo 5.7: İçerik Fonksiyonları

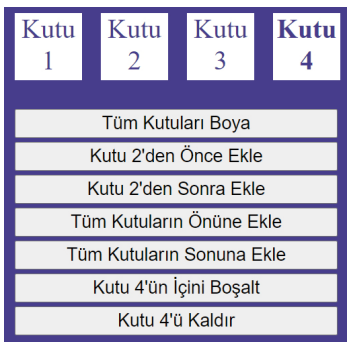
FONKSİYON	AÇIKLAMA	KULLANIMI
append()	Seçili elemanların sonuna içerik ekler.	$("p").append("Eklenecek içerik");
prepend()	Seçili elemanların başına içerik ekler.	$("p").prepend(" Eklenecek içerik ");
before()	Seçili elemanlardan önce içerik ekler.	$("p").befor("Eklenecek içerik");
after()	Seçili elemanlardan sonra içerik ekler.	$("p").after(" Eklenecek içerik ");
remove()	Seçili elemanları ve alt öğelerini kaldırır.	$("p").remove();
empty()	Seçili elemanların alt öğelerini kaldırır.	$("p").empty();

Stil Ekleme Fonksiyonu: Seçili elemanlar üzerinde stil değişikliği yapabilmek için css() fonksiyonu kullanılır.

```
$ (".baslik ").css({" color ": " red ");
$ ("#kutu ").css({" background-color ": " orange ", " font-size ": " 16px "});
$ (" div ").css({" color ": " orange ", " font-size ": " 16px ", " width ": " 168px" });
```

18. Uygulama

jQuery kütüphanesi ile sayfaya yeni içerik ekleme ve elemanların stil özelliklerini değiştirme işlemlerini yönergeler doğrultusunda Görsel 5.22'de görüldüğü gibi yapınız.



Görsel 5.22: İçerik fonksiyonları

1. Adım: Uygulama klasörünüzde birer adet HTML, Javascript ve CSS dosyası oluşturarak HTML sayfasına gerekli bağlantı kodlarını yazınız.

```
<script src= " https://code.jquery.com/jquery-3.6.0.js " ></script>
<link rel= " stylesheet" href= "stil.css" >
```

2. Adım: HTML dosyasının içine aşağıdaki kodu yazınız.

HTML

```
<body style="background-color: darkslateblue;color:white">
<div id="cerceve">
<span id="kutu1">Kutu <br> 1</span>
<span id="kutu2">Kutu <br>2</span>
<span id="kutu3">Kutu <br> 3</span>
<span id="kutu4"><b>Kutu <br>4</b></span>
</div>
<br>
<input type="button" id="islem1"  value="Tüm Kutuları Boya"><br>
<input type="button" id="islem2"  value="Kutu 2'den Önce Ekle"><br>
<input type="button" id="islem3"  value="Kutu 2'den Sonra Ekle"><br>
<input type="button" id="islem4"  value="Tüm Kutuların Önüne Ekle"><br>
<input type="button" id="islem5"  value="Tüm Kutuların Sonuna Ekle"><br>
<input type="button" id="islem6"  value="Kutu 4'ün İçini Boşalt"><br>
<input type="button" id="islem7"  value="Kutu 4'ü Kaldır"><br>
</body>
```

3. Adım: CSS dosyasına aşağıdaki kodu yazınız.

CSS

```
 span{ width:50px;  height:50px;  background-color:white;
      margin-right:10px;display: inline-block;font-size:20px;
      color:darkslateblue; text-align: center; }
input{ width: 240px;margin-top: 3px; }
```

4. Adım: Javascript dosyasına aşağıdaki kodu yazınız.

Javascript

```
$(document).ready(function(){
  $("#islem1").click(function(){
     $("span").css({"backgroundColor":"tomato","color":"white"});
  });
  $("#islem2").click(function(){
     $("#kutu2").before("<span><u>Yeni<br>Kutu</u></span>");
  });
  $("#islem3").click(function(){
     $("#kutu2").after("<span><u>Yeni<br>Kutu</u></span>");
  });
  $("#islem4").click(function(){
     $("#cerceve").prepend("<span><u>Yeni<br>Kutu</u></span>");
  });
  $("#islem5").click(function(){
     $("#cerceve").append("<span><u>Yeni<br>Kutu</u></span>");
  });
  $("#islem6").click(function(){
     $("#kutu4").empty();
  });
  $("#islem7").click(function(){
     $("#kutu4").remove();
  });
});
```

ÖLÇME VE DEĞERLENDİRME

A. Aşağıdaki cümlelerde boş bırakılan yerlere doğru sözcükleri yazınız.

1. Javascript dilinde elemanları seçmek için etiket (tag) seçici, isim (name) seçici, kimlik (id) seçici ve …… seçicileri kullanılır.

2. Seçili elemanlara yeni içerik veya HTML kodu eklemek için …… Javascript kodu kullanılır.

3. document.getElementsByClassName("kutular")[3].style.fontSize="22px"; kodu, sınıf adı "kutular" olan …… numaralı elemanı seçer.

4. Parametreli veya parametresiz fonksiyonların tümünde yapılacak işlemlerin sonunda, geriye bir değer döndürmek için …… komutu kullanılır.

B. Aşağıdaki açık uçlu soruların cevaplarını ilgili boşluklara yazınız.

5. Javascript dilindeki kontrol yapıları nelerdir? Yazınız.

6. var sehirler = ["İzmir", "İstanbul", "Ankara", "Bursa", "Ordu"]; kodundaki değerleri, dizi yerine değişkenleri kullanarak tutunuz?

```
var sehir1="İzmir";
……
……
……
……
```

7. Aşağıdaki kodlar çalıştığında mesaj olarak ekrana hangi sayı verilecektir?

```
var say=0;
for(i=0;i<4;i++){
  for(j=0;j<5;j++){
     if(j%2==0){
       say++;
     }
  }
}
alert(say);
```

8. Aşağıdaki <script> etiketleri arasına jQuery kütüphanesinin en temel fonksiyonu olan Ready fonksiyonu tanımlayınız. Fonksiyonun içinde çalışacak kodların içinde, id değeri "ekran" olan elemanın arka plan rengini gri ve yazı rengini turuncu olarak ayarlayınız.

```
<script>



</script>
```

6.
ÖĞRENME BİRİMİ

ARKA UÇ YAZILIM GELİŞTİRME

NELER ÖĞRENECEKSİNİZ?

Bu öğrenme birimi ile;

- ASP.NET Core ile ilgili temel bilgileri,
- ASP.NET Core kurulumu yapabilmeyi,
- MVC tasarım deseni ile ilgili temel bilgileri,
- Ara katman mimarisi tasarlamayı,
- Yönlendirme mekanizmalarını kullanabilmeyi,
- Formları kullanarak kullanıcı etkileşimli uygulamalar yapabilmeyi,
- Etiket yardımcılarını,
- Doğrulama işlemlerini gerçekleştirmeyi,
- Paket yöneticisini kullanabilmeyi,
- Entity Framework Core ile veri tabanı işlemlerini gerçekleştirebilmeyi,
- Web servis oluşturabilmeyi,
- ASP.NET Core uygulamasını yayınlayabilmeyi öğreneceksiniz.

ANAHTAR KELİMELER

Ara katman, ASP.NET Core, doğrulama, Entity Framework Core, etiket yardımcıları, formlar, MVC, paket yöneticisi, Razor sayfaları, uygulama dağıtma, uygulama yayınlama, web servis, yönlendirme, .NET, .NET Core.

Hazırlık Çalışmaları

1. MVC (Model View Controller) dışındaki diğer tasarım mimarilerinden bildiklerinizi ve bunların MVC tasarım deseni ile farklarının neler olduğunu arkadaşlarınızla paylaşınız.

2. Kullanıcı etkileşimli web sayfaları hangi amaçla kullanılıyor olabilir? Açıklayınız.

6.1. .NET Core Teknolojisi

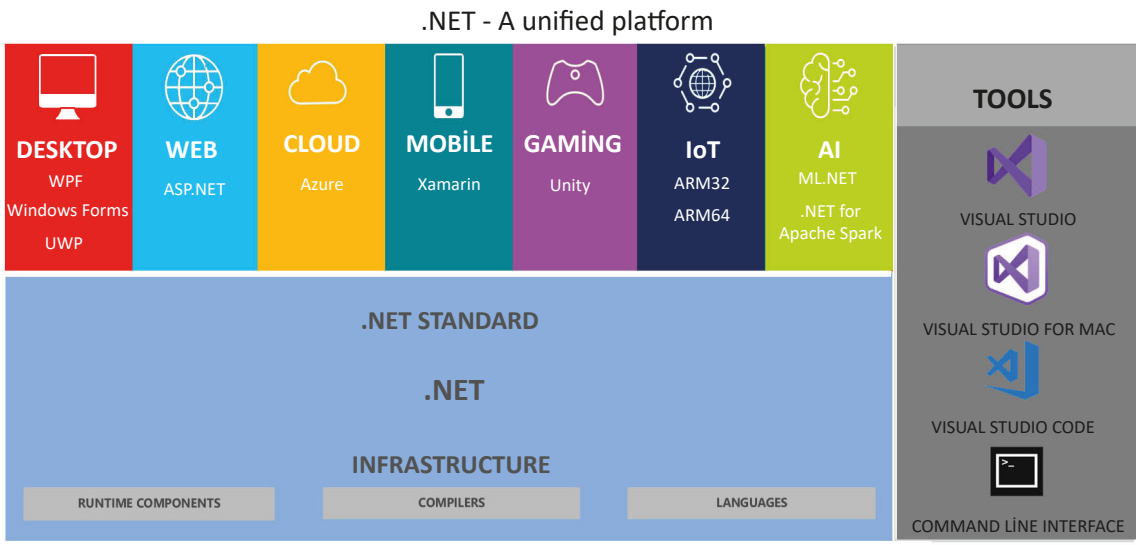
.NET, platformdan bağımsız geliştirme yapılabilen (cross platform), yüksek performans sağlayan ve açık kaynak kodlu bir yazılım geliştirme ortamıdır (Görsel 6.1). Önceden .NET Framework ile geliştirilen uygulamalar, sadece belirli bir işletim sistemi üzerinde çalışabiliyorken artık .NET ile çeşitli işletim sistemlerinde çalıştırılabilmektedir.

Görsel 6.1: .NET

Not

.NET kaynak kodlarına https://github.com/dotnet/core/blob/master/Documentation/core-repos.md adresinden erişilebilir.

Görsel 6.2'de de görüldüğü gibi .NET ile yapılabilecekler şunlardır:



Görsel 6.2: .NET platformu

- Web uygulamaları, Web API'leri ve mikro hizmetler,
- Bulutta sunucusuz işlemler,
- Bulutta yerel uygulamalar,
- Mobil uygulamalar,
- Masaüstü uygulamaları,
- Windows Presentation Foundation (WPF) uygulamaları,
- Evrensel Windows Platformu (UWP) uygulamaları,
- Oyun,
- Nesnelerin İnterneti (IoT),
- Makine öğrenmesi,
- Konsol uygulamaları,
- Windows servisleri geliştirilebilir.

Desteklenen İşletim Sistemleri: .NET ile çok çeşitli işletim sistemlerine yönelik uygulamalar geliştirilebilir. Bu işletim sistemlerinden bazıları şunlardır:

- Windows
- macOS
- Linux
- Android
- iOS
- tvOS
- watchOSÜ

Desteklenen İşlemci Mimarileri: .NET'in desteklediği işlemci mimarileri şunlardır:

- x64
- x86
- ARM32
- ARM64

Desteklenen Programlama Dilleri: .NET şu programlama dillerini destekler:

- C#
- Visual Basic
- F#

Tarihçesi: .NET, şu anda kâr amacı gütmeyen bir açık kaynak kuruluşu olan **.NET Foundation** altında yönetilmektedir. .NET, C# ve C++ ile yazılmıştır ve MIT lisansı altında lisanslanmıştır. İlk sürüm olan .NET Core 1.0'ın, sınırlı işlevsellikle 2016 yılında yayınlanmasından bu yana .NET Core oldukça fazla gelişim göstermiş ve son olarak 5. sürümü kullanıma sunulmuştur. 5. sürüm ile birlikte adından **Core** çıkartılmış ve **.NET Framework** ile sürüm çakışmasını engellemek amacıyla sürüm adı **.NET 5** olmuştur.

Tablo 6.1'de şu an desteklenen .NET sürümleri listelenmiştir.

Tablo 6.1: .NET Sürüm Destekleri

Sürüm	İlk Yayınlanma Tarihi	Son Destek Tarihi
.NET 5	10 Kasım 2020	Tahmini olarak .NET 6 sürümü yayınlandıktan 3 ay sonra
.NET Core 3.1	3 Aralık 2019	3 Aralık 2022

Daha önceki sürümler şu an desteklenmemektedir.

Not

Bu öğrenme biriminin yazımı aşamasında mevcut olan sürümler listelenmiştir. Son güncel sürümler, https://dotnet.microsoft.com/download/dotnet-core adresinden takip edilebilir.

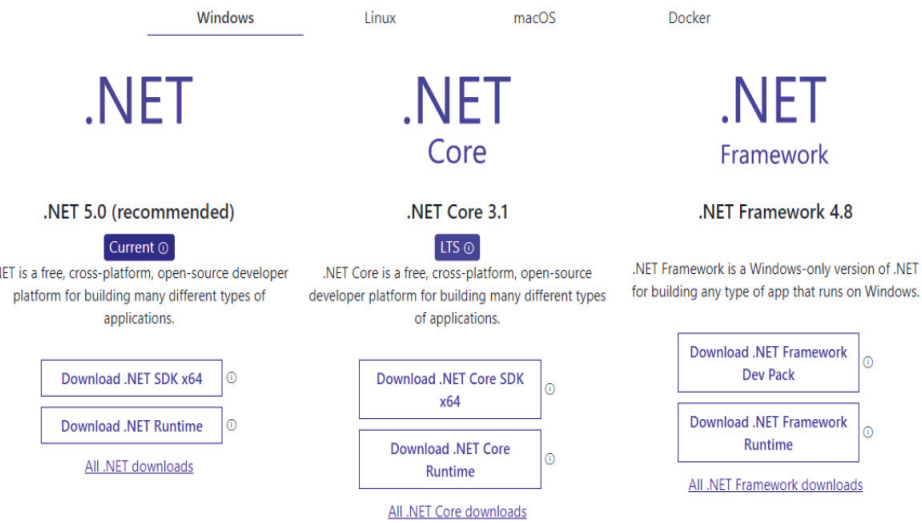
6.1.1. .NET Kurulumu

.NET'te uygulama geliştirme ve yazılan uygulamaları çalıştırabilmek için iki ayrı indirme seçeneği bulunur.

.NET SDK (Software Development Kit): Uygulama geliştirme ve uygulamaları çalıştırmak için gerekli bileşenleri içerir.

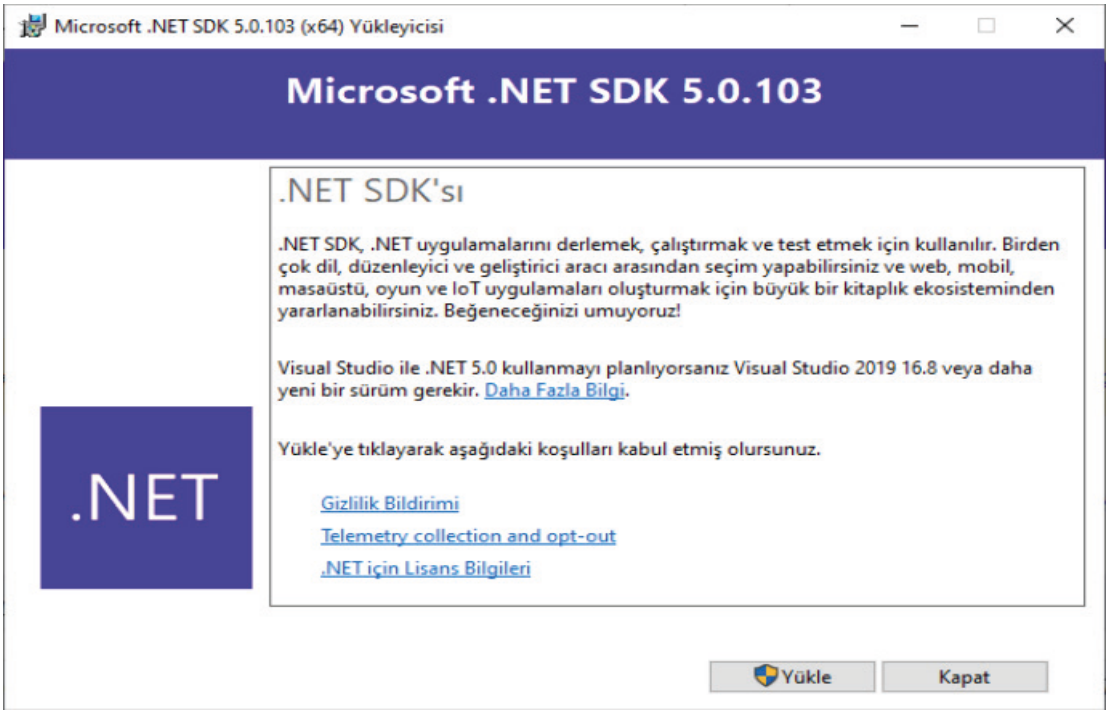
.NET Runtime: Uygulamaları çalıştırmak için gerekli bileşenleri içerir.

https://dotnet.microsoft.com/download adresinde, üzerinde çalışılmak istenen işletim sistemine göre güncel en son sürümler görüntülenebilir ve indirilebilir (Görsel 6.3).



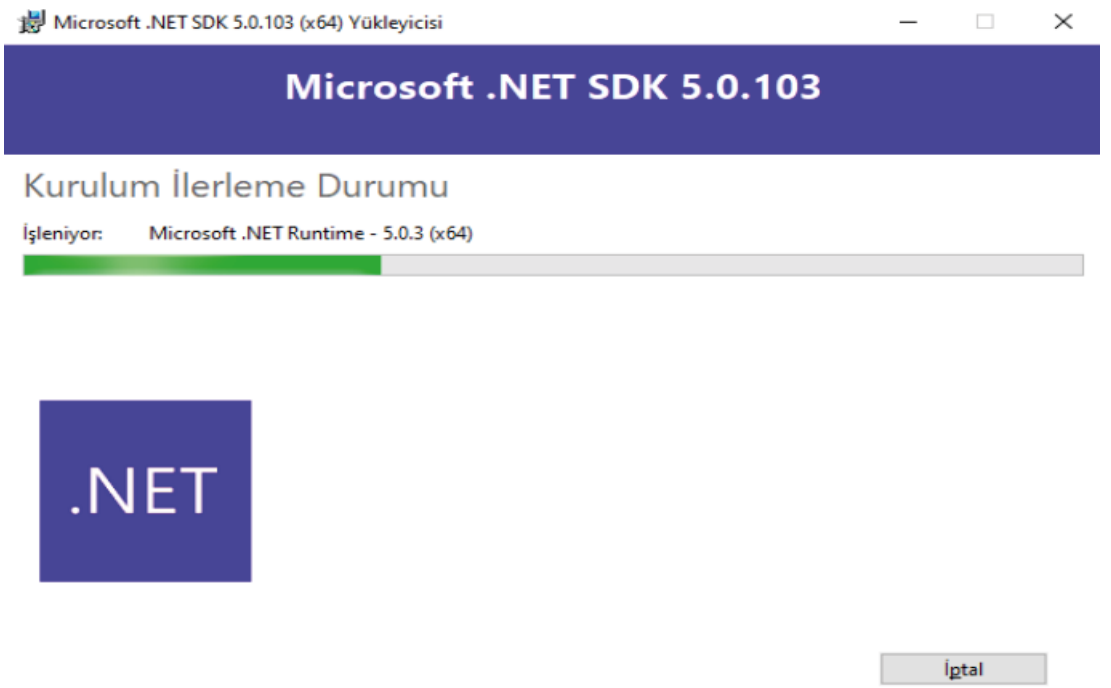
Görsel 6.3: .NET indirme seçenekleri

Download .NET SDK x64 düğmesine tıklandıktan sonra indirme işlemi başlatılır.



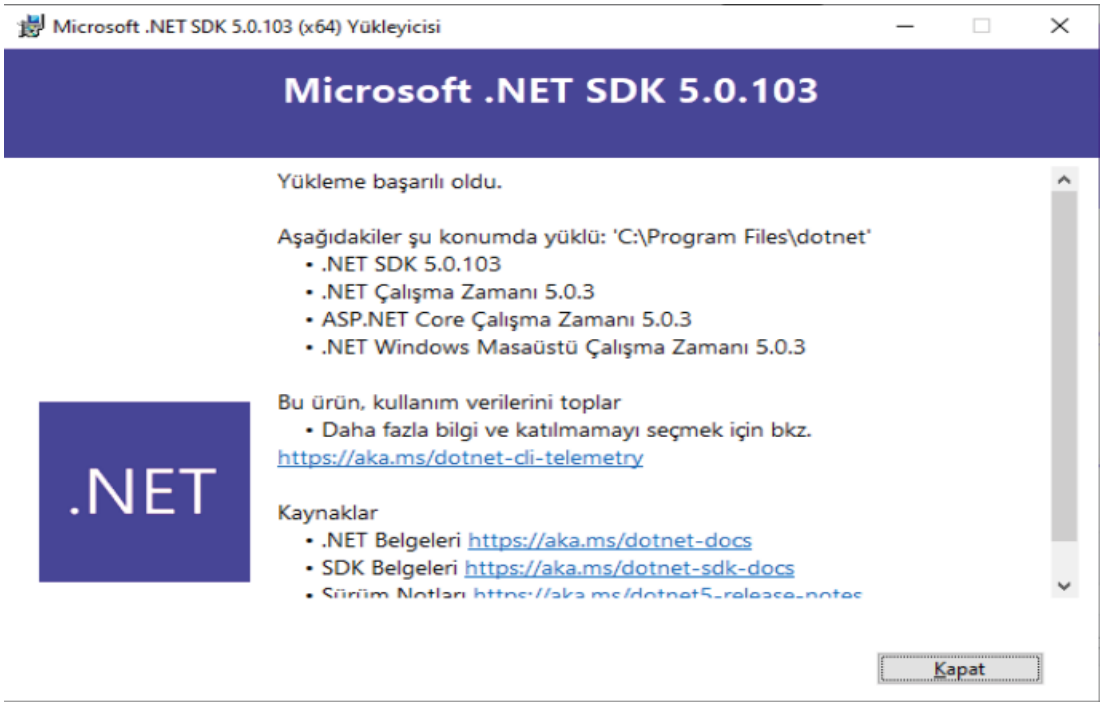
Görsel 6.4: Kuruluma başlama

İndirme işlemi tamamlandıktan sonra dosya çalıştırılır ve kurulum başlatılır (Görsel 6.4).



Görsel 6.5: Kurulum aşaması

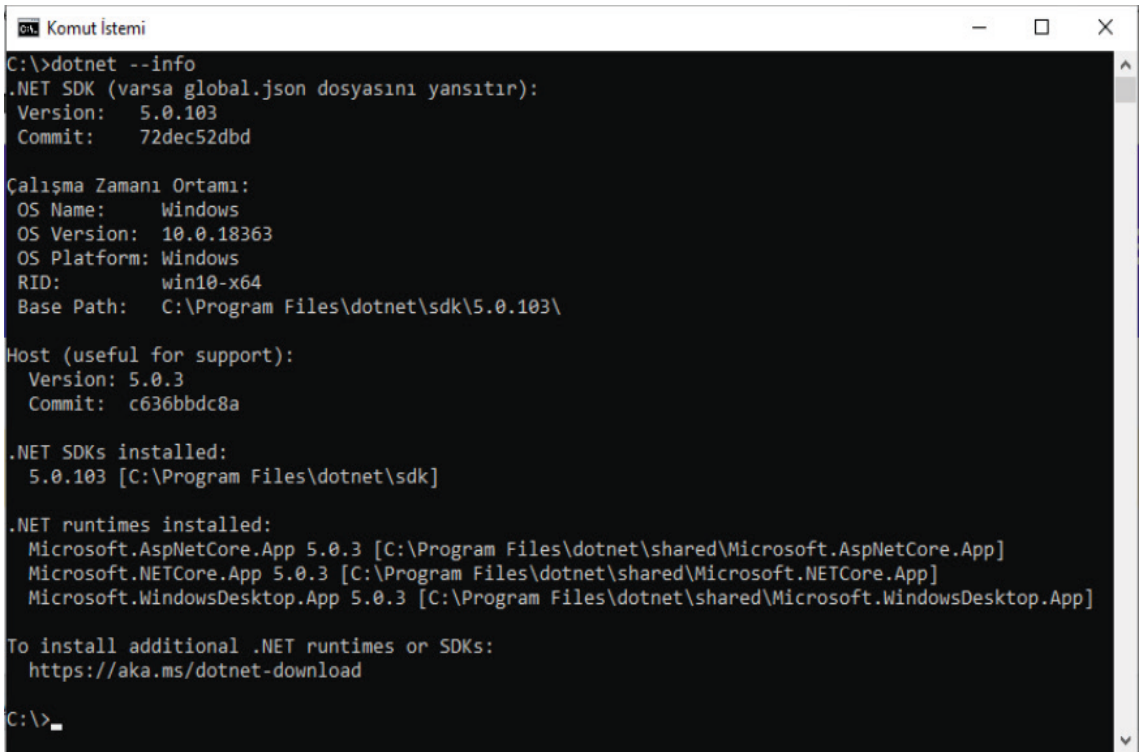
Kurulumun tamamlanması beklenir (Görsel 6.5).



Görsel 6.6: Kurulumun tamamlanması

Kurulum işlemi başarıyla tamamlandıktan sonra Görsel 6.6'daki pencere gösterilir.

Kurulumu test etmek ve önceden kurulmuş tüm .NET sürümlerini görüntülemek için komut satırından "dotnet --info" komutu çalıştırılabilir (Görsel 6.7).



Görsel 6.7: Mevcut .NET sürümlerinin görüntülenmesi

6.1.2. ASP.NET Core ve Diğer Teknolojilerden Farkı

Günümüzde programcıların hizmetine sunulmuş pek çok arka uç (backend) yazılım geliştirme platformu mevcuttur.

Görsel 6.8: ASP.NET Core

ASP.NET Core (Görsel 6.8), .NET üzerine inşa edilmiş açık kaynak kodlu bir web uygulama geliştirme teknolojisidir. Yazılımcılara pek çok kolaylık ve avantaj sağlar. Bunlar;

Çapraz platform desteği sunması,

Açık kaynak kodlu olması,

Topluluk odaklı olması,

Gelişmiş programlama özellikleri barındırması;

- Otomatik hafıza yönetimi,
- Paket yönetimi,
- Çöp toplayıcısı (Garbage collector),
- Birden fazla programlama dili desteği,
- Asenkron programlama,
- MVC ve Razor sayfaları desteği,
- Tam API desteği,
- Blazor ile tarayıcıda C# kodlarının çalıştırılması,

Hızlı, ölçeklenebilir ve yüksek performans,

Birden fazla .NET sürümünün yan yana kullanılabilmesi,

HTTP/1.1, HTTP/2 ve HTTP/3 desteği sunması,

Yazılımcıların birlikte çalışmasını kolaylaştırması,

Modern web geliştirmeyi basitleştiren araçlar sunması olarak özetlenebilir.

Yukarıdaki özellikler göz önünde bulundurulduğunda ASP.NET Core'un, arka uç yazılımcıları için oldukça işlevsel bir platform sunduğu anlaşılmaktadır.

6.2. MVC Tasarım Deseni

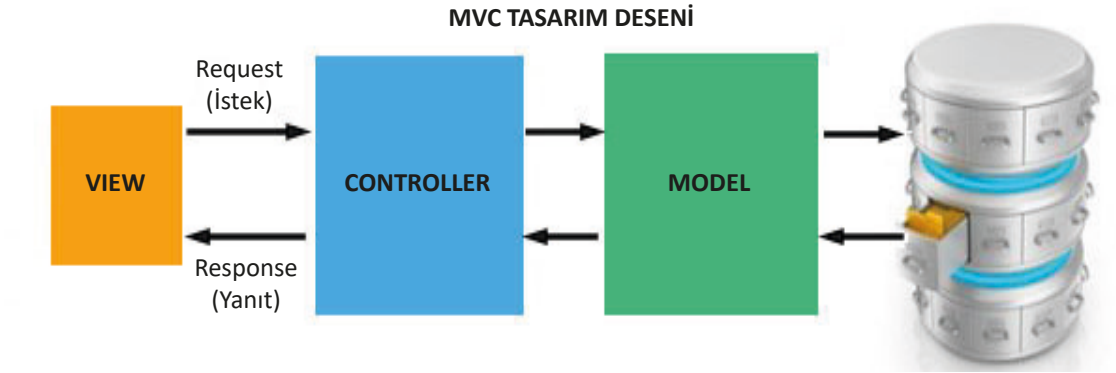
Tasarım deseni, yazılımcıların karşılaştığı çeşitli sorunlar karşısında zaman içerisinde üretilen çözümleri ele alan standart çözüm yöntemleridir. Tasarım desenleri; iyi bir tasarımın yapılması, yazılım esnekliğinin ve okunabilirliğinin arttırılması, sık karşılaşılan sorunlara çözüm sunması için deneyimlerin derlenmesiyle oluşturulmuştur.

Geliştirilen uygulamalar genellikle verilere erişim için gerekli kodların, uygulama mantığını oluşturan kod parçalarının ve kullanıcı arayüzlerinin birleşiminden oluşur ve MVC (Model-View-Controller) tasarım deseni, uygulama geliştirme aşamasını bu nedenden dolayı üç parçaya ayırır.

View (Görünüm): Geliştirilen projede kullanıcının gördüğü arayüzlerin tasarlandığı bölümdür. Bu bölümde HTML, Javascript ve CSS kullanılır.

Controller (Kontrolör): Projedeki tüm hesaplamaların, veri tabanına erişimin ve diğer işlemlerin yapıldığı bölümdür. Uygulamadaki model ve view bölümleri arasındaki koordinasyonu sağlar.

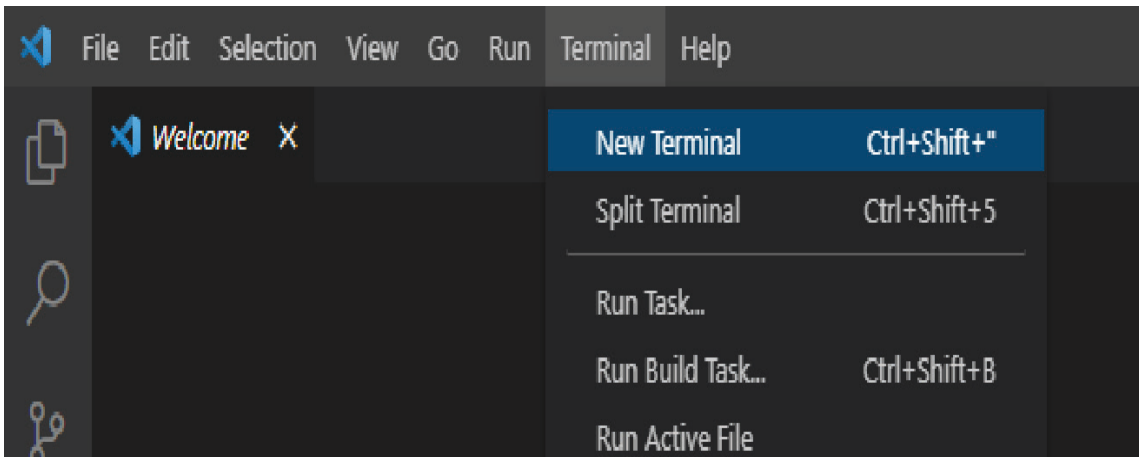
Model: Geliştirilen uygulamada kullanılacak verilerin özelliklerinin tutulduğu bölümdür. Örnek olarak kütüphane uygulamasında kitap bilgilerinin (adı, sayfa sayısı, vb.) özellikleri bu bölümde tutulur (Görsel 6.9).



Görsel 6.9: View, Controller ve Model arasındaki ilişki

6.2.1. MVC Projesi Oluşturma

Visual Studio Code editöründe MVC projesi oluşturmak için öncelikle yeni bir terminal penceresi açılır (Görsel 6.10).



Görsel 6.10: Terminal penceresi açma

Terminal penceresinde projenin bulunacağı klasöre gidilir (Klasör yoksa oluşturulur.).

Not

Bu aşamada klasör değiştirmek için **cd** ve klasör oluşturmak için **mkdir** komutları kullanılır.

MVC projesi oluşturmak için **"dotnet new MVC"** komutu yazılır ve gerekli indirme işlemlerinin tamamlanması beklenir. Ardından **"code ."** komutu ile oluşturulan yeni proje Visual Code editöründe açılır (Görsel 6.11).

Eğer gerekli bileşenlerin olmadığına dair bir hata mesajı alınırsa **Evet** seçeneği seçilerek kurulum onayı verilmelidir (Örnek: C# Extensions).

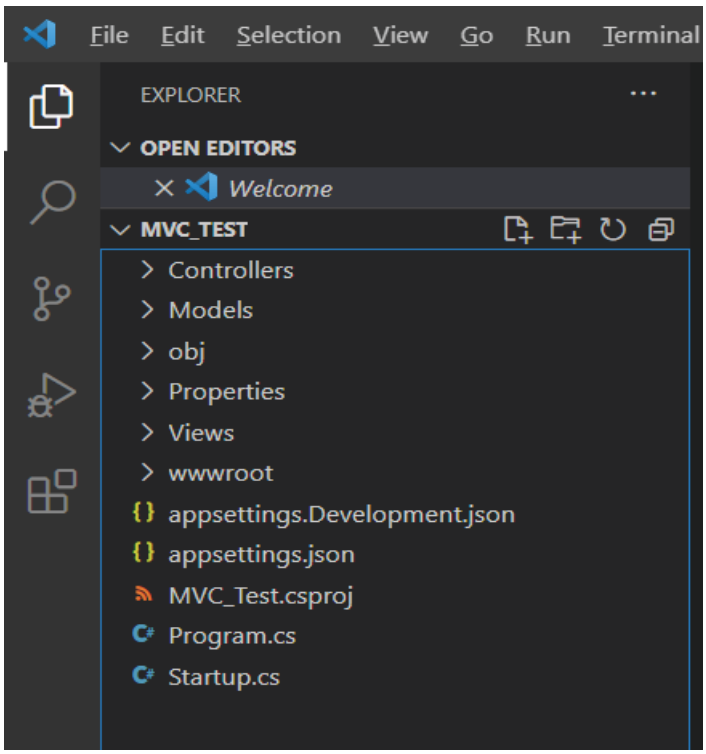
Not

İlerleyen aşamalarda kullanılmak üzere "Nuget Package Manager"in de bu aşamada yüklenmesi önerilmektedir (Nuget Paket Yöneticisi https://marketplace.visualstudio.com/vscode adresinden indirilebilir).

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

PS E:\> mkdir MVC_Test


    Directory: E:\


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        14/05/2021     12:41                MVC_Test


PS E:\> cd MVC_Test
PS E:\MVC_Test> dotnet new MVC
Getting ready...
The template "ASP.NET Core Web App (Model-View-Controller)" was created successfully.
This template contains technologies from parties other than Microsoft, see https://aka.ms/aspnetcore/5.0-third-party-notices for details.

Processing post-creation actions...
Running 'dotnet restore' on E:\MVC_Test\MVC_Test.csproj...
  Geri yüklenecek projeler belirleniyor...
  E:\MVC_Test\MVC_Test.csproj geri yüklendi (70 ms içinde).
Restore succeeded.

PS E:\MVC_Test> code .
```

Görsel 6.11: MVC Projesi oluşturma

Proje başarılı bir şekilde oluşturulduğunda Visual Code editöründe Görsel 6.12'deki klasör / dosya yapısı görüntülenir.

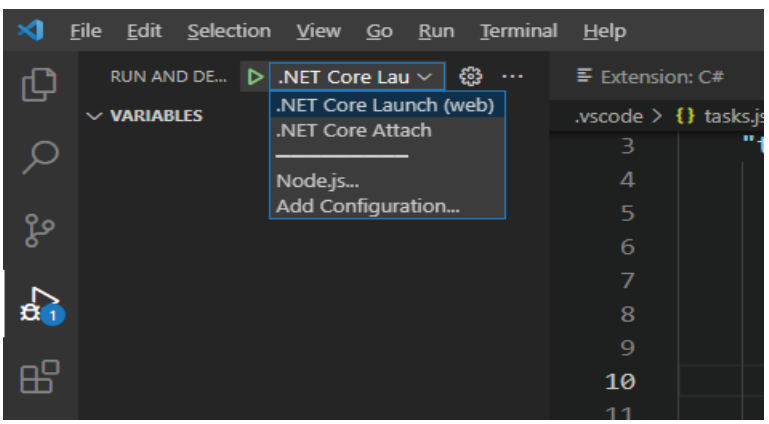


Görsel 6.12: MVC Projesi klasör / dosya yapısı

Not

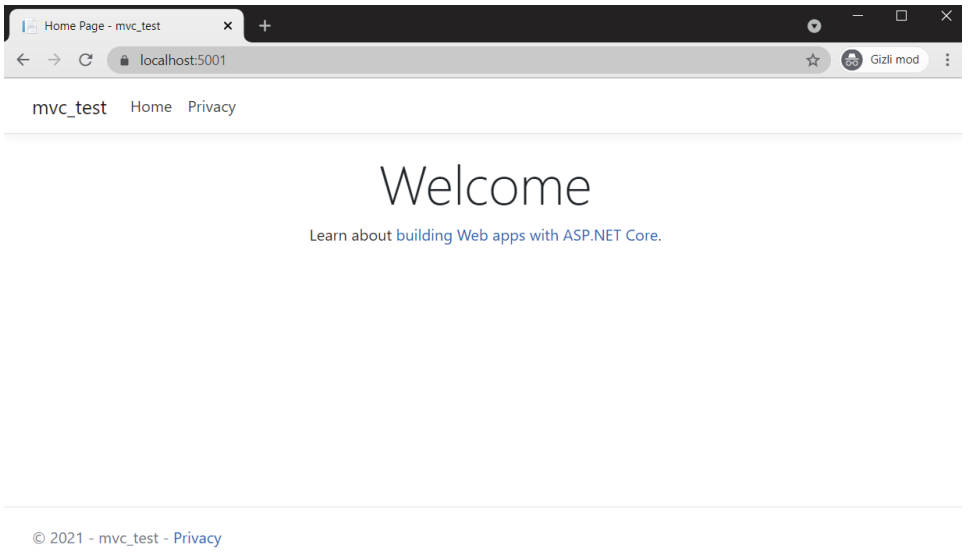
Yeni bir MVC projesi açıldığında **C# Extensions** eklentisi "Required assets to build and debug are missing from 'mvc_test'. Add them?" mesajını gösterir. **YES** butonuna basıldığında **debug** işlemleri için gerekli ayar dosyaları otomatik oluşturulur.

Tüm gerekli eklentiler ve ayar dosyaları (.vscode klasörü altındaki json uzantılı dosyalar) oluşturulduktan sonra **Run and Debug** sekmesinde **.NET Core Launch (web)** seçili iken yeşil renkli çalıştır düğmesi ile proje çalıştırılır (Görsel 6.13).



Görsel 6.13: Projeyi başlatma

Bu aşamada tarayıcı penceresi açılır ve web sitesi görüntülenir (Görsel 6.14).



Görsel 6.14: Çalışan web uygulaması

Not

Çalışan projeyi durdurmak için **Run** menüsünden **Stop Debugging** seçeneği seçilir (Shift + F5 kısayol tuş kombinasyonu da kullanılabilir.).

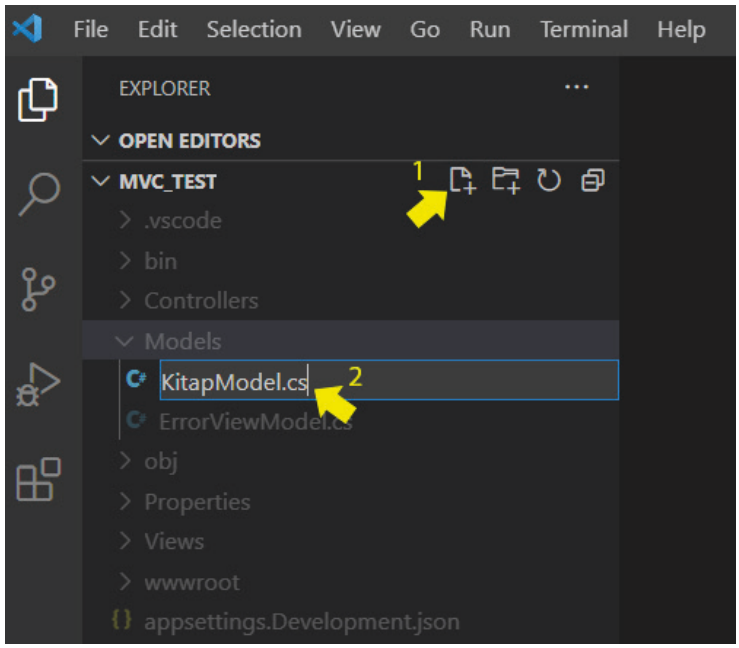
Sıra Sizde

Farklı klasörler altında yeni proje oluşturma ve çalıştırma işlemini birkaç kez tekrarlayınız.

6.2.2. Model Katmanı

MVC tasarım deseninde **model katmanı,** geliştirilen uygulamada kullanılacak olan verilerin özelliklerinin tanımlandığı katmandır. Veri özelliklerini **(Properties)** tanımlarken her bir veri için ayrı ayrı **sınıf (class)** tanımlaması yapılmalıdır. Model sınıfları projenin ana klasörü altındaki **Models** alt klasörü altında yer almalıdır.

Yeni bir model tanımlamak için **Models** klasörü seçili iken **New File** butonuna tıklanır ve modele bir isim verilir (Görsel 6.15). Model dosya uzantısının **.cs** olması zorunludur.



Görsel 6.15: Yeni model oluşturma

Oluşturulan **KitapModel.cs** dosyasına **KitapModel** sınıf tanımlaması yazılır.

```
public class KitapModel
{
   public int KitapId { get; set; }
   public string KitapAdi { get; set; }
   public int SayfaSayisi { get; set; }
}
```

Sıra Sizde

Kitapların kategorilere göre kayıtlarının tutulması amacıyla bir Kategori modeli oluşturunuz ve şu özellikleri ekleyiniz:

- Kategori ID,
- Kategori Adı
- Kitap listesi (List<KitapModel> kullanmalısınız. Gerekli isim uzayını eklemelisiniz.)

6.2.3. Controller Katmanı

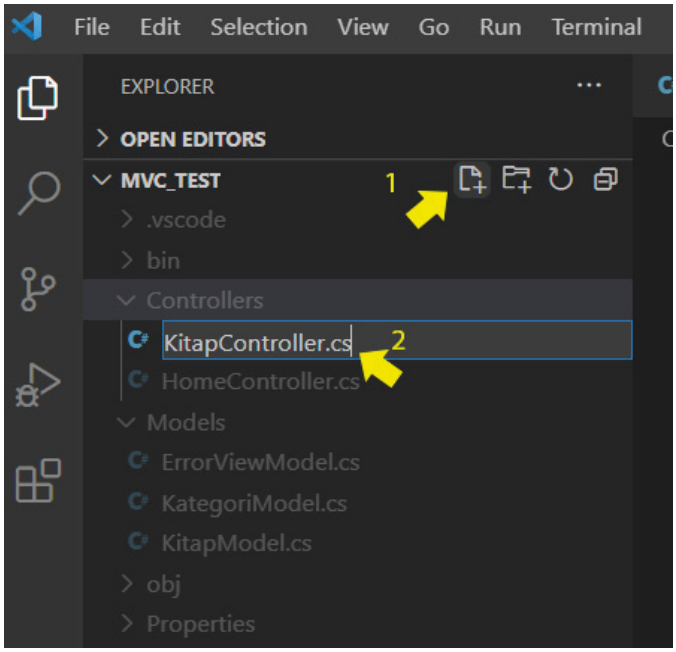
MVC tasarım deseninde **controller katmanı;** içinde barındırdığı **action** metotları ile birlikte tarayıcıdan gelen sayfa isteklerini (URL) yöneten, view ve model katmanları arasındaki bağlantıyı sağlayan ve projenin iş sürecini kontrol eden katmandır.

Her bir controller, **Microsoft.AspNetCore.Mvc.Controller** sınıfından türetilen bir C# sınıfıdır ve içerisinde **action** metotlarını barındırır. Sınıf isimlendirmesinde dikkat edilmesi gereken husus; sınıf adının "Controller" kelimesi ile bitmesi gerektiğidir. Örneğin; Anasayfa controller sınıfının adı **HomeController**, Kitap controller sınıfının adı **KitapController** olmalıdır. Önemli bir diğer husus da controller sınıflarının, projenin ana dizininde bulunan **Controllers** alt klasörü altında yer alması gerektiğidir.

Not

Tarayıcıdan gelen **https://localhost:5001/Kitap/xxx** şeklindeki bir istek Kitap controller tarafından yönetilecek anlamına gelir. Sondaki xxx controller içerisindeki bu isimdeki action metodu tarafından kontrol edilir.

Yeni bir controller tanımlamak için **Controllers** klasörü seçiliyken **New File** butonuna tıklanır ve dosyaya "..Controller.cs" ile bitecek şekilde bir isim verilir (Görsel 6.16).



Görsel 6.16: Yeni controller oluşturulması

Kitap controller sınıfı şu şekilde tanımlanabilir:

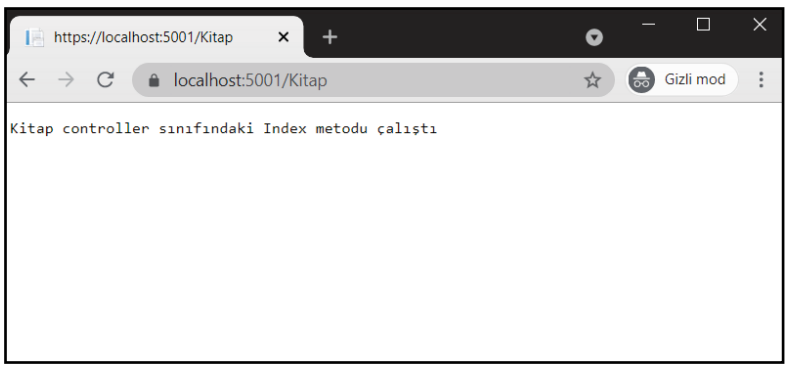
```
using Microsoft.AspNetCore.Mvc;

public class KitapController: Controller
{
  public ActionResult Index()
     {
        return View();
     }
   // Diğer action metotlar tanımlanır
}
```

Action metotlar genellikle view döndürmek için kullanılır. Yukarıdaki örnekte **Views** klasörü altındaki **Kitap** alt klasöründe yer alan **Index.cshtml** adlı dosya döndürülmektedir (Views> Kitap> Index.cshtml).

Controller sınıfı tanımlandıktan sonra, proje ihtiyaçlarına uygun action metotlar oluşturulmalıdır. **Index** adlı action metot, **https://localhost:5001/Kitap/** gibi sonda action belirtilmediği durumlarda çalışan metottur. Aynı şekilde **https://localhost:5001/Kitap/Index** şeklinde de çağrılabilir. Bu action metot şu şekilde test edilebilir:

```
using Microsoft.AspNetCore.Mvc;

public class KitapController : Controller
{
   public string Index()
   {
      return "Kitap controller sınıfındaki Index metodu çalıştı";
   }
}
```

Proje çalıştırılıp **https://localhost:5001/Kitap/** adresi çağrıldığında ekran görüntüsü Görsel 6.17'deki gibi olacaktır.



Görsel 6.17: Kitap controller sınıfı

Sıra Sizde

Projeye **Yazar** controller sınıfı ekleyiniz.

6.2.3.1. Action Metotlar

Temel olarak controller sınıfında **public** olarak tanımlanan tüm metotlar **action metot** olarak isimlendirilir. Her bir controller sınıfı için **Index()** adlı metot varsayılan action metodudur. Yani action belirtilmediğinde çalışacak metot, **Index() metodudur.**

Her bir action metodu;

- View (View katmanındaki ilgili dosya),
- Dosya (Resim, video, vb.),

- JSON (JavaScript Object Notation)
- string ve int gibi C# veri tipleri, gibi çeşitli tiplerde değer (ActionResult) döndürebilir ya da bir başka controller / actiona yönlendirme yapabilir.

Örnek: JSON döndüren action metot

```
using Microsoft.AspNetCore.Mvc;

public class KitapController : Controller
{
   public JsonResult JsonCikti()
   {
      return Json(new
      {
         Id = 1,
         Seviye = "11. Sınıf",
         KitapAdi = "WEB TABANLI UYGULAMA GELİŞTİRME"
      });
   }
}
```

Proje çalıştırıldığında ekran görüntüsü Görsel 6.18'deki gibi olacaktır.

Görsel 6.18: JSON çıktısı

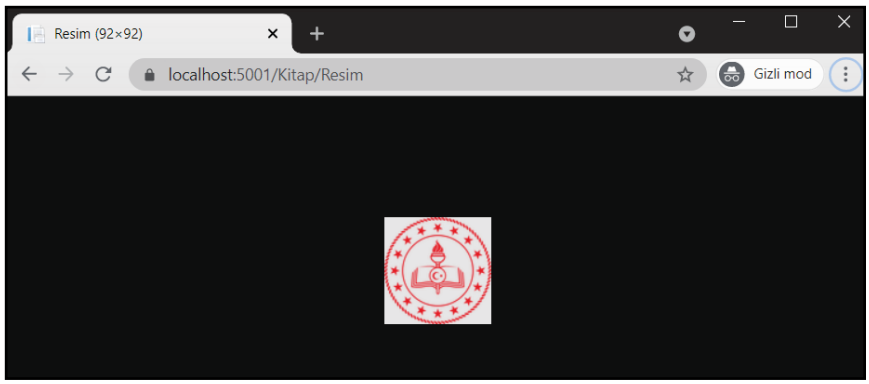
Örnek: Resim dosyası döndüren action metot

```
using Microsoft.AspNetCore.Mvc;

public class KitapController : Controller
{
   public FileResult Resim()
   {
      return File("~/meblogo.png", "image/png");
   }
}
```

Not

Resim dosyası projenin ana klasöründeki **wwwroot** alt klasörü altında yer almaktadır.

Proje çalıştırıldığında ekran görüntüsü Görsel 6.19'daki gibi olacaktır.



Görsel 6.19: Dosya çıktısı

Action metotlarda parametre kullanımı da mümkündür. **https://localhost:5001/Kitap/Liste/1** şeklinde bir adres çağrıldığında sondaki parametreyi kullanmak için şu şekilde bir action metot yazılmalıdır:

```
using Microsoft.AspNetCore.Mvc;

public class KitapController : Controller
{
   public string Liste(int id)
   {
      return $"Parametreden gelen değer: {id}";
   }
}
```

Proje çalıştırıldığında ekran görüntüsü Görsel 6.20'deki gibi olacaktır.

Görsel 6.20: Parametre kullanımı

Birden fazla parametre kullanımı da mümkündür.

Örnek: https://localhost:5001/Kitap/Liste/1/MEB şeklinde hem int hem de string tipli parametre alan bir action metot şu şekilde yazılabilir:

```
using Microsoft.AspNetCore.Mvc;

public class KitapController : Controller
{
   [Route("Kitap/Liste/{id}/{ad}")]
   public string Liste(int id, string ad)
   {
      return $"Parametreden gelen id={id}, ad={ad}";
   }
}
```

Proje çalıştırıldığında ekran görüntüsü Görsel 6.21'deki gibi olacaktır.

Görsel 6.21: Çoklu parametre kullanımı

Sıra Sizde

Oluşturduğunuz Yazar controller sınıfına YazarListesi, YazarinKitaplari action metotlarını ekleyiniz.

6.2.4. View Katmanı

MVC tasarım deseninde view katmanı, kullanıcının göreceği tüm arayüzlerin oluşturulduğu katmandır. Bu katmanda yer alan dosyalar **.cshtml** uzantılı olmalıdır.

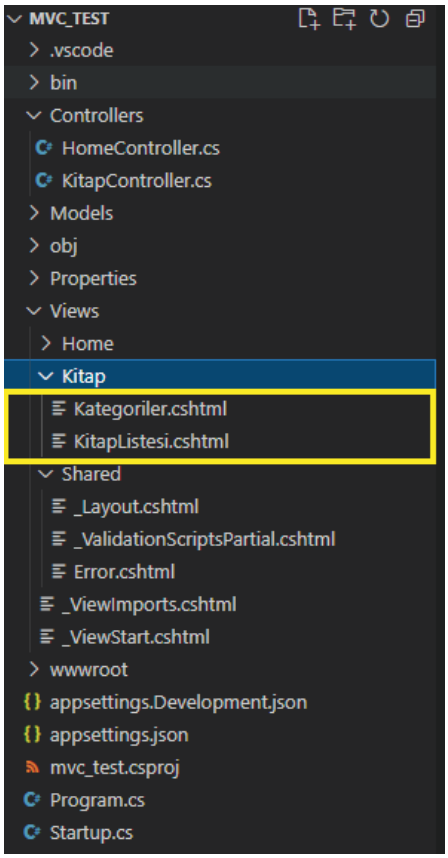
Bir önceki bölümde işlendiği üzere, bir controller bir veya birden fazla action metot bulundurabilir. Bir başka deyişle bir controller birden fazla view döndürebilir. MVC tasarım deseninde viewlar, **Views** klasörü altında **Controller** adıyla aynı adlı bir alt klasör içerisinde yer almalıdır.

Örneğin, HomeController sınıfı altındaki action metotları Views > Home klasörü altındaki View dosyalarını, KitapController sınıfı altındaki action metotları Views > Kitap klasörü altındaki View dosyalarını çağırır.

Yeni bir view eklemek için **Views** klasörü altına, bu view dosyasıyla ilişkili controller sınıfı adıyla bir klasör oluşturulur. Oluşturulan klasör seçili iken **New File** butonuna tıklanır ve view dosyasına ilişkili action metodunun ismi verilir (Görsel 6.15). View dosya uzantısının **.cshtml** olması zorunludur.

Kitap controller sınıfı aşağıdaki şekilde düzenlendiğinde, view dosyaları Görsel 6.22'deki gibi eklenmelidir.

```
using Microsoft.AspNetCore.Mvc;

public class KitapController : Controller
{
    public IActionResult Kategoriler()
    {
        return View();
    }

    public IActionResult KitapListesi(int id)
    {
        return View();
    }
}
```



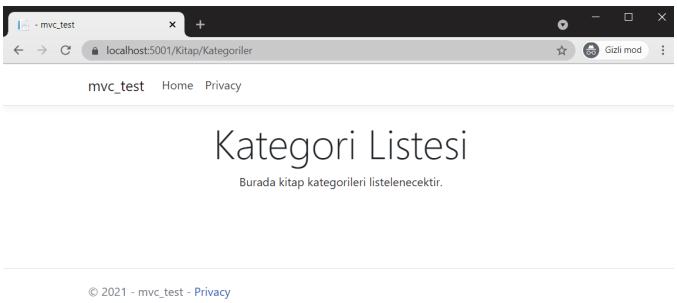
Görsel 6.22: View dosyaları

Not

Shared klasörü, birden fazla View dosyasında ortak olarak kullanılan dosyaları barındırır.

Şimdi **Kategoriler.cshtml** dosyasının içeriği düzenlenebilir ve tarayıcıda görüntülenebilir (Görsel 6.23).

```
<div class="text-center">
   <h1 class="display-4">Kategori Listesi</h1>
   Burada kitap kategorileri listelenecektir.
</div>
```



Görsel 6.23: Kategori listesi sayfası

Not

Dikkat edilirse standart bir HTML dosyasında yer alan HTML, head, body gibi etiketlerin kullanılmadığı görülmektedir. Bu etiketler, Shared > _Layout.cshtml dosyasında bulunmaktadır ve Kategoriler.cshtml dosyasının bu layoutu kullanacağı ise Shared > _ViewStart.cshtml dosyasında tanımlanmaktadır. View dosyasının üst tarafında layout kullanılıp kullanılmayacağı veya hangi layout dosyasının kullanılacağı tanımlanabilir.

```
@{
   Layout = null; //  ---> Layout kullanılmayacak
   // Layout = "_Layout";   ---> Shared>_Layout.cshtml kullanılacak
}
<div class="text-center">
   <h1 class="display-4">Kategori Listesi</h1>
   Burada kitap kategorileri listelenecektir.
</div>
```

Sıra Sizde

Yazarlar controllerına eklediğiniz action metotların viewlarını oluşturunuz.

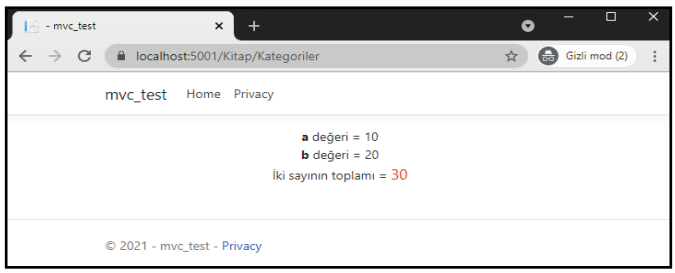
6.2.4.1. Razor View Motoru

Razor View Motoru, view dosyaları içerisinde sunucu taraflı kodlar ile HTML kodlarının bir arada kullanımını sağlayan bir yapıdır. Razor view, **@** karakteri ile başlayan ifadeleri sunucu taraflı çalıştırır ve HTML çıktısı oluşturur.

İki sayının toplamını HTML kodları ile beraber ekrana yazan bir view şu şekilde oluşturulabilir:

```
<div class="text-center">
   @{
      var a = 10;
      var b = 20;
   }
   <b>a</b> değeri = @a
   <br />
   <b>b</b> değeri = @b
   <br />
   İki sayının toplamı <span style="color:red;font-size:larger;">@(a+b)</span>
</div>
```

View görüntülendiğinde ekran çıktısı Görsel 6.24'teki gibi olacaktır.



Görsel 6.24: Razor view ekran çıktısı

6.2.4.2. Action Metottan View'e Veri Aktarımı

Bir action metottan view'e birkaç yöntemle veri aktarılabilir.

ViewBag Kullanarak: Controller içerisinde ViewBag ile üzerinden veriler dinamik (dynamic) olarak viewa aktarılabilir:

```
public IActionResult Kategoriler()
{
   var kitap = new KitapModel
   {
      KitapId = 1,
      KitapAdi = "WEB TABANLI UYGULAMA GELİŞTİRME"
   };
   ViewBag.Kitap = kitap;
   ViewBag.Sahip = "Millî Eğitim Bakanlığı";
   return View();
}
```

View tarafında kullanımı:

```
<div class="text-center">
   Kitap adı: <b>@ViewBag.Kitap.KitapAdi</b>
   <br/>
   Sahibi: <b>@ViewBag.Sahip</b>
</div>
```

ViewData Kullanarak: ViewBag'e benzer şekilde veriler viewa aktarılır. Ancak ViewData'da anahtar / değer ikilileri kullanılır:

```
public IActionResult Kategoriler()
{
   var kitap = new KitapModel
   {
      KitapId = 1,
      KitapAdi = "WEB TABANLI UYGULAMA GELİŞTİRME"
   };
   ViewData["Kitap"] = kitap;
   ViewData["Sahip"] = "Millî Eğitim Bakanlığı";
   return View();
}
```

View tarafında kullanımı:

```
@{
   var kitap = ViewData["Kitap"] as KitapModel;
}
<div class="text-center">
   Kitap adı: <b>@kitap.KitapAdi</b>
   <br />
   Sahibi: <b>@ViewData["Sahip"]</b>
</div>
```

TempData Kullanarak: Kullanımı ViewData ile aynıdır. ViewData'dan tek farkı başka bir action metoda yönlendirme durumunda verilerin kullanılabilir olmasıdır.

Ancak TempData kullanımında dikkat edilmesi gereken husus, transfer edilecek verinin serileştirilmesi gerektiğidir. Bunun için yeni bir terminal penceresi içerisinde **"dotnet add package Newtonsoft.Json"** komutu yazılarak serileştirme işlemleri için kullanılacak kütüphanenin yüklenmesi gereklidir. Kurulum tamamlandıktan sonra TempData ile transfer işlemi gerçekleştirilebilir.

```
public IActionResult Kategoriler()
{
   var kitap = new KitapModel
   {
      KitapId = 1,
      KitapAdi = "WEB TABANLI UYGULAMA GELİŞTİRME"
   };
   TempData["Kitap"]  = JsonConvert.SerializeObject(kitap);
   TempData["Sahip"] = "Millî Eğitim Bakanlığı";
   return RedirectToAction("EnCokOkunanlar");
}
public IActionResult EnCokOkunanlar()
{
   return View();
}
```

View tarafında kullanımı:

```
@{
    var kitap = Newtonsoft.Json.JsonConvert.DeserializeObject<KitapModel>(TempData["Kitap"].ToString());
}
<div class="text-center">
  Kitap adı: <b>@kitap.KitapAdi</b>
  <br />
  Sahibi: <b>@TempData["Sahip"]</b>
</div>
```

Model Kullanarak: Bir diğer veri aktarma seçeneği de **return View();** komutuna parametre vermektir. View tarafında gönderilen parametreye **model** nesnesi üzerinden erişilebilir.

```
public IActionResult Kategoriler()
{
  var kitap = new KitapModel
  {
    KitapId = 1,
    KitapAdi = "WEB TABANLI UYGULAMA GELİŞTİRME"
  };
  return View(kitap);
}
```

```
@model KitapModel

<div class="text-center">
  Kitap adı: <b>@Model.KitapAdi</b>
</div>
```

Sıra Sizde

Yazarlar controllerındaki action metotlardan viewlara yukarıda sayılan yöntemlerle veri gönderip viewlar içerisinde görüntülenmesini sağlayınız.

6.2.5. Razor Pages

ASP.NET Core 2.0 ile beraber duyurulan **Razor Pages**, MVC alt yapısını kullanarak sayfa bazlı uygulamaların çok daha kolay ve etkili bir şekilde geliştirilmesini sağlayan bir yapı sunar. MVC tasarım deseninin yerine geldiğini ya da ona alternatif olduğunu düşünmek doğru değildir. MVC'deki klasör sayısını azaltmayı ve sayfa bazlı uygulamaları daha kolay geliştirmeyi hedefler.

Razor Pages'te sayfalar birer **PageModel** ile tanımlanır. MVC'deki controller ve model katmanlarının bir arada olduğu bir yapı olarak düşünülebilir. Gelen sayfa istekleri controllerdaki gibi PageModel'ler tarafından karşılanır.

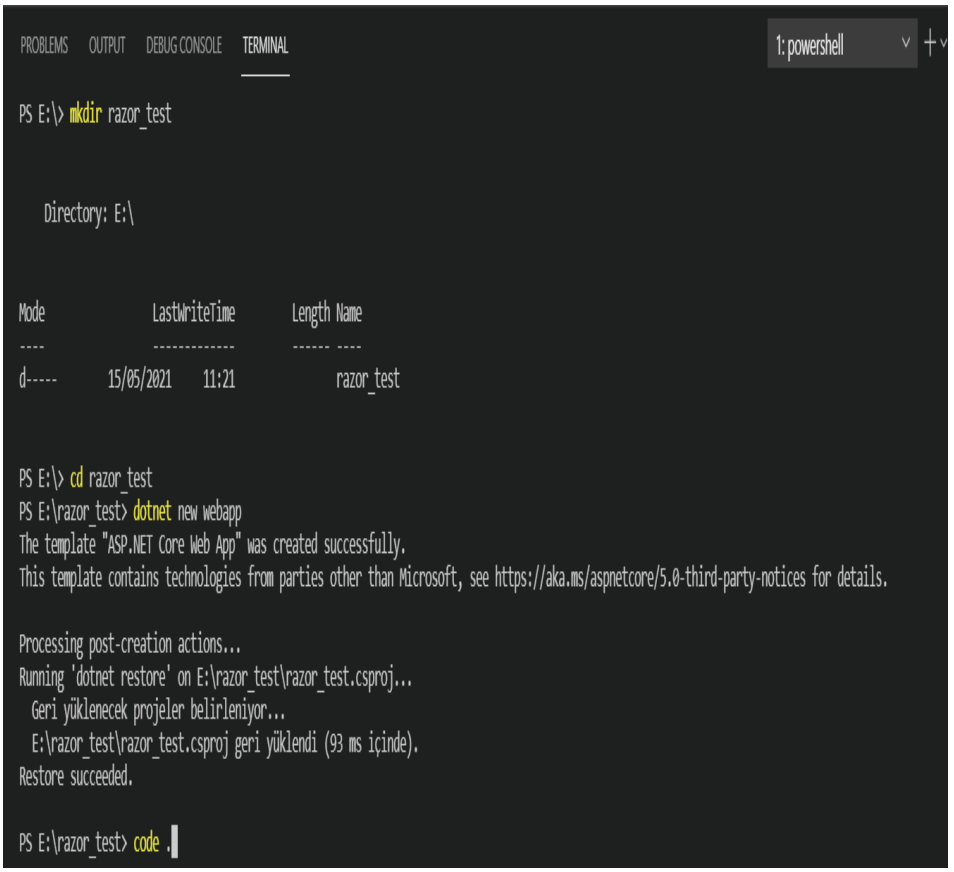
View dosyalarının, MVC'deki view dosyalarından farklı olarak ek bir özelliği bulunur. Bu da view dosyalarının ilk satırında **@page** direktifinin bulunmasıdır. Başında **@page** direktifi bulunan view dosyaları Razor Pages olarak kabul edilir.

6.2.5.1. Razor Pages Projesi Oluşturma

Visual Studio Code editöründe, Razor Pages projesi oluşturmak için yeni bir terminal penceresi açılır, projenin oluşturulacağı klasöre geçilir (klasör yoksa oluşturulur) ve **dotnet new webapp** komutu girilir. Proje oluşturma aşaması tamamlandıktan sonra **code .** komutu ile proje açılır (Görsel 6.25).

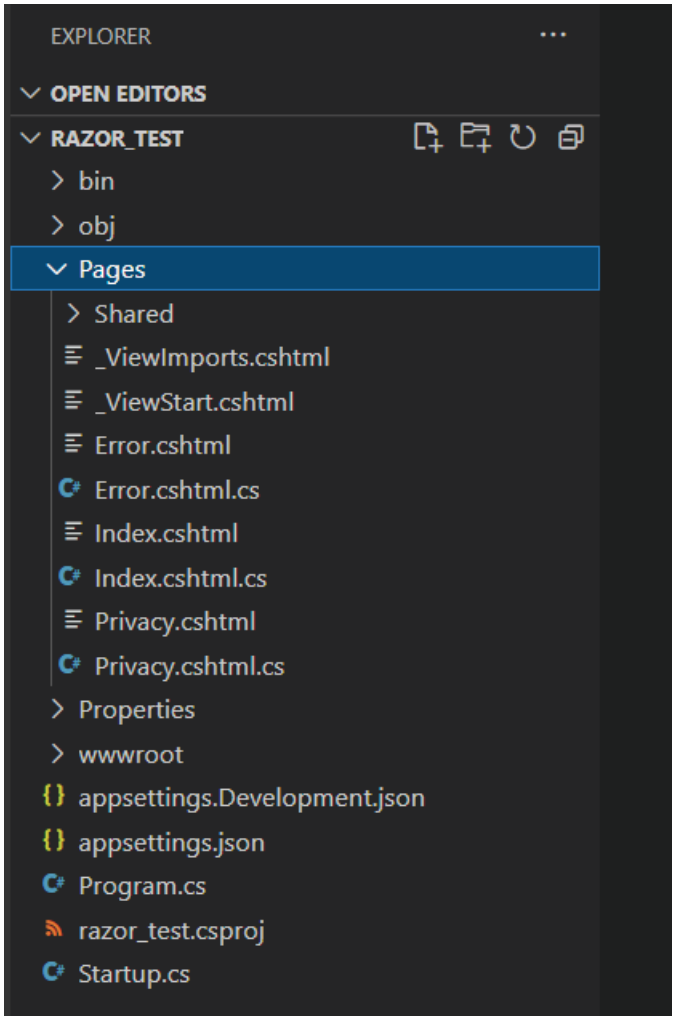
Not

Proje açıldıktan sonra editörün verdiği uyarılar dikkate alınarak gerekli eklentilerin yüklenmesi gerekir.



Görsel 6.25: Razor Pages projesi oluşturma

Klasör / dosya listesi incelendiğinde; MVC projelerindeki standart klasörlerin olmadığı, farklı olarak **Pages** klasörü olduğu görülür (Görsel 6.26). Razor Pages'te sayfalar bu klasörde tutulur. View dosyalarının uzantısı **.cshtml**, PageModel dosyalarının uzantısı ise **.cshtml.cs** şeklindedir ve hem view hem de PageModel dosyalarının adları aynıdır.



Görsel 6.26: Razor Pages proje yapısı

Örnek bir Razor Pages sayfası aşağıdaki gibidir.

Kitap.cshtml:

```
@page
@model KitapModel

<div class="text-center">
   Kitap adı= <b>@Model.KitapAdi</b>
</div>
```

Kitap.cshtml.cs:

```
using Microsoft.AspNetCore.Mvc.RazorPages;
using Microsoft.Extensions.Logging;

namespace razor_test.Pages
{
    public class KitapModel : PageModel
    {
        public string KitapAdi { get; set; }

        public KitapModel()
        {
        }

        public void OnGet()
        {
            KitapAdi = "WEB TABANLI UYGULAMA GELİŞTİRME";
        }
    }
}
```

6.2.5.2. Razor Pages ile MVC Karşılaştırılması

Razor Pages ile MVC aynı işleve sahiptir. Aralarında benzerlikler bulunduğu gibi bazı temel farklılıklar da bulunur.

Startup.cs Dosyası

Önceden oluşturulan örnek MVC ve Razor Pape projeleri karşılaştırıldığında **Startup.cs** dosyasında farklılıklar bulunduğu görülür. Bu farklılıklar Tablo 6.2'de gösterilmiştir.

Tablo 6.2: Startup.cs Dosyalarındaki Farklılıklar

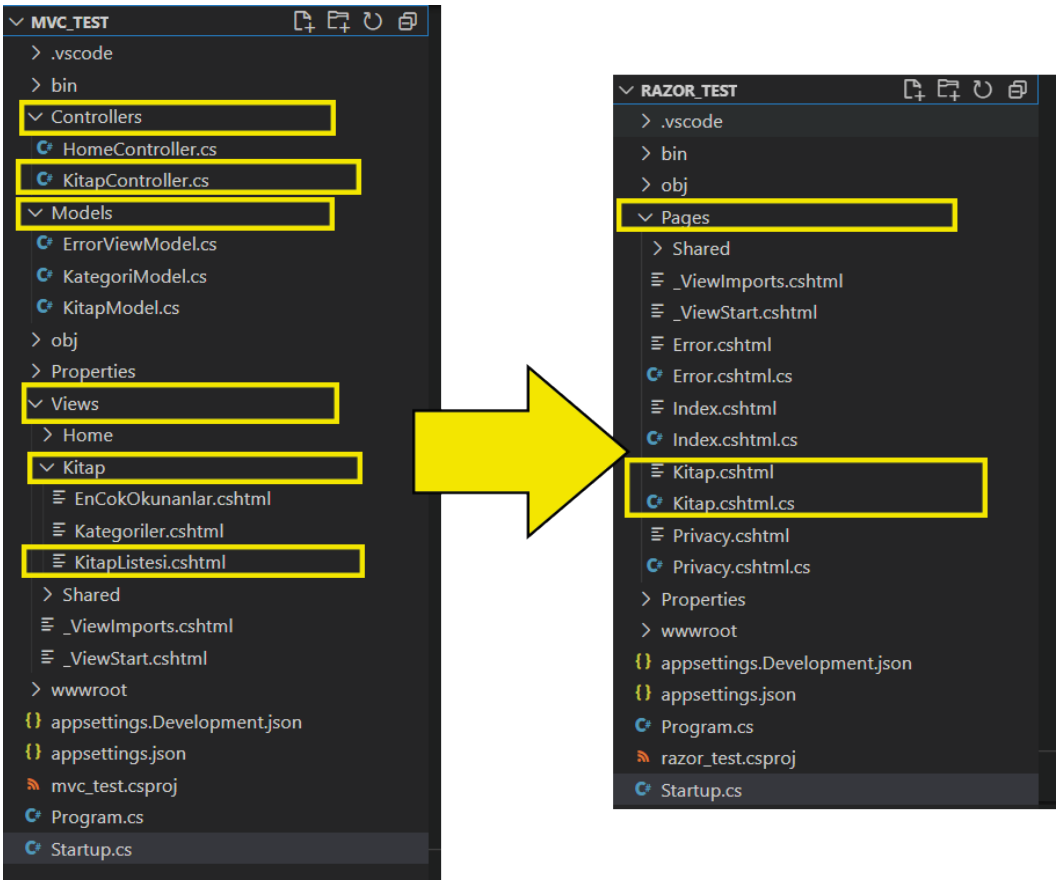
MVC	Razor Page

public void ConfigureServices(IServiceCollection services)
{
services.AddControllersWithViews();
}
	public void ConfigureServices(IServiceCollection services)
	{
	services.AddRazorPages();
	}

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
app.UseEndpoints(endpoints =>
{
endpoints.MapControllerRoute(
name: "default",
pattern: "{controller=Home}/{action=Index}/{id?}");
});
}
	public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
	{
	app.UseEndpoints(endpoints =>
	{
	endpoints.MapRazorPages();
	});
	}

Klasör / Dosya Yapıları

MVC'deki Models, Controllers ve Views klasörlerinin yerine Razor Pages'te Pages klasörü bulunur (Görsel 6.27).



Görsel 6.27: MVC ve Razor Pages proje yapıları

Razor View Motoru

Hem MVC'de hem de Razor Pages'te view tarafında server bazlı kodların çalıştırılması ve veri görüntüleme işlemleri için Razor View motoru kullanılır.

Kullanım Alanları

Her projede MVC veya Razor Page tercih edilebilir. Ancak genel olarak küçük çaplı veya sayfa bazlı uygulamalarda Razor Page kullanılması daha büyük projelerde ise MVC yapısının kullanılmasının daha uygun olacağı söylenebilir. Bir projede bu iki alt yapıdan sadece birinin kullanılması zorunluluğu yoktur. Aynı proje içerisinde hem MVC hem de Razor Page alt yapıları birlikte kullanılabilir.

6.3. Standart Klasör ve Dosyalar

Bir MVC projesi oluşturulduğunda bazı klasör ve dosyaların standart olarak bulunduğu görülür. Bu klasör ve dosyaların her birinin ayrı ayrı işlevleri bulunur. Bu klasör ve dosyaların özellikleri şunlardır:

.vscode

Visual Studio Code editörü ayarlar için iki farklı kapsam sunar.

- Kullanıcı ayarları (Editörün tema, font ayarları vb.),
- Projeye özgü ayarlar

Projeye özgü ayarlar, proje klasör listesindeki **.vscode** klasörü altında bulunur. Projenin derlenmesi ve yayınlanması gibi standart işlemlerin otomatikleştirilmesi için **task.json** dosyasındaki ayarlar, projenin çalıştırılması için **launch.json** dosyasındaki ayarlar kullanılır.

bin

Proje derlendiğinde oluşturulan dosyalar **bin** klasörü altında bulunur.

Properties

Bu klasör altında bulunan **launchSettings.json** dosyası aracılığıyla projenin çalışma zamanı ayarları değiştirilebilir. Örneğin, projenin 5001 No.lu portta değil de başka bir portta çalıştırılması istenirse bu dosyadaki ayarlar değiştirilmelidir.

Models, Controllers ve Views

MVC tasarım deseni kullanılan .NET projelerinde bu üç klasör, her bir katmanda kullanılan dosyaların ayrı ayrı klasörlenmesini sağlar.

wwwroot

Proje içerisinde kullanılan her türlü statik (CSS, JS, resim, video, vb.) dosya, bu klasör altında bulunur. Bu klasöre eklenen dosyaları **https://localhost:5001/css/site.css** gibi bir link ile tarayıcıda görüntülemek mümkündür.

```
<link rel="stylesheet" href="~/css/site.css" />
<script src="~/lib/jquery/dist/jquery.min.js"></script>
<img src="~/images/meblogo.png" alt="MEB Logo" />
```

gibi statik dosya kullanımlarında dosya adreslerinin başındaki **~ (tilde)** sembolü, wwwroot klasörünü temsil eder.

Views > _ViewImports.cshtml

Projedeki tüm sayfalar için gerekli isim uzaylarını kullanmak gibi tüm sayfalar için ortak olan yönergeler bu dosya içerisinde tanımlanır.

Aşağıda örnek bir _ViewImports.cshtml dosyası gösterilmiştir.

```
@using WebApplication1
@using WebApplication1.Models
@using WebApplication1.Models.AccountViewModels
@using WebApplication1.Models.ManageViewModels
@using Microsoft.AspNetCore.Identity
@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
```

Bu dosyada kullanılabilecek yönergeler şunlardır: @addTagHelper, @removeTagHelper, @tagHelperPrefix, @using, @model, @inherits, @inject

Views > _ViewStart.cshtml

Her sayfanın gösterilmesinden önce çalıştırılması istenen kodlar bu dosya içerisinde bulunur. Aşağıda örnek bir **_ViewStart.cshtml** dosyası gösterilmiştir.

```
@{
    Layout = "_Layout";
}
```

Views > Shared > _Layout.cshtml

_Layout.cshtml dosyası, tüm sayfalar için ortak bir şablon yapısı tanımlamak için kullanılır. Bu dosya içerisindeki **@RenderBody()** yönergesi ile bu şablon dosyasını kullanan sayfaların içeriğinin görüntüleneceği kısım tanımlanır.

```
<!DOCTYPE html>
<html>
<head>
...
</head>
<body>
...
   @RenderBody()
...
</body>
</html>
```

Program.cs

Her C# programında olduğu gibi programın başlangıç noktası public ve static erişim tiplerine sahip **Main()** metodudur. Main metodu **Program.cs** içerisinde yer alır ve web uygulamasını barındıran bir nesne oluşturur. Bu nesne, projelerin çalışmasını sağlar.

Aşağıda Program.cs dosyasının içeriği görüntülenmiştir.

```
public class Program
{
   public static void Main(string[] args)
   {
      CreateHostBuilder(args).Build().Run();
   }
   public static IHostBuilder CreateHostBuilder(string[] args) =>
      Host.CreateDefaultBuilder(args)
         .ConfigureWebHostDefaults(webBuilder =>
         {
            webBuilder.UseStartup<Startup>();
         });
}
```

Startup.cs

Startup.cs dosyası, .NET projelerindeki en önemli dosyalardan biridir. İçerisinde iki adet metot bulunur. Bu metotların önemli işlevleri şunlardır:

- **ConfigureServices**

Projede kullanılacak hizmetleri eklemek ve yapılandırmak için kullanılır. Proje ilk çalıştırıldığında öncelikle **ConfigureServices** metodu çağrılır.

Aşağıda örnek bir ConfigureService metodu görülmektedir.

```
public void ConfigureServices(IServiceCollection services)
{
   services.AddControllersWithViews();
}
```

services.AddControllersWithViews() ifadesi ile projede MVC tasarım deseni kullanılacağı bildirilmektedir. Ayrıca projede; veri tabanı, çerez (cookie), lokalizasyon (localization), oturum (session) gibi servisler kullanılması gerektiğinde bu servisler, **ConfigureServices** metodunda yapılandırılmalıdır.

Not

Razor Pages'te bu noktada **services.AddRazorPages()** ifadesinin kullanıldığı hatırlanmalıdır.

• **Configure**

Bu metot, uygulamadaki her bir isteğe nasıl yanıt verileceğini tanımlamak için kullanılır. ConfigureServices metoduna herhangi bir hizmet eklendiğinde eklenen hizmet bu metot içerisinde kullanılabilir.

Örnek bir Configure metodu aşağıdaki gibidir.

```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
   if (env.IsDevelopment())
   {
      app.UseDeveloperExceptionPage();
   }
   else
   {
      app.UseExceptionHandler("/Home/Error");
      app.UseHsts();
   }
   app.UseHttpsRedirection();
   app.UseStaticFiles();

   app.UseRouting();

   app.UseAuthorization();

   app.UseEndpoints(endpoints =>
   {
      endpoints.MapControllerRoute(
         name: "default",
         pattern: "{controller=Home}/{action=Index}/{id?}");
   });
}
```

Proje geliştirme aşamasındayken detaylı hata mesajlarının gösterilmesi, HTTP isteklerinin HTTPS'ye yönlendirilmesi, statik dosyalar için wwwroot klasörünün kullanılması, varsayılan controller / action tanımlamaları gibi ayarlar bu metot içerisinde tanımlanır.

appsettings.json

Projede yapılandırma ayarlarının tanımlandığı dosyadır. Veri tabanı bağlantı bilgisi gibi çalışma ortamına bağlı bilgiler bu dosyada tutulur.

Aşağıda örnek bir appsettings.json dosyası görülmektedir.

```
{
 "ApplicationInfo": {
  "Version": "1.2",
  "Build": 2136
 },
 "ConnectionStrings": {
  "DefaultConnection": "server=localhost;user=root;password=1234;database=testdb;"
 }
}
```

Not

appsettings.json dosyası projenin ana klasöründe yer almalıdır.

Sıra Sizde

Yeni bir MVC projesi oluşturup standart olarak eklenen klasör ve dosyaların içeriklerini inceleyiniz.

6.4. Ara Katman (Middleware)

Ara katmanlar, istek / cevap (request/response) süreçlerinde özelleştirilmiş kod bloklarının çalıştırılmasını sağlayan modüler ve etkili yapılardır.

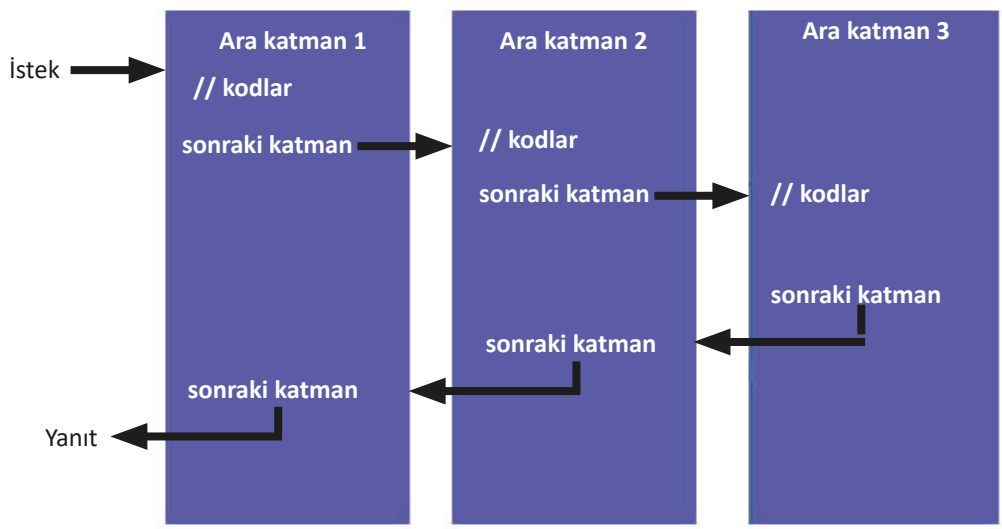
Ara katmanlar ile;

- Gelen isteklerin kontrolden geçirilmesi,
- Cevapların önbellek üzerinden döndürülmesi,
- Uygulamada kayıt mekanizmalarının (logging) oluşturulması,
- Hata yakalama çözümlerinin sunulması sağlanabilir.

Ara katman yapıları; **IApplicationBuilder** arayüzünden türetilen bir sınıfa, uzantı (extension) olarak tanımlanan metotların istenilen sırada çalıştırılmasını sağlar. Çalıştırma sırası **"işlem hattı"** (Pipeline) olarak isimlendirilir. Önceden oluşturulan projelerdeki **Startup.cs** dosyası incelendiğinde **configure** metodunda kullanılan ifadelerin hepsinin bir ara katman olduğu söylenebilir.

```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
   if (env.IsDevelopment())
   {
      app.UseDeveloperExceptionPage();
   }
   else
   {
      app.UseExceptionHandler("/Home/Error");
      app.UseHsts();
   }
   app.UseHttpsRedirection();
   app.UseStaticFiles();

   app.UseRouting();

   app.UseAuthorization();

   app.UseEndpoints(endpoints =>
   {
      endpoints.MapControllerRoute(
         name: "default",
         pattern: "{controller=Home}/{action=Index}/{id?}");
   });
}
```

Her katman, kendisinden sonraki katmandan önce ve / veya sonra işlemler gerçekleştirebilir (Görsel 6.28). Katmanların configure metodundaki sırayla çağrıldığı unutulmamalıdır. Örneğin, hata yakalama işlemi gerçekleştirilmek isteniyorsa işlem hattında erken bir yerde çağrılmalıdır ki kendisinden sonraki katmanlarda oluşan hataları yakalayabilsin.



Görsel 6.28: İşlem hattı

Aşağıda en basit hâliyle gelen tüm istekleri karşılayan bir ara katman görülmektedir.

```
public class Startup
{
   public void Configure(IApplicationBuilder app)
   {
      app.Run(async context =>
      {
         await context.Response.WriteAsync("Merhaba Dünya!");
      });
   }
}
```

app.Run() ifadesi ile gelen istek ne olursa olsun sayfaya "Merhaba Dünya!" yazılması sağlanmıştır.

app.Use() ifadesindeki **next** ifadesi, işlem hattında bir sonraki katmanı temsil eder. Aşağıda, gelen isteği karşılayan ve bir sonraki katmanı çağıran kod örneği görülmektedir.

```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
   app.Use(async (context, next) =>
         {
            await context.Response.WriteAsync("1. katmana gelindi.");
            await next.Invoke();
            await context.Response.WriteAsync("1. katmandan çıkıldı.");
         });

   app.Run(async context =>
   {
      await context.Response.WriteAsync("2. katmana gelindi");
   });
}
```

Bu uygulama çalıştırıldığında herhangi bir istek karşısında sayfa çıktısının şu şekilde olacağı görülecektir:

```
1. katmana gelindi.
2. katmana gelindi.
1. katmandan çıkıldı.
```

Not

app.Run() metodunda next temsilcisinin olmadığına dikkat edilmelidir.

6.4.1. İşlem Hattı (Pipeline)

İşlem hattı sırası, Startup.cs'teki configure metodundaki sırayı ve yanıtın ters sırasını tanımlar. İşlem hattı sırası performans, güvenlik ve işlevsellik açısından önemlidir. Bu nedenlerden dolayı işlem hattı sırasına dikkat edilmelidir.

Aşağıdaki Configure metodu ara katmanların önerilen sıralamasını göstermektedir.

```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
   if (env.IsDevelopment())
   {
      app.UseDeveloperExceptionPage();
      app.UseDatabaseErrorPage();
   }
   else
   {
      app.UseExceptionHandler("/Error");
      app.UseHsts();
   }

   app.UseHttpsRedirection();
   app.UseStaticFiles();
   app.UseCookiePolicy();

   app.UseRouting();
   app.UseRequestLocalization();

   app.UseCors();
   app.UseAuthentication();
   app.UseAuthorization();

   app.UseSession();
   app.UseResponseCompression();
   app.UseResponseCaching();

   app.UseEndpoints(endpoints =>
   {
      endpoints.MapControllerRoute(
         name: "default",
         pattern: "{controller=Home}/{action=Index}/{id?}");
   });
}
```

Bu sıralamada UseCors(), UseAuthentication(), UseAuthorization() belirtilen sıralaması önemlidir ve değiştirilmemelidir. Ancak projede sıkıştırılmış statik dosyaların önbelleğe alınmasına izin vermek için sıralama şu şekilde değiştirilebilir:

```
app.UseResponseCaching();
app.UseResponseCompression();
app.UseStaticFiles();
```

6.4.2. İşlem Hattını Bölümleme

app.Map() ifadesi ile işlem hattını bölümlere ayırmak mümkündür. Gelen isteğin adresine göre bölümlendirme örneği aşağıda gösterilmiştir.

```
public class Startup
{
   private static void Istek1(IApplicationBuilder app)
   {
      app.Run(async context =>
      {
         await context.Response.WriteAsync("İstek 1");
      });
   }

   private static void Istek2(IApplicationBuilder app)
   {
      app.Run(async context =>
      {
         await context.Response.WriteAsync("İstek 2");
      });
   }

   public void Configure(IApplicationBuilder app)
   {
      app.Map("/istek1", Istek1);

      app.Map("/istek2", Istek2);

      app.Run(async context =>
      {
         await context.Response.WriteAsync("İstek anlaşılmadı!");
      });
   }
}
```

Gelen isteğe göre ekran çıktıları Tablo 6.3'teki gibi olacaktır.

Tablo 6.3: İşlem Hattının Bölümlendirilmesi

İstek (Request)	Cevap (Response)
https://localhost:5001/	İstek anlaşılmadı!
https://localhost:5001/istek1	İstek 1
https://localhost:5001/istek1	İstek 2
https://localhost:5001/istek3	İstek anlaşılmadı!

6.4.3. Dâhilî Ara Katmanlar ve Öncelik Sıraları

ASP.NET Core ile dâhilî olarak gelen ara katmanlardan bazıları Tablo 6.4'te listelenmiştir. Ara katmanın işlem hattındaki sırası tablodan incelenebilir.

Tablo 6.4: Dâhilî Ara Katmanlar

Ara Katman	Açıklama	Sırası
Authentication	Kimlik doğrulama desteği sunar.	HttpContext.User kullanımından önce
Authorization	Yetkilendirme desteği sunar.	Authentication'dan hemen sonra
Cookie Policy	Kişisel bilgilerin tarayıcıda depolanmasını sağlar.	Authentication, Session ve MVC'den önce
CORS	Kaynak paylaşımını yapılandırır	CORS kullanımından önce
Diagnostics	Hata gösterme, hata yakalama, durum kodu sayfaları için destek sağlar.	Hata oluşturabilecek katmanlardan önce
HTTPS Redirection	Tüm HTTP isteklerini HTTPS'ye yönlendirir.	URL kullanımından önce
HSTS	Güvenlik arttırma	Yanıtlar gönderilmeden önce
Response Caching	Yanıtları ön belleğe alma desteği sunar.	Ön bellek kullanımından önce (CORS'tan sonra olmalıdır)
Response Compression	Yanıtları sıkıştırmak için destek sağlar.	Sıkıştırma kullanımından önce
Request Localization	Yerelleştirme desteği sunar.	Yerelleştirme kullanımından önce
Endpoint Routing	İstekleri ilgili controllera yönlendirir.	-
Session	Oturum yönetimi desteği sunar.	Oturum kullanımından önce
Static Files	Statik dosyaların wwwroot klasörü altında kullanımını sağlar.	-

Sıra Sizde

Projenize, her istekte isteğin adresini ve saatini bir metin dosyasına yazan ara katmanı ilave ediniz.

6.5. Yönlendirme (Routing)

Yönlendirme, gelen isteklerin hangi controller ve hangi action metoda yönlendirileceğini belirleyen ASP.NET Core'daki önemli bileşenlerden birisidir.

Web sitelerinde; gelen isteklerin URL yapılarının, arama motorlarının anlayabileceği, sayfa içeriğine uygun, anlamlı ve kullanıcılar tarafından okunabilir olması önemlidir. Yönlendirme konusu bu açıdan önemlidir.

6.5.1. Varsayılan Yönlendirme

Yeni bir MVC projesi oluşturulduğunda varsayılan olarak aşağıdaki şekilde bir yönlendirme oluşturulur:

```
app.UseEndpoints(endpoints =>
{
   endpoints.MapControllerRoute(
      name: "default",
      pattern: "{controller=Home}/{action=Index}/{id?}");
});
```

Bu yönlendirme tanımında:

• **name=”default”** ifadesi ile yönlendirme tanımına bir isim verilmiştir. İsteklerin eşleştirilmesinde hiçbir etkisi yoktur. Her yönlendirme tanımının ismi farklı olmalıdır.

• **controller=Home** ifadesi ile eğer controller ismi belirtilmezse yönlendirilecek varsayılan controller tanımlanır.

•**action=Index** ifadesi ile eğer action ismi belirtilmezse yönlendirilecek varsayılan action metot tanımlanır.

Proje çalıştırılıp **https://localhost:5001/** adresi görüntülendiğinde controller ve action belirtilmemesine rağmen HomeController’daki ındex action metodunun çalıştırılması bundan dolayıdır. **https://localhost:5001/Home/Index** adresi de aynı sayfaya yönlendirilmektedir.

Sondaki **id?** ifadesi ise bu action metoda parametre gönderilmesini sağlar. **“?”** karakteri ile bu parametrenin seçimlik olacağı belirtilir.

Not

“?” kullanılmadığı durumda **https://localhost:5001/123** şeklinde bir parametre tanımlanması zorunludur.

Tablo 6.5’te örnek isteklerin yönlendirileceği controller ve action metotlar listelenmiştir.

Tablo 6.5: Yönlendirme Örnekleri

İstek	Controller / Action
https://localhost:5001	Home / Index
https://localhost:5001/Home/Index	Home / Index
https://localhost:5001/Home/Index/123	Home / Index
https://localhost:5001/Home/Hakkinda	Home / Hakkinda
https://localhost:5001/Haberler	Haberler / Index
https://localhost:5001/Haberler/123	Haberler / Index
https://localhost:5001/Haberler/EnCokOkunanlar	Haberler / EnCokOkunanlar
https://localhost:5001/Haberler/EnCokOkunanlar/123	Haberler / EnCokOkunanlar
https://localhost:5001/Duyurular	Duyurular / Index
https://localhost:5001/Duyurular/SonEklenenler	Duyurular / SonEklenenler

6.5.2. Özel Yönlendirme Oluşturma

Yönlendirme ifadeleri istenildiği kadar eklenebilir. Aşağıda **https://localhost:5001/Haberler** şeklindeki bir isteği, HaberlerController'daki HaberListesi action metoduna yönlendiren ifade gösterilmiştir.

```
endpoints.MapControllerRoute(
   name: "haberler",
   pattern: "Haberler",
   defaults: new { controller = "Haberler", action = "HaberListesi" });
```

pattern ifadesi, eşleştirilmesi istenen isteğin yolunu; **defaults** ifadesi ise yönlendirmenin yapılacağı controller ve action metodun tanımlanmasını sağlar.

Aşağıda **https://localhost:5001/Haber/123** şeklindeki bir isteğin, HaberlerController'daki HaberDetay action metoduna yönlendirilmesi gösterilmiştir.

```
endpoints.MapControllerRoute(
   name: "haberdetay",
   pattern: "Haber/{id}",
   defaults: new { controller = "Haberler", action = "HaberDetay" });
```

Bu yönlendirmeye uygun action metot şu şekilde yazılmalıdır:

```
public class HaberlerController : Controller
{
   public IActionResult HaberDetay(int id)
   {
      return View(id);
   }
}
```

Artık HaberDetay view sayfası içerisinde parametre gösterilebilir:

```
@model int

<h1>Haber detay sayfası</h1>
Gelen parametre: @Model
```

Not

İsteklerin tanımlanma sırasına dikkat edilmelidir. Bir istek, eşleşen ilk yönlendirme ifadesinde ilgili controller ve action metoda yönlendirilir.

Şu ana kadar gösterilen yönlendirme yöntemleri "Geleneksel Yönlendirme" olarak isimlendirilmektedir.

Sıra Sizde

Projenize eklediğiniz YazarListesi, YazarinKitaplari action metotları için özel yönlendirmeler oluşturunuz.

6.5.3. Parametre Kısıtlamaları

Gelen istekteki parametrelere kısıtlama ekleyerek bu kısıtlamaların dışında bir değer gönderilmesi engellenebilir. Sonuçta hiçbir yönlendirme ifadesi ile eşleşmeyen isteklere karşı 404 durum kodu [Page not found (Sayfa bulunamadı)] hatası döndürülecektir.

```
endpoints.MapControllerRoute(
name: "...",
pattern: "Haber/{id}",
defaults: new {...});
```

Bu örnekte **id** olarak tanımlanan parametrenin tipi ile ilgili bir bildirim yapılmamıştır. Sadece **int** tipindeki parametrelerin action metoda yönlendirilmesini sağlamak için aşağıdaki yönlendirme ifadesi kullanılmalıdır.

```
endpoints.MapControllerRoute(
name: "...",
pattern: "Haber/{id:int}",
defaults: new {...});
```

Eğer parametrenin seçimlik olması isteniyorsa bu durumda action metotta parametreye varsayılan değer atanabilir.

```
endpoints.MapControllerRoute(
name: "haberdetay",
pattern: "Haber/{id?}",
defaults: new { controller = "Haberler", action = "HaberDetay" });
```

Yönlendirme ifadesindeki **"?"** karakterine dikkat edilmelidir. Bundan sonra action metotta parametrenin varsayılan değeri tanımlanabilir.

```
public class HaberlerController : Controller
{
   public IActionResult HaberDetay(int id = 100)
   {
      return View(id);
   }
}
```

Alternatif olarak yönlendirme tanımında da parametreye varsayılan bir değer atanabilir. Bu durumda, parametre seçimlik olarak tanımlanamaz.

```
endpoints.MapControllerRoute(
name: "haberdetay",
pattern: "Haber/{id:int}",
defaults: new { controller = "Haberler", action = "HaberDetay", id = 100 });
```

6.5.3.1. Diğer Parametre Kısıtlamaları

Tablo 6.6'da parametre kısıtlamaları ve örnek eşleşmeler listelenmiştir.

Tablo 6.6: Parametre Kısıt Listesi

Kısıtlama	Kullanımı	Örnek Eşleşmeler	Notlar
int	{sayi:int}	123456789 -123456789	Herhangi bir tamsayıyla eşler.
bool	{aktif:bool}	true false	bool veri tipi ile eşler.
datetime	{tarih:datetime}	2016-12-31 2016-12-31 7:32pm	DateTime tipi ile eşler.
decimal	{fiyat:decimal}	49.99 -1,000.01	decimal veri tipi ile eşler.
double	{hacim:double}	1.234, -1,001.01e8	double veri tipi ile eşler.
float	{hacim:float}	1.234 -1,001.01e8	float veri tipi ile eşler.
GUID	{referans:guid}	CD2C1638-1638 72D5-1638-DEADBEEF1638	Geçerli bir GUID ile eşler.
long	{milisaniye:long}	123456789 -123456789	long veri tipi ile eşler.
minlength(value)	{ad:minlength(4)}	Mehmet	En az 4 karakterlik bir string ile eşler.
maxlength(value)	{dosyaadi:maxlength(8)}	Ali	En fazla 8 karakterlik bir string ile eşler.
length(length)	{dosyaadi:length(9)}	rapor.txt	9 karakterlik bir string ile eşler.
length(min,max)	{dosyaadi:length(5,16)}	raporlar.txt	5-16 arası karakterlik string ile eşler.
min(value)	{yas:min(18)}	19	Tamsayı değeri en az 18 olan int veri tipi ile eşler.
max(value)	{yas:max(120)}	91	Tamsayı değeri en fazla 120 olan int veri tipi ile eşler.
range(min,max)	{yas:range(18,120)}	91	Tamsayı değeri 18-120 arası olan int veri tipi ile eşler.
alpha	{ad:alpha}	Ahmet	Bir veya daha fazla karakterden (a-z) oluşan string tipi ile eşler.
regex(expression)	{kredikarti:regex(^4[0-9]{12}(?:[0-9]{3})?$)}	123-45-6789	Düzenli ifadeye (Regular Expression) uygun ise eşler.
required	{bakanlik:required}	MEB	İstekte olmayan bir değerin kullanılmasını gerektirir.

Bu kısıtlamalar aynı parametrede ":" karakteri ile kullanılabilir. Örneğin bir şehrin hava durumu bilgisini göstermek için **sehirId** parametresi kullanılması gerektiği varsayılırsa bu parametrenin int tipinde ve 1-81 arası bir değer alması beklenir. Bu durumda aşağıdaki yönlendirme ifadesi kullanılabilir:

```
endpoints.MapControllerRoute(
   name: "havadurumu",
   pattern: "HavaDurumu/{sehirId:int:range(1,81)}",
   defaults: new { controller = "HavaDurumu", action = "Sehir" });
```

6.5.4. Öznitelik Yönlendirmeleri

ASP.NET Core'da yönlendirmeler, geleneksel yönlendirme ifadeleri ile yapılabileceği gibi controller ve action metot seviyesinde **Route("..")** ifadesi ile de yapılabilir. Controller ve action metot seviyesinde yapılan yönlendirmeler "Öznitelik Yönlendirmesi" olarak adlandırılır.
Öznitelik yönlendirmelerinin kullanılabilir olması için Startup.cs dosyasında bir değişiklik yapılması gerekir. Geleneksek yönlendirmedeki **endpoints.MapControllerRoute(...);** ifadesi **endpoints.MapControllers();** ile değiştirilmelidir.

```
app.UseEndpoints(endpoints =>
   {
      endpoints.MapControllers();
   });
```

Bir controller veya bir action metot için birden fazla yönlendirme ifadesi kullanılabilir. Örnek bir tanımlama aşağıda gösterilmiştir.

```
[Route("")]
[Route("Home")]
public class HomeController : Controller
{
   [Route("")]
   [Route("Index")]
   [Route("Index/{id?}")]
   public IActionResult Index(int? id)
   {
      return View();
   }

   [Route("Hakkinda")]
   public IActionResult Hakkinda()
   {
      return View();
   }
```

Route("") ifadesi ile controller veya action metot boş geçildiğinde yönlendirilecek controller ve action metodu tanımlanır.

Proje çalıştırıldığında şu isteklerin başarıyla yönlendirildiği görülecektir:

- https://localhost:5001/
- https://localhost:5001/Home
- https://localhost:5001/Home/Index
- https://localhost:5001/Home/Index/123
- https://localhost:5001/Home/Hakkinda

Not

İsteklerin tanımlanma sırasına dikkat edilmelidir. Bir istek, eşleşen ilk yönlendirme ifadesinde ilgili controller ve action metoda yönlendirilir.

Sıra Sizde

Projenize eklediğiniz YazarListesi ve YazarinKitaplari action metotları için öznitelik yönlendirmelerini tanımlayınız.

6.5.5. İstek Yolunu Değiştirme

Gelen isteklerin yollarının controller ve action metot ile aynı olması zorunlu değildir. İstenilen herhangi bir istek için yönlendirme yapılabilir.

```
public class HaberlerController : Controller
{
   [Route("Haberler")]
   public IActionResult HaberListesi()
   {
      return View();
   }
   [Route("Detay/{id:int}")]
   public IActionResult HaberDetay(int id)
   {
      return View(id);
   }
}
```

Yukarıdaki yönlendirmeler şu istekleri başarıyla gerçekleştirecektir.

- https://localhost:5001/Haberler
- https://localhost:5001/Detay/123

Bu kullanım, arama motoru optimizasyonu için oldukça faydalıdır.

Sıra Sizde

Projenize eklediğiniz YazarListesi ve YazarinKitaplari action metotları “Yazarlar” ve “Yazar/KitapListesi” yönlendirmelerini yapınız.

6.5.6. Belirteç Değiştirme

Geleneksel yönlendirmeden hatırlanacağı üzere "{controller=Home}/{action=Index}/{id?}" ifadesi ile controller ve action metotlara yönlendirme yapılabiliyordu. Öznitelik yönlendirmesinde de benzer şekilde tanımlandığı controller ve action metodu kullanan bir yönlendirme ifadesi kullanılabilir.

```
public class HomeController : Controller
{
   [Route("")]
   [Route("Home")]
   [Route("[controller]/[action]")]
   public IActionResult Index()
   {
      return View();
   }

   [Route("[controller]/[action]")]
   public IActionResult About()
   {
      return View();
   }
}
```

[Route("[controller]/[action]")] ifadesinde [controller] Home'u, [action] ise ilgili action metodun adını temsil eder. Dolayısıyla aşağıdaki;

- https://localhost:5001/
- https://localhost:5001/Home
- https://localhost:5001/Home/Index
- https://localhost:5001/Home/Hakkinda istekleri otomatik olarak bu controller ve action metotlar tarafından karşılanacaktır.

6.6. Form İşlemleri

Formlar web dünyası için büyük öneme sahiptir. Formlar sayesinde kullanıcı; arama motorları, e-ticaret, bankacılık vb. sitelerle etkileşime geçer. Formlar olmasaydı internet sadece okunabilen belgelerden oluşan bir depo hâlini alırdı.

Formlar, kullanıcının girdiği verilerin işlenmesi ve web sunucusuna aktarılması için kullanılır. Sunucuya aktarılan veriler; veri tabanına kaydetme, e-posta oluşturma veya algoritmaya dayalı işlemler yapma gibi aksiyonlarda kullanılır.

Formların içerisinde bulunan input elemanları ile sayfa içerisinden veri girişleri yapılır ve bu veriler sunucuya gönderilir. Sunucuya gönderilen bilgilerin hangi input elemanına ait olduğunun belirlenmesi için input elemanlarının **name** özelliği kullanılır. Veriler gönderilirken hangi sunucuya gideceği ve nasıl gönderileceği form etiketi içinde bulunan **Action** ve **Method** adında iki özellikle belirlenir.

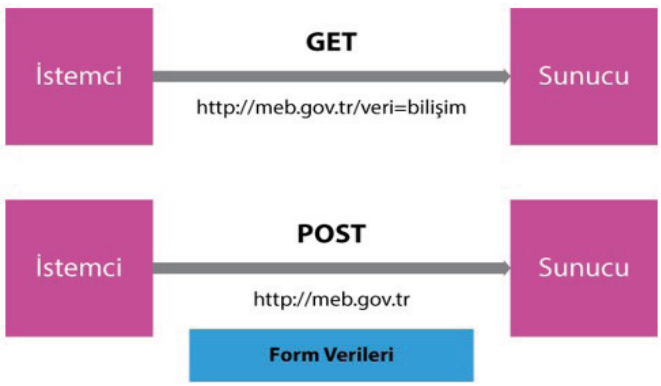
Action özelliği, tarayıcıya bilgilerin gideceği sunucunun URL adresini belirtir. Bu URL adresi aynı sunucuda göreceli veya mutlak adres olabileceği gibi farklı bir sunucu adresi de olabilir. ASP.NET Core MVC mimarisinde genellikle formun action özelliği son hedef (endpoint) olarak controller içindeki bir action olarak belirtilir.

Method özelliği, tarayıcıya bilgilerin nasıl gideceğini belirtir. Temel olarak **GET** ve **POST** metotları kullanılır. Tablo 6.7'de GET ve POST metotlarının karşılaştırılması verilmiştir.

Tablo 6.7: GET ve POST Metotlarının Karşılaştırılması

GET Metodu	POST Metodu
Hızlıdır fakat güvenli değildir.	Yavaştır fakat güvenlidir.
Varsayılan metottur.	Varsayılan metot değildir.
Form verileri adres satırına eklediğinden, veriler diğer kullanıcılar tarafından görülebilir.	Form verileri diğer kullanıcılar tarafından görülemez.
Önemsiz verileri göndermek için kullanışlıdır.	Önemli verileri göndermek için kullanılır.
Sadece metin verileri taşır.	Hem metin hem de binary verileri taşır.
Veri taşıma sınırı vardır, query string'in uzunluğu ile sınırlıdır.	Veri taşıma sınırı yoktur, sınırsız form verileri taşınabilir.

Görsel 6.29'da veriler GET metodu ile adres satırından gönderilirken POST metodunda ise HTTP istekleri içerisinde paketlenerek gönderilmektedir.



Görsel 6.29: GET ve POST metotları veri iletim yöntemleri

6.6.1. Form GET Metodu Kullanımı

Form içerisinden girilen bilgiler GET metodu kullanılarak gönderilmek istendiğinde, form içindeki input elemanlarının name değerleri parametre; input içine girilen bilgiler ise parametrelerin değerleri olarak adres satırından query string olarak gönderilir. Form içinde bulunan buton tıklandığında veri gönderim işlemi başlatılır. Form içerisinden GET metodu ile gönderilen veriler **HttpGet** niteliğine sahip action tarafından alınarak işlenir.

Not

Controller içindeki actionlara herhangi bir nitelik verilmediği takdirde, ön tanımlı olarak HttpGet niteliğine sahip olurlar.

1. Uygulama

Bir form oluşturarak GET metodu ile sunucuya veri gönderme işlemini yönergeler doğrultusunda gerçekleştiriniz.

1. Adım: Bilgisayarınıza FormUygulama adında MVC şablonunu kullanarak bir proje oluşturunuz. Projenin Views/Home klasöründeki Index.cshtml dosyasını açarak aşağıdaki kodlamaları yapınız.

```
<form method= "GET" action= "/Home/VeriAlGet">
  <div class= "form-group">
    <label>Okul No</label>
    <input type= "text" name="okulno" class="form-control" ></input>
  </div>
  <div class= "form-group">
    <label>Ad</label>
    <input type= "text" name="ad" class="form-control" ></input>
  </div>
  <div class= "form-group">
    <label>Soyad</label>
    <input type= "text" name="soyad" class="form-control" ></input>
  </div>
  <div class= "form-group">
    <label>Sınıf</label>
    <input type= "text" name="sinif" class="form-control"></input>
  </div>
  <div class= "form-group">
    <button type= "submit" class= "btn btn-primary" >Gönder</button>
  </div>
</form>
```

2. Adım: HomeController içindeki Index Action metodunu aşağıdaki gibi düzenleyiniz.

```
[HttpGet]
public IActionResult Index()
{
  return View();
}
```

3. Adım: Home Controller içine **VeriAlGet** adında bir action metodu oluşturunuz. Metot parametrelerini form içindeki input elemanlarının name değerleri aynı olacak şekilde oluşturunuz.

```
[HttpGet]
public IActionResult VeriAlGet(int okulno,string ad,string soyad,string sinif)
{
  return Content(okulno+"-"+ad+"-"+soyad+"-"+sinif);
}
```

4. Adım: Uygulamayı çalıştırarak formdan gönderilen verilerin sunucu tarafından alındığını gözlemleyiniz.

2. Uygulama

Sunucuya GET metodu ile veri gönderirken action metot içine parametre yazmak yerine **HttpContext** sınıfı kullanılarak formdan gönderilen verileri alma işlemini yönergeler doğrultusunda gerçekleştiriniz.

1. Adım: Uygulama 1'deki VeriAlGet actionı aşağıdaki gibi düzenleyiniz.

```
[HttpGet]
public IActionResult VeriAlGet()
{
  int okulno=int.Parse(HttpContext.Request.Query["okulno"]);
  string ad=HttpContext.Request.Query["ad"];
  string soyad=HttpContext.Request.Query["soyad"];
  string sinif=HttpContext.Request.Query["sinif"];
  return Content(okulno+"-"+ad+"-"+soyad+"-"+sinif);
}
```

2. Adım: Uygulamayı çalıştırarak formdan gönderilen verilerin sunucu tarafından alındığını gözlemleyiniz.

6.6.2. Form POST Metodu Kullanımı

POST metodunda formdan alınan bilgiler sunucuya tarayıcı tarafından **Form Data** olarak gönderilir. GET metodunun aksine Form Data bilgileri adres çubuğunda görünmez. Gönderilen bu bilgiler; ASP.NET Core uygulamalarında veri tabanına kaydetme, e-posta olarak gönderme, işlem yapma vb. amaçları doğrultusunda kullanılır. Form içerisinden POST metodu ile gönderilen veriler **HttpPost** niteliğine sahip action tarafından alınarak işlenir.

POST metodu ile Controller'a veri gönderme farklı yollardan yapılabilir. Bu yollar;

- Parametre kullanma,
- IFormCollection sınıfı kullanma,
- Model Binder (Model Bağlama) kullanmadır.

3. Uygulama

POST metodunu kullanarak form içine girilen bilgileri action parametreleri ile sunucuya aktarma işlemini yönergeler doğrultusunda gerçekleştiriniz.

Not

Action parametreleri ile form içindeki input name değerleri aynı olmalıdır.

1. Adım: FormUygulama projesinin Views/Home klasörü içindeki Index.cshtml dosyasındaki formu aşağıdaki gibi düzenleyiniz.

```
<form method="POST" action="/Home/Index">
  <div class="form-group">
    <label>Okul No</label>
    <input type="text" name="okulno" class="form-control"></input>
  </div>
  <div class="form-group">
    <label>Ad</label>
    <input type="text" name="ad" class="form-control"></input>
  </div>
  <div class="form-group">
    <label>Soyad</label>
    <input type="text" name="soyad" class="form-control"></input>
  </div>
  <div class="form-group">
    <label>Sınıf</label>
    <input type="text" name="sinif" class="form-control"></input>
  </div>
  <div class="form-group">
    <button type="submit" class="btn btn-primary">Gönder</button>
  </div>
</form>
```

2. Adım: HomeController içinde Index Action metotlarını hem GET hem de POST işlemleri için aşağıdaki gibi düzenleyiniz.

```
[HttpGet]
public IActionResult Index()
{
  return View();
}

[HttpPost]
public IActionResult Index(string okulno,string ad,string soyad,string sinif)
{
  return Content(okulno+"-"+ad+"-"+soyad+"-"+sinif);
}
```

3. Adım: Uygulamayı çalıştırarak formdan gönderilen verilerin sunucu tarafından alındığını gözlemleyiniz.

4. Uygulama

IFormCollection sınıfı kullanılarak formdan gelen verileri alma işlemini yönergeler doğrultusunda gerçekleştiriniz.

1. Adım: IFormCollection sınıfını kullanabilmek için aşağıdaki isim uzayını HomeController içine ekleyiniz.

```
using Microsoft.AspNetCore.Http;
```

2. Adım: HttpPost özelliği olan Index Action metodu için aşağıdaki kodlamaları yapınız.

```
[HttpPost]
public IActionResult Index(IFormCollection collection)
{
   string okulno = collection["okulno"];
   string ad = collection["ad"];
   string soyad = collection["soyad"];
   string sinif = collection["sinif"];

   return Content(okulno + "-"+ ad + "-" + soyad + "-" + sinif);
}
```

3. Adım: Uygulamayı çalıştırarak formdan gönderilen verilerin sunucu tarafından alındığını gözlemleyiniz.

Model Binding; formdaki input elemanların name değerleri aynı olan bir modelle eşleştirme veya bağlama işlemidir. Oluşturulan model sınıfının özellik isimleri ile input elemanlarının name değerleri aynı olmalıdır. Model sınıfından oluşturulan nesne ile formdan gönderilen verilere erişim sağlanarak arka uç tarafında işlenir.

5. Uygulama

POST işlemi yapılan form verileri, model binding yöntemi ile controller içinden erişim yapılması işlemini yönergeler doğrultusunda gerçekleştiriniz.

1. Adım: Models klasörü içine **Ogrenci** adında model oluşturarak aşağıda verilen özellikleri ekleyiniz.

```
public class Ogrenci
{
   public int OkulNo { get; set; }
   public string OgrenciAdi { get; set; }
   public string OgrenciSoyadi{ get; set; }
   public string Sinifi { get; set; }
}
```

2. Adım: Views/Home klasöründeki Index.cshtml dosyası içindeki input elemanlarının name değerlerini Ogrenci model sınıfının özellik isimleri ile aynı olacak şekilde düzenleyiniz.

```
<form method="POST" action="/Home/Index">
  <div class="form-group">
    <label>Okul No</label>
    <input type="text" name="OkulNo" class="form-control"></input>
  </div>
  <div class="form-group">
    <label>Ad</label>
    <input type="text" name="OgrenciAdi" class="form-control"></input>
  </div>
  <div class="form-group">
    <label>Soyad</label>
    <input type="text" name="OgrenciSoyadi" class="form-control"></input>
  </div>
  <div class="form-group">
    <label>Sınıf</label>
     <input type="text" name="Sinifi" class="form-control"></input>
  </div>
  <div class="form-group">
    <button type="submit" class="btn btn-primary">Gönder</button>
  </div>
</form>
```

3. Adım: HomeController içinde HttpPost olan Index action metodunu aşağıdaki gibi düzenleyiniz.

```
[HttpPost]
public IActionResult Index(Ogrenci ogrenci)
{
  int okulno=int.Parse(ogrenci.OkulNo);
  string ad=ogrenci.OgrenciAdi;
  string soyad=ogrenci.OgrenciSoyadi;
   string sinif=ogrenci.Sinifi;

   return Content(okulno + "-" + ad + "-" + soyad + "-" + sinif);
}
```

4. Adım: Uygulamayı çalıştırarak formdan gönderilen verilerin sunucu tarafından alındığını gözlemleyiniz.

Sıra Sizde

Uygulama 5'teki form içindeki input elemanlarının name değerlerini küçük harflerle değiştiriniz. Değişim sonrası formdan verilerin gelip gelmediğini kontrol ediniz. Sonuçları sınıf arkadaşlarınızla paylaşınız.

6.6.3. Form ile Dosya Yükleme

ASP.NET Core uygulamalarında bir form ile dosya yükleme (file upload) yapmak için formun **enctype** özelliği **multipart/form-data** olarak ayarlamak gerekir. Bu özellik ayarlandığında dosya içeriği form verileriyle birlikte sunucuya gönderilecektir.

ASP.NET Core, form ile gönderilen bir dosyayı sunucu tarafın da alabilmesi için **IFormFile** arayüzünü kullanır. Bu arayüz ile gönderilen dosyanın; adı, boyutu, içerik tipi vb. bilgilerine ulaşılacağı gibi gönderilen dosyayı sunucuya kaydetme işlemi de gerçekleştirilir.

6. Uygulama

Form verileri ile birlikte sunucuya dosya gönderme işlemini yönergeler doğrultusunda gerçekleştiriniz.

1. Adım: FormUygulama projesi Views/Home klasöründeki Index view dosyasına aşağıdaki kodlamaları yapınız.

```
<form method="POST" action="/Home/Index" enctype="multipart/form-data">
  <div class="form-group">
    <label>Okul No</label>
    <input type="text" name="OkulNo" class="form-control"></input>
  </div>
  <div class="form-group">
    <label>Ad</label>
    <input type="text" name="OgrenciAdi" class="form-control"></input>
  </div>
   <div class="form-group">
    <label>Soyad</label>
    <input type="text” name="OgrenciSoyadi" class="form-control"></input>
  </div>
  <div class="form-group">
    <label>Sınıf</label>
    <input type="text" name="Sinifi" class="form-control"></input>
  </div>
   <div class="form-group">
    <label>Öğrenci Resmi</label>
    <input type="file" name="Resim" class="form-control"></input>
  </div>
  <div class="form-group">
    <button type="submit" class="btn btn-primary">Gönder</button>
  </div>
</form>
```

2. Adım: Formdan gönderilen verileri ve dosyayı sunucu tarafın da alması için HomeController dosyasına aşağıdaki komutları yazınız.

Not

Dosya işlemleri için **System.IO** isim uzayını ekleyiniz.

```
[HttpPost]
public IActionResult Index(Ogrenci ogrenci, IFormFile Resim)
{
  if (Resim == null || Resim.Length == 0)
     return Content("Resim yüklenemedi.");

  var yol=Path.Combine(Directory.GetCurrentDirectory(),"wwwroot",Resim.FileName);
  Resim.CopyTo(new FileStream(yol,FileMode.Create));
  return Content("Dosya yükleme tamamlandı.");
}
```

3. Adım: Formda gönderilen verileri ve dosyayı sunucu tarafından almanın diğer bir yolu model sınıfında **IFormFile** özelliğini kullanmaktır. Aşağıdaki kodları Ogrenci model sınıfı içerisine yazınız.

Not

IFormFile türü için Microsoft.AspNetCore.Http kütüphanesini ekleyiniz.

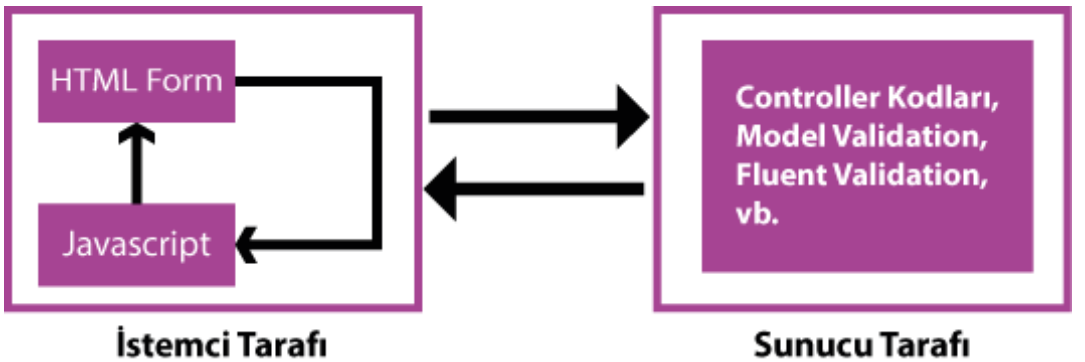
```
public class Ogrenci
{
   public int OkulNo { get; set; }
   public string OgrenciAdi { get; set; }
   public string OgrenciSoyadi{ get; set; }
   public string Sinifi { get; set; }
   public IFormFile Resim { get; set; }
}
```

4. Adım: Home Controller dosyasına aşağıdaki komutları yazınız.

```
 [HttpPost]
public IActionResult Index(Ogrenci ogrenci)
 {
   if (ogrenci.Resim == null || ogrenci.Resim.Length == 0)
     return Content("Dosya yüklenemedi.");
    var yol=Path.Combine(Directory.GetCurrentDirectory(),"wwwroot",ogrenci.Resim.FileName);
   ogrenci.Resim.CopyTo(new FileStream(yol,FileMode.Create));
   return Content("Dosya yükleme tamamlandı.");
 }
```

6.6.4. Doğrulama İşlemleri (Validation)

Bilgisayar alanında **doğrulama**, bir verinin önceden belirlenmiş gerekliliklere uygunluk durumunun denetlenmesidir. ASP.NET Core MVC uygulamalarında veriler formlar aracılığı ile Input elemanlarından gelir. Input elemanlarına girilen verilerin doğrulanması hem istemci tarafında (client-side) hem de sunucu tarafında (server-side) yapılabilir. İstemci taraflı doğrulama Javascript kodları ile gerçekleştirilir. Server taraflı doğrulama ise programsal kodlamalarla veya model validation işlemleri ile gerçekleştirilir. Doğrulama sonucunda girilen hatalı veriler için kullanıcı bilgilendirme mesajları ile uyarılmalıdır. Şekil 6.1'de istemci ve sunucu taraflı doğrulamanın döngüsü gösterilmiştir.



Şekil 6.1: İstemci ve sunucu taraflı doğrulama döngüsü

İstemci taraflı doğrulamanın önemi şunlardır:

- Daha iyi bir kullanıcı etkileşimi sağlar.
- Doğrulama, istemcinin tarayıcısında gerçekleştiği için daha hızlıdır.
- Veriler doğrulama için sunucuya gönderilmediği için sunucuyu iş yükünden kurtarır ve bant genişliğinden tasarruf sağlar.

Sunucu taraflı doğrulamanın önemi şunlardır:

- Javascript, kullanıcının tarayıcısında devre dışı bırakılabilir.
- Kötü niyetli kullanıcı tarafından, uygulamayı kullanmadan veriler doğrudan sunucuya gönderilebilir veya gönderilen verileri değiştiren bazı engelleyiciler kullanılabilir.
- Hatalı veriler geçersiz olsa dahi sunucuya gönderilebilir.

İstemci tarafında doğrulama yapılsa dahi mutlaka sunucu tarafında da doğrulama yapılması gerekliliği unutulmamalıdır.

ASP.NET Core ile sunucu taraflı doğrulamada temelde iki yöntem kullanılır. Birinci yöntem; veriler, controller içerisindeki action metotlara gönderildiğinde programsal olarak doğrulama işlemi gerçekleştirilebilir. Kodlarda sunucuya gönderilen verilerde programsal olarak doğrulama işlemi gerçekleştirilmiştir.

```
public IActionResult Ekle(Ogrenci ogrenci)
{

   if (ogrenci.OkulNo < 0)
    {
        return Content("Geçerli bir okul numarası giriniz.");
    }

    if (string.IsNullOrEmpty(ogrenci.OgrenciAdi))
    {
        return Content("Öğrenci adı boş bırakılamaz.");
    }
    return View();
}
```

Yukarıdaki kodlarda Ogrenci modeline ait bazı özelliklerin doğrulaması yapışmıştır. Uygulamada birden fazla yerde, yukardaki kodlarda oldu gibi, doğrulama yapılmak istenildiğinde veya bir modelin birden fazla özelliğinin kontrol edilmesi gerektiğinde programsal doğrulama pratikte kullanışlı sayılmaz.

İkinci yöntem, ASP.NET Core **Model Doğrulama (Model Validation)** adında daha pratik ve kullanışlı bir yapı sunar.

6.6.5. Model Doğrulama (Model Validation)

Model doğrulamada, formdan gelen veriler model bağlama (model binding) işleminden sonra çalışır. Bağlanan modelin içindeki özelliklere, doğrulama kuralları eklenebilecek birden çok doğrulama niteliği bulunur. ASP.NET Core içerisinde birden çok hazır doğrulama nitelikleri bulunur. Hazır doğrulama nitelikleri modelin özelliklerine eklenerek doğrulama kuralı oluşturulur. Bu doğrulama nitelikleri **DataAnnotations** olarak adlandırılır ve **System.ComponentModel.DataAnnotations** isim uzayı içerisinde yer alır.

ASP.NET Core projelerinde kullanılan doğrulama nitelikleri Tablo 6.8'de verilmiştir.

Tablo 6.8: Model Doğrulama Nitelikleri (Model Validation Attributes)

Nitelik	Açıklama
[Required]	Model özelliğinin boş olmaması gerektiğini belirtir.
[StringLength]	Model özelliği değerinin karakter uzunluğunu belirtir.
[Range]	Model özelliğinin sayısal aralık değerini belirtir.
[EmailAddress]	Model özeliğinin değerini bir e-posta biçimine sahip olduğunu doğrular.
[Compare]	Modelin iki özelliğinin aynı olduğunun doğrulmasını yapar.
[RegularExpression]	Model özeliğinin değerini belirlenen bir formatta olması gerektiğini belirtir.
[Phone]	Model özeliğinin değerini telefon numarası biçimine sahip olduğunu doğrular.
[CreditCard]	Model özeliğinin değerini kredi kartı biçimine sahip olduğunu doğrular.
[MaxLength]	Model özeliğinin değerini maksimum karakter uzunluğunu belirtir.
[MinLength]	Model özeliğinin değerini minimum karakter uzunluğunu belirtir.
[Url]	Model özeliğinin değerini URL biçimine sahip olduğunu doğrular.

Gönderilen verilerde, model binding işleminden sonra doğru verinin girildiğinin denetlenmesi için **ModelState** nesnesi kullanılır. ModelState nesnesinin **IsValid** özelliği **true** ise doğrulama işlemi başarılı **false** ise doğrulanmamış veriler bulunduğu anlamına gelir.

7. Uygulama

Formdan gelen verilere, model binding işleminden sonra doğru veri girildiğinin kontrol edilmesi işlemini yönergeler doğrultusunda gerçekleştiriniz.

1. Adım: Bilgisayarınıza **ModelValidation** adında ASP.NET Core MVC projesi oluşturunuz.

2. Adım: Models klasörünün içine **Kullanici** adında bir model sınıfı oluşturunuz. Oluşturulan model sınıfına aşağıdaki kodlamaları yapınız.

```
using System.ComponentModel.DataAnnotations;

namespace ModelValidation.Models
{
   public class Kullanici
   {
      [Required(ErrorMessage = "Ad alanı boş bırakılamaz.")]
      [StringLength(50,ErrorMessage = "50 karakterden fazla girilmez.")]
      public string Ad { get; set; }

     [Required(ErrorMessage = "Soyad alanı boş bırakılamaz.")]
    [StringLength(50,ErrorMessage = " 3 ile 50arası karakter giriniz. ", MinimumLength =
3)]
     public string Soyad { get; set; }
     [Required(ErrorMessage ="Eposta adresi boş bırakılamaz.")]
     [EmailAddress(ErrorMessage  = "Geçerli eposta adresi giriniz.")]
     public string Eposta { get; set; }
     [Phone(ErrorMessage = "Geçerli telefon numarası giriniz.")]
     public string Telefon { get; set; }
     [Required(ErrorMessage = "Şifre alanı boş bırakılamaz.")]
    [RegularExpression(@"^(?=.*[A-Za-z])(?=.*\d)[A-Za-z\d]{8,}$", ErrorMessage = "Harf
ve sayıdan oluşan 8 karakter giriniz.")]
     public string Sifre { get; set; }

    [Compare("Sifre", ErrorMessage = "Şifreler aynı değil.")]
    public string SifreTekrar { get; set; }

   }
}
```

3. Adım: Controllers klasörü içine **KullaniciController** adında bir controller oluşturunuz. Oluşturulan controller dosyasına aşağıdaki kodlamaları yapınız.

```
using Microsoft.AspNetCore.Mvc;
namespace ModelValidation.Controllers
{
   public class KullaniciController:Controller
   {
      [HttpGet]
      public IActionResult Ekle()
      {
         return View();
      }
   }
}
```

4. Adım: Views klasörüne **Kullanici** adında klasör oluşturunuz. Oluşturulan klasörün içine **Ekle.cshtml** adında view dosyası oluşturarak içine aşağıdaki kodlamaları yapınız.

```
<form action="/Kullanici/Ekle" method="POST">
   <div class="form-group">
      <label>Ad</label>
      <input name="Ad" class="form-control"></input>
   </div>
   <div class="form-group">
      <label>Soyad</label>
      <input name="Soyad"class="form-control"></input>
   </div>
   <div class="form-group">
      <label>E-posta</label>
      <input name="Eposta" class="form-control"></input>
   </div>
   <div class="form-group">
      <label>Telefon</label>
      <input name="Telefon" class="form-control"></input>
   </div>
   <div class="form-group">
      <label>Sifre</label>
      <input name="Sifre" class="form-control"></input>
   </div>
   <div class="form-group">
      <label>SifreTekrar</label>
      <input name="SifreTekrar" class="form-control"></input>
   </div>
   <div class="form-group">
      <button type="submit" class="btn btn-primary">Gönder</button>
   </div>
</form>
```

5. Adım: Form sunucuya gönderildiğinde **ModelState** nesnesinin **IsValid** özelliğini kullanarak doğrulama işlemi kontrolü sağlayan kodlamaları yapınız.

```
[HttpPost]
public IActionResult Ekle(Kullanici kullanici)
{
   if (ModelState.IsValid == true)
   {
      return Content("Doğrulama işlemi başarılı.");
   }
   else
   {
      return Content("Doğrulama işlemi başarısız.");
   }
}
```

Doğrulama işleminin sonucunda, doğrulanmamış alanlarda oluşan hata mesajlarını göstermek için **@Html.ValidationSummary()** HTML yardımcı metodu kullanılır.

Not

HTML yardımcı metotları yerine daha kullanışlı olan **etiket yardımcı metotları** bir sonraki konuda işlenecektir.

8. Uygulama

Model doğrulamada, hata mesajlarını view sayfasında gösterme işlemini yönergeler doğrultusunda gerçekleştiriniz.

1. Adım: Views/Kullanici klasöründeki Ekle.cshtml dosyasında, hata mesajlarını göstermek istediğiniz yere **@Html.ValidationSummary()** yardımcı metodu yazınız.

```
@Html.ValidationSummary()
<form action="/Kullanici/Ekle" method="POST">
```

2. Adım: Doğrulama mesajlarını Ekle.cshtml sayfasında göstermek için KullaniciController dosyasındaisimli POST action metodunu aşağıdaki gibi düzenleyiniz.

```
[HttpPost]
public IActionResult Ekle(Kullanici kullanici)
{
   if (ModelState.IsValid == true)
   {
      return Content("Doğrulama işlemi başarılı.");
   }
   else
   {
      return View(kullanici);
   }
}
```

3. Adım: Uygulamayı çalıştırarak sonuçları gözlemleyiniz.

6.6.6. İstemci Taraflı Doğrulama (Client Side Validation)

Model Validation, sunucu tarafında gerçekleşir ve doğrulanacak veriler sunucuya gider. Kontrol edildikten sonra hata varsa tekrar istemciye hata mesajları gönderilir. Bu gidip gelme işlemleri bir gecikme meydana getirir. Gecikme, kullanıcı deneyimi açısından tercih edilen bir durum değildir. Ayrıca basit bir doğrulama için sunucu kaynakları boşa harcanmış olur. Bu durumun çözümü için kullanıcı taraflı doğrulama kullanılmalıdır.

Not

Kullanıcı taraflı doğrulama sadece kullanıcı deneyimi ve performans için kullanılır. Asla sunucu taraflı doğrulamanın yerini almamalıdır.

ASP.NET Core uygulaması MVC şablonu ile oluşturulduğunda kullanıcı taraflı doğrulamada kullanılacak Javascript kütüphaneleri, otomatik olarak gelir. Bu kütüphaneler **wwwroot/lib** klasörü içindeki **jquery, jquery-validation, jquery-validation-unobtrusive** isimli kütüphanelerdir. Bir view sayfasında kullanıcı taraflı doğrulama işlemi yapmak için MVC şablonu ile birlikte oluşturulan **Views/Shared** klasörü içerisinde yer alan **_ValidationScriptsPartial.cshtml** dosyasını view sayfasının altına eklemek yeterlidir.

6.7. Etiket Yardımcıları (Tag Helper)

Tage Helper 'lar içinde bulunan metotlar sayesinde sunucuda çalışan kodları view sayfalarında da çalıştırılabilme olanağı sağlayan yardımcı metotlardır. Tag Helper'lar sunucu taraflı kodların HTML etiketleri gibi kullanılmasını sağlar. Helper'lar uygulamanın view tarafında daha kolay geliştirilebilmesine olanak sağlar. ASP.NET Core uygulamalarında helperların **UrlHelper, HtmlHelper** ve **TagHelper** olmak üzere üç kategorisi bulunur. ASP.NET Core uygulamalarında Tag Helper'lar, HTML etiket yapılarına benzediği için öğrenilmesinin ve kullanımının kolaylığı bakımından daha çok tercih edilir. Bu öğrenme biriminde Tag Helper kullanımı anlatılacaktır.

UrlHelper ve HtmlHelper ASP.NET framework içerisinde bulunurken Tag Helper, ASP.NET Core ile birlikte gelir.

Tag Helper kullanmanın avantajları şunlardır:

- Tag Helper'lar HTML etiketlerine benzerlikleri nedeniyle öğrenilmesi ve kullanılması kolaydır.
- Kodlama yaparken IntelliSense (otomatik tamamlama) özelliği kullanılabilir.
- Temiz ve okunabilir kodlama yapılabilir.
- Hazır (built-in) Tag Helper kullanımının yanında kişisel (custom) Tag Helper'lar oluşturabilir.

Tag Helper'lar Microsoft.AspNetCore.Mvc.TagHelpers kütüphanesinin içerisinde bulunur. View sayfalarında Tag Helper kullanılmak istenildiğinde @addTagHelper komutu ile birlikte kütüphanenin yüklenmesi gerekir.

```
@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
```

Not

Yukarıdaki kodda kullanılan ("*") karakteri kütüphane içerisinde tüm hazır Tag Helper'ların yükleneceğini ifade eder. Her View sayfası için yukarıdaki kodu eklemek yerine **_ViewImports.cshtml** dosyasına bir kere yazılarak tüm View'ler için kullanılabilir hâle getirilebilir.

Belirli View'ler için Tag Helper kullanılmak istenmediğinde aşağıdaki kodlama yapılarak Tag Helper'ları kaldırma işlemi gerçekleştirilebilir.

```
@removeTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
```

6.7.1. Input Tag Helper Kullanımı

Input Tag Helper, HTML elemanlarından **Input** etiketine uygulanan Tag Helper'dır. Input Tag Helper'lar, HTML input etiketlerine **asp-for** niteliği ile uygulanır. Bir modelin özelliği ile ilişkilendirilir. İlişkilendirilen modelin özelliğinin adına, tipine ve ek açıklamalarına (annotations) göre bir HTML çıktısı üretilir.

9. Uygulama

Input Tag Helper kullanım işlemini yönergeler doğrultusunda gerçekleştiriniz.

1. Adım: TagHelperUygulama adında MVC projesi oluşturunuz.
2. Adım: Models klasörü içine **Kullanici** adında bir model dosyası oluşturunuz. Oluşturulan model dosyası içerisine aşağıdaki kodlamaları yapınız.

```
public class Kullanici
{
  public string Eposta { get; set; }
}
```

3. Adım: Controllers klasörüne **KullaniciContoller** adında bir controller ekleyiniz.
4. Adım: KullaniciController içine **Ekle** adında GET Action metodu oluşturunuz.

```
[HttpGet]
public IActionResult Ekle()
{
  return View();
}
```

5. Adım: Views klasörü içerisine **Kullanici** adından bir klasör oluşturunuz. Kullanici klasörünün içerisine **Ekle.cshtml** adında bir View dosyası oluşturunuz.
6. Adım: Ekle View dosyasına aşağıdaki kodlamaları yazınız. Uygulamayı çalıştırarak HTML çıktısını inceleyiniz.

```
@model TagHelperUygulama.Models.Kullanici
<input asp-for="Eposta" />
```

7. Adım: Kullanici modeline Veri Açıklamaları (Data Annotations) ekleyerek aşağıdaki gibi düzenleyiniz.

```
using System;
using System.ComponentModel.DataAnnotations;

public class Kullanici
{
    [EmailAddress]
    public string EPosta { get; set; }

    [DataType(DataType.DateTime)]
    public DateTime DogumTarihi { get; set; }

    [DataType(DataType.PhoneNumber)]
    public string Telefon { get; set; }

    [DataType(DataType.PostalCode)]
    public string PostaKodu { get; set; }

    [DataType(DataType.Password)]
    public string Sifre { get; set; }
}
```

8. Adım: Ekle View dosyasına aşağıdaki kodlamaları yazınız.

```
@model TagHelperUygulama.Models.Kullanici
<input asp-for="EPosta"><br>
<input asp-for="DogumTarihi"><br>
<input asp-for="Telefon"><br>
<input asp-for="PostaKodu"><br>
<input asp-for="Sifre"><br>
```

9. Adım: Uygulamayı çalıştırarak HTML çıktısını inceleyiniz.

6.7.2. Label Tag Helper Kullanımı

HTML elemanlarından Label etiketine uygulanan Tag Helper'dır. Label etiketleri **asp-for** niteliği ile uygulamadaki bir modelin özelliği ile ilişki kurulması için kullanılır. İlişki kurulan model özelliğinin **Display** açıklamaları label etiketinde otomatik olarak oluşturulur.

10. Uygulama

Label Tag Helper kullanım işlemini yönergeler doğrultusunda gerçekleştiriniz.

1. Adım: TagHelperUygulama isimli ASP.NET Core projesini açınız.
2. Adım: Kullanici modelini aşağıdaki gibi düzenleyiniz.

```
public class Kullanici
{
    [Display(Name = "Ad Soyad")]
    public string AdSoyad { get; set; }

    [EmailAddress]
    [Display(Name = "Eposta Adresi")]
    public string EPosta { get; set; }

    [DataType(DataType.Date)]
    [Display(Name = "Doğum Tarihi")]
    public DateTime DogumTarihi { get; set; }

    [DataType(DataType.PhoneNumber)]
    [Display(Name = "Telefon Numarası")]
    public string Telefon { get; set; }

    [DataType(DataType.PostalCode)]
    [Display(Name = "Posta Kodu")]
    public string PostaKodu { get; set; }

    [DataType(DataType.Password)]
    [Display(Name = "Şifre")]
    public string Sifre { get; set; }
 }
```

3. Adım: View/Kullanici klasörü içindeki **Ekle.cshtml** dosyasını aşağıdaki gibi düzenleyiniz.

```
@model TagHelperUygulama.Models.Kullanici
<div class="form-group">
  <label asp-for="AdSoyad"></label>
  <input asp-for="AdSoyad" class="form-control">
</div>
<div class="form-group">
  <label asp-for="EPosta"></label>
  <input asp-for="EPosta" class="form-control">
</div>
<div class="form-group">
  <label asp-for="DogumTarihi"></label>
  <input asp-for="DogumTarihi" class="form-control">
</div>
<div class="form-group">
  <label asp-for="Telefon"></label>
  <input asp-for="Telefon" class="form-control">
</div>
<div class="form-group">
  <label asp-for="PostaKodu"></label>
  <input asp-for="PostaKodu" class="form-control">
</div>
<div class="form-group">
  <label asp-for="Sifre"></label>
  <input asp-for="Sifre" class="form-control">
</div>
```

4. Adım: Uygulamayı çalıştırarak oluşturulan HTML çıktısını inceleyiniz.

6.7.3. Form Tag Helper Kullanımı

HTML elemanlarından Form etiketine uygulanan Tag Helper'dır. HTML Form etiketi içinde kullanılan bu Helper'lar, Form etiketine eklenen niteliklerle işlevsel kullanımını sağlar.

Bu niteliklerden bazıları şunlardır:

• **asp-controller:** Form verilerinin gideceği Controller adını belirtir. Belirtilmediği takdirde View'in oluşturulduğu controller kullanılır.

• **asp-action:** Form verilerinin gideceği Action metodunu belirtir. Belirtilmediği takdirde View'in oluşturulduğu Action kullanılır.

• **asp-route:** Kullanılacak olan routing adını belirtir.

• **asp-route-*:** Routing yapılanmasında * karakteri ile parametre adı belirtilir.

• **asp-antiforgery:** CSRF (Cross-Site Request Forgery) saldırılarına karşı Anti Forgery Token oluşturmak için kullanılır.

Form Tag Helper sayesinde kötü niyetli kullanıcılara karşı CSRF'i (Cross-Site Request Forgery - Siteler Arası İstek Sahtekârlığı) önlemek için formda gizli bir Request Verification Token oluşturur.

11. Uygulama

Form Tag Helper kullanım işlemlerini yönergeler doğrultusunda gerçekleştiriniz.

1. Adım: TagHelperUygulama ASP.NET Core projesini açınız.
2. Adım: Controllers klasörüne **KitapContoller** adında bir controller ekleyiniz.
3. Adım: KitapController içine Ekle adında GET özelliği olan bir Action metot oluşturunuz.

```
[HttpGet]
public IActionResult Ekle()
{
  return View();
}
```

4. Adım: Views klasörünün içine Kitap adında klasör oluşturunuz. Kitap klasörünün içerisine **Ekle.cshtml** adında bir View dosyası oluşturunuz.
5. Adım: Ekle view dosyasına aşağıdaki kodlamaları yapınız.

```
<form  asp-controller="kitap" asp-action="ekle" method="POST">
  <input  name="kitap">
  <button type="submit">Gönder</button>
</form>
```

6. Adım: KitapController içine Ekle adında POST özelliğine sahip bir Action metot oluşturunuz.

```
[HttpPost]
 [ValidateAntiForgeryToken]
 public IActionResult Ekle(string kitap)
 {
   return Content(kitap);
 }
```

7. Adım: Uygulamayı çalıştırınız ve oluşturulan HTML çıktısını inceleyiniz.
8. Adım: Sayfadaki input elemanı içerisine herhangi bir kitap adı yazarak gönder butonuna tıklayınız.
9. Adım: Tarayıcı uygulamasında /kitap/ekle sayfasının kaynak kodlarını açarak form içerisindeki **__RequestVerificationToken** değerlerini değiştiriniz. Gönder butonu ile input içerisine yazılan kitap bilgilerini gönderme işlemini gerçekleştirmeyi deneyiniz.

Sıra Sizde

Tarayıcı üzerinden **__RequestVerificationToken** değeri değiştirildikten sonra gönder butonuna tıklandığında nasıl bir sonuç ile karşılaştınız? Sınıf arkadaşlarınızla paylaşınız.

12. Uygulama

Form etiketine Form Tag Helper ekleme işlemini yönergeler doğrultusunda gerçekleştiriniz.

1. Adım: Ekle View dosyasının içine aşağıdaki kodlamaları yapınız. HTML çıktısını inceleyiniz.

```
<form asp-controller="kitap" asp-action="ekle" asp-route-id="10">
</form>
```

2. Adım: ASP.NET Core Routing yapılandırmasını **{controller}/{action}/{id?}/{val?}** olarak değiştiriniz. Aşağıdaki kodlamaları Ekle view dosyasına yazarak HTML çıktısını inceleyiniz.

```
<form asp-controller="kitap" asp-action="ekle" asp-route-id="10" asp-route-val="roman">
</form>
```

3. Adım: ASP.NET Core Routing yapılandırmasını **{controller=Home}/{action=Index}** olarak değiştiriniz. Aşağıdaki kodlamaları Ekle view dosyasına yazarak HTML çıktısını inceleyiniz.

```
<form asp-controller="kitap" asp-action="ekle" asp-route-id="10">
</form>
```

4. Adım: ASP.NET Core Routing yapılandırmasını **{controller}/{action}/{id?}** olarak değiştiriniz. Aşağıdaki kodlamaları Ekle view dosyasına yazarak HTML çıktısını inceleyiniz.

```
@{
  var data= new Dictionary<string, string>
  {
    { "Id", "12" },
    { "tur", "roman" }
  };
}

<form asp-controller="kitap" asp-action="ekle"  asp-all-route-data="data">
</form>
```

5. Adım: Ekle View dosyasının içine aşağıdaki kodlamaları yapınız. HTML çıktısını inceleyiniz.

```
<form asp-antiforgery="false" asp-controller="kitap" asp-action="ekle">
</form>
```

6.7.4. Validation Tag Helper Kullanımı

Validation Tag Helper, ASP.NET Core uygulamalarında, model doğrulama işlemi sonucunda doğrulama mesajlarını view sayfalarında görüntülenmesi sağlayan etiket yardımcılarıdır. ASP.NET Core doğrulama mesajlarını görüntülemek için iki etiket yardımcısı kullanır.

- Doğrulama Mesaj Etiket Yardımcısı (Validation Message Tag Helper)
- Doğrulama Özet Etiket Yardımcısı (Validation Summary Tag Helper)

Validation Message Tag Helper, asp-validation-for niteliği ile view sayfalarında **** etiketine uygulanır ve modelin bir özelliğinde belirtilen hata mesajını gösterir.

```
<label asp-for="Ad"></label>
<input asp-for="Ad" class="form-control"></input>
<span asp-validation-for="Ad"></span>
```

Doğrulama mesajının sitilini değiştirmek için aşağıdaki sınıf seçicisi kullanılır.

```
<style>
  .field-validation-error { color:red }
</style>
```

Validation Summary Tag Helper, **asp-validation-summary** niteliği ile view sayfalarında ** ** etiketine uygulanır ve tüm hata mesajları toplu hâlde sırasız liste olarak gösterilir.

```
<form asp-controller="Kullanici" asp-action="Ekle" method="POST">
  <div asp-validation-summary="All"></div>
```

Doğrulama mesajının sitilini değiştirmek için aşağıdaki sınıf seçicisi kullanılır.

```
<style>
  .validation-summary-error { color:red }
</style>
```

Sıra Sizde

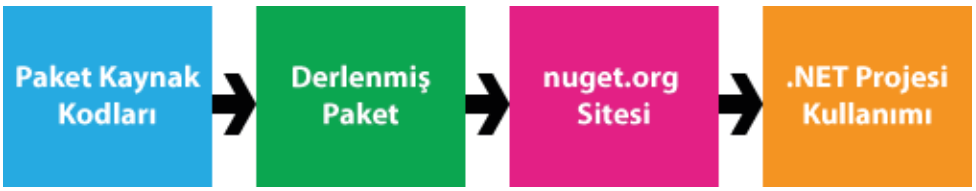
Validation Tag Helper'ları kullanarak istemci taraflı (client-side) doğrulama işlemini gerçekleştiriniz.

6.8. Paket Yöneticisi (NuGet)

Programcıların uygulama geliştirirken ihtiyaç duyacakları her kod parçası ve kütüphanenin, destekleyici firma tarafından sunulması mümkün değildir. Dünya genelindeki geliştiriciler, hazırladıkları kodları ve kütüphaneleri paylaşarak uygulama geliştirme sürecinin kısalmasında etkili olur.

NuGet, .NET'in resmî paket yöneticisidir. Paketler, kodların derlenerek kütüphane (DLL) hâline getirilmesiyle oluşur. NuGet, yazılım geliştiricilerinin yararlı kodlar oluşturabileceği, paylaşabileceği ve kullanabileceği bir araçtır. NuGet, açık kaynak ve ücretsiz bir platformdur. NuGet derlenmiş olan kodların versiyon numarası ile birlikte **nupkg** uzantılı sıkıştırılmış tek bir dosyadan oluşur.

Nuget paketlerinin oluşturulması, barındırılması ve kullanılmasının akışı Şekil 6.2'de gösterilmiştir.



Şekil 6.2: Nuget paketleri oluşturma ve kullanma şeması

6.8.1. Paket Yöneticisi Araçları

Visual Studio Code geliştirme ortamında NuGet kullanıcı arayüz penceresi bulunmaz. Visual Studio Code ile NuGet paketlerini yükleme, güncelleme ve kaldırma işlemleri için **Dotnet CLI** [Command Line Interface (Komut Satırı Arayüzü)] kullanılır. Visual Studio Code terminal paneline aşağıda belirtilen yapıda komut kullanılarak paket yükleme işlemi gerçekleştirilir.

```
> dotnet add package <Paket_Adı>
```

Komut içerisinde **Paket_Adı** yüklenmek istenen paketi belirtir. Paket adları **nuget.org** sitesinden bulunabilir.

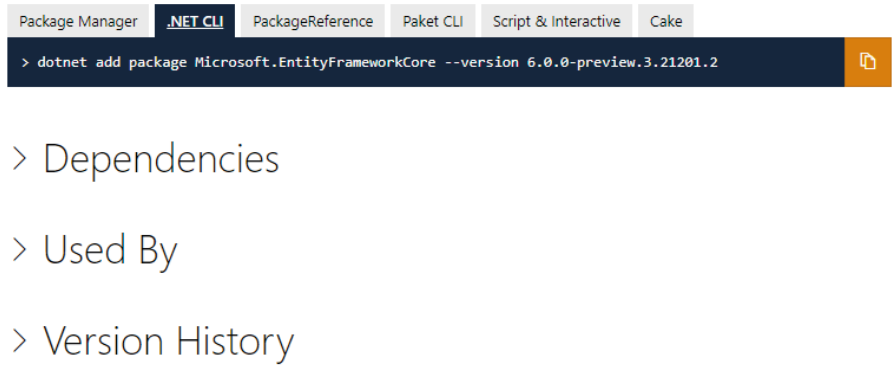
13. Uygulama

Visual Studio Code geliştirme ortamı için NuGet paketler yükleme, güncelleme ve kaldırma işlemlerini yönergeler doğrultusunda gerçekleştiriniz.

1. Adım: Visual Studio Code uygulamasını açınız. UygulamaNuget adında bir klasör oluşturarak Visual Studio Code uygulamasında bu klasöre geçiniz.
2. Adım: Terminal panelini açarak ve ASP.NET Core uygulaması için boş veya MVC şablonlarından birini oluşturunuz.
3. Adım: İnternet tarayıcı programını açarak **nuget.org** sitesine gidiniz.
4. Adım: Açılan sitede arama kutusuna yüklemek istediğiniz paketin adını yazarak arayınız. Bu kutuya örnek olarak "entity" kelimesini yazınız. Bulunan sonuçlar içerisinden yüklemek istediğimiz paket olarak **Microsoft.EntityFrameworkCore** paketini seçiniz.
5. Adım: Görsel 6.30'da .NET CLI sekmesinde yükleme komutları versiyon numarası ile birlikte verilmektedir. .NET CLI sekmesinde verilen komutu kopyalayınız.

Not

Versiyon numarası, öğrenme biriminin hazırlandığı tarihteki sürüm numarasıdır. Versiyon seçeneği kullanılmadığı takdirde en son kararlı sürümü otomatik olarak yüklenecektir. . Dependencies kısmından kurulacak olan paketin bağımlıklarını görebilirsiniz.



Görsel 6.30: Nuget.org komut satırı paket yükleme

6. Adım: Terminal panelini açarak kopyalanan komutu aşağıdaki gibi yapıştırınız.

```
> dotnet add package Microsoft.EntityFrameworkCore
```

7. Adım: Proje içindeki U**ygulamaNuget.csproj** isimli dosyayı açarak yüklenen paketin eklediğini gözlemleyiniz.

8. Adım: Yüklü olan paketleri güncellemek için terminal paneline aşağıdaki kodu yazınız.

```
> dotnet restore
```

9. Adım: Yüklü olan paketleri kaldırmak için terminal paneline aşağıdaki kodu yazınız.

```
> dotnet remove package Microsoft.EntityFrameworkCore
```

Sıra Sizde

- Visual Studio Code geliştirme ortamında NuGet paketleri eklemek ve yönetmek için bir Extensions olup olmadığını araştırınız. Bulduğunuz sonuçları sınıfta arkadaşlarınız ile paylaşınız.
- Küçük gruplar oluşturarak ASP.NET Core için en çok kullanılan paketlerden beş tanesini bulunuz. Bu paketlerin yaptıkları işler hakkında sunu hazırlayarak arkadaşlarınızla paylaşınız.

6.9. Entity Framework Core ile Veri Tabanı İşlemleri

Bilişim alanında yazılımların birçoğu verileri saklamak için veri tabanlarına ihtiyaç duyar. Örneğin, kütüphane uygulamasında kitap bilgilerinin kaydedilmesi, güncellenmesi, listelenmesi veya silinmesi gibi işlemlerde veri tabanı kullanılır.

Veri tabanı bağlantısı, verilerin okunması ve yazılması gibi işlemler için SQL sorguları ve kodlamalar kullanır. Kullanılan SQL sorguları ve kodlamalar uygulamalar geliştirirken yazılımcı için fazladan zaman harcamasına ve iş yüküne neden olur. Bu durumda Entity Framework Core gibi **Object Relational Mappers** (ORM) araçları kullanılır.

Entity Framework Core, C# dilinde SQL komutları kullanmadan veri tabanı üzerinde işlem yapılmasını sağlayan bir ORM aracıdır. **ORM,** ilişkisel veri tabanı ile nesneye yönelik programlama (OOP) arasında bir köprü görevi gören araçtır. Entity Framework Core sayesinde direkt veri tabanı şeması ile etkileşime geçmek yerine, .NET nesneleri kullanılarak veri tabanı üzerinde işlem yapılmasını sağlar. Kullanılan .NET nesneleri, **Entity** olarak adlandırılan basit sınıflardır.

Entity Framework Core, Şema 6.1'de belirtildiği gibi ASP.NET Core uygulaması ile veri tabanı arasında köprü görevini görür.

Şema 6.1: ASP.NET Core, EF Core ve veri tabanı ilişkisi

Geleneksel veri tabanı işlemleri gerçekleştirmek için SQL sorguları kullanılır. SQL sorguları direkt veri tabanı üzerinde çalıştırıldığı için performans avantajı sağlar. Ancak SQL sorgularında yapılan değişiklikler, uygulamadaki kodların değişmesi gibi nedenlerle bakımı zor olabilir. Ayrıca SQL sorgularını kullanırken yazım veya söz dizimi hatası çokça karşılaşılan problemlerdendir. Entity Framework Core'da SQL sorguları yerine daha basit ve anlaşılır LINQ (Language Integrated Query) sorguları kullanılır. **LINQ**, farklı veri kaynaklarındaki veriler ile işlem yapılmasını sağlayan ortak bir sorgu dilidir.

6.9.1. Entity Framework Core Kurulumu

Entity Framework Core kullanmak için **Microsoft.EntityFrameworkCore** paketinin kurulması gerekir. Bir Core kütüphanesi olan Microsoft.EntityFrameworkCore'u tek başına kurmak yeterli değildir. Kullanılacak veri tabanı sağlayıcıların kurulması gerekir. Örneğin, SQL Server veri tabanı kullanılacak ise Microsoft.EntityFrameworkCore.SqlServer; SQLite veri tabanı kullanılacak ise Microsoft.EntityFrameworkCore.Sqlite veri tabanı sağlayıcı paketlerinin kurulması gerekir. Veri tabanı sağlayıcıları (database provider) kurulduğunda Microsoft.EntityFrameworkCore otomatik olarak kurulur.

Aşağıda veri tabanı sağlayıcılarından bazıları verilmiştir:

- **MS SQL Server,** Microsoft.EntityFrameworkCore.SqlServer
- **SQLite,** Microsoft.EntityFrameworkCore.Sqlite
- **PostgreSQL,** Npgsql.EntityFrameworkCore.PostgreSQL
- **Mysql,** MySql.EntityFrameworkCore veya MySql.Data.EntityFrameworkCore
- **Mysql ve MariaDB,** Pomelo.EntityFrameworkCore.MySql
- **Oracle,** Oracle.EntityFrameworkCore
- **Microsoft Access**, EntityFrameworkCore.Jet

Entity Framework Core ile birlikte Entity Framework Core araçlarının da kurulmasına ihtiyaç duyulabilir. Bu araçlar, modellerde yapılan değişikliklerin veri tabanına yansıtılmasını sağlayan **migrations** işlemlerinde veya hazır bir veri tabanından model oluşturma işlemlerinde yazılımcıya yardımcı olacak araçlardır. Entity Framework Core araçları **Microsoft.EntityFrameworkCore.Tools** paketinin yüklenmesiyle projeye dâhil edilir.

14. Uygulama

"Kütüphane Takip Sistemi" oluşturmak için Entity Framework Core paketinin yüklemesi işlemlerini yönergeler doğrultusunda gerçekleştiriniz.

1. Adım: KutuphaneTakip adında ASP.NET Core MVC uygulaması oluşturunuz.
2. Adım: Oluşturulan uygulamada Microsoft SQL Server veri tabanı kullanılacağı için terminal paneline aşağıdaki kodlamaları yaparak Entity Framework Core veri tabanı sağlayıcı paketlerini projeye dâhil ediniz.

```
> dotnet add package Microsoft.EntityFrameworkCore.SqlServer
```

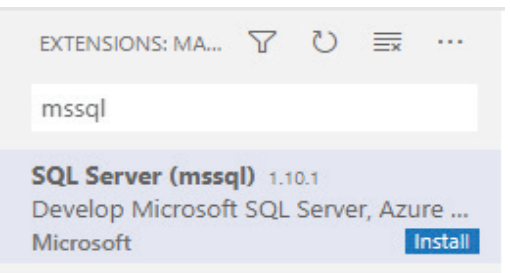
3. Adım: Entity Framework Core araçlarını yüklemek için terminal paneline aşağıdaki kodlamaları yaparak Entity Framework Core Tools paketini projeye dâhil ediniz.

```
> dotnet add package Microsoft.EntityFrameworkcore.Tools
```

4. Adım: SqlServer bağlantı ve sorguları çalıştırmak için Visual Code eklentilerinden **mssql** eklentisini Görsel 6.31'de gösterildiği gibi kurunuz.

Not

Farklı bir veri tabanı yönetim uygulaması da kullanılabilir.



Görsel 6.31: SQL Server eklentisi

Kullanılacak veri tabanı sağlayıcılarından herhangi biri seçilebilir. Veri tabanı olarak MS SQL Server, MySql, Oracle gibi büyük büyük boyutlarda, gelişmiş DBMS [Database Management System (Veri Tabanı Yönetim Sistemi)] kullanılabileceği gibi uygulama geliştirme aşamasında **SQL Express LocalDB** gibi küçük boyutlarda veri tabanları da kullanılabilir. Böylece uygulama geliştirme aşaması bittikten sonra projeyi yayına alma aşamasında daha gelişmiş veri tabanlarına geçiş yapılabilir.

15. Uygulama

SQL Express LocalDB kurulum işlemlerini yönergeler doğrultusunda gerçekleştiriniz.

1. Adım: Sql Express Installer uygulaması indirmek için aşağıda verilen adresi tarayıcı uygulamasında açınız. Açılan sayfada Express sürümünü indir butonuna tıklayarak indirme işlemini gerçekleştiriniz.

```
https://www.microsoft.com/tr-tr/sql-server/sql-server-downloads
```

2. Adım: İndirme işlemi tamamlandığında indirilen dosyayı çalıştırarak açılan pencereden **Download Media** butonuna tıklayınız.

3. Adım: Açılan pencereden **LocalDB** seçeneğini seçiniz ardından **Download** butonuna tıklayarak bilgisayarınıza kurulum dosyasını indirme işlemini gerçekleştiriniz.

4. Adım: İndirilen **SqlLocalDb** adındaki kurulum dosyasını açarak kurulum işlemini tamamlayınız.

5. Adım: Kurulum işlemi tamamlandığında kurulumu kontrol etmek için komut istemi (cmd) veya Windows PowerSell uygulamasını açarak aşağıdaki komutu yazınız.

```
> sqllocaldb info
```

6. Adım: Yukarıdaki komut çalıştırıldığında ekrana **MSSQLLocalDB** yazısı geliyorsa "kurulumun başarıyla tamamlandığı" anlamına gelir. Buradaki MSSQLLocalDB yazısı kurulan LocalDB'nin otomatik oluşturulan örnek adıdır (instance name). Veri tabanına bağlantı için bu örnek adı kullanılacaktır. Otomatik oluşturulan örnek ad yerine farklı bir örnek ad kullanılmak isteniyor ise aşağıdaki komut kullanılabilir.

```
> sqllocaldb create meb
```

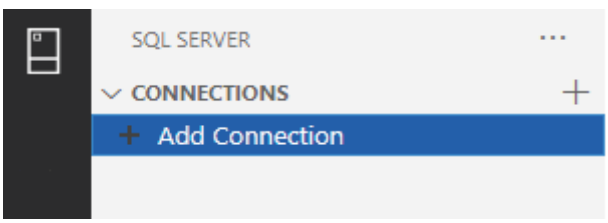
7. Adım: Veri tabanı sunucusunun başlatılması için aşağıdaki kodları yazınız.

```
> sqllocaldb start MSSQLLocalDB
```

8. Adım: Veri tabanı sunucusunun durumunun kontrolü için aşağıdaki kodları yazınız.

```
> sqllocaldb info MSSQLLocalDB
```

9. Adım: KutuphaneTakip uygulamasını açınız. Visual Studio Code kenar menüsünde Görsel 6.32'de belirtilen SQL Server butonuna tıklayınız. Açılan panelden **Add Connection** linkine tıklayınız.



Görsel 6.32: SQL Server bağlantı paneli

10. Adım: Açılan komut paneline aşağıdaki bağlantı komutunu yazınız. Enter tuşuna basarak bir sonraki işleme geçiş yapınız.

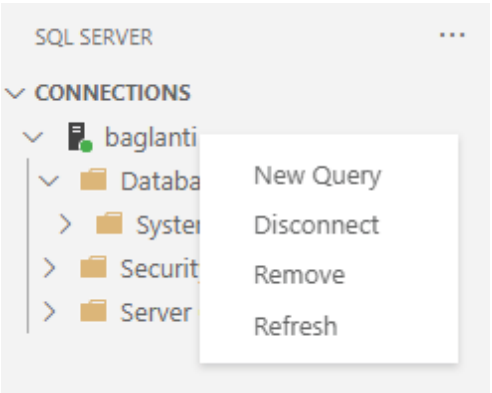
```
(localdb)\ mssqllocaldb
```

11. Adım: Komut panelinde belirli bir veri tabanına bağlanmak isteniyorsa veri tabanı adı yazılmalı veya tüm veri tabanları görüntülemek isteniyorsa bu kısım boş geçilmelidir.

12. Adım: Authentication Type (Kimlik Doğrulama Tipi) alanında **Integrated** bölümünü seçerek enter tuşuna basınız.

13. Adım: Veri tabanına bağlantı için kullanılacak profil adına **baglanti** ismini vererek bağlantı işlemlerini tamamlayınız.

14. Adım: Görsel 6.33'te gösterildiği gibi profil isminin üzerinde sağ tıklayarak açılan menüden sql sorgusu oluşturmak için **New Query** menü elemanına tıklayınız.



Görsel 6.33: Sql sorgu komutları oluşturma

15. Adım: Açılan pencerede veri tabanı oluşturma komutu olan **sqlCreateDatebase** komutu yazarak oluşturulan kod blokunda belirtilen yerlere oluşturulacak olan veri tabanının adını yazınız.

```
USE master
GO
IF NOT EXISTS (
   SELECT name
      FROM sys.databases
      WHERE name = N'KutuphaneDb'
)
CREATE DATABASE  KutuphaneDb
GO
```

16. Adım: Sağ üst köşede bulunan **Execute Query** (Ctrl + Shift+ E) butonuna tıklayarak veri tabanı oluşturacak olan kodları çalıştırınız. Sql Server paneli yenileyerek (refresh) oluşturulan veri tabanını görüntüleyiniz.

6.9.2. Entity Framework Core DbContext Sınıfı

DBContext sınıfı, Entity sınıflarını veya modelleri yönetir. Entity Framework Core'un kalbidir. Veri tabanı sağlayıcısını kullanarak veri tabanına bağlanmak için **DBContext** sınıfı kullanılır. DBContext veri tabanına bağlanmakla beraber entity model ile veri tabanı arasında köprü görevi gören önemli bir sınıftır. DBContext sınıfı entity modelleri kullanarak veri tabanı üzerinde sorgulama, ekleme, silme ve güncelleme işlemlerini gerçekleştirir. Veri tabanı ayarları için gerekli bilgileri bulundurur.

16. Uygulama

DbContext sınıfı oluşturarak veri tabanına bağlantı işlemlerini yönergeler doğrultusunda gerçekleştiriniz.

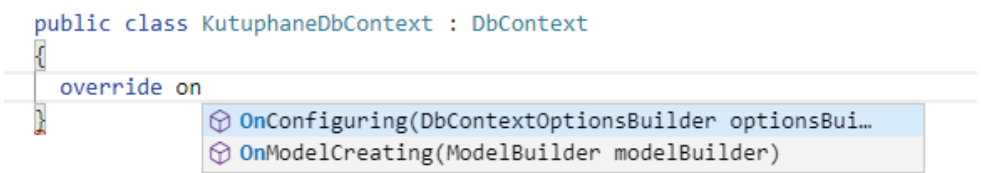
1. Adım: Models klasörünün içerisine Kitap adında model sınıfı oluşturarak sınıf içerisine aşağıdaki kodlamaları yapınız.

```
namespace KutuphaneTakip.Models
{
   public class Kitap
   {
      public int KitapId { get; set; }
      public string  KitapAdi { get; set; }
      public string KitapYazar { get; set; }
      public string KitapTur { get; set; }
   }
}
```

2. Adım: Models klasörü içerisine **KutuphaneDbContext** adında sınıf dosyası oluşturunuz. **Microsoft.EntityFrameworkCore** isim uzayını ekledikten sonra oluşturulan sınıfın DbContext sınıfından miras alma işlemini gerçekleştiren kodlamaları yapınız.

```
using Microsoft.EntityFrameworkCore;
namespace KutuphaneTakip.Models
{
   public class KutuphaneDbContext:DbContext
   {

   }
}
```

3. Adım: Görsel 6.34'te olduğu gibi **OnConfiguring** metodunu oluşturmak için override işlemini gerçekleştiriniz.



Görsel 6.34: OnConfiguring metodu override işlemi

4. Adım: Override yapılan OnConfiguring metodunun **optionsBuilder** parametresini kullanarak veri tabanı bağlantı cümlesi ile veri tabanına bağlantı işlemlerini gerçekleştiren kodlamaları yapınız.

```
string baglanti = "Server=(localdb)\\mssqllocaldb;Database=KutuphaneDb;Trusted_
Connection=True; ";
protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
{
   optionsBuilder.UseSqlServer(baglanti);
}
```

6.9.3. Entity Framework Core DbSet Özelliği

Bir Model (Entity Model) oluşturmak, onu veri tabanı işlemlerinde kullanmak için yeterli değildir. Veri tabanı ile eşleştirilecek olan modeller için DbSet özelliği oluşturulmalıdır. Entity Framework Core sadece DbSet özelliğine sahip olan modeller ile işlem yapar. Yapılan eşleştirme sonucunda oluşturulan modeli kullanarak ekleme, silme ve güncelleştirme işlemleri veri tabanına yansıtılacaktır.

Aşağıdaki kodlamaları yaparak Kitap modeline DbSet özelliğini verme işlemini gerçekleştiriniz.

```
protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
 {
   optionsBuilder.UseSqlServer(baglanti);
 }
 public DbSet<Kitap> Kitaplar {get;set;}
```

6.9.4. Entity Framework Core Migrations

Mirations, oluşturulan Entity tipinde bir modeli veya model üzerinde yapılan değişikleri veri tabanına yansıtma işlemidir. Uygulama geliştirirken modeli veri tabanına aktarma, yeni modeller oluşturma veya mevcut modeller üzerinde değişiklik yapma durumlarında kullanılır. Migrations işlemi eğer model DbContext sınıfında DbSet özelliği verilmiş ise gerçekleşir.

Migrations işlemleri yapabilmek için Entity Framework Tools paketi kurulduktan sonra bu paketi kullanabilmek için **dotnet ef** adında komut satırı aracına ihtiyaç duyulur.

17. Uygulama

Oluşturulan modelin veri tabanına yansıtılma işlemini yönergeler doğrultusunda gerçekleştiriniz.

1. Adım: Bu adımda dotnet ef komut satırı aracını kurmak ve bir defa kurulduğunda diğer projelerde tekrar kurmamak için aşağıdaki komutu çalıştırınız.

```
> dotnet tool install --global dotnet-ef
```

2. Adım: Entity Framework aracının doğru yüklendiğinin kontrolü için aşağıdaki komutu çalıştırınız.

```
> dotnet ef
```

3. Adım: Migrations işlemini başlatmak için aşağıdaki komutu çalıştırınız.

Not

Komutun sonuna eklenen **birinci** kelimesi migration adıdır, farklı bir ad kullanılabilir.

```
> dotnet ef migrations add birinci
```

4. Adım: Explorer panelinde Migrations adında klasör ve klasör içerisinde migrations dosyalarının otomatik oluşturulduğunu gözlemleyiniz.

5. Adım: Oluşturulan migrations dosyalarını kullanarak modelin veri tabanına yansıtılması işlemi için aşağıdaki komutu çalıştırınız.

```
> dotnet ef database update
```

Adım 6: Sql Server panelinde modelin veri tabanı içine tablo olarak eklendiğini gözlemleyiniz.

Not

Eğer mevcut bir veri tabanı yok ise migrations komutu veri tabanı oluşturma işlemini de gerçekleştirir.

Sıra Sizde

- Kitap modeline string tipinde **YayinEvi** adında özellik ekleyerek güncellenen modelin migrations işlemini gerçekleştiriniz.
- Migration adı olarak daha önceden verilen ad kullanıldığında nasıl bir hata ile karşılaşılmaktadır? Bu hatanın çözümünü sınıf arkadaşlarınızla paylaşınız.

Migration işlemi sonrasında veri modeli veri tabanına tablo olarak ve modelin özellikleri ise tablonun sütunları olarak aktarılır. Sınıf içerisinde Data Annotation (Veri Açıklamaları) kullanarak veri tabanına aktarılacak olan modelin nitelikleri değiştirilebilir. System.ComponentModel.DataAnnotations ve System.ComponentModel.DataAnnotations.Schema isim uzayında yer alan niteliklerin kullanımı aşağıda belirtilmiştir.

Key niteliği, sınıfın özelliğini primary key (birincil anahtar) olarak ayarlar. Bu nitelik, sınıf içindeki int türündeki adı **Id** veya **<Sınıf Adı>Id** olan özellik var ise eklenmeyebilir.

```
[Key]
public int KitapId { get; set; }
```

ForeignKey niteliği, sınıfın özelliğini başka bir tablo ile ilişkilendirmek için kullanılır. Bu nitelik belirtilmediği takdirde ilişkili sütun otomatik olarak oluşturulur.

```
[ForeignKey("KitapTurleri")]
public int KitapTurId { get; set; }
```

Required niteliği, bu nitelik veri tabanı tablosunda ilgili sütunu NOT NULL (Boş Bırakılamaz) sütun yapmak için kullanılır.

```
[Required]
public string KitapAdi { get; set; }
```

MinLength niteliği, bu nitelik eşlenmesi gereken sütun için minimum karakter veya bayt sayısını belirtir.

```
[MinLength(10)]
public string KitapYazar { get; set; }
```

MaxLength niteliği, bu nitelik eşlenmesi gereken sütun için maksimum karakter veya bayt sayısını belirtir.

```
[MaxLength(150)]
public string KitapYazar { get; set; }
```

StringLength niteliği, sınıfın string özellikleri maksimum karakter sayısını belirtir.

```
[StringLength(250)]
public string KitapAdi { get; set; }
```

Table niteliği, bu nitelik tablo adı olarak DbSet ile verilen isim yerine farklı bir isim vermek için kullanılır.

```
[Table("Books")]
public class Kitap
```

Column niteliği, bu nitelik tablo içindeki sütun adı olarak sınıf özelliğinin ismi yerine farklı bir isim vermek için kullanılır.

```
[Column("BookName")]
public string KitapAdi { get; set; }
```

NotMapped niteliği, bu nitelik sınıf özelliğinin tablo içinde sütun olarak oluşturulmasını engeller.

```
[NotMapped]
public string YayinEvi { get; set; }
```

18. Uygulama

Kitap modeline hem migrations işlemi ile veri tabanı yansımasında hem de Tag Helper'lar için Data Annotations nitelikleri ekleme işlemlerini yönergeler doğrultusunda gerçekleştiriniz.

1. Adım: Kitap modeline aşağıda verilen Data Annotation niteliklerini ekleyiniz.

```
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace KutuphaneTakip.Models
{
   [Table("KitapTablo")] // DbSet ile verilen isim yerine kullanılır.
   public class Kitap
   {
      [Key] // Sınıf Adı + Id olduğu için kullanılmayabilir.
      public int KitapId { get; set; }

      [Required(ErrorMessage = "Kitap Adı Boş Bırakılmaz.")]
      [MaxLength(200, ErrorMessage = "Kitap Adı En Fazla 200 Karakter.")]

      [Display(Name = "Kitap Adı")]
      public string KitapAdi { get; set; }

      [Required(ErrorMessage = "Yazar Adı Boş Bırakılamaz.")]
      [MaxLength(100, ErrorMessage  = "Yazar Adı En Fazla 100 Karakter.")]

      [Display(Name ="Kitap Yazarı")]
      public string KitapYazar { get; set; }

      [Display(Name  = "Kitap Türü")]
      [Required(ErrorMessage  = "Kitap Türü Boş bırakılamaz.")]
      [MinLength(5, ErrorMessage = "Kitap Türü En Az 5 Karakter.")]
      public string KitapTur { get; set; }

      [MaxLength(200)]
      public string YayinEvi { get; set; }
   }
}
```

2. Adım: Kitap modelini veri tabanına aktarmak için migrations işlemlerini uygulayınız.

Sıra Sizde

Migration işlemi sonucu veri tabanında meydana gelen değişimleri sınıf arkadaşlarınızla paylaşınız.

6.9.5. Entity Framework Core Veri Ekleme

Entity Framework Core'da DbContext sınıfının **Add** ve **SaveChanges** isimli metotları kullanılarak veri ekleme işlemleri gerçekleştirilir.

19. Uygulama

KutuphaneTakip uygulamasında DbContext sınıfı kullanarak **Kitaplar** modelinin özelliklerine aktarılan verileri veri tabanına kaydetme işlemini yönergeler doğrultusunda gerçekleştiriniz.

1. Adım: Controllers klasörü içine **KitapController** adında controller oluşturunuz. KitapController sınıfına **Ekle** adında GET Action metodu oluşturunuz.

```
using Microsoft.AspNetCore.Mvc;
namespace KutuphaneTakip.Controllers
{
   public class KitapController:Controller
   {
      [HttpGet]
      public IActionResult Ekle()
      {
         return View();
      }
   }
}
```

2. Adım: Views klasörüne, **Kitap** adında klasör oluşturunuz. Kitap klasörüne **Ekle.cshtml** adında view dosyası oluşturarak aşağıdaki kodlamaları yapınız.

```
@model KutuphaneTakip.Models.Kitap;
<form asp-controller="kitap" asp-action="ekle" method="post">
   <div asp-validation-summary="All"></div>
   <div class="form-group">
      <label asp-for="KitapAdi"></label>
      <input asp-for="KitapAdi" class="form-control">
   </div>
   <div class="form-group">
      <label asp-for="KitapYazar"></label>
      <input asp-for="KitapYazar" class="form-control">
   </div>
   <div class="form-group">
      <label asp-for="KitapTur"></label>
      <input asp-for="KitapTur" class="form-control">
   </div>
   <div class="form-group">
      <label asp-for="YayinEvi"></label>
      <input asp-for="YayinEvi" class="form-control">
   </div>
   <div class="form-group">
      <button type="submit" class="btn btn-primary"> Kaydet</button>
   </div>
</form>
```

3. Adım: KitapController içinde Ekle adında POST action metot oluşturarak aşağıdaki kodlamaları yapınız.

```
[HttpPost]
 [ValidateAntiForgeryToken]
 public IActionResult Ekle(Kitap kitap)
 {
    if (ModelState.IsValid)
    {
         using (var db = new KutuphaneDbContext())
         {
               db.Add(kitap);
               db.SaveChanges();
               return Content("Kitap Başarıyla Kaydedildi");
          }
    }
    return View(model);
 }
```

4. Adım: Veri ekleme işlemi sonrası veri tabanında verilerin eklendiğini kontrol ediniz.

6.9.6. Entity Framework Core Veri Alma ve LINQ Sorguları

LINQ (Language Integrated Query) bir veri kaynağındaki veriler üzerinde işlem yapmak için kullanılan bir teknolojidir. LINQ sorguları en büyük özelliği farklı veri kaynaklarında aynı komutları kullanarak işlem yapma yeteneğidir. LINQ ile bir veri tabanına veri ekleme, silme, güncelleme ve alma işlemi yapılabilir. Bu öğrenme biriminde LINQ'nun sadece veri alma metotlarından yararlanılacak diğer işlemler için **DbContext** sınıfı kullanılacaktır.

LINQ'nun Query Syntax (Sorgu Yazımı) ve Method Syntax (Metot Yazımı) olmak üzere iki kullanımı bulunur. Aşağıdaki kodlarda, sorgu ve metot yazımları ile örnekler verilmiştir.

Sorgu Yazımı	Metot Yazımı
var sonuc = from k in db.Kitaplar where k.KitapId==1 orderby k.KitapAdi select k;	var sonuc = db.Kitaplar .Where(k =>KitapId==1) .OrderBy(k => k.KitapAdi) .ToList();

Not

Bu öğrenme biriminde Metot Yazımı kullanılacaktır.

En çok kullanılan LINQ metotları şunlardır:

First(): Belirlenen koşula göre bulunan ilk değeri verir. Koşula göre kayıt bulunmaz ise hata verir.

```
var sonuc=db.Kitaplar.First(x=> x.KitapAdi=="Çalı Kuşu");
```

FirstOrDefault(): First metoduna benzer çalışır farkı koşula göre kayıt bulunmaz ise hata yerine NULL değerini verir.

```
var sonuc=db.Kitaplar.FirstOrDefault(x=> x.KitapAdi=="Çalı Kuşu");
```

Single(): Belirlenen koşula göre tek bir değer verir. Koşula göre birden fazla değer var ise hata verir.

```
var sonuc=db.Kitaplar.Single(x=> x.KitapAdi=="Çalı Kuşu");
```

SingleOrDefault(): Single metoduna bezer çalışır farkı koşula göre birden fazla değer var ise hata yerine NULL değerini verir.

```
var sonuc=db.Kitaplar.SingleOrDefault(x=> x.KitapAdi=="Çalı Kuşu");
```

Any(): Belirlenen koşula göre kayıt var ise true yok ise false değerini verir.

```
var sonuc=db.Kitaplar.Any(x=> x.KitapAdi=="Çalı Kuşu");
```

Select(): Bir tablo içerisinden belirlenen alan veya alanların değerlerini verir.

```
//Bir alana göre seçim yapma
 var sonuc=db.Kitaplar.Select(x=> x.KitapAdi);
```

```
//Birden fazla alana göre seçim yapma
var sonuc=db.Kitaplar.Select(x=>new {x.KitapAdi,x.KitapYazar});
```

Where(): Belirlenen koşula göre tüm kayıtları verir.

```
var sonuc=db.Kitaplar.Where(x=>x.KitapYazar=="Peyami Safa");
```

OrderBy(): Belirlenen ölçüte göre kayıtların sıralı hâlini verir.

```
//A dan Z ye sıralama
var sonuc=db.Kitaplar.OrderBy(x=> x.KitapAdi);
//Z den A ya sıralama
var sonuc1=db.Kitaplar.OrderByDescending(x=> x.KitapAdi);
// Birden fazla ölçüte göre sıralama
var sonuc2=db.Kitaplar.OrderBy(x=> x.KitapAdi).ThenBy(x=>x.KitapYazar);
```

ToList(): Veri tabanından gelen verileri liste hâline dönüştürerek View sayfalarında kullanılmasını sağlar.

```
var sonuc2=db.Kitaplar.ToList();
```

20. Uygulama

Veri tabanına eklenen kayıtların LINQ kullanılarak View sayfasında gösterilmesi işlemini yönergeler doğrultusunda gerçekleştiriniz.

1. Adım: Kütüphane uygulamasında Kitap Controller'ı içine **Liste** adında action metot ekleyerek aşağıdaki kodlamaları action içine yazınız.

```
[HttpGet]
 public IActionResult Liste()
 {
        var liste=db.Kitaplar.Find(2);
        return View(liste);
 }
```

2. Adım: Views/Kitap klasörü içine Liste adında view ekleyerek aşağıdaki kodlamaları yapınız. Uygulamayı çalıştırarak sonuçları gözlemleyiniz.

```
@model KutuphaneTakip.Models.Kitap

<ul class="list-group">
 <li class="list-group-item">@Model.KitapAdi</li>
 <li class="list-group-item">@Model.KitapYazar</li>
 <li class="list-group-item">@Model.KitapTur</li>
 <li class="list-group-item">@Model.YayinEvi</li>
</ul>
```

3. Adım: Liste action metodunu aşağıdaki gibi düzenleyerek sonuçları tekrar gözlemleyiniz.

```
[HttpGet]
 public IActionResult Liste()
{
        var liste=db.Kitaplar.First();
        return View(liste);
}
```

4. Adım: Birden çok veriyi liste hâlinde göstermek için Liste action metodunu aşağıdaki gibi düzenleyiniz.

```
[HttpGet]
 public IActionResult Liste()
{
    var liste=db.Kitaplar.ToList();
    return View(liste);
}
```

5. Adım: Liste view dosyasında verileri tablo içerisinde göstermek için aşağıdaki gibi düzenleyiniz. Uygulamayı çalıştırarak sonuçları gözlemleyiniz.

```
@model IList<KutuphaneTakip.Models.Kitap>
<table class="table">
  <thead>
    <tr>
      <th>Kitap No</th>
      <th>Kitap Adı</th>
      <th>Yazar</th>
      <th>Tür</th>
    </tr>
  </thead>
  <tbody>
    @foreach (var item in Model)
    {
      <tr>
        <td>@item.KitapAdi</td>
        <td>@item.KitapYazar</td>
        <td>@item.KitapTur</td>
   <td>@item.YayinEvi</td>
      </tr>
    }
  </tbody>
</table>
```

6. Adım: Belirlenen kritere göre verileri liste hâlinde göstermek için Liste action metodunu aşağıdaki gibi düzenleyerek sonuçları gözlemleyiniz.

```
 [HttpGet]
 public IActionResult Liste()
 {
    var liste=db.Kitaplar.Where(x=>x.KitapId>=2).ToList();
    return View(liste);
 }
```

7. Adım: Belirlenen kritere göre verileri liste hâlinde göstermek için Liste action metodunu aşağıdaki gibi düzenleyerek sonuçları gözlemleyiniz.

```
 [HttpGet]
 public IActionResult Liste()
{
  var liste=db.Kitaplar.Where(x=>x.KitapAdi.StartsWith("A")).ToList();
  return View(liste);
}
```

8. Adım: Verileri sıralı olarak göstermek için Liste action metodunu aşağıdaki gibi düzenleyerek sonuçları gözlemleyiniz.

```
[HttpGet]
public IActionResult Liste
{
  var liste=db.Kitaplar.OrderBy(x=>x.KitapAdi).ToList();
  return View(liste);
}
```

Sıra Sizde

Veri tabanındaki alınan verileri kitap adlarına göre tersten sıralayarak view içinde gösteren kodlamaları yapınız.

6.9.7. Entity Framework Core Veri Güncelleme

Veri güncelleme DbContext sınıfını kullanarak veri tabanında var olan bir kayıt alınarak üzerinde değişiklik yapıldıktan sonra tekrar veri tabanına kaydetme işlemleridir. İlk olarak **Find()** metodu ile güncelleştirilecek kayıt bulunmalıdır. Find() metodu bir kaydın ID (primary key) alanına göre arama yaparak bulunan kaydı getirir. Bulunan kayıt güncellendikten sonra SaveChanges() metodu ile yapılan değişiklikler veri tabanına aktarılır.

21. Uygulama

KutuphaneTakip projesinde, kitap bilgilerini güncelleme işlemlerini yönergeler doğrultusunda gerçekleştiriniz.

1. Adım: Güncelleme işlemi yapılacak kayıtların seçilebilmesi için Liste view dosyasına aşağıdaki düzenlemeleri yapınız. Yapılan düzenleme ile her bir kaydın ID numarası, link etiketi içerisinde parametre olarak oluşturulacaktır.

```
    @foreach (var item in Model)
    {
      <tr>
        <td>@item.KitapAdi</td>
        <td>@item.KitapYazar</td>
        <td>@item.KitapTur</td>
  <td>@item.YayinEvi</td>
   <td><a asp-controller="kitap" asp-action="guncelle" asp-route-id="@item.KitapId">-
Güncelle</a> </td>
 </tr>
}
```

2. Adım: KitapController içine Guncelle adında GET Action metodu oluşturarak güncellenecek olan kaydın view sayfasında gösterilmesini sağlayan kodlamaları yapınız.

```
[HttpGet]
public IActionResult Guncelle(int id)
{
   using (var db = new KutuphaneDbContext())
   {
      var kitap = db.Kitaplar.Find(id);
      return View(kitap);
   }
}
```

3. Adım: Views/Kitap klasörüne **Guncelle.cshtml** adında view dosyası oluşturarak aşağıdaki kodlamaları yapınız. Bu view dosyası, GET özelliğine sahip Guncelle Action metodundan gönderilen kayıt bilgilerini görüntüleme ve değiştirme işlemi yapacaktır.

```
 @model KutuphaneTakip.Models.Kitap;
<form asp-controller="kitap" asp-action="guncelle" method="post">
   <div asp-validation-summary="All"></div>
   <input type="hidden" asp-for="KitapId">
   <div class="form-group">
      <label asp-for="KitapAdi"></label>
      <input asp-for="KitapAdi" class="form-control">
   </div>
   <div class="form-group">
      <label asp-for="KitapYazar"></label>
      <input asp-for="KitapYazar"  class="form-control">
   </div>
   <div class="form-group">
      <label asp-for="KitapTur"></label>
      <input asp-for="KitapTur"  class="form-control">
   </div>
   <div class="form-group">
      <label asp-for="YayinEvi"></label>
      <input asp-for="YayinEvi"  class="form-control">
   </div>
  <div class="form-group">
     <button type="submit" class="btn btn-primary"> Güncelle</button>
   </div>
</form>
```

4. Adım: KitapController'ı içine Guncelle adında POST Action metodu oluşturarak aşağıdaki kodlamaları yapınız.

```
[HttpPost]
[ValidateAntiForgeryToken]
public IActionResult Guncelle(Kitap kitap)
{
   if (ModelState.IsValid)
   {
       using (var db = new KutuphaneDbContext())
       {
          db.Update(kitap);
          db.SaveChanges();
          return RedirectToAction("Liste");
       }
    }
     return View(kitap);
}
```

6.9.8. Entity Framework Core Veri Silme

Veri silme DbContext sınıfını kullanarak veri tabanında var olan bir kaydın silinmesi işlemidir.

22. Uygulama

KutuphaneTakip projesinde, veri tabanında tablo üzerinde kitap bilgilerinin bulunduğu satırı silme işlemini yönergeler doğrultusunda gerçekleştiriniz.

1. Adım: Silme işlemi yapılacak kayıtların seçilebilmesi için Liste.cshtml view dosyasına aşağıdaki düzenlemeleri yapınız. Yapılan düzenleme ile her bir kaydın ID numarası link etiketi içerisinde parametre olarak oluşturulacaktır.

```
@foreach (var item in Model)
{
  <tr>
  <td>@item.KitapAdi</td>
  <td>@item.KitapYazar</td>
  <td>@item.KitapTur</td>
<td>@item.YayinEvi</td>
  <td><a asp-controller="kitap" asp-action="guncelle" asp-route-id="@item.KitapId">-
Güncelle</a></td>
   <td><a asp-controller="kitap" asp-action="sil" asp-route-id="@item.KitapId">Sil</a>
</td>
  </tr>
}
```

2. Adım: KitapController içine Sil adında GET Action metodu oluşturarak silinecek olan kaydın view sayfasında gösterilmesini sağlayan kodlamaları yapınız.

```
[HttpGet]
public IActionResult Sil(int? id)
{
   if (id == null)
   {
     return NotFound();
   }
   using (var db = new KutuphaneDbContext())
   {
      var kitap  = db.Kitaplar.Find(id);
      if (kitap == null)
      {
        return NotFound();
      }
      return View(kitap);
   }
}
```

3. Adım: Views içindeki Kitap klasörüne **Sil** adında view dosyası oluşturarak aşağıdaki kodlamaları yapınız.

```
@model KutuphaneTakip.Models.Kitap;
<p><b>Kitap Adı:</b> @Model.KitapAdi</p>
<p><b>Kitap Yazar:</b> @Model.KitapYazar</p>
<p><b>Kitap Türü:</b>@Model.KitapTur</p>
<p><b>Yayın Evi:</b> @Model.YayinEvi</p>
<form asp-controller="kitap" asp-action="silonay" method="post" asp-route-id="@Mo-
del.KitapId">
      <button class="btn btn-danger">Sil</button>
</form>
```

4. Adım: KitapController'ı içine SilOnay adında POST Action metodu oluşturarak aşağıdaki kodlamaları yapınız.

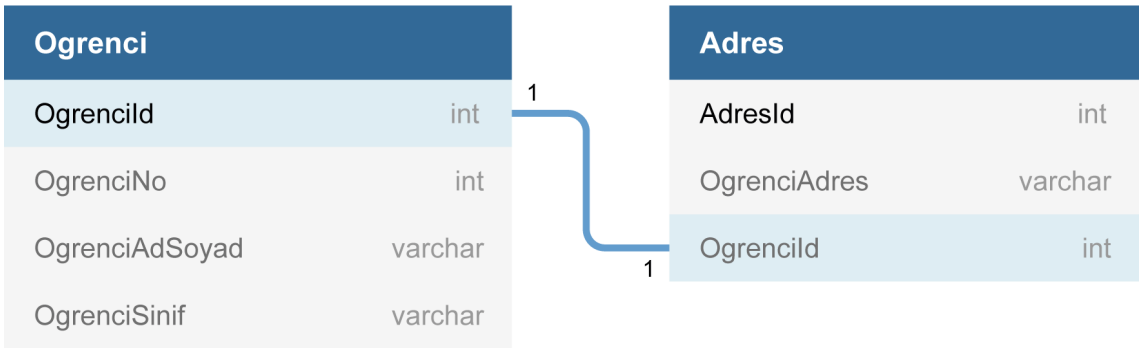
```
[HttpPost]
[ValidateAntiForgeryToken]
public IActionResult SilOnay(int id)
{
    using (var db = new KutuphaneDbContext())
    {  var kitap  = db.Kitaplar.Find(id);
       db.Remove(kitap);
       db.SaveChanges();
       return RedirectToAction("Liste");
     }
}
```

6.9.9. Entity Framework Core Veri Tabanı İlişkileri

Veri tabanı ilişkileri, farklı tablolarda bulunan verilerin birbirleriyle ilişkili alanları arasında bağlantı kurulmasıyla oluşur. A tablosundaki ilişkilendirilen alan üzerinden, B tablosundaki verilere erişim sağlanır. Böylelikle veri tabanı organize bir şekilde yönetilebilir.

Veri tabanında üç tip ilişki mevcuttur. Bu ilişkiler şunlardır:

One to One (Bire Bir) İlişkiler: Bağlantı kurulan tabloların ilişkili alanlarında benzersiz sadece bir kayıt olmasıdır. A tablosundaki bir satıra karşılık, B tablosunda da sadece bir satır bulunur. Görsel 6.35'te her öğrenciye ait bir adres ve her adrese ait bir öğrenci bulunmaktadır. Adresler tablosu OgrenciId alanına veri kaydedilmek istendiğinde Ogrenciler tablosunda bulunan OgrenciId numaralarından farklı veya daha önceden kaydedilmiş numara verilmek istendiğinde veri tabanı hata verecektir.



Görsel 6.35: One to One (Bire Bir) tablo ilişkisi

23. Uygulama

Entity Framework Core ile modeller arasındaki bire bir ilişki tanımlaması yapılarak öğrenci ve öğrencilere ait adres bilgilerini kaydetme işlemini yönergeler doğrultusunda gerçekleştiriniz.

1. Adım: KutuphaneTakip projesi Models klasörüne, **Ogrenci** ve **Adres** adında iki model sınıfı oluşturarak sınıfın özelliklerini aşağıdaki gibi veriniz.

```
public class Ogrenci
{
  public int OgrenciId {get;set;}
  public int  OgrenciNo {get;set;}
  public string OgrenciAdSoyad {get;set;}
  public string OgrenciSinif {get;set;}
  public  Adres Adres {get;set;}
 }
```

```
public class Adres
{
  public int AdresId {get;set;}
  public string OgrenciAdres {get;set;}
  public int OgrenciId {get;set;}  //Yabancı Anahtar (Foreign Key)
  public Ogrenci Ogrenci {get;set;}
}
```

2. Adım: KutuphaneDbContext sınıfı içerisine Ogrenci ve Adres sınıflarına DbSet özelliği vererek migrations işlemlerini gerçekleştiriniz.

```
public DbSet<Ogrenci> Ogreciler {get;set;}
public DbSet<Adres> Adresler {get;set;}
```

3. Adım: Controllers klasörüne **OgrenciController** adında controller oluşturarak öğrenci bilgilerinin kaydedilmesi işlemini gerçekleştiren kodlamaları yapınız.

```
public class OgrenciController:Controller
 {
    KutuphaneDbContext db=new KutuphaneDbContext();

    [HttpGet]
    public IActionResult Ekle()
    {
       return View();
    }
    [HttpPost]
    [ValidateAntiForgeryToken]
    public IActionResult Ekle(Ogrenci ogrenci)
    {
       if(ModelState.IsValid)
       {
          db.Ogreciler.Add(ogrenci);
          db.SaveChanges();
          return RedirectToAction("Liste");
       }
       return View(ogrenci);
    }
 }
```

4. Adım: Views/Ogrenci klasörüne Ekle.cshtml adında öğrenci bilgilerinin kaydedilmesini sağlayan view dosyasını oluşturunuz, içerisine aşağıdaki kodlamaları yapınız.

Not

Tag Helper kullanımı için Ogrenci sınıfına veri açıklamalarını ekleyiniz.

```
@model KutuphaneTakip.Models.Ogrenci;
<form asp-controller="ogrenci" asp-action="ekle" method="POST">
  <div asp-validation-summary="All"></div>
  <div class="form-group">
    <label asp-for="OgrenciNo"></label>
    <input asp-for="OgrenciNo" class="form-control">
  </div>
  <div class="form-group">
    <label asp-for="OgrenciAdSoyad"></label>
    <input asp-for="OgrenciAdSoyad" class="form-control">
  </div>
  <div class="form-group">
    <label asp-for="OgrenciSinif"></label>
    <input asp-for="OgrenciSinif" class="form-control">
  </div>
  <div class="form-group">
    <button type="submit" class="btn btn-primary"> Kaydet</button>
  </div>
</form>
```

5. Adım: OgrenciController dosyasına öğrenciler listesini oluşturmak için **Liste** adında action metot oluşturunuz.

```
[HttpGet]
public IActionResult Liste()
{
    var ogrenciler = db.Ogrenciler.ToList();
    return View(ogrenciler);
}
```

6. Adım: Views/Ogrenci klasörüne **Liste.cshtml** adında view dosyası oluşturarak aşağıdaki kodlamaları yapınız.

```
@model List<KutuphaneTakip.Models.Ogrenci>
<table class="table">
  <thead>
    <tr>
      <th scope="col">Öğrenci Id</th>
      <th scope="col">Öğrenci Numara</th>
      <th scope="col">Ad Soyad</th>
      <th scope="col">Sınıf</th>
    </tr>
  </thead>
  <tbody>
    @foreach (var item in Model)
    {
     <tr>
```

```
        <td>@item.OgrenciId</th>
        <td>@item.OgrenciNo</td>
        <td>@item.OgrenciAdSoyad</td>
        <td>@item.OgrenciSinif</td>
       <td><a asp-controller="ogrenci" asp-action="adresekle" asp-route-id="@item.Og-
renciId">Adres Ekle</a> </td>
      </tr>
      }

  </tbody>
</table>
```

7. Adım: OgrenciController dosyasına öğrencilerin listesini oluşturmak için **AdresEkle** adında action metot oluşturunuz.

```
[HttpGet]
public IActionResult AdresEkle(int id)
{
   var ogrenci = db.Ogreciler.Find(id);
   ViewBag.Ogrenci = ogrenci;
   return View();
}
```

8. Adım: Views/Ogrenci klasörü içine AdresEkle.cshtml adından view dosyası ekleyiniz. Oluşturulan view dosyasında seçilen öğrenciye ait adres bilgilerinin girişi yapılacaktır.

```
@model KutuphaneTakip.Models.Adres
<p><b>Öğrenci Id:</b>@ViewBag.Ogrenci.OgrenciId </p>
<p><b>Öğrenci Numara:</b> @ViewBag.Ogrenci.OgrenciNo</p>
<p><b>Ad Soyad:</b> @ViewBag.Ogrenci.OgrenciAdSoyad</p>
<p><b>Sınıf:</b>@ViewBag.Ogrenci.OgrenciSinif</p>
<form asp-controller="ogrenci" asp-action="adresekle" method="post">
    <input value="@ViewBag.Ogrenci.OgrenciId" type="hidden" name="OgrenciId"></
input>
  <textarea asp-for="OgrenciAdres" class="form-control"></textarea>
  <button class="btn btn-primary">Adres Ekle</button>
</form>
```

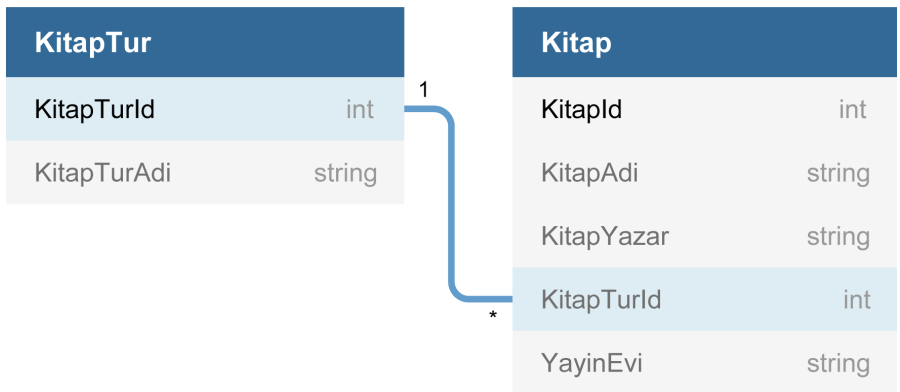
9. Adım: OgrenciController dosyasına öğrenci adres bilgilerini kaydetmek için AdresEkle POST action metoduna aşağıdaki kodlamaları yapınız.

```
[HttpPost]
[ValidateAntiForgeryToken]
public IActionResult AdresEkle(Adres adres)
{
   db.Adresler.Add(adres);
   db.SaveChanges();
   return RedirectToAction("Liste");
}
```

Sıra Sizde

Daha önce adres bilgileri kaydedilmiş bir öğrenciye tekrar adres bilgileri kaydetme işlemi gerçekleştirildiğinde nasıl bir hata ile karşılaşılır? Arkadaşlarınızla gruplar oluşturarak karşılaşılan bu hatanın çözümü için ne yapılması gerektiğini araştırınız ve bulduğunuz sonuçları sınıfta paylaşınız.

One to Many (Bire Çok) İlişkiler: Bağlantı kurulan tabloların ilişkili alanlarında, bir tablodaki kayda karşılık diğer tabloda çok sayıda kaydın olmasıdır. A tablosundaki bir satıra karşılık, B tablosunda birden fazla satır olabilir. Görsel 6.36 'da her kitabın bir türü ve her türe ait birden çok kitap bulunmaktadır.



Görsel 6.36: One to Many (Bire Çok) tablo ilişkisi

24. Uygulama

Entity Framework Core ile modeller arasındaki **bire çok** ilişki tanımlaması yaparak **kitap türler** ile **kitap bilgiler** arasında ilişki kurup bilgilerini kaydetme işlemini yönergeler doğrultusunda gerçekleştiriniz.

1. Adım: KutuphaneTakip projesi Models klasörüne KitapTur adında model sınıfı oluşturarak aşağıdaki kodlamaları yapınız. ICollection özelliği KitapTur sınıfı ile Kitap sınıfı arasında bire çok ilişki kurulması için kullanılmaktadır.

Not

ICollection özelliği için System.Collections.Generic isim uzayını ekleyiniz.

```
public class KitapTur
{
  public int KitapTurId { get; set; }
  public string KitapTurAdi { get; set; }

  public ICollection<Kitap> Kitap { get; set; }
}
```

2. Adım: Kitap modelini KitapTur modeli ile ilişkilendirmek için aşağıdaki düzenlemeleri yapınız.

```
public class Kitap
{
  public int KitapId { get; set; }
  public string KitapAdi { get; set; }
  public string KitapYazar { get; set; }
  public int KitapTurId { get; set; }
  public string YayinEvi { get; set; }

  public KitapTur KitapTur { get; set; }
}
```

3. Adım: KutuphaneDbContext sınıfı içerisine KitapTur sınıfına DbSet özelliği vererek migrations işlemlerini gerçekleştiriniz.

```
public DbSet<KitapTur> KitapTurler { get; set; }
```

4. Adım: Controllers klasörüne **KitapTurController** adında controller ekleyerek aşağıdaki kodlamaları yapınız.

```
[HttpGet]
public IActionResult Ekle()
{
  return View();
}
```

5. Adım: Views/KitapTur klasörüne **Ekle.cshtml** adından view dosyası oluşturarak aşağıdaki kodlamaları yapınız.

```
@model KutuphaneTakip.Models.KitapTur;
<form asp-controller="kitaptur" asp-action="ekle" method="post">
   <div asp-validation-summary="All"></div>
   <div class="form-group">
      <label asp-for="KitapTurAdi"></label>
      <input asp-for="KitapTurAdi" class="form-control">
   </div>
     <div class="form-group">
      <button type="submit" class="btn btn-primary"> Kaydet</button>
   </div>
</form>
```

6. Adım: KitapTurController dosyasına **Ekle** adından POST action metodu oluşturarak kitap türlerini kaydetme işlemini gerçekleştiriniz.

```
[HttpPost]
public IActionResult Ekle(KitapTur kitaptur)
{
   if (ModelState.IsValid)
   {
      var db = new KutuphaneDbContext();
      db.Add(kitaptur);
      db.SaveChanges();
      return Content ("Kitap Türü Eklendi.");
   }

   return View(kitaptur);
}
```

7. Adım: KitapController içerisinde Ekle GET Action metodu ile kitap türlerini view sayfasında select etiketi kullanarak göstermek için aşağıdaki kodlamaları yapınız.

```
[HttpGet]
public IActionResult Ekle()
{
 var db = new KutuphaneDbContext();
 ViewBag.KitapTurList = db.KitapTurler.OrderBy(x => x.KitapTurAdi).ToList();
 return View();
}
```

8. Adım: Views/Kitap Ekle view sayfasında kitap türleri select etiketi içinde göstermek için aşağıdaki düzenlemeyi yapınız.

```
<div class="form-group">
 <label asp-for="KitapTurId"></label>
 <select class="form-control" asp-for="KitapTurId"
 asp-items= "@(new SelectList(ViewBag.KitapTurList,"KitapTurId","KitapTurAdi"))">
 </select>
</div>
```

9. Adım: Kitap isimlerinin listelendiği List view içerisinde, kitap türlerini ID olarak değil de tür adı olarak görüntülemek için "Eager Loading" kullanılacaktır. **Eager Loading,** Entity Frame Core'da bir model sınıfına ilişkilendirmiş başka model sınıfının yüklenmesi için kullanılan tekniğin adıdır. Eager Loading işlemi için **Include** ve **ThenInclude** adında iki metot bulunur. KitapController içindeki List GET Action metoduna, Kitaplar tablosuna KitapTur tablosunu dâhil ederek içine aşağıdaki kodlamaları yapınız.

Not

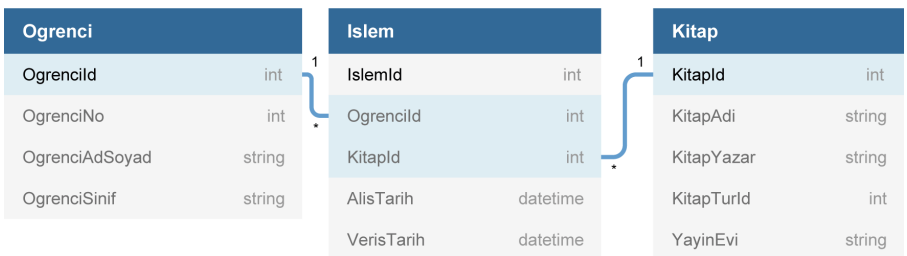
Controller dosyasında Include metodunu kullanabilmek için **Microsoft.EntityFrameworkCore** isim uzayını dâhil ediniz.

```
[HttpGet]
public IActionResult Liste()
{
   using (var db = new KutuphaneDbContext())
   {
      var sonuc = db.Kitaplar.Include(x=>x.KitapTur).ToList();
      return View(sonuc);
   }
}
```

10. Adım: Kitap listesi içerisinde kitap türlerini Id numaraları yerine tür adlarının görüntülenmesi için Views/Kitap klasörü içindeki Liste.cstml dosyasında aşağıdaki değişiklikleri yapınız.

```
<td>@item.KitapTur.KitapTurAdi</td>
```

Many to Many (Çoka Çok) İlişkiler: A tablosundaki bir satıra karşılık, B tablosunda birden fazla satır olabildiği gibi, B tablosundaki bir satıra karşılık da A tablosunda birden fazla satır olabilir. Many to Many ilişki aslında birden fazla One to Many ilişkinin bir araya gelmesinden oluşur. Bu ilişkileri bir araya getirecek olan ve Many to Many ilişkiyi sağlayacak olan bir ara bağlantı tablosunun bulunması gereklidir. Görsel 6.37'de Many to Many ilişkisi gösterilmiştir.



Görsel 6.37: Many to Many (Çoka çok) tablo ilişkisi

25. Uygulama

Kitap modeli ile **Ogrenci** modeli arasına **Islem** adında model oluşturarak **çoka çok** ilişki tanımlaması ve Islem modelinde kitapları ödünç alan öğrencilerin kaydedilmesi işlemlerini yönergeler doğrultusunda gerçekleştiriniz.

1. Adım: Models klasörüne **Islem** adında model sınıfı oluşturunuz. Model özellikleri için aşağıdaki kodlamaları yapınız.

```
public class Islem
{
   public int IslemId { get; set; }
   public int OgrenciId { get; set; }
   public int KitapId { get; set; }
   public DateTime AlisTarih { get; set; }
   public DateTime VerisTarih { get; set; }
   public Ogrenci Ogrenci { get; set; }
   public Kitap Kitap { get; set; }
}
```

2. Adım: Islem modelini Kitap ve Ogrenci modelleri ile ilişkilendirmek için bu sınıflarda aşağıdaki düzenlemeleri yapınız.

```
public class Ogrenci
{
  public int OgrenciId { get; set; }
  public int OgrenciNo { get; set; }
  public string OgrenciAdSoyad { get; set; }
  public string OgrenciSinif { get; set; }

  public Adres Adres{ get; set; }
  public ICollection<Islem> Islem { get; set; }
 }
```

```
public class Kitap
{
  public int KitapId { get; set; }
  public string KitapAdi { get; set; }
  public string KitapYazar { get; set; }
  public int KitapTurId { get; set; }
  public string YayinEvi { get; set; }

  public KitapTur KitapTur { get; set; }
  public ICollection<Islem> Islem { get; set; }
}
```

3. Adım: KutuphaneDbContext sınıfı içerisine Islem sınıfına DbSet özelliği vererek migrations işlemlerini gerçekleştiriniz.

```
public DbSet<Islem> Islemler { get; set; }
```

4. Adım: Controller klasörüne **IslemController** adında dosya oluşturunuz. Oluşturulan dosyaya **Ekle** adından GET Action metodu oluşturarak aşağıdaki kodlamaları yapınız.

```
[HttpGet]
    public IActionResult Ekle()
    {
      var db=new KutuphaneDbContext();
     ViewBag.Ogreciler=db.Ogrenciler.Select(x=> new {x.OgrenciId,x.OgrenciAdSoyad}).
ToList();
      ViewBag.Kitaplar=db.Kitaplar.Select((x=> new {x.KitapId,x.KitapAdi}).ToList();
      return View();
    }
```

5. Adım: Views /Islem klasörüne **Ekle.cshtml** adında view dosyası oluşturarak kodlamaları yapınız.

```
@model KutuphaneTakip.Models.Islem;
<form asp-controller="islem" asp-action="ekle" method="post">
  <div asp-validation-summary="All"></div>
  <div class="form-group">
    <label> Öğrenci Adı</label>
     <select class="form-control" asp-for="OgrenciId"
 asp-items="@(new SelectList(ViewBag.Ogrenciler,"OgrenciId","OgrenciAdSoyad"))">
</select>
  </div>
   <div class="form-group">
    <label> Kitap Adı</label>
     <select class="form-control" asp-for="KitapId"
    asp-items="@(new SelectList(ViewBag.Kitaplar,"KitapId","KitapAdi"))">
</select>
  </div>
   <div class="form-group">
    <label>Alış Tarihi</label>
    <input asp-for="AlisTarih" class="form-control">
  </div>
   <div class="form-group">
    <label>Veriş Tarihi</label>
    <input asp-for="VerisTarih" class="form-control">
  </div>
    <div class="form-group">
    <button type="submit"class="btn btn-primary"> Kaydet</button>
  </div>
</form>
```

6. Adım: IslemController dosyasına Ekle adından POST Action metodu oluşturarak aşağıdaki kodlamaları yapınız.

```
[HttpPost]
 public IActionResult Ekle(Islem model)
 {
   var db = new KutuphaneDbContext();
   db.Add(model);
   db.SaveChanges();
   return Content("İşlem Başarıyla Gerçekleşti.");
 }
```

6.10. Yayınlama (Publish) İşlemleri

Yayınlama (Publish), local ortamda geliştirilen ASP.NET Core uygulamasını internet üzerinden dış dünyaya açma işlemidir. Yayınlama, uygulamayı **Dağıtım** [Deployment (Sunucuya Aktarma)] için hazırlayan süreçtir. Yayınlama işlemi ile Controller ve Model klasörlerinde yer alan **".cs"** uzantılı dosyaları, **bin** klasörü içine **.dll** uzantılı paketlenmiş dosyalar hâlinde oluşturulur. Böylelikle kaynak kodlar geliştiricide, dll dosyaları ise sunucuda bulunur.

ASP.NET Core uygulamasını yayınlamak için terminal panelinden **dotnet publish** komutu kullanılır. Uygulama **Self Contained** (Bağımsız) ve **Runtime Dependent** (Çalışma Zamanına Bağımlı veya Framework bağımlı) olmak üzere iki şekilde yayınlanabilir. Runtime-dependent şeklinden yayınlandığında, uygulamanın çalışacağı ortamda .NET Core Framework'ünün yüklü olması gerekir. Self Contained olarak yayınlandığında ise .NET Core Framework'ünün yüklü olmasına gerek kalmadan yayınlama dosyalarına ek olarak frameworkü ekler.

6.10.1. Çalışma Zamanına Bağlı Yayınlama

Çalışma zamanına bağlı yayınlamada aşağıdaki dotnet komutları kullanılır.

```
 > dotnet publish
# veya
> dotnet publish -c Release
```

Not

Release olarak yayınlama yapıldığında uygulamadaki kodlar, biraz daha optimize edilerek development (geliştirme) aşamasında kullandığımız kodlar dâhil edilmez.

Avantajları şunlardır:

- Dağıtım boyutu küçüktür.
- Çapraz platformlarda kullanılabilir.
- Otomatik olarak .NET Core çalışma zamanının en son sürümü kullanılır.

Dezavantajları şunlardır:

- Uygulamanın çalışacağı bilgisayarda .NET Core yüklü olması gerekir.
- Uygulamanın çalışacağı bilgisayarda .NET Core versiyonu değişebilir.

6.10.2. Bağımsız Yayınlama

Bağımsız yayınlamada aşağıdaki dotnet komutları kullanılır.

```
 > dotnet publish -r <RID> --self-contained
#veya
> dotnet publish -c Release -r <RID> --self-contained
```

RID [RUNTIME IDENTIFIER (Çalışma Zamanı Tanımlayıcısı)], uygulamanın çalıştırılacağı hedef platformları belirlemek için kullanılır. Örneğin: linux-x64, ubuntu.14.04-x64, win10-x64 veya osx.10.12-x64

Avantajları şunlardır:

Birden fazla uygulamayı farklı .NET Core versiyonlarında çalıştırabilir.

- Hangi .NET Core sürümünün yayınlanacağını kullanıcı kontrol eder.
- Platformu belirleme özelliğine sahiptir.

- Birden fazla uygulamayı farklı .NET Core versiyonlarında çalıştırabilir.

  Dezavantajları şunlardır:

- Dağıtım boyutu büyüktür.
- NET Core güncellemesi zordur.

6.10.3. Windows ile IIS Sunucuda Yayınlama

IIS (Internet Information Services) sunucusu, web sitelerini internet üzerinden kullanıcılara sunmak için kullanılan Windows Sunucu tabanlı web uygulamasıdır. IIS ile web sitelerini veya web uygulamalarını yönetme ve yayınlama işlemleri gerçekleştirilir. IIS tüm Windows sistemlerinde bulunur ama kurulu olarak gelmez. ASP.NET Core uygulamaları IIS üzerinde yayınlanmak istendiğinde işletim sistemine kurulması gerekir.

26. Uygulama

Yayınlanan (publish) bir ASP.NET Core uygulamasının IIS sunucu üzerinde dağıtım (deploy) işlemini yönergeler doğrultusunda gerçekleştiriniz.

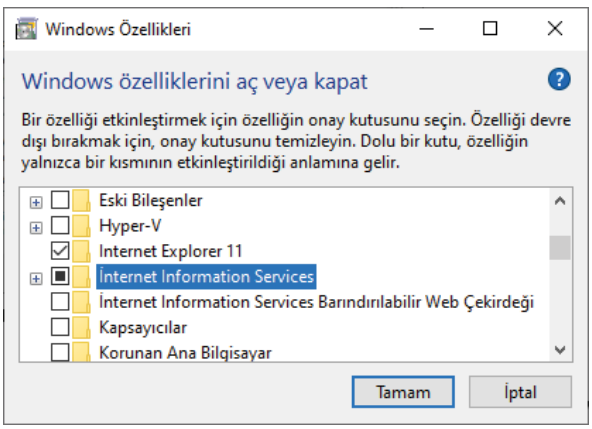
1. Adım: ASP.NET Core uygulamasını çalışma zamanı bağımlı (runtime dependent) olarak yayınlamak için Visual Code programı ile açıldıktan sonra terminal panelinde veya komut istemi penceresinde (cmd) uygulamanın bulunduğu klasöre geçerek aşağıdaki komutu yazınız.

```
> dotnet publish -c Release
```

2. Adım: Uygulama klasörünün içindeki **bin\Release\<Versiyon_Adı>\publish** klasörü içindekiler sunucuda çalışacak olan dosya ve klasörlerdir.

3. Adım: Uygulamayı Windows işletim sisteminde yayınlamak için Denetim masasında **Programlar ve Özellikler** uygulamasında sol taraftaki **Windows özelliklerini aç veya kapat** linkine tıklayınız.

4. Adım: Açılan pencerede Görsel 6.38'de belirtildiği gibi **Internet Information Services** özelliğinin yanındaki kutucuğu işaretleyerek tamam butonuna tıklayınız.

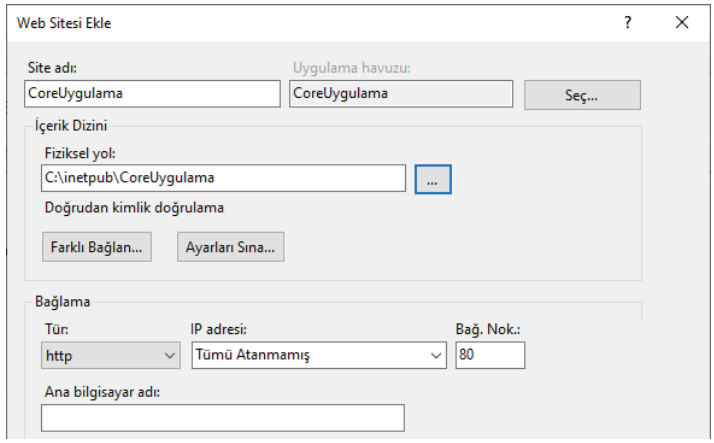


Görsel 6.38: Windows özellikleri penceresi

5. Adım: IIS sunucusunun çalışma durumunu kontrol için web tarayıcı uygulamasının adres satırına **localhost** yazarak IIS karşılama sayfası görüntüleyiniz.
6. Adım: ASP.NET Core uygulamasının bin\Release\<Versiyon_Adı>\publish klasörü içindeki tüm dosyaları kopyalayınız. Bilgisayarın C: \inetpub içine **CoreUygulama** adında klasör oluşturarak kopyalanan dosyaları yapıştırınız.
7. Adım: Çalıştır'a **Inetmgr** komutunu yazarak veya arama kutusuna **Internet Information Management** (Yönetim) yazarak IIS yönetim panelini açınız.
Adım 8: Sol taraftaki menüden, **Siteler** linkine sağ tıklayarak, açılan menüden **Web Sitesi Ekle** seçerek, açılan pencerede Görsel 6.39'daki alanları doldurarak Tamam butonuna tıklayınız.

Not

Default Web Site ile yeni eklediğimiz site aynı port numarasına sahip olduğu için uyarı mesajı verilecektir. Uyarı mesajına **tamam** dedikten sonra Default Web Sitenin çalışmasını sağ taraftaki menüden durdurunuz. Ardından CoreUygulama isimli sitenin çalışmasını başlatınız.



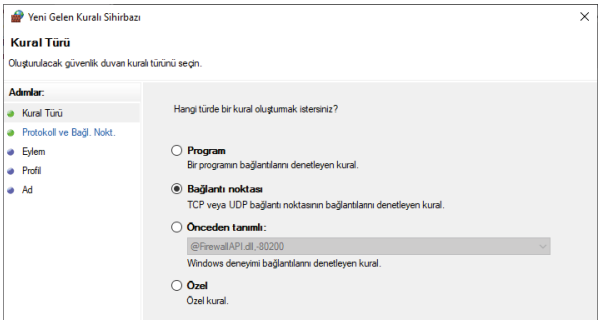
Görsel 6.39: Web sitesi ekleme penceresi

9. Adım: ASP.NET Core uygulamalarının IIS ile çalışmasına izin veren IIS runtime, kütüphaneler ve modülleri yükleyen "Barındırma Paketini" (Hosting Bundle) yüklemek için aşağıdaki adresi tarayıcınızda açınız.

https://dotnet.microsoft.com/permalink/dotnetcore-current-windows-runtime-bundle-installer

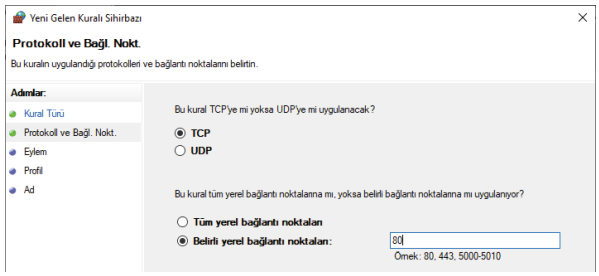
10. Adım: İndirilen paketinin kurulumunu tamamladıktan sonra tarayıcıda **localhost** yazarak ASP.NET Core uygulamasının çalıştırıldığını gözlemleyiniz.
11. Adım: Uygulamanızı internet veya yerel ağ üzerinden diğer kullanıcıların hizmetine açmak için Gelişmiş Güvenlik Özellikleri Windows Defender Güvenlik Duvarı uygulamasını açınız. Sol taraftaki menüde bulunan Gelen Kuralları'nı seçtikten sonra sağ menüde yer alan Yeni Kural linkine tıklayınız.

Açılan pencerede Görsel 6.40'daki Bağlantı Noktası seçeneğini seçerek İleri butonuna tıklayınız.



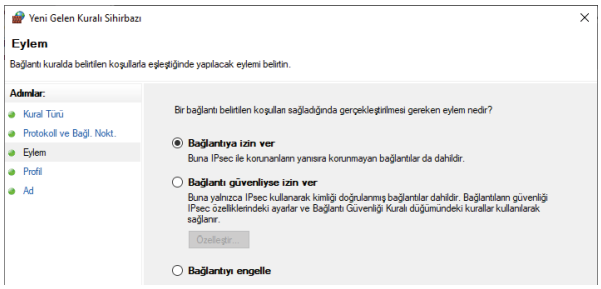
Görsel 6.40: Güvenlik kuralı kural türü penceresi

12. Adım: Görsel 6.41'de Protokol ve Port ayarlarının yapıldığı pencerede, **TCP** protokolü ve **80** numaralı port seçildikten sonra ileri butonuna tıklayınız.



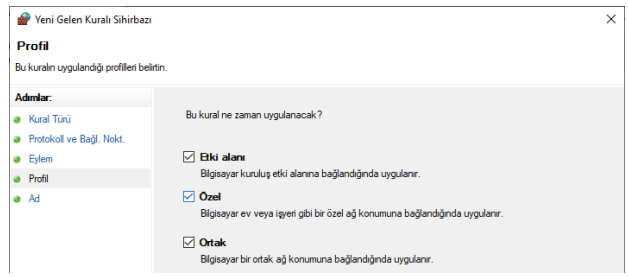
Görsel 6.41: Protokol ve port seçim penceresi

13. Adım: Görsel 6.42'de **Bağlantıya izin ver** (Allow the connection) seçeneğini seçerek İleri butonuna tıklayınız.



Görsel 6.42: Bağlantıya izin verme penceresi

14. Adım: Görsel 6.43'de kuralın uygulanacağı profillerin (Etki Alanı- Özel- Ortak) hangi ağlarda kullanılacağı belirlendikten sonra İleri butonu tıklayınız.



Görsel 6.43: Kuralın uygulanacağı profiller penceresi

15. Adım: Oluşturulan kurala herhangi bir isim verdikten sonra kural oluşturma işlemini tamamlayınız.
16. Adım: Aynı ağda bulunan diğer bilgisayarların tarayıcılarında bilgisayarınızın IP numarasını girerek uygulamanın çalıştığını gözlemleyiniz.

6.10.4. Web Servisler

Web servisler farklı platformların birbirleri arasında iletişim kurarak veri alışverişi yapmaları sağlayan yapılardır. Veri alışverişi için HTTP, HTTPS veya TCP protokolleri kullanılır. Platformlar birbirleriyle **XML (Extensible Markup Language)** veya **JSON (JavaScript Object Notation)** formatında mesajlar ile veri alışverişi sağlarlar.

6.10.5. Web Servis Çeşitleri

Web servis geliştirmek için günümüzde iki farklı yaklaşım kullanılır. Bunlardan birisi **REST (Representation State Transfer),** diğeri de **SOAP (Simple Object Access Protocol)** yaklaşımıdır.

REST mimarisi, istemci – sunucu arasındaki haberleşmeyi HTTP protokolü üzerinden gerçekleştiren bir mimaridir. İstemci ve sunucu arasında XML veya JSON verileri üzerinden haberleşme sağlanır. GET, POST, PUT, DELETE gibi HTTP metotları ile işlemler gerçekleştirilir. REST mimarisini kullanan servislere ise **RESTful** servis denir.

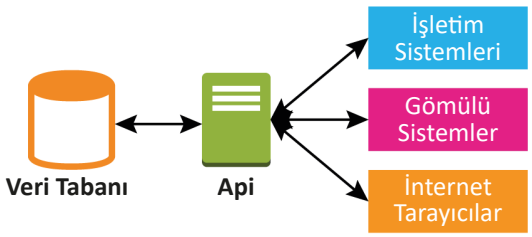
SOAP, istemci – sunucu arasındaki haberleşmeyi HTTP, TCP/IP ve SMTP üzerinden gerçekleştiren bir protokoldür. İstemci ve sunucu arasında XML verileri üzerinden haberleşme sağlanır. SOAP tabanlı bir servisinin nasıl kullanılması gerektiğini istemciye bildiren **WSDL** [Web Service Description Language (Web Servisleri Açıklama Dili)] dosyasının oluşturulması gerekir.

REST ile SOAP arasındaki farklar Tablo 6.9'da verilmiştir.

Tablo 6.9: SOAP ile REST Arasındaki Farklar

SOAP	REST
Bir protokoldür.	Mimari bir tarzdır.
Yalnızca XML veri formatına izin verir.	Düz metin, HTML, XML, JSON vb. farklı veri formatlarına izin verir.
HTTP, SMTP, UDP ve diğer transfer protokollerini kullanır.	Sadece HTTP protokolünü kullanır.
Yüksek güvenlik, standartlaştırılmış yapı, genişletilebilir.	Daha az güvenlik, ölçeklenebilirlik.
Kurumsal uygulamalar, yüksek güvenlikli uygulamalar, finansal hizmetler, ödeme ağ geçitleri, telekomünikasyon hizmetlerinde daha çok kullanılır.	Web servisler, mobil servisler, sosyal ağlar için daha çok kullanılır.
Daha çok bant genişliğine ve kaynak kullanır.	Daha az bant genişliğine ve kaynak kullanır.
Düşük performanslıdır.	Yüksek Performanslıdır.

ASP.NET Core ile hem REST hem de SOAP kullanılarak web servis geliştirilebilir. Bu öğrenme biriminde ASP.NET Core Web Api kullanarak REST mimarisi üzerinden RESTful servisler oluşturma işlemleri gerçekleştirilecektir. **API** (Application Programming Interface) "Uygulama Geliştirme Arayüzü" anlamına gelmektedir. Günümüzde internet erişimi farklı platformlarda ve cihazlarda bulunmaktadır dolayısıyla kullanıcıların ihtiyaçlarını sadece web sitelerinden sağlamaları pek mümkün değildir. Kullanıcıların ihtiyaç duydukları veriler veya işlevsellikler Api'ler sayesinde dış dünyaya açılır. Web Api'leri kullanarak farklı platformlar ve cihazlar, Şekil 6.3'de belirtildiği gibi ihtiyaç duydukları verilerden veya işlevselliklerden yararlanabilir.



Şekil 6.3: Web API'nin farklı platformlar ile etkileşimi

27. Uygulama

ASP.NET Core ile Web Api projesi yapma işlemini yönergeler doğrultusunda gerçekleştiriniz.

1. Adım: Dotnet komutları içinde Web Api şablonunu otomatik oluşturan komutlar bulunur. Fakat Web Api yapısını daha iyi kavrayabilmek için bilgisayarınıza KutuphaneApi adında klasör oluşturarak aşağıdaki dotnet komutu ile boş bir web projesi oluşturunuz.

```
> dotnet new web -o KutuphaneApi
```

2. Adım: KutuphaneApi uygulamasını çalıştığının kontrolü için aşağıdaki komutu terminal paneline yazdıktan sonra tarayıcının adres satırında http://localhost:5000 veya https://localhost:5001 yazarak sayfada "Hello World!" yazısını görüntüleyiniz.

```
> dotnet run
```

3. Adım: KutuphaneApi uygulaması içindeki Startup.cs dosyasını açınız. Açılan dosyada **ConfigureServices** metodu içinde KutuphaneApi uygulamasında Controller'ların kullanımı etkinleştirmek için **AddControllers()** servisini ekleme işlemini gerçekleştiriniz.

```
public void ConfigureServices(IServiceCollection services)
 {
    services.AddControllers();
 }
```

4. Adım: Startup.cs dosyasında C**onfigure** metodu içine gelen istekleri Controller'a iletmek için **MapControllers** uç noktasını (enpoint) ekleme işlemini gerçekleştiriniz.

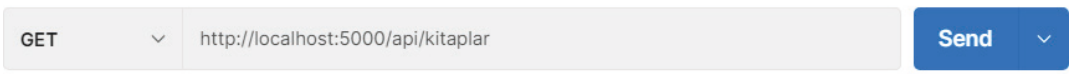
```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    if (env.IsDevelopment())
    {
      app.UseDeveloperExceptionPage();
    }
    app.UseRouting();

    app.UseEndpoints(endpoints =>
    {
       endpoints.MapControllers();
    });
}
```

5. Adım: KutuphaneApi uygulamasına Controllers adında klasör oluşturunuz. Oluşturulan klasörün içine **KitaplarController** adında Controller dosyası oluşturarak içine aşağıdaki kodlamaları yapınız.

```
using Microsoft.AspNetCore.Mvc;
namespace KutuphaneApi.Controllers
{
   [Route("api/[controller]")]
   [ApiController]
   public class KitaplarController : ControllerBase
   {
      [HttpGet]
      public IActionResult Get()
      {
         return Content("Tüm Kitaplar Gösterilecek...");
      }
      [HttpGet("{id}")]
      public IActionResult Get(int id)
      {
         return Content(id+". Numaralı Kitap Gösterilecek...");
      }
      [HttpPost]
      public IActionResult Post()
      {
         return Content("Kitap Ekleme İşlemi Gerçekleştirilecek...");
      }
      [HttpPut]
     public IActionResult Put()
      {
        return Content("Kitap Güncelleme İşlemi Gerçekleştirilecek...");
      }
```

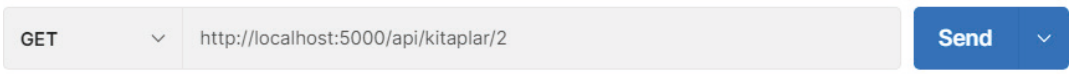
```
  [HttpDelete]
      public IActionResult Delete()
      {
          return Content("Kitap Silme İşlemi Gerçekleştirilecek...");
      }
   }
}
```

6. Adım: Görsel 6.44'deki Postman uygulamasını kullanarak KutuphaneApi 'ye gönderilen GET isteğinin (request) sonucu olan cevabı (response) gözlemleyiniz.



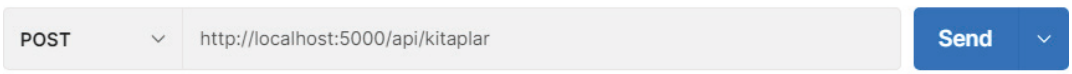
Görsel 6.44: API'ye gönderilen GET isteği

7. Adım: Görsel 6.45'deki Postman uygulamasını kullanarak KutuphaneApi 'ye parametre ile birlikte gönderilen GET isteğinin sonucu olan cevabı gözlemleyiniz.



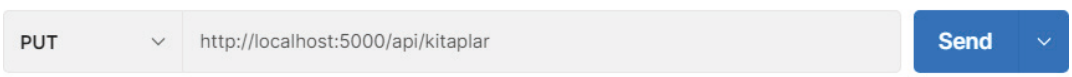
Görsel 6.45: API'ye parametre ile birlikte gönderilen GET isteği

8. Adım: Görsel 6.46'daki Postman uygulamasını kullanarak KutuphaneApi 'ye POST isteğinin sonucu olan cevabı gözlemleyiniz.



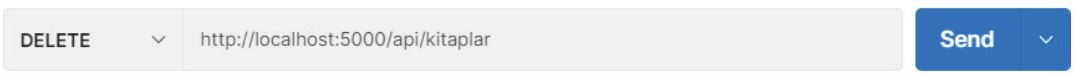
Görsel 6.46: API gönderilen POST isteği

9. Adım: Görsel 6.47'deki Postman uygulamasını kullanarak KutuphaneApi 'ye PUT isteğinin sonucu olan cevabı gözlemleyiniz.



Görsel 6.47: API gönderilen PUT isteği

10. Adım: Görsel 6.48'deki Postman uygulamasını kullanarak KutuphaneApi 'ye DELETE isteğinin sonucu olan cevabı gözlemleyiniz.



Görsel 6.48: API gönderilen DELETE isteği

Web Api'lerle genellikler veri kaynakları üzerinde işlem yapılır. Web Api'yi kullanarak veri tabanında **CRUD** (Create, Read, Update, Delete) işlemleri gerçekleştirilmesi RESTful Api'lerin en çok kullanım alanlarıdır.

Tablo 6.10'da verilen URL adresleri ve HTTP metotları kullanılarak yazılımcının sınırını belirlediği ölçüde işlemler gerçekleştirilir.

Tablo 6.10: HTTP Metotlarının Kullanımları

Metot	URL Adresi	İşlem	Açıklama
GET	/api/controller	Okuma	Verileri listelemek, görüntülemek için kullanılır.
GET	/api/controller/id	Okuma	Id numarasına göre tek kayıt görüntülemek için kullanılır.
POST	/api/controller	Oluşturma	Veri eklemek için kullanılır.
PUT	/api/controller/id	Değiştirme (Tamamı)	ID numarasına göre bir veriyi güncellemek için kullanılır.
PATCH	/api/controller/id	Değiştirme (Bölüm)	ID numarasına göre verinin bir parçasını güncellemek için kullanılır.
DELETE	/api/controller/id	Silme	ID numarasına göre bir veriyi silmek için kullanılır.

6.10.6. Web Servisler Katmanlı Mimari

Katmanlı mimari, bir yazılım geliştirme tekniğidir. Bir proje oluşturulduğunda projeyi katmanlara ayırarak birbirine referans gösterme mantığına dayalı mimaridir.

Katmanlı mimarinin faydaları şunlardır:

- DRY (Don't Repeat Yourself) kod tekrarı yapmamak.
- Başka uygulamalarda da kullanmak.
- Aynı projede bulunan mobil, web, servis gibi uygulamaları bütün hâlinde yönetmek.
- Tekrar kullanabilmek.

Katmanlı mimaride temel olarak üç katman bulunur. Bu katmanlar şunlardır:

- Veri Erişim Katmanı (Data Access Layer)
- İş Katmanı (Business Layer)
- Sunum Katmanı (Presentation Layer)

Veri Erişim Katmanı (Data Access Layer): Veri tabanı işlemlerinin gerçekleştirildiği katmandır. Bu katmanda veri tabanı ile ilgili ekleme, okuma, güncelleme ve silme işlemlerinin gerçekleştirildiği sınıflardan oluşur.

İş Katmanı (Business Layer): Veri erişim katmanı ile iletişime geçerek veriler ile ilgili işlemlerin, kodlamaların ve metotların bulunduğu katmandır.

Sunum Katmanı (Presentation Layer): Kullanıcı ile etkileşime geçildiği katmandır. Kullanıcıyı veri göndermek veya kullanıcıdan alınan verileri iş katmanı aracılığı ile veri erişim katmanına iletmek için kullanılır.

28. Uygulama

Katmanlı mimari kullanılarak web servis uygulaması işlemini yönergeler doğrultusunda gerçekleştiriniz.

1. Adım: Bilgisayarınıza Kutuphane isminde klasör oluşturunuz. Oluşturulan klasörü Visual Studio Code ile açınız. Katmanlı mimari yer alacak projelerimizi kapsayıcı görevi gören çözüm (solution) oluşturmak için terminal panelinde aşağıdaki komutu çalıştırınız.

```
> dotnet new sln
```

2. Adım: Projede kullanılacak olacak tüm model sınıfları için **Entities** adında katman oluşturulacaktır. Entities katmanı bir Class Library projesi olacaktır. Entities katmanını oluşturmak için aşağıdaki komutu terminal paneline yazınız.

Not

Class Library içine otomatik oluşturulan Class1.cs dosyasını siliniz.

```
> dotnet new classlib -o Entities
```

3. Adım: Oluşturulan Class Library projesini çözüme eklemek için aşağıdaki komutu kullanınız.

```
> dotnet sln add Entities/Entities.csproj
```

4. Adım 4: Entities klasörü içine **Kitap.cs** adında model sınıf dosyası oluşturarak içerisine aşağıdaki kodlamaları yapınız.

```
namespace Entities
{
   public class Kitap
   {
      public int KitapId { get; set; }
      public string KitapAdi { get; set; }
      public string KitapYazar { get; set; }
      public string KitapTur { get; set; }
   }
}
```

5. Adım: Bu adımda DataAccess adında Data Access Layer (Veri Erişim Katmanı) oluşturulacak ve bu katmanda veri tabanı işlemleri gerçekleştirilecektir. Bu katmanın görevi veriyi ekleme, silme, güncelleme ve veri tabanından çekme işlemini gerçekleştirmektir. DataAccess katmanı bir Class Library projesi olacaktır. DataAccess katmanını oluşturmak ve oluşturulan Class Library projesini çözüme eklemek için aşağıdaki komutu terminal paneline yazınız.

Not

Class Library içine otomatik oluşturulan Class1.cs dosyasını siliniz.

```
> dotnet new classlib -o DataAccess
> dotnet sln add DataAccess/DataAccess.csproj
```

6. Adım: Veri erişim katmanında Entitiy Framework Core ve SqlServer veri tabanı kullanılacaktır. Bu katmana veri tabanı sağlayıcısı için EntityFrameworkCore.SqlServer ve migrations için EntityFrameworkCore.Tools paketlerinin kurulumunu yapan aşağıdaki komutları terminal paneline yazınız.

```
> cd DataAccess
DataAccess > dotnet add package Microsoft.EntityFrameworkCore.SqlServer
DataAccess >  dotnet add package Microsoft.EntityFrameworkCore.Tools
```

7. Adım: Entities katmanındaki model sınıflarının DataAccess katmanında kullanabilmesi için bağlanmaları gerekmektedir. Bu bağlanma işlemi referans ekleme ile gerçekleşir. DataAccess katmanı içindeki DataAccess.csproj dosyasına aşağıdaki kodları ekleyerek bağlantıyı gerçekleştiriniz.

```
<Project Sdk="Microsoft.NET.Sdk">
 <ItemGroup>
  <ProjectReference Include="..\Entities\Entities.csproj" />
 </ItemGroup>
 ...
```

8. Adım: Bu adımda DataAccess katmanına Entity Framework Core'un kalbi olan DbContext sınıfı oluşturulacaktır. DbContext sınıfında veri tabanı olarak SqlServer LocalDb kullanılacaktır (Tercih ettiğiniz farklı veri tabanını da kullanabilirsiniz). DataAccess katmanına **KutuphaneDbContext.cs** adında DbContext sınıfı oluşturarak aşağıdaki kodlamaları yapınız.

```
using Microsoft.EntityFrameworkCore;
using Entities;
namespace DataAccess
{
   public class KutuphaneDbContext : DbContext
   {
     string baglanti = "Server=(localdb)\\MSSQLLocalDB;Database=KutuphaneData;Trusted_Connection=True;";
      protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
      {
        optionsBuilder.UseSqlServer(baglanti);
      }
      public DbSet<Kitap> Kitaplar { get; set; }
   }
}
```

9. Adım: DbContext sınıfı içerisinde **KutuphaneData** adında veri tabanı ve bu veri tabanı aktarılmak üzere **Kitaplar** adında **Kitap** sınıfından nesne kullanılmıştır. Veri tabanı oluşturmak ve DbSet türü verilmiş Kitaplar tablosu oluşturmak için aşağıdaki komutları kullanarak migrations işlemlerini gerçekleştiriniz.

```
DataAccess > dotnet ef migrations add ilk
DataAccess > dotnet ef database update
```

10. Adım: DataAccess katmanına **Abstract** adında bir klasör oluşturunuz.

Abstract klasörü içine **IKitapRepository.cs** adında interface dosyası oluşturarak kodlamaları yapınız.

Not

Abstract klasörü içerisinde Interface sınıflar oluşturularak veri tabanında CRUD işlemleri için hangi metotların kullanılacağının tanımı yapılacaktır.

Not

Interface sınıf adlarının başına "I" harfi eklenir.

```
using System.Collections.Generic;
using Entities;

namespace DataAccess.Abstract
{
   public interface IKitapRepository
   {
      List<Kitap> GetAll();
      Kitap GetById(int id);
      string Create(Kitap kitap);
      string Update(Kitap kitap);
      void Delete(int id);
   }
}
```

GetAll Metodu: Veri tabanındaki tüm kayıtları çekerek, Kitap modeline uyarlayarak, List koleksiyonu olarak döndürecektir.

GetById Metodu: Veri tabanındaki ID numarasına göre bulunan kaydı, **Kitap** modeline uyarlayarak döndürecektir.

Create Metodu: Kendisine parametre olarak gönderilen Kitap modelini veri tabanına kaydettikten sonra string değer döndürecektir.

Not

Döndürülen string değer, mesaj verme amacı ile kullanılacaktır. **void** veya **int** kullanılabilir.

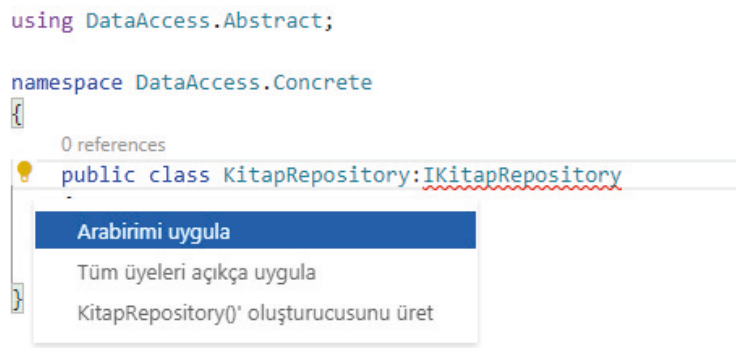
Update Metodu: Kendisine parametre olarak gönderilen Kitap modelini, veri tabanında bularak güncelledikten sonra string değer döndürecektir.

Delete Metodu: Veri tabanındaki ID numarasına göre bulunan kaydı silme işlemi gerçekleştirecek geriye herhangi bir değer döndürme işlemi yapmayacaktır.

11. Adım: DataAccess katmanına **Concrete** adında bir klasör oluşturunuz. Concrete klasörü içine **KitapRepository.cs** adında sınıf dosyası oluşturunuz. Görsel 6.49'daki gibi IKitapRepository Interface'ini kullanarak **implement** işlemi uygulayınız.

Not

Implement işlemi sonrasında IKitapRepository interface içindeki tüm metotlar otomatik oluşturulacaktır.



Görsel 6.49: Veri erişim katmanı interface implement işlemi

12. Adım: KitapRepository dosyası içerisinde veri tabanı ile ilgi işlemlerin yapılacağı kodlar bulunur. IKitapRepository dosyasından implement edilen metotlarda çalışacak olan kodlamaları yapınız.

```
using System.Collections.Generic;
using System.Linq;
using DataAccess.Abstract;
using Entities;

namespace DataAccess.Concrete
{
   public class KitapRepository : IKitapRepository
   {
      public string Create(Kitap kitap)
      {
         using (var context = new KutuphaneDbContext())
         {
```

```
            context.Kitaplar.Add(kitap);
            context.SaveChanges();
            return "Kayıt Oluşturma Başarılı.";
        }
    }

    public void Delete(int id)
    {
        using (var db = new KutuphaneDbContext())
        {
            var kitap = GetById(id);
            db.Remove(kitap);
            db.SaveChanges();
        }
    }


    public Kitap GetById(int id)
    {
        using (var db = new KutuphaneDbContext())
        {
            return db.Kitaplar.Find(id);
        }
    }

    public string Update(Kitap kitap)
    {
        using (var db = new KutuphaneDbContext())
        {
            db.Kitaplar.Update(kitap);
            db.SaveChanges();
            return "Kayıt Güncelleme Başarılı.";
        }

    }
  }
}
```

13. Adım: Bu adımda **Business** adında Business Layer (İş Katmanı) oluşturulacaktır. Katmanlı mimaride veri erişim katmanı direkt olarak kullanılmaz. Sunum katmanından gelen bilgileri veri erişim katmanına aynı şekilde veri erişim katmanından gelen bilgileri sunum katmanına iletmekle görevli katmandır. Business katmanı bir Class Library projesi olacaktır. Business katmanını oluşturmak ve oluşturulan Class Library projesini çözüme eklemek için aşağıdaki komutu terminal paneline yazınız.

```
> dotnet new classlib -o Business
> dotnet sln add Business/Business.csproj
```

14. Adım: Entities katmanındaki model sınıflarını ve DataAccess katmanındaki metotları Business katmanında kullanabilmek için bağlanması gerekmektedir. Business katmanı içindeki Business.csproj dosyasına aşağıdaki kodları ekleyerek bağlantıyı gerçekleştiriniz.

```
<Project Sdk="Microsoft.NET.Sdk">
 <ItemGroup>
  <ProjectReference Include="..\Entities\Entities.csproj" />
  <ProjectReference Include="..\DataAccess\DataAccess.csproj" />
 </ItemGroup>
 ...
```

15. Adım: İş Katmanı içerisine Abstract ve Concrete adında iki klasör oluşturunuz. Abstract klasöründe Interface dosyaları, Concrete klasöründe ise Interface dosyalarından implement edilen sınıflar yer alacaktır. Abstract klasörüne **IKitapService.cs** adında interface oluşturarak aşağıdaki kodlamaları yazınız.

Not

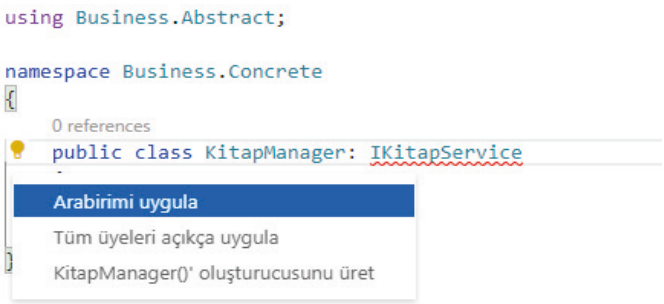
Business katmanından **Interface** dosya isimleri **Service** kelimesi eklenerek kullanılacaktır.

```
using System.Collections.Generic;
using Entities;

namespace Business.Abstract
{
   public interface IKitapService
   {
      List<Kitap> KitapGetir();
      Kitap KitapGetir(int id);
      string KitapEkle(Kitap kitap);
      string KitapGuncelle(Kitap kitap);
      void KitapSil(int id);
   }
}
```

16. Adım: Business Katmanındaki Concrete klasörünün içerisine **KitapManager.cs** adında bir sınıf dosyası oluşturunuz. Oluşturulan sınıf dosyasını Görsel 6.50'deki gibi IKitapService interface'inden implement işlemini gerçekleştiriniz.

Not

Business katmanından interface dosyasından implement edilen sınıf dosyalarının isimleri **Manager** kelimesi eklenerek kullanılacaktır.



Görsel 6.50: İş katmanı interface implement işlemi

17. Adım: Bu adımda DataAccess katmanı ile iletişime geçilerek DataAccess katmanında kullanılabilecek metotların kullanılması gerçekleştirilecektir. Bu metotları kullanabilmek için **_kitapRepository** adında nesne kullanılmaktadır. Bu nesne, Interface türündedir fakat metot sınıfından örnek alınmıştır. Aşağıdaki kodlamaları yaparak DataAccess katmanı içerisindeki metotları kullanarak işlem yapılmasını sağlayan kodlamaları yazınız.

```
using System.Collections.Generic;
using Business.Abstract;
using DataAccess.Abstract;
using DataAccess.Concrete;
using Entities;

namespace Business.Concrete
{
   public class KitapManager : IKitapService
   {
      private IKitapRepository _kitapRepository = new KitapRepository();
      public string KitapEkle(Kitap kitap)
      {
         return _kitapRepository.Create(kitap);
      }

      public List<Kitap> KitapGetir()
      {
         return _kitapRepository.GetAll();
      }
```

```
        public Kitap KitapGetir(int id)
        {
            return _kitapRepository.GetById(id);
        }

        public string KitapGuncelle(Kitap kitap)
        {
            return _kitapRepository.Update(kitap);
        }

        public void KitapSil(int id)
        {
            _kitapRepository.Delete(id);
        }
    }
}
```

18. Adım: Business katmanı DataAccess katmanından farklı olarak mantıksal kodlamaların yapıldığı katmandır. Aşağıda mantıksal kodlamalardan birkaç örnek verilmiştir.

```
public Kitap KitapGetir(int id)
{
    if (id > 0)
    {
        return _kitapRepository.GetById(id);
    }
    return null;
}
public string KitapGuncelle(Kitap kitap)
{
    if (kitap.KitapId == null || kitap.KitapId< 0)
    {
        return "Kitap Id null veya sıfırdan küçük";
    }
    return _kitapRepository.Update(kitap);
}
```

19. Adım: RESTful servis katmanı için **API** adında web projesini ekleyerek Web Servis Katmanını oluşturunuz. Oluşturulan Web Servis Katmanını çözüme ekleyiniz.

```
> dotnet new web -o Api
> dotnet sln add Api/Api.csproj
```

20. Adım: Api klasörü içindeki Startup.cs dosyasını açarak Uygulama 27'deki Adım 3 ve Adım 4'teki işlemleri gerçekleştiriniz.

21. Adım: Entities katmanındaki model sınıflarını ve Business katmanındaki metotları Api katmanında kullanabilmek için bağlanmaları gerekmektedir. Api katmanı içindeki Api.csproj dosyasına aşağıdaki kodları ekleyerek bağlantıyı gerçekleştiriniz.

```
<Project Sdk="Microsoft.NET.Sdk">
 <ItemGroup>
  <ProjectReference Include="..\Entities\Entities.csproj" />
  <ProjectReference Include="..\Business\Business.csproj" />
 </ItemGroup>
 ...
```

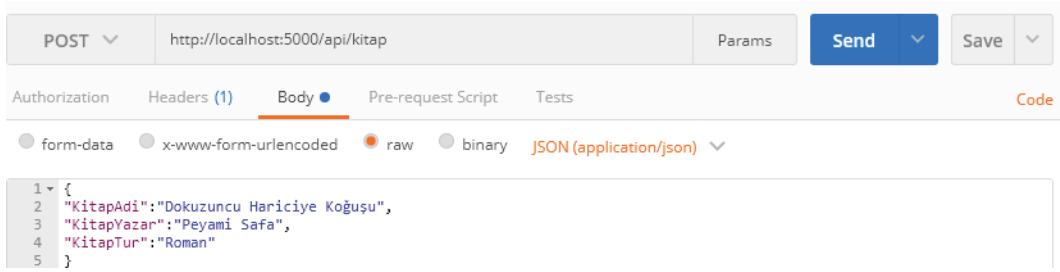
22. Adım: Api klasörüne **Controllers** adında klasör oluşturunuz. Controllers klasörü içine **KitapController** adında controller sınıfı oluşturarak aşağıdaki kodlamaları yapınız.

```
using System.Collections.Generic;
using Business.Abstract;
using Business.Concrete;
using Entities;
using Microsoft.AspNetCore.Mvc;

namespace Api.Controllers
{
   [Route("api/[controller]")]
   [ApiController]
   public class KitapController : ControllerBase
   {
      private IKitapService _kitapService = new KitapManager();
      [HttpGet]
      public List<Kitap> Get()
      {
         return _kitapService.KitapGetir();
      }
      [HttpGet("{id}")]
      public Kitap Get(int id)
      {
         return _kitapService.KitapGetir(id);
      }
      [HttpPost]
      public string Post(Kitap kitap)
      {
         return _kitapService.KitapEkle(kitap);
      }
```

```
        [HttpPut]
        public string Put(Kitap kitap)
        {
            return _kitapService.KitapGuncelle(kitap);
        }
        [HttpDelete("{id}")]
        public void Delete(int id)
        {
            _kitapService.KitapSil(id);
        }
    }
}
```

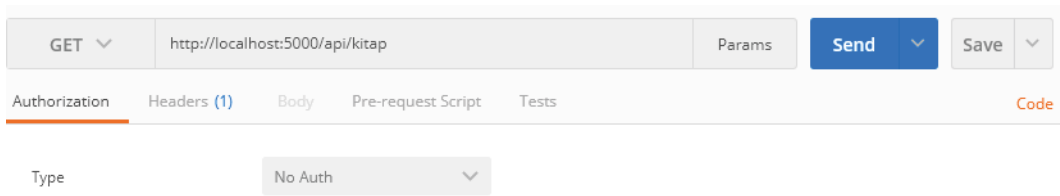
23. Adım: Postman uygulaması açarak **Headers** sekmesindeki **Content-Type** özelliğinin değerini **application/json** olarak değiştiriniz.

24. Adım: Uygulamada Görsel 6.51'deki POST metodunu kullanarak veri ekleme işlemini gerçekleştiriniz.



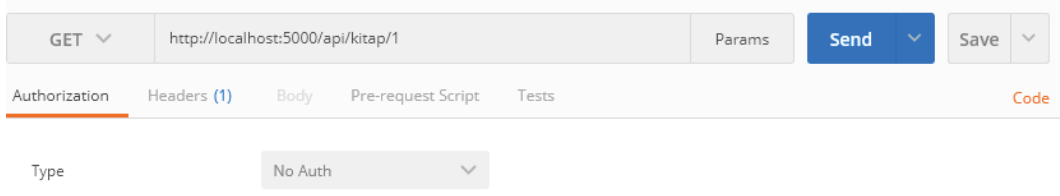
Görsel 6.51: POST ile veri ekleme işlemi

25. Adım: Uygulamada Görsel 6.52'deki GET metodunu kullanarak veri çekme işlemini gerçekleştiriniz.



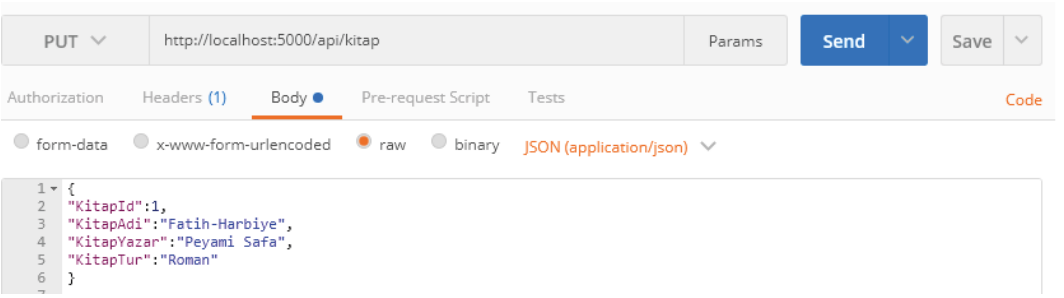
Görsel 6.52: GET metodu ile veri çekme işlemi

26. Adım: Uygulamada Görsel 6.53'deki GET metodunu kullanarak **id** parametresine göre veri çekme işlemini gerçekleştiriniz.



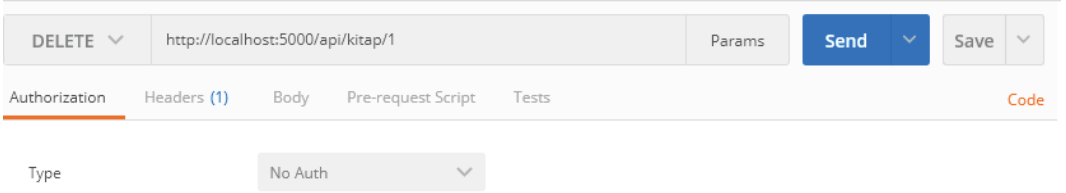
Görsel 6.53: GET metodu id parametresi ile veri çekme işlemi

27. Adım: Uygulamada Görsel 6.54'deki PUT metodunu kullanarak veri güncelleme işlemini gerçekleştiriniz.



Görsel 6.54: PUT metodu veri güncelleme işlemi

28. Adım: Uygulamada Görsel 6.55'deki DELETE metodunu kullanarak **id** parametresine göre veri silme işlemini gerçekleştiriniz.



Görsel 6.55: DELETE metodu id parametresi ile veri silme işlemi

ÖLÇME VE DEĞERLENDİRME

A. Aşağıdaki cümlelerde boş bırakılan yerlere doğru sözcükleri yazınız.

1. HTML ve C# kodlarının bir arada çalıştırılabilmesini sağlayan yapıya …… denir.

2. Startup.cs'teki configure metodundaki sırayı ve yanıtın ters sırasını tanımlayan yapıya …… denir.

B. Aşağıdaki soruları dikkatlice okuyunuz ve doğru seçeneği işaretleyiniz.

3. Aşağıdakilerden hangisi .NET'in sunduğu avantajlardan biri değildir?

A) Çapraz platform desteği sağlanması B) Yüksek performans sağlaması
C) Kapalı kaynak kodlu olması D) Birlikte çalışmayı kolaylaştırması
E) Birden fazla .NET sürümünün yan yana çalışabilmesi

4. Hangisi Startup.cs dosyasının özelliklerinden biri değildir?

A) ASP.NET Core uygulamalarının başlangıç noktasıdır.
B) Her ASP.NET Core projesinde bulunmak zorundadır.
C) Program.cs içerisinden çağrılır.
D) Uygulama ayarlarının yapıldığı yerdir.
E) Adının Startup olması zorunludur.

5. Aşağıdakilerden hangisi standart klasör ve dosyalar arasında yer almaz?

A) wwwroot B) Views > _ViewStart.cshtml C) Program.cs
D) Settings.json E) Startup.cs

6. Formlarla ilgili aşağıdaki bilgilerden hangisi yanlıştır?

A) Veri girişleri input etiketleri ile sağlanır.
B) Kullanıcıdan alınan verileri sunucuya aktarır.
C) Girilen verileri göndermek için submit butonu kullanılır.
D) Kullanıcı ile web siteleri arasında etkileşim için kullanılır.
E) Formlar içerisinde sadece input etiketleri kullanılır.

7. Formdan gönderilen dosyayı sunucu tarafında alabilmek için aşağıdakilerden hangisi kullanılır?

A) CopyTo() B) FileFormData C) FormData D) IFormCollection E) IFormFile

8. POST metodu ile ilgili aşağıdaki bilgilerden hangisi doğrudur?

A) Veriler adres satırında görünür
B) Veri taşıma sınırı vardır
C) Veri gönderimi güvenlidir
D) Varsayılan metottur
E) Sadece metin verileri taşır

9. Aşağıdakilerden hangisi TagHelper'ların özelliklerinden değildir?

A) HTML etiketlerine benzer. B) Kurulum yapılarak kullanılır.
C) InteliSense özelliği kullanılır. D) Temiz ve okunabilir kodlama yapılır.
E) Hazır metotlar bulundurur.

10. Aşağıdakilerden hangisi Form TagHelper niteliklerinden değildir?

A) asp-action B) asp-antiforgery C) asp-controller D) asp-path E) asp-route

11. Label TagHelper'lar HTML Label etiketlerine uygulanırken aşağıdaki Data Annotations niteliklerinden hangisi kullanılır?

A) DataType B) Display C) DisplayFormat D) EmailAddress E) Key

12. Sunucu taraflı doğrulama ile ilgili aşağıdaki verilen bilgilerden hangisi doğrudur?

A) Sunucu kaynaklarını kullanmaz
B) Doğrulama tarayıcıda gerçekleştirilir
C) Güvenli değildir
D) Javascript kullanılır
E) Model doğrulama kullanılır

13. Entity Framework Core'da veri tabanı ile ilgili işlemleri gerçekleştiren sınıf aşağıdakilerden hangisidir?

A) DbContext B) DbSession C) DbSet D) Entity E) LINQ

14. Aşağıda verilen model doğrulama nitelikleri açıklamalarından hangisi doğrudur?

A) StringLength, model değerinin sayısal uzunluğunu ifade eder.
B) Compare, iki model değerinin karşılaştırma durumunu gösterir.
C) Range, model değerinin tarih aralığını gösterir.
D) MaxLength, minimum karakter uzunluğunu gösterir.
E) Required, modele değeri boş olabilir.

15. Sunucu taraflı model doğrulama işleminde girilen verilerin doğruluğunun kontrolü için aşağıdakilerden hangisi kullanılır?

A) ModelState.Errors B) ModelState.IsValid C) Validation.Check
D) Validation.IsValid E) Validation.Summary

16. Aşağıdakilerden hangisi paket yükleme komutudur?

A) dotnet package install B) nuget add package
C) dotnet add package D) nuget package add
E) dotnet install package

17. Aşağıdakilerden hangisi Entity Framework Core için doğru bir ifadedir?

A) MVC Framework'üdür. B) Açık kaynak ORM aracıdır.
C) Veri tabanı yönetim aracıdır. D) Object Mapping aracıdır.
E) İlişkisel veri tabanıdır.

18. Aşağıdakilerden hangisi bir LINQ sorgusu <u>değildir</u>?

A) FirstOrDefault B) OrderBy C) SaveChange
D) Select E) ToList

19. Ülkelerin ve başkentlerin bilgilerinin saklandığı iki farklı tablo arasında ilişki kurulacak olur ise ilişki türü olarak aşağıdakilerden hangisi olur?

A) Bire – Bir B) Bire - Çok C) Bire - Sıfır D) Çoka - Bir E) Çoka – Çok

20. ASP.NET Core WebApi aşağıdaki protokollerden hangisini kullanır?

A) HTTP B) SMTP C) SOAP D) TCP E) WSDL

C. Aşağıdaki soruyu dikkatlice okuyunuz ve cevaplayınız.

21. ASP.NET Core'da yönlendirme nasıl gerçekleşir?

KAYNAKÇA

Bilişim Teknolojileri Alanı Çerçeve Öğretim Programı

Alscher, D. (2019). The 6 color schemes to keep everything picture perfect. https://learn.g2.com/color-schemes, Şubat 10, 2021, kaynağından alınmıştır.

Anderson, R. (2020). Öğretici: kullanmaya başlayın Razor Core'da sayfalar Asp.Net kullanma. https://docs.microsoft.com/tr-tr/aspnet/core/tutorials/razor-pages/razor-pages-start?view=aspnetcore-5.0&tabs=visual-studio, Nisan 10, 2021, kaynağından alınmıştır.

Anderson, R. (2021). Asp.Net Core MVC ile çalışmaya başlama. https://docs.microsoft.com/tr-tr/aspnet/core/tutorials/first-mvc-app/start-mvc?view=aspnetcore-5.0&tabs=visual-studio, Nisan 5, 2021, kaynağından alınmıştır.

Anderson, R. ve Smith, S. (2020). Asp.Net Core ara yazılımı. https://docs.microsoft.com/tr-tr/aspnet/core/fundamentals/middleware/?view=aspnetcore-5.0, Nisan 10, 2021, kaynağından alınmıştır.

Becer, E. (1997). İletişim ve grafik tasarım. Ankara: Dost Kitabevi Yayınları.

Bilirdönmez, K. (2020). Tipografide Renk ve Rengin Kullanımı. Journal of Humanities and Tourism Research, 10(4), 863-883. https://dergipark.org.tr/en/download/article-file/1470370, Şubat 10, 2021, kaynağından alınmıştır.

Bootstrap, homepage. (2021). https://getbootstrap.com/, Mart 5, 2021, kaynağından alınmıştır.

Dikener, O. (2011). İnternet Reklamcılığında Web Sitesi Tasarımının Önemi. Erciyes İletişim Dergisi "akademia", 2(1), 152-166. https://dergipark.org.tr/tr/download/article-file/66311, Şubat 15, 2021, kaynağından alınmıştır.

Eğitim Bilişim Ağı (EBA). (t.y.). https://www.eba.gov.tr

jQuery, homepage. (2021). https://jquery.com/, Nisan 10, 2021, kaynağından alınmıştır.

MEB Meslekî ve Teknik Eğitim Genel Müdürlüğü. (t.y.) mtegm.meb.gov.tr

Nowak, R., Larkin, K., ve Anderson, R. (2020). Asp.Net Core yönlendirme. https://docs.microsoft.com/tr-tr/aspnet/core/fundamentals/routing?view=aspnetcore-5.0, Nisan 10, 2021, kaynağından alınmıştır.

Roth, D., Anderson, R., ve Shaun, L. (2020). Asp.Net core'a giriş. https://docs.microsoft.com/tr-tr/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-5.0, Nisan 2, 2021, kaynağından alınmıştır.

Şenel, S., Şenel, H. C., ve Günaydın, S. (2019). Herkes için mobil öğrenme: Mobil uygulamaların evrensel tasarım ilkelerine göre incelenmesi. Ankara Üniversitesi Eğitim Bilimleri Fakültesi Özel Eğitim Dergisi, 20(1), 73-92. https://dergipark.org.tr/en/download/article-file/660477, Şubat 20, 2021, kaynağından alınmıştır.

Türk Dil Kurumu, sözlükleri. (t.y.). https://sozluk.gov.tr/

Türk Dil Kurumu. (t.y.). https://www.tdk.gov.tr/

W3C, homepage. (2021). https://www.w3.org/, Mart 1, 2021, kaynağından alınmıştır.

W3Schools, homepage. (2021). https://www.w3schools.com/, Mart 1, 2021, kaynağından alınmıştır.

***Kaynakça kısmı APA6 referanslama sistemi kullanılarak oluşturulmuştur.**

GÖRSEL KAYNAKÇA

http://kitap.eba.gov.tr/karekod/Kaynak.php?KOD=2398

ÖĞRENME BİRİMLERİ
ÖLÇME VE DEĞERLENDİRME CEVAP ANAHTARLARI

ÖĞRENME BİRİMİ 1'İN CEVAP ANAHTARI

1. (B), (A), (D), (C), (E)
2. CSS
3. ARKA UÇ/BACKEND
4. B
5. A

ÖĞRENME BİRİMİ 2'NİN CEVAP ANAHTARI

1. D
2. Y
3. Y
4. D
5. D
6. (B), (D), (G), (C)

ÖĞRENME BİRİMİ 3'ÜN CEVAP ANAHTARI

1. D
2. Y
3. D
4. Y
5. D
6. D
7. E
8. B
9. A
10. E

ÖĞRENME BİRİMİ 4'ÜN CEVAP ANAHTARI

1. Inline
2. Releative
3. Z-index
4. Kırmızı
5. 0-255 arası
6. Padding, margin, border, content (içerik)
7. Satır içi, sayfa içi, sayfa dışı
8. Çocuk seçici, belirtilen elemanın bir alt seviyesinde bulunan tüm elemanlar arasından eşleşenleri seçer. Torun seçici ise belirtilen elemanın kapsadığı tüm elemanlar arasından eşleşenleri seçer.
9. % (Yüzde), em, vw (Ekran Genişliği), vh (Ekran Yüksekliği)
10.

```
<img src="resim1.jpg" class="rounded">
<img src="resim2.jpg" class="rounded-circle">
<img src="resim3.jpg" class="img-thumbnail">
```

ÖĞRENME BİRİMİ 5'İN CEVAP ANAHTARI

1. Sınıf (class) seçici
2. innerHTML
3. Dört
4. Return;
5. if yapısı, if else yapısı, else if yapısı, switch case yapısı
6. var sehir2="İstanbul";
var sehir3="Ankara";
var sehir4="Bursa";
var sehir5="Ordu";
7. Mesaj olarak 12 sayısını verir.
8.

```
<script>
  $(document).ready(function(){
    $("#ekran").css({"backgroundColor":"gray","color":"orange"});
  });
</script>
```

ÖĞRENME BİRİMİ 6'NIN CEVAP ANAHTARI

1. Razor View Motoru
2. İşlem hattı (Pipeline)
3. C
4. E
5. D
6. E
7. E
8. C
9. B
10. D
11. B
12. E
13. A
14. B
15. B
16. C
17. B
18. C
19. A
20. A
21. Yönlendirme, gelen isteklerin hangi controller ve hangi action metoda yönlendirileceğini belirleyen ASP.NET Core'daki önemli bileşenlerden birisidir.

NOTLAR: .....

NOTLAR: .....