

Ağ; belli bir alan içerisindeki cihazların birbiri ile iletişim halinde olacak şekilde düzenlenmesidir. Ağ yapısında cihazlar belli başlı şartlar (protokoller) çerçevesinde iletişime geçer.

İnternet; en basit hali ile ağların ağı (network of networks) olarak tanımlanabilir.

Browser: Web sunucusuna bağlanan, oradaki HTML belgesini alıp okuyan ve okuduklarını yorumlayıp ekrana getiren bir araçtır. Chrome, Opera ...

WWW (World Wide Web): Yazı, grafik, ses, film gibi pek çok farklı yapıdaki verilere kompakt ve etkileşimli bir şekilde ulaşmamızı sağlayan bir çoklu hiper ortam sistemidir.

HTTP (Hyper Text Transfer Protokol): Web üzerinde iletişim ve haberleşmenin sağlanmasında kullanılan yaygın ve basit protokollerden biridir.

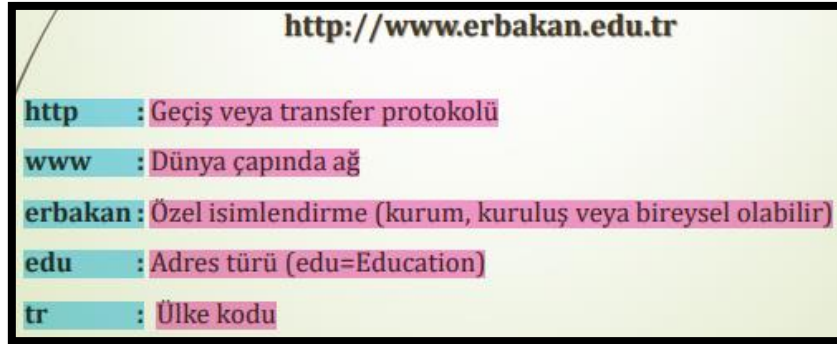
Bu protokol temel olarak 4 aşamadan oluşur:

1. Bağlantının açılması,
2. Açılan bağlantı üzerinden istek gönderilmesi,
3. İsteğe cevap gönderilmesi,
4. Bağlantının kapatılması

FTP (File Transfer Protokol): İnternete bağlı iki bilgisayar arasında karşılıklı dosya aktarımı yapmak için geliştirilen bir internet protokoldür. Bir web sitesine eklenmek istenen dosyalar da FTP üzerinden yüklenir.

URL (Uniform Resource Loader): Web Browser ile bir web servisine ya da diğer bazı İnternet servislerine yönlendirme yapılabilmesini sağlar.

Aslında web adresleri IP adresleri (numerik değerler) ile temsil edilir ancak kullanıcıların daha basit bir şekilde kullanması ve akılda kalıcı olması sebebiyle web adresleri kullanılır.



Sunucu (Server): İnternete bağlı diğer bilgisayarlara hizmet sunan cihazlardır. Sunucu cihazlar her zaman internete bağlı (online) ve istemcilerden gelen isteklere cevap verme durumundadır.

İstemci (Client): Ağdaki sunuculardan hizmet alan cihazlardır. İstemcinin her zaman online olması gerekmez, ihtiyaç zamanında bağlanıp taleplerde bulunur.

HTML (HYPER TEXT MARKUP LANGUAGE) TEMEL KOD YAPISI

<html> </html> Tüm HTML kodlarımız html etiketleri arasına yazılır.

<head> </head> Head etiketi, web sitemizin baş kısmını oluşturan etikettir. Bu kısma web sitemizin özelliklerini, içeriğini, web sitesinin yazarını, web sitesinin hangi konu ile alakalı olduğunu belirten kodlamalar gelir. **Head etiketinin içine title etiketinin yazılması zorunludur,** çünkü title sitemizin başlığı olduğu gibi html dosyamızın da başlığı olacaktır.

<title> </title> Title etiketi sitemizin başlığını belirler.

<body> </body> Body etiketi için sayfamızın ana gövdesi diyebiliriz. HTML belgesindeki içerikler; bağlantılar, resimler, tablolar vb. bu etiket içerisinde yer alır.




1. **text = "renk"** Sayfanızdaki yazıların rengine renk ile belirtilen değeri verir.
2. **link = "renk"** Sayfanızdaki bağların rengine renk ile belirtilen değeri verir.
3. **vlink = "renk"** Sayfanızdaki ziyaret edilmiş bağların rengine renk ile belirtilen değeri verir.
4. **alink = "renk"** Sayfanızdaki bağların tıklandığı andaki rengine renk ile belirtilen değeri verir.
5. **bgcolor = "renk"** Sayfanızın arka plan rengine renk ile tanımlı değeri verir.
6. **background = "resim_dosyası"** Eğer arka planda sade bir renk değil de bir resim kullanmak istiyorsanız, resim_dosyası kullanacağınız resmin adını temsil eder.
7. **topmargin = "değer"** Sayfamızda kullanacağımız bileşenlerin üst tarafa olan uzaklığını belirler.
8. **leftmargin = "değer"** Sayfamızda kullanacağımız bileşenlerin sol tarafa olan uzaklığını belirler.
9. **rightmargin = "değer"** Sayfamızda kullanacağımız bileşenlerin sağ tarafa olan uzaklığını belirler.
10. **onload = "betik"**
11. **onunload = "betik"**

Bir sayfanın yüklenmesi, yani bir sayfanın ziyaret edilmesi **load** olayıdır. Sayfanın terk edilmesi ise **onunload** olayıdır.

BAŞLIK – AÇIKLAMA – YORUM ETİKETİ

- **<h?> ... </h?>** başlık oluşturmak için kullandığımız bir etikettir. h1'den h6'ya kadardır. h1 en büyük boyutu h6 ise en küçük boyutu belirler. **<body> </body>** etiketleri arasında yer almalıdır.
- Başlıkların boyutunu büyütmek için **<h2 style = "font-size: 60px;"> ... </h2>** komutunu kullanırız.
- **<!-- -->** Komutu yorum satırı oluşturmamızı sağlar. **<!--KODLAR -->** bu şekilde kullanılmasının sebebi kullanılan browser o kodları desteklemezse karışıklık çıkmasın diye yorum satırına alınır.



```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="utf-8">
5   <title></title>
6 </head>
7 <body>
8   <h1 style="font-family:courier;">Hasan</h1>
9   <h2 style="background: aliceblue;">Hasan</h2>
10  <h3 style="color: tomato;">Hasan</h3>
11  <h4 style="font-size: 50px;">Hasan</h4>
12  <h5 style="text-align: right;">Hasan</h5>
13  <h6>Hasan</h6>
14 </body>
15 </html>
```

PARAGRAF ETİKETİ – SATIR ATLATMA – ÇİZGİ – KALINLAŞTIRMA – ALT&ÜST İNDİSLER – ITALIC YAZI TİPİ

- **<p> </p>** Etiketini paragrafları tanımlamak için kullanılır.
- **
** Etiketini bir alt satıra inmemizi sağlar.
- **<hr>** Sayfada yatay bir çizgi çizer.
- ** ... ** ya da ** ... ** Etiketleri metni kalınlaştırmamızı sağlar.
- **<i> ... </i>** ya da ** ... ** Etiketini ile metni italic, eğik biçimde yazabiliriz.
- **_{...}** ve **^{...}** Etiketleri metni alt ve üst indis olarak işaretlememizi sağlar.
- **<ins> ... </ins>** ya da **<u> ... </u>** etiketi metnin altını, ** ... ** ya da **<strike>...</strike>** etiketi de metnin üzerini çizmemizi sağlar.

```
<p> <b>Sıcak bir yaz günüydü.</b>
  <strong> Havada o kadar güzel süzülüyordu ki papatya onu hayranlıkla izledi.</strong>
<hr> <br>
  <i>Arı kardeş ne kadar güzel uçuyorsun.</i>
</p>
5<sub>2</sub>
6<sup>1</sup>
<br><br>
<ins>WEB PROGRAMLAMA</ins>
<br> <br>
<del>WEB TASARIMI</del>
</body>
```

Sıcak bir yaz günüydü. Havada o kadar güzel süzülüyordu ki papatya onu hayranlıkla izledi.

Arı kardeş ne kadar güzel uçuyorsun.

5₂ 6¹

WEB PROGRAMLAMA

~~WEB TASARIMI~~

RESİM EKLEME

`` etiketi ile sitemize resim ekleyebiliriz. `img` etiketinin iki zorunlu özelliği vardır bunlar: `src` ve `alt`

- **DOSYA ADRESİ** yazan yere resmin adresini yapııştırarak sayfamıza resim ekleyebiliriz.
- **src** Resmin adresini yani hangi klasörde olduğunu belirler.
- **alt** Resim verilen dosya adresinde bulunamazsa sayfada buraya yazılan yazı görüntülenir.
- **height** Resmin yüksekliğini belirler. **width** Resmin genişliğini belirler. (Eğer `auto` yazarsak genişliği kendisi ayarlar.)
- **border** Resmin kenarlık genişliğini belirler
- **vspace** ve **hspace**: Sırası ile resmin metne olan dikey ve yatay uzaklığı



LİNK VERME

- `<a ... > ... ` etiketi metinlere, resimlere bağlantı oluşturmak için kullanılır. Bağlantının nereye yönlendirileceği `href` özelliği içerisinde belirtilir.
- `target=""` özelliği ise gideceğimiz sayfanın nerede açılacağını belirler. Yani yeni bir sayfada mı açılacak yoksa aynı sayfa üzerinden o linke mi gidileceğini belirtmek için bu özelliği kullanırız. Bunlar `_blank` ve `_self`'dir. `_blank` özelliği tıklanan linki yeni bir sekmede ya da yeni bir pencerede açar. `_self` ise tıklanan linki aynı sayfada açar.
- Ziyaret edilmemiş bir bağlantının altı çizili ve mavidir, ziyaret edilen bağlantının altı çizili ve mor.



DİV ETİKETİ

- `<div ... > ... </div>` etiketinin yapısı dört köşelidir ve içerisine aldığı elementlerle yüksekliği ve boyutu değişebilir. Aslında bir çerçeve görevi görüyor diyebiliriz. id, class ve style parametrelerine sahiptir.
- Aşağıdaki örnekte border, kırmızı çerçevenin büyüklüğünü temsil etmektedir. Solid, çerçevenin rengini, font-size iste yazı boyutunu temsil etmektedir.

```
<div style="border:10px solid red; padding:10px; font-size:20px">
<p>Yeni bir dil öğrenmek için burdayız.</p>
<p>Evet diyorsanız devam edin.</p>
</div>
```

Yeni bir dil öğrenmek için burdayız.

Evet diyorsanız devam edin.

SIRALI VE SIRASIZ LİSTELER & AÇIKLAMA LİSTELERİ

- HTML üzerinde sıralı ve sırasız olmak üzere listeleri kullanabiliriz. Sıralı liste için ` ... ` etiketini açtıktan sonra liste elemanlarını ` ... ` olarak tanımlamamız gerekiyor.
 - 1: Onluk tabanda numaralama (1,2,3,4,...)
 - i: Küçük rakamlarla romen sayıları (i,ii,iii,iv,...)
 - l: Büyük rakamlarla romen sayıları (l,II,III,IV,...)
 - a: Küçük harflerle alfabetik (a,b,c,...)
 - A: Büyük harflerle alfabetik (A,B,C,...)
- Sırasız liste için ` ... ` etiketini açtıktan sonra liste elemanlarını ` ... ` olarak tanımlamamız gerekiyor.
 - **disc:** İçi dolu bir daire görüntüler
 - **circle:** İçi boş bir daire görüntüler
 - **square:** İçi dolu ya da boş bir kare görüntüler
- HTML ayrıca açıklama listelerini de destekler. `<dl>` etiketi açıklama listesi tanımlar, `<dt>` etiketi terimi (ad) tanımlar ve `<dd>` etiketi terimleri açıklar.

<code><body></code>	1. Seviye 1
<code></code>	2. Seviye 2
<code>Seviye 1</code>	o Level 1
<code>Seviye 2</code>	o Level 2
<code></code>	3. Seviye 3
<code>Level 1</code>	4. Seviye 4
<code>Level 2</code>	Kahve
<code></code>	Sıcak İçecek
<code>Seviye 3</code>	Süt
<code>Seviye 4</code>	Soğuk İçecek
<code></code>	
<code><dl></code>	
<code><dt>Kahve</dt></code>	
<code><dd>Sıcak İçecek</dd></code>	
<code><dt>Süt</dt></code>	
<code><dd>Soğuk İçecek</dd></code>	
<code></dl></code>	
<code></body></code>	

list-style-type işaretleyici türünü belirtir.

```
<head>
<style>
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;
}
</style>
</head>
<body>
```

```
<ul class="a">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ul>
```

```
<ul class="b">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ul>
```

```
<ol class="c">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ol>
```

```
<ol class="d">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ol>
```

```
</body>
```

- Coffee
- Tea
- Coca Cola

- Coffee
- Tea
- Coca Cola

- I. Coffee
- II. Tea
- III. Coca Cola

- a. Coffee
- b. Tea
- c. Coca Cola

```
<body>
  <ol type="i">
    <li> Birinci öge...</li>
    <li> İkinci öge ...
      <ul type="square">
        <li>İkinci ögenin bir ögesi...
        <li>İkinci ögenin başka bir ögesi...
      </ul>
    </li>
    <li> Üçüncü öge...</li>
  </ol>
</body>
```

- i. Birinci öge...
- ii. İkinci öge ...
 - İkinci ögenin bir ögesi...
 - İkinci ögenin başka bir ögesi...
- iii. Üçüncü öge...

list-style-position özellik listesi maddelik belirteçlerin konumunu belirler.

```
<head>
<style>
ul.a {
  list-style-position: outside;
}
ul.b {
  list-style-position: inside;
}
</style>
</head>
<body>

<h2>list-style-position: outside (default):</h2>
<ul class="a">
  <li>Coffee -nt</li>
  <li>Tea n evee to Asia</li>
  <li>Coca Cola -ves</li>
</ul>

<h2>list-style-position: inside:</h2>
<ul class="b">
  <li>Caerrfea plant</li>
  <li>Tea -ush)to Asia</li>
  <li>Coca Colas</li>
</ul>

</body>
```

list-style-position: outside (default)

- Coffee -nt
- Tea n evee to Asia
- Coca Cola -ves

list-style-position: inside:

- Caerrfea plant
- Tea -ush)to Asia
- Coca Colas

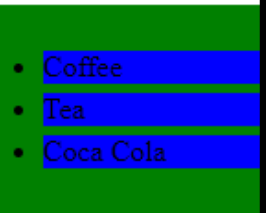
Listelerin Boyutunu ve Rengini Değiştirme

```
<head>
<style>
ul {
  background: green;
  padding: 20px;
}

ul li {
  background: blue;
  margin: 5px;
}
</style>
</head>
<body>

<ul>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ul>

</body>
```

- 
- Coffee
 - Tea
 - Coca Cola

TABLolar

- Bir tablo oluşturmak için `<table>` etiketini kullanırız.
- `<thead>` başlığı oluştururken `<tbody>` tablonun gövde kısmını oluşturmaktadır.
- Verilerin olduğu satır ve sütunlar ise `<tbody></tbody>` arasına yazılmaktadır.
- Her tablo satırı `<tr>` etiketi ile başlar, `</tr>` etiketi ile biter. Sütunları belirtmek için `<td>` etiketlerini kullanırız. Sütun başlıklarını belirtmek için `<th>` etiketini kullanabiliriz.
- Tablomuzun kodlarını yazarken `<border>` özelliğini kullandık, bu özellik tablomuzun kenarlık kalınlığını ayarlar. Bu özelliği kullanmasaydık tablomuzun kenarlık çizgileri oluşmayacaktı. `align="center"` yazdığımız için tablo, web sayfasının ortasında yer alıyor. `align="left"` yazmış olsaydık tablo, web sayfasının solunda yer alacaktı. `align="right"` yazmış olsaydık web sayfasının sağında yer alacaktı.
- Tablonun en altına bir açıklama eklenmek istenirse `<caption>` tag'i kullanılabilir.

```
<style>
table {
  border:2px solid red;
}
td{
  border: 3px solid yellow
}
</style>
<body>

<table style="width:100%">
  <tr>
    <th>NAME</th>
    <td>Emil</td>
    <td>Tobias</td>
  </tr>
  <tr>
    <th>NUMBER</th>
    <td>16</td>
    <td>14</td>
  </tr>
</table><br>

<table border="4" align="center">
  <tr>
    <th>İSİM</th>
    <th>NUMARA</th>
  </tr>
  <tr>
    <td>Hasan</td>
    <td>123</td>
  </tr>
  <tr>
    <td>Beyza</td>
    <td>456</td>
  </tr>
</table>
</body>
```

NAME	Emil	Tobias
NUMBER	16	14

İSİM	NUMARA
Hasan	123
Beyza	456

Tablo Başlığı (thead)	Tablo ve sütun başlıkları	Tablo gövdesi												
<table><thead><tr><th>1.Sütun</th><th>2.Sütun</th><th>3.Sütun</th></tr></thead><tbody><tr><td>hücre1</td><td>hücre2</td><td>hücre3</td></tr><tr><td>hücre4</td><td>hücre5</td><td>hücre6</td></tr><tr><td>hücre7</td><td>hücre8</td><td>hücre9</td></tr></tbody></table>	1.Sütun	2.Sütun	3.Sütun	hücre1	hücre2	hücre3	hücre4	hücre5	hücre6	hücre7	hücre8	hücre9	<pre><table border="1"> <thead> Tablo Başlığı (thead) </thead> <caption align="bottom"> alt-yazı (caption) <caption> <tr> <th>1.Sütun</th> <th>2.Sütun</th> <th>3.Sütun</th> </tr> </table></pre>	<pre><tbody> <tr> <td>hücre1</td> <td>hücre2</td> <td>hücre3</td> </tr> <tr> <td>hücre4</td> <td>hücre5</td> <td>hücre6</td> </tr> <tr> <td>hücre7</td> <td>hücre8</td> <td>hücre9</td> </tr> </tbody> </table></pre>
1.Sütun	2.Sütun	3.Sütun												
hücre1	hücre2	hücre3												
hücre4	hücre5	hücre6												
hücre7	hücre8	hücre9												

- **cellpadding**, tablo hücresi içerisine eklenen metin/graifk/resim gibi elemanların hücre kenarlarına olan mesafesini temsil eden parametredir.

<div style="border: 1px solid black; padding: 5px; width: 50px; height: 30px; display: flex; align-items: center; justify-content: center;">hücre</div>	<pre><table border="1" cellpadding="5"> <tr> <td>hücre</td> </tr> </table></pre>
<div style="border: 1px solid black; padding: 10px; width: 50px; height: 30px; display: flex; align-items: center; justify-content: center;">hücre</div>	<pre><table border="1" cellpadding="10"> <tr> <td>hücre</td> </tr> </table></pre>

- **cellspacing**, tablo hücreleri arasındaki mesafe değerini temsil eden parametredir.

<div style="border: 1px solid black; padding: 5px; width: 50px; height: 30px; display: flex; align-items: center; justify-content: center;">hücre1</div> <div style="border: 1px solid black; padding: 5px; width: 50px; height: 30px; display: flex; align-items: center; justify-content: center;">hücre1</div>	<pre><table border="1" cellspacing="5"> <tr> <td>hücre1</td> </tr> <tr> <td>hücre2</td> </tr> </table></pre>
<div style="border: 1px solid black; padding: 10px; width: 50px; height: 30px; display: flex; align-items: center; justify-content: center;">hücre1</div> <div style="border: 1px solid black; padding: 10px; width: 50px; height: 30px; display: flex; align-items: center; justify-content: center;">hücre1</div>	<pre><table border="1" cellspacing="10"> <tr> <td>hücre1</td> </tr> <tr> <td>hücre2</td> </tr> </table></pre>

- **width**: Tablonun genişlik değerini temsil eden parametredir.
- **height**: Tablonun yükseklik değerini temsil eden parametredir.

<div style="border: 1px solid black; padding: 5px; width: 50px; height: 30px; display: flex; align-items: center; justify-content: center;">hücre1</div> <div style="border: 1px solid black; padding: 5px; width: 50px; height: 30px; display: flex; align-items: center; justify-content: center;">hücre2</div>	<pre><table border="1" width=150 height=200> <tr> <td>hücre1</td> </tr> <tr> <td>hücre2</td> </tr> </table></pre>
<div style="border: 1px solid black; padding: 5px; width: 50px; height: 30px; display: flex; align-items: center; justify-content: center;">hücre1</div> <div style="border: 1px solid black; padding: 5px; width: 50px; height: 30px; display: flex; align-items: center; justify-content: center;">hücre2</div>	<pre><table border="1" width=80 height=80> <tr> <td>hücre1</td> </tr> <tr> <td>hücre2</td> </tr> </table></pre>

hücre1	hücre2	hücre3	hücre4
--------	--------	--------	--------

```

<table border="1" cellpadding="7">
<tr>
<td background="resim1.jpg">hücre1</td>
</tr>
<tr>
<td background="resim2.jpg">hücre2</td>
</tr>
<tr>
<td background="resim3.jpg">hücre3</td>
</tr>
<tr>
<td background="resim4.jpg">hücre4</td>
</tr>
</table>

```

hücre1
hücre2
hücre3
hücre4

```

<table border="1" cellpadding="7">
<tr><td width=120 height=20>hücre1</td></tr>
<tr><td width=120 height=40>hücre2</td></tr>
<tr><td width=120 height=60>hücre3</td></tr>
<tr><td width=100 height=80>hücre4</td></tr>
</table>

```

- **valign**: <td> tag'inin parametresidir. Hücrenin içerisindeki elemanın dikeydeki hizalanmasını temsil eder.
- **colspan**: Aynı satırdaki hücreleri birleştirmek için kullanılır. **rowspan**: Aynı sütundaki hücreleri birleştirmek için kullanılır. Birleştirilen hücreye ait <td></td> tag'i silinir.

A	B	C	D
E	F	G	H
I	J	K	L

```

<table border="1" cellpadding="12">
<tr><td>A</td><td>B</td><td>C</td><td>D</td></tr>
<tr><td>E</td><td>F</td><td>G</td><td>H</td></tr>
<tr><td>I</td><td>J</td><td>K</td><td>L</td></tr>
</table>

```

- A ve B hücrelerini birleştirmek için A hücrelerine ait <td> etiketine **colspan=2** parametresini ekliyoruz ve B hücrelerine ait <td></td> etiketini siliyoruz.
- E F ve G hücrelerini birleştirmek için E hücrelerine ait <td> etiketine **colspan=3** parametresini ekliyoruz ve F ve G hücrelerine ait <td></td> etiketlerini siliyoruz.

A	C	D	
E		H	
I	J	K	L

```
<table border="1" cellpadding="12">
<tr><td colspan="2">A</td><td>C</td><td>D</td></tr>
<tr><td colspan="3">E</td><td>H</td></tr>
<tr><td>I</td><td>J</td><td>K</td><td>L</td></tr>
</table>
```

- E ve I hücrelerini birleştirmek için E hücrelerine ait `<td>` etiketine `rowspan=2` parametresini ekliyoruz ve I hücrelerine ait `<td>` etiketini siliyoruz.
- C, G ve K hücrelerini birleştirmek için C hücrelerine ait `<td>` etiketine `rowspan=3` parametresini ekliyoruz ve G ve K hücrelerine ait `<td>` etiketlerini siliyoruz.

A	B	C	D
E	F		H
J	L		

```

<table border="1" cellpadding="12">
<tr><td>A</td><td>B</td>
<td rowspan="3">C</td><td>D</td></tr>
<tr><td rowspan="2">E</td><td>F</td><td>H</td></tr>
<tr><td>J</td><td>L</td></tr>
</table>

```

```

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

<table>
<tr>
<th>Firstname</th>
<th>Lastname</th>
</tr>
<tr>
<td>Peter</td>
<td>Griffin</td>
</tr>
<tr>
<td>Lois</td>
<td>Griffin</td>
</tr>
</table>

</body>

```

Firstname	Lastname
Peter	Griffin
Lois	Griffin

Yukarıdaki örnekte tablonun çift kenarlıklı olduğuna dikkat edin. Bunun nedeni, hem tablonun hem de `<th>` ve `<td>` öğelerinin ayrı kenarlıklara sahip olmasıdır. Çift kenarlıkları kaldırmak için aşağıdaki örneğe bakın.

border-collapse tek kenarlık elde ederiz.

```
<head>
<style>
table, td, th {
  border: 1px solid black;
}
table {
  width: 100%;
  border-collapse: collapse;
}
</style>
</head>
<body>

<table>
  <tr>
    <th>Firstname</th>
    <th>Lastname</th>
  </tr>
  <tr>
    <td>Peter</td>
    <td>Griffin</td>
  </tr>
</table>

</body>
```

Firstname	Lastname
Peter	Griffin

hover Fareyle üzerine gelindiğinde tablo satırlarını (<tr>) vurgulamak için kullanılır.

```
<head>
<style>
table {
  border-collapse: collapse;
  width: 100%;
}

th, td {
  padding: 8px;
  text-align: left;
  border-bottom: 1px solid #ddd;
}

tr:hover {background-color: yellow;}
</style>
</head>
<body>

<table>
  <tr>
    <th>First Name</th>
    <th>Last Name</th>
    <th>Points</th>
  </tr>
  <tr>
    <td>Peter</td>
    <td>Griffin</td>
    <td>$100</td>
  </tr>
</table>

</body>
```

First Name	Last Name	Points
Peter	Griffin	\$100

FAREYİ METNİN ÜZERİNE GETİRDİĞİMİZDE AÇIKLAMANIN UZUN HALİNİN GÖSTERİLMESİ

- Fareyi öğenin üzerine getirdiğinizde kısaltmanın açıklamasını göstermek için `<abbr...></abbr>` ya da `<acronym></acronym>` etiketini kullanırız.

```
3 <head>
4   <meta charset="utf-8">
5   <title></title>
6 </head>
7 <body>
8 <p><dfn><abbr title="Mehmet-Münevver Kurban Anadolu Lisesi">MMKAL</abbr></p>
9 </body>
```



PRE Etiketi

İçindeki boşlukların korunmasını gerektiren metinleri görüntülemek için kullanılır.

```
<PRE>
switch($i){
  case "2":
    echo "Değeri 2";
    break;
  case "3":
    echo "Değeri 3";
    break;
  default:
    echo "Değeri
bilinmiyor";
    break;
}
</PRE>
```

FRAME (ÇERÇEVE) OLUŞTURMA

- **Frame** bir web sayfasının içerisine farklı bir web sayfasını çağırıp, görüntülememize yardımcı olan bir HTML etiketidir.
- **<body>** etiketleri arasında yer alır.
- **Frame** etiketi ise büyüklükleri belirlenen bu sayfaları oluşturmak ve detaylandırmak için kullanılır. frame etiketinin kendine has parametreleri mevcuttur:
 - **src = «dizin/dosya_adi»** Pencere içerisine yüklenecek dosyayı seçmek için kullanılır. Bu değer sonucu içinde herhangi bir dosya olduğu gibi değişik bir Internet adresi de olabilir.
 - **name=«frame_adi»** Daha sonra sayfayı çağırarak için referans olarak kullanılacak isim bu değere atanır.
 - **marginwidth** ve **marginheight**: Frame içerisindeki ilk nesne (element)'nin sırasıyla soldan ve yukarıdan uzaklığını belirtir.
- **Frameset** etiketi ile **cols (dikey)** ve **rows (yatay)** bazlı siteler oluşturabiliriz.
 - **<frameset cols = "150, *">** ne anlama gelir?
Yan yana (dikey bazlı) 2 frame eklenecek, ilk frame 150 px genişliğinde iken geriye kalan genişlik 2.frame için kullanılacak.
 - **<frameset rows="30%, *">** ne anlama gelir?
Üst üste (satır bazlı) iki frame eklenecek, ilk frame sayfanın mevcut yüksekliğinin %30'unu kullanırken geri kalan yükseklik 2. frame için kullanılacak.
 - **<frameset cols="1*,250,3*">** ne anlama gelir?
Yan yana (sütun bazlı) 3 frame eklenecek, ortadaki (2.) frame 250 px genişliğinde olacak, kalan genişlik ise 1. ve 3. frame arasında bölüştürülecek ve 3. frame 1. frame'in 3 katı genişlikte olacak.
- **frameborder**: Frameler arasında kenarlık çizgilerinin olup olmayacağına karar veren parametredir. **1=Kenarlık var, 0=Kenarlık yok.**
- **border**: Frame'ler arasındaki çerçevenin kalınlığını belirleyen parametredir.
- **bordercolor**: Çerçeveler arasındaki kenarlık renklerini belirleyen parametredir.
- **noresize**: Framelerin ölçülerinin (genişlik-yükseklik) değiştirilmesini engeller.
- **scrolling**: Frameler için kaydırma çubuklarının olup olmayacağını belirleyen parametredir. **scrolling = auto|yes|no** değerlerini alabilir.



Sol Çerçeve

Sağ Çerçeve

Sol Alt
Çerçeve

Sağ Alt
Çerçeve

```
</head>
<frameset cols="50%,50%" rows="20%,80%">
  <frame name="bir" src="frame-1.html" scrolling="yes">
  <frame name="iki" src="frame-2.html" scrolling="no">
  <frame name="uc" src="frame-3.html">
  <frame name="dort" src="frame-4.html">
<noframes>Browser'ınız Frame'leri desteklemiyor. Lütfen yenileyin!</noframes>
</body>
```

Sol Çerçeve

Sağ Çerçeve

Sol Alt
Cerceve

Sağ Alt Çerçeve

```
</head>
<frameset rows="*, *, *" // 3 satır
  <frame name="a" src="frame-1.html" noresize // büyütüp küçültemeyiz.
  <frameset cols="*,*" // 2 sütun
    <frame name="b" src="frame-2.html" noresize // büyütüp küçültemeyiz.
    <frame name="c" src="frame-3.html">
  </frameset>
  <frame name="d" src="frame-4.html">
</frameset>
</body>
```


HTML FORMLARI [\(Geri dönmek için tıklayın.\)](#)

- **<form>** etiketi, kullanıcıdan aldığımız verileri işlemek üzere aynı sayfa üzerinde veya farklı sayfaya bu verileri POST ya da GET metotları ile göndermemize yarar.
- **method** özelliği form verileri gönderilirken hangi yönteminin (GET/POST) kullanılacağını tanımlar.
- Form elemanlarına girilen verilerin gönderileceği sayfayı **action** kısmından belirtiyoruz. Action kısmını boş bırakırsanız form gönderilme işlemi mevcut işlem yapılan sayfa üzerinde gerçekleşecektir.
- **<label>** etiketi, bir **<input>** elementi oluşturabilmemiz için kullandığımız bir ara etikettir. Kullanıcının **<label>** elementi içindeki yazıya tıklaması **<input>** öğesinin işlevini harekete geçirir ve fare kullanım kolaylığı sağlar.
<label> etiketinin **for** parametresi, onları birbirine bağlamak için **<input>** öğesinin **ID** parametresine eşit olmalıdır.

- **name=" "** input elemanının işlenmek üzere ismini belirler. Sayfada birden fazla input elemanı olacağı için form gönderme (POST/GET) işlemlerinde verilerin gönderilen tarafta işlenmesi için name alanında yer alan isimlendirme kullanılır. Name kısmına verilecek isimler programlamada değişken oluşturma kurallarına uymalıdır.
- **disabled value = " "** disabled, input form elemanını pasif hâle getirir. input elemanının rengi değişir ve içeriği değiştirilemez. Input elemanı bu özellik kullanıldığında devre dışı kalacağından içindeki veri POST veya GET işlemlerinde kullanılamaz.

value, input giriş elemanının varsayılan değerini belirtir.

```
<form action="/action_page.php">
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" value="John" disabled>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname" value="Doe">
  <input type="submit" value="Submit">
</form>
```



- **readonly**, input giriş elemanının sadece okuma işlevinde kullanılacağını belirtir. **disabled**'a benzer. İki özellikte de input elemanına veri girişi yapılamaz. Fakat disabled özelliği kullanıldığında input elemanın içindeki veri POST edilemese de readonly özelliği kullanıldığında veriler kullanılabilir ve POST edilebilir.
- **<select>** etiketi aşağıdaki gibi listeler oluşturmak için kullanılır.
- **<option>** etiketi **<select>** etiketi içerisinde öğeleri tanımlamak için kullanılır. Önceden seçilmiş bir seçeneği tanımlamak için seçeneğe **selected** özelliği kullanılır yani sayfa ilk açıldığında "Fiat" maddesi seçiliydi (2. fotoğraf).
- **size** maddelerin sayısını belirtmek için kullanılır. **multiple** kullanıcının birden fazla değer seçmesine izin verir.

```
<form action="/action_page.php">
  <label for="cars">Choose a car:</label>
  <select id="cars" name="cars" size="4" multiple>
    <option value="volvo">Volvo</option>
    <option value="saab">Saab</option>
    <option value="fiat">Fiat</option>
    <option value="audi">Audi</option>
  </select>
  <input type="submit">
</form>
```



```
<form action="/action_page.php">
  <label for="cars">Choose a car:</label>
  <select id="cars" name="cars">
    <option value="volvo">Volvo</option>
    <option value="saab">Saab</option>
    <option value="fiat" selected>Fiat</option>
    <option value="audi">Audi</option>
  </select>
  <input type="submit">
</form>
```



- **<datalist>** etiketi **<input>** etiketi için önceden tanımlanmış açılabilir liste tanımlar. **<datalist>** etiketi **otomatik tamamlama** özelliğini tanımlar. Kullanıcılar, verileri girerken önceden tanımlanmış seçeneklerin bir açılır listesini göreceklerdir. **<datalist>** etiketi **<input>** etiketine bağlanırken input etiketinin **list** özelliği kullanılır.

```

<form action="/action_page.php">
  <input list="browsers" name="browser">
  <datalist id="browsers">
    <option value="Internet Explorer">
    <option value="Firefox">
    <option value="Chrome">
    <option value="Opera">
  </datalist>
  <input type="submit">
</form>

```

- **<button>** elemanı tıklanabilir bir düğme tanımlar. **<button>** etiketi **içerisine resim veya metinler eklenebilir**, **<input>** etiketi ile oluşturulan butonlardan **en önemli farkı budur**. Her zaman **<button>** etiketinin **type** özelliğini tanımlayın. Farklı tarayıcılar varsayılan olarak başka bir değer atayabilir.

```

<button type="button" onclick="alert('Hello World!')">Click Me!</button>

```

- **<textarea>** etiketi, bir metin alanı tanımlar. **rows** satır sayısını, **cols** ise sütun sayısını belirtir. İstersek **style** etiketini kullanarak metin alanının boyutunu **CSS** kullanarak da tanımlayabiliriz.

```

<form action="/action_page.php">
  <textarea name="message" rows="10" cols="30">C++. </textarea>
  <input type="submit">
</form>

```

```

<form action="/action_page.php">
  <textarea name="message" style="width:200px; height:600px;">The cat was playing in the garden.
</textarea>
  <input type="submit">
</form>

```

<input type="text">

Tek satırlık bir veri giriş alanı tanımlar.

<input type="password">

Bir şifre alanı tanımlar.

```

<form action="/action_page.php">
  <label for="username">Username:</label>
  <input type="text" id="username" name="username">
  <label for="pwd">Password:</label><br>
  <input type="password" id="pwd" name="pwd">
  <input type="submit" value="Submit">
</form>

```

`<input type="submit">`

Gönder düğmesini görüntüler.(Submit)

<pre><form action="/action_page.php"> <label for="fname">First name:</label>
 <input type="text" id="fname" value="Hasan">

 <input type="submit" value="Submit"> </form></pre>	<p>First name:</p> <input type="text" value="Hasan"/> <input type="submit" value="Submit"/>
--	--

`<input type="radio">`

Kullanıcınıza iki veya daha fazla seçenek arasında bir seçim yaptırmak istediğinizde radyo düğmeleri kullanılmalıdır. Bir radyo düğmesi sizi yalnızca birini seçmeye zorlar.

<pre><form> <input type="radio" id="html" name="fav_language" value="HTML"> <label for="html">HTML</label>
 <input type="radio" id="css" name="fav_language" value="CSS"> <label for="css">CSS</label>
 <input type="radio" id="javascript" name="fav_language" value="JavaScript"> <label for="javascript">JavaScript</label> </form></pre>	<p><input type="radio"/> HTML <input checked="" type="radio"/> CSS <input type="radio"/> JavaScript</p>
--	---

`<input type="checkbox">`

Seçilmesi istenen nesneler seçilir.

<pre><form action="/action_page.php"> <input type="checkbox" id="vehicle1" name="vehicle1" value="Bike"> <label for="vehicle1"> I have a bike</label>
 <input type="checkbox" id="vehicle2" name="vehicle2" value="Car"> <label for="vehicle2"> I have a car</label>
 <input type="checkbox" id="vehicle3" name="vehicle3" value="Boat"> <label for="vehicle3"> I have a boat</label>

 <input type="submit" value="Submit"> </form></pre>	<p><input checked="" type="checkbox"/> I have a bike <input checked="" type="checkbox"/> I have a car <input checked="" type="checkbox"/> I have a boat <input type="submit" value="Submit"/></p>
--	---

`<input type="reset">`

Tüm form değerlerini varsayılan değerlerine sıfırlayacak bir sıfırlama düğmesi tanımlar.

<pre><form action="/action_page.php"> <label for="fname">First name:</label> <input type="text" id="fname" name="fname" value="John"> <label for="lname">Last name:</label>
 <input type="text" id="lname" name="lname" value="Doe"> <input type="submit" value="Submit"> <input type="reset"> </form></pre>	<p>First name: <input type="text" value="John"/> Last name: <input type="text" value="Doe"/> <input type="submit" value="Submit"/> <input type="reset" value="Sıfırla"/></p>
---	--

`<input type="email">`

Bir e-posta adresini içermelidir.

<pre><h2>Email Field</h2> <form action="/action_page.php"> <label for="email">Enter your email:</label> <input type="email" id="email" name="email"> <input type="submit" value="Submit"> </form></pre>	<h3>Email Field</h3> <p>Enter your email: <input type="text" value="hassan42-42@hotmail.com"/> <input type="submit" value="Submit"/></p>
---	--

`<input type="file">`

Dosya yüklemeleri için bir düğme tanımlar.

<pre><h1>File upload</h1> <form action="/action_page.php"> <label for="myfile">Select a file:</label> <input type="file" id="myfile" name="myfile"> <input type="submit" value="Submit"> </form></pre>	<h3>File upload</h3> <p>Select a file: <input type="button" value="Dosya Seç"/> Dosya seçilmedi</p> <p><input type="submit" value="Submit"/></p>
--	--

`<input type="date">`

Tarih giriş alanları için kullanılmaktadır. Tarihlerle kısıtlamalar eklemek için `min` ve `max` kullanılır.

<pre><h2>Date Field Restrictions</h2> <form action="/action_page.php"> <label for="datemin">Enter a date after 2000-01-01:</label> <input type="date" id="datemin" name="datemin" min="2000-01-02"> <label for="datemax">Enter a date before 1980-01-01:</label> <input type="date" id="datemax" name="datemax" max="1979-12-31"> <input type="submit" value="Submit"> </form></pre>	<h3>Date Field Restrictions</h3> <p>Enter a date after 2000-01-01: <input type="text" value="gg.aa.yyyy"/> <input type="button" value="📅"/></p> <p>Enter a date before 1980-01-01: <input type="text" value="gg.aa.yyyy"/> <input type="button" value="📅"/></p> <p><input type="submit" value="Submit"/></p>
--	--

`<input type="search">`

Arama için metin alanı tanımlar.

<pre><h2>Search Field</h2> <form action="/action_page.php"> <label for="gsearch">Search Google:</label> <input type="search" id="gsearch" name="gsearch"> <input type="submit" value="Submit"> </form></pre>	<h3>Search Field</h3> <p>Search Google: <input type="text"/> <input type="submit" value="Submit"/></p>
--	--

`<input type="tel">`

Bir telefon numarası içermelidir.

<pre><form action="/action_page.php"> <label for="phone">Enter a phone number:</label> <input type="tel" id="phone" name="phone" placeholder="123-45-678" pattern="[0-9]{3}-[0-9]{2}-[0-9]{3}" required> <small>Format: 123-45-678</small> <input type="submit" value="Submit"> </form></pre>	<p>Enter a phone number:</p> <p><input type="text" value="123-45-678"/></p> <p>Format: 123-45-678</p> <p><input type="submit" value="Submit"/></p>
---	--

`<input type="number">`

Sayı girilmesi için bir alan tanımlar. Aşağıdaki örnekte, 1'den 5'e kadar bir değer girebiliriz.

<pre><h2>Number Field</h2> <form action="/action_page.php"> <label for="quantity">Quantity (between 1 and 5):</label> <input type="number" id="quantity" name="quantity" min="1" max="5"> <input type="submit" value="Submit"> </form></pre>	Number Field Quantity (between 1 and 5): <input type="text" value="3"/> <input type="submit" value="Submit"/>
--	---

`<input maxlength="4">`

Giriş alanının alabileceği max karakter sayısını belirtir.

`<input min="4">`

Giriş alanının alabileceği minimum değeri belirtir.

<pre><form action="/action_page.php"> <label for="fname">First name:</label> <input type="text" id="fname" name="fname" size="5"> <label for="pin">PIN:</label> <input type="text" id="pin" name="pin" maxlength="4" size="4"> <input type="submit" value="Submit"> </form></pre>	First name: <input type="text"/> PIN: <input type="text"/> <input type="submit" value="Submit"/>
---	--

`<input pattern="[A-Za-z]{3}">`

Giriş alanına girilen bilgilerin doğruluğunu kontrol eder.

<pre><form action="/action_page.php"> <label for="country_code">Country code:</label> <input type="text" id="country_code" name="country_code" pattern="[A-Za-z]{3}" title="Three letter country code"> <input type="submit" value="Submit"> </form></pre>	Country code: <input type="text" value="l"/> <input type="submit" value="Submit"/>  Lütfen istenen biçimi eşleştirin. Three letter country code
--	--

`<input placeholder="123-45-678">`

Giriş alanına girilecek değer hakkında ipuçları belirtir.

<pre><form action="/action_page.php"> <label for="phone">Enter a phone number:</label> <input type="tel" id="phone" name="phone" placeholder="123- 45-678" pattern="[0-9]{3}-[0-9]{2}-[0-9]{3}"> <input type="submit" value="Submit"> </form></pre>	Enter a phone number: <input type="text" value="123-45-678"/> <input type="submit" value="Submit"/>
---	--

`<input step="3">`

Giriş alanı için artış eksiliş sayısı belirtir.

<pre><form action="/action_page.php"> <label for="points">Points:</label> <input type="number" id="points" name="points" step="3"> <input type="submit" value="Submit"> </form></pre>	Points: <input type="text" value="9"/> <input type="submit" value="Submit"/>
---	--

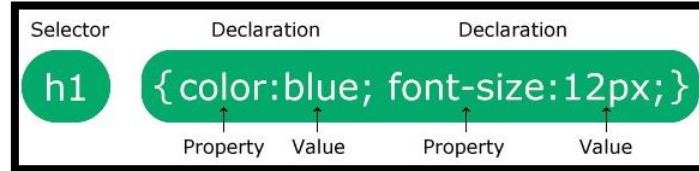
`<input autofocus>`

Nesneye otomatik olarak odaklanılacağını belirtir.

<pre><form action="/action_page.php"> <label for="fname">First name:</label> <input type="text" id="fname" name="fname" autofocus> <label for="lname">Last name:</label> <input type="text" id="lname" name="lname"> <input type="submit" value="Submit"> </form></pre>	First name: <input type="text"/> Last name: <input type="text"/> <input type="submit" value="Submit"/>
---	--

CSS (CASCADING STYLE SHEETS) TEMEL KOD YAPISI

- Sayfanın `<head>...</head>` etiketleri arasında kullanılır.
- CSS dosyasında yorum satırı `/* */` arasına yazılır.
- Her bir stil özelliğine değer verirken iki nokta üst üse koyarız.
Birden çok CSS özelliği noktalı virgülle ayrılır ve son olarak küme paranteziyle çevrilir.



CSS NASIL EKLENİR?

Satır İçi (Yerel) CSS

```
<!DOCTYPE html>
<html>
<body>

<h1 style="color:blue;text-align:left;">Heading</h1>
<p style="color:red;">Paragraph</p>

</body>
</html>
```

Heading

Paragraph

Dahili (Global) CSS

Dahili CSS ile `<head><head>` etiketleri arasına yazacağımız `<style><style>` etiketi sayesinde stil grubu oluşturup tek tek özellik verme zahmetinden kurtuluruz.

Aşağıdaki örnekte `p` etiketleri arasına yazılanlar turuncu olarak gözükecek ve bu durumdan bütün site etkilenecektir.

Bu dahili CSS.

Bu bir paragraftır.

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4
5 <style>
6   body{
7     background-color: black;}
8   h1{
9     color: red;
10    margin-left: 20px;}
11   p{
12     color: orange;}
13 </style>
14
15 </head>
16 <body>
17 <h1>Bu dahili CSS.</h1>
18 <p style="font-size: 10px; color: blue;">Bu bir paragraftır.</p>
19 </body>
20 </html>
```

Harici (Bağıntılı) CSS

- `<link rel="stylesheet" type="text/CSS" href="DERS7.css">` komutunu sayfamızda `<head>` etiketi içerisinde yazıyoruz.

```
DER57.html
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="utf-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1">
6   <link rel="stylesheet" type="text/CSS" href="DERS7.css">
7   <title>Harici CSS</title>
8 </head>
9 <body>
10  <h1>This is a heading</h1>
11  <p>This is a paragraph.</p>
12 </body>
13 </html>

DER57.css
1 body {
2   background-color: powderblue;
3 }
4 h1 {
5   font-family: inherit;
6   color: blue;
7   font-size: 30px;
8 }
9 p {
10  font-weight: bolder;
11  color: red;
12 }
```



NOT: Bir HTML ögesi için birden fazla stil belirtildiğinde hangi stil kullanılır?

Böyle bir durumda satır içi stil en yüksek önceliğe sahiptir. Harici ve dahili sitlerin hangisi en son olarak tanımlanmışsa o geçerli olacaktır.

NOT: Yanlış -----> `margin-left: 20 px;` Doğru -----> `margin-left: 20px;`

ID (KİMLİK) SEÇİCİ

- CSS'te ID'ler bir sayfada yalnızca bir kez kullanılırlar. Bir HTML belgesinde aynı ID'ye sahip birden fazla öğeniz olamaz. ID kullanılırken seçiciden önce (#) işareti kullanılır.

```
id(Kimlik) Oluşturma.html
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="utf-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1">
6   <title>ID (KİMLİK)</title>
7   <style type="text/css">
8     #ozellik1{
9       color: red;
10      font-size: 20px;
11    }
12  </style>
13 </head>
14 <body>
15  <p id="ozellik1">PARAGRAF-1</p>
16  <h1 id="ozellik1">BASLIK-1</h1>
17  <p>PARAGRAF-2</p>
18 </body>
19 </html>
```

PARAGRAF-1

BASLIK-1

PARAGRAF-2

CLASS SEÇİCİ

- Aynı sayfada birden fazla HTML etiketine aynı CSS özelliklerini vermek için kullanılır. Belirli bir sınıfa sahip öğeleri seçmek için, **nokta (.) karakteri ve ardından sınıf adı yazılır.**
- **Stil belirlenirken . (nokta) ile başlar ve kullanıcının vermek istediği isimle devam eder.**

BASLIK-1

PARAGRAF-1

PARAGRAF-2

```
Class Oluşturma.html
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="utf-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1">
6      <title>CLASS</title>
7      <style type="text/css">
8          p.ozellik1{
9              text-align: left;
10             color: red;
11             font-size: 20px;
12         }
13         p.ozellik2{
14             color: limegreen;
15             font-family: sans-serif;
16         }
17     </style>
18 </head>
19 <body>
20     <h1 class="ozellik1" >BASLIK-1</h1>
21     <p class="ozellik1">PARAGRAF-1</p>
22     <p class="ozellik2">PARAGRAF-2</p>
23 </body>
24 </html>
```

BASLIK-1

PARAGRAF-1

```
Class Oluşturma-2.html
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="utf-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1">
6      <title>CLASS</title>
7      <style type="text/css">
8          .ozellik1{
9              text-align: left;
10             color: red;
11             font-size: 20px;
12         }
13     </style>
14 </head>
15 <body>
16     <div class="ozellik1">
17         <h1>BASLIK-1</h1>
18         <p>PARAGRAF-1</p>
19     </div>
20 </body>
21 </html>
```

SINIFIMSI SEÇİCİLER

Sınıfımsı seçiciler yalnızca **bağlantı (<a>)** ve **paragraf (<p>)** etiketlerine uygulanır.

a:link { ziyaret edilmemiş bir bağlantının rengini söyler.

```
color: red;
```

}

a:visited { kullanıcı'nın linke tıkladığındaki rengi söyler.

```
color: green;
```

}

a: hover { kullanıcı fareyi üzerine getirdiğindeki rengi söyler.

```
color: hotpink;
```

}

a:active { tıklıldığı andaki bağlantının rengini söyler.

```
color: blue;
```

}

NOT: hover, link ve visited'den sonra gelmelidir. Active ise hover'dan sonra gelmelidir.

Sınıfımsı Seçiciler.html

```
1 <!DOCTYPE html>
```

```
2 <html>
```

```
3 <head>
4   <meta charset="utf-8">
```

```
6      <title>SINIMSI SEÇİCİLER</title>
```

```
7 <style type="text/css">
8     a:link {
```

```
8     a.link {
9         color: black;
```

10 }

```
11     a:active {
```

```
11     a.active {
12         color: greenyellow;
```

13 }

```
14     a:visited { 1PARAGRAF-1PARAGRAF-1PARAGRAF-1PARAGRAF-1PARAGR
15         color: red; 1PARAGRAF-1PARAGRAF-1PARAGRAF-1
```

```
16 }
17
```

```
17         a:hover {
18             color: blue;
```

19 }

```
20
21     </style>
```

```
22 <a href="http://www.erbakan.edu.tr">NEU</a> <br>
```

```
23 Paragrafın ilk satırı <br>
```

```
24 Paragrafın ilk harfi. <br>
25 <style type="text/css">
```

```
25 <style type= "text/css" >
26     p:first-line {
```

```
font-size: 12px
```

28 }

```
29     p:first-letter {
30         font-size: 24px
```

31 }

```
32     </style>
33     <p>PARAGRAPH-1PARAGRAPH-1PARAGRAPH-1PARAGRAPH-1PARAGRAPH-1PARAGRAPH-1PARAGRAPH-1PAR
```

AGRAF-1PARAGRAF-1PARAGRAF-1PARAGRAF-1PARAGRAF-1PARAGRAF-1PARAGRAF-1</p>

34 <p>PARAGRAF-2</p>

```
35     </head>
```

```
36     <body>
37
```

```
37  
38 </body>
```

```
39 </html>
```


CSS UNIVERSAL SEÇİCİ

Evrensel seçici, (*) sayfadaki tüm HTML öğelerini seçer.



```
<!DOCTYPE html>
<html>
<head>
<style>
* {
  text-align: left;
  color: blue;
}
</style>
</head>
<body>

<h1>Hello world!</h1>

<p>Every element on the page.</p>
<p id="para1">Me too!</p>
<p>And me!</p>

</body>
</html>
```

CSS GROUPING SELECTOR (GRUP SEÇİCİ)

Aynı stil tanımlarına sahip tüm HTML öğelerini seçer. Her seçiciyi virgülle ayırırız.



```
<!DOCTYPE html>
<html>
<head>
<style>
h1, h2, p {
  text-align: left;
  color: red;
}
</style>
</head>
<body>

<h1>Hello World!</h1>
<h2>Smaller heading!</h2>
<p>This is a paragraph.</p>

</body>
</html>
```

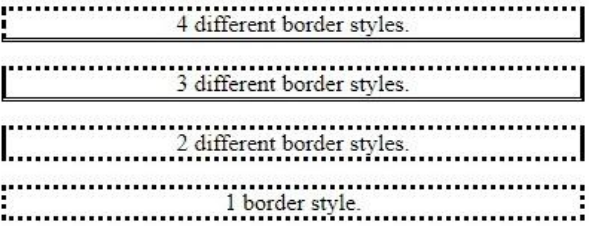
KENARLIK

border etiketini kullanarak yazılarımıza kenarlık (çerçeve) yapabiliriz.

Eğer border-style dört değer içeriyorsa;	üst kenarlık, sağ kenarlık, alt kenarlık, sol kenarlık.
Eğer border-style üç değer içeriyorsa;	üst kenarlık, alt kenarlık, sağ ve sol kenarlık.
Eğer border-style iki değer içeriyorsa;	üst ve alt kenarlık, sağ ve sol kenarlık.
Eğer border-style tek değer içeriyorsa;	bütün kenarlıklar.

```
<head>
<style>
body {
  text-align: center;
}
p.four {
  border-style: dotted solid double dashed;
}
p.three {
  border-style: dotted solid double;
}
p.two {
  border-style: dotted solid;
}
p.one {
  border-style: dotted;
}
</style>
</head>


<body>
<p class="four">4 different border styles.</p>
<p class="three">3 different border styles.</p>
<p class="two">2 different border styles.</p>
<p class="one">1 border style.</p>
</body>
```



border-radius: 12px komutu sayesinde kenarlığın sivriliği ayarlayabiliriz.

```
<head>
<style>
p.round1 {
  border: 2px solid red;
  border-radius: 12px;
  padding: 5px;
}
</style>
</head>

<body>
<p class="round1">Border</p>
</body>
```



CSS MARGIN and PADDING

Margin etiketi kenarlık (border) ile web sayfasının arasındaki boşluğu ayarlar.

margin-top -----> Üstten boşluk	margin-right -----> Sağdan boşluk
margin-bottom -----> Alttan boşluk	margin-left -----> Soldan boşluk
auto -----> Kenarlıklar tarayıcı tarafından otomatik ayarlanır.	
% -----> İçerik genişliğine göre yüzdeli olara boşluk belirtme.	
Bir resmi ortalamak için margin-left ve margin-right komutlarını auto özelliğiyle kullanınız.	

Margin üstten başlar daha sonra sağ yönünde ilerler. (Üst, Sağ, Alt ve Sol)

Eğer padding ve margin 'in 4 değeri varsa; Kenarlığın üst kenar boşluğu 25 pikseldir. Kenarlığın sağ kenar boşluğu 50 pikseldir. Kenarlığın alt kenar boşluğu 75 pikseldir. Kenarlığın sol kenar boşluğu 100 pikseldir.	Eğer padding ve margin 'in tek değeri varsa; Dört kenarın boşluğu eşittir.
---	---

Padding etiketi kenarlık (border) ile kenarlığın içine yazılan yazının arasındaki boşluğu ayarlar.

padding-top -----> Üstten boşluk	padding-right -----> Sağdan boşluk
padding-bottom -----> Alttan boşluk	padding-left -----> Soldan boşluk
auto -----> Kenarlıklar tarayıcı tarafından otomatik ayarlanır.	
% -----> İçerik genişliğine göre yüzdeli olara boşluk belirtme.	

```
3 <head>
4   <meta charset="utf-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1">
6   <title>MARGIN-PADDING</title>
7   <style type="text/css">
8     #div1{
9       border: 1px solid red;
10      margin: 25px 50px 75px 100px;
11      background: whitesmoke;
12    }
13    #div2{
14      border: 1px solid blue;
15      padding: 50px 30px 55px 90px;
16      background: whitesmoke;
17    }
18  </style>
19 </head>
20 <body>
21   <div id="div1">BURASI MARGIN ETİKETİYLE HAZIRLANMIŞ BİR YAPIDIR.
22     BURASI MARGIN ETİKETİYLE HAZIRLANMIŞ BİR YAPIDIR.
23     BURASI MARGIN ETİKETİYLE HAZIRLANMIŞ BİR YAPIDIR.</div>
24
25   <div id="div2">BURASI PADDING ETİKETİYLE OLUSTURULMUŞ BİR YAPIDIR.
26     BURASI PADDING ETİKETİYLE OLUSTURULMUŞ BİR YAPIDIR.
27     BURASI PADDING ETİKETİYLE OLUSTURULMUŞ BİR YAPIDIR.</div>
28 </body>
```

BURASI MARGIN ETİKETİYLE HAZIRLANMIŞ BİR YAPIDIR. BURASI
MARGIN ETİKETİYLE HAZIRLANMIŞ BİR YAPIDIR. BURASI
MARGIN ETİKETİYLE HAZIRLANMIŞ BİR YAPIDIR.

BURASI PADDING ETİKETİYLE OLUSTURULMUŞ BİR YAPIDIR. BURASI
PADDING ETİKETİYLE OLUSTURULMUŞ BİR YAPIDIR. BURASI
PADDING ETİKETİYLE OLUSTURULMUŞ BİR YAPIDIR.

CSS DÜZENİ - KONUM

position:fixed; konumun sabit kalmasını, **position: absolute;** konumun koordinat girilerek belirlenmesini sağlar.

top: Nesnenin sayfanın üst kenarından uzaklığı

left: Nesnenin sayfanın sol kenarından uzaklığı

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="utf-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1">
6      <title>KONUM</title>
7      <style type="text/css">
8          div.sabit{
9              position: fixed; /*sayfa kaydırılsa bile her zaman aynı yerde
10                 kalır. */
11              top: 50px;
12              left: 50px;
13              width: 300px;
14              border: 3px solid black;
15              background: red;
16          }
17          .sagaYaz{
18              position: absolute; /* Koordinat belirleyerek sayfada bir yere
19                 yerleştirir.*/
20              right: 50px;
21              font-size: 20px;
22              background: greenyellow;
23          }
24      </style>
25  </head>
26  <body>
27      <div class="sabit">
28          <p>PARAGRAF-1</p>
29      </div>
30      <div class="sagaYaz">
31          <p>PARAGRAF-2</p>
32      </div>
33  </body>
34  </html>
```

PARAGRAF-1

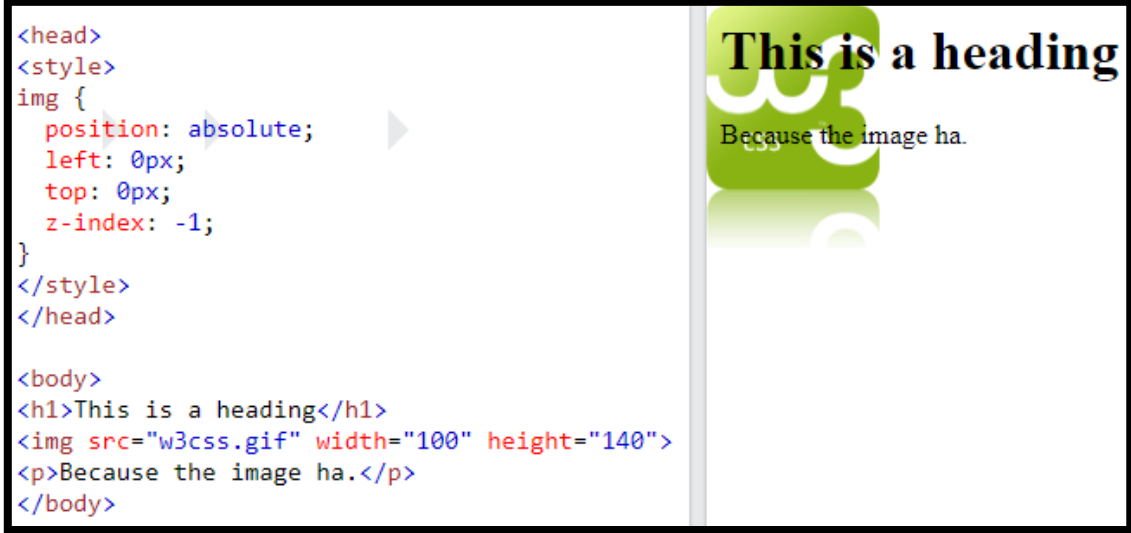
PARAGRAF-2

CSS DÜZENİ - Z ENDEKSİ

Sayfanızın içinde etiketlerin önceliği durumunda arkada kalan veya bir objenin diğer üstüne çıkması durumunda öne alma işlemi için kullanılmaktadır. Örneğin;

- Fotoğrafın bir Div'in arkasında olması.
- Bir butonun herhangi bir objenin arkasında olması.
- Objeleri konumlandırma işlemleri.

Aşağıdaki örnekte, fotoğraf yazıların arkasında kalmıştır.



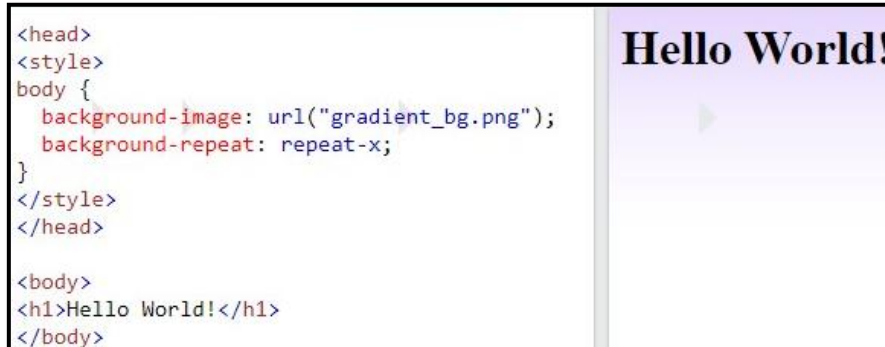
VISIBILITY

Nesnenin görünür veya gizli durumda bulunacağını bildirir. Nesneyi görünür ve gizli yapmak için sırayla visible ve hidden etiketleri kullanılır.



CSS BACKGROUND IMAGE REPEAT

- Bir görüntüyü dikey olarak tekrarlamak için **background-repeat: repeat-y;** yatay olarak tekrarlamak içinse **background-repeat: repeat-x;** komutu kullanılır.

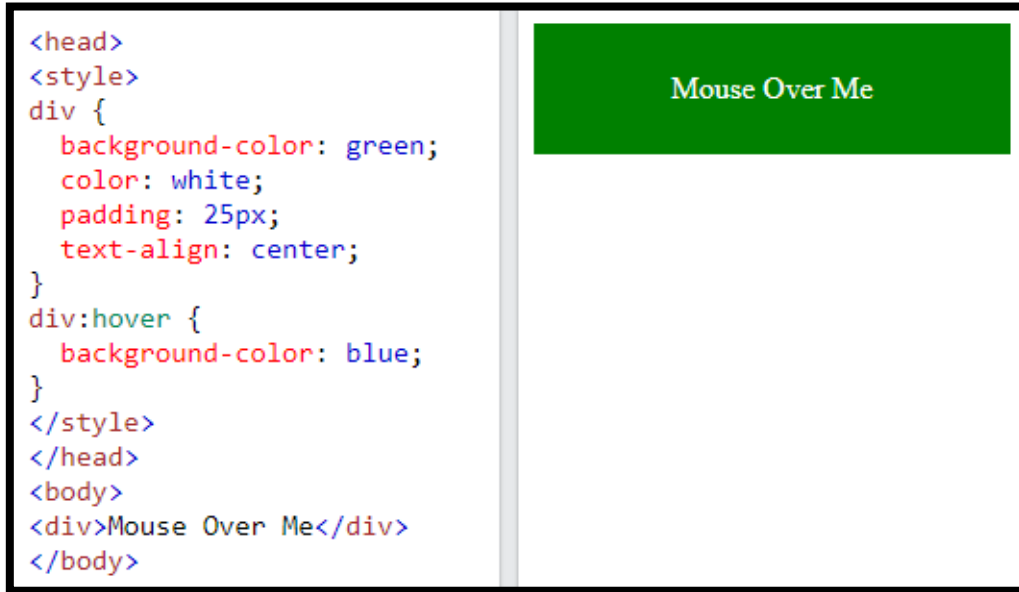


- Arka plan görüntüsü bir kez gösterilecekse **background-repeat: no-repeat;** komutunu kullanılır.
- Arka plan görüntüsünün konumunu belirtmek için **background-position** komutu kullanılır.
- Sayfayı aşağı çektikçe resminde aşağıya inmesini istiyorsak; **background-attachment: fixed;** komutunu kullanırız. Eğer biz aşağı indikçe resim yukarıda kalacaksa **background-attachment: scroll;** komutunu kullanırız.



CSS SÖZDE SINIFLAR

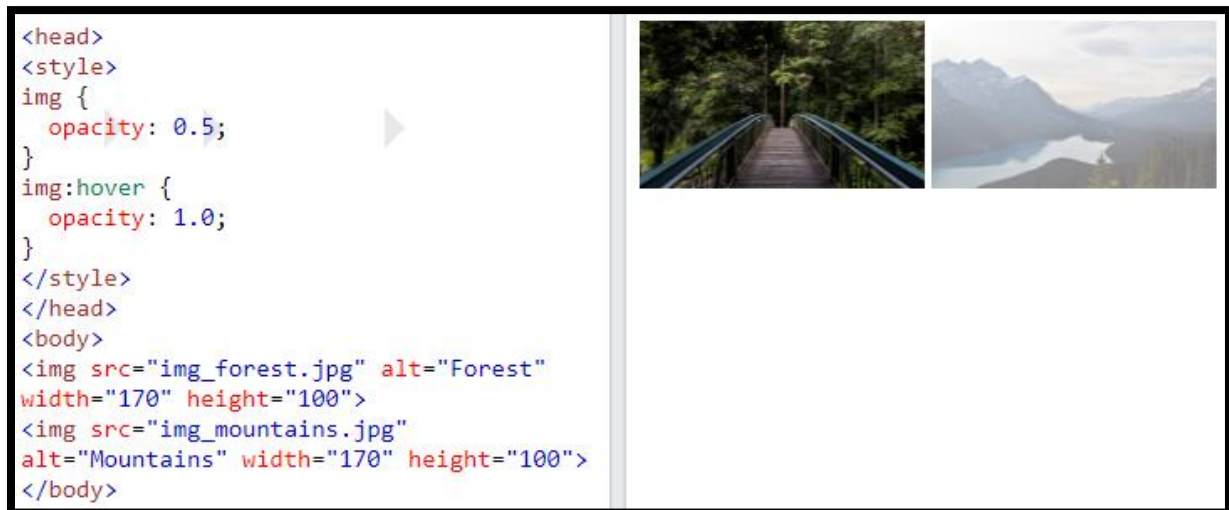
Aşağıdaki örnekte, imleci cümlemin üstüne getirdiğimizde arka plan mavimsi olacak.



CSS OPAKLIĞI

opacity özelliği 0-1 arasında bir değer alabilir. Değer ne kadar düşükse, o kadar şeffaftır.

Aşağıdaki örnekte, imleci hangi resmin üzerine getirirsek o resmin renkleri daha belirgin hale gelir. Bu örnekte imleci soldaki resmin üstüne getirdik.



CSS NAVIGATION BAR - DİKEY

Temel bir dikey gezinme çubuğu oluşturun ve kullanıcı fareyi üzerlerine getirdiğinde bağlantıların arka plan rengini maviyle değiştirin.

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4  <style>
5      ul {
6          background-color: whitesmoke;
7          border: 1px solid black;
8          margin: 0;
9          padding: 0;
10         width: 200px;
11     }
12
13     li a{
14         display: block;
15         color: black;
16         padding: 8px 16px;
17         text-decoration: none;
18     }
19
20     li{
21         text-align: center;
22         border-bottom: 1px solid black;
23     }
24
25     li a.active{
26         background-color: seagreen;
27         color: white;
28     }
29
30     li a:hover:not(.active) {
31         background-color: blue;
32         color: white;
33     }
34 </style>
35 </head>
36 <body>
37     <h2>Vertical Navigation Bar</h2>
38     <ul>
39         <li><a class="active" href="#home">Home</a></li>
40         <li><a href="#news">News</a></li>
41         <li><a href="#contact">Contact</a></li>
42         <li><a href="#about">About</a></li>
43     </ul>
44 </body>
```

Vertical Navigation Bar

Home
News
Contact
About

CSS NAVIGATION BAR - YATAY

li öğelerinin listenin dışına çıkmasını önlemek için **ul** öğesine **overflow:hidden** ekledik.

Liste öğelerini sağa kaydırarak bağlantıları sağa hizalayın (**float:right**):

border-right bağlantı bölücüler oluşturmak için bu özelliği **** öğesine ekledik.

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <style>
5          ul{
6              list-style-type: none;
7              background-color: black;
8              margin: 0;
9              padding: 0;
10             overflow: hidden;
11         }
12         li{
13             float: left;
14             border-right:1px solid red;
15         }
16         li a{
17             display: block;
18             color: white;
19             text-align: center;
20             padding: 14px 16px;
21             text-decoration: none;
22         }
23         li a:hover:not(.active){
24             background-color: blue;
25         }
26         .active {
27             background-color: green;
28         }
29     </style>
30 </head>
31 <body>
32     <ul>
33         <li><a class="active" href="#home">Home</a></li>
34         <li><a href="#news">News</a></li>
35         <li><a href="#contact">Contact</a></li>
36         <li style="float:right"><a href="#about">About</a></li>
37     </ul>
38 </body>
```

Home

News

Contact

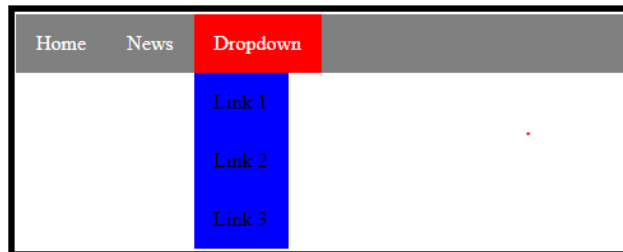
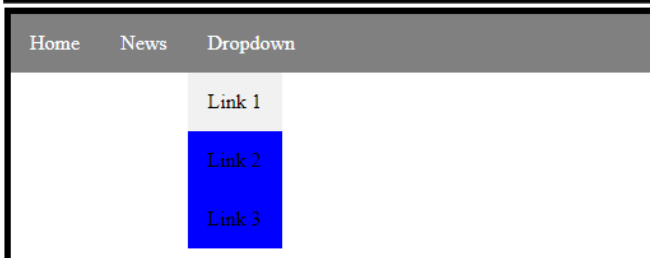
About

CSS DROPDOWNS (AÇILIR LİSTE)

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4  </head>
5  <link rel="stylesheet" type="text/css" href="styleasd.css">
6  <body>
7    <ul>
8      <li><a href="#home">Home</a></li>
9      <li><a href="#news">News</a></li>
10     <li class="dropdown">
11       <a href="javascript:void(0)" class="dropbtn">Dropdown</a>
12       <div class="dropdown-content">
13         <a href="#">Link 1</a>
14         <a href="#">Link 2</a>
15         <a href="#">Link 3</a>
16       </div>
17     </li>
18   </ul>
19 </body>
20 </html>

```



```

1  ul {
2    list-style-type: none;
3    margin: 0;
4    padding: 0;
5    overflow: hidden;
6    background-color: grey;
7  }
8
9  li {
10   float: left;
11 }
12
13 li a, .dropbtn {
14   display: inline-block;
15   color: white;
16   padding: 14px 16px;
17   text-decoration: none;
18 }
19
20 li a: hover {
21   background-color: red;
22 }
23
24 li.dropdown {
25   display: inline-block;
26 }
27
28 .dropdown-content {
29   display: none;
30   position: absolute;
31   background-color: blue;
32 }
33
34 .dropdown-content a {
35   color: black;
36   display: block;
37   text-align: left;
38 }
39
40 .dropdown-content a: hover { background-color: #f1f1f1; }
41
42 .dropdown: hover .dropdown-content {
43   display: block;
44 }
45

```

JAVASCRIPT

JavaScript ile HTML'in kaynak kodlarına müdahale edilebilir, sayfa post işlemi (form gönderme) yapılabilir, kullanıcıya mesaj verilebilir vb.

JavaScript kodları `<script>` `</script>` etiketleri arasına yazılır.

JavaScript ile web sayfamızdaki etiketlerin ve form elemanlarının içeriklerine/değerlerine ulaşabilir, bunlarla işlem yapabilir ve değiştirebiliriz.

Form elemanlarına ve etiketlerine ulaşmak için metotlar kullanılır.

1. `getElementById`

Web sayfasındaki html etiketlerine ulaşmak için en çok kullanılan metotlardan biridir.

Ulaşılmak istenen html nesnesinin id'si belirtilerek, ilgili elemana ulaşmamızı sağlar.

- `var deger = document.getElementById(«txtName»);`value;
- **txtName** ulaşılmaya çalışılan elementin ID değeridir.

2. `innerHTML`

Web sayfasındaki html etiketlerinin içeriğine ulaşmak veya bu içeriği değiştirmek için kullanılan metotlardan biridir.

Ulaşılmak istenen html nesnesinin id'si belirtilerek, ilgili elemana ulaşmamızı sağlar.

- `<p id=«mesaj»> Kayıt için ilgili alanları doldurunuz. </p>`
- `document.getElementById(«mesaj»).innerHTML = «Kayıt işlemi başarı ile yapıldı»;`

3. `getElementsByClassName`

Id özelliği her etiket için farklı olması gereken bir özelliktir (**Hatırla!**)

Bu durumda `getElementById` metodu ile aynı anda birden fazla elemana ulaşamayız.

Aynı Class özelliği sayfada bir çok etikete uygulanmış olabilir. Sayfada aynı class'ın uygulandığı tüm elemanlar ile işlem yapmak istersek `getElementsByClassName` metodunu kullanabiliriz.

`document.getElementsByClassName(«link»);` //bu bize link class'ı kullanan tüm elementleri verir

`document.getElementsByClassName(«link»)[0];` //bu ise dizi şeklinde gelen elementlerin ilkinin verir.

`document.getElementsByClassName(«link»)[i].style.color = «red»;` // **Ne yapar?**

4. `getElementsByName`

Class adı ile çağırma yöntemine benzer şekilde kullanılır.

Elementlere name özelliği üzerinden erişir.

```
document.getElementsByName(«link»);  
document.getElementsByName(«link»)[0];
```

Not: Eğer sayfada form kullanıldı ise ve bu form içerisindeki elemanlara ulaşmaya çalışılırsa:

`document.formAdi.elementAdi;`

şeklinde kullanılmalıdır. **ElementAdi** burada **name** özelliğinin değeridir.

PHP (HYPERTEXT PREPROCESSOR)

HTML kodları içerisinde dilediğiniz zaman PHP kipine yer verebiliriz. PHP kodları `<body>` `</body>` etiketleri arasında yer almalıdır.

PHP'yi Javascript gibi kullanıcı tarafında çalışan dillerden ayıran, sunucu tarafında çalıştırılıyor olmasıdır. Bunu kendi sunucunuzda çalıştırırsanız, sitenize bağlanan kullanıcılar kodu göremeyecekler ve müdahale edemeyecekler, yalnızca sonucu görebileceklerdir.

PHP Neler Yapabilir?

Sunucu taraflı programlama. Bu PHP için en geleneksel ve en temel olan alandır. Sunucu taraflı programlama için üç şeye sahip olmanız gerekir. PHP çözümleyici (sunucu modülü), bir HTTP sunucusu ve bir tarayıcı.

Ekrana Çıktı Verme, Yorum Satırı, Birleştirme İşlemi ve Satır Atlama

`echo` ekrana çıktı vermemizi, `echo "
"` ise satır atlamamızı sağlayan fonksiyondur.

Tek satırlık ekran çıktılarında kodun sonuna ";" koymamıza gerek yok ancak birden fazla çıktı verecek olursak o zaman ";" ile birbirinden ayırdıklarını belirtmemiz gerekir.

Birleştirme "." operatörünü kullanarak string ifadelerde birleştirme işlemlerini yapabiliriz.

```
<?php echo "Ekrana Yazdir-1" ?> <br>
<?php echo "<h1> Baslik-1 </h1>"; echo "<br/>"; // Başlık
echo "<b> Mühendislik Fakultesi </b>"; # Kalın yazmamızı sağlar.
echo "<p> Bilgisayar Muhendisligi </p>"; // Paragraf yapmamızı sağlar.
/* Ancak birden fazla satırlık çıktıda ";" koyarak bunların birbirinden
ayrıldığını göstermeliyiz. */

echo "Merhaba hocam"." ben 123 numaralı"." ogrenciyim."; echo "<br>";
$ad = "Hasan";
$ad.= " CEYHAN";
echo $ad; echo "<br/>";
$sonuc=356;
echo "İşlem Sonucu: " . $sonuc;
?>
```

Ekrana Yazdir-1

Baslik-1

Mühendislik Fakultesi

Bilgisayar Muhendisligi

Merhaba hocam ben 123 numaralı ogrenciyim.

Hasan CEYHAN

İşlem Sonucu: 356

Hataları Kapatma `error_reporting(0);`

Çalışırken olası bir kod hatasında kullanıcı arayüzünde bu hataların gizlenmesini sağlayabiliriz.

Aşağıda yer alan kodda, header.php isimli dosya dâhil edilmeye çalışılıyor. Fakat böyle bir dosya olmadığı için hata ile karşılaşılacaktır. `error_reporint(0);` kodu sayesinde bu hata gizlenebilir.

```
<?php
    error_reporting(0);
    include 'header.php';
?>
```

Değişkenler ve Operatörler

Değişken tanımlarken **\$** işaretini kullanırız ve değişkenler **\$** işareti ile başlar.

Aşağıdaki koda göz attığımızda **\$soyisim** değişkenine "ATA" atamasını gerçekleştirdik. Daha sonra **\$soyisim** değişkenine "CEYHAN" atamasını yaptık ve tekrar ekrana yazdırma işlemini gerçekleştirdik. Değişken adları aynı olmasına rağmen çıktıları farklı. Çünkü **en son aktarılan değer geçerli olur**.

```
<?php
    $isim = "Beyza"; echo $isim; echo "<br>";
    $soyisim = "ATA"; echo $soyisim; echo "<br>";
    $soyisim = "CEYHAN"; echo $soyisim;
?>
```

Beyza
ATA
CEYHAN

PHP’de değişkenlerin türleri, kullanıldıkları duruma ve aldıkları değerlere göre PHP yorumlayıcısı tarafından belirlenir.

```
<?php
    $sayi1 = "10"; echo "<br>";
    $sayi2 = 20; echo "<br>";
    $sayi3 = $sayi1+$sayi2; echo $sayi3;
    # CEVAP
    # 30
?>
```

Değişken Türünü Öğrenme

`var_dump()` fonksiyonu ile değişkenlerin türünü ve içeriğini öğrenebiliriz.

```
<?php
    $degisken1 = TRUE;
    $degisken2 = 1234;
    $degisken3 = "Hasan CEYHAN";
    $degisken4 = [ "1" => "a",
                  "2" => "b", ];
    $degisken5=1.234;
    var_dump($degisken1); echo"<br>";
    var_dump($degisken2); echo"<br>";
    var_dump($degisken3); echo"<br>";
    var_dump($degisken4); echo"<br>";
    var_dump($degisken5);
?>
```

bool(true)
int(1234)
string(12) "Hasan CEYHAN"
array(2) { [1]=> string(1) "a" [2]=> string(1) "b" }
float(1.234)

Aritmetik Operatörler

```
<?php
    $sayi1 = 10; $sayi2 = 5;
    echo "Toplama İşleminin Sonucu:".($sayi1+$sayi2);

    $sayi3 = $sayi2-$sayi1; echo "<br>";
    echo "Çıkarma İşleminin Sonucu: ".$sayi3;

    $sayi4 = $sayi2*$sayi1; echo "<br>";
    echo "Çarpma İşleminin Sonucu: ".$sayi4;

    $sayi5 = $sayi2/$sayi1; echo "<br>";
    echo "Bölme İşleminin Sonucu: ".$sayi5;

    $sayi6 = $sayi2%$sayi1; echo "<br>";
    echo "Mod Alma İşleminin Sonucu: ".$sayi6;
?>
```

Toplama İşleminin Sonucu: 15
Çıkarma İşleminin Sonucu: -5
Çarpma İşleminin Sonucu: 50
Bölme İşleminin Sonucu: 0.5
Mod Alma İşleminin Sonucu: 5

Arttırma ve Eksiltme Operatörleri

++	Önceden ve Sonradan Arttırma	<pre><?php \$deger = 4; \$deger++; // \$deger 5 oldu. echo \$deger; // İşlem Sonucu 5 echo "
"; \$deger=10 - ++\$deger; // \$deger 6 oldu echo \$deger; // İşlem Sonucu 4 /* \$deger değişkeninin çıkarma işlemine girmeden değeri 1 arttı.*/ ?></pre>
--	Önceden ve Sonradan Eksiltme	<pre><?php \$deger = 4; \$deger--; // \$deger 3 oldu. echo \$deger; // İşlem Sonucu 3 echo "
"; \$deger=10 - --\$deger; // \$deger 2 oldu echo \$deger; // İşlem Sonucu 8 /* \$deger değişkeninin çıkarma işlemine girmeden değeri 1 eksildi.*/ ?></pre>

Mantıksal Operatörler

Operatör	Açıklama	Örnek İşlem
\$a and \$b \$a && \$b	Ve	\$a ve \$b her ikisi de doğruysa sonuç doğrudur (TRUE).
\$a or \$b \$a \$b	Veya	\$a veya \$b doğruysa sonuç doğrudur.
\$a xor \$b	Ayrıcalıklı Veya	\$a veya \$b doğruysa sonuç doğru, her ikisinde doğruysa sonuç yanlıştır (FALSE).
! \$a	Değil	\$a doğru değilse sonuç doğrudur.

Atama Operatörleri

=	Atama İşlemi	<pre><?php \$adSoyad="Emrah Yüksel"; \$sayi=5; \$deger=TRUE; ?></pre>
+=	Toplayarak Atama	<pre><?php \$sayi=5; echo \$sayi+=10; # CEVAP # İşlem Sonucu 10 ?></pre>
-=	Çıkartarak Atama	<pre><?php \$sayi=5; echo \$sayi-=10; # CEVAP # İşlem Sonucu -5 ?></pre>
=	Çarparak Atama	<pre><?php \$sayi=5; echo \$sayi=10; # CEVAP # İşlem Sonucu 50 ?></pre>
/=	Bölerek Atama	<pre><?php \$sayi=5; echo \$sayi/=10; # CEVAP # İşlem Sonucu 0.5 ?></pre>
%	Mod Alarak Atama	<pre><?php \$sayi=5; echo \$sayi%=10; # CEVAP # İşlem Sonucu 5 ?></pre>
.=	Birleştirerek Atama	<pre><?php \$ad="Emrah"; echo \$ad.=" Yüksel"; # CEVAP #Emrah Yüksel ?></pre>

Karşılaştırma Operatörleri

Operatör	Açıklama	Örnek İşlem
==	Eşittir	\$a == \$b
===	Denktir	\$a === \$b
!=	Eşit değildir	\$a != \$b
!==	Denk değildir	\$a !== \$b
<>	Eşit değildir	\$a <> \$b
<	Küçüktür	\$a < \$b
>	Büyüktür	\$a > \$b
<=	Küçük veya eşittir	\$a <= \$b
=>	Büyük veya eşittir	\$a >= \$b

Tür Dönüşümleri

Değişkenler üzerinde tür değişimi uygulayabiliriz.

1- string <pre><?php \$deger = (string) 12; // \$deger = "12"; \$deger = (string) 9.71; // \$deger = "9.71"; \$deger = (string) 2.3E+6; // \$deger = "2300000"; \$deger = (string) true; // \$deger = "1"; \$deger = (string) false; // \$deger = ""; \$deger = 4 + "8"; // integer 12; ?></pre>	2- unset() Fonksiyonu Değişkeni tanımsız kılar. unset() fonksiyonu ile değişken içeriği silinir. <pre><?php \$a = "Emrah Yüksel"; unset(\$a); // \$a değişkeni artık tanımsız. echo \$a; ?></pre>
3- boolean <pre><?php \$deger = (bool) "PHP"; // true \$deger = (bool) 0; // false \$deger = (bool) 5; // true \$deger = (bool) ""; // false ?></pre>	4- int <pre><?php \$tutar=10.20; \$sonuc = (int) \$tutar; echo \$sonuc; ?></pre>

Tek Tırnak (') ve Çift Tırnak (")Arasındaki Fark

Çift tırnak içerisinde değişken içerikleri okunabilir, tek tırnak içerisinde değişken içerikleri okunmaz.

```
<?php
$ad = "Hasan";
echo "Benim adım $ad"; echo "<br>";
echo 'Benim adım $ad';
# CEVAP
# Benim Adım Hasan
# Benim Adım #ad
?>
```

Yoksayma ve Kaçış İşaretleri

Bazı verileri işaret etmemiz ya da değişkenlerin içeriğini yok saymamız gerekebilir.

\ işaretini yok saymak istediğimiz ifadenin sol tarafına getiririz.

```
<?php
echo "Ben $ad \"Udemy\" Kursuna Kayıt Oldum"; echo "<br>";
echo "Ben \$ad \"Udemy\" Kursuna Kayıt Oldum";
# CEVAP
# Ben Hasan "Udemy" Kursuna Kayıt Oldum
# Ben #ad "Udemy" Kursuna Kayıt Oldum
?>
```


Sık Kullanılan Hazır String Metin Fonksiyonları

1. strtolower ve strtoupper Fonksiyonları

strtolower() fonksiyonu, verilen verilerin hepsini küçük harfe çevirir.

strtoupper() fonksiyonu, verilen verilerin hepsini büyük harfe çevirir.

```
<?php
$yazi="BEN UDEMY ILERI SEVIYE PHP KURSUNA KAYITLIYIM.";
echo $yazi=strtolower($yazi);      echo "<br>";
# CEVAP
# ben udemy ileri seviye php kursuna kayıtlıyım.

echo $yazi=strtoupper($yazi);
# CEVAP
# BEN UDEMY ILERI SEVIYE PHP KURSUNA KAYITLIYIM.
?>
```

2. ucwords() ve ucfirst() Fonksiyonları

ucwords() fonksiyonu, metnin kelimelerinin ilk harflerini büyük yazar.

ucfirst() fonksiyonu, metnin ilk kelimesinin ilk harfini büyük yazar.

```
<?php
$yazi="BEN UDEMY ILERI SEVIYE PHP KURSUNA KAYITLIYIM.";
echo $yazi=ucwords($yazi);      echo "<br>";
# CEVAP
# Ben UdemY Ileri Seviye Php Kursuna KayitliYim.

echo $yazi=ucfirst($yazi);
# CEVAP
# Ben udemy ileri seviye php kursuna kayitliYim.
?>
```

3. strlen() ve substr() Fonksiyonları

strlen() fonksiyonu, metnin karakter sayısını verir.

substr() fonksiyonu, metnin belirtilen kısmı kadarını gösterir.

```
<?php
$yazi="BEN UDEMY ILERI SEVIYE PHP KURSUNA KAYITLIYIM.";      echo "<br>";
echo "$yazi değişkeninde olan karakter sayısı :".strlen($yazi);
# CEVAP
# BEN UDEMY ILERI SEVIYE PHP KURSUNA KAYITLIYIM. Değişkeninde olan karakter sayısı :46

echo substr($yazi,0,10);
# CEVAP
# BEN UDEMY
?>
```

Sabitler

Sabitler de değişkenler gibi bir değer tutar ancak değişkenler gibi değerleri değiştirilemez.

Sabitler isimlendirilirken; değişkenlerdeki gibi \$ işareti kullanmaya gerek yoktur.

Sabitler **boolean**, **integer**, **float** ve **string** türünde değer alabilir.

1. Sabit Tanımlama

Sabitler **define()** ya da **const** deyimiyle tanımlanabilir.

Sabit değerlerini okumak için başına \$ işareti koymuyoruz.

Bir sabitin tanımlanıp tanımlanmadığını **defined()** fonksiyonuyla öğrenebiliriz. Biz isimSoyisim sabitini tanımladığımız için örnekteki kod bize 1 sonucunu döndürecektir.

```
<?php
define("Hasan", "CEYHAN");
const isimSoyisim = "Beyza ATA";
echo isimSoyisim; echo "<br>";
echo defined("isimSoyisim");
# CEVAP
# Beyza ATA
# 1
?>
```

Diziler

Dizi içinde birden fazla değer barındığı için echo fonksiyonu ile aldıkları indis numarası belirtilmeden çıktı üretmezler.

1. Dizi Tanımlama ve print_r() Fonksiyonu

print_r() fonksiyonu, dizi değişkenlerin tüm elemanlarının indis numaralarıyla birlikte çıktısını verir.

Dizi değişkenlerin içeriğini print_r() fonksiyonu ile düzenli görüntüleyebilmek için **<pre>** HTML etiketinden yararlanabiliriz.

```
<?php
    $kurslar = array(
        "PHP",
        "BOOTSTRAP",
        "JS"
    );

    $kurslar = [
        "PHP",
        "BOOTSTRAP",
        "JS"
    ];
    echo $kurslar;      echo "<br>";
    echo $kurslar[1];  echo "<br>";
    print_r($kurslar);

    echo "<pre>";
    print_r($kurslar);
    echo "</pre>";
?>
```

```
Array
BOOTSTRAP
Array ( [0] => PHP [1] => BOOTSTRAP [2] => JS )

Array
(
    [0] => PHP
    [1] => BOOTSTRAP
    [2] => JS
)
```

2. Çok Boyutlu Diziler

Dizilerin içinde de dizi tanımlayabiliriz. Buna çok boyutlu dizi diyoruz. Örnekte gördüğümüz gibi \$kurslar dizisinin altında PHP, Bootstrap ve Javascript dizileri mevcut.

```
<?php
    $kurslar=[
        "PHP"=>    ["Egitmen"=>"Emrah", "Sure"=>"42 Saat", "Puan"=>"5"],
        "BOOTSTRAP"=> ["Egitmen"=>"Emrah", "Sure"=>"12 Saat", "Puan" =>"5"],
        "JAVASCRIPT"=>["Egitmen"=>"Emrah", "Sure"=>"11 Saat", "Puan"=>"5"]
    ];

    echo $kurslar["PHP"]["Egitmen"]; echo "<br>";
    echo $kurslar["PHP"]["Sure"];   echo "<br>";
    echo $kurslar["PHP"]["Puan"];   echo "<br>";
    # CEVAP
    # Emrah
    # 42 Saat
    # 5
?>
```

3. Dizi Fonksiyonları

count() Dizinin toplam eleman sayısını verir.	implode() Dizi elemanlarını, belirttiğimiz bir ayraç karakterle string ifadeye çevirir.
explode() String ifadeleri belirtilen ayraç karakter aracılığı ile parçalayarak dizi hâline çevirir.	is_array() Değişkenin dizi olup olmadığı sonucunu döndürür. Değer 1 olarak dönerse dizi, değer dönmezse dizi değildir.
array_reverse() Dizi elemanlarını şekil bakımından tersten sıralama yapar.	asort() ve arsort() Dizi elemanlarını A'dan Z'ye doğru sıralar. Bir dizinin değerlerini anahtarlarıyla ilişkilerini bozmadan tersine sıralar.
ksort() ve krsort() Bir diziyi anahtarlarına göre A'dan Z'ye doğru sıralar. Bir diziyi anahtarlarına göre Z'den A'ya doğru sıralar.	current() Dizinin gösterilebilen ilk elemanını verir. Ayrıca pos() takma adıyla kullanılabilir.
end() Dizinin son elemanını verir.	shuffle() Her sorguda dizi elemanlarının yerlerini değiştirir. Buna indis numaraları da dâhildir.
array_rand() Bir diziden belli sayıda rasgele eleman döndürür. Dönen değerler indis olduğundan erişmek için bu değerlerin belirtilmesi gerekir.	array_search() Bir dizide belirtilen değeri arar ve bulursa ilgili anahtarı döndürür.
array_unshift() Dizinin başlangıcına bir ya da daha fazla eleman ekler.	array_shift() Dizini başlangıcından bir eleman çıkartır.
array_push() Dizinin sonuna bir ya da birden fazla eleman ekler.	array_pop() Dizinin sonundan bir eleman çıkartır.
array_slice() Dizide belirtilen aralıktaki elemanları döndürür.	array_splice() Dizinin belli bir bölümünü siler ve yerine başka bir şeyler koyar.
array_sum() Dizi elemanlarının toplam değerini bulur.	array_product() Dizi elemanlarının çarpım değerini bulur.
next() ve prev() Dizide önceki ve sonraki elemanlar arasında geçiş yapar.	array_search() Dizide belirtilen değeri arar. Değer bulunursa değer anahtarı döndürülür.
in_array() Dizide aradığımız değer varsa 1 değerini döndürür.	array_flip() Dizide anahtarlar ile elemanların yerini değiştirir.
array_merge() Dizileri birleştirir.	array_unique() Dizi içerisinde yinelenen değerleri siler.
array_replace() Bir dizinin elemanlarını belirtilen dizilerden günceller. Bir ya da daha fazla dizi ile kullanılabilir.	array_count_values() Dizide yinelenen elemanların kaç kez yinlendiği bilgisini verir.

4. Dizilerde Kullanılan Hazır Fonksiyonlar

a. sort() ve rsort Fonksiyonları

sort(), verileri küçükten büyüğe doğru sıralar.

rsort(), verileri büyükten küçüğe doğru sıralar.

<pre><?php \$dizi=array(7,8,9,6,5,4,1,2,3,1); echo "<pre>"; sort(\$dizi); print_r(\$dizi); echo "</pre>"; ?></pre>	Array ([0] => 1 [1] => 1 [2] => 2 [3] => 3 [4] => 4 [5] => 5 [6] => 6 [7] => 7 [8] => 8 [9] => 9)
<pre><?php \$dizi=array(10,9,15,101,8,7,6,12,5,4,3); echo "<pre>"; rsort(\$dizi); print_r(\$dizi); echo "</pre>"; ?></pre>	Array ([0] => 101 [1] => 15 [2] => 12 [3] => 10 [4] => 9 [5] => 8 [6] => 7 [7] => 6 [8] => 5 [9] => 4 [10] => 3)
<pre><?php \$dizi=array("a","b","c","d"); echo "<pre>"; rsort(\$dizi); print_r(\$dizi); echo "</pre>"; ?></pre>	Array ([0] => d [1] => c [2] => b [3] => a)

b. implode() Fonksiyonu

implode() fonksiyonu, dizi değerlerini birleştirmeye yarar.

```
<?php
$dizi=array("a","b","c","d");
$sonuc=implode("+", $dizi);
echo $sonuc;
# CEVAP
# a+b+c+d
?>
```

c. in_array() Fonksiyonu

Dizi içerisinde aradığımız değerin olup olmadığını denetler. Varsa 1 değerini döndürür.

```
<?php
$dizi=array("a","b","c","d");
echo $sonuc=in_array("z",$dizi);
if ($sonuc){
    echo "var";
}
else{
    echo "yok";
}
# CEVAP
# yok
?>
```

d. explode() Fonksiyonu

explode() fonksiyonu, değişkeni parçalayarak dizi haline getirir.

```
<?php
$zaman="27-10-2017 19:08";
$sonuc=explode(" ",$zaman);

echo "<pre>";
print_r($sonuc);
echo "</pre>"; echo "<br>";

echo "Tarih : ".$sonuc[0]. " Saat: ".$sonuc[1];
?>
```

```
Array
(
    [0] => 27-10-2017
    [1] => 19:08
)
```

Tarih : 27-10-2017 Saat:.19:08

Koşullar (Şartlar)

```
<?php
    $sayi=5;
    if ($sayi==4){
        echo "Sayı değişkeninin değeri 4";
    }
    elseif($sayi==5){
        echo "Sayı değişkeninin değeri 5";
    }
    else{
        echo "Değer tespit edilemedi.";
    }
    # CEVAP
    # Sayı değişkeninin değeri 5
?>
```

1. Farklı Söz Dizimleri

Aşağıda yer alan örnek söz dizimin de PHP kapanış etiketleri kullanıldığından HTML kodlarını da koşul içerisinde direkt olarak kullanabilirsiniz.

```
<?php
    $sayi=5;
    if ($sayi==5):
?>
    <p>Koşul Doğru</p>
<?php
    endif;
    # CEVAP
    # Koşul Doğru
?>
```

İki noktalı söz dimi, else ve elseif için de kullanılabilir.

```
<?php
    $sayi=5;
    if ($sayi == 5):
        echo "Sayı değişkeninin değeri 5";
    elseif ($sayi == 6):
        echo "Sayı değişkeninin değeri 6";
    else:
        echo "Değişken değeri tespit edilemedi.";
    endif;
?>
```

if koşullarımızı amacına uygun olarak daha kısa bir şekilde yazabiliriz.

```
<?php
    $sayi=5;
    echo $sayi>5 ? "Sayı 5'ten Büyük" : "Yanlış";
    # CEVAP
    # Yanlış
?>
```

2. Koşulların Mantıksal Operatörler ile Kullanımı

```
<?php
$sayi1=5;
$sayi2=10;
if ($sayi1==5 || $sayi2==3){
    echo "Doğru";
}

$sayi1=5;
$sayi2=10;
if ($sayi1==5 and $sayi2>3) {
    echo "Doğru";
}

<?php
# XOR, her ikiside doğruysa sonuç doğru olur.
$sayi1=5;
$sayi2=10;
if($sayi1==5 XOR $sayi2==10){
    echo "XOR TRUE";
}
else{
    echo "XOR FALSE";
}

$deger=FALSE;
if (!$deger) {
    echo "FALSE";
}
?>
```

3. Switch Case

```
<body>
<?php
if (isset($_POST['notislemi'])){
    $sayi=$_POST['not'];
    switch ($sayi){
        case '1':    echo "KALDINIZ";
                     break;
        case '2':    echo "ZORLA GEÇTİNİZ";
                     break;
        case '3':    echo "ORTA İLE GEÇTİNİZ";
                     break;
        default:     echo "TANIMLANAMAYAN SAYI";
                     break;
    }
}
?> <hr>
<form action="" method="POST">
    Notunuzu Girin : <input type="text" name="not" placeholder="Notunuzu Girin">
    <button type="submit" name="notislemi">Sonuçlandır</button>
</form>
</body>
```

Notunuzu Girin :

Döngüler

1. for Döngüsü ve break Deyimi

```
<?php
    for($i = 1; $i<3; $i++){
        echo "1. Döngü $i <br>";
        for($j = 3; $j >1; $j--){
            echo "2. Döngü $j <br>";
            break;
        }
    }
# CEVAP
# 1. Döngü 1
# 2. Döngü 3
# 2. Döngü 2
# 1. Döngü 2
# 2. Döngü 3
# 2. Döngü 2
?>
```

a. Farklı Söz Dizimi

```
<?php
    for ($sayi = 1, $topla = 0; $sayi <= 10; $topla += $sayi, $sayi++);
    echo $topla;
# CEVAP
# 55
?>
```

b. for Döngüsünde HTML Kipine Geçiş

```
<?php
    for ($i=1; $i <=20; $i+=2){
        ?>
        <b style="background-color: green; color:white"><?php echo $i ?></b>
    }
?>
```

3 7 11 15 19

2. continue Deyimi

Continue deyimi, döngüyü yarıda kestirerek işleme devam edilmesini sağlar.

```
<?php
    for ($i=1; $i <=5 ; $i++){
        if (($i % 2)){
            # Tek sayıları atla
            continue;
        }
        echo $i."<br/>";
    }
# CEVAP
# 2
# 4
?>
```

3. goto Deyimi

Aşağıdaki örnekte goto deyimi, döngü içerisinde koşul sağlandığında belirlediğimiz komuta geçiş yapacak.

```
<?php
    for($i=0; $i<5; $i++){
        if($i == 3){
            goto metin3;
        }
    }
    metin3: echo "3 Numaralı Sayıda Döngü Dışına Çıkıldı.";
    # CEVAP
    # 3 Numaralı Sayıda Döngü Dışına Çıkıldı.
?>
```

4. while Döngüsü

while döngüsü bir şart sağlanıyorsa sürekli içindeki komutları çalıştırır.

```
<?php
$veri=TRUE;
$i = 0;
$satirSayisi = 3;
while ($veri){
    if ($i != $satirSayisi) {
        $i++;
        echo $i." . veri çekildi.<br/>";
    }
    else{
        echo "Çekme Tamamlandı.";
        break;
    }
}
# CEVAP
# 1. veri çekildi.
# 2. veri çekildi.
# 3. veri çekildi.
# Çekme Tamamlandı.
?>
```

a) HTML Başlık Etiketlerini Yazdırma

```
<?php
    $i = 1;
    while ($i <= 3){
        echo "<h$i>$i. Kodlab Yayınevi </h$i>";
        $i++;
    }
    # CEVAP
    # Kodlab Yayınevi
    # Kodlab Yayınevi
    # Kodlab Yayınevi
?>
```

5. do while Döngüsü

do-while döngüsü, while döngüsünden farklı olarak koşul gerçekleşmese dahi 1 kez çalışır.

Aşağıda yer alan örnekte while döngüsü içerisinde yer alan koşul gerçekleşmediği hâlde kod bloğunda yer alan kodun çalıştığını görebilirsiniz.

```
<?php
    $i = 0;
    do{
        echo $i;
    }
    while($i > 0);
    # CEVAP
    # 0
?>
```

6. foreach Döngüsü

Foreach döngüsü dizilerde yer alanları göstermek için kullanılır.

\$değer değişkeninin önüne **&** işaretini koyarak dizi elemanları üzerinde kolayca değişiklik yapılabilir. Böylece döngü, değer gönderimli hâle gelir.

```
<?php
$dizi=["PHP", "Bootstrap", "jQuery"];
foreach ($dizi as $key){
    echo $key."<br/>";
}
# CEVAP
# PHP
# Bootstrap
# JQuery
?>
```

```
<?php
$dizi = array(1, 2, 3, 4);
foreach ($dizi as &$değer){
    echo $değer = $değer * 2;
}
echo "<br/>";
print_r($dizi);
# CEVAP
# 2468
# Array ( [0] => 2 [1] => 4 [2] => 6 [3] => 8 )
?>
```

a) foreach Anahtarları ile Değer Döndürme

Yaptığımız örneklerde sadece dizinin değerlerine ulaştık. Dilersek hem anahtar hem de değere ulaşabiliriz.

```
<?php
$dizi=[
    "1" => "Javascript",
    "2" => "PHP",
    "3" => "Bootstrap",
];
foreach ($dizi as $key => $deger) {
    echo "Anahtar $key => Değer $deger."<br/>";
}
# CEVAP
# Anahtar 1 => Değer Javascript
# Anahtar 2 => Değer PHP
# Anahtar 3 => Değer Bootstrap
?>
```

Aşağıda yer alan dizi elemanlarının toplamını bulan örneği uygulayalım.

```
<?php
$toplam = 0;
$dizi = [1,3,5,7,9];
foreach($dizi as $d)
    $toplam+=$d;
echo "Dizinin Elemanları Toplamı: $toplam";
# CEVAP
# Dizinin Elemanları Toplamı: 25
?>
```

Dosyaları Dâhil Etme

Çalıştığımız proje dosyalarına dilediğimiz zaman başka .php dosyalarını dâhil edebiliriz.

1. **include**

Belirtilen dosyayı çalıştığımız sayfa içerisine dâhil eder. Dâhil edilmek istenen dosya bulunamazsa uyarı mesajı ile karşılaşırız.

```
<?php
include 'header.php';
# CEVAP
# WARNING ...
?>
```

header.php dosyasını dâhil etmeye çalıştık. Fakat böyle bir dosya olmadığı için hata aldık.

2. **include_once**

include_once deyiminin çalışma biçimi include deyimi ile benzerdir. Aralarındaki fark dâhil edilmiş dosya kodun başka bir yerinde tekrar dâhil edilmek istenirse buna izin vermemesidir.

Eğer yazdığınız kod içerisinde dâhil edilen dosyanın birden fazla kez dâhil edilme ihtimali varsa ve bundan emin olunmak istenirse include_once kullanılmalıdır.

HTML Formlar ile Çalışma

1. Form Verilerini İşleme

[Formlarla ilgili bilgilere ulaşmak için tıklayınız.](#)

2. post Metodu

Bu dizi değişkeni formdan gelen tüm verileri içerisinde barındırır ve anahtar değerler belirtilerek formdan gelen verilere **\$_POST** dizi değişkeni kullanılarak ulaşılabilir. Formdan gönderilen verilerin anahtar değerleri form elemanlarında belirttiğimiz **name** alanlarında yazdığımız değerlerdir.

index.php Dosyası <pre><body> <h1>Kullanıcı Girişi</h1> <form action="islem.php" method="POST"> Kullanıcı Adı <input type="text" name="kadi"> Şifre <input type="text" name="pass"> <input type="submit" value="Giriş Yap" name="kullanici_giris"> </form> </body></pre>	islem.php Dosyası <pre><?php echo "<pre>"; print_r(\$_POST); echo "<pre/>"; ?></pre>
	Array ([kadi] => hasan [pass] => 12345 [kullanici_giris] => Giriş Yap)
localhost/BilgisayarMuhendisligiUygulamaları/PHP%20Kodları/islem.php	

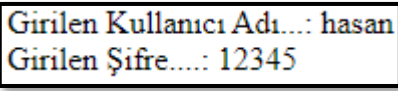
Örnekte görmüş olduğunuz gibi **\$_POST** dizi değişkeni içeriğinde tüm form eleman verilerini barındırıyor.

Tek bir anahtar değere erişmek istersek;

```
$_POST['kadi'];
```

şeklinde kullanabiliriz. Şimdi, bununla ilgili bir örnek yapalım. Formdan gelen verileri tek tek işleyelim ve ekrana çıktı verelim.

islem.php dosya içeriğini şu şekilde değiştirelim;

<pre><?php echo "Girilen Kullanıcı Adı...: ".\$_POST['kadi']; echo "
"; echo "Girilen Şifre....: ".\$_POST['pass']; echo "
"; ?></pre>	
--	--

3. get Metodu

index.php Dosyası

```
<body>
  <h1>Kullanıcı Girişi</h1>
  <form action="islem.php" method="GET">
    Kullanıcı Adı <input type="text" name="kadi">
    Şifre <input type="text" name="pass">
    <input type="submit" value="Giriş Yap" name="kullanici_giris">
  </form>
</body>
```

islem.php Dosyası

```
<?php
  echo "Girilen Kullanıcı Adı....: ".$_POST['kadi'];
  echo "<br/>";
  echo "Girilen Şifre.....: ".$_POST['pass'];
  echo "<br/>";
?>
```

Kullanıcı Girişi

Kullanıcı Adı Şifre

Girilen Kullanıcı Adı....: hasan
Girilen Şifre.....: 12345

localhost/BilgisayarMuhendisligiUygulamalari/PHP%20Kodlari/islem.php?kadi=hasan&pass=12345&kullanici_%0D%0Aagiris=Giris+Yap

HTML formundan gönderilen verilerin adres satırında (URL) gözüktüğünü görebilirsiniz.

GET metodu ile veri gönderimini sadece form aracılığı ile değil, oluşturduğumuz linkler aracılığıyla da gerçekleştirebiliriz.

4. get ve post Arasındaki Farklar

İşlem	GET	POST
Görünürlük	Veriler URL'deki herkes tarafından görülebilir.	URL'de veriler gösterilmez.
Kısıtlamalar	GET ile veriler gönderilirken URL'ye eklendiğinden maksimum uzunluk 2048 karakterle sınırlıdır.	Kısıtlama yoktur.
Güvenlik	Gönderilen veriler URL'nin bir parçası olduğu için GET, POST ile karşılaştırıldığında daha az güvenlidir.	POST, parametreler tarayıcı geçmişinde veya web sunucusunda depolanmadığından GET'den biraz daha güvenlidir.
Geri Butonu / Yeniden Yükleme	Uygulanabilir.	Kullanıcı verilerin tekrar gönderildiği konusunda uyarılır.
Yer İşareti	Uygulanabilir.	Uygulanamaz.
Ön Bellekleme	Uygulanabilir.	Uygulanamaz.

5. Link Üzerinden GET Metodu ile Veri Gönderme

GET metodunu kullanarak oluşturacağımız bir link üzerinden sabit verileri hedef sayfaya iletebiliriz. Bu verileri dilersek bir değişkene aktararak da kullanabiliriz. index.php dosyasının içeriğinde form düzenini bozmadan altına aşağıdaki örnek kodu yazalım.

```
<a href="islem.php?kadi=kodlab&pass=123456">Kullanıcı Giriş</a>
```

Örnek kod üzerinde gördüğümüz href özelliğinde verilerin işleneceği dosya adı islem.php'yi belirttik. Dosya isminden sonra GET metodu ile göndereceğimiz verileri belirtmeden önce ? işaretini kullanıyoruz. Daha sonra aynı form elemanlarında verinin anahtar değerini belirttiğimiz name kısmında yazdığımız gibi anahtar değeri belirttikten sonra taşınacak değeri belirtiyoruz. Birden fazla değeri aralarında & işaretini kullanarak ekleyebiliriz.



index.php dosyasını ön izlediğimizde "Kullanıcı Giriş" linkinin üzerine gelelim ve durum çubuğuna göz atalım. Yazdığımız GET metodu ile gönderilecek verilerin link içerisinde izlendiğini görebilirsiniz. Linke tıklayalım.

```
<?php
    echo "Girilen Kullanıcı Adı...: ".$_GET['kadi'];      echo "<br/>";
    echo "Girilen Şifre....: ".$_GET['pass'];            echo "<br/>";
?>
```

Ekran çıktısında gördüğümüz gibi veriler GET metodu ile islem.php dosyasına iletildi ve \$_GET dizi değişkeni ile belirttiğimiz anahtar değer aracılığı ile verilere tekrar ulaştık.

6. isset() Fonksiyonu

isset() fonksiyonu bir değişkenin tanımlı olup olmadığını kontrol eder. Belirtilen değişken tanımlıysa ve NULL değilse TRUE döner.

<pre><?php \$sayi=5; echo isset(\$sayi); # CEVAP # 1 ?></pre>	<pre><?php \$sayi=5; if (isset(\$sayi)){ echo "Tanımlı"; } else{ echo "Değil"; } # CEVAP # Tanımlı ?></pre>
---	---

7. Birden Fazla Formdan Veri Yakalama

Oluşturulan dosyalar üzerinde birden fazla form için işlem yapmamız gerekebilir. Bu durumlarda hangi verinin hangi formdan geldiğini bilmemiz ve buna göre işlem bloklarını yönetmemiz gerekmektedir. Bu noktada PHP ile değişkenlerin varlığını kontrol eden fonksiyonlardan yararlanacağız.

```
<body>
  <!-- Form 1 -->
  <form action="islem.php" method="POST">
    Kullanıcı Adı <input type="text" name="kullanici_kadi">
    Şifre <input type="text" name="kullanici_pass">
    <input type="submit" value="Giriş Yap" name="kullanici_giris">
  </form>

  <!-- Form 2 --> <br>
  <form action="islem.php" method="POST">
    Admin Adı <input type="text" name="admin_kadi">
    Admin Şifre <input type="text" name="admin_pass">
    <input type="submit" value="Giriş Yap" name="admin_giris">
  </form>

<?php
  if(isset($_POST['kullanici_giris'])){
    echo "Kullanıcı Giriş Formu İşlemleri";
  }
  if(isset($_POST['admin_giris'])){
    echo "Admin Giriş Formu İşlemleri";
  }
?>
</body>
```

Kullanıcı Adı	<input type="text"/>	Şifre	<input type="text"/>	Giriş Yap
Admin Adı	<input type="text"/>	Admin Şifre	<input type="text"/>	Giriş Yap

8. empty() Fonksiyonu

Empty fonksiyonu bir değişkenin boş olup olmadığını tespit eder.

• " " (boş bir dizge)	• 0 (bir tamsayı olarak 0)	• "0 " (bir dizge olarak 0)	• var \$var; (bir sınıf içinde bir değeri olmaksızın bildirilmiş bir değişken)
• FALSE	• array() (boş bir dizi)	• NULL	

isset fonksiyonu değişkenin var olup olmadığını kontrol ederken; **empty fonksiyonu** değişkenin içeriğinin boş olup olmadığını bakar.

9. header() Fonksiyonu

PHP 'de yönlendirme, yenileme, içerik türü belirleme ve ön belleğe alma gibi amaçlarla kullanılabilir.

```
Header("Location:index.php?durum=ok");
```


a. İçerik Türü Belirleme

Aşağıdaki örnekte, kullanıcıya bir PDF dosyası kaydetmesi için sayfa başlık bilgisini ileteceğiz. Tarayıcı bunun bir PDF dosyası olduğunu ilettiğimiz header başlığı sayesinde tespit edecektir.

```
<?php
// Bir PDF çıktılayacağız.
header('Content-type: application/pdf');
// Dosya ismi indirilen.pdf olsun.
header('Content-Disposition: attachment; filename="indirilen.pdf"');
?>
```

b. Ön Bellekleme

Tarayıcılar verileri daha hızlı işlemek için web sayfalarını ön belleklerine alırlar. Bu sayede kullanıcının bir sonraki ziyaretinde veriler önbellekten getirildiği için daha hızlı açılır.

Aşağıdaki örnekte tarayıcının ön bellekleme işlemini kapatmasını sağlayabiliriz.

```
<?php
header("Cache-Control: no-cache, must-revalidate"); // HTTP/1.1
header("Expires: Sat, 26 May 2018 05:00:00 GMT"); // Geçmişte bir tarih
?>
```

c. Yönlendirme

Location özelliği ile belirttiğimiz bir linke yönlendirme işlemi yapabiliriz.

```
<?php
header("Location:https://www.kodlab.tv");
?>
```

Örnekte yer alan kodu uyguladığımızda sayfa Location kısmında belirttiğimiz linke yönlenecektir.

d. Yenileme (Refresh)

Sayfanın yenilenmesini ya da belirtilen linke süreli olarak gönderilmesini sağlayabiliriz.

```
<?php
header("Refresh: 3; url=https://www.kodlab.tv");
echo "3 saniye sonra yönlendiriliyorsunuz...";
# CEVAP
# 3 saniye sonra yönlendiriliyorsunuz...
?>
```

Örnek kodu uyguladığımız da sayfamız Refresh özelliğinde belirtilen süre sonra url özelliğinde belirlediğimiz linke yönlenecektir

\$_SESSION Oturum Yönetimi

Session kullanarak kullanıcıların giriş ve çıkışlarını kontrol edebilir, **şifreli sayfalar** oluşturabilir veya Session üzerinde çeşitli verileri saklayabiliriz. Session verilerini kendimiz silebileceğimiz gibi tarayıcımızı kapatarak da silinmesini sağlayabiliriz. Session verileri sunucu tarafında saklanır ve bu yüzden güvenilir oldukları için oturum açma ve kapama işlemlerinde kullanılır.

1. Session Başlatma

Session işlemlerini yapabilmemiz için **session_start()** fonksiyonu ile başlatmalıyız. Hem okuma hem de oluşturma işlemlerinde session'u başlatmalıyız.

```
<?php
    session_start() // Session'u başlatır.
?>
```

2. Session Oluşturma ve Okuma

Bir session oluşturalım ve bunu mevcut sayfada ve bir başka bir sayfada okuyalım.

```
<?php
    session_start();
    $_SESSION['kadi'] = "Kodlab";
    $_SESSION['pass'] = "123456";
?>
```

Örnekte session oturumunu başlattık ve ardından **\$_SESSION** dizi değişkenine belirlediğimiz anahtar değerler ile kadi ve pass değerlerini atadık. Bu şekilde session'larımız tanımlanmış oldu. Bu verileri okumak istersek şu şekilde işlem yapabiliriz.

```
<?php
    echo $_SESSION['kadi'];
    echo $_SESSION['pass'];
?>
```

Session verileri, tarayıcı kapatılana kadar tutulduğu için tarayıcımızda açacağımız yeni sayfalarda bu verileri okumaya devam edebiliriz. **islem.php** dosyasında bu verileri okuyalım.

```
<?php
    session_start();
    echo $_SESSION['kadi'];
    echo $_SESSION['pass'];
    # CEVAP
    # Kodlab123456
?>
```

Görüldüğü üzere biz silmediğimiz ve tarayıcıyı kapatmadığımız müddetçe oluşturduğumuz session verilerine erişebiliriz.

3. Session Silme

Tek bir session'u silmek için **unset();**, toplu silmek için **session_destroy();** fonksiyonunu kullanırız.

```
<?php
    unset($_SESSION['kadi']);
    session_start();
    session_destroy();
?>
```

4. Session Dizi Atama

Session'lara ayrıca dizi değerleri de atayabiliriz. index.php dosyasının içeriği;

```
<?php
    session_start();
    $kurslar=["PHP",
              "JAVASCRIPT",
              "BOOTSTRAP",
              "JQUERY"];
    $_SESSION['kurslar']=$kurslar;
?>
<!-- islem.php dosya içeriği -->
<?php
    session_start();
    print_r($_SESSION['kurslar']);
    # CEVAP
    # Array ( [0] => PHP [1] => JAVASCRIPT [2] => BOOTSTRAP [3] => JQUERY )
?>
```

\$_COOKIE (Çerezler)

Çerezler, bilgisayarımızda yer alan tarayıcıda veri saklamayı ve böylece geri dönen kullanıcılar hakkında **veri takibi** yapmayı sağlar. Örnek kullanım alanı olarak; daha önceden oturum açtığımız bir sitede kullanıcı adı ve şifremizin tekrar girişte kayıtlı olduğunu görebiliriz, bu işlem çerezler sayesinde sağlanmaktadır.

Çerezler direkt ya da süre atanarak tanımlanabilir. Süre ataması yapıldığında belirlenen süre kadar bilgisayarımızda saklanabilirler.

1. Cookie (Çerez) Tanımlama

Cookie oluşturma işlemi için setcookie() fonksiyonunu kullanıyoruz. Tanımlama işleminde okuma işlemlerinde kullanacağımız bir değer anahtarı ve değer göndermemiz gerekiyor.

cookie adında bir klasör açalım ve içine **index.php**, **oku.php** ve **sil.php** adında üç dosya oluşturalım.

index.php dosya içeriği

```
<?php
    //Çerezimizi oluşturduk.
    setcookie("ad", "Emrah Yüksel");
?>
<!-- Çerezi okuyacağımız sayfaya link verdik -->
<a href="oku.php">Çerez Oluşturma Kontrolü</a>
```

2. Cookie (Çerez) Okuma

Çerez okuma işlemleri için **\$_COOKIE** dizisini kullanacağız.

oku.php dosya içeriği

```
<?php
    echo "Merhaba " . $_COOKIE['ad'];
?>
<br>
<a href="sil.php"><button>Sil</button></a>
```

Uyguladığımız örnekte **index.php** dosyası üzerinde oluşturduğumuz Cookie'yi okumak için **oku.php** bağlantısını kullandık ve ekran çıktısı aşağıdaki gibi oldu.

Tanımladığımız **ad** anahtar değerli **cookie** içeriğini göstermiş olduk.



3. Cookie (Çerez) Silme

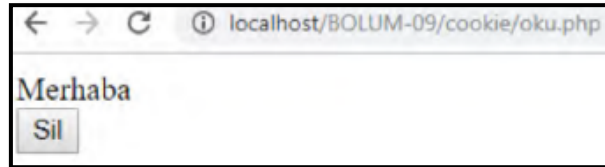
Tanımladığımız bir çerezi silmek için yine tanımlama yaparken kullandığımız **setcookie()** fonksiyonunu kullanacağız.

Çerez oluştururken çerezin silineceği bir **son kullanım tarihi** belirtebiliriz. Bunu, bir sonraki konu başlığında göreceğiz. Biz herhangi bir silinme süresi belirtmediğimiz için 1 saat öncesini işaret ederek Cookie'nin silinmesini sağlayacağız.

sil.php dosya içeriği

```
<?php
    setcookie("ad", "Emrah Yüksel", time() - 3600);
    header("Location:oku.php");
?>
```

Sil butonuna tıkladığınızda **sil.php** dosyasında cookie silinerek tekrar **oku.php** dosyasına dönüş yapılacak ve ekran sadece "**Merhaba**" yazdığını göreceksiniz.



Cookie silinmiş oldu. Tekrar oluşturulması için **index.php** dosyasını ziyaret etmeniz yeterli olacaktır.

4. Süreli Cookie (Çerez) Tanımlama

Çerezlerde zaman saniye cinsinden tutulur. Çerezler 1 saat süreyle saklanacaksa;

60 Saniye * 60 Dakika = 3600 Saniye // 1 Saat

60 Saniye * 60 Dakika * 24 Saat = 86400 Saniye // 1 Gün

60 Saniye * 60 Dakika * 24 Saat * 30 Gün = 2592000 Saniye // 1 Ay olarak tanımlanır.

```
<?php
    //Süreli Tanımlama 1 Gün;
    setcookie("ad", "Emrah Yüksel", time() + (60*60*24));
    setcookie("ad", "Emrah Yüksel", time() + (60*60*24));
    echo strtotime("+1 week 2 days 4 hours 2 seconds"), "\n";
?>
```

Bu işlemleri daha kısaltmak için **strtotime()** fonksiyonundan yararlanabiliriz. Strtotime fonksiyonu ile yazılı tarihi **unix** zaman damgasına çevirebiliriz.

\$_FILES Dosya Yükleme

PHP’de **dosya yükleme işlemlerini** yapmamızı sağlar. **\$_FILES** küresel değişkeni yüklemek istediğimiz tüm dosya bilgilerini içerir. Dosyanın türünü ve boyutunu öğrenmemize yardımcı olur.

<ul style="list-style-type: none">• \$_FILES['kullanici_dosyasi']['name'] İstemci makinesindeki asıl dosya adıdır.• \$_FILES['kullanici_dosyasi']['type'] Eğer tarayıcı bu bilgiyi sağladıysa dosyanın MIME türüdür. Örneğin, “image/gif”.• \$_FILES['kullanici_dosyasi']['size'] Yüklenen dosyanın bayt cinsinden boyutudur.• \$_FILES['kullanici_dosyasi']['tmp_name'] Yüklenen dosyanın sunucuda saklandığı sıradaki geçici dosya adıdır.• \$_FILES['kullanici_dosyasi']['error'] Bu dosya yüklemesiyle ilişkili hata kodudur. \$_FILES['kullanici_dosyasi']['error'] ‘un muhtemel üreteceği hata kodları.• UPLOAD_ERR_OK Değeri: 0; Hata yoktur, dosya yükleme başarılıdır.• UPLOAD_ERR_INI_SIZE Değeri: 1; Yüklenen dosya php.ini içindeki upload_max_filesize yönergesindeki değeri aşmaktadır.	<ul style="list-style-type: none">• UPLOAD_ERR_FORM_SIZE Değeri: 2; Yüklenen dosya HTML form içinde belirtilen MAX_FILE_SIZE değerini aşmaktadır.• UPLOAD_ERR_PARTIAL Değeri: 3; Dosya kısmen yüklenmiştir.• UPLOAD_ERR_NO_FILE Değeri: 4; Dosya yüklenmemiştir.• UPLOAD_ERR_NO_TMP_DIR Değeri: 6; Geçici dizin yoktur.• UPLOAD_ERR_CANT_WRITE Değeri: 7; Dosya diske yazılamamıştır.• UPLOAD_ERR_EXTENSION Değeri: 8; Dosya yükleme bir PHP eklentisi nedeniyle durmuştur. PHP buna hangi eklentinin sebep olduğunu bulmak için bir yol sağlamaz. phpinfo() ile yüklenen eklentilerin listesini alıp incelemek işe yarayabilir.
--	--

1. Dosya Yükleme İşlemi

upload isminde bir klasör açalım ve içerisine **index.php** ve **islem.php** dosyalarımızı oluşturalım. Ayrıca yüklediğimiz dosyaları ayrı bir klasöre kaydetmek için upload klasörü altına **files** adında bir klasör daha açalım.

index.php dosya içeriği

```
<body>
  <!-- Formumuzda enctype
  özelliğini mutlaka belirtmemiz
  gerekiyor -->
  <form action="islem.php"
  method="POST"
  enctype="multipart/formdata">
    <input type="file"
    name="dosya">
    <input type="submit"
    value="Dosya Yükle"
    name="dosyayukle">
  </form>
</body>
```

islem.php dosya içeriği

```
<?php
  if(isset($_POST['dosyayukle'])){
    //Dosyanın yükleneceği dizin.
    $uploads_dir="files";
    //Geçici dosya adı:
    $tmp_name=$_FILES['dosya']['tmp_name'];
    //Asıl dosya adı:
    $name = $_FILES['dosya']['name'];
    // Bu fonksiyon karşıya yüklenen bir dosyayı yeni bir
    yere taşır.
    if(@move_uploaded_file($tmp_name,$uploads_dir/$name)){
      echo "Dosya Yüklendi.<br>";
      //Yüklenen dosyaya link veriyoruz.
      echo "<a href=\"\$uploads_dir/$name\">Dosya Aç </a>";
    }
    else{
      echo "Dosya Yüklenemedi Sebebi!<br>";
      //Dosya yüklemesinde oluşan hatanın kodunu verir.
      echo $_FILES['dosya']['error'];
    }
  }
}>
```

2. Dosya Boyut Kontrolü

`$_FILES['kullanici_dosyasi']['size']` özelliği ile yüklenen dosyanın byte cinsinden boyutunu öğrenebiliriz. **1 Megabyte => 1048576 Byte**

islem.php dosyasına **dosya yükleme işlemlerinden önce** şu kodu ekleyelim.

```
<?php
    if ($_FILES['dosya']['size']>1048576) {
        echo "Bu dosyanın boyutu izin verilenden yüksek...";
        exit;
    }
?>
```

3. Dosya Uzantı Kontrolü

islem.php dosyasına dosya yükleme işlemlerinden önce şu kodu ekleyelim.

```
<?php
//Dosya Uzantı Kontrolü;
$izinli_uzantilar=['jpg',
                  'png'
];
$ext=strtolower(substr($_FILES['dosya']['name'],strpos($_FILES['dosya']['name'],'.')+1));
if (in_array($ext, $izinli_uzantilar) === false) {
    echo "Bu uzantı kabul edilmiyor";
    exit;
}
?>
```

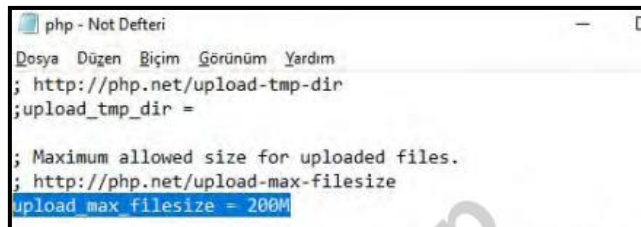
4. unlink() Fonksiyonu Dosya Silme

PHP ile yüklediğimiz dosyaları **unlink()** fonksiyonu ile silebiliriz.

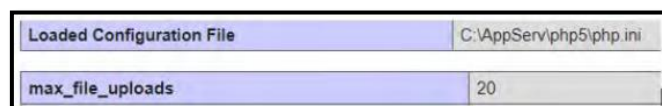
```
<?php
    unlink("metin.txt");
    echo "metin.txt dosyası silinmiştir.";
?>
```

5. Dosya Yükleme Limiti

php.ini dosyamızda (dosyanın konumunu **phpinfo()** fonksiyonu ile bulabiliriz) **upload_max_filesize** alanında belirtilen maksimum dosya yükleme limiti bulunmaktadır. Bu değeri değiştirerek lokal sunucunuzda dosya yükleme boyutunun limitini arttırabiliriz.



Loaded Configuration File alanının karşısında değişiklik yapmanız gereken **php.ini** dosyasının yolunu görebilirsiniz. Ayrıca **max_file_uploads** alanından da tek bir istek ile yüklenebilecek maksimum dosya sayısını kontrol edebilirsiniz.



Ayrıca aşağıda örneklendirdiğimiz **php_ini_loaded_file()** fonksiyonuda yüklü **php.ini** dosyanın yolunu vermektedir.

```
<?php
    $inipath = php_ini_loaded_file();
    if($inipath) {
        echo 'Yüklü php.ini: ' . $inipath;
    }
    else{
        echo 'Bir php.ini dosyası yüklü değil';
    }
?>
```

6. Çoklu (Multiple) Dosya Upload

PHP ile tek bir istekte birden fazla dosya yükleyebiliriz. HTML dizisi özelliği sayesinde birden fazla seçtiğimiz dosyalar dizi olarak iletilip işlenebilir. Çoklu dosya seçebilmemiz için oluşturduğumuz file input'unda multiple özelliğini tanımlamalıyız. Dosya yükleme için kullandığımız örnekteki kodlarımızı multiple adında oluşturacağımız yeni bir klasöre kopyalayalım ve aşağıdaki değişiklikleri uygulayalım.

```
<?php
if (isset($_POST['dosyayukle'])){
    //Dizi şeklinde gelen dosyaları yazdırıyoruz.
    foreach ($_FILES["dosya"]["name"] as $key => $name){
        //Dosyanın yükleneceği dizin.
        $uploads_dir="files";
        //Geçici dosya adı:
        $tmp_name=$_FILES['dosya']['tmp_name'][$key];
        //Asıl dosya adı:
        $name = $_FILES['dosya']['name'][$key];
        // Bu fonksiyon karşıya yüklenen dosyayı yeni bir yere taşır.
        if (@move_uploaded_file($tmp_name,$uploads_dir/$name)){
            echo "$name Dosya Yüklendi.<br/>";
            //Yüklenen dosyaya link veriyoruz.
            echo "<a href=\"\$uploads_dir/$name\">Dosya Aç </a>";
            echo "<br/>";
        }
        else{
            echo "Dosya Yüklenemedi Sebebi!<br/>";
            //Dosya yüklemesinde oluşan hatanın kodunu verir.
            print_r($_FILES['dosya']['error']);
            echo "<br/>";
        }
    }
}
?>
```

```
< <body>
    <!-- Formumuzda enctype özelliğini mutlaka belirtmemiz gerekiyor -->
    <form action="islem.php" method="POST" enctype="multipart/form data">
        <!-- Birden fazla dosya seçimi yapabilmemiz için multiple özelliğini ekliyoruz -->
        <input type="file" name="dosya[]" multiple>
        <!-- Yüklenecek dosyaları dizi olarak iletiyoruz -->
        <input type="submit" value="Dosya Yükle" name="dosyayukle[]">
    </form>
</body>
```

Fonksiyonlar

Fonksiyonu tam olarak anlamamız için şimdi hazır bir fonksiyonun kullanımını görelim. Aşağıda yer alan örnekte hazır **rand()** fonksiyonunu inceleyeceğiz. Bu fonksiyon bizim belirlediğimiz bir aralıkta sayı üretmektedir.

```
<?php
    echo rand(1,10);
    # CEVAP
    # 5 // Her yenilemede 1 ile 10 arasında rastgele bir sayı üretir.
?>
```

1. Fonksiyon Oluşturma

Fonksiyon oluşturmak için şu söz dizimini kullanırız.

```
<?php
    function fonksiyon_adi() (parametre1,parametre2...){
        # işlemler
    }
?>
```

2. Parametresiz Fonksiyon Oluşturma

Aşağıdaki örnekte herhangi bir parametre tanımlamadan çağırdığımızda içindeki kod bloklarını çalıştıran bir fonksiyon yazdık.

Bu fonksiyon geriye herhangi bir değer döndürmediği için echo deyimi ile fonksiyonu ekrana yazdırmalıyız.

```
<?php
    function ciftSayilar(){
        for ($i=0; $i <=10 ; $i+=2){
            echo $i." ";
        }
    }
    echo ciftSayilar();
    # CEVAP
    # 0 2 4 6 8 10
?>
```

3. Parametrelili Fonksiyon Oluşturma

Dışarıdan veri alarak bu verileri işleyip sonuç üretebiliriz.

```
<?php
    function topla($sayi1,$sayi2){
        echo $sayi1+$sayi2;
    }
    echo topla(5,6);
    # CEVAP
    # 11
?>
```

```
<?php
    function karsilama($kadi = 'Misafir'){
        echo 'Merhaba ' . $kadi . ' ';
    }
    karsilama();
    echo "<br/>";
    karsilama("kodlab");
    # CEVAP
    # Merhaba Missafir
    # Merhaba kodlab
?>
```


4. Return Geriye Değer Döndürme

```
<?php
function kare($sayi){
    return $sayi * $sayi;
}
kare(2);
# CEVAP
# BOŞ
$sonuc=kare(2);
echo "Girdiğiniz sayının karesi : ".$sonuc;
# CEVAP
# Girdiğiniz sayının karesi : 4
?>
```

5. Fonksiyon Kontrolü

function_exists() fonksiyonu ile bir fonksiyonun tanımlı olup olmadığını kontrol edebiliriz.

6. Global Değişken Kullanımı

Fonksiyonlar dışarıdan gelen değişken değerlerini direkt olarak kullanamaz, aynı şekilde kendi içlerinde oluşturulan değişkenlerin değerleri de dışarıdan okunamaz. Bunu bir örnekle anlamlandıralım.

```
<?php
$sayi=rand(1,5);
function bul() {
    echo "rand'ın ürettiği sayı...";
}
bul();
# CEVAP
# rand'ın ürettiği sayı...
?>
```

```
<?php
$sayi=rand(1,5);
function bul(){
    //global olarak tanımlıyoruz.
    global $sayi;
    echo "rand fonksiyonunun ürettiği sayı...: ".$sayi;
}
bul();
# CEVAP
# rand fonksiyonunun ürettiği sayı...: 2
?>
```

Soldaki örnekte uyguladığımız fonksiyon içerisinde malesef **\$sayi** değişkeninin değerini okuyamadık. Bu işlemi uygulayabilmemiz için dışarıdaki değişkeni fonksiyon içerisinde global değişken olarak tanımlamamız gerekiyor.

7. Referans Gönderimli Değer Atama

Ön tanımlı olarak, fonksiyon referansları, değerleriyle aktarılırlar. Referansın değeri fonksiyon içinde değiştirildiğinde işlevin çağrıldığı yerdeki değeri bundan etkilenmez. Fonksiyonun çağrıldığı yerdeki değerinin de değişmesini istiyorsak **referans değiştirge (&)** kullanmalıyız.

```
<?php
$sayi=59;
function referans(&$sayi){
    # sayi değişkeninin içeriğine 21 atıyoruz & işareti ile
    # referans atadığımız için artık atanan değer kullanımda olacak.
    $sayi = 21;
}
referans($sayi);
echo $sayi;
# CEVAP
# 21
?>
```

8. İç İçe Fonksiyon

Fonksiyon içerisinde oluşturduğumuz başka bir fonksiyonu üst fonksiyon işleme sokulmadan çağırıp kullanamayız.

```
<?php
function cagir(){
    function yaz(){
        echo "Ben fonksiyonun içinde ki fonksiyonum";
    }
}
cagir(); //Bu fonksiyon çağrılmadan alt fonksiyona ulaşamaz.
yaz();
?>
```

Uyguladığımız örnekte **yaz()** fonksiyonu, **cagir()** fonksiyonunun içinde olduğundan öncelikle **cagir()** fonksiyonu çağrılmalıdır.

9. Recursive (Kendini Tekrar Eden) Fonksiyonlar

Fonksiyonların içinde tekrar kendisini çağırarak kullanabiliriz. Aşağıda uyguladığımız örnekte fonksiyona gönderilen değer içinde sağlanan koşula ulaşana kadar fonksiyon tekrar tekrar çalıştırılıyor.

```
<?php
function tekrar($sayi){
    if ($sayi<5){
        echo $sayi++;
        tekrar($sayi);
    }
}
tekrar(1);
# CEVAP
# 1234
?>
```

10. Tanımlı Tüm Fonksiyonları Listeleme

get_defined_function() fonksiyonu PHP’de tanımlanmış tüm fonksiyonları dizi şeklinde döndürür.

11. Fonksiyonlarda Varsayılan Değer Ataması

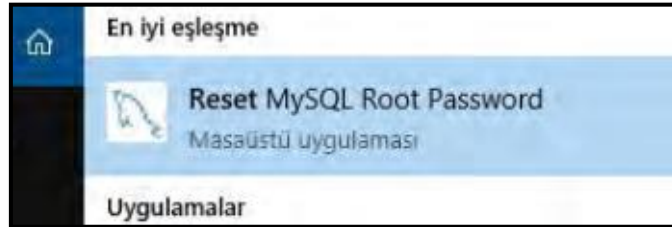
```
<?php
function yaz($ad,$soyad="Soy isim girilmedi!"){
    return $ad." ".$soyad;
}
echo yaz("Emrah",);
# CEVAP
# Emrah Soy isim girilmedi!
?>
```

MySQL VERİTABANI

Veritabanı (Database) birbiriyle ilişkili bilgilerin belirli bir düzene göre depolandığı alanlardır.

1. phpMyAdmin Şifremi Unuttum

AppServ kurulumunda yazdığınız şifreyi hatırlamıyorsanız paket ile birlikte gelen "Reset MySQL Root Password" uygulaması ile şifrenizi basit bir şekilde sıfırlayabilirsiniz.



"Reset MySQL Root Password" uygulamasını çalıştırdığınızda en az 8 karakterden oluşan bir şifre tanımlamanız gerekiyor



Şifrenizi girip enter tuşuna basın ve işlemlerin uygulamasını bekleyin. Kullanıcı adınız root ve yeni şifreniz ile birlikte giriş yapabilirsiniz.

2. Veri Tipleri

Sabit uzunluklu veriler için **CHAR**, **NCHAR**, **VARCHAR** ve **NVARCHAR** değişkenleri kullanılır.

Örneğin telefon numarası, TC kimlik numarası gibi.

isim **CHAR(10)**

Herbir karakteri 1 byte olan ve en fazla 10 karakter içerebilen metinsel bir ifadeyi tutabilir.

isim **NCHAR(10)**

Her bir karakteri 2 byte olan ve en fazla 10 karakter içerebilen metinsel bir ifadeyi tutabilir. Ayrıca uluslararası kodlar (UNICODE) içeren metinsel ifadeler tutulabilir. CHAR ile NCHAR arasındaki fark budur.

isim **CHAR(10)** -----> Data, 'DENEME ' şeklinde tutulur ve 10 byte yer kaplar.

isim **NCHAR(10)** -----> Data, 'DENEME ' şeklinde tutulur ve 20 byte yer kaplar.

isim **VARCHAR(10)** ---> Data, 'DENEME' şeklinde tutulur ve 6 byte yer kaplar. (DENEME 6 karakterden oluşuyor. VAR yazdığımızdan dolayı geriye kalan 4 karakter CHAR tipindeki gibi boşlukla tamamlanmak yerine, görmezden geliniyor. Bu sayede veri gereğinden fazla yer kaplamamış oluyor.)

isim **NVARCHAR(10)** --> Data 'DENEME' şeklinde tutulur ve 12 byte yer kaplar.

Date: Yıl, ay ve gün tutar.

Datetime: Yıl, ay, gün, saat, dakika ve saniyeyi tutar.

Açılır Listedeki "utf8_turkish_ci" seçerek dil kodlamasını Türkçe olarak değiştirebiliriz.

PDO (PHP DATA OBJECT - PHP Veri Nesneleri)

PDO, PHP'nin OOP desteği sayesinde, PHP geliştiricileri tarafından yazılmış veritabanı işlemleri için ortak bir yapı sunar.

PDO sınıfını kullanarak, PDO için oluşturulmuş veritabanı sistemlerinde veri ekleme, seçme, güncelleme vb. işlemlerini yapmaya imkan verir. PDO içerisinde bulunan ön hazırlık sorgu yapısı sayesinde **SQL Injection (veri tabanına dayalı uygulamalara saldırmak için kullanılan bir saldırı tekniğidir.)** gibi istemeyen durumlar içinde çözüm sunar.

1. PDO Kurulumu

PDO, PHP 5 ile geldiği için güncel bir php sürümü yüklemeniz yeterlidir.

2. PDO MySQL Veritabanı Bağlantısı

MySQL veritabanımıza bağlantı kurmak, kullanacağımız en temel kod yapısı aşağıdaki gibidir.

```
<?php
$db = new PDO('mysql:host=localhost; dbname=db_isim', "kullanici_adi", "sifre");
?>
```

host kısmında sunucumuzun adını, **dbname** kısmında veritabanımızın adını belirtiyoruz. Daha sonra da veritabanımıza bağlanırken kullandığımız **kullanıcı adı** ve **şifremizi** giriyoruz. **phpmyAdmin'e** girişte kullandığınız kullanıcı adı ve şifreyle aynıdır.

try catch yapısını kullanarak oluşabilecek muhtemel hataları da bu sayede görebiliriz.

PDO nesnesinin ömrü boyunca bağlantı etkin kalır. Bağlantıyı kapatmak için nesneyi tutan değişkene **NULL** değerini atamalıyız. Bunu siz yapmazsanız, PHP betiğiniz sonlandığında bunu sizin yerinize yapacaktır.

```
<?php
try{
    $db=new PDO("mysql:host=localhost; dbname=e-ticaret; charset=utf8", 'root', '12345678');
    echo "Veritabanı bağlantısı başarılı";
}
catch(PDOException $e){
    echo $e->getMessage();
}
# Bağlantı kapatılıyor!
$db = NULL;
?>
```

3. Kullanılabilir PDO Sürücülerini Görme

PDO::getAvailableDrivers() ile sunucumuzda mevcut kullanılabilir sürücülerini görebiliriz. Sonucu dizi şeklinde döndürecektir.

```
<?php
print_r(PDO::getAvailableDrivers());
# CEVAP
# Array ( [0] => mysql [1] => sqlite )
?>
```

4. prepare() Metodu

Eğer SQL sorgularımızda dışarıdan veri girişi mevcut ise mutlaka **prepare()** metodu kullanılmalıdır. Bu metod **SQL Injection'lardan** kurtulmak için büyük bir güvenlik sağlar. Hazırlanan SQL deyimi **execute()** metodu ile çalıştırılır.

prepare(), execute(), bindParam(), bindValue() ve binColumn() metodları ile birlikte çalışır.

5. execute() Metodu

execute() metodu, **prepare()** metodu ile hazırlanan bir SQL deyimini çalıştırır.

SQL sorguları oluşturulurken dışarıdan gelen verileri **soru işareti** veya önünde **üst üste iki nokta** yazarak oluşturacağımız bir isim ile sorgumuzda uygun yerlere dahil edebiliriz.

a. SELECT

Soru işaretli girdi değiştirge kullanımı: kitap_id'si 3 olan kaydı soru işareti değiştirge kullanarak seçiyoruz. <pre><?php require_once 'db.php'; \$id=3; \$sql=\$db->prepare("SELECT * FROM kitap WHERE kitap_id=?"); \$sql->execute([\$id]); print_r(\$sql->fetch()); ?></pre>	İsimli girdi değiştirge kullanımı: kitap_id'si 3 olan kaydı, isimli değiştirge kullanarak seçiyoruz. <pre><?php require_once 'db.php'; \$id=3; \$sql=\$db->prepare("SELECT * FROM kitap WHERE kitap_id=:id"); \$sql->execute(['id' => \$id]); print_r(\$sql->fetch()); ?></pre>
kitap_id=? şeklinde kitap_id alanı için dışarıdan veri alacağımız alan belirlenmiştir. execute() ile dışarıdan verileri sorgumuza dahil ederken sorgumuzda soru işaretlerini (?) kullandığımız sıraya dikkat etmemiz gerekmektedir.	Bu yöntemin soru işaretinden farkı sorgumuzda isimleri tanımlamış olacağımızdan execute() içerisinde sırası önem taşımamaktadır. Sadece tanımladığımız isimle uygun bir anahtar ile diziyi oluşturmamız yeterlidir.

b. INSERT

HTML formdan gelen bir **\$_POST** dizisini simule etmek için kitap tablomuza ekleyeceğimiz veriyi dizi şeklinde **\$_POST** değişkenine aktaralım. Kayıt işlemlerinde bunu kullanacağız.

Soru işaretli girdi değiştirge kullanımı: <pre><?php require_once 'db.php'; \$_POST=[yazar_id => 1, kitap_ad => "C#", kitap_detay => "İçerik", kitap_fiyat => 50]; \$sql=\$db->prepare("INSERT INTO kitap SET yazar_id=?,kitap_ad=?,kitap_detay=?,kitap_fiyat=?"); \$sonuc=\$sql->execute([\$_POST['yazar_id'], \$_POST['kitap_ad'], \$_POST['kitap_detay'], \$_POST['kitap_fiyat']]); if(\$sonuc){ echo "Başarılı"; } else{ echo "Başarısız"; } ?></pre>

İsimli girdi değiştirge kullanımı:

```
<?php
require_once 'db.php';
$_POST=[ yazar_id => 1,
        kitap_ad => "C#",
        kitap_detay => "İçerik",
        kitap_fiyat => 50
];
$sql=$db->prepare("INSERT INTO kitap SET
yazar_id=:yazar_id,kitap_ad=:kitap_ad,kitap_detay=:kitap_detay,kitap_fiyat=:kitap_fiyat");
$sonuc=$sql->execute([ 'yazar_id' => $_POST['yazar_id'],
                       'kitap_ad' => $_POST['kitap_ad'],
                       'kitap_detay' => $_POST['kitap_detay'],
                       'kitap_fiyat' => $_POST['kitap_fiyat']
]);
if($sonuc){
    echo "Başarılı";
}
else{
    echo "Başarısız";
}
?>
```

c. UPDATE

Soru işaretli girdi değiştirge kullanımı:

```
<?php
require_once 'db.php';
$kitap_id=9; # Silinecek kaydın ID'si
$sql=$db->prepare("DELETE FROM kitap WHERE kitap_id=?");
$sonuc=$sql->execute([$kitap_id]);
if($sonuc){
    echo "Başarılı";
}
else{
    echo "Başarısız";
}
?>
```

İsimli girdi değiştirge kullanımı:

```
<?php
require_once 'db.php';
$_POST=[ kitap_id => 8, # Güncellenecek Bilgiler
        kitap_ad => "Yeni C#11",
        kitap_detay => "İçerik Güncel",
        kitap_fiyat => 55
];
$sql=$db->prepare("UPDATE kitap SET
kitap_ad=:kitap_ad,kitap_detay=:kitap_detay,kitap_fiyat=:kitap_fiyat WHERE kitap_id=:kitap_id");
$sonuc=$sql->execute([ 'kitap_ad' => $_POST['kitap_ad'],
                       'kitap_detay' => $_POST['kitap_detay'],
                       'kitap_fiyat' => $_POST['kitap_fiyat'],
                       'kitap_id' => $_POST['kitap_id']
]);
if ($sonuc) {
    echo "Başarılı";
}
else{
    echo "Başarısız";
}
?>
```

6. Diziyi Tek Seferde Kaydetme

? soru işaretli girdi değiştirge kullandığımızda gelen verileri **array_value()** fonksiyonunu kullanarak tek seferde çalıştırmak için **execute()** metoduna gönderebiliriz.

```
<?php
require_once 'db.php';
$_POST=[ yazar_id => 1,
        kitap_ad => "C#",
        kitap_detay => "İçerik",
        kitap_fiyat => 50
];
$sql=$db->prepare("INSERT INTO kitap SET yazar_id=?,kitap_ad=?,kitap_detay=?,kitap_fiyat=?");
$sonuc=$sql->execute(array_values($_POST));
if($sonuc){
    echo "Başarılı";
}
else{
    echo "Başarısız";
}
?>
```

Burada dikkat etmemiz gereken **\$_POST** dizisi içerisinde sıralanan verinin SQL sorgusunda sıralanan sütun isimleri ile indis numaraları denk gelecek şekilde sıralanarak eşleşmenin doğru yapılmasıdır.

7. bindparam() Metodu

bindparam() metodu, **prepare()** metodu içindeki özel parametrelere (? veya :isimli_parametre) verileri güvenli biçimde yerleştirir.

Metodun ilk parametresi ön hazırlık sorguya göre değer alır, ikinci parametresi dışarıdan alınacak değerin değişkenini belirtir, üçüncü parametre ise verinin tipini belirtmek için kullanılır. Metodun üçüncü parametresine girilen;

PDO::PARAM_INT	Sayısal veri
PDO::PARAM_STR	Metinsel veri
PDO::PARAM_LOB	Binary veri
PDO::PARAM_INPUT_OUTPUT	Saklı yordam girdi/çıkıktı verisi
PDO::PARAM_NULL	NULL veri tipi olduğunu belirtir

Ayrıca çok kullanılmayan dördüncü uzunluk parametresi ve beşinci veritabanı sistemi ayar parametresi mevcuttur.

a. SELECT

İsimli girdi değiştirge kullanımı:

```
<?php
require_once 'db.php';
$id = 1;
$ad = 'Php';
$sql = $db->prepare("SELECT * FROM kitap
WHERE yazar_id=:id and kitap_ad=:ad");
$sql->bindParam(':id', $id, PDO::PARAM_INT);
$sql->bindParam(':ad', $ad, PDO::PARAM_STR);
$sql->execute();
print_r($sql->fetch());
?>
```

Soru işaretli girdi değiştirge kullanımı:

```
<?php
require_once 'db.php';
$id = 1;
$ad = 'Php';
$sql = $db->prepare("SELECT * FROM kitap
WHERE yazar_id=? and kitap_ad=?");
$sql->bindParam(1, $id, PDO::PARAM_INT);
$sql->bindParam(2, $ad, PDO::PARAM_STR);
$sql->execute();
print_r($sql->fetch());
?>
```

8. bindValue() Metodu

Metot **bindParam()** ile aynı işleve sahiptir. Ancak bindParam metodunda çok kullanılmayan dördüncü ve beşinci parametreler yoktur.

bindParam() metodundan farkı nedir?

bindParam() ile sadece değişkenleri paslayabilirsiniz.
Yani sadece **bindParam(":name", \$name)** şeklindekiler kabul edilecektir.
bindValue()'de ise hem değişken hem de doğrudan veri paslayabilirsiniz.
İster **bindValue(":title", \$title)**
isterseniz **bindValue(":title", "başlık")**

```
<?php
require_once 'db.php';
# bindParam() metodu
$id = 1;
$ad = 'Php';
$sql = $db->prepare("SELECT * FROM kitap WHERE yazar_id=:id ");
$sql->bindParam(':id', $id, PDO::PARAM_INT);
# Girdi değerlerini (id ve ad) değiştiriyoruz.
$id=3;
$ad="Python";
$sql->execute();
print_r($sql->fetch());

# bindValue() metodu
$id = 1;
$ad = 'Php';
$sql = $db->prepare("SELECT * FROM kitap WHERE yazar_id=:id and kitap_ad=:ad");
$sql->bindValue(':id', $id, PDO::PARAM_INT);
$sql->bindValue(':ad', $ad, PDO::PARAM_STR);
# Girdi değerlerini değiştirsek bile ilk çalıştırıldığı anda ki değerleri kullanılacak.
$id=3;
$ad="Python";
$sql->execute();
print_r($sql->fetch());

# CEVAP
# bindParam() Metodu
/* Array ( [kitap_id] => 7 [0] => 7 [yazar_id] => 3 [1] => 3 [kitap_ad] => Python [2] =>
Python [kitap_detay] => Python Kitap Detayları [3] => Python Kitap Detayları [kitap_fiyat] =>
45.00 [4] => 45.00 ) */

# bindValue() Metodu
/* Array ( [kitap_id] => 1 [0] => 1 [yazar_id] => 1 [1] => 1 [kitap_ad] => Php [2] => Php
[kitap_detay] => Php Kitap Detayları [3] => Php Kitap Detayları [kitap_fiyat] => 50.00 [4] =>
50.00 ) */
?>
```

Yukarıdaki sorguları incelediğimizde **bindParam()** metodu, **execute()** metodu çalıştırıldığı andaki yeni değerleri kullandı. Fakat **bindValue()** metodu ilk atanan çalıştırıldığı andaki değerleri kullandı.

9. bindColumn() Metodu

prepare() metodu ile hazırlanan bir girdi değıştirgesini belirtilen değışkenle ilişkilendirir.

Soru işaretili girdi değıştirge kullanımı:

```
<?php
require_once 'db.php';
$id=1;
$ad='Php';
$sql=$db->prepare("SELECT * FROM kitap WHERE kitap_id=? AND kitap_ad=?");
$sql->execute(array($id, $ad));
$sql->bindColumn(1,$kitap_numara);
$sql->bindColumn(2,$kitap_isim);
$sql->fetch(PDO::FETCH_BOUND);
echo $kitap_numara." numaralı kitabın ismi ".$kitap_isim;
# CEVAP
# 1 numaralı kitabın ismi Php
?>
```

10. errorInfo()

Çalıştırılan SQL deyimindeki son işlemle ilgili hata bilgisini dizi olarak döndürür.

```
<?php
require_once 'db.php';
$sql = $db->prepare('SELECT ROM kitap');
//Sorgumuzda hatalı yazım yapıyoruz.
$sql->execute();
echo "<pre>";
print_r($sql->errorInfo());
# CEVAP
/*
    Array
    (
        [0] => 42000
        [1] => 1064
        [2] => You have an error in your SQL syntax; check
the manual that corresponds to your MySQL server version for
the right syntax to use near 'FROM kitaps' at line 1 )
    */
?>
```

11. columnCount()

Çalıştırılan SQL deyimi sonucunda oluşan sonuç kümesindeki sütun sayısını tam sayı olarak döndürür.

```
<?php
require_once 'db.php';
$id=3;
$sql=$db->prepare("SELECT * FROM kitap");
$sql->execute();
echo "Bir satırda".$sql->columnCount()." sütun bulunmakta";
# CEVAP
# Sorgu sonucu oluşan satırda 5 sütun bulunmakta
?>
```

12. lastInsertId()

Eklenen son satırın **ID (kimlik numarasını)** verir. **INSERT** işlemi ile eklediğimiz kaydın kimlik numarası **A_I (auto increment)** olarak ayarlandığı durumda eklenen kayda verilen kimlik numarasını **lastInsertId()** metodu ile elde edebiliriz.

```
<?php
require_once 'db.php';
$_POST=[ yazar_id => 1,
        kitap_ad => "C#",
        kitap_detay => "İçerik",
        kitap_fiyat => 50
];
$sql=$db->prepare("INSERT INTO kitap SET yazar_id=?,kitap_ad=?,kitap_fiyat=?,kitap_detay=?");
$sonuc=$sql->execute(array_values($_POST));
echo "Eklenen Kaydın ID Numarası...: ".$db->lastInsertId();
?>
```

13. rowCount()

Çalıştırılan DELETE, INSERT veya UPDATE SQL deyimlerinden etkilenen satır sayısını döndürür.

```
<?php
require_once 'db.php';
$id=3;
$sql=$db->prepare("SELECT * FROM kitap");
$sql->execute();
echo "Kitap tablosunda ".$sql->rowCount()." kayıt bulunmakta";
# CEVAP
# Kitap tablosunda 18 kayıt bulunmakta
?>
```

DELETE SQL sorgu sonucunu test edelim.

```
<?php
require_once 'db.php';
# Silinecek kaydın ID'si
$yazar_id=1;
$sql=$db->prepare("DELETE FROM kitap WHERE yazar_id=?");
$sonuc=$sql->execute([$yazar_id]);
echo "Kitap tablosundan ".$sql->rowCount()." kayıt silindi";
# CEVAP
# Kitap tablosundan 17 kayıt silindi
?>
```

Kitap tablosunda **yazar_id**'si 1 olan 17 kayıt silindi ve etkilenen satırı **rowCount()** ile alarak ekrana yazdırdık.

14. fetch() Metodu

prepare() metodu ile hazırlanmış çalıştırılan bir SQL sorgusu sonucunda ilk satırı listelememizi sağlar. Satırın nasıl döndürüleceği alım tarzı değiştirilmesinde **PDO::FETCH_*** sabitleriyle belirlenir. **PDO::FETCH_BOTH** sabiti ön tanımlıdır.

PDO::FETCH_ASSOC Sütun isimlerine göre indisli bir dizi döndürelim. <pre><?php require_once 'db.php'; \$id=3; \$sql=\$db->prepare("SELECT * FROM kitap"); \$sql->execute(); echo "<pre>"; print_r(\$sql->fetch(PDO::FETCH_ASSOC)); # CEVAP /* Array ([kitap_id] => 5 [yazar_id] => 2 [kitap_ad] => Php OOP [kitap_detay] => Php OOP Kitap Detayları [kitap_fiyat] => 40.00) */ ?></pre>	PDO::FETCH_NUM Sütun numaralarına göre indislenmiş bir dizi döndürelim. <pre><?php require_once 'db.php'; \$id=3; \$sql=\$db->prepare("SELECT * FROM kitap"); \$sql->execute(); echo "<pre>"; print_r(\$sql->fetch(PDO::FETCH_NUM)); # CEVAP /* Array ([0] => 1 [1] => 1 [2] => Php [3] => Php Kitap Detayları [4] => 50.00) */ ?></pre>
--	---

15. fetchAll() Metodu

prepare() metodu ile hazırlanmış çalıştırılan bir SQL sorgusu sonucunda tüm satırları listelememizi sağlar. Satırın nasıl döndürüleceği alım tarzı değiştirilmesinde **PDO::FETCH_*** sabitleriyle belirlenir. **PDO::FETCH_BOTH** sabiti ön tanımlıdır. **fetch()** ve **fetchAll()** metodu arasındaki fark; **fetch()** metodu tek satır, **fetchAll()** metodu tüm satırları listelememizi sağlar.

16. fetchColumn() Metodu

prepare() metodu ile hazırlanmış çalıştırılan bir SQL sorgusu sonucunda ilk satırdan, tek bir sütunu dizi olarak döndürür. **Sütunun numarası**: İlk sütunun indisi yani **0'dır**. Değiştirilmede hiçbir değer belirtilmemişse ilk sütunun değeri döner.

```
<?php
require_once 'db.php';
$id=3;
$sql=$db->prepare("SELECT * FROM kitap");
$sql->execute();
echo "<pre>";
print_r($sql->fetchColumn(PDO::FETCH_ASSOC));
# CEVAP
# Php
?>
```

17. Döngüler ile Verileri Listeleme

while() döngüsünü kullanarak kitap ve yazar tablomuzda yer alan verileri listeleyelim.

```
<?php
    require_once 'db.php';
    $sql=$db->prepare("SELECT * FROM kitap");
    $sql->execute();
    while ($list=$sql->fetch(PDO::FETCH_ASSOC)){
        echo $list['kitap_ad']."<br/>";
    }
    # CEVAP
    /*Kitap tablosunda yer alan kitap isimlerini
    listeletmiş olduk. */
    # Php
    # Bootstrap
    # Javascript
    # JQuery
    # Php OOP
    # Laravel 5
    # Python
?>
```

foreach() döngüsünü kullanarak kitap ve yazar tablomuzda yer alan verileri listeleyelim.

```
<?php
    require_once 'db.php';
    $sql=$db->prepare("SELECT * FROM kitap");
    $sql->execute();
    foreach ($sql as $list){
        echo $list['kitap_ad']."<br/>";
    }
    # CEVAP
    /*Kitap tablosunda yer alan kitap isimlerini
    listeletmiş olduk. */
    # Php
    # Bootstrap
    # Javascript
    # JQuery
    # Php OOP
    # Laravel 5
    # Python
?>
```

18. exec() Metodu

exec() metodu, SQL komutlarını (INSERT, DELETE, UPDATE) çalıştırır ve geri dönüş değeri olarak eklenen, silinen veya güncellenen kayıt sayısını verir. **Metodun geri dönüş değeri sayısal bir değer olduğundan sonuç döndüren veri seçme-çekme (SELECT, SHOW) işlemlerinde kullanılmaz.** Etkilenen satır yoksa 0 değerini döndürür.

exec() metodu *SELECT* deyiminin sonucunu döndürmez. *INSERT*, *UPDATE*, ve *DELETE* SQL deyimlerinde kullanılır ve etkilenen satırı döndürür.

19. query() Metodu

query() metodu SQL deyimini (SELECT, INSERT, UPDATE, DELETE vb.) çalıştırır ve sonucu **PDOStatement** nesnesi olarak döndürür. Ekstra ikinci parametreyle **fetch()** metodunda kullandığımız sabitler belirtilerek sonuç biçim belirtilebilir. **PDO::FETCH_BOTH** ön tanımlıdır. SQL sorgusu değer döndürüyorsa (veri çekme varsa), **PDOStatement** sınıfındaki **fetch**, **fetchAll**, **fetchColumn**, **fetchObject** metotları kullanılabilir.

Varsayılan PDO::FETCH_BOTH hem sütun isimlerine hem de sütun numaralarına göre indislenmiş dizi.

```
<?php
    require_once 'db.php';
    $sql=$db->query("SELECT * FROM Kitap");
    foreach($sql as $key){
        echo $key[2];
    }
?>
```

Sütun isimlerine göre indisli dizi:

```
<?php
    require_once 'db.php';
    $sql=$db->query("SELECT * FROM Kitap",PDO::FETCH_ASSOC);
    foreach($sql as $key){
        echo $key['kitap_ad'];
    }
?>
```

20. quote() Metodu

exec() ve **query()** metodları SQL komutlarına dışarıdan değer alıyorsa **quote()** metodu ile istenmeyen durumlara karşı koruma sağlanır. **SQL Injection'a** karşı tehlikeli olabilecek söz dizimlerini yoksayar. Ayrıca SQL satırında dışarıdan gelen verileri ' ' tek tırnak içine almamıza gerek kalmaz. Bu işlemi **quote()** metodu gerçekleştirir.

```
<?php
require_once 'db.php';
$metin='C:\files';
echo $db->quote($metin);
# CEVAP
# 'C:\\files'
/* quote() metodu veriyi ' ' tek tırnak içine aldı ve
içerisinde yer alan \ işaretini (önceledi) yoksaydı. */
?>
```

POST dizi değişkenimizde HTML form'dan geldiğini varsaydığımız değerler var. **kitap_ad** kısmında kullanıcı bir **JS** kodu girdiğinde ve bu değeri biz filtrelemeden SQL'e kaydettiğimizde, çağırma işleminde oluşabilecek durumu aşağıdaki kodu çalıştırarak görebilirsiniz.

```
<?php
require_once 'db.php';
$_POST=[ kitap_id => 1,
          kitap_ad => "<script>alert('SQL
Injection')</script>",
          kitap_detay => "İçerik",
          kitap_fiyat => 50
];
echo $_POST['kitap_ad'];
# CEVAP
# SQL Injection diye bir alert butonu
oluşturur.
?>
```

Bu sorunu giderelim.

```
<?php
require_once 'db.php';
$_POST=[ kitap_id => 1,
          kitap_ad => "<script>alert('SQL
Injection')</script>",
          kitap_detay => "İçerik",
          kitap_fiyat => 50
];
$_POST['kitap_ad']=$db->quote($_POST['kitap_ad']);
echo $_POST['kitap_ad'];
?>
```

Ayrıca tüm HTML formundan gelen tüm dizi elemanlarını tek tek işlemek yerine **array_map()** fonksiyonu ile tek seferde **quote()** metoduna yönlendirebiliriz.

```
<?php
require_once 'db.php';
$_POST=[kitap_id => 1,
          kitap_ad => "<script>alert('SQL Injection')</script>",
          kitap_detay => "İçe'r'ik",
          kitap_fiyat => 50
];
$_POST=array_map(array($db,'quote'), $_POST);
echo "<pre>";
print_r($_POST);
# CEVAP
/*
Array
(
    [kitap_id] => '1'
    [kitap_ad] => ''
    [kitap_detay] => 'İçe\'r\'ik'
    [kitap_fiyat] => '50'
)
*/
?>
```

21. Ön Tanımlı Sabitler

PDO metodları ile çalışırken aşağıdaki tabloda yer alan **PDO::PRAM...** sabitlerini kullanacağız.

Sabitler PDO için tanımlanmış olup sadece PHP içinde derlenmiş olması veya çalışma anında işler biçimde yüklenmesi hâlinde kullanılabilir.

Sabit	Açıklama
PDO::PARAM_BOOL	Mantıksal veri türünü ifade eder.
PDO::PARAM_NULL	SQL NULL veri türünü ifade eder.
PDO::PARAM_INT	SQL INTEGER veri türünü ifade eder.
PDO::PARAM_STR	SQL CHAR, VARCHAR ve benzeri dizge veri türlerini ifade eder.
PDO::PARAM_STR_CHAR	Dizgenin normal karakter kümesini kullanacağını belirtir.
PDO::PARAM_LOB	SQL büyük nesne veri türünü ifade eder.
PDO::PARAM_STMT	Kayıt kümesi (recordset) türünü ifade eder. Her sürücü desteklemez.
PDO::PARAM_INPUT_OUTPUT	Değiştirgenin bir saklı yordam için bir girdi/çıkış değiştirgesi olduğunu gösterir. Bu değeri bir PDO::PARAM_* veri türü ile bit seviyesinde VEYAlamanız gerekir.
PDO::FETCH_LAZY	Yöntemin, sonuç kümesindeki her satırı, değişken isimlerinin sütun isimlerine karşılık geldiği bir nesne olarak döndüreceğini belirtir. PDO::FETCH_LAZY sütun isimlerine nesnenin değişkenleri olarak erişilebilmesini sağlar. PDOStatement::fetchAll() içinde geçersizdir.
PDO::FETCH_ASSOC	Sütun isimlerine göre indisli bir dizi döner. Eğer sonuç kümesi aynı isimde birden fazla sütun içeriyorsa her sütun için tek bir değer döner.
PDO::FETCH_NAMED	Yöntemin, sonuç kümesindeki her satırının, sütun isimleriyle indislenmiş bir dizi olarak döndüreleceğini belirtir. Eğer sonuç kümesi aynı isimde birden fazla sütun içeriyorsa her sütun için bir değerler dizisi döner.
PDO::FETCH_NUM	Sütun numaralarına göre indislenmiş bir dizi döner. İlk sütunun indisi 0'dır
PDO::FETCH_BOTH	Hem sütun isimlerine hem de sütun numaralarına göre indislenmiş bir dizi döner. İlk sütunun indisi 0'dır.
PDO::FETCH_OBJ	Yöntemin, sonuç kümesindeki her satırın, özellik isimlerinin sütun isimlerine karşılık geldiği bir nesne olarak döndüreceğini belirtir.
PDO::FETCH_BOUND	Sütun değerlerini PDOStatement::bindColumn() ile ilişkilendirilmiş PHP değişkenlerine atar ve TRUE döndürür.
PDO::FETCH_COLUMN	Yöntemin, sonuç kümesindeki sonraki satırdan istenen tek bir sütunu döndüreceğini belirtir.
PDO::FETCH_CLASS	Yöntemin, sınıf özellikleri sütun isimleriyle eşlenerek elde edilen sınıfın yeni bir örneğini döndüreceğini belirtir.
PDO::FETCH_INTO	Yöntemin, sınıf özellikleri sütun isimleriyle eşlenerek elde edilen sınıfın mevcut bir örneğini güncelleyeceğini belirtir.
PDO::FETCH_FUNC	Verinin anında işlenmesi yoluyla tamamen özelleştirmeye izin verir (sadece PDOStatement::fetchAll() içinde geçerlidir).
PDO::FETCH_GROUP	Grup değer olarak döner. Normal olarak PDO::FETCH_COLUMN veya PDO::FETCH_KEY_PAIR ile birlikte.
PDO::FETCH_UNIQUE	Sadece eşsiz değerleri getirir.
PDO::FETCH_KEY_PAIR	İlk sütunun anahtar, diğer sütunların ise birer değer olduğu bir dizi döner.
PDO::FETCH_CLASSTYPE	Sütun ismini ilk sütunun değerinden belirler.
PDO::FETCH_SERIALIZE	Nesnenin dizgeleştirilmiş olarak döndürülmesi dışında PDO::FETCH_INTO gibidir.

E-TİCARET

<p>ayar_title</p> <p>ayar_description</p>	
ayar_keywords	
ayar_analytic	Sitenin istatistiklerini tutacağız.
ayar_zopim	<p>Müşterilere web siteleri, mobil uygulamalar ve mesajlaşma hizmetleri üzerinden hızlı ve etkili destek hizmeti sağlar.</p>
<p>ayar_smtpost</p> <p>ayar_smtpassword</p> <p>ayar_smtpport</p>	<p>SMTP (Simple Mail Transfer Protocol), bir e-posta göndermek için sunucu ile istemci arasındaki iletişim şeklini belirleyen protokoldür. Farklı işletim sistemleri için geliştirilmiş <u>e-posta</u> protokolleri bulunmaktadır.</p>

Kullanılan – Yapılan İşlemler

1. Ck Editör kullanıldı.
2. Font Awesome kullanıldı.

.htaccess Dosyası Nedir?

htaccess Dosyası (Hypertext Access), Apache gibi ağ sunucusu tarafından kullanılan web alanları üzerinde ayar değişimleri yapmanızı sağlayan dosyadır. Böylece sitede her türlü değişim, yetki ve kısıtlama işlemleri yapılabilir.

.htaccess dosyası yardımıyla örneğin bir klasöre parola koruması ekleyebilir ve yetkilendirme sağlayabilirsiniz. Bu dosya uygulamasının en güzel yanlarından birisi de herhangi bir kod bilgisine ihtiyaç duyulmaz.

1. URL Yönlendirme (Rewriting URL)

URL yönlendirme fonksiyonu ile uzun adresler daha kısa, hatırlanabilir adreslere veya farklı adreslere yönlendirilebilir. URL yönlendirme için aşağıdaki kodu alan adlarınızı belirterek .htaccess dosyanıza yapıştırınız.

```
RewriteEngine On

RewriteCond %{HTTP_HOST} ^(www\.)?eskialanadiniz\.com$ [NC]

RewriteRule ^(.*)$ http://www.yenialanadiniz.com/$1 [R=301,L]
```

2. Hata Mesajları

"301 Moved Permanently", "404 Not Found" vb. sunucu tarafı hata sayfalarının alındığında sayfa yönlendirilmesi yapılarak bu durum kullanıcıdan gizlenebilir bir hale getirilebilir.

```
ErrorDocument 404 gidilecekSayfa.html
```

404 not found yani hata aldığımızda veya o sayfaya ulaşamadığımızda site aşağıdaki linke yönlendirilir. Eğerki çalışmayacak olursa aşağıdaki linkte yer alan video izlenerek çözüme ulaşılabilir.

https://www.emrahyuksel.com.tr/php-7-mod_rewrite-aktif-etme-kesin-cozum/