WEB KARTOGRAFYA

Harita Mashup Hizmetleri Kullanarak İnternet Yoluyla Harita Yayınlama

HTML, CSS, JavaScript, Leaflet

İbrahim Öztuğ BİLDİRİCİ

Konya Teknik Üniversitesi Mühendislik ve Doğa Bilimleri Fakültesi
Harita Mühendisliği Bölümü
KONYA

2021

İçindekiler

1 Giriş **1**

1.1 Ders Web Sayfası ve Editörler 1

2 HTML DİLİ **3**

2.1 Giriş 3

2.2 HTML Elemanları 4

2.3 HTML Öznitelikleri 5

2.4 HTML Başlıkları 6

2.5 HTML Paragrafları 6

2.6 Format Bildirim Etiketleri 7

2.7 Yorum Satırları 7

2.8 HTML Bağlantıları 7

2.9 Head Elemanı 8

2.10 HTML Stilleri (CSS) 10

2.10.1 Inline Stiller 11

2.10.2 Dahili (Internal) Stil Sayfası 11

2.10.3 Harici (External) Stil Sayfası 11

2.11 HTML Resimleri 12

2.12 Tablolar 12

2.13 Listeler 15

2.14 HTML Blokları (div ve span) 17

2.15 HTML Sayfa Düzeni . . . 17
2.16 HTML Formları ve Giriş . . . 19
2.17 Iframe . . . 20
2.18 HTML Renkleri . . . 21
2.19 HTML Betikleri (Skriptleri) . . . 21
2.20 HTML Karakter Varlıkları . . . 22
2.21 Karakter Setleri . . . 23
2.22 URL . . . 23

3 CSS **25**

3.1 CSS Söz Dizimi . . . 26
3.2 CSS Seçicileri . . . 26
3.3 CSS Nasıl Eklenir? . . . 27
3.4 CSS Yorum Satırları . . . 27
3.5 CSS Renkleri . . . 28
3.6 Arka Plan . . . 29
3.7 Sınırlar . . . 30
3.8 Marjlar . . . 31
3.9 İç Boşluk: Padding . . . 31
3.10 Yükseklik ve Genişlik . . . 31
3.11 Box Model . . . 32
3.12 Yazı . . . 33
3.12.1 Fontlar . . . 34
3.13 Bağlantı (Link) . . . 36
3.14 Listeler . . . 37
3.15 Tablolar . . . 38
3.16 CSS Sayfa Düzeni (Layout) . . . 39

4 JavaScript Programlama Dili **41**

4.1 Giriş . . . 41
4.2 JavaScript Kodlarının Yeri . . . 42
4.3 JavaScript'de Çıkış . . . 43
4.4 JavaScript Sözdizimi . . . 44

4.5 Deyimler . . . 47
4.6 Yorum Satırları . . . 48
4.7 Değişkenler . . . 49
4.8 Veri Türleri . . . 50
4.9 Objeler . . . 52
4.10 Fonksiyonlar . . . 53
4.11 Faaliyet Alanı (Scope) . . . 54
4.12 Olaylar (Events) . . . 55
4.13 Dizgeler . . . 56
4.14 JavaScript'te Sayılar . . . 60
4.15 Operatörler . . . 62
4.16 Math Objesi . . . 64
4.17 Date Objesi . . . 65
4.18 Diziler . . . 67
4.19 Mantıksal Değerler . . . 69
4.20 Tür Dönüşümleri . . . 70
4.21 Koşul Deyimleri . . . 71
4.22 Döngüler . . . 74

5 HTML Grafiği **79**

5.1 SVG . . . 79
5.1.1 SVG: Dikdörtgen . . . 80
5.1.2 SVG: Daire ve Elips . . . 81
5.1.3 SVG: Doğru, Çokludoğru ve Çokgen . . . 81
5.1.4 SVG: Çizgi (Path) . . . 81
5.1.5 SVG: Yazı . . . 82
5.2 Canvas . . . 82

6 Leaflet JavaScript Kütüphanesi **85**

6.1 Leaflet: Başlangıç . . . 85
6.2 Map Objesi . . . 88
6.3 Leaflet'de Bindirmeler . . . 89
6.3.1 Marker . . . 90

6.3.2 Çoklu Doğru 91
6.3.3 Poligon ve Daire 91
6.3.4 Bilgi Penceresi - Popup 91
6.4 Temel Objeler 92
6.4.1 latLng 92
6.4.2 LatLngBounds 92
6.4.3 Point 93
6.5 Olaylar 94
6.6 GeoJSON Kullanımı 95
6.6.1 GeoJSON Objelerinin Dış Dosyalarda Bulunması 99
6.6.2 GeoJSON Tematik Harita Uygulaması 99

Bölüm 1

Giriş

Günümüz bilim ve teknoloji düzeyinde olmasa da insanoğlunun haritalar yapması ve kullanması insanlık tarihi kadar eskidir. İlk haritalara tarih öncesi dönemde rastlanmaktadır. Haritalar ilk çağlardan beri analog materyaller üzerine (kağıt vb.) çizilmektedir. İlk haritalar çok az sayıda kopyadan hatta tek nüshadan oluşuyordu. Matbaanın icadı ve kitapların çoğaltılarak kitlelere ulaşmasına paralel olarak haritalar da çoğaltılmaya başlandı. Teknolojideki ilerlemelere paralel olarak İnternet, haritaların da kitlelere ulaşmasının bir aracı haline geldi. Bu kapsamda Web2.0, harita *mashupları* gibi kavramlar ortaya çıktı.

Bu ders notu İnternet yolu ile harita yayınlama konusundaki teknolojileri ele almaktadır. Konu beş ana bölüm halinde ele alınmıştır. İkici bölüm HTML sayfa tanımlama dilini, üçüncü CSS, dördüncü JavaScript programlama dili, son bölüm Leaflet Javascript kütüphanesi konularını ele almaktadır. İlk dört bölüm `http://www.w3schools.com` sitesinden, son bölüm Leaflet Web sayfalarından derlenmiştir (Leaflet, 2018).

1.1 Ders Web Sayfası ve Editörler

Ders notu, örnek kodlar ve dersle ilgili diğer dokümanlar ders web sayfasında yer almaktadır (`http://galileo.ktun.edu.tr/1205641`).

HTML ve Javascript kodlarını yazmak için bir metin editörü programına ihtiyaç vardır. Bu amaçla Windows ortamında Not Defteri (Notepad), Wordpad gibi işletim sistemi ile gelen yazılımlar kullanılabilir. Kodların oluşturduğu web sayfaları ise web tarayıcılar (İnternet Explorer, Google Chrome vb) ile görülebilir. Linux işletim sistemi kullanılması durumunda daha farklı çözümler mevcuttur. Derste yapılan uygulamalarda Windows ortamında ücretsiz bir yazılım olan PsPad kullanılması önerilmektedir (`http://www.pspad.com`). Yine ücretsiz bir yazılım olan Notepad++ da (`http://notepad-plus-plus.org`) kullanılabilir.

Bölüm 2

HTML DİLİ

2.1 Giriş

HTML kısaltmasının açılımı Hypertext Markup Language olup, Web sayfalarını tanımlamak için geliştirilmiş bir sayfa tanımlama dildir. Sayfa tanımlama dili (markup language) doküman (sayfa) içeriğini tanımlayan etiketlerden oluşur. HTML dili düz metin ve etiketlerden oluşur.

Web Tarayıcıları

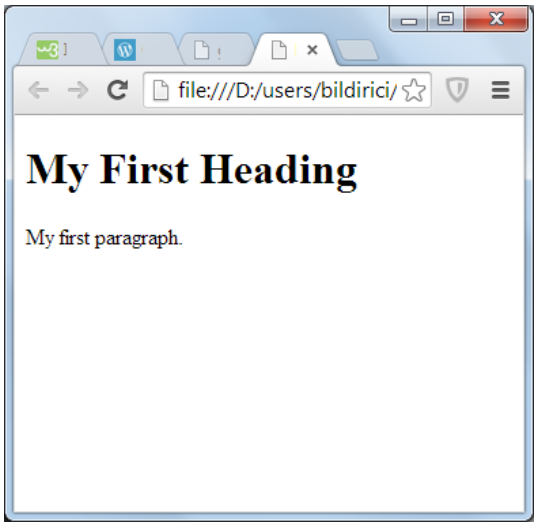
Web tarayıcılarının amacı (Internet Explorer, Google Chrome, Firefox vb) HTML dokümanlarını okumak ve web sayfası olarak görüntülemektir. Tarayıcılar HTM etiketlerini göstermez, etiketlere göre içeriğin nasıl görüntüleneceğini belirler (Örnek: Şekil 2.1).

HTML Sayfa Yapısı

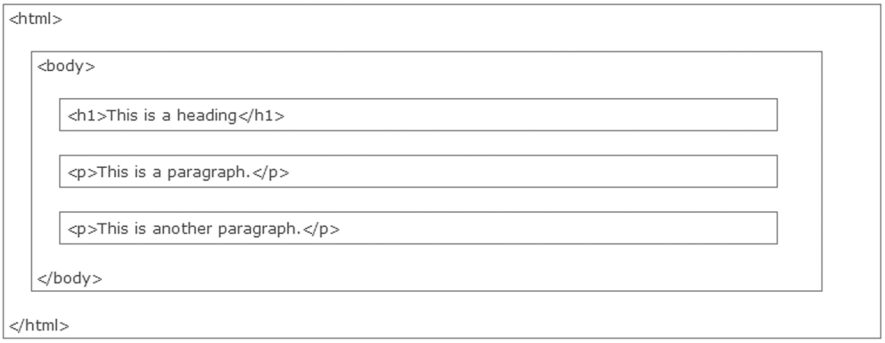
HTML sayfa yapısı görsel olarak Şekil 2.2'de görülmektedir.

DOCTYPE Bildirimi

`<!DOCTYPE>` bildirimi, web tarayıcısının (örneğin Google Chrome) web sayfasını doğru görüntülemesine yardım eder. Web ortamında çok çeşitli dokümanlar olduğundan web tarayıcı bir web sayfasını, doküman türünü ve HTML versiyonunu bilirse eksiksiz olarak görüntüleyebilir. Yaygın kullanılan bildirimler Çizelge 2.1'de verilmiştir.



Şekil 2.1: Web sayfası görünümü



Şekil 2.2: HTML sayfa yapısı

Çizelge 2.1: Doctype bildirimi

Versiyon	Bildirim
HTML5	`<!DOCTYPE html>`
HTML 4.01	`<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">`
XHTML 1.0	`<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">`

2.2 HTML Elemanları

Bir HTML elemanı başlangıç etiketinden bitiş etiketine kadar her şeydir.

Başlangıç etiketi	Eleman içeriği	Bitiş etiketi
`<p>`	This is a paragraph	`</p>`
`<a href="default.htm">`	This is a link	`</a>`
` `		

HTML Eleman Söz Dizimi

- Bir HTML elemanı başlangıç etiketi ile başlar.
- Bitiş etiketi ile sona erer.
- Eleman içeriği başlangıç ve bitiş etiketi arasında yer alan her şeydir.
- Boş elemanlar başlangıç etiketinde kapatılır.
- Çoğu HTML elemanının öznitelikleri (attribute) vardır.
- Çoğu HTML elemanı iç içe olabilir. HTML dokümanları iç içe HTML elemanlarından oluşur.

HTML doküman örneği

```
<!DOCTYPE html>
<html>
<body>
<p>This is my first paragraph.</p>
</body>
</html>
```

Yukarıdaki örnek 3 elemandan oluşmaktadır.

- `<p>` elemanı HTML dokümanı içinde bir paragraf tanımlar.
- `<body>` elemanı HTML dokümanının gövdesini tanımlar. İçerisinde `<p>` elemanı bulunmaktadır (iç içe elemanlar).
- `<html>` elemanı tüm HTML dokümanını tanımlar. İçerisinde `<body>` elemanı bulunmaktadır.

Bazı HTML elemanları kapanış etiketi unutulsa da doğru görüntülenir. Ancak bu durum web tarayıcılarına göre değişebilir ve kapanış etiketleri unutulmamalıdır.

İçeriği olmayan HTML elemanları, **boş elemanlar** olarak adlandırılır.`<br>` kapanış etiketi olmayan bir boş elemandır. Bir satır alta geçilmesini sağlar.

HTML etiketleri büyük küçük harf duyarlı değildir. `<p>` ile `<P>` eşdeğerdir. W3C (World Wide Web Consortium) küçük harf kullanılmasını tavsiye etmektedir.

2.3 HTML Öznitelikleri

- HTML elemanlarının öznitelikleri (attribute) olabilir.
- Öznitelikler bir eleman hakkında ek bilgiler sağlar.
- Öznitelikler daima başlangıç etiketi içinde belirtilir.
- `isim="değer"` formatında kullanılırlar.

Örnek

```
<a href="http://www.w3schools.com">This is a link</a>
```

Yukarıda `href` özniteliği ile bağlantı (link) adresi tanımlanmaktadır.

Öznitelikler daima çift tırnak ya da tek tırnak içine alınır. Genellikle çift tırnak kullanılır ancak tek tırnak da kabul edilir. Öznitelik tanımları da büyük küçük harf duyarlı değildir. Ancak W3C küçük harflerle yazılmasını önermektedir.

Her HTML elemanının değişik öznitelikleri vardır. Aşağıda her HTML elemanında kullanılabilecek bazı öznitelikler verilmiştir.

Öznitelik	Tanımı
`class`	Bir HTML elemanının bir ya da daha çok sınıf ismini belirler.
`id`	Bir eleman için tek bir id belirler.
`style`	Bir eleman için CSS stilini belirler.
`title`	Bir eleman için ekstra bilgi belirler.

2.4 HTML Başlıkları

HTML dokümanlarında başlıklar önemlidir. `<h1>` `<h6>` etiketleri ile tanımlanırlar. `<h1>` en önemli, `<h6>` en az önemlidir.

- HTML başlıkları başlık olarak kullanılmalı, yazıyı daha büyük ya da kalın (bold) yapmak için kullanılmamalıdır.
- Arama makineleri web sayfanızın içeriğini ve yapısını indekslemek için başlıkları kullanırlar.
- Kullanıcılar başlıklarına göre sayfa özetini çıkarabilirler. Bu nedenle başlıkların dokümanın yapısını yansıtması önemlidir.
- H1 ana başlık için kullanılmalı, onu H2, H2 yi de H3 vd. takip etmelidir.

HTML satırları: `<hr>` etiketi yatay bir çizgi oluşturur. Bu eleman ile sayfa içeriği bölümlere ayrılabilir.

2.5 HTML Paragrafları

Paragraflar `<p>` etiketi ile tanımlanır. Web tarayıcıları otomatik olarak paragraf öncesi ve sonrasında birer boş satır eklerler.

```
<p>This is a paragraph</p>
<p>This is another paragraph</p>
```

Uyarı: Paragraf elemanlarında kapanış etiketi olmasa da paragraf görüntülenir. Ancak bunun bir garantisi yoktur ve web tarayıcılara bağlı olarak beklenmedik sonuçlar ortaya çıkabilir.

Satır Sonu

`<br>` etiketi ile satır sonu tanımlanır. Bu şekilde bir alt satıra geçilir. Bu etiket boş eleman olduğundan kapanış etiketi yoktur.

2.6 Format Bildirim Etiketleri

`<b>` `<i>` gibi etiketler yazıların kalın veya italik çıkmasını sağlayan formatlama etiketleridir.

Yazı format etiketleri aşağıdaki tabloda görülmektedir.

Etiket	Tanımlama
`<b>`	Kalın (bold)
`<em>`	Vurgulanmış (italik)
`<i>`	İtalik
`<small>`	Küçük
`<strong>`	Önemli yazı (kalın yazı)
`<sub>`	İndis
`<sup>`	Üs
`<ins>`	Eklenmiş yazı (inserted text)
`<del>`	Silinmiş yazı
`<mark>`	İşaretlenmiş, vurgulanmış yazı

2.7 Yorum Satırları

HTML yorum satırları `<!--` ve `-->` arasında yazılır.

```
<!-- Yorum satırı -->
```

Yorum satırları web tarayıcısı tarafından gösterilmez. Programlama dillerinde yorum satırları programcılara yardımcı olur. Bu şekilde HTML kodu içinde notlar almak ve uyarılar koymak mümkün olur.

```
<!-- Bu bir yorum satırı -->
<p>Bu bir paragraftır.</p>
<!-- Buraya daha fazla bilgi ekleyebilirsiniz. -->
```

2.8 HTML Bağlantıları

- `<a>` etiketi bir hyperlink (bağlantı) tanımlar.
- Bir hyperlink bir ya da birden çok kelime ya da görüntü olup, bunlara tıklanarak başka dokümanlara ulaşılır.
- Bir web sayfasında bir link üzerine imleç geldiğinde bir el sembolü belirir.
- `<a>` elemanının en önemli özniteliği ulaşılacak dokümanın belirlendiği `href` özniteliğidir.
- Aksi belirtilmedikçe bağlantılar (link, hyperlink) tüm web tarayıcılarda aşağıdaki gibi görüntülenir.
  - Ziyaret edilmemiş bağlantı altı çizili ve mavi renktedir.
  - Ziyaret edilmiş bağlantı altı çizili ve mor renktedir.
  - Aktif bağlantı altı çizili ve kırmızı renktedir.

HTML Bağlantı Söz Dizimi

Bağlantılar için HTML kodu basittir.

```
<a href="url">Link text</a>
```

Örnek:

```
<a href="http://www.w3schools.com/">Visit W3Schools</a>
```

Hedef Özniteliği (Target Attribute)

Hedef (target) özniteliği bağlantılanan (link edilen) dokümanın nerede açılacağını belirler. Aşağıdaki örnekte bağlantılı doküman yeni bir tarayıcı penceresinde (ya da yeni bir sekmede) açılacaktır.

```
<a href="http://www.w3schools.com/" target="_blank">Visit W3Schools</a>
```

id Özniteliği

id özniteliği bir HTML dokümanı içinde yer imi (bookmark) oluşturmak için kullanılır. Yer imleri özel bir biçimde görüntülenmez.

Örnek: Bir HTML dokümanı içinde id ile bir paragraf tanımlansın.

```
<p id="tips">Yararlı Bilgiler Bölümü</p>
```

Buraya bağlantı verme:

```
<a href="#tips">Yararlı bilgiler bölümünü ziyaret edin.</a>
```

"Yararlı Bilgiler Bölümü" ne başka bir sayfadan bağlantı verme:

```
<a href="http://www.w3schools.com/html_links.htm#tips">
Yararlı bilgiler ...</a>!
```

2.9 Head Elemanı

`<head>` elemanı diğer tüm html elemanlarını kapsar. `<head>` içindeki elemanlar betikler (script), web tarayıcıya stil sayfalarını (style sheets) nerede bulacağını gösteren açıklamalar, meta veriler vb. olabilir.

Title Elemanı

`<title>` etiketi dokümanın başlığını belirler. `<title>` elemanı tüm HTML ve XHTML dokümanlarında gereklidir.

- Web tarayıcı araç çubuğunda başlığı tanımlar.
- Sayfa sık kullanılanlara eklendiğinde bu başlık eklenir.
- Arama motoru sonuçlarında bu başlık gösterilir.

Basitleştirilmiş bir HTML dokümanı:

```
<!DOCTYPE html>
<html>
<head>
<title>Title of the document</title>
</head>

<body>
The content of the document......
</body>

</html>
```

Base Elemanı

`<base>` etiketi bir sayfadaki tüm göreceli URL'ler için baz URL ya da hedefi belirtir.

```
<head>
<base href="http://www.w3schools.com/images/" target="_blank">
</head>
```

Link Elemanı

`<link>` etiketi bir doküman ile bir dış kaynak arasındaki ilişkiyi tanımlar.

```
<head>
<link rel="stylesheet" type="text/css" href="mystyle.css">
</head>
```

Style Elemanı

`<style>` etiketi bir HTML dokümanı için stil bilgisini tanımlamak için kullanılır. `<style>` elemanı içinde HTML elemanlarının web tarayıcı nasıl görüntüleneceği belirlenir.

```
<head>
<style type="text/css">
body {background-color:yellow;}
p {color:blue;}
</style>
</head>
```

Meta Elemanı

Meta veri veri hakkında veri demektir. `<meta>` etiketi HTML dokümanı hakkında meta veri sağlar. Meta veri sayfada gösterilmez, ancak bilgisayarda ayrıştırılabilir (parse edilebilir).

Meta elemanları tipik olarak sayfa tanımlamasını, anahtar kelimeleri, dokümanın yazarı vb. bilgileri belirtmek için kullanılırlar. Meta veri web tarayıcılar, arama motorları ve diğer web servisleri tarafından kullanılır. `<meta>` elemanı her zaman `<head>` elemanı içinde yer alır.

Örnekler:

Arama motorları için anahtar kelime tanımları

```
<meta name="keywords" content="HTML, CSS, XML, XHTML, JavaScript">
```

Web sayfası tanımı

```
<meta name="description" content="Free Web tutorials on HTML and CSS">
```

Sayfanın yazarının tanımı

```
<meta name="author" content="Bildirici">
```

Dokümanın her 30 saniyede bir yenilenmesi

```
<meta http-equiv="refresh" content="30">
```

`<script>` Elemanı

`<script>` etiketi, JavaScript gibi istemci taraflı (client-side) scriptlerin (betik) tanımlanması için kullanılır.

HTML Head Elemanları

Etiket	Tanımlama
`<head>`	Doküman hakkında bilgi tanımlama
`<title>`	Doküman başlığını tanımlama
`<base>`	Sayfadaki tüm linkler için varsayılan bağlantı ya da hedef
`<link>`	Dış kaynaklar ile doküman arasındaki ilişki tanımları
`<meta>`	Metaveri tanımları
`<script>`	İstemci taraflı script tanımları
`<style>`	Stil bilgileri

2.10 HTML Stilleri (CSS)

HTML elemanlarının stillerini belirlemek için CSS (Cascading Style Sheets) kullanılır. CSS HTML elemanlarına daha iyi stil tanımlamaları yapabilmek için HTML 4 ile geldi. CSS HTML koduna aşağıdaki yollarla eklenir.

- Inline: HTML elemanlarında style özniteliğini kullanarak
- Internal: `<head>` bölümünde `<style>` elemanını kullanarak
- External: Bir dış CSS dosyası kullanarak

2.10.1 Inline Stiller

Inline stiller bir elemanda bir defalık stil tanımlaması yapılacak ise kullanılır. Inline stiller ilgili etiketin style özniteliği kullanılarak belirlenir. Stil özniteliği herhangi bir CSS özelliğini içerebilir. Aşağıdaki örnek bir paragrafın sol marjı ve metin renginin nasıl değiştirildiğini göstermektedir.

```
<p style="color:blue;margin-left:20px;">This is a paragraph.</p>
```

CSS stilleri hakkında burada daha fazla ayrıntıya girilmeyecek olup aşağıdaki örneklerle yetinilecektir.

Font, Renk ve Yazı Büyüklüğü

font-family, color ve font-size özellikleri bir eleman içindeki metinin font, renk ve büyüklüğünü tanımlarlar.

```
<!DOCTYPE html>
<html>
<body>
<h1 style="font-family:verdana;">A heading</h1>
<p style="font-family:arial;color:red;font-size:20px;">A paragraph.</p>
</body>
</html>
```

Metin Hizalama

text-align özelliği bir eleman içindeki metnin yatay hizalamasını belirler.

```
<!DOCTYPE html>
<html>
<body>
<h1 style="text-align:center;">Center-aligned heading</h1>
<p>This is a paragraph.</p>
</body>
</html>
```

2.10.2 Dahili (Internal) Stil Sayfası

Bir doküman tek bir stile sahip ise dahili (internal) stil sayfası ile tanımlama yapılır. Internal stiller `<head>` bölümü altında `<style>` etiketi ile aşağıdaki örneğe benzer olarak tanımlanır.

```
<head>
<style>
body {background-color:yellow;}
p {color:blue;}
</style>
</head>
```

2.10.3 Harici (External) Stil Sayfası

Bir stil bir çok sayfaya uygulanmak isteniyorsa harici (external) stil sayfası kullanmak idealdir. Dış stil sayfası ile tüm web sitesinin görünümünü bir dosya ile değiştirmek mümkündür. Böyle bir

durumda her sayfa `<head>` bölümünde `<link>` etiketi ile stil sayfasıyla ilişkilendirilir.

```
<head>
<link rel="stylesheet" type="text/css" href="mystyle.css">
</head>
```

2.11 HTML Resimleri

HTML'de resimler (image) `<img>` etiketi ile tanımlanır. Boş bir eleman olup öznitelikleri vardır, kapanış etiketi yoktur. Bir sayfada bir resim görüntülemek için `src` özniteliği kullanılmalıdır. `src` kaynak anlamında olup, değeri görüntülenmek istenen resim dosyasının URL adresidir.

Resim tanımlamanın söz dizimi:

```
<img src="url" alt="some_text">
```

URL resmin saklandığı (resim dosyası) konumu gösterir. Bir görüntü dosyası adı x.gif ve www.x.com adresinde images klasöründe bulunuyor ise URL http://www.x.com/images/x.gif olur.

Web tarayıcılar resmi `<img>` etiketinin geçtiği yerde görüntüler.

Alt Özniteliği

Alt özniteliği resim görüntülenmesi mümkün olmadığında görüntülenecek metni belirlemek için kullanılır. Bu şekilde resme alternatif, resmi açıklayıcı bir bilgi verilir. Örnek:

```
<img src="smiley.gif" alt="Smiley face">
```

Resmin Genişliği ve Yüksekliği

Height ve width öznitelikleri resim boyutlarını belirlemek için kullanılır. Boyutların varsayılan birimi pikseldir.

```
<img src="smiley.gif" alt="Smiley face" width="42" height="42">
```

HTML Resim Etiketleri

Etiket	Description
`<img>`	Resim tanımlaması
`<map>`	image-map tanımı
`<area>`	image-map içinde tıklanabilir alan tanımı

2.12 Tablolar

- Tablolar `<table>` etiketi ile tanımlanır.
- Tablolar `<tr>` etiketi ile satırlara ayrılır.
- Tablo satırları `<td>` etiketi ile tablo verilerine (table data, sütunlar) ayrılır.

- Bir tablo satırı `<th>` etiketi ile tablo başlıklarına da ayrılabilir.
- Tablo verileri `<td>` bir tabloda verileri barındıran etiketlerdir. Metin, resim gibi tüm HTML elemanlarını barındırabilirler.

Örnek:

```
 <table>
  <tr>
    <td>Jill</td>
    <td>Smith</td>
    <td>50</td>
  </tr>
  <tr>
    <td>Eve</td>
    <td>Jackson</td>
    <td>94</td>
  </tr>
</table>
```

Bir tabloda border özniteliği tanımlanmazsa tablo satır ve sütunları ayıran çizgiler olmaksızın görüntülenir. Örnek:

```
<table border="1" style="width:100%">
```

Border özniteliği yerine CSS border özelliği de kullanılabilir.

```
<head>
<style>
table, th, td {
    border: 1px solid black;
}
</style>
</head>
```

Tablolarda çizgiler tablonun dışına ve her hücrenin etrafına çizilir. Aralarında boşluk vardır. Böyle bir boşluklar oluşmaması için (tek çizgili bir görünüm için) CSS border-collapse tanımlaması yapılabilir.

```
table, th, td {
    border: 1px solid black;
    border-collapse: collapse;
}
```

Hücre içeriği ile hücreyi sınırlayan çizgiler arasındaki boşluk CSS Padding özelliği ile tanımlanabilir.

```
th,td {
    padding: 15px;
}
```

Tablo başlıkları `<th>` etiketi ile belirlenir. Çoğu web tarayıcı varsayılan olarak tablo başlıklarını kalın ve ortalanmış görüntüler. Örnek:

```
 <table style="width:100%">
  <tr>
    <th>Firstname</th>
    <th>Lastname</th>
    <th>Points</th>
  </tr>
  <tr>
    <td>Eve</td>
    <td>Jackson</td>
    <td>94</td>
  </tr>
</table>
```

Sütunları birleştirmek için `colspan` özniteliği kullanılır.

```
<table>
<tr>
<th>Adı Soyadı</th>
<th colspan="2">Telefon</th>
</tr>
<tr>
<td>Ö. Bildirici</td>
<td>11122233 (Oda)</td>
<td>33311122 (Lab)</td>
</tr>
</table>
```

Satırları birleştirmek için `rowspan` özniteliği kullanılır.

```
<table>
<tr>
<th>Adı Soyadı:</th>
<td>Ö. Bildirici</td>
</tr>
<tr>
<th rowspan="2">Telefon:</th>
<td>111222333 (Oda)</td>
</tr>
<tr>
<td>333222111 (Lab) </td>
</tr>
</table>
```

Tablolara başlık eklemek için `caption` elemanı kullanılır. `<table>` etiketinden hemen sonra kullanılmalıdır.

```
<table>
<caption>Başlık</caption>
...
```

Tablo `<thead>,<tbody>,<tfoot>` elemanları ile bölümlere ayrılabilir. Web tarayıcılar başlık (thead) ve altlıktan (tfoot) bağımsız tablo gövdesinde kaydırma yapılmasını sağlayabilirler. Birden çok sayfaya yayılan tablolar yazıcıya gönderildiğinde her sayfada başlık ve altlık çıkması sağlanabilir. Aşağıdaki örneği inceleyin.

```
<table>
  <thead>
   <tr>
   <th>Ay</th>
   <th>Harcama</th>
   </tr>
  </thead>
  <tbody>
    <tr>
    <td>Ocak</td>
    <td>500</td>
    </tr>
    <tr>
    <td>Şubat</td>
    <td>750</td>
    </tr>
  </tbody>
  <tfoot>
    <tr>
    <td>Toplam</td>
    <td>1250</td>
    </tr>
  </tfoot>
</table>
```

2.13 Listeler

Sıralı olmayan HTML listeleri `<ul>` etiketi ile başlar. Listedeki her eleman `<li>` etiketi ile başlar. Liste elemanları küçük siyah noktalarla (bullet) işaretlenir. Örnek:

```
 <ul>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ul>
```

Style Özniteliği

Sıralı olmayan liste işaret türü style özniteliği ile değiştirilebilir.

```
<ul style="list-style-type:square">
```

Stil	Tanım
list-style-type:disc	Küçük siyah noktalarla işaretleme (varsayılan)
list-style-type:circle	Daire ile
list-style-type:square	Kare ile

Sıralı HTML listeleri `<ol>` etiketi ile, listedeki her eleman ise `<li>` etiketi ile başlar. Liste elemanları numaralanır. Örnek:

```
 <ol>
```

```
  <li>Coffee</li>
  <li>Milk</li>
</ol>
```

Sıralama (numaralama) şeklini type özniteliği ile belirlemek mümkündür. Örnek:

```
<ol type="A">
```

Tip	Tanımlama
type=”1”	Rakamlarla numaralama (varsayılan)
type=”A”	Büyük harflerle numaralama
type=”a”	Küçük harflerle numaralama
type=”I”	Büyük roma rakamları ile numaralama
type=”i”	Küçük roma rakamları ile numaralama

Tanımlama listeleri bir grup kavram ve tanımlarını içeren listelerdir. `<dl>` etiketi tanımlama listesini tanımlar. Kavramlar `<dt>`, tanımları `<dd>` etiketleri ile belirtilir.

```
 <dl>
  <dt>Coffee</dt>
  <dd>- black hot drink</dd>
  <dt>Milk</dt>
  <dd>- white cold drink</dd>
</dl>
```

Listeler iç içe olabilir.

```
 <ul>
  <li>Coffee</li>
  <li>Tea
    <ul>
      <li>Black tea</li>
      <li>Green tea</li>
    </ul>
  </li>
  <li>Milk</li>
</ul>
```

Önemli Hususlar

- `<ul>` sıralı olmayan listeleri tanımlamak için kullanılır.
- Bulet stili style özniteliği ile tanımlanır.
- `<ol>` sıralı listeleri tanımlamak için kullanılır.
- Numaralama biçimi type özniteliği ile belirlenir.
- Liste elemanları `<li>` etiketi ile tanımlanır.
- `<dl>` etiketi tanımlama listeleri için kullanılır.
- `<dt>` kavramları `<dd>` açıklamaları tanımlamak için kullanılır.
- Listeler iç içe kullanılabilir.
- Listeler diğer HTML elemanlarını içerebilir.

2.14 HTML Blokları (div ve span)

HTML elemanları `<div>` ve `<span>` etiketleri ile gruplanabilirler. Çoğu HTML elemanı,

- blok seviyesi ve
- inline

elemanlar olmak üzere iki şekilde tanımlanır. Inline elemanlar yeni bir satıra geçilmeksizin görüntülenir. Örneğin: `<b>, <td>, <a>, <img>`

`<div>` Elemanı

Blok seviyesinde bir eleman olup, diğer HTML elemanlarını gruplamak için kullanılan bir taşıyıcı elemandır. `<div>` elemanının özel bir anlamı yoktur. Ancak blok seviyesi bir eleman olduğundan web tarayıcılarda öncesi ve sonrasında birer boş satır görüntülenir. `<div>` elemanı CSS ile birlikte büyük blokların stil tanımlamalarını topluca yapmak için kullanılabilir.

Sayfanın düzeni (layout) için de yaygın olarak `<div>` kullanılır. Sayfa düzeni tablolar ile de yapılabilir. Ancak bu eski bir tarz olup bunun yerine `<div>` kullanılmalıdır. Tabloların amacı sayfa düzeni belirlemek değildir.

`<span>` Elemanı

Inline bir eleman olup metin için bir taşıyıcı olarak kullanılabilir. Özel bir anlamı olmayıp, CSS ile birlikte bir metnin bir kısmının stil özelliklerinin belirlenmesinde kullanılabilir.

2.15 HTML Sayfa Düzeni

Web sayfasının görünümü açısından web sayfa düzeni tasarımı önemlidir. Çoğu web sayfasında içerik kolonlar halinde (dergi ve gazetelere benzer olarak) sunulur. Çoklu kolon uygulaması `<div>` ya da `<table>` elemanları ile yapılır. Tablolar ile sayfa düzeni tasarlaması yapılabilir. Ancak tabloların asıl amacı veri sunumudur, sayfa düzeni tasarımı değildir.

Div elemanı HTML elemanlarını gruplamak için kullanılan blok seviyesinde bir elemandır. Aşağıdaki örnek çok kolonlu bir sayfa düzeni oluşturmak için beş `div` elemanı kullanmaktadır.

```
<!DOCTYPE html>
<html>
<body>
<div id="container" style="width:500px">
<div id="header" style="background-color:#FFA500;">
<h1 style="margin-bottom:0;">Main Title of Web Page</h1></div>
<div id="menu" style="background-color:#FFD700;height:200px;width:100px;
float:left;">
<b>Menu</b><br>
HTML<br>
CSS<br>
JavaScript</div>
<div id="content" style="background-color:#EEEEEE;height:200px;width:400px;
float:left;">
Content goes here</div>
<div id="footer" style="background-color:#FFA500;clear:both;
```

```
text-align:center;">
Copyright © W3Schools.com</div>
</div>
</body>
</html>
```

Aşağıdaki örnekte ise 3 satır ve 2 sütun kullanılarak tablo ile sayfa düzeni yapılmıştır.

```
<!DOCTYPE html>
<html>
<body>
<table style="width:500px;" cellpadding="0" cellspacing="0">
<tr>
<td colspan="2" style="background-color:#FFA500;">
<h1 style="margin:0;padding:0;">Main Title of Web Page</h1>
</td>
</tr>
<tr>
<td style="background-color:#FFD700;width:100px;vertical-align:top;">
<b>Menu</b><br>
HTML<br>
CSS<br>
JavaScript
</td>
<td style="background-color:#eeeeee;height:200px;width:400px;
vertical-align:top;">
Content goes here</td>
</tr>
<tr>
<td colspan="2" style="background-color:#FFA500;text-align:center;">
Copyright © W3Schools.com</td>
</tr>
</table>
</body>
</html>
```

Şekil 2.3'de her iki kodun oluşturduğu web sayfası görüntüsü yer almaktadır.

Şekil 2.3: Web sayfa düzeni

2.16 HTML Formları ve Giriş

Kullanıcıların çeşitli yollarla veri girişi yapmaları formlar kullanılarak sağlanır. Formlar sunuculara veri aktarmak için kullanılırlar.

Bir HTML formu metin kutuları (text field), onay kutuları (checkbox), radyo butonları, gönderme butonları (submit button) gibi giriş elemanları içerebilir. Formlar ayrıca seçim listeleri, metin alanları, etiketler vb de içerebilir.

Bir HTML formu oluşturmak için `<form>` etiketi kullanılır.

```
<form>
.
input elements
.
</form>
```

Input Elemanı

- En önemli form elemanı `<input>` elemanıdır.
- Kullanıcının bilgi girişi için kullanılır.
- `<input>` elemanı type özniteliğine bağlı olarak değişik şekiller alır.
- `<input>` elemanı, metin kutusu, onay kutusu, şifre kutusu, gönderme butonu vb olabilir.
- En yaygın kullanılan türler aşağıda açıklanacaktır.

Metin Kutusu

`<input type="text">` kullanıcının metin girebileceği bir metin kutusu tanımlar.

```
<form>
First name: <input type="text" name="firstname"><br>
Last name: <input type="text" name="lastname">
</form>
```

Formun kendisi görüntülenmez, varsayılan metin kutusu karakter genişliği 20'dir.

Şifre Kutusu

```
<form>
Password: <input type="password" name="pwd">
</form>
```

Şifre kutusunda (password field) bulunan karakterler maskelenir (yıldız ya da daireler biçiminde).

Radyo Butonları

Radyo butonları birden çok seçenek arasından birini seçme olanağı verir.

```
<form>
<input type="radio" name="cinsiyet" value="e">Erkek<br>
<input type="radio" name="cinsiyet" value="k">Kadın
</form>
```

Onay Kutusu

Onay kutuları (check box) belli sayıda seçenek arasından birden fazlasını (ya da hiçbirini) seçme olanağı verir.

```
<form>
<input type="checkbox" name="vehicle" value="Bike">I have a bike<br>
<input type="checkbox" name="vehicle" value="Car">I have a car
</form>
```

Gönderme (Submit) Butonu

`<input type="submit">` bir gönderme (submit) butonu tanımlar. Form verilerini bir sunucuya göndermek için kullanılırlar. Veri, formun action özniteliğinde belirtilen sayfaya gönderilir. Bu öznitelikte belirtilen dosya, alınan veriler ile bir işlem yapar.

```
<form name="input" action="demo_form_action.asp" method="get">
Username: <input type="text" name="user">
<input type="submit" value="Submit">
</form>
```

Yukarıdaki örnekte metin kutusuna giriş yapılıp, submit butonuna basılırsa web tarayıcı metin kutusu içeriğini `demo_form_action.asp` adlı bir sayfaya gönderir. Form etiketleri Çizelge 2.2'de özetlenmiştir.

2.17 Iframe

Iframe elemanı bir web sayfasını başka bir sayfa içinde göstermek için kullanılır. Söz dizimi: `<iframe src="URL"></iframe>`

`height` ve `width` (yükseklik ve genişlik) öznitelikleri bir Iframe elemanının boyutlarını belirlemek için kullanılır. Boyutlar piksel birimindedir. Yüzdelik oran biçiminde de belirtilebilir.

```
<iframe src="demo_iframe.htm" width="200" height="200"></iframe>
```

Çerçeve çizgisi `frameborder` özniteliği 0 değeri alırsa görüntülenmez.

Bir iframe, bir bağlantının hedefi olarak kullanılabilir. Bu durumda bağlantının (link) `target` özniteliği bir iframe'in `name` özniteliği ile eşleşmelidir.

```
<iframe src="demo_iframe.htm" name="iframe_a"></iframe>
<p><a href="http://www.w3schools.com" target="iframe_a">W3Schools.com</a></p>
```

Çizelge 2.2: HTML Form Etiketleri

Etiket	Tanımlama
`<form>`	HTML form tanımlaması
`<input>`	Input elemanı tanımlaması
`<textarea>`	Çok satırlı input elemanı(text area)
`<label>`	Bir input elemanı için etiket tanımlaması
`<fieldset>`	Bir form için birbirleri ile ilişkili elemanların oluşturduğu grup
`<legend>`	Bir `<fieldset>` elemanı için başlık (capton) tanımlaması
`<select>`	Açılır liste
`<optgroup>`	Açılır liste içinde bir seçenek grubu tanımlama
`<option>`	Açılır liste içinde bir seçenek
`<button>`	Tıklanabilir buton
`<datalist>`	Input elemanları için önceden tanımlanmış seçeneklerin listesini belirtme (HTML 5)
`<keygen>`	Formlar için anahtar çifti üreten bir alan tanımlama (HTML 5)
`<output>`	Bir hesaplamanın sonucunu tanımlama (HTML 5)

2.18 HTML Renkleri

HTML kodunda toplayıcı renkler kullanılır. Toplayıcı renk sisteminde herhangi bir renk içindeki kırmızı, yeşil mavi bileşenlerinin oranları (miktarları) ile ifade edilir. Bilgisayar sistemlerinde kırmızı, yeşil ve mavi bileşenleri için birer byte yer ayrılır. Bu şekilde her bir bileşen (ana renk) 0-255 arası değerler alır, $256^3 = 16777216$ değişik renk tanımlanması mümkündür.

Renk Değerleri

CSS renkleri, kırmızı, yeşil ve mavi değerlerinin kombinasyonu için hekzadesimal (onaltılı sayı sisteminde) yazım biçimi kullanılarak tanımlanır. En küçük değer 0 (00), en büyük değer 255 (FF)dir. Hekzadesimal değerler üç çift rakamla, `#` ile başlayarak yazılırlar. Örneğin `#000000` (rgb(0,0,0)), `#FF0000` (rgb(255,0,0)).

Günümüzdeki bilgisayar sistemleri 16 milyon rengi sorunsuz görüntüler.

Hekzadesimal yazım biçimi ile 16 milyon renk kullanılmasına rağmen HTML dilinde belli sayıda renge isim verilmiştir. Bu şekilde 140 renk tanımlanmıştır. Örneğin Aqua (`##00FFFF`), Blue (`##0000FF`). Tanımlı renklerin tam listesi için `http://www.w3schools.com/html/html_colornames.asp` sayfasından yararlanılabilir.

2.19 HTML Betikleri (Skriptleri)

JavaScript ile yazılan betikler HTML sayfalarını daha dinamik ve etkileşimli hale getirir.

JavaScript gibi istemci taraflı (client-side) betikleri tanımlamak için `<script>` etiketi kullanılır. `<script>` elemanı ya betik ifadelerini içerir ya da src özniteliğinde belirtilen bir dış betik dosyasını gösterir.

Aşağıdaki örnek kod, id özniteliği "demo" olarak tanımlanmış bir HTML elemanının içine "Hello JavaScript!" yazdırmaktadır.

```
<script>
document.getElementById("demo").innerHTML = "Hello JavaScript!";
</script>
```

`<noscript>` etiketi ile betiklerin engellendiği ya da istemci taraflı betiklerin desteklenmediği durumlarda kullanıcılara bilgi verilebilir.

```
<noscript>Sorry, your browser does not support JavaScript!</noscript>
```

JavaScript bir sonraki bölümde ele alınacağından burada daha fazla ayrıntıya girilmeyecektir.

2.20 HTML Karakter Varlıkları

HTML'de rezerve edilmiş karakterler, karakter varlıkları (character entities) ile değiştirilerek kullanılabilirler. Klavyede bulunmayan karakterler de bu şekilde görüntülenebilir. Örneğin < ve > karakterlerini metin içerisinde büyüktür küçüktür karakterleri olarak kullanırsanız görüntülenmez. Web tarayıcıları bu karakterler arasında HTML etiketleri arar. İki şekilde yazılabilirler: `&karakter_adi` ya da `&#karakter_numarası`. Örneğin küçüktür işareti `&lt` ya da `&#60` yazılarak görüntülenir.

Karakter adları kullanmak kolay hatırlanma açısından kullanışlıdır. Ancak bazı web tarayıcılarda karakter adları tanımlanmamış olabilir. Bu nedenle sayı kullanmak uyum sorunları açısından daha güvenlidir.

Yaygın kullanılan karakter varlıklarından biri boşluktur (` `). Web tarayıcıları HTML sayfalarındaki fazladan boşlukları kaldırırlar. Bu nedenle birden çok boşluk kullanılmak istenirse ` ` kullanılmalıdır.

Çok kullanılan karakter varlıkları

Görüntü	Açıklama	Varlık Adı	Varlık Numarası
	non-breaking space	` `	` `
<	küçüktür	`<`	`&#60;`
>	büyüktür	`>`	`&#62;`
&	ampersand	`&`	`&#38;`
¢	cent	`&cent;`	`&#162;`
£	pound	`&pound;`	`&#163;`
¥	yen	`&yen;`	`&#165;`
€	euro	`&euro;`	`&#8364;`
©	copyright	`&copy;`	`&#169;`
®	registered trade mark	`&reg;`	`&#174;`

Benzer şekilde aritmetik operatörler, oklar, yunan alfabesi karakterleri vb. klavyeden girişi yapılamayacak karakterler de tanımlanabilir. Aşağıdaki web kaynaklarında ayrıntılı listeler bulunmaktadır.

- Para sembolleri `http://www.w3schools.com/charsets/ref_utf_currency.asp`
- Ok sembolleri `http://www.w3schools.com/charsets/ref_utf_arrows.asp`
- Tüm semboller `http://www.w3schools.com/charsets/ref_utf_arrows.asp`

2.21 Karakter Setleri

- ASCII ilk karakter kodlama (karakter seti) standardıdır. Internet ortamında kullanılabilecek 127 farklı alfanümerik karakter tanımlar.
- ASCII rakamları (0-9), İngiliz alfabesini (A-Z) ve bazı özel işaretleri ($, & vb) destekler.
- ANSI (Windows-1252) Windows 95'e kadar Windows varsayılan karakter seti idi. 256 farklı kod desteklemekteydi.
- ASCII ya ek olarak tanımlanan ISO-8859-1, HTML 4 için varsayılan karakter setiydi. Bu da 256 farklı kod destekliyordu.
- ANSI ve ISO karakter setleri kısıtlı olduğu için HTML 5'de varsayılan karakter kodlama standardı `Unicode (UTF8)` olarak değiştirildi.
- `Unicode` dünyada olası hemen hemen tüm karakterleri içerir.

HTML Charset Özniteliği

Bir HTML sayfasını doğru görüntüleyebilmek için web tarayıcı kullanılan karakter setini bilmelidir. Bu `<meta>` etiketi içinde belirtilir.

```
HTML4:
<meta http-equiv="Content-Type" content="text/html;charset=ISO-8859-1">
HTML5:
<meta charset="UTF-8">
```

Karakter setleri arasındaki farklılıklar için
`http://www.w3schools.com/html/html_charset.asp`
adresindeki tablodan yararlanılabilir.

2.22 URL

URL kısaltmasının açık yazımı "Uniform Resource Locator" olup, web adresleri anlamında kullanılır. Bir URL "www.selcuk.edu.tr" gibi karakterlerden (isim) ya da bir IP (Internet Protocol) adresinden (193.140.140.103 gibi) oluşur. IP adreslerinin hatırlanması daha zor olduğundan web sayfası isimleri kullanmak tercih edilir.

Web tarayıcıları sunuculardan sayfaları URL bilgisi ile isterler. Bir HTML sayfasındaki bir bağlantıya (linke) tıklanınca arka plandaki `<a>` etiketi bir Web adresini gösterir. Bu bağlamda URL, web üzerinde bir dokümanı adreslemek için kullanılır.
`http://www.w3schools.com/html/default.htm` gibi bir web adresi aşağıdaki söz dizimine uyar.

```
scheme://host.domain:port/path/filename
```

- scheme: İnternet servisinin türünü tanımlar. En yaygın olanı http'dir.
- host: Etki alanı ana bilgisayar (domain host) tanımlaması (http için varsayılan "www" dir.)
- domain: İnternet etki alanı tanımlaması, örneğin: w3schools.com
- port: Ana bilgisayardaki port numarası (http için varsayılan port 80)

- path: Sunucudaki dosya yolu (belirtilmez ise doküman web sitesi kök dizininde olmalıdır.)
- filename: Doküman ya da kaynak adı

Yaygın kullanılan URL şemaları

Şema	Açık Yazımı	Kullanım Amacı
http	HyperText Transfer Protocol	`http://` ile başlayan sıradan web sayfaları (şifrelenmemiş)
https	Secure HyperText Transfer Protocol	Güvenli web sayfaları, alınıp verilen tüm bilgiler şifrelenir.
ftp	File Transfer Protocol	Dosya indirme ve dosya yükleme, web sayfalarının bakımı için yararlıdır.
file		Kendi bilgisayarınızda bir dosya

URL Kodlama

- URL bilgisi İnternet üzerinden yalnızca ASCII karakter seti ile gönderilebilir.
- URL bilgisi ASCII karakter setinde bulunmayan karakterler içerebileceğinden URL metni ASCII formatına dönüştürülür.
- URL kodlama karakterleri İnternetten gönderilebilecek bir biçime dönüştürür.
- ASCII karakter setinde bulunmayan karakterler % ile başlayan ve iki hekzadesimal sayı ile devam eden üç karakterle değiştirilir. (Örnek: %A9)
- URL metninde boşluk bulunamaz. Bu nedenle boşluklar + ile değiştirilir.

Bazı Örnekler:

Karakter	URL Kodlama
€	%80
£	%A3
©	%A9
®	%AE

Bölüm 3

CSS

CSS (Cascading Style Sheets) bir HTML dokümanını biçimlendirmek için kullanılan dildir. HTML elemanlarının nasıl görüntüleneceğini tanımlar. CSS kodlamada önemli ölçüde kısaltma sağlar. Bir defada bir çok web sayfasının görünümünü belirlemek mümkündür. Dış stil tanımları CSS uzantılı dosyalarda saklanır.

Bazı örnekler:

```
body {
  background-color: lightblue;
}

h1 {
  color: white;
  text-align: center;
}

p {
  font-family: verdana;
  font-size: 20px;
}
```

HTML'in hiç bir zaman web sayfasını biçimlendirmek için etiketler içermesi amaçlanmadı. HTML 3.2 ile birlikte gelen `<font>` gibi etiketler web geliştiricileri için korkulu rüya olmaya başladı. Hacimli sayfalar için font ve renk bilgilerinin her sayfaya eklenmesi uzun ve masraflı bir proses oldu. Bu problemi çözmek için W3C (World Wide Web Consortium) CSS'yi geliştirdi.

CSS tanımlamaları genel olarak css uzantılı dosyalardadır. Bu dosyalar değiştirilerek web sayfası görünümleri değiştirilebilir.

Bu bölümde CSS konularından önemli olanları ele alınacaktır. İngilizce okuyabilen okurlar için `https://www.w3schools.com/css/default.asp` sayfasında daha ayrıntılı bilgiler bulunmaktadır.

Şekil 3.1: CSS Söz Dizimi

3.1 CSS Söz Dizimi

Bir CSS kural seti (rule-set) bir seçici ve bir bildirim bloğundan oluşur. Seçici, biçimlendirilmek istenen HTML elemanını işaret eder. Bildirim bloğu, birbirinden noktalı virgül ile ayrılmış bir ya da daha çok bildirim içerir. Her bir bildirim `özellik adı: değer` biçiminde kodlanır (Şekil 3.1).

Örnek:

```
p {
  color: red;
  text-align: center;
}
```

`p` seçici olup, biçimlendirilecek elemanı (paragraf) göstermektedir. `color` özellik, `red` ise bu özelliğin değeridir.

3.2 CSS Seçicileri

CSS seçicileri biçimlendirilmek istenen HTML elemanını tanımlar. 5 tür seçici vardır.

- Basit seçiciler (name, id ve class öznitelikleri ile elemanların seçilmesi)
- Birleşik seçiciler (Aralarındaki bir ilişkiye dayalı olarak elemanların seçilmesi)
- Pseudo-sınıf seçiciler (Belli biri duruma dayalı elemanların seçilmesi)
- Pseudo-eleman seçiciler (Bir elemanın bir kısmının seçilmesi ve biçimlendirilmesi)
- Öznitelik seçiciler (Elemanların öznitelik ya da öznitelik değerlerine göre seçilmesi)

Eleman seciçi ile html elemanlarının isimleri kullanılır.

```
p {
  color: blue;
  text-align: center;
}
```

ID seçici belli bir HTML elemanının `id` özniteliğini kullanılır. ID bir html sayfasında tekrar etmez. Bu nedenle ID seçici bir sayfadaki bir elemanı biçimlendirir! id değeri önüne `#` konulur. Aşağıdaki örnekle id değeri prg1 olan bir eleman söz konusudur. id değerleri rakamla başlayamaz..

```
#prg1 {
  color: blue;
  text-align: center;
}
```

Sınıf seçici HTML elemanlarını `class` özniteliği değerine göre seçer. Değerin önüne nokta eklenir. Aşağıdaki örnekte `class="ortala"` olarak tanımlanmış elemanlar biçimlendirilmektedir.

```
.ortala {
  color: blue;
  text-align: center;
}
```

Aşağıdaki örnekte ise class tanımlaması yukarıdaki gibi olan paragraf elemanları biçimlendirilmektedir.

```
p.ortala {
  color: blue;
  text-align: center;
}
```

HTML elemanları birden fazla sınıfa sahip olabilirler. Örneğin: `<p class="center large"> ... </p>`

Dikkat: Sınıf değerleri rakamla başlayamaz.

* evrensel seçicidir. Tüm HTML elemanlarını seçer.

```
* {
  color: brown;
  text-align: center;
}
```

Grup seçici ile virgülle ayrılarak seçim yapılır.

```
p,h1,h2 {
  color: blue;
  text-align: center;
}
```

3.3 CSS Nasıl Eklenir?

CSS tanımlamaları üç şekilde yapılır.

- Dış
- İç
- Satır içi

Bu konu 2.10 Başlığı altında ele alınmıştır.

3.4 CSS Yorum Satırları

CSS yorum satırları `style` elemanı içinde `/* */` arasında yer alır. Birden çok satıra yayılabilir.

```
/* This is a single-line comment */
p {
  color: red;
}
h1 {
  color: red;  /* Set text color to red */
}
```

```
/* This is
a multi-line
comment */

h2 {
  color: red;
}
```

3.5 CSS Renkleri

CSS'de önceden tanımlanmış renk isimleri, RGB, HEX, HSL, RGBA, HSLA değerleri kullanılır. HTML için önceden tanımlanmış 140 renk ismi CSS'de de geçerlidir[1].

`background-color` özelliği ile HTML elemanlarının arka plan renkleri belirlenebilir.

```
<h1 style="background-color:SlateBlue;">Hello World</h1>
<p style="background-color:Pink;">Lorem ipsum...</p>
```

`color` özelliği ile yazı rengi ayarlanabilir.

```
<h1 style="color:Tomato;">Hello World</h1>
<p style="color:DodgerBlue;">Lorem ipsum...</p>
<p style="color:MediumSeaGreen;">Ut wisi enim...</p>
```

`border` özelliği ile HTML çerçeve rengi ayarlanabilir.

```
<h1 style="border:2px solid Tomato;">Hello World</h1>
<h1 style="border:2px solid DodgerBlue;">Hello World</h1>
<h1 style="border:2px solid Violet;">Hello World</h1>
```

Renk değerleri ile ilgili örnekler:

```
<h1 style="background-color:rgb(255, 99, 71);">...</h1>
<h1 style="background-color:#ff6347;">...</h1>
<h1 style="background-color:hsl(9, 100%, 64%);">...</h1>

<h1 style="background-color:rgba(255, 99, 71, 0.5);">...</h1>
<h1 style="background-color:hsla(9, 100%, 64%, 0.5);">...</h1>
```

RGB değerleri 0-255 arasında değişir ve toplayıcı renklerin (kırmızı, yeşil, mavi) yoğunluğunu ifade eder. Örneğin `rgb(255,0,0)` kırmızıya karşılık gelir. (0,0,0) siyaha, (255,255,255) beyaza karşılık gelir.

RGBA değerleri RGB renk uzayının genişletilmiş biçimdir. Burada RGB değerlerine alfa kanalı olarak ifade edilen bir parametre daha eklenmiştir. 0-1 arasında değişir, reel sayıdır. 0 tam şeffaflık, 1 tam matlık anlamındadır. Örneğin `rgba(255,0,0,0.5)` %50 şeffaf kırmızıya karşılık gelir.

[1] `https://www.w3schools.com/colors/colors_names.asp`

3.6 Arka Plan

CSS arka plan özellikleri HTML elemanlarının arka plan görünümlerini belirlemek için kullanılır. Bu kapsamda aşağıdaki özellikler vardır.

- background-color
- background-image
- background-repeat
- background-attachment
- background-position

background-color özelliği renk konusunda incelenmişti. Her HTML elemanının arka plan rengi tanımlanabilir.

```
h1 {
  background-color: green;
}

div {
  background-color: lightblue;
}

p {
  background-color: yellow;
}
```

`opacity` özelliği ile bir HTML elemanının matlığı/şeffaflığı ayarlanabilir. 0.0 -1.0 arasında ayarlanır. Bir elemanın opacity değeri onun kapsadığı elemanları da etkiler. Bu istenmiyorsa RGBA renk tanımı ile matlık ayarı yapılabilir.

`background-image` özelliği ile arka planın bir resim dosyası olması sağlanır. Varsayılan ayarlamada resim gerektiği kadar tekrar edilir. Seçilen arka plan resimlerinin yazı ile kontrast oluşturmasına dikkat edilmelidir. Genel olarak `<body>` elemanının arka planında resim kullanılır. Ancak `<p>` elemanının bile arka planı bir resim dosyası olabilir.

`background-repeat` özelliği ile yalnızca yatay (`repeat-x`) ya da yalnızca düşey (`repeat-y`) tekrarlanma sağlanabilir. `no-repeat` ile bir kez görüntülenme sağlanabilir.

```
body {
  background-image: url("gradient_bg.png");
  background-repeat: repeat-x;
}
```

`background-position` ile resmin konumu belirlenebilir.

```
body {
  background-image: url("img_tree.png");
  background-repeat: no-repeat;
  background-position: right top;
}
```

`background-attachment` özelliği arka plan resminin kaydırılabilir ya da sabit olmasını ayarlar.

```
body {
  background-image: url("img_tree.png");
  background-repeat: no-repeat;
  background-position: right top;
  background-attachment: fixed;
}
```

3.7 Sınırlar

`border` özellikleri bir elemanın sınırlarını biçimlendirmeye yarar.

`border-style` özelliği aşağıdaki değerleri alır.

- `dotted` noktalı çizgiler
- `dashed` kesikli çizgiler
- `solid` sürekli çizgiler
- `double` çift çizgi
- `groove` 3B
- `ridge` 3B
- `inset` 3B
- `outset` 3B
- `none` çizgi yok
- `hidden` çizgi yok

`border-width` özelliği çizgi kalınlığını belirler. thin, medium, thick değerlerini alabilir. px,pt,cm,em birimlerinde sayısal değer verilebilir.

`border-color` çizgi rengini belirler. Renk değerleri önceki bölümlerde ele alınmıştır.

Her bir sınır çizgisini farklı biçimlendirmek mümkündür (top, bottom, left, right).

```
p {
  border-top-style: dotted;
  border-right-style: solid;
  border-bottom-style: dotted;
  border-left-style: solid;
}
```

`border-radius` özelliği ile köşe yuvarlatma yapılabilir. px,pt,cm,em birimlerinde sayısal değer verilebilir.

3.8 Marjlar

`margin` özelliği sınırlar dışında elemanların etrafında boşluklar tanımlar. Dört yönde farklı marjlar tanımlamak içinde özellikler vardır (`margin-top` vb.).

Marjin özelliklerinin alabileceği değerler:

- `auto`: Tarayıcı marjı belirler.
- Genişlik px,pt,cm birimlerinde sayısal değer olarak verilir.
- % : Marj elemanın genişliğinin yüzdesi olarak belirlenir.
- `parent`: Marjin kapsanılan elemandan alınır.

Not: Negatif değerler mümkündür.

```
p {
  margin-top: 100px;
  margin-bottom: 100px;
  margin-right: 150px;
  margin-left: 80px;
}
```

3.9 İç Boşluk: Padding

`padding` özelliği bir elemanın sınırları içinde boşluk oluşturmak için kullanılır. Dört yönde farklı değerler alabilir. Kullanımı ve alabileceği değerler `margin` ile benzerdir. En önemli farkı negatif değerlerin geçerli olmamasıdır.

```
div {
  padding-top: 50px;
  padding-right: 30px;
  padding-bottom: 50px;
  padding-left: 80px;
}
```

3.10 Yükseklik ve Genişlik

`height` ve `width` özellikleri bir elemanın genişlik ve yüksekliğini belirlemek için kullanılır. Bu özelliklerde sınır, marj ve padding dikkate alınmaz. Elemanın kendi genişlik ve yüksekliği söz konusudur.

Aşağıdaki değerleri alabilirler.

- `auto`: Tarayıcı değerleri belirler.
- px,pt,cm birimlerinde sayısal değer verilir.
- % : Kapsayan blok yükseklik ve genişliğinin yüzdesi

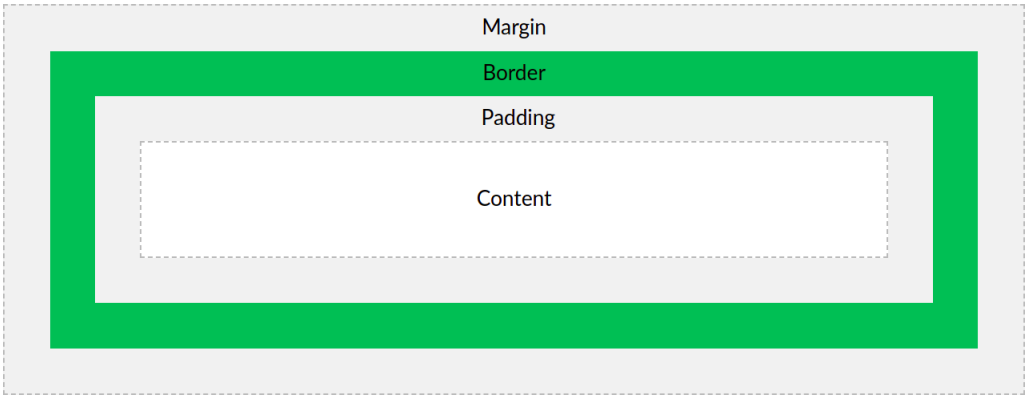
- `initial`: Varsayılan değerler
- `inherit`: Değerler kapsayan elemandan alınır.

```
div {
  height: 300px;
  width: 50%;
  background-color: orange;
}
```

`max-width` özelliği bir elemanın maksimum genişliğini belirler. Maksimum genişlik genişliği etkisiz hale getirir. Benzer şekilde `max-height, min-height, min-width` özellikleri de vardır. px, pt, cm biriminde değerler alırlar.

3.11 Box Model

Bir HTML elemanı kutular halinde düşünülebilir. CSS'de box model (kutu model) terimi tasarım ve çıktı (layout) söz konusu olduğunda ortaya çıkar. Kutu modeli her bir HTML elemanının etrafındaki iç içe kutuları ifade eder: Marj, sınır, padding ve içerik (Şekil 3.2).



Şekil 3.2: CSS kutu modeli

Dikkat: `height` ve `width` özellikleri ile içerik (content) bölgesinin boyutları belirlenir. Elemanın tamamen kapladığı alan için marj, sınır ve padding değerleri de dikkate alınmalıdır.

Dış çizgi (outline) sınırların dışına çizilen, HTML elemanının dikkat çekmesi amaçlı çizgidir. Dış çizgi özellikleri:

- `outline-style`
- `outline-color`
- `outline-width`
- `outline-offset`
- `outline`

Dış çizgi sınırdan farklıdır. Dış çizgi eleman sınırının dışına çizilir. Diğer elemanların üstüne çıkabilir. Dış çizgi eleman boyutlarına dâhil değildir. Elemanın toplam genişlik ve yüksekliği dış çizgiden etkilenmez.

`outline-style` özelliği `border-style` ile aynı değerleri alır. `outline-style` özelliğine atama yapılmazsa diğer dış çizgi özellikleri etkisizdir.

Diğer özellikler:

`outline-width` : Dış çizgi genişliği
`outline-color` : Çizgi rengi
`outline-offset` : Dış çizgi le sınır arasına boşluk ekler.

3.12 Yazı

`color` özelliği yazı rengini belirler. Alacağı değerler renk konusunda (3.5) ele alınmıştır. Sayfa varsayılan yazı rengi body seçicisinde tanımlanandır.

W3C uyumlu CSS kodlamasında `color` özelliği tanımlanırsa, `background-color` özelliği de tanımlanmalıdır.

```
body {
  background-color: lightgrey;
  color: blue;
}

h1 {
  background-color: black;
  color: white;
}
```

`text-align` özelliği yazının yatay hizalamasını tanımlamak için kullanılır. `center, left,right,justify` değerlerini alabilir.

`vertical-align` özelliği düşey hizalama içindir. `top, middle, bottom` değerlerini alabilir.

`text-decoration` yazının görünümünü değiştirmek için kullanılır. `none` değeri tipik olarak bağlantıların altının çizilmemesi amaçlı atanır. `overline, line-through, underline` değerleri atanabilir.

`text-transformation` metnin büyük/küçük harfe dönüştürülmesi için kullanılır. Tüm kelimelerin ilk harflerinin büyük yapılması da mümkündür.

```
p.uppercase {
  text-transform: uppercase;
}

p.lowercase {
  text-transform: lowercase;
}

p.capitalize {
  text-transform: capitalize;
}
```

`text-indent` yazının ilk satırının içten başlamasını sağlar. `letter-spacing` karakterler arasında boşlukları belirlemekte kullanılır. `line-height` özelliği satır aralıklarını belirler. `word-spacing` kelimeler arasındaki boşluklar içindir.

`text-shadow` yazıya gölge ekler. En basit kullanımda yatay ve düşey gölge belirlenir.

```
h1 {
  text-shadow: 2px 2px;
}
```

Renk eklenebilir.

```
h1 {
  text-shadow: 2px 2px red;
}
```

3.12.1 Fontlar

Uygun yazı fontu seçmek web sayfasının görünümü açısından önemlidir. CSS'de 5 genel font ailesi vardır.

- Serif
- Sans-serif
- Monospace
- Cursive
- Fantasy

Çeşitli font isimleri bu font ailelerinden birine aittir. CSS'de `font-family` özelliği yazı fontu belirlemek için kullanılır. Fontlar işletim sistemleri ve web tarayıcıya göre değişir. Bu nedenle belli bir font ile birlikte font ailesini de belirtmek gerekir. Bu şekilde tarayıcı bulamadığı bir fontun yerine benzer bir font seçebilir. Eğer font ismi birden fazla kelimeden oluşuyorsa tırnak içine alınmalıdır.

```
.p1 {
  font-family: "Times New Roman", Times, serif;
}

.p2 {
  font-family: Arial, Helvetica, sans-serif;
}

.p3 {
  font-family: "Lucida Console", "Courier New", monospace;
}
```

Web-güvenli fontlar genel olarak tüm tarayıcılarda ve işletim sistemlerinde yüklü olan fontlardır. Bu nedenle `font-family` özelliğine benzer fontlar eklemek gerekir. İlk font bulunamadığında ikinci, üçüncü font kullanılır. Burada belirtilen son font font ailesi olmalıdır. Yukarıda bu şekilde örnek kodlama görülmektedir.

Çizelge 3.1: Web-güvenli fonlar ve aileleri

Font	**Aile**
Arial	sans-serif
Verdana	sans-serif
Helvetica	sans-serif
Tahoma	sans-serif
Trebuchet MS	sans-serif
Times New Roman	serif
Georgia	serif
Garamond	serif
Courier New	monospace
Brush Script MT	cursive

%100 güvenli (her yerde bulunan) font mümkün değildir. En uygun web-güvenli fontlar Çizelge 3.1'de verilmiştir.

`font-style` özelliği ile yazının eğik olması sağlanır. Üç değer alabilir.

- `normal` : Normal yazı
- `italic` : İtalik (eğik)
- `oblique` : İtalik ile çok benzer, kullanımı yaygın değildir.

`font-weight` özelliği yazının kalın olmasını sağlar. `normal` ve `bold` değerlerini alabilir.

`font-variant` ile yazıda küçük harflerin büyük harflerle, büyük harflerin daha büyük gösterilmesi sağlanabilir. `normal` ve `small-caps` değerlerini alabilir.

`font-size` ile yazı boyutu belirlenir. Alacağı değer mutlak ya da göreceli olabilir.

- Mutlak boyut
  - Yazıyı yüksekliği belirlenen değerde olur.
  - Kullanıcının tarayıcılarda yazı yüksekliğini değiştirmesine izin verilmez.
  - Çıktıda fiziksel yazı yüksekliği belli ise mutlak büyüklük kullanmak yararlıdır.
- Göreceli boyut
  - Çevreleyen elemanlar göre göreceli boyut belirlenir.
  - Kullanıcının tarayıcılarda yazı yüksekliğini değiştirmesine izin verilir.

Yazı yüksekliği piksel biriminde verilebilir. Bir başka birim em olup, 1 em varsayılan yazı yüksekliğidir. Bu değer çoğu tarayıcıda 16 pikseldir. Yüzdelik olarak boyut verildiğinden varsayılan yazı yüksekliğine göre boyut belirlenir. Aşağıdaki örnekleri inceleyin.

```
h1 {
  font-size: 40px;
}
h2 {
  font-size: 1.875em; /* 30px/16=1.875em */
```

```
}
body {
  font-size: 100%;
}
```

`font` özelliği ile bir özellik içinde birden fazla fonlarla ilgili özellik değeri belirlenerek kod kısaltılır. `font` aşağıdaki özellikler için kısaltılmış özelliktir.

- `font-style`
- `font-variant`
- `font-weight`
- `font-size/line-height`
- `font-family`

`font-size` ve `font-family` özelliklerinin değerlerinin belirlenmesi zorunludur.

```
p.a {
  font: 20px Arial, sans-serif;
}

p.b {
  font: italic small-caps bold 12px/30px Georgia, serif;
}
```

3.13 Bağlantı (Link)

Bağlantı elemanları (`<a>`) `color, font-family` vb. CSS özellikleri ile biçimlendirilirler. Bunun yanında durumlarına göre de biçimlendirme yapılabilir. 4 durum söz konusudur.

- `a:link`: Normal, ziyaret edilmemiş
- `a:visited`: Ziyaret edilmiş bağlantı
- `a:hover`: Kullanıcı fare ile üzerinden geçmiş
- `a:active`: Tıklanılan andaki bağlantı

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

a:visited {
  color: green;
}
```

Kodlamada `a:hover`, `a:link` ve `a:visited`'den sonra, `a:active`, `a:hover`'den sonra olmak zorundadır.

`text-decoration` özelliği genellikle bağlantıların altının çizilmemesi için kullanılır.

```
a:link {
  text-decoration: none;
}

a:visited {
  text-decoration: none;
}

a:hover {
  text-decoration: underline;
}

a:active {
  text-decoration: underline;
}
```

3.14 Listeler

HTML'de sıralı ve sırasız olmak üzere iki tür liste vardır. CSS özellikleri ile aşağıdakiler yapılabilir.

- Sıralı ve sırasız liste maddeleri için farklı işaretler kullanılabilir.
- Madde işaretleri için resim kullanılabilir.
- Liste ve maddelere arka plan renkleri atanabilir.

`list-style-type` özelliği ile madde işaretleri belirlenir.

```
ul.a {
  list-style-type: circle;
}

ul.b {
  list-style-type: square;
}

ol.c {
  list-style-type: upper-roman;
}

ol.d {
  list-style-type: lower-alpha;
}
```

`list-style-image` ile madde işaretleri için resim tanımlanır.

```
ul {
  list-style-image: url('sqpurple.gif');
}
```

`list-style-position` ile madde işaretlerinin konumları belirlenir. `inside, outside` değerleri atanabilir.

Kısaltılmış `list-style` özelliği ile birden çok özellik birlikte tanımlanır. Buradaki sıra:

- `list-style-type`
- `list-style-position`
- `list-style-image`

```
ul {
  list-style: square inside url("sqpurple.gif");
}
```

Listelerde `color,padding` vb. özellikler kullanılabilir.

3.15 Tablolar

CSS ile HTML tablolarının görünümleri önemli derecede iyileştirilebilir. Tablo sınırları `border` özelliği ile belirlenir.

```
table, th, td {
  border: 1px solid black;
}
```

`width` ve `height` özelliği ile tablo genişliği ve yüksekliği belirlenebilir. Örneğin `width:100%` ile tablonun genişliği sayfa genişliği kadar olur. Piksel biriminde değer ataması da yapılabilir.

Tablo çizgileri hem `<table>` hem de `<th>,<td>` elemanlarının kendi sınırları olduğundan çift çizgidir. Bunu `border-collapse:collapse` ataması ile tek çizgi yapmak mümkündür. Yalnızca tablonun çizgileri görüntülensin isteniyorsa tablo elemanının border özelliğine atama yapılmalıdır.

`text-align`, `vertical-align` özellikleri tablolarda da kullanılabilir. Sınırlar ve içerik arasındaki boşluk `<td>, <th>` elemanlarında `padding` özelliği ile kontrol edilir. Satırlarda yatay çizgiler için `<td>, <th>` elemanlarında `border-bottom` özelliği kullanılır.

```
th, td {
  border-bottom: 1px solid #ddd;
}
```

`:hover` seçici `<tr>` elemanında uygulandığında fare satırların üzerine geldiğinde vurgulanma sağlanır.

```
tr:hover {background-color: #f5f5f5;}
```

`nth-child()` seçici ile `background-color` kullanılarak zebra desenli tablo tanımlanabilir.

```
tr:nth-child(even) {background-color: #f2f2f2;}
```

Tablolarda `color` ve `background-color` özellikleri kullanılır.

3.16 CSS Sayfa Düzeni (Layout)

`display` sayfa düzeni bakımından en önemli özelliktir. Bir elemanın gösterilip gösterilmeyeceğini ve basıl gösterileceğini kontrol eder. Her HTML elemanının varsayılan display özelliği tanımlıdır. Çoğu elemanın display özelliği `block` ya da `inline`'dır.

Blok düzeyi elemanlar yeni bir satırda başlar. Mümkün olan tüm genişliği kaplar. `div` blok düzeyinde bir elemandır. Diğer tipik blok düzeyi elemanlar:

- `<h1>...<h6>`
- `<p>`
- `<form>`
- `<header>`
- `<footer>`
- `<section>`

Satır İçi (inline) elemanlar yeni bir satırdan başlamazlar. Gerektiği kadar yer kaplarlar. Tipik satır içi elemanlar:

- `<span>`
- `<a>`
- `<img>`

`display:none` JavaScript'te elemanları silmeden ve yeniden oluşturmadan gizlemek ve göstermek için yaygın kullanılır. `<script>` elemanının varsayılan ayarı `display:none`'dır. Varsayılan display değeri değiştirilebilir. Yaygın bir örnek `<li>` elemanının blok düzeyinden satır içi düzeyine dönüştürülerek listenin yatay görünmesinin sağlanmasıdır (`li { display:inline}`).

Önceden de belirlendiği gibi blok elemanları sayfa genişliğini kaplar. `width` özelliği ile elemanın genişliği kontrol edilebilir. `margin:auto` atamasıyla elemanın kendisini kapsayan elemana göre yatay olarak ortalanması sağlanabilir. `<div>` elemanında tarayıcı penceresi küçük ise tarayıcı bir kaydırma çubuğu (scrollbar) ekler. `width` yerine `max-width` kullanılırsa tarayıcı bu tür durumları daha iyi kontrol eder.

```
div.ex1 {
  width: 500px;
  margin: auto;
  border: 3px solid #73AD21;
}

div.ex2 {
  max-width: 500px;
  margin: auto;
  border: 3px solid #73AD21;
}
```

Bölüm 4

JavaScript Programlama Dili

4.1 Giriş

JavaScript dünyanın en tanınan programlama dili olup, aynı zamanda HTML'in programlama dilidir. JavaScript ile HTML elemanları değiştirilebilir. HTML DOM (Document Object Model) HTML elemanlarına erişim için resmi W3C standardıdır. Aşağıdaki JavaScript kodu id özniteliği "demo" olarak (id="demo") tanımlanmış bir HTML elemanının içeriğini değiştirmektedir.

Java ile JavaScript birbirinden faklı programlama dilleridir.

```
document.getElementById("demo").innerHTML = "Hello JavaScript";
```

`document.getElementById()` HTML DOM yöntemlerinden (method) biridir.

JavaScript ile

- HTML elemanları değiştirilebilir,
- silinebilir,
- yeni HTML elemanları oluşturulabilir,
- HTML elemanları kopyalanabilir, klonlanabilir.

Aşağıdaki örnekte `<image>` elemanının `src` özniteliğinin değeri değiştirilmektedir. JavaScript ile hemen hemen tüm HTML elemanlarının öznitelikleri değiştirilebilir.

```
<!DOCTYPE html>
<html>
<body>
<script>
function changeImage() {
```

```
    var image = document.getElementById('myImage');
    if (image.src.match("bulbon")) {
        image.src = "pic_bulboff.gif";
    } else {
        image.src = "pic_bulbon.gif";
    }
}
</script>
<img id="myImage" onclick="changeImage()" src="pic_bulboff.gif" width="100" height="180">
<p>Click the light bulb to turn on/off the light</p>
</body>
</html>
```

Yukarıdaki örnekten görüleceği üzere, JavaScript söz dizimi olarak C diline çok benzer.

JavaScript ile HTML stilleri değiştirilebilir. Bir HTML elemanının stilini değiştirmek, özniteliğini değiştirmek gibidir. Örnek:

```
document.getElementById("demo").style.fontSize = "25px";
```

4.2 JavaScript Kodlarının Yeri

HTML dilinde JavaScript kodları `<script>` ve `</script>` etiketleri arasında yer alır. Bu etiketler kodun nerede başlayıp nerede bittiğini belirtirler. Kodlar HTML sayfasının hem `<body>` hem de `<head>` bölümünde yer alabilir. Örnek:

```
<script>
function myFunction() {
    document.getElementById("demo").innerHTML = "My First JavaScript Function";
}
</script>
```

JavaScript kodları genel olarak bir olay (event) gerçekleşirse (örneğin kullanıcı bir butona bastığında) icra edilirler. Bir fonksiyon içinde bulunan kod daha sonra bir olay gerçekleştiğinde çalıştırılabilir. Fonksiyonlar ve olaylar konusunda ilerleyen bölümlerde daha fazla bilgi verilecektir.

`<head>` ya da `<body>` içinde JavaScript kodu:

- Bir HTML dokümanı içinde sınırsız sayıda skript (betik) bulunabilir.
- Skriptler hem `<head>` hem de `<body>` bölümlerinde ya da her ikisinde olabilir.
- Genellikle skriptler ya `<head>` bölümünde ya da `<body>` bölümünün sonunda yer alır.
- Skriptlerin belli bir bölümde yer alması kodun okunabilirliği açısından yararlıdır.

Aşağıdaki örnekte `<head>` bölümünde yer alan bir fonksiyon, bir butona basılınca çağrılmaktadır.

```
<!DOCTYPE html>
<html>
<head>
<script>
```

```
function myFunction() {
    document.getElementById("demo").innerHTML = "Paragraph changed.";
}
</script>
</head>
<body>
<h1>My Web Page</h1>
<p id="demo">A Paragraph</p>
<button type="button" onclick="myFunction()">Try it</button>
</body>
</html>
```

Aynı kod `<body>` bölümünde de yer alabilir.

```
<!DOCTYPE html>
<html>
<body>
<h1>My Web Page</h1>
<p id="demo">A Paragraph</p>
<button type="button" onclick="myFunction()">Try it</button>
<script>
function myFunction() {
   document.getElementById("demo").innerHTML = "Paragraph changed.";
}
</script>
</body>
</html>
```

- JavaScript kodları dış dosyalarda da yer alabilir.
- Bu şekilde çalışmak aynı kodların birden fazla HTML sayfasında kullanılması açısından yararlıdır.
- JavaScript dosya uzantısı "js" dir.
- Dış dosya adı `<script>` etiketinin src özniteliğinde belirtilir.

```
<!DOCTYPE html>
<html>
<body>
<script src="myScript.js"></script>
</body>
</html>
```

4.3 JavaScript'de Çıkış

JavaScript yazdırma (print) ya da çıkış (output) fonksiyonlarına sahip değildir. Yalnızca HTML elemanlarını değiştirmek için kullanılır.

HTML elemanlarını değiştirmek için `document.getElementById(id)` yöntemi kullanılır. Hangi HTML elemanı üzerinde işlem yapılacağı elemanın `id` özniteliği ile belirlenir. `innerHTML` elemanın içeriğini gösterir. Yukarıdaki örnekleri inceleyin.

- `<script> </script>` etiketleri arasındaki JavaScript ifadeleri web tarayıcı tarafından icra edilir.
- `document.getElementById("demo")` bir HTML elemanını id özniteliği yardımıyla bulmak için gerekli koddur.
- `innerHTML = "Paragraph changed."` elemanın HTML içeriğini değiştirmek için gerekli koddur.

Bu bölümdeki örneklerde genel olarak id özniteliği "demo" olan bir paragraf elemanı kullanılacaktır.

Test amaçlı olarak doğrudan HTML dokümanına yazmak da mümkündür.

```
<script>
document.write(Date());
</script>
```

Kullanılan web tarayıcısı debug (hata ayıklama) desteği sağlıyorsa `console.log()` yöntemi ile bazı değerler yazdırılabilir. Debug moduna geçmek için web tarayıcıda F12 tuşuna basılarak gelen menüde "Console" seçilir.

```
<!DOCTYPE html>
<html>
<body>
<h1>My First Web Page</h1>
<script>
a = 5;
b = 6;
c = a + b;
console.log(c);
</script>
</body>
</html>
```

4.4 JavaScript Sözdizimi

JavaScript HTML sayfalarına dinamik bir yapı kazandırmak için geliştirilmiş bir programlama dilidir. Kullanıcı tarafında web tarayıcı altında çalıştığı için aynı zamanda bir skript (betik) dilidir. Skript dilleri derlenmez, yorumlanırlar.

Bir programlama dilinde cümleler, bilgisayar deyimlerine ya da sadece deyimlere karşılık gelir.

JavaScript sözdizimi `C` diline çok yakındır.

Sabitler

Sayısal sabitler ondalık noktalı, noktasız ya da bilimsel biçimde yazılabilir.

```
3.1415
1002
321e6
```

Dizge (string) sabitler çift ya da tek tırnaklar arasında yer alır.

```
"Ali Ak"
```

Dizi sabitler:

```
[10,5,100,75]
```

Objeler:

```
{firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"}
```

Fonksiyonlar:

```
function myFunction(a, b) { return a * b;}
```

Değişkenler

Bir programlama dilinde bir isim verilmiş değişkenler verileri saklamak için kullanılır. JavaScript'de değişken tanımlamak için `var` anahtar kelimesi kullanılır. Değişkenlere atama eşittir işareti ile yapılır.

```
var x, length
x = 5
length = 6
```

JavaScript kodunun icra edilmesi sırasında değişkenlerin değerleri değişir, sabitler değişmez.

Operatörler

Aritmetik hesaplamalar için aritmetik operatörler kullanılır (`+ - * /`). Atama operatörü (`=`) ile değişkenlere atama yapılır.

İlişkisel ve mantıksal operatörler: `== != < >`

Deyimler

HTML'de JavaScript deyimleri HTML tarayıcısına gönderilen bir dizi komuttan oluşur. İfadeler birbirinden ; ile ayrılır (sonlandırıcı).

```
x=y+5;
z=x*5;
```

Anahtar Kelimeler ve Belirleyiciler

Deyimler (statements) genellikle bir anahtar kelime (keyword) ile başlar. Sık kullanılan anahtar kelimelerden `var` web tarayıcıya yeni bir değişken yaratması gerektiğini söyler.

```
var x=5;
var y=8*x;
```

Tüm programlama dilleri değişkenleri, fonksiyonları ve objeleri özgün isimleri ile belirlemek zorundadır. Bu özgün isimler belirleyiciler (identifers) olarak adlandırılırlar. İsimler harfler, rakamlar, alt çizgi ve dolar işaretleri içerebilirler, sayı ile başlayamazlar. Rezerve edilmiş kelimeler (anahtar kelimeler gibi) isim olarak verilemez (belirleyici olamazlar).

Yorum Satırları

JavaScript kodunun tamamı komutlardan oluşmaz. Çift bölü işaretlerinden sonraki satırlar yorum satırlarıdır. Web tarayıcı tarafından dikkate alınmaz.

```
// Yorum satırı
```

Veri Türleri

Değişkenler, sayılar, metin dizgeleri (text string), diziler, objeler gibi çeşitli veri türlerini tutabilirler.

```
var length = 16;                                // Sabit sayı atanıyor
var points = x * 10;                            // Bir ifadenin sonucu olarak sayı atanıyor
var lastName = "Johnson";                       // Dizge
var cars = ["Saab", "Volvo", "BMW"];            // Dizi
var person = {firstName:"John", lastName:"Doe"}; // Obje sabiti ataması
```

Programlamada veri türleri önemli bir kavramdır. İlerleyen bölümlerde bu konuda daha fazla bilgi verilecektir.

Fonksiyonlar

Bir fonksiyon içine yazılmış kodlar bir çok kez icra edilebilir (yeniden kullanılabilir). Örneğin aşağıdaki fonksiyon argüman olarak aldığı iki değişkenin çarpımını geri döndürmektedir.

```
function myFunction(x, y) {
    return x * y;
}
```

Büyük/Küçük Harf Duyarlılığı, Karakter Seti

- Tüm belirleyiciler büyük-küçük harf duyarlıdır.
- `lastName` ile `lastname` değişkenleri iki farklı değişkendir.
- `myFunction` ve `myfunction` farklı fonksiyonlardır.
- `Var` `var` olarak kabul edilmez.
- JavaScript **Unicode** karakter setini kullanır.

4.5 Deyimler

HTML'de JavaScript deyimleri web tarayıcıya iletilen komutlardır. Bu şekilde web tarayıcıya ne yapması gerektiği bildirilir. Noktalı virgül deyimleri ayırt etmek için kullanılır. Bu şekilde aynı satırda birden çok deyim kullanılması da mümkündür.

```
a=5;
b=7;
c=b+a;
```

```
a=5;b=7;c=b+a;
```

Noktalı virgül ile sonlandırılmamış JavaScript kodlarına rastlamak mümkündür. Sonlandırıcı kullanımı zorunlu olmamasına rağmen, kullanılmaması durumunda kodlar genellikle çalışmaz.

Her bir deyim kod içerisinde yazıldığı sıra ile icra edilir.

Kod Blokları

Javascript deyimleri bloklar halinde gruplanabilirler. Bloklar küme parantezleri arasındadır, { ile başlar } ile biter.

```
function myFunction() {
    document.getElementById("demo").innerHTML = "Hello Dolly.";
    document.getElementById("myDIV").innerHTML = "How are you?";
}
```

Deyim Belirleyicileri

- JavaScript deyimleri bazen belirleyiciler ile başlar.
- Deyim belirleyicileri rezerve edilmiş kelimelerdir ve isimlendirmede kullanılamazlar.
- Bazı belirleyiciler aşağıda listelenmiştir.

Deyim	Açıklama
`break`	Döngüyü keser.
`continue`	Döngüyü bir sonraki aşamaya atlatır.
`do ... while`	Döngü oluşturur.
`for`	Döngü oluşturur.
`function`	Fonksiyon tanımlama

Boşluklar, Satır Uzunlukları ve Satır Sonları

Kodun çalıştırılması sırasında fazladan boşluklar dikkate alınmaz. Kodun okunabilirliği açısından fazladan boşluklar kullanılabilir. Aşağıdaki iki satır arasında bir fark yoktur.

```
var person = "Hege";
var person="Hege";
```

Programcılar okunabilirlik açısından satır uzunluklarında 80 karakteri geçmezler. Deyimler ; ile sonlandığından uzun satırlar ikiye bölünebilir.

```
document.getElementById("demo").innerHTML =
    "Hello Dolly.";
```

Kod satırları bir dizgede ters bölü (\) kullanılarak bölünebilirler.

```
document.getElementById("demo").innerHTML = "Hello \
    Dolly!";
```

Ancak dizge dışında ters bölü ile satırlar bölünmez.

4.6 Yorum Satırları

- Yorumlar kodu açıklamak ve kodun okunabilirliğini artırmak için kullanılır.
- Kodun bir kısmı da test amaçlı olarak yorum haline getirilebilir. Böylece bu kısım icra edilmez.
- Tek satırlık yorumlar // ile başlar. // ile satır sonu arasındaki metin kodun çalıştırılması sırasında dikkate alınmaz.
- Birden çok satırdan oluşan yorumlar `/*` ile başlar, `*/` ile biter.

Örnekler:

```
// Change heading:
document.getElementById("myH").innerHTML = "My First Page";
// Change paragraph:
document.getElementById("myP").innerHTML = "My first paragraph.";

var x = 5;      // Declare x, give it the value of 5
var y = x + 2;  // Declare y, give it the value of x + 2

/*
The code below will change
the heading with id = "myH"
and the paragraph with id = "myp"
in my web page:
*/
document.getElementById("myH").innerHTML = "My First Page";
document.getElementById("myP").innerHTML = "My first paragraph.";

//document.getElementById("myH").innerHTML = "My First Page";
document.getElementById("myP").innerHTML = "My first paragraph.";

/*
document.getElementById("myH").innerHTML = "My First Page";
document.getElementById("myP").innerHTML = "My first paragraph.";
*/
```

4.7 Değişkenler

Değişkenler bilgi saklamak için kullanılan taşıyıcılardır. Matematikte değişken kavramı ile programlamada değişken kavramları benzerdir.

- Değişkenler kısa (x,y gibi) ya da uzun (toplamHacim, ortalamaX gibi) isimler alabilirler.
- Değişken isimleri harfler, rakamlar, `$` ve `_` içerebilir.
- Değişken isimleri rakamla başlayamaz, harfle başlar. `$` ve `_`ile başlayabilir, ancak tercih edilmez.
- Değişken isimleri büyük küçük harf duyarlıdır. x ve X farklı değişkenlerdir.
- Rezerve edilmiş kelimeler (anahtar kelimeler) değişken ismi olarak verilemez.

Atama Operatörü

JavaScript'te `=` atama operatörüdür. Eşittir anlamında değildir. Aşağıdaki kod matematik olarak yanlış olmasına rağmen programlama açısından doğrudur. `x` değişkeninin değerini 6 artırıp kendine atama yapmak anlamındadır.

```
x=x+6;
```

Eşittir operatörü ise `==` dir.

Veri türleri

- Değişkenler metin ya da sayısal bilgileri tutabilirler.
- Metin bilgileri dizge (string) ya da metin dizgesi olarak adlandırılır.
- Pek çok veri türü olabilir. Bu aşamada sayıları ve dizgeleri göz önüne almak yeterlidir.
- Eğer bir değişkene dizge ataması yapılacak ise çift ya da tek tırnak arasına almak gerekir.
- Sayısal değerler için tırnak kullanılmaz.
- Sayısal değerler tırnak içine alınırsa sayı olarak değil metin olarak dikkate alınırlar.

```
var pi = 3.14;
var person = "John Doe";
var answer = 'Yes I am!';
var n=10;
```

Değişken Tanımlama

Değişkenler `var` anahtar kelimesi kullanılarak tanımlanır.

```
var carName;
```

Tanımlama sonrası değişken boş (empty) durumdadır. Değişkene bir değer ataması yapmak için atama operatörü kullanılır.

```
carName="Audi";
```

Tanımlama ve atama birlikte de yapılabilir.

```
var carName="Ford";
```

Birden çok değişken bir satırda tanımlanabilir. Bu durumda deyim `var` ile başlar, değişkenler birbirinden virgül ile ayrılır.

```
var lastName = "Doe", age = 30, job = "carpenter";
```

Ya da

```
var lastName = "Doe",
age = 30,
job = "carpenter";
```

Programlamada değişkenler bir değer içermeden tanımlanırlar. Bu tür değişkenler daha sonra değer alacaklardır. Tanımlanmış ancak herhangi bir değer atanmamış değişkenlerin değeri `undefined`'dır.

Bir değişken yeniden tanımlanırsa, değerini kaybetmez. Aşağıdaki iki deyim icra edildikten sonra `carName` değişkeninin değeri "Volvo" dur.

```
var carName = "Volvo";
var carName;
```

const ve let

2015'e kadar değişken tanımlamanın tek yolu `var` anahtar kelimesi idi. 2015 sonrası `const` ve `let` ile değişken tanımlama geldi (ES6 - ECMAScript 2015 versiyonu). `const` ile tanımlanan değişkenlerin değeri değiştirilemez, bu yolla sabit tanımlaması yapılır. `let` ise `var` ile benzer şekilde değişken tanımlamak için kullanılır. Aralarında bilinirlik alanı bakımından farklar vardır. Bu dokümanda genel olarak `var` ile örnekler verilecektir. Eski web tarayıcılarda `const` ve `let` desteği olmadığından özel bir gerekçe yok ise `var` tercih edilmesinde yarar vardır.

4.8 Veri Türleri

JavaScript dinamik türlere sahiptir. Bir değişken farklı türler için kullanılabilir.

```
var x;               // x değeri undefined
var x = 5;           // Şimdi x bir sayı
var x = "John";      // Şimdi x bir dizge
```

Dizgeler

Dizge (string) bir dizi karakteri (örneğin "Ali Akdeniz") saklayan bir değişkendir. Dizge sabitler tek ya da çift tırnaklar arasında yazılırlar. Dizgeyi sınırlayan tırnaklarla karışmadıkça dizge içinde tek ya da çift tırnak kullanılabilir.

```
var carName = "Volvo XC60";   // çift tırnak
```

```
var carName = 'Volvo XC60';   // tek tırnak

var answer = "It's alright";             // Çift tırnak içinde tek tırnak
var answer = "He is called 'Johnny'";    // Çift tırnak içinde tek tırnaklar
var answer = 'He is called "Johnny"';    // Tek tırnak içinde çift tırnaklar
```

Sayılar

JavaScript'te tek tür sayı kullanılır. Reel ve tamsayı ayrımı yoktur. Sayılar ondalık noktalı, noktasız ya da bilimsel (üstel) yazılabilirler.

```
var x1=34.00;
var x2=34;
var z=123e5;
var t=123e-5;
```

Mantıksal Tür (Boolean)

Mantıksal veri türünde iki değer söz konusudur: Doğru (true) ve Yanlış (false).

```
var x = true;
var y = false;
```

Diziler

Diziler (array) köşeli parantezlerle tanımlanırlar. Dizilere sabit değer atamalarında elemanlar virgül ile ayrılır.

```
var cars = ["Saab", "Volvo", "BMW"];
```

Objeler

Objeler küme parantezleri arasında tanımlanırlar. Obje özellikleri (property) isim:değer biçiminde virgüllerle ayrılarak yazılırlar.

```
var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};
```

Undefined ve Null

Değer atanmamış bir değişkenin değeri `undefined` olur. Değişkenlere `null` değeri atanarak içleri boşaltılabilir.

```
var cars;              // Değer undefined
var person = null;     // Değer null
```

typeof Operatörü

Bir değişkenin türünü belirlemek için `typeof` operatörü kullanılır.

İfade	Geri dönüş değeri
`typeof "John"`	string
`typeof 3.14`	number
`typeof false`	boolean
`typeof [1,2,3,4]`	object
`typeof {name:'John', age:34}`	object

Diziler de bir tür obje olduğundan `typeof` operatörü `object` değeri döndürür.

4.9 Objeler

Objeler özellikler ve yöntemler eklenmiş verilerdir.

- Özellikler (properties) objelerle ilgili değerlerdir. Obje altında tanımlı değişkenler gibi düşünülebilir.
- Yöntemler (methods) objelerin gerçekleştirebileceği eylemlerdir.

Gerçek hayattan bir obje örneği: Araba

Obje (object)	Özellikler (properties)	Yöntemler (methods)
car	`car.name = Fiat`	`car.start()`
	`car.model = 500`	`car.drive()`
	`car.weight = 850kg`	`car.brake()`
	`car.color = white`	`car.stop()`

- Tüm arabalar aynı özelliklere sahiptir, ancak özelliklerin değerleri değişmektedir.
- Tüm arabalar aynı yöntemlere sahiptir, ancak yöntemler farklı zamanlarda uygulanır.

JavaScript'te objeler yöntemleri ve özellikleri olan veriler ya da değişkenlerdir. Hemen her şey obje olarak değerlendirilir (Dates, Arrays, Strings, Functions ...). Kullanıcı tanımlı objeler oluşturmak da mümkündür. Aşağıdaki örnekte "person" adlı bir obje oluşturulmakta ve ona dört özellik eklenmektedir.

```
var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};
```

Obje tanımlaması birden çok satıra da yayılabilir.

```
 var person = {
    firstName:"John",
    lastName:"Doe",
    age:50,
    eyeColor:"blue"
};
```

Yeni objeler tanımlamak için farklı yollar da vardır. Halihazırda var olan objelere yeni özellikler ve yöntemler de eklenebilir.

Obje özelliklerine iki farklı yolla erişilebilir.

```
person.lastName;
person["lastName"];
```

Obje yöntemleri aşağıdaki söz dizimine göre çağrılır.

```
objectName.methodName()
```

Örnek:

```
Name = person.fullName();
```

4.10 Fonksiyonlar

Bir JavaScript fonksiyonu belli bir görevi yerine getirmek üzere düzenlenmiş bir kod blokudur. Fonksiyonlar bir şekilde tetiklendikleri (çağrıldıkları) zaman icra edilirler.

```
function myFunction(p1, p2) {
    return p1 * p2;           // Bu fonksiyon p1 ile p2 çarpımını geri döndürür.
}
```

Fonksiyon Sözdizimi

- Bir JavaScript fonksiyonu sıra ile `function` anahtar kelimesi, fonksiyon ismi ve parantezlerle () tanımlanır.
- Fonksiyon isimlendirme kuralları değişken isimlendirme kuralları ile aynıdır.
- Fonksiyon parantezi içinde virgülle ayrılmış olarak parametreler ya da parametre değişkenleri yer alabilir.
- Fonksiyon parametreleri fonksiyon tanımlamasında listelenmiş isimlerdir.
- Fonksiyon argümanları fonksiyon çalıştığında (icra edildiğinde) fonksiyonun aldığı gerçek değerlerdir.
- Fonksiyon tarafından çalıştırılacak kod küme parantezleri `{ }` arasında yer alır.
- Fonksiyon içinde argümanlar yerel değişkenler olarak kullanılır.

```
functionName(parameter1, parameter2, parameter3) {
    çalıştırılacak kod
}
```

Fonksiyonların Çağrılması

Fonksiyon içindeki kod fonksiyon çağrıldığı zaman icra edilir. Bu üç şekilde olur.

- Bir olay gerçekleşirse (Örneğin kullanıcı bir butona basarsa)

- Fonksiyon başka bir kod içinden çağrılırsa
- Otomatik olarak (self invoked)

Geri Dönüş Değeri

Fonksiyon kodunda `return` deyimine ulaşıldığında fonksiyonun çalışması durur. Eğer fonksiyon bir deyim içinden çağrılmış ise fonksiyonu çağıran deyimden sonraki kod çalıştırılmaya devam edilecektir. Fonksiyonlar çoğunlukla bir geri dönüş değeri hesaplarlar. Geri dönüş değeri çağıran koda iletilir.

```
var x = myFunction(4, 3);        // Fonksiyon çağırılıyor, geri dönüş
                                 // değeri x değişkenine atanıyor.

function myFunction(a, b) {
    return a * b;                // Fonksiyon a ve b nin
}                                // çarpımını geri döndürüyor.
```

Fonksiyon geri dönüş değeri çağrılan kod içinde değişken gibi kullanılabilir. Başka bir deyişle fonksiyon geri dönüş değeri operatörler ile birlikte sağ taraf değeri (atama operatörünün sağ tarafında) olarak kullanılabilir.

```
text = "The temperature is " + toCelsius(32) + " Centigrade";
```

Neden Fonksiyon?

- Kod tekrar kullanılabilir. Kodu bir kez tanımlayıp, çok kez kullanabilirsiniz.
- Aynı kod farklı argümanlar ile pek çok kez kullanılıp, farklı sonuçlar üretilebilir.

4.11 Faaliyet Alanı (Scope)

Faaliyet alanı, kod içinde değişkenlere erişilebilen bölgeyi ifade eder. JavaScript'te fonksiyonlar ve objeler aynı zamanda değişkenlerdir. Bu bağlamda JavaScript'de faaliyet alanı değişkenlere, fonksiyonlara ve objelere erişilebilen bölgeyi ifade eder.

JavaScript fonksiyon faaliyet alanına sahiptir. Bir fonksiyon içinde tanımlanmış değişkenler fonksiyon içinde yereldir.

```
 // carName erişilebilir değil

function myFunction() {
    var carName = "Audi";
    // carName erişilebilir
}
```

Yerel değişkenler yalnızca tanımlandıkları fonksiyonlar içinde geçerli olduğundan aynı isimli değişkenler farklı fonksiyonlar içinde kullanılabilir. Yerel değişkenler fonksiyon çalıştığında yaratılır. Fonksiyon sona erdiğinde silinirler.

Fonksiyonların dışında tanımlanan değişkenler globaldir. Global bir değişken global faaliyet alanına sahiptir. Web sayfası içindeki tüm fonksiyonların içinden global değişkenlere erişilebilir.

```
var carName = " Volvo";

// carName erişilebilir.

function myFunction() {

    // carName erişilebilir.
}
```

Tanımlanmamış bir değişkene atama yapıldığında otomatik olarak global faaliyet alanına sahip olur.

```
// carName erişilebilir.

function myFunction() {
    carName = "Volvo";

    // carName erişilebilir.
}
```

Fonksiyon argümanları (parametreler) fonksiyon içinde yereldir.

Değişkenlerin Ömrü

- Değişkenlerin ömürleri tanımlandıkları zaman başlar.
- Yerel değişkenler fonksiyonun çalışması sona erdiğinde silinirler.
- Global değişkenler sayfa kapanınca silinirler.

4.12 Olaylar (Events)

HTML olayları HTML elemanlarında oluşur. HTML sayfalarında JavaScript kullanılırsa JavaScript bu olaylara tepki verir.

Bir HTML olayı web tarayıcı ya da kullanıcının bir eylem yapması ile oluşur. Bir web sayfasının yüklenmesinin tamamlanması, bir input alanının değişmesi, bir butona basılması HTML olaylarına örnek olarak verilebilir. JavaScript ile bir olay olduğunda bir kodun icra edilmesi mümkündür. HTML elemanlarına JavaScript kodları ile olay işleyici öznitelikler eklenmesi mümkündür.

```
<some-HTML-element some-event='some JavaScript'>
<some-HTML-element some-event="some JavaScript">
```

Aşağıdaki örneklerde bir buton elemanına `onclick` özniteliği eklenmektedir. İkinci kod butonun kendinde değişiklik yapmaktadır.

```
<button onclick='getElementById("demo").innerHTML=Date()'>The time is?</button>
<button onclick="this.innerHTML=Date()">The time is?</button>
```

Olay işleme amaçlı özniteliklere atanan JavaScript kodları çoğu kez tek satırdan oluşmaz. Böyle durumlarda fonksiyon çağrısı yapılmalıdır.

```
<button onclick="displayDate()">The time is?</button>
```

Yaygın kullanılan HTML olayları:

Olay (Event)	Açıklama
onchange	Bir HTML elemanı değişiyor.
onclick	Kullanıcı bir HTML elemanına tıklıyor.
onmouseover	Kullanıcı fareyi bir HTML elemanı üzerinde sürüklüyor.
onmouseout	Kullanıcı fareyi bir HTML elemanının üzerinden uzaklaştırıyor.
onkeydown	Kullanıcı klavyeden bir tuşa basıyor.
onload	Web tarayıcı sayfanın yüklenmesini tamamlıyor.

HTML olaylarının tam listesi için `http://www.w3schools.com/jsref/dom_obj_event.asp` sayfasından yararlanılabilir.

4.13 Dizgeler

Metinleri saklamak ve yönetmek için dizgeler (string) kullanılır. javaScript "Ali Ak" gibi bir dizi karakteri basitçe saklayabilir. Bir dizge iki tırnak (tek ya da çift tırnak) arasındaki herhangi bir metindir. Dizgeyi sınırlayan tırnakla karışmamak üzere metin içinde tek ya da çift tırnak kullanılabilir. \ karakteri(escape karakteri) kullanılarak da dizge içinde tırnak kullanılabilir.

```
var answer = "It's alright";
var answer = "He is called 'Johnny'";
var answer = 'He is called "Johnny"';

var answer = 'It\'s alright';
var answer = "He is called \"Johnny\""
```

Dizge Uzunluğu

Bir dizge değişkeni ya da dizge objesinin uzunluğu (karakter sayısı) yerleşik özelliklerden `length` kullanılarak bulunur.

```
var txt = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
var sln = txt.length;
```

Özel Karakterler

Dizgeler iki tırnak arasında yazılırlar. Bazı karakterler (tek ya da çift tırnak gibi) iki tırnak arasına program kodunda özel anlamları olduğundan yazılamaz, önlerine \ konularak yazılır. Aşağıdaki tabloda bu tür karakterler listelenmiştir.

Code	Outputs
\'	tek tırnak
\"	çift tırnak
\\	tersbölü
\n	yeni satır
\r	carriage return
\t	tab
\b	backspace
\f	form feed

Dizgeler obje olabilirler. Ancak dizgeleri obje olarak tanımlamaktan kaçınmak gerekir. Kodun çalışma hızını olumsuz etkilerler ve çeşitli yan etkileri vardır.

```
var x = "John";               //türü dizge
var y = new String("John");  //türü obje
```

Dizge Özellikleri ve Yöntemleri

Dizge özellikleri ve yöntemleri Çizelge 4.1 ve 4.2 özet olarak verilmiştir. Bazı örnek kodlar da aşağıda yer almaktadır.

Dizge içinde dizge bulma: `indexOf()` yöntemi belirtilen metinin bir dizge içinde başladığı yerin (ilk karakterin) konumunu verir.

```
var str = "Please locate where 'locate' occurs!";
var pos = str.indexOf("locate");
```

`lastIndexOf()` yöntemi ise aranan metnin son geçtiği konumu (metinin ilk karakteri) verir.

```
var str = "Please locate where 'locate' occurs!";
var pos = str.lastIndexOf("locate");
```

- Her iki yöntem de aranan metin bulunmaz ise -1 değerini verir.
- JavaScript'de hem dizlerde hem de dizgelerde indis sıfırdan başlar. Bu bağlamda bir dizgenin ilk karakterinin konumu (indisi) sıfırdır.
- `indexOf()` ve `search()` yöntemleri eşdeğerdir. Argümanları ve geri dönüş değerleri aynıdır. Ancak `search()` daha güçlüdür.

Dizgelerin bir parçasının ayrılması için üç yöntem vardır: `slice(start,end)`, `substring(start,end)`, `substr(start,length)`

Çizelge 4.1: Dizge Özellikleri

Özellik (Property)	Tanım
constructor	Dizge objenin prototipini yaratan fonksiyonu döndürür.
length	Uzunluk
prototype	Objeye yöntemler ve özellikler eklemeye olanak verir.

Çizelge 4.2: Dizge Yöntemleri

Yöntem (Method)	Tanım
charAt()	Verilen konumdaki karakteri verir.
charCodeAt()	Verilen konumdaki karakterin unicode değerini verir.
concat()	Birden çok dizgeyi birleştirir, sonuç olarak dizge geri döndürür.
fromCharCode()	Unicode değerini karaktere dönüştürür.
indexOf()	Bir dizgede belirtilen değerin bulunduğu ilk konumu verir.
lastIndexOf()	Bir dizgede belirtilen değerin bulunduğu son konumu verir.
localeCompare()	Geçerli bölgesel ayarlara göre iki dizgeyi karşılaştırır.
match()	Dizge içinde dizge arama
replace()	Bul değiştir işlemi yaparak değiştirilmiş dizgeyi geri döndürür.
search()	Bir dizgede bir değer arar ve bulunduğu ilk konumu geri döndürür.
slice()	Bir dizgenin bir parçasını ayırarak yeni bir dizge olarak geri döndürür.
split()	Bir dizgenin elemanlarını bir diziye dönüştürür.
substr()	Bir dizgenin bir parçasını verilen bir konumdan verilen karakter kadar ayırır.
substring()	Bir dizgenin verilen iki konum arasındaki kısmını ayırır.
toLocaleLowerCase()	Bölgesel ayarları gözeterek küçük harfe dönüştürme
toLocaleUpperCase()	Bölgesel ayarları gözeterek büyük harfe dönüştürme
toLowerCase()	Küçük harfe dönüştürme
toString()	Returns the value of a String object
toUpperCase()	Büyük harfe dönüştürme
trim()	Bir dizgenin başıdaki ve sonundaki boşlukları yok etme
valueOf()	Dizgedeki sayısal değeri verir

`slice()` bir dizgenin bir parçasını ayırarak geri döndürür. İki parametre alır: Başlangıç ve bitiş konumu (indisi). Aşağıdaki örnekte kod çalıştıktan sonra res değişkeninin değeri "Banana" olur.

```
var str = "Apple, Banana, Kiwi";
var res = str.slice(7,13);
```

Parametrelerden biri negatif ise konum sondan başa doğru sayılır. yukarıdaki örnekte ikinci satır `var res = str.slice(-12,-6);` olursa res değişkeninin değeri yine "Banana" olur. İkinci parametre boş bırakılırsa verilen konumdan dizge sonuna kadar ayırma yapılır. Bu durumda ilk parametre negatif verilirse verilen konumdan başlangıca kadar ayırma yapılır.

Uyarı: Negatif konumlar Internet Explorer 8.0 ve üstü sürümlerde çalışmaz.

`substring()` yöntemi `slice()` yöntemine benzer. Aralarındaki tek fark negatif parametre almamasıdır.

`substr()` yöntemi de `slice()` yöntemine benzer. Burada ikinci parametre ayrılacak parçanın uzunluğunu belirtir. İlk parametre negatif verilirse konum dizgenin sonundan itibaren sayılır. İkinci parametre uzunluk belirtmesi nedeniyle negatif olamaz. İkinci parametre boş bırakılırsa verilen konumdan dizgenin sonuna kadar olan kısmı ayrılır.

Dizge içeriğinin değiştirilmesi için `replace()` yöntemi kullanılır.

```
str = "Please visit Microsoft!";
var n = str.replace("Microsoft","W3Schools");
```

Büyük ve küçük harfe dönüştürme

```
var text1 = "Hello World!";       // Dizge
var text2 = text1.toUpperCase();  // text2 text1'in büyük harfe dönüşmüş hali

var text1 = "Hello World!";       // Dizge
var text2 = text1.toLowerCase();  // text2 text1'in küçük harfe dönüşmüş hali
```

`concat()` **yöntemi** ile iki ya da daha fazla dizge birleştirilebilir. Birleştirme + operatörü ile de yapılabilir.

```
var text1 = "Hello";
var text2 = "World";
text3 = text1.concat(" ",text2);

var text = "Hello" + " " + "World!";
var text = "Hello".concat(" ","World!");
```

Dizge içindeki karakterlere erişmek için iki yöntem vardır. `charAt()` belli bir indisteki (konumdaki) karakteri döndürür. `charCodeAt()` yöntemi ise belli bir indisteki (konumdaki) karakterin unikodunu (tamsayı bir değer) döndürür.

```
var str = "HELLO WORLD";
str.charAt(0);            // geri dönüş değeri H
str.charCodeAt(0);        // geri dönüş değeri 72
```

Dizge elemanlarına dizi gibi (örneğin `str[0]`) erişilebilen kod örneklerine rastlanır. Ancak bu şekilde çalışmak güvenli değildir. Beklenmeyen yan etkileri görülebilir. Tüm web tarayıcılarda

da çalışmaz. Dizge elemanlarına dizi gibi erişmek için dizgeyi diziye dönüştürmek gereklidir. Bu amaçla `split()` yöntemi kullanılır. `split()` yönteminin parametresi ayırt edici karakterdir (virgül, boşluk vb). Bu parametre boş bırakılırsa dizgenin tamamı dizinin sıfırıncı elemanı olur. `""` olursa dizgenin her bir karakteri dizinin bir elemanına atanır.

```
var txt = "a,b,c,d,e";   // Dizge
txt.split(",");          // virgüllere göre ayır
txt.split(" ");          // boşluklara göre ayır
txt.split("|");          // pipe karakterlerinden ayır
```

4.14 JavaScript'te Sayılar

JavaScript'te tek tür sayı vardır. Sayılar ondalık noktalı ya da ondalık noktasız yazılabilirler.

```
var x = 34.00;    // A number with decimals
var y = 34;       // A number without decimals
var x = 123e5;    // 12300000
var y = 123e-5;   // 0.00123
```

Diğer programlama dillerinden farklı olarak JavaScript'te sayılar farklı türlere ayrılmaz. Tüm sayılar double incelikli sayılar olarak saklanırlar. Tam sayılar 15 rakam uzunluğunda olabilir.

Sayısal sabitler `0x` ile başlarsa hekzadesimal olarak kabul edilirler.

```
var x = 0xFF;             // x will be 255
```

Varsayılan olarak rakamlar onluk sistemde görüntülenir. Onaltılık, sekizlik ve ikilik sistemlerde sayıları görüntülemek için `toString()` yöntemi kullanılabilir.

```
var myNumber = 128;
myNumber.toString(16);     // returns 80
myNumber.toString(8);      // returns 200
myNumber.toString(2);      // returns 10000000
```

Bir sayı mümkün olan en büyük sayıyı aştığı zaman değeri `Infinity` ya da `-Infinity` olur. `NaN` ise bir değerin sayı olamayacağını ifade eder (Not a number).

Sayılar değişken olarak tanımlanabilecekleri gibi obje olarak da tanımlanabilirler.

```
var x = 123;
var y = new Number(123);
```

Sayılar değişken olarak tanımlansalar bile JavaScript'te obje gibi kabul edilirler. Bu nedenle özellikleri ve yöntemleri vardır.

Özellik	Açıklama
MAX_VALUE	JavaScript'de kullanılabilir en büyük sayıyı verir.
MIN_VALUE	JavaScript'de kullanılabilir en küçük sayıyı verir.
NEGATIVE_INFINITY	Negatif sonsuz değeri temsil eder.
NaN	"Not-a-Number" değerini temsil eder.
POSITIVE_INFINITY	Sonsuz değeri temsil eder.

Sayı özellikleri `Number` adlı özel bir objeye aittir. `Number.MAX_VALUE` şeklinde kullanılırlar.

Sayı Yöntemleri

Aşağıdaki yöntemler tüm veri türlerinde kullanılabilir.

Yöntem	Açıklama
Number()	Verilen argümanı sayıya dönüştürür.
parseFloat()	Verilen argümanı reel sayıya dönüştürür.
parseInt()	Verilen argümanı tam sayıya dönüştürür.

Sayılarda kullanılan yöntemler:

Yöntem	Açıklama
toString()	Sayıyı dizgeye dönüştürür
toExponential()	Üstel biçimde yazılmış şekilde bir sayıyı dizgeye dönüştürür.
toFixed()	Belli sayıda basamakla yazılmış biçimde bir sayıyı dizgeye dönüştürür.
toPrecision()	Belli bir uzunlukta (basamak sayısı) dizgeye dönüştürme
valueOf()	Sayıyı sayıya dönüştürür.

Örnekler:

```
var x = 123;
x.toString();            // returns 123 from variable x
(123).toString();        // returns 123 from literal 123
(100 + 23).toString();   // returns 123 from expression 100 + 23

var x = 9.656;
x.toExponential(2);     // returns 9.66e+0
x.toExponential(4);     // returns 9.6560e+0
x.toExponential(6);     // returns 9.656000e+0

var x = 9.656;
x.toFixed(0);           // returns 10
x.toFixed(2);           // returns 9.66
x.toFixed(4);           // returns 9.6560
x.toFixed(6);           // returns 9.656000

var x = 9.656;
x.toPrecision();        // returns 9.656
x.toPrecision(2);       // returns 9.7
x.toPrecision(4);       // returns 9.656
x.toPrecision(6);       // returns 9.65600

x = true;
Number(x);        // returns 1
x = false;
Number(x);        // returns 0
x = new Date();
Number(x);        // returns 1404568027739
x = "10"
Number(x);        // returns 10
```

```
x = "10 20"
Number(x);        // returns NaN

parseInt("10");         // returns 10
parseInt("10.33");      // returns 10
parseInt("10 20 30");   // returns 10
parseInt("10 years");   // returns 10
parseInt("years 10");   // returns NaN

parseFloat("10");        // returns 10
parseFloat("10.33");     // returns 10.33
parseFloat("10 20 30");  // returns 10
parseFloat("10 years");  // returns 10
parseFloat("years 10");  // returns NaN

var x = 123;
x.valueOf();            // returns 123 from variable x
(123).valueOf();        // returns 123 from literal 123
(100 + 23).valueOf();   // returns 123 from expression 100 + 23
```

4.15 Operatörler

Aritmetik Operatörler

Aritmetik operatörler toplama, çıkarma, çarpma gibi işlemleri yaparlar. Aşağıdaki örneklerde y=5 iken, örnek deyimin yürütülmesinden sonra operatörlerin ürettiği değerler verilmiştir.

Operatör	Açıklama	Örnek	y	x
`+`	Toplama	`x=y+2`	5	7
`-`	Çıkarma	`x=y-2`	5	3
`*`	Çarpma	`y=x*2`	5	10
`/`	Bölme	`x=y/2`	5	2.5
`%`	Kalan	`x=y%2`	5	1
`++`	Artırma	`x=++y`	6	6
		`x=y++`	6	5
`--`	Eksiltme	`x=--y`	4	4
		`x=y--`	4	5

Artırma ve eksiltme operatörleri bir artırma ya da bir eksiltme işlemi yaparlar. Ön ek ve son ek olmalarına göre etkileri farklıdır (yukarıdaki örnekleri inceleyin). Kalan operatörü (%)ise tam sayı bölmeden kalanı verir.

Aritmetik operatörler sayı türü için kullanılmalarına rağmen `+` operatörü, dizge türü değişkenlerle de kullanılabilir. Bu durumda iki ya da daha fazla dizgeyi arka arkaya ekler.

```
txt1 = "What a very";
txt2 = "nice day";
txt3 = txt1 + " " + txt2;
```

Hem sayı türüne hem de dizge türüne toplama operatörü (+) uygulanabilir.

```
x = 5 + 5;      //x değeri 10
y = "5" + 5;    //y değeri 55
z= "Hello" + 5; //z değeri Hello5
```

+ operatörü tek terimli olarak da kullanılır. Bu kullanımda sayısal olmayan türleri sayıya dönüştürür. Sayıya dönüşüm yapılamazsa NaN değerini üretir.

```
var y = "5";      // y dizge
var x = + y;      // x sayı

var y = "John";   // y dizge
var x = + y;      // x sayı değeri NaN
```

Atama Operatörleri

Atama operatörleri ile değişkenlerin değerleri değiştirilir. Aşağıdaki tabloda verilen örneklerde x=10, y=5 olduğuna göre operatörlerin ürettiği değerler gösterilmiştir.= atama operatörü, diğerleri ise işlemli atama operatörleridir.

Operatör	Örnek	Eşdeğeri	Sonuç
=	x = y	x = y	x = 5
+=	x += y	x = x + y	x = 15
-=	x -= y	x = x - y	x = 5
*=	x *= y	x = x * y	x = 50
/=	x /= y	x = x / y	x = 2
%=	x %= y	x = x % y	x = 0

typeof Operatörü

`typeof` operatörü bir değişken ya da ifadenin türünü verir.

```
var x = 5;
var y = "John";               //  Geri dönüş değeri:
typeof x                      //  number
typeof y                      //  string
typeof 3.14                   //  number
typeof false                  //  boolean
typeof [1,2,3,4]              //  object
typeof {name:'john', age:34}  //  object
```

İlişkisel Operatörler

İlişkisel operatörlere karşılaştırma operatörleri de denir. Değişkenlerin ya da sabitlerin eşit olup olmadıkları vb ilişkileri belirlemek için kullanılırlar. Aşağıdaki tabloda ilişkisel operatörler örneklerle listelenmiştir. Örneklerdeki x değişkeninin değeri 5 olduğuna göre operatörlerin ürettiği sonuçlar da tabloda gösterilmiştir.

Operatör	Tanımı	Örnek	Ürettiği Sonuç
==	Eşittir	x==8	false
		x==5	true
===	Değer ve tür eşitliği	x==="5"	false
		x===5	true
!=	Eşit değil	x!=8	true
!==	Değer ve tür eşit değil	x!=="5"	false
		x!==5	true
>	Büyüktür	x>8	false
<	Küçüktür	x<8	true
>=	Büyük ya da eşit	x>=8	false
<=	Küçük ya da eşit	x<=8	true

İlişkisel operatörler karşılaştırma deyimlerinde (`if`) kullanılırlar.

Mantıksal Operatörler

Mantıksal operatörler değişkenler ve sabitler arasındaki mantıksal ilişkileri belirlemek için kullanılırlar. Tüm programlama dillerinde üç tane mantıksal operatör vardır (ve, veya, değil). Aşağıdaki tabloda mantıksal operatörler örneklerle açıklanmıştır (x=6, y=5 olduğuna göre).

Operatör	Tanımlama	Örnek	Ürettiği Sonuç
&&	Ve	x<10&&y>1	true
||	Veya	x==5||y==5	false
!	Değil	!(x==y)	true

Koşul Operatörü

JavaScript, belli bir koşula bağlı olarak bir değişkene bir değer atayan bir koşul operatörü de içermektedir. Söz dizimi:

```
variablename = (condition) ? value1:value2
```

Koşul sağlanıyorsa değişkene birinci değer, sağlanmıyorsa ikinci değer atanır. Örnek:

```
voteable = (age < 18) ? "Too young":"Old enough";
```

4.16 Math Objesi

Math objesi matematiksel işlemleri yapmak için kullanılır, matematiksel yöntem ve özellikler içerir. Aşağıda Math objesi yöntemleri tablo halinde verilmiştir.

Yöntem	Tanımı
abs(x)	Mutlak değer
acos(x)	Arccos (radyan olarak)
asin(x)	Arcsin (radyan olarak)
atan(x)	Arctan ($\frac{-\pi}{2}$, $\frac{\pi}{2}$aralığında, radyan)
atan2(y,x)	Bir kesirin arctan değeri ($-\pi$,π aralığında, radyan)
ceil(x)	Üst tam sayıya yuvarlatma
cos(x)	$\cos x$ (radyan olarak)
exp(x)	e^x
floor(x)	Alt tamsayıya yuvarlatma
log(x)	$\ln x$
max(x,y,z,...,n)	En büyük değer
min(x,y,z,...,n)	En küçük değer
pow(x,y)	x^y
random()	0,1 aralığında rastgele tamsayı üretme
round(x)	En yakın tamsayıya yuvarlatma
sin(x)	$\sin x$
sqrt(x)	$\sqrt{x}$
tan(x)	$\tan x$

Matematiksel sabitler, Math objesi özellikleri olarak tanımlanmıştır.

Sabit	Açıklama
Math.E	e sayısı
Math.PI	π sayısı
Math.SQRT2	$\sqrt{2}$
Math.SQRT1_2	$\sqrt{\frac{1}{2}}$
Math.LN2	$\ln 2$
Math.LN10	$\ln 10$
Math.LOG2E	$\log_2 e$
Math.LOG10E	$\log e$

4.17 Date Objesi

Date (tarih) objesi zamanla çalışırken (yıl, ay, gün, dakika, saniye, milisaniye) gereklidir. Date objeleri `new Date()` kodu ile tanımlanır. Date objesinin başlangıcı dört şekilde tanımlanabilir.

```
new Date()
new Date(milliseconds)
new Date(dateString)
new Date(yil, ay, gun, saat, dakika, saniye, milisaniye)
```

Parametre kullanılmadığında yeni bir date objesi o anki tarih ve saat ile oluşur. Parametre olarak bir tarih dizgesi (date string) kullanıldığında belirtilen tarih ve saatle obje yaratılır. Bir sayı ile kullanıldığında 01.01.1970 den beri geçen milisaniye sayısı ile obje yaratılır. Parametre sayısı 7 olduğunda yukarıda verilen sıra ile tarih bilgisi objeye atanır. Burada sondan 4 parametre ihmal edilebilir (saat, dakika, saniye, milisaniye).

```
var d = new Date();
document.getElementById("demo").innerHTML = d;

var d = new Date(0);
document.getElementById("demo").innerHTML = d;

var d = new Date(99,5,24,11,33,30,0);
document.getElementById("demo").innerHTML = d;

var d = new Date(99,5,24);
document.getElementById("demo").innerHTML = d;
```

Date Yöntemleri

Date objesi oluşturulduktan sonra bir dizi yöntem kullanılabilir. Bir date objesi HTML'de görüntülendiğinde otomatik olarak `toString()` yöntemi ile bir dizgeye dönüştürülür. Aşağıdaki kodlar eşdeğerdir.

```
document.getElementById("demo").innerHTML = d;
document.getElementById("demo").innerHTML = d.toString();
```

`toUTCString` yöntemi tarihi bir UTC dizgesine dönüştürür (bir tarih görüntüleme standardı). `toDateString()` yöntemi ise daha okunaklı bir formatta tarih görüntüler (saat, dakika saniye görüntülenmez).

```
document.getElementById("demo").innerHTML = d.toUTCString();
document.getElementById("demo").innerHTML = d.toDateString();
```

Tarih alma (date get) yöntemleri tarih ve saatin belli bir parçasını elde etmek için kullanılırlar. Bazıları aşağıda listelenmiştir.

Yöntem	Tanım
getDate()	Günü sayı olarak alma (1-31 arası)
getDay()	Haftanın günü sayı olarak (0-6)
getFullYear()	4 basamaklı yıl (yyyy)
getHours()	Saat (0-23)
getMilliseconds()	Milisaniye (0-999)
getMinutes()	Dakika (0-59)
getMonth()	Ay (0-11)
getSeconds()	Saniye (0-59)
getTime()	Zaman (01.01.1970'den itibaren geçen milisaniye değeri)

```
var d = new Date();
document.getElementById("demo").innerHTML = d.getTime();

document.getElementById("demo").innerHTML = d.getFullYear();

document.getElementById("demo").innerHTML = d.getDay();
```

Tarih ayarlama (date set) yöntemleri tarih ve zamanın bir kısmını ayarlamak için kullanılır. En yaygın kullanılanları aşağıda listelenmiştir.

Yöntem	Tanım
setDate()	Günü sayı olarak ayarlama (1-31 arası)
setFullYear()	Yıl ayarlama (opsiyonel olarak ay ve gün yyyy.mm.dd)
setHours()	Saat ayarlama (0-23)
setMilliseconds()	Milisaniye (0-999)
setMinutes()	Dakika (0-59)
setMonth()	Ay (0-11)
setSeconds()	Saniye (0-59)
setTime()	Zaman ayarlama (01.01.1970'den itibaren milisaniye olarak)

```
var d = new Date();
d.setFullYear(2020, 0, 14);
d.setDate(20);
```

Tarih Ayrıştırma

Bir tarih değerini ya da dizgesini `Date.parse()` yöntemi ile milisaniye değerine dönüştürmek (01.01.1970'den itibaren sayılan milisaniye) mümkündür.

```
var msec = Date.parse("March 21, 2012");
```

Date objeleri ilişkisel operatörler ile kolayca karşılaştırılabilirler. Aşağıdaki örnekte güncel tarih bir başka tarih ile karşılaştırılmaktadır.

```
ey, someday, text;
today = new Date();
someday = new Date();
someday.setFullYear(2020, 0, 14);

if (someday > today) {
    text = "Bugün 14 Ocak 2020'den öncedir.";
} else {
    text = "Bugün 14 Ocak 2020'den sonradır.";
}
document.getElementById("demo").innerHTML = text;
```

4.18 Diziler

Tüm programlama dillerinde aynı türden sıralı olarak bir grup değişken dizilerde saklanır. Diziler, alt değişkenler içeren (elemanlar) özel değişkenler olarak da düşünülebilir. Diziler elemanlarının başlangıç değerleriyle birlikte iki şekilde tanımlanabilir.

```
var cars = ["Saab", "Volvo", "BMW"];
var cars = new Array("Saab", "Volvo", "BMW");
```

Yukarıdaki iki satır tamamen aynıdır. `new Array()` kullanılması gerekli değildir. Bu nedenle ilk yazım biçimi daha okunaklı olduğundan tercih edilir. Dizi elemanlarına ise indis (elemanın dizideki sırası) ile erişilir (Örnek `cars[0]="Ford"`). Dizi indisleri sıfırdan başlar, negatif sayı olamaz.

Diziler objelerin özel bir türüdür. Bir dizi altında yer alan değişkenlerin (dizinin elemanları) türleri farklı olabilir. Bir dizi içinde objeler, fonksiyonlar hatta başka diziler olabilir.

```
myArray[0] = Date.now;
myArray[1] = myFunction;
myArray[2] = myCars;
```

Dizi Özellikleri ve Yöntemleri

JavaScript dizilerinin gücü onların yerleşik özellikleri ve yöntemlerinden kaynaklanır. `length` özelliği dizinin uzunluğunu (dizi eleman sayısı) bulmak için kullanılır (Örneğin `fruits.length`). Dizi indisi sıfırdan başladığı için dizi uzunluğu en büyük indisten bir fazladır. Örneğin dizi uzunluğu 4 ise, indisler 0,1,2,3 şeklindedir.

Dizilere son indis artırılarak elemanlar eklenebilir. Son indisten daha büyük indisle elemanlar eklenirse dizi elemanları içinde tanımlanmamış boşluklar oluşur. Ancak bellek taşması olmaz.

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits[fruits.length] = "Lemon";     // dizi eleman sayısı bir artıyor
```

Dizi elemanlarına sıralı erişimde döngüler kullanılır. Aşağıdaki örnekte `for` döngüsü kullanılmıştır.

```
var text;
var index;
var fruits = ["Banana", "Orange", "Apple", "Mango"];
for (index = 0; index < fruits.length; index++) {
    text += fruits[index];
}
```

Dizileri Dizgeye Dönüştürme

JavaScript'de tüm objeler (diziler de objedir) `valueOf()` ve `toString()` yöntemlerine sahiptirler. Her iki yöntem de diziyi (elemanlarını sıra ile arka arkaya ekleyerek) dizgeye dönüştürür.

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.getElementById("demo").innerHTML = fruits.valueOf();

document.getElementById("demo").innerHTML = fruits.toString();
```

`join()` yöntemi de bir diziyi elemanlarını arka arkaya ekleyerek dizgeye dönüştürür. Burada ayraç belirtmek mümkündür (Örnek: `fruits.join(" * ");`).

`pop()` yöntemi bir dizideki son elemanı siler. `push()` ise dizinin sonuna yeni bir eleman ekler. `pop()` yöntemi "popped out" şeklinde bir dizge, `push()` yöntemi ise dizinin eklem yapıldıktan sonraki uzunluğunu geri döndürür.

`shift()` yöntemi dizideki ilk elemanı siler, diğer elemanları başa doğru öteler. `unshift()` yöntemi ise dizinin başına bir eleman ekler diğerlerini sona doğru öteler. `shift()` yöntemi "shifted out" şeklinde bir dizge, `unshift()` yöntemi ise dizinin ekleme yapıldıktan sonraki uzunluğunu geri döndürür.

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.shift();            //fruits dizisinde ilk eleman "Banana" silinir.

var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits[0] = "Kiwi";        // ilk elemanı "Kiwi" olarak değiştirir.
```

Diziden eleman silmek için ise `delete` operatörü kullanılır. Örneğin yukarıdaki kodlarda tanımlı dizide silme yapılırsa (`delete fruits[0]`) ilk elemanın değeri `undefined` olur.

`splice()` yöntemi bir diziye yeni elemanlar eklemek için kullanılır. İlk parametre yeni elemanların ekleneceği konumu, ikinci parametre kaç tane elemanın silineceğini belirler. Sonraki parametreler eklenecek elemanların değerleridir.

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.splice(2, 0, "Lemon", "Kiwi");
```

`splice()` ile dizide boşluklar oluşturmadan elemanlar silmek mümkündür.

```
fruits.splice(0,1);        // İlk elemanı siler
```

Sıralama

`sort()` yöntemi diziyi alfabetik olarak sıralar (Örnek: `fruits.sort();`). `reverse()` yöntemi ise bir dizide sıralamayı tersine döndürür. Önce `sort()` sonra `reverse()` kullanılırsa dizi sondan başa (Z den A ya) sıralanmış olur.

`sort()` yöntemi alfabetik sıralama yaptığı için sayısal dizilerde kullanılamaz (alfabetik sıralamada 25 100 den büyüktür). Bunu çözmek için `sort()` yöntemine parametre olarak bir fonksiyon tanımlanabilir. Bu fonksiyon -1,0,1 değerlerini üretir.

```
var points = [40, 100, 1, 5, 25, 10];
points.sort(function(a, b){return a-b}); //Küçükten büyüğe sıralama
points.sort(function(a, b){return b-a}); //Büyükten küçüğe sıralama
```

`concat()` yöntemi ile iki dizi birbirine eklenerek yeni bir dizi oluşturulabilir. Parametresi birden çok dizi olabilir.

```
var myGirls = ["Cecilie", "Lone"];
var myBoys = ["Emil", "Tobias","Linus"];
var myChildren = myGirls.concat(myBoys); //myGirls ile myBoys birleştiriliyor.
```

`slice()` yöntemi bir dizinin bir parçasını ayırır.

```
var fruits = ["Banana", "Orange", "Lemon", "Apple", "Mango"];
var citrus = fruits.slice(1,3); //citrus dizisi elemanları Orange ve Lemon
```

4.19 Mantıksal Değerler

Programlamada iki değer alabilen bir değişken türüne ihtiyaç vardır (Örnek: yes/no, on/off, true/false). Bu amaçla JavaScript'de `Boolean` veri türü vardır. Bu türde yalnızca `true false` değerleri kullanılabilir.

Boolean() Fonksiyonu

Bir ifadenin doğru olup olmadığını belirlemek için `Boolean()` fonksiyonu kullanılabilir. Örneğin `Booelan(10>9)` için geri dönüş değeri `true` olur. `(10>9)` parantezinin sonucu da `true` dur. İfadelerin ve sabitlerin mantıksal olarak yorumlanması `C` diline benzer. Sıfır dışındaki tüm reel sayılar (negatif ya da pozitif) doğru (`true`) olarak yorumlanır. Sıfır, `undefined, null, NaN, ""` (boş dizge) yanlış (`false`) olarak yorumlanır.

4.20 Tür Dönüşümleri

JavaScript'de değer içerebilen 5 farklı veri türü vardır:

- dizge
- sayı
- boolean
- obje
- fonksiyon

Üç tür obje vardır.

- Obje
- Date (Tarih)
- Dizi (Array)

Değer içermeyen iki tür vardır.

- `null`
- `undefined`

Bir değişkenin değerinin türünü bulmak için `typeof` operatörü kullanılabilir. Örnekler:

```
typeof "John"                // geri dönüş: string
typeof 3.14                  // geri dönüş: number
typeof NaN                   // geri dönüş: number
typeof false                 // geri dönüş: boolean
typeof [1,2,3,4]             // geri dönüş: object
typeof {name:'John', age:34} // geri dönüş: object
typeof new Date()            // geri dönüş: object
typeof function () {}        // geri dönüş: function
typeof myCar                 // geri dönüş: undefined (myCar tanımlanmadı ise)
typeof null                  // geri dönüş: object
```

`constructor` özelliği ile tüm değişkenler için yaratıcı fonksiyon adı (constructor function) geri döndürülebilir. Bu şekilde bir değişkenin dizi olup olmadığı test edilebilir.

JavaScript değişkenlerinin türleri değiştirilebilir. Tür dönüşümleri ya bir JavaScript fonksiyonu ile ya da otomatik olarak yapılır. Sayı yöntemlerinden `toString()` tür dönüşümü yapan fonksiyonlardan biridir. Benzer sayı-dizge dönüşümü yapan sayı yöntemlerine `toExponential(), toFixed()` örnek olarak verilebilir.

Global `String()` yöntemi boolean türünü dizge türüne dönüştürür. Örneğin `String(false)` "false"değerini döndürür. Boolean yöntemlerinden `toString()` aynı işleve sahiptir.

Date türünden dizge türüne dönüşümde de `toString()` yöntemi kullanılabilir. Date dizge dönüşümü yapan başka yöntemler de vardır.

Global `Number()` yöntemi dizge sayı dönüşümünü yapar. Sayı içeren dizgeler içerdikleri sayıya, boş dizgeler (`""`)sıfıra, bunun dışındaki kalan her tür dizge NaN'a dönüştürülür.

```
Number("3.14")    // geri dönüş:  3.14
Number(" ")       // geri dönüş:  0
Number("")        // geri dönüş:  0
Number("99 88")   // geri dönüş:  NaN
```

Sayı yöntemleri içerisinde dizge (string) sayı dönüşümü yapan başka yöntemler de vardır (`parseFloat(), parseInt()` gibi).

`Number()` yöntemi boolean türünü sayı türüne dönüştürebilir. `false` 0, `true` 1 sayılarına dönüştürülür. `Number()` ile tarih sayı dönüşümü de yapılabilir.

Otomatik Tür Dönüşümleri

Tür uyuşmazlığı ortaya çıktığında otomatik tür dönüşümü yapılarak uyuşumsuzluk giderilmeye çalışılır. Otomatik tür dönüşümleri her zaman beklenildiği gibi sonuç vermeyebilir.

```
5 + null    // sonuç: 5         null 0'a dönüştüğü için
"5" + null  // sonuç: "5null"   null "null" a dönüştüğü için
"5" + 1     // sonuç: "51"      1  "1" e dönüştüğü için
"5" - 1     // sonuç: 4         "5" 5 e dönüştüğü için
```

4.21 Koşul Deyimleri

Koşul deyimleri farklı koşullarda farklı eylemleri gerçekleştirebilmek için gerekli olup, tüm programlama dillerinde vardır. JavaScript'de koşul deyimleri:

- Belli bir koşula bağlı olarak bir kod bloğu icra edilecek ise **if**
- Koşul sağlanmadığında da bir kod bloğu icra edilecek ise **else**
- Koşul sağlanmadığında yeni bir koşul test edilecek ise **else if**
- Bir koşula bağlı olarak çok sayıda kod bloğu seçeneği var ise **switch**

If Deyimi

Bir koşul oluştuğunda bir kod bloğu icra edilecek ise if deyimi kullanılır. Söz dizimi:

```
if (koşul ifadesi) {
    icra edilecek kod bloğu
}
```

if küçük harflerle yazılmalıdır. Büyük/küçük harf eşleştirilmesi yapılmaz! Koşul ifadesinde genellikle ilişkisel ve mantıksal operatörler kullanılır.

```
if (time < 20) {
    greeting = "iyi günler!";
}
```

Koşul ifadesi doğru olmadığında icra edilecek kod bloğu var ise bu durumda else kullanılır. Söz dizimi:

```
if (koşul ifadesi) {
    koşul doğru ise icra edilecek kod bloğu
}else{
    koşul yanlış ise icra edilecek kod bloğu
}
```

```
if (time < 20) {
    greeting = "iyi günler!";
} else {
    greeting = "iyi akşamlar";
}
```

İlk koşul ifadesi yanlış değer ürettiğinde yeni bir koşula daha bakılması gerekiyorsa else if kullanılır. Söz dizimi ve örnek kod:

```
if (koşul1) {
    koşul 1 doğru ise icra edilecek kod
} else if (koşul2) {
    koşul 2 doğru ise icra edilecek kod
} else {
    iki koşul da yanlış ise icra edilecek kod
}
```

```
if (time < 10) {
    greeting = "Günaydın!";
} else if (time < 20) {
    greeting = "İyi Günler!";
} else {
    greeting = "İyi Akşamlar!";
}
```

Switch Deyimi

Çok sayıda seçenek var ise switch deyimi kullanmak yararlıdır. Söz dizimi:

```
switch(ifade) {
    case n:
```

```
        kod bloğu
        break;
    case n:
        kod bloğu
        break;
    default:
        varsayılan kod bloğu
}
```

`switch` ifadesinin değeri belirlenir. `case` ifadeleri ile karşılaştırılır. Eşleşme durumunda ilgili kod bloğu ve sonrası çalıştırılır.

```
switch (new Date().getDay()) {
    case 0:
        day = "Sunday";
        break;
    case 1:
        day = "Monday";
        break;
    case 2:
        day = "Tuesday";
        break;
    case 3:
        day = "Wednesday";
        break;
    case 4:
        day = "Thursday";
        break;
    case 5:
        day = "Friday";
        break;
    case 6:
        day = "Saturday";
        break;
}
```

JavaScript yorumlayıcısı `break` anahtar kelimesine ulaştığı zaman `switch` bloğu dışına çıkar. Bu şekilde daha fazla test yapılmaz. `default` anahtar kelimesi ile hiç bir `case` ifadesi eşleşmediğinde yürütülecek kod belirtilir. Kullanılması zorunlu değildir. En sonda yer alması da gerekmez. Bazen birden fazla `case` ifadesinin aynı kodu paylaşması da gerekebilir. Aşağıdaki örnekleri inceleyiniz.

```
switch (new Date().getDay()) {
    case 6:
        text = "Today is Saturday";
        break;
    case 0:
        text = "Today is Sunday";
        break;
    default:
        text = "Looking forward to the Weekend";
}
```

```
switch (new Date().getDay()) {
    case 1:
```

```
    case 2:
    case 3:
    default:
        text = "Looking forward to the Weekend";
        break;
    case 4:
    case 5:
       text = "Soon it is Weekend";
        break;
    case 0:
    case 6:
       text = "It is Weekend";
}
```

4.22 Döngüler

Bir kod ya da kod bloğu birden fazla icra edilecek ise döngü kullanılması gerekir. JavaScript'de desteklenen çeşitli döngü türleri vardır.

- **for**: Bir kod bloğu belli sayıda icra edilir.
- **for/in**: Bir objenin özelliklerine bağlı döngü
- **while**: Belli bir koşula bağlı döngü
- **do/while** Belli bir koşula bağlı döngü

For Döngüsü

Sık kullanılan bir döngü türüdür. Söz dizimi:

```
for (ifade 1; ifade 2; ifade 3) {
    icra edilecek kod
}
```

- ifade 1 döngü başlamadan önce bir kez icra edilir.
- ifade 2 döngü koşulunu belirler. Döngünün başında ve her turun sonunda icra edilir.
- ifade 3 döngünün her turunun sonunda icra edilir.

```
for (i = 0; i < 5; i++) {
    text += "The number is " + i + "<br>";
}
```

Yukarıdaki örnekte ifade 1'de bir değişkene (i) döngü başlamadan bir değer atanıyor (i=0). Buradaki değişkene sayaç değişkeni de denir. İfade 2'de döngünün dönme koşulu belirleniyor (i 5'ten küçük olmalı). İfade 3, i değişkeninin değerini döngünün her turundan sonra bir artırmaktadır.

Genel olarak ifade 1 ile döngünün sayaç değişkeninin başlangıç değeri atanır. Ancak bu her zaman böyle olmak zorunda değildir. Boş bırakılabilir. Virgülle ayrılarak birden çok ifade burada yer alabilir.

İfade 2 ile döngünün dönme koşulu belirlenir. Kullanılması zorunlu değildir, boş bırakılabilir. Ancak bu durumda sonsuz döngü oluşacağından döngü içinde bir koşula bağlı olarak `break` kullanılarak döngünün sona ermesi sağlanmalıdır.

İfade 3 sayaç değişkeninin değerini artırmak ya da eksiltmek için kullanılır. Artma ve eksiltme birer birer olabileceği gibi (`i++ i--`) daha farklı da olabilir (örneğin `i=i+10`). İfade 3, sayacın değeri döngü içinde değişiyor ise boş bırakılabilir.

Her üç ifade de boş bırakılabilir. Ancak for parantezi içinde iki adet noktalı virgül bulunmak zorundadır (`for(;;)`).

```
for (i = 0, len = cars.length, text = ""; i < len; i++) {
    text += cars[i] + "<br>";
}

var i = 2;
var len = cars.length;
var text = "";
for (; i < len; i++) {
    text += cars[i] + "<br>";
}

var i = 0;
len = cars.length;
for (; i < len; ) {
    text += cars[i] + "<br>";
    i++;
}
```

For/In Döngüsü

For/In deyimi bir objenin özellikleri ile döngü oluşturmak için kullanılır.

```
var person = {fname:"John", lname:"Doe", age:25};

var text = "";
var x;
for (x in person) {
    text += person[x]+" ";
}
```

Döngü sona erdiğinde text değişkeninin değeri "John Doe 25 " olur.

While Döngüsü

While döngüsünde belli bir koşul sağlandıkça döngü devam eder. Söz dizimi:

```
while (koşul) {
    çalıştırılacak kod bloğu
}
```

Örnek:

```
while (i < 10) {
    text += "The number is " + i;
    i++;
}
```

Koşulun bağlı olduğu değişkenin değerinin döngü içinde değiştirilmesi gerekir. Aksi halde sonsuz döngü oluşur.

Do/While Döngüsü

Do/While döngüsüde belli bir koşula bağlı olarak çalışır. While döngüsünden farkı koşulun sağlanıp sağlanmadığının döngünün bir tur çalışmasından sonra test edilmesidir. Bu bağlamda döngü kod bloğu koşula bakılmaksızın bir kez çalıştırılır, sonra koşul kontrol edilir. Bu nedenle While döngüsüne üstten kontrollü döngü, Do/While döngüsüne ise alttan kontrollü döngü denir. Söz dizimi:

```
do {
    çalıştırılacak kod bloğu
}
while (koşul);
```

Örnek:

```
do {
    text += "The number is " + i;
    i++;
}
while (i < 10);
```

Break Deyimi

`break` deyimi bir döngünün kesilip (döngü koşuluna bakılmaksızın) döngüden sonraki adıma geçilmesini sağlar. Önceki bölümlerde `switch` deyiminde seçeneklerden çıkılmasını sağlamak için kullanıldığını gördük.

```
for (i = 0; i < 10; i++) {
    if (i == 3) { break }
    text += "The number is " + i + "<br>";
}
```

Continue Deyimi

`continue` deyimi döngünün bir turunu (bir adımını) keserek bir sonraki tura geçilmesini sağlar. Döngüyü kesmez.

```
for (i = 0; i <= 10; i++) {
    if (i == 3) continue;
    text += "The number is " + i + "<br>";
}
```

JavaScript'de Etiketler

`switch` deyiminde satırların etiketlenebileceğini gördük. Etiketler (label) etiket adı ve iki nokta ile tanımlanır.

```
label:
deyimler
```

```
break labelname;
continue labelname;
```

- `continue` deyimi yalnızca döngülerin içinde kullanılabilir.
- `break` deyimi etiket belirtilmeden yalnızca döngülerin ve `switch` deyiminin içinde, etiket belirtilerek her yerde kullanılabilir.

```
cars = ["BMW", "Volvo", "Saab", "Ford"];
list: {
    text += cars[0] + "<br>";
    text += cars[1] + "<br>";
    text += cars[2] + "<br>";
    text += cars[3] + "<br>";
    break list;
    text += cars[4] + "<br>";
    text += cars[5] + "<br>";
}
```

Bölüm 5

HTML Grafiği

HTML sayfalarında raster formatlı dosyalar yanında vektör çizimler de kullanılabilir. Bu amaçla iki HTML elemanı kullanılır, `<svg>` ve `<canvas>`. Bu bölümde bu iki eleman ele alınacaktır.

5.1 SVG

SVG açık yazımı, Scalable Vector Graphics olup, HTML'in tarihsel gelişiminde önemli aşamalardan biridir. SVG dosyaları çeşitli yazılımlardan oluşturulabilir. Örneğin vektör grafik yazılımı Inkscape dosyaları bu formatta kaydeder. CBS yazılımlarında çıktı pencereleri SVG formatında kaydedilebilir.

svg dosyaları `<img>` elemanı ile görüntülenebilir. Svg vektör formatında olduğunu için görüntünün büyütülmesi çözünürlüğü etkilemez. Baskı alınması durumunda baskı kalitesi yüksektir.

```
<img src="test.svg">
```

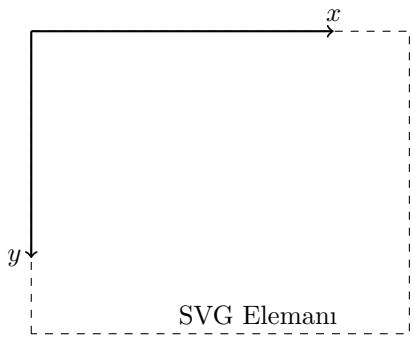
`<svg>` elemanı ile doğrudan HTML kodu içinde çizim yapılabilir. Eleman içeriği XML formatındadır. Aşağıdaki örnekte bir daire çizimi gerçekleştirilmektedir.

```
<html>
<body>

<h1>SVG Örneği</h1>

<svg width="100" height="100">
  <circle cx="50" cy="50" r="40" stroke="red" stroke-width="4" fill="yellow" />
</svg>

</body>
</html>
```

Çizelge 5.1: SVG Şekil Elemanları

Eleman	Kodu
Dikdörtgen	`<rect>`
Daire	`<circle>`
Elips	`<ellipse>`
Doğru	`<line>`
Çokludoğru	`<polyline>`
Çokgen	`<polygon>`
Çizgi	`<path>`

SVG koordinat sistemi bilgisayar grafiğindeki çoğu uygulama gibidir. Orijin SVG elemanının sol üst köşesinde, x sağa y aşağı doğru artar (Şekil 5.1). SVG içindeki şekil elemanlarının koordinatları SVG boyutlarını taşmamalıdır.



Şekil 5.1: SVG koordinat sistemi

SVG'de önceden tanımlanmış şekil elemanları vardır (Çizelge 5.1). Bunların özellikleri aşağıda incelenecektir.

5.1.1 SVG: Dikdörtgen

Dikdörtgen elemanının bir örnek üzerinde inceleyelim.

```
<svg width="400" height="160">
  <rect x="50" y="20" width="300" height="150"
  style="fill:rgb(255,0,0);stroke-width:3;stroke:rgb(0,0,0)" />
</svg>
```

- width ve height öznitelikleri dikdörtgenin genişliğini ve yüksekliğini belirler.
- x,y öznitelikleri elemanın sol üst köşesinin koordinatlarıdır. Diğer bir deyişle x dikdörtgenin soldan y üstten uzaklığını belirler. Bu koordinatlar `<svg>` elemanının içinde geçerlidir.
- style özniteliği dikdörtgenin CSS özelliklerini belirlemek için kullanılmıştır.
- fill özelliği iç dolgu rengini, stroke-width dış çizgi kalınlığını, stroke ise dış çizgi rengini belirlemektedir.

5.1.2 SVG: Daire ve Elips

`circle` elemanı ile daire oluşturulur. `cx, cy` daire merkezinin koordinatlarını, `r` yarıçapı tanımlar. CSS özellikleri dikdörtgen ile aynıdır.

```
<svg height="100" width="100">
  <circle cx="50" cy="50" r="40" stroke="black" stroke-width="3" fill="red" />
</svg>
```

Elips daire ile benzerdir. x ve y yönlerinde `rx,ry` öznitelikleri ile tanımlanan farklı yarıçaplara sahiptir. Aşağıdaki örneği inceleyin.

```
<svg height="140" width="500">
  <ellipse cx="200" cy="80" rx="100" ry="50"
  style="fill:yellow;stroke:purple;stroke-width:2" />
</svg>
```

5.1.3 SVG: Doğru, Çokludoğru ve Çokgen

`line` elemanı ile iki noktayı birleştiren doğru parçası çizilir. `x1,y1,x2,y2` başlangıç ve bitiş noktalarının koordinatlarıdır.

```
<svg height="210" width="500">
  <line x1="0" y1="0" x2="200" y2="200" style="stroke:rgb(255,0,0);stroke-width:2" />
</svg>
```

`polyline` elemanı ile çokludoğru çizilir. Çoklu doğruyu oluşturan noktaların koordinatları points özniteliğinde x,y koordinat çiftleri olarak tanımlanır. Koordinat çiftleri virgül ile ayrılır. Koordinat çiftleri ise boşlukla ayrılır.

```
<svg height="200" width="500">
  <polyline points="20,20 40,25 60,40 80,120 120,140 200,180"
  style="fill:none;stroke:black;stroke-width:3" />
</svg>
```

`polygon` elemanı ise çokgen tanımlamak için kullanılır. En az köşeli çokgen üçgen olduğuna göre çokgeni oluşturan x,y koordinat çifti en az 3 olmalıdır. Çokgeni oluşturan nokta koordinat tanımlaması `polyline` ile aynıdır. `polygon` elemanında son nokta ilk noktaya birleştirilerek çokgen oluşturulur.

```
<svg height="250" width="500">
  <polygon points="220,10 300,210 170,250 123,234"
  style="fill:beige;stroke:red;stroke-width:1" />
</svg>
```

5.1.4 SVG: Çizgi (Path)

`path` elemanı ile doğru ve eğri parçaları içeren çizgiler oluşturulur. Koordinatların önüne yazılan komutlarla çalışır. Komutlar büyük harfle yazılır ise mutlak koordinat, küçük harfle yazılırsa göreceli koordinat verilmiş demektir. Komutlar ve koordinatlar d özniteliği içinde tanımlanır. Aşağıdaki kod ile bir üçgen çizimi gerçekleşmektedir.

Çizelge 5.2: Path Komutları

Komut	İşlem	Anlam
M	moveto	Koordinata git
L	lineto	Doğru
H	horizontal lineto	Yatay doğru
V	vertical lineto	Düşey doğru
C	curveto	Eğri
S	smooth curveto	Yumuşak eğri
Q	quadratic Bézier curve	Bézier eğrisi
T	smooth quadratic Bézier curveto	Yumuşak Bézier eğrisi
A	elliptical Arc	Elips yayı
Z	closepath	Çizgiyi kapat

```
<svg height="210" width="400">
  <path d="M150 0 L75 200 L225 200 Z" />
</svg>
```

Buradaki çizim aşağıdaki gibi ifade edilebilir.

1. 150,0 koordinatına git.
2. Buradan 75,200 koordinatına doğru çiz.
3. Buradan 225,200 koordinatına doğru çiz.
4. Çizgiyi kapat.

`path` elemanının kullanımı öncekilere göre daha karmaşıktır. Burada daha fazla ayrıntıya girilmeyecektir. Daha fazla bilgi gereksinimi olan okuyucuların `https://www.w3schools.com/graphics/default.asp` vb. İnternet kaynaklarına ya da kitaplara başvurmaları gerekmektedir.

5.1.5 SVG: Yazı

HTML ortamında çizim yaparken, çizimlerde yazı kullanılması kaçınılmazdır. Bu amaçla `text` elemanı bulunmaktadır. İstenilen koordinattan başlanarak yatay ya da açılı yazı yazdırılabilir.

```
<svg height="60" width="200">
  <text x="0" y="15" fill="red" transform="rotate(30 20,40)">I love SVG</text>
</svg>
```

Örnekte transform özniteliği olmazsa yazı yatay yazılır.

5.2 Canvas

`<canvas>` elemanı ile JavaScript kullanılarak HTML dokümanında "on the fly" çizim yapılır. `<canvas>` elemanı yalnızca grafik taşıyıcısıdır. Çizim skript ile gerçekleştirilir.

```
<canvas id="myCanvas" width="200" height="100"></canvas>
```

id özniteliği boş olamaz. id ile JavaScript ortamından `<canvas>` elemanına erişilir. Canvas boyutlarının belirlenmesi de zorunludur. Bir HTML sayfası birden çok `<canvas>` içerebilir. Varsayılan değerlerle `<canvas>` elemanının içeriği olmaz, sınırları görülmez. CSS özellikleri ile görünümü değiştirilebilir.

```
<canvas id="myCanvas" width="200" height="100" style="border:1px solid #000000;">
</canvas>
```

Bu elemana erişip üzerinde çizim yapalım.

```
<script>
var canvas = document.getElementById("myCanvas");
var ctx = canvas.getContext("2d");
ctx.fillStyle = "#FF0000";
ctx.fillRect(0, 0, 150, 75);
</script>
```

Buradaki işlemler:

1. `<canvas>` elemanına id özniteliğinden erişilip bir değişkene atanır.
2. Çizim objesi yaratılır. `getContext()` yerleşik HTML çizim objesidir.
3. Çizim elemanı dolgu sitili belirlenir.
4. İçi dolu dikdörtgen çizimi yapılır.

`<canvas>` koordinat sistemi de `<svg>`'de olduğu gibidir (Şekil 5.1).

Bir doğru parçası çizmek için `moveTo` yöntemiyle bir koordinata gidilir. Oradan başka bir koordinata `lineTo` ile doğru parçası çizilir. Çizginin görüntülenmesi için `stroke` çağrılır.

```
var canvas = document.getElementById("myCanvas");
var ctx = canvas.getContext("2d");
ctx.moveTo(0, 0);
ctx.lineTo(200, 100);
ctx.stroke();
```

`arc` yöntemi ile daire çizimi yapılır.

```
var canvas = document.getElementById("myCanvas");
var ctx = canvas.getContext("2d");
ctx.beginPath();
ctx.arc(95, 50, 40, 0, 2 * Math.PI);
ctx.stroke();
```

1. `beginPath` ile çizgiye başlanır.
2. `arc(x,y,r,baş. açı, bit. açı)` ile daire çizilir. `x,y` yay merkezi, `r` yay yarıçapıdır. Açı birimi radyandır. Burada $0 - 2\pi$ aralığında yay çizilerek daire elde edilmiştir.
3. Çizginin görüntülenmesi için `stroke` çağrılır.

Bölüm 6

Leaflet JavaScript Kütüphanesi

Laeflet çevrimiçi harita yayınlamak amaçlı açık kaynak kodlu bir JavaScript kütüphanesi ya da bir Javascript API'sidir. Google Maps vb. ticari sistemlere alternatif açık kaynak kodlu çözümler için en çok tanınanları arasındadır. Yalnızca 38 KB hacminde bir JavaScript kodu ile benzer sistemlere eşdeğer işlevleri yerine getirmektedir.

Leaflet harita verileri sağlamaz. Harita verileri Google gibi ticari sistemlerden ya da OSM (Open Street Map) gibi açık kaynak kodlu sistemlerden alınabilir. Leaflet dokümanlarında genel olarak MapBox API (`http://www.mapbox.com`) ile Open Street Map kullanılan örnekler verilmiştir. Leaflet harita verisi sağlamadığından uygun bir API ile birlikte kullanılması gerekmektedir. Mapbox ile çalışmak için üye olunup access token (giriş kodu) almak gerekmektedir. Sisteme üye olunduğu zaman tasarım aşamasında kullanılabilecek default public token kullanıcılara gösterilir. access token sistemin kullanıcıları tanımasına yönelik bir metindir. Anlamlı kelimelerden oluşmaz.

Bu bölümde verilen bazı örnekler MapBox ile verilmiş olup, okuyucuların access token almaları ve örnek kodlarda ***** olarak gösterilen yerlere kendilerine ait metni yazmaları gerekmektedir. Bu metin elle yazılması zahmetli olduğundan genellikle kopyalanıp yapıştırılır. Bazı örnekler ise OpenStreetMap API ile verilmiştir.

6.1 Leaflet: Başlangıç

Leaflet kullanılacak HMTL sayfasının `head` bölümüne leaflet kütüphane kodlarına erişimi sağlayacak iki eleman yazılmalıdır. Bunlardan ilki Leaflet CSS dosyasına verilecek bağlantıdır ve ilk olarak bu yazılmalıdır (Leaflet, 2018).

```
<link rel="stylesheet" href="https://unpkg.com/leaflet@1.3.4/dist/leaflet.css"
integrity="sha512-puBpdR0798OZvTTbP4A8Ix/l+A4dHDD0DGqYW6RQ+9jxkRFclaxxQb/
SJAWZfWAkuyeQUytO7+7N4QKrDh+drA=="
crossorigin=""/>
```

Daha sonra kütüphane kodları için `script` elemanı oluşturulur. CSS linki skript tanımlamasından

önce olmak zorundadır.

```
<script src="https://unpkg.com/leaflet@1.3.4/dist/leaflet.js"
integrity="sha512-nMMmRyTVoLYqjP9hrbed9S+
FzjZHW5gY1TWCHA5ckwXZBadntCNs8kEqAWdrb9O7rxbCaA4lKTIWjDXZxflOcA=="
crossorigin=""></script>
```

Yukarıda tek satıra sığmayan integrity dizgesi kodlamada tek satırda yazılmalıdır. Bu kodların elle yazımı zahmetli olduğundan Leaflet (2018) adresinden kopyalanıp yapıştırılabilir. Leaflet kaynak kodu indirilebilir olduğundan web sunucuya indirilerek de kullanılabilir. Bu durumda skript kaynağı yerel bir klasör olacaktır.

Tüm harita API'lerinde olduğu gibi burada da harita HTML `div` elemanı ile ilişkilendirilir. Leaflet kodları `div` elemanından sonra olmak zorunda olduğundan Leaflet kodları `body` bölümünün (elemanının) son kısmında yer alır. Yukarıdaki başlangıç tanımlamaları yapılan dosyada id'si "mapid" olan bir div elemanı boyutları da belirtilerek (600×400 piksel) aşağıdaki gibi tanımlanabilir. `*****` ile belirtilen yere MapBox sitesine üye olunarak alınan access token yazılmalıdır.

```
<div id="mapid" style="width: 600px; height: 400px;"></div>
```

Leaflet harita objesinin en az bir katmanı (Layer) olmalıdır. Katman MapBox.street olabilir. Bu amaçla access token alınmış olması gerekir. Aşağıdaki kod `body` içinde yer alan `script` elemanı içinde yer almalıdır.

```
var mymap = L.map('mapid').setView([37.8,32.4], 10);
L.tileLayer('https://api.tiles.mapbox.com/v4/{id}/{z}/{x}/{y}.png?access_token=*****',
{
  id: 'mapbox.streets',
  accessToken: '*****'
}).addTo(mymap);
```

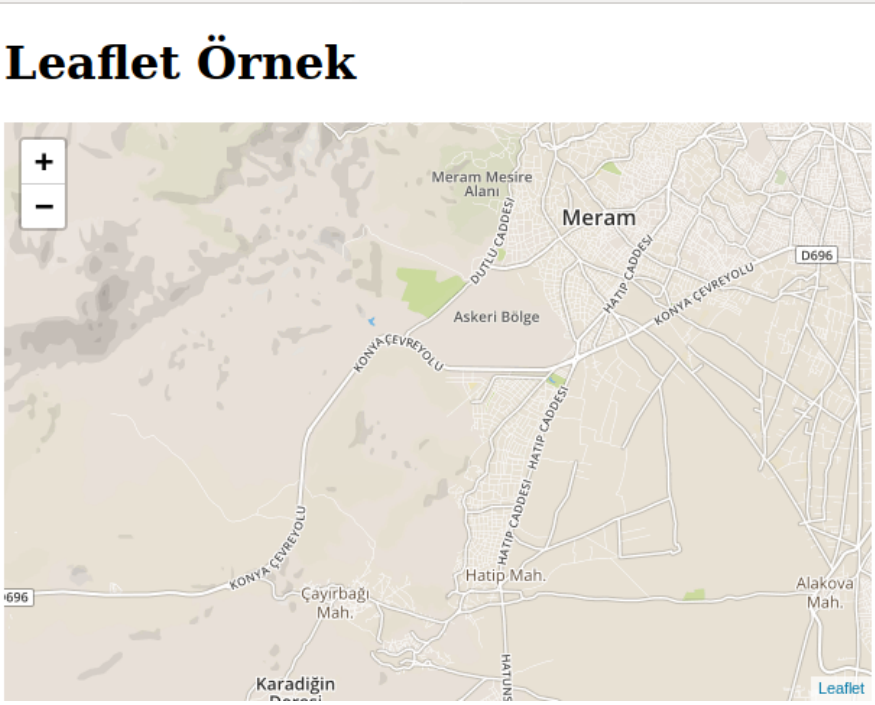
Yukarıdaki kod MapBox yol haritası (streets) katmanını $\varphi = 37.8°, \lambda = 32.4°$ coğrafi koordinatı merkez olmak üzere zoom düzeyi 10 olmak üzere görüntüler. Görüldüğü üzere nokta koordinatları `[enlem,boylam]` şeklinde yazılır. Burada sabit koordinat girilmiştir. Enlem boylam objesi biçiminde değişken olarak da tanımlanabilir. Zoom düzeyi 1 dünya haritası görünümü, 18 en büyük ölçekli görünümüdür. Bu kod ile harita objesi `mymap` olarak adlandırılmıştır. Kod maksimum zoom düzeyi ve harita sağ alt köşesinde yer alacak bilgilendirme ile genişletilerek aşağıdaki gibi olabilir.

```
var mymap = L.map('mapid').setView([37.8,32.4], 10);
L.tileLayer('https://api.tiles.mapbox.com/v4/{id}/{z}/{x}/{y}.png?access_token=*****',
{
  id: 'mapbox.streets',
  accessToken: '*****'
}).addTo(mymap);
```

Bu kodlarla oluşan sayfa görünümü Şekil 6.1'deki gibidir.

`tileLayer` Open Street Map (OSM) olarak da kullanılabilir. OSM ticari bir veri sağlayıcı olmadığı için herhangi bir üyelik ve giriş kodu gerekli değildir. Aşağıdaki kod yukarıda tanımlanan `mymap` objesi ile OSM haritasını ilişkilendirmektedir. Minimum ve maksimum zoom düzeyleri 2 ile 18 arasında ayarlanmış, `attribution` ile harita sağ alt köşesindeki açıklamalar tanımlanmıştır.

```
var osmUrl='https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png';
```



Şekil 6.1: Mapbox ile Leaflet sayfa görünümü

```
var osmAttrib='Map data © <a href="https://openstreetmap.org">OpenStreetMap</a>';
var osm = new L.TileLayer(osmUrl, {minZoom: 2, maxZoom: 18, attribution: osmAttrib})
.addTo(mymap);
```

Bu aşamada sayfanın tamamının kodu aşağıdaki gibidir.

```
<!DOCTYPE html>
<html>
<head>
<title>Leaflet Örnek</title>
<meta charset="UTF-8">
<link rel="stylesheet" href="https://unpkg.com/leaflet@1.3.4/dist/leaflet.css"
   integrity="sha512-puBpdR0798OZvTTbP4A8Ix/l+A4dHDD0DGqYW6RQ+9jxkRFclaxxQb/
SJAWZfWAkuyeQUytO7+7N4QKrDh+drA=="
   crossorigin=""/>
<script src="https://unpkg.com/leaflet@1.3.4/dist/leaflet.js"
   integrity="sha512-nMMmRyTVoLYqjP9hrbed9S+
FzjZHW5gY1TWCHA5ckwXZBadntCNs8kEqAWdrb9O7rxbCaA4lKTIWjDXZxflOcA=="
   crossorigin=""></script>
</head>
<body>
<h1>Leaflet Örnek</h1>
<div id="mapid" style="width: 600px; height: 400px;"></div>
<script>
var mymap = L.map('mapid').setView([37.8,32.4], 12);
var osmUrl='https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png';
var osmAttrib='Map data © <a href="https://openstreetmap.org">OpenStreetMap</a>';
var osm = new L.TileLayer(osmUrl, {minZoom: 2, maxZoom: 18, attribution: osmAttrib})
.addTo(mymap);
```

```
</script>
</body>
</html>
```

Alternatif olarak aşağıdaki katmanlar da kullanılabilir. Burada `attribution` dizgesine katman tanımından önce uygun bir atama yapılmalıdır.

```
var osmBlackWhite = L.tileLayer('http://{s}.tiles.wmflabs.org/bw-mapnik/{z}/{x}/{y}.png', {
    attribution:attribution,
    maxZoom: 18
});
var openTopoMap = L.tileLayer('http://{s}.tile.opentopomap.org/{z}/{x}/{y}.png', {
    maxZoom: 17,
    attribution:attribution
});
var googlestreet = L.tileLayer('http://{s}.google.com/vt/lyrs=m&x={x}&y={y}&z={z}',{
    attribution:attribution,
    maxZoom: 20,
    subdomains:['mt0','mt1','mt2','mt3']
});
var googleHybrid = L.tileLayer('http://{s}.google.com/vt/lyrs=s,h&x={x}&y={y}&z={z}',{
    attribution:attribution,
    maxZoom: 20,
    subdomains:['mt0','mt1','mt2','mt3']
});
var googleSat = L.tileLayer('http://{s}.google.com/vt/lyrs=s&x={x}&y={y}&z={z}',{
    attribution:attribution,
    maxZoom: 20,
    subdomains:['mt0','mt1','mt2','mt3']
});
var googleTerrain = L.tileLayer('http://{s}.google.com/vt/lyrs=p&x={x}&y={y}&z={z}',{
    attribution:attribution,
    maxZoom: 20,
    subdomains:['mt0','mt1','mt2','mt3']
});
var tkgmparselleri = L.tileLayer('https://megsistile.tkgm.gov.tr/tkgm.ows.tile/tms/1.0.0
/TKGM:parsel_tematik_3857/{z}/{x}/{-y}.png',{
    attribution:attribution,
    maxZoom: 17
});
var arcgis= L.tileLayer('http://server.arcgisonline.com/ArcGIS/rest/services/
World_Imagery/MapServer/tile/{z}/{y}/{x}', {
    attribution:attribution,
    maxZoom: 18,
});
```

6.2 Map Objesi

Bir önceki bölümde de örneklerle açıklandığı üzere `map` en temel leaflet objesidir. Çok sayıda özelliği, seçenekleri ve yöntemleri vardır. Map olayları 6.5 başlığında ele alınacaktır. Map objesi örnek kodu aşağıdaki gibidir. Center ve zoom özellikleri tanımlanmak zorundadır.

```
var map = L.map('map', {
```

```
center: [51.505, -0.09],
zoom: 13
});
```

Bazı seçenekler (özellikler) ve yöntemler Çizelge 6.1 ve 6.2'de görülmektedir.

Çizelge 6.1: Bazı map seçenekleri

Seçenek	Tür	Varsayılan	Açıklama
center	latLng	Belirsiz	Harita merkez koordinatı
zoom	Sayı	Belirsiz	Zoom düzeyi
minZoom	Sayı		En küçük zoom düzeyi, belirtilmez ise görüntülenen katmanın (Tile Layer) en küçük zoom değeri kullanılır.
maxZoom	Sayı		En büyük zoom düzeyi, belirtilmez ise görüntülenen katmanın (Tile Layer) en büyük zoom değeri kullanılır.
layers	Layer[]	Dizi	Haritaya eklenmiş katmanları içeren dizi
maxBounds	LatLngBounds	null	Tanımlanır ise harita yalnızca verilen dörtgen bölgeyi görüntüler. Dinamik olarak belirlemek için setMaxBounds yöntemi kullanılabilir.

Çizelge 6.2: Map Yöntemleri

Yöntem	Geri Dönüş Türü	Açıklama
addControl (control)	this	Haritaya verilen kontrolü ekler.
removeControl (control)	this	Verilen kontrolü haritadan çıkartır.
addLayer (layer)	this	Verilen katmanı haritaya ekler.
setZoom (zoom level, zoom options)	this	Harita zoom düzeyi belirler.
fitBounds (LatLng-Bounds, fitBounds options)	this	Haritada verilen dikdörtgen bölgeyi görüntüler.
getCenter()	LatLng	Harita orta noktası koordinatlarını döndürür.
getZoom()	Sayı	Harita zoom düzeyi
getBounds()	LatLngBounds	Harita alanını LatLngBounds olarak verir.

6.3 Leaflet'de Bindirmeler

Benzer sistemlerde olduğu gibi Leaflet'tede harita üzerine marker (nokta işaret), polyline (çoklu doğru), polygon (kapalı çoklu doğru), popup (bilgi penceresi), görüntü vb. bindirmeler eklenebilir. Bu bölümde bindirmeler ele alınacaktır.

6.3.1 Marker

Haritaya Leaflet standart marker objesi eklemek için gerekli kod,

```
var marker = L.marker([37.87,32.5]).addTo(mymap);
```

şeklindedir. `[37.8,32.4]` marker konumunu (enlem, boylam) belirtmektedir. Bu şekilde oluşan marker ikonu mavi damla biçimindedir, istenirse değiştirilebilir. Bu amaçla iki resim dosyasına biri ikon görüntüsü diğer, ikonun gölgesi olmak üzere iki resim dosyasına ihtiyaç vardır. Resim dosyaları png formatında olmalı, zemin rengi şeffaf tanımlı olmalıdır. Örnek olarak `https://leafletjs.com/examples/custom-icons/` sayfasındaki yaprak görüntülerini ve gölgelerini kullanalım.

İkon Oluşturma

Marker ikonları marker kodlamasına bir seçenek olarak eklenebilen `L.Icon` objesi ile tanımlanır.

```
var greenIcon = L.icon({
iconUrl: 'leaf-green.png',
shadowUrl: 'leaf-shadow.png',

iconSize:     [38, 95], // ikon boyutu
shadowSize:   [50, 64], // golge boyutu
iconAnchor:   [22, 94], // ikonda marker uygulama noktasi
shadowAnchor: [4, 62],  // golgede uygulama noktasi
popupAnchor:  [-3, -76] // popup uygulama noktasi
});
```

Boyut ve uygulama noktası tanımlamaları piksel biriminde olup, piksel koordinat sistemi orijini resim sol üst köşesidir. x koordinatı sağa, y koordinatı aşağı doğru artar. Yukarıdaki ikonu kullanarak marker tanımı aşağıdaki gibi kolayca yapılır. Tanımlanan bir ikon bir çok marker ile ilişkilendirilebilir.

```
L.marker([51.5, -0.09], {icon: greenIcon}).addTo(map);
```

İkon Sınıfı

Ortak özellikleri olan çok sayıda ikon kullanımı için ikon sınıfı tanımlanabilir. Örnek:

```
var LeafIcon = L.Icon.extend({
options: {
shadowUrl: 'leaf-shadow.png',
iconSize:     [38, 95],
shadowSize:   [50, 64],
iconAnchor:   [22, 94],
shadowAnchor: [4, 62],
popupAnchor:  [-3, -76]
}
});
```

Bu sınıfı kullanarak üç değişik ikon tanımı aşağıdaki gibi olabilir.

```
var greenIcon = new LeafIcon({iconUrl: 'leaf-green.png'}),
redIcon = new LeafIcon({iconUrl: 'leaf-red.png'}),
orangeIcon = new LeafIcon({iconUrl: 'leaf-orange.png'});
```

6.3.2 Çoklu Doğru

Çoklu doğru oluşturmak için `Polyline` objesi kullanılır. Çoklu doğruyu oluşturan noktalardan oluşan bie noktalar dizisi gereklidir. Örnek kod:

```
var polygon = L.polyline([
    [37.866,32.466],
    [37.868,32.485],
    [37.871,32.492],
    [37.870,32.505]
]).addTo(mymap);
```

6.3.3 Poligon ve Daire

Poligon oluşturmak için coğrafi koordinatlardan (enlem, boylam) oluşan bir dizi gereklidir.

```
var polygon = L.polygon([
    [37.87,32.48],
    [37.86,32.50],
    [37.88,32.49]
]).addTo(mymap);
```

Daire eklemek için daire merkezinin derece biriminde coğrafi koordinatlarına ve metre biriminde yarıçapına gerek vardır. Daire poligon (alan) objesinin özel bil hali olduğundan çizgi renk ve kalınlığı, dolgu rengi vb. parametreleri de belirtilebilir. Örnek kod:

```
var circle = L.circle([37.87,32.49], {
    color: 'red',
    fillColor: '#f03',
    fillOpacity: 0.5,
    radius: 500
}).addTo(mymap);
```

Çoklu doğru, poligon ve daire objelerinin `path` sınıfından aldıkları çizgi ile dolgu renk ve kalınlıkları ile ilgili bazı ortak özellikleri Çizelge 6.3'de görülmektedir.

6.3.4 Bilgi Penceresi - Popup

Bir objeye bağlı olarak açılan bilgi pencereleri Leaflet'de `popup` olarak adlandırılmıştır. GoogleMaps API'deki infoWindow ile aynı özelliklerdedir. Popup'lar var olan objelere bağlanabilir ya da kendi başlarına tanımlanabilirler. Bağlama ile ilgili örnek kodlar[1]:

```
marker.bindPopup("<b>Merhaba!</b><br>Bu bir bilgi penceresidir.").openPopup();
circle.bindPopup("Daire");
polygon.bindPopup("Çokgen");
```

[1] marker, circle ve polygon önceden tanımlanmış olmalıdır.

Çizelge 6.3: Çizgi ve dolgu özellikleri

Özellik	Türü	Varsayılan	Açıklama
stroke	Boolean	true	Çizgi görüntülensin/görütülenmesin
color	String	'#3388ff'	Çizgi rengi
weight	Number	3	Çizgi kalınlığı
opacity	Number	1.0	Çizgi opaklığı (0-1 arasında)
fill	Boolean		Dolgu görüntülensin/görütülenmesin
fillColor	String		Dolgu rengi
fillOpacity	Number	0.2	Dolgu opaklığı

Herhangi bir objeden bağımsız popup örneği ise aşağıdaki gibi olabilir.

```
var popup = L.popup()
    .setLatLng([37.87,32.5])
    .setContent("Kendi başına popup örneği")
    .openOn(mymap);
```

Popup'ların içeriğinde düz metin olabileceği gibi HTML elemanları da desteklenir.

6.4 Temel Objeler

6.4.1 latLng

Coğrafi koordinatları (enlem, boylam) içeren objedir. Koordinatlar ondalık derece biriminde olmalıdır. Örnek kod:

```
var koord = L.latLng(50.5, 30.5);
```

Tüm leaflet yöntemleri `latLng` objelerini hem basit dizi, hem de obje sabiti olarak kabul eder. Aşağıdaki kodlar eşdeğerdir.

```
map.panTo([50, 30]);
map.panTo({lon: 30, lat: 50});
map.panTo({lat: 50, lng: 30});
map.panTo(L.latLng(50, 30));
```

Özellikler ve yöntemler Çizelge 6.4 ve 6.5'de verilmiştir.

6.4.2 LatLngBounds

Haritada dikdörtgen bir coğrafi alanı temsil eder. Buradaki dikdörtgen alan meridyen ve paralellerle sınırlanmış olup, kullanılan Web Merkator projeksiyonu nedeniyle haritada dikdörtgen bir görünüme sahiptir. Örnek kod:

```
var corner1 = L.latLng(40.712, -74.227),
corner2 = L.latLng(40.774, -74.125),
bounds = L.latLngBounds(corner1, corner2);
```

Çizelge 6.4: latLng Yöntemleri

Yöntem	Geri Dönüş Türü	Açıklama
equals(latLng)	Mantıksal	Verilen iki koordinatın belli bir hata içerisinde aynı olup, olmadığı bulur.
toString	Dizge	Koordinatı yazıya dönüştürür.
distanceTo (latLng, latLng)	Sayı	Küresel kosinüs teoremi ile iki nokta arası uzaklığı metre biriminde hesaplar (Bkz. (Bildirici, 2019)).
wrap()	latLng	Boylamın ±180° arasında kalmasını sağlayacak şekilde koordinat üretir.
toBounds (Number, boyut_metre)	LatLngBounds	Verilen koordinat ortada olacak şekilde verilen boyutta çerçeve obje oluşturur.

Çizelge 6.5: latLng Özellikleri

Özellik	Türü	Açıklama
lat	Sayı	Enlem (derece)
lng	Sayı	Boylam (derece)
alt	Sayı	Yükseklik, metre (opsiyonel)

LatLngBounds gerektiren tüm yöntemlere basit dizi şeklinde de kodlama yapılabilir. Örnek:

```
map.fitBounds([
[40.712, -74.227],
[40.774, -74.125]
]);
```

Bazı yöntemler Çizelge 6.6'de verilmiştir.

Çizelge 6.6: Bazı LatLngBounds Yöntemleri

Yöntem	Geri Dönüş Türü	Açıklama
extend (LatLng)	Kendisi	Verilen koordinatı kapsayacak şekilde genişletme
extend (LatLng-Bounds)	Kendisi	Verilen dörtgene güncelleme
getCenter()	LatLng	Merkez koordinatları

6.4.3 Point

Piksel koordinatlarında bir noktayı (x,y) temsil eder. Örnek kod:

```
var point = L.point(200, 300);
```

Nokta kabul eden tüm yöntemler basit dizi olarak da point kabul eder. Aşağıdaki kodlar eşdeğerdir.

Çizelge 6.7: Point Yöntemleri

Yöntem	Geri Dönüş Türü	Açıklama
distanceTo (Point, otherPoint)	Sayı	Kartezyen uzaklık hesabı
toString()	Dizge	Koordinatları yazıya dönüştürür.

```
map.panBy([200, 300]);
map.panBy(L.point(200, 300));
```

Özellikleri `x` ve `y`'dir. Bazı yöntemler Çizelge 6.7'de verilmiştir.

Piksel koordinatları haritanın göründüğü pencereye göre olup, orijin sol üst köşedir. `x` sağa, `y` aşağıya doğrudur. Zoom düzeyi değiştikçe piksel koordinatları farklı coğrafi koordinatlara karşılık gelirler. Bu açıdan `Point` ile `latLng` arasındaki farklılığa dikkat edilmesi gerekir.

6.5 Olaylar

Leaflet'te kullanıcının haritaya tıklaması gibi bir olay meydana geldiğinde ilgili obje bir fonksiyon yardımıyla izlenebilen bir kod üretir.

```
function onMapClick(e) {
    alert("Haritaya şurada tıkladınız: " + e.latlng);
}
mymap.on('click', onMapClick);
```

Yukarıda kod ile haritaya tıklandığında `onMapClick` fonksiyonu çalışmaktadır. Fonksiyon adı keyfi olarak seçilmiş olup, `e` parametresinin `latlng` özelliği ile haritada tıklanan yerin koordinatlarına ulaşılabilmektedir. Burada `mymap.on` ile olay ve olayın gerçekleşmesi halinde çalışacak fonksiyon belirtilmektedir. Harita objesinin duyarlı olduğu çok sayıda olay vardır. Harita-kullanıcı etkileşimi ilgili olaylardan bazıları Çizelge 6.8'de verilmiştir. Her objenin Leaflet (2018) adresinde ayrıntıları

Çizelge 6.8: Harita-kullanıcı etkileşimi ile ilgili olaylar

Olay	Türü	Tetiklenme
click	MouseEvent	Kullanıcı haritaya tıklarsa/dokunursa
dblclick	MouseEvent	Kullanıcı haritaya çift tıklarsa/dokunursa
mousedown	MouseEvent	Kullanıcı haritada fare butonuna basarsa
mouseup	MouseEvent	Kullanıcı haritada fare butonunu bırakırsa
mouseover	MouseEvent	İmleç harita üzerine gelirse
mouseout	MouseEvent	İmleç harita dışına çıkarsa
mousemove	MouseEvent	Fare harita üzerinde ise
keypress	KeyboardEvent	Harita aktifken bir tuşa basılırsa

verilmiş kendine ait olayları vardır. Olayları dinleyen fonksiyonların parametresi (`e`) olaylar hakkında yararlı bilgileri kapsar. Tipik olarak `e.latlng` ile konum bilgisine ulaşılır. Yukarıdaki kod, konum bilgisi bir `popup` içeriğinde gösterilerek geliştirilebilir.

```
var popup = L.popup();
```

```
function onMapClick(e) {
    popup
        .setLatLng(e.latlng)
        .setContent("Haritada tıklanan yer " + e.latlng.toString())
        .openOn(mymap);
}
mymap.on('click', onMapClick);
```

6.6 GeoJSON Kullanımı

GeoJSON, CBS teknolojilerinde yaygın kullanılan bir vektör veri formatı olup Leaflet'de desteklenir. Anlaşılması ve kullanımı kolaydır. RFC 7946 tanımı:

GeoJSON çeşitli coğrafi veri yapılarını kodlamak için bir formattır. Bir GeoJSON objesi uzayda bir bölgeyi (Geometry), mekansal sınırlı bir detayı (Feature) ya da detaylar kümesini (Feature Collection) temsil edebilir. GeoJSON nokta (point), çizgi (LineString), çokgen (Polygon), Multipoint, MultiLineString, Multipolygon ve GeometryCollection geometrilerini destekler. GeoJSON'da detaylar bir geometri objesi ve ek özellikleri içerirler. Detay kümesi (Feature Collection) ise detaylar listesini içerir.

Leaflet yukarıdaki GeoJSON obje yapısını destekler. Aşağıda bir GeoJSON detay örneği yer almaktadır.

```
var geojsonFeature = {
  "type": "Feature",
    "properties": {
    "name": "Büyükşehir Stadyumu",
    "amenity": "Stadyum",
    "popupContent": "Konyaspor sahası!"
  },
  "geometry": {
    "type": "Point",
    "coordinates": [32.488,37.946]
  }
};
```

GeoJSON formatında Leaflet'ten farklı olarak coğrafi koordinat yazımı [boylam,enlem] biçimindedir (tam tersi!). GeoJSON objeleri haritaya GeoJSON katmanı ile eklenirler.

```
L.geoJSON(geojsonFeature).addTo(map);
```

Yukarıdaki gibi eklenen bir nokta varsayılan ikon ile görüntülenir.

GeoJSON objesi bir objelerin dizisi olarak da katmana aktarılabilir.

```
var myLines = [{
  "type": "LineString",
  "coordinates": [[32.458492,37.848932],[32.458513,37.848563],
  [32.458472,37.848233],[32.457951,37.848146]]
}, {
  "type": "LineString",
  "coordinates": [[32.45761039662455,37.84858525357361],
```

```
  [32.45789250717565,37.84811013969157],
  [32.45828161677994,37.8474450795229]]
}];
```

Boş bir GeoJSON katmanı oluşturup daha sonra katmana ekleme yapılabilir. Bu şekilde daha sonra ekleme yapmak mümkün olur.

```
var myLayer = L.geoJSON().addTo(map);
myLayer.addData(geojsonFeature);
```

Seçenekler

`style` seçeneği iki farklı biçimde kullanılabilir. Birinci kullanımı aynı stili paylaşan basit bir obje olabilir.

```
var myLines = [{
  "type": "LineString",
  "coordinates": [[32.458492,37.848932],[32.458513,37.848563],
  [32.458472,37.848233],[32.457951,37.848146]]
}, {
  "type": "LineString",
  "coordinates": [[32.45761039662455,37.84858525357361],
  [32.45789250717565,37.84811013969157],
  [32.45828161677994,37.8474450795229]]
}];

var myStyle = {
  "color": "#117800",
  "weight": 4,
  "opacity": 0.5
};

L.geoJSON(myLines, {style: myStyle}).addTo(map);
```

Buna alternatif obje özelliklerine bağlı biçimlendirme yapan bir fonksiyon kullanılabilir. Aşağıdaki örnekte "party" özelliği kontrol edilerek çokgenlerin stili buna bağlı tanımlanmaktadır.

```
var states = [{
  "type": "Feature",
  "properties": {"party": "Republican"},
  "geometry": {
  "type": "Polygon",
  "coordinates": [[
    [-104.05, 48.99],
    [-97.22,  48.98],
    [-96.58,  45.94],
    [-104.03, 45.94],
    [-104.05, 48.99]
  ]]
}
}, {
  "type": "Feature",
```

```
  "properties": {"party": "Democrat"},
  "geometry": {
  "type": "Polygon",
  "coordinates": [[
    [-109.05, 41.00],
    [-102.06, 40.99],
    [-102.03, 36.99],
    [-109.04, 36.99],
    [-109.05, 41.00]
  ]]
}
}];

L.geoJSON(states, {
  style: function(feature) {
  switch (feature.properties.party) {
    case 'Republican': return {color: "#ff0000"};
    case 'Democrat':   return {color: "#0000ff"};
  }
}
}).addTo(map);
```

Noktalar çoklu doğrular ve çokgenlerden farklı ele alınır. Aksi belirtilmedikçe geoJSON noktalarında basit markerlar görüntülenir. Bunu GeoJSON seçeneklerinde `pointToLayer` fonksiyonu kullanarak değiştirebiliriz. Bu fonksiyona `LanLng` ile koordinatlar geçirilir ve `Marker` ya da `CircleMarker` geri döndürülür. Aşağıdaki örnekte `pointToLayer` fonksiyonu `CircleMarker` görüntülenmek üzere kullanılmaktadır.

```
var geojsonMarkerOptions = {
  radius: 8,
  fillColor: "#ff7800",
  color: "#000",
  weight: 1,
  opacity: 1,
  fillOpacity: 0.8
};

L.geoJSON(someGeojsonFeature, {
  pointToLayer: function (feature, latlng) {
    return L.circleMarker(latlng, geojsonMarkerOptions);
  }
}).addTo(map);
```

`CircleMarker` nokta koordinatını merkez alarak bir daire görüntülenmesini sağlar. `radius` özelliği ile piksel biriminde yarıçap belirlenir. Diğer özellikleri `Circle` bindirmesi ile aynıdır.

`onEachFeature` seçeneği her obje GeoJSON katmanına eklenmeden önce çağrılan bir fonksiyondur. Bu fonksiyonun yaygın kullanım alanlarından biri objelere tıklanma halinde bilgi penceresi açılmasını sağlamaktadır. Aşağıda bunun bir örneği görülmektedir.

```
function onEachFeature(feature, layer) {
  // popupContent isimli bir özellik var mı?
  if (feature.properties && feature.properties.popupContent) {
    layer.bindPopup(feature.properties.popupContent);
```

```
  }
}

var geojsonFeature = {
  "type": "Feature",
  "properties": {
    "name": "Coors Field",
    "amenity": "Baseball Stadium",
    "popupContent": "This is where the Rockies play!"
  },
  "geometry": {
    "type": "Point",
    "coordinates": [-104.99404, 39.75621]
  }
};

L.geoJSON(geojsonFeature, {
  onEachFeature: onEachFeature
}).addTo(map);
```

`filter` seçeneği GeoJSON objelerinin görüntülenmelerini kontrol etmek için kullanılır. Aşağıda buna ait bir örnek bulunmaktadır.

```
var someFeatures = [{
  "type": "Feature",
  "properties": {
  "name": "Coors Field",
  "show_on_map": true
},
  "geometry": {
  "type": "Point",
  "coordinates": [-104.99404, 39.75621]
}
}, {
  "type": "Feature",
  "properties": {
  "name": "Busch Field",
  "show_on_map": false
},
  "geometry": {
  "type": "Point",
  "coordinates": [-104.98404, 39.74621]
}
}];

L.geoJSON(someFeatures, {
  filter: function(feature, layer) {
    return feature.properties.show_on_map;
}
}).addTo(map);
```

6.6.1 GeoJSON Objelerinin Dış Dosyalarda Bulunması

GeoJSON objeleri JavaScript kodunun içinde olmak yerine harici bir dosyada olabilir. Veri hacmi fazla ise okunaklı kod yazabilmek için böyle olması zorunludur. Yaygın kullanılan vektör dosyalarını çoğu CBS yazılımı (Örneğin QGIS) GeoJSON formatına dönüştürmektedir. Bu amaçla çevrimiçi hizmet veren siteler de mevcuttur. GDAL paketinde bulunan ogr2ogr programının dönüşüm yaptığı formatlar arasında GeoJSON da bulunmaktadır.

Çeşitli yazılımlarla oluşturulan GeoJSON dosyaları bir feature collection içerirler. Dosya içeriğinin bir katmana aktarılması için JQuery kütüphanesinden yararlanılabilir. Bu amaçla `<head>` elemanında aşağıdaki script olmalıdır.

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js">
</script>
```

Aşağıdaki örnek kod ile turkiye.geojson adlı dosya haritaya bir katman olarak eklenmektedir.

```
function addDataToMap(data, map) {
    var dataLayer = L.geoJson(data);
    dataLayer.addTo(map);
}
$.getJSON("turkiye.geojson", function(data) { addDataToMap(data, map); });
```

Bu örnekte HTML dosya ile GeoJSON dosyasının aynı klasörde olması gereklidir. Bunun yerine bir URL de belirtilebilir. Ancak yeni tarayıcılardaki güvenlik nedeniyle yerel olarak bu kod çalışmaz. Sunucuda çalışır.

İkinci bir yol GeoJSON dosyasının başına kod ekleyerek feature collection'ı bir değişken olarak tanımlamaktır. Bu seçenekte tarayıcı güvenliğinden kaynaklanan problem elimine edildiği gibi JQuery kullanımına da gerek kalmaz. Yukarıdaki örnekteki dosyanın en başına `var turkiye=` eklenerek dosya başlangıcı aşağıdaki gibi yapılır.

```
var turkiye={
"type": "FeatureCollection",
"name": "turkiye",
..........
```

Bu dosya içeriği dış js dosyası gibi `<head>` içinde script olarak tanımlanır.

```
<script type="text/javascript" src="turkiye.geojson"></script>
```

Tanımlı haritaya (map) eklenir.

```
L.geoJson(turkiye).addTo(map);
```

Burada `turkiye` GeoJSON dosyanın başına eklenen değişken adıdır.

6.6.2 GeoJSON Tematik Harita Uygulaması

Yukarıdaki örnek dosyada bazı öznitelikler ile birlikte alan olarak iller yer almaktadır. Dosyaya ders web sayfasından ulaşılabilir[2]. Burada nüfus yoğunluğunu gösteren bir koroplet harita örneği verilecektir. Dosyada ALAN_KURE ve N_2019 (2019 nüfusu) alanları bulunmaktadır. 2019

[2] `http://galileo.ktun.edu.tr/1205641`)

nüfus yoğunluğu bunların oranlanmasından elde edilecektir. Bu değer illere göre 20-3000 arasında değişmektedir. ColorBrewer (`https://colorbrewer2.org/`) renkleri ve keyfi bir sınıflandırma ile veriye göre renk değeri veren bir fonksiyon yazalım.

```
function renkVer(d) {
    return d > 1000 ? '#800026' :
           d > 500  ? '#BD0026' :
           d > 200  ? '#E31A1C' :
           d > 100  ? '#FC4E2A' :
           d > 50   ? '#FD8D3C' :
           d > 20   ? '#FEB24C' :
           d > 10   ? '#FED976' :
                      '#FFEDA0';
}
```

Burada koşul operatörü kullanılmıştır. Nüfus yoğunluğuna göre il alanlarının stilini belirleyen bir fonksiyon yazalım.

```
function stilVer(feature) {
    return {
        fillColor: renkVer(feature.properties.N_2019/feature.properties.ALAN_KURE),
        weight: 2,
        opacity: 1,
        color: 'white',
        dashArray: '3',
        fillOpacity: 0.7
    };
}
```

Dosya içeriğini katman olarak bu fonksiyondan stil tanımı alarak yükleyelim.

```
L.geoJson(turkiye,{style: stilVer}).addTo(map);
```

Bu uygulama daha da geliştirilebilir.

Kaynaklar

Bildirici, İ. Ö. (2019). *Kartografya: Harita Tasarımı ve Kullanımı için Gerekli Bilim, Sanat ve Teknik*. Atlas Akademi Yayınevi, Konya, 2. edition.

Leaflet (2018). Leaflet: an open-source javascript library for mobile-friendly interactive maps. `https://leafletjs.com/`. Giriş: 01.11.2018.