Programming Applications
Spring 2017-2018

Ast. Prof. Dr. Pelin GÖRGEL
(paras@istanbul.edu.tr)

02.03.2018

Checkboxes

- <input type="checkbox"> defines a checkbox. Checkboxes let a user select ZERO or MORE options of a limited number of choices.

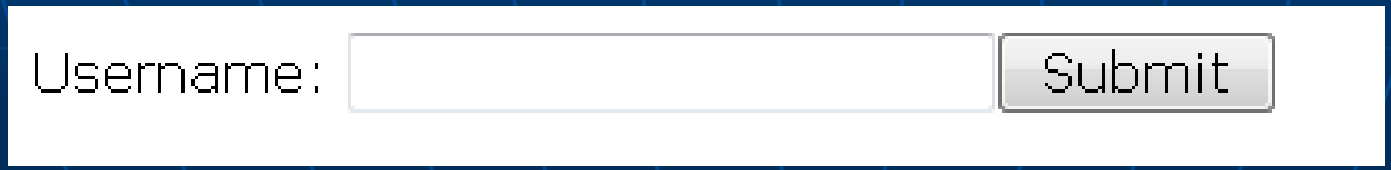
```
<form>
<input type="checkbox" name="vehicle"
value="Bike">I have a bike<br>
<input type="checkbox" name="vehicle"
value="Car">I have a car
</form>
```

Submit Button

- <input type="submit"> defines a submit button.
- A submit button is used to send form data to a server.
- The data is sent to the page specified in the form's action attribute.

Submit Button (Continued)

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



XHTML

- XHTML stands for E**X**tensible **H**yperText **M**arkup **L**anguage
- XHTML is almost identical to HTML 4.01
- XHTML is a stricter and cleaner version of HTML 4.01
- XHTML is supported by all major browsers.

Why XHTML?

- Many pages on internet contain missing HTML code.
- <html>
  <head>
  <title>This is bad HTML</title>
  <body>
  <h1>Bad HTML
  <p>This is a paragraph
  </body>

Why XHTML?

- Today's market consists of different browser technologies. Some browsers run on computers, and some browsers run on mobile phones or other small devices.
- Smaller devices often lack the resources or power to interpret a "bad" markup language.
- Therefore - by combining the strengths of HTML and XML, XHTML was developed. XHTML is HTML redesigned as XML.

The Most Important Differences from HTML

- XHTML DOCTYPE is **mandatory**
- The XML namespace attribute in <html> is **mandatory**
- <html>, <head>, <title>, and <body> is **mandatory**
- XHTML elements must always be **closed**
- XHTML elements must be in **lowercase**

The Most Important Differences from HTML

- XHTML elements must be **properly nested.**

  Some elements can be improperly nested within each other:

- This text is bold and italic *wrong*
- This text is bold and italic *right*

The Most Important Differences from HTML

- Attribute names must be in **lower case**
- Attribute values must be double **quoted**
- <table width=100%>*wrong*
- <table width="100%"> *right*

A Sample Web Page-1

- <!DOCTYPE html>
- <html>
- <body bgcolor= "pink">

- This text is bold
- This text is strong
- This text is emphasized
- This text is italic
- <small>This text is small</small>
- This is_{subscript} and ^{superscript}

- </body>
- </html>

A Sample Web Page-2

- <!DOCTYPE html>
- <html>
- <body>
- Create a link of an image:
-
- <img border="1" src="smiley.gif" alt="HTML tutorial" width="32" height="32">
- No border around the image, but still a link:
-
- <img border="0" src="smiley.gif" alt="HTML tutorial" width="32" height="32">
- </body></html>

A Sample Web Page-3

- <!DOCTYPE html>
- <html>
- <body>
- <h4>Cell that spans two columns:</h4>
- <table border="1">
- <tr>
- <th>Name</th>
- <th colspan="2">Telephone</th>
- </tr>
- <tr>
- <td>Bill Gates</td>
- <td>555 77 854</td>
- <td>555 77 855</td>
- </tr></table>

A Sample Web Page-3 (Continued)

```
<h4>Cell that spans two rows:</h4>
<table border="1">
<tr>
  <th>First Name:</th>
  <td>Bill Gates</td>
</tr>
<tr><th rowspan="2">Telephone:</th>
  <td>555 77 854</td>
</tr>
<tr>
  <td>555 77 855</td>
</tr>
</table></body></html>
```

HTML Layouts - Using Elements

- You can also code "divisions," anything from a word to a line of text to an entire page, to be justified a certain way.
- Start your division with the division alignment tag, including the justification you wish the text or images to take on (i.e. right, left, center):
- Example:
- End the division justification alignment with the tag:
-

HTML Layouts - Using Elements

- The div element is a block level element used for grouping HTML elements.
- The following example uses five div elements to create a multiple column layout, creating the same result as in the previous example:

A Simple Page With Tag

- <!DOCTYPE html>
  <html>
  <body>

-
  <h1 style="margin-bottom:0;">Main Title of Web Page</h1>
  <div id="menu" style="background-color:#FFD700;height:200px;width:100px;float:
- left;">
  Menu
  HTML
  CSS
  JavaScript

A Simple Page With (continued)

- <div id="content" style="background-color:#EEEEEE;height:200px;width:400px;float:
- left;">
  Content goes here

  Copyright © W3Schools.com

  </body>
  </html>

The HTML code above will produce that page:

HTML Styles - CSS

- CSS (Cascading Style Sheets) is used to style HTML elements:
- Inline - using the style **attribute** in HTML elements
- Internal - using the <style> **element** in the <head> section
- External - using an external CSS **file**

Inline CSS

```
<!DOCTYPE html>
<html>
<body style="background-color:yellow;">
<h1 style="font-family:verdana;">heading 1</h1>
<p style="font-family:arial;color:red;font-
size:20px;">1st paragraph.</p>
<h2 style="background-color:red;“>heading 2</h2>
<p style="background-color:green;"> 2nd
paragraph.</p>
<h3 style="text-align:center;“> heading 3</h1>
<p>3rd paragraph.</p>
<p style="color:blue;margin-left:20px;“>4th
paragraph.</p>
</body>
</html>
```

Internal CSS

- An internal style sheet can be used if one single document has a unique style. Internal styles are defined in the <head> section of an HTML page, by using the <style> tag, like this:

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

External CSS

- An external style sheet is ideal when the style is applied to many pages.
- With an external style sheet, you can change the look of an entire Web site by changing one file.
- Each page must link to the style sheet using the <link> tag. The <link> tag goes inside the <head> section:
- This has the attributes **rel**, **type**, and **href**. The rel attribute tells what you are linking (in this case a stylesheet), the type defines the MIME-Type for the browser, and the href is the path to the .css file.

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

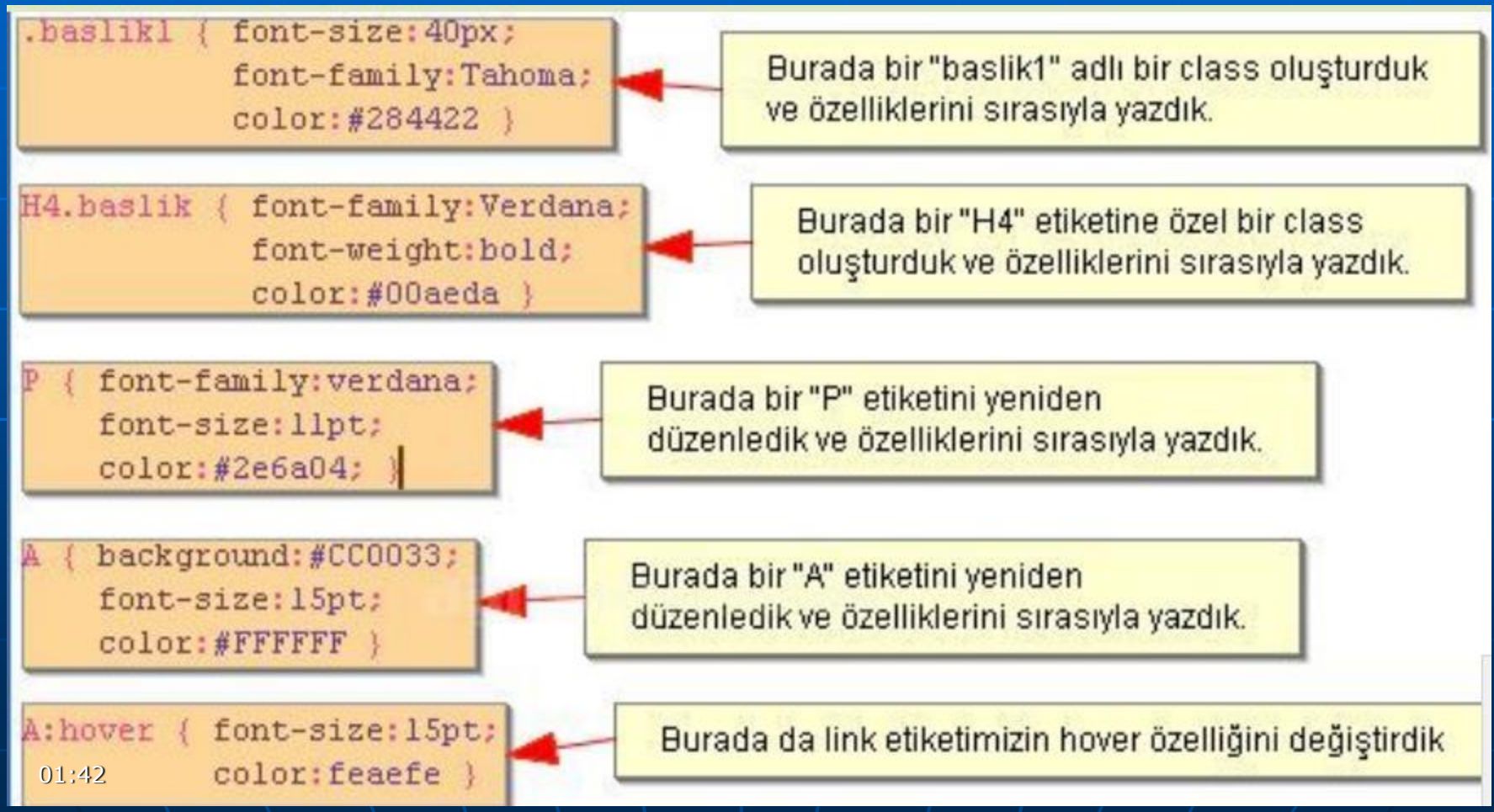
CSS Examples -1 (Internal CSS)

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Css'in HTML içerisinde kullanım şekilleri</title>
<style type="text/css">
.yazi {
    font-family: Verdana;
    font-size: 10px;
    letter-spacing: 1px;
    color: #666666;
}
</style>
</head>
<body>
<p class="yazi"> Paragraf yazıları, metin girdileri... </p>
</body>
</html>
```

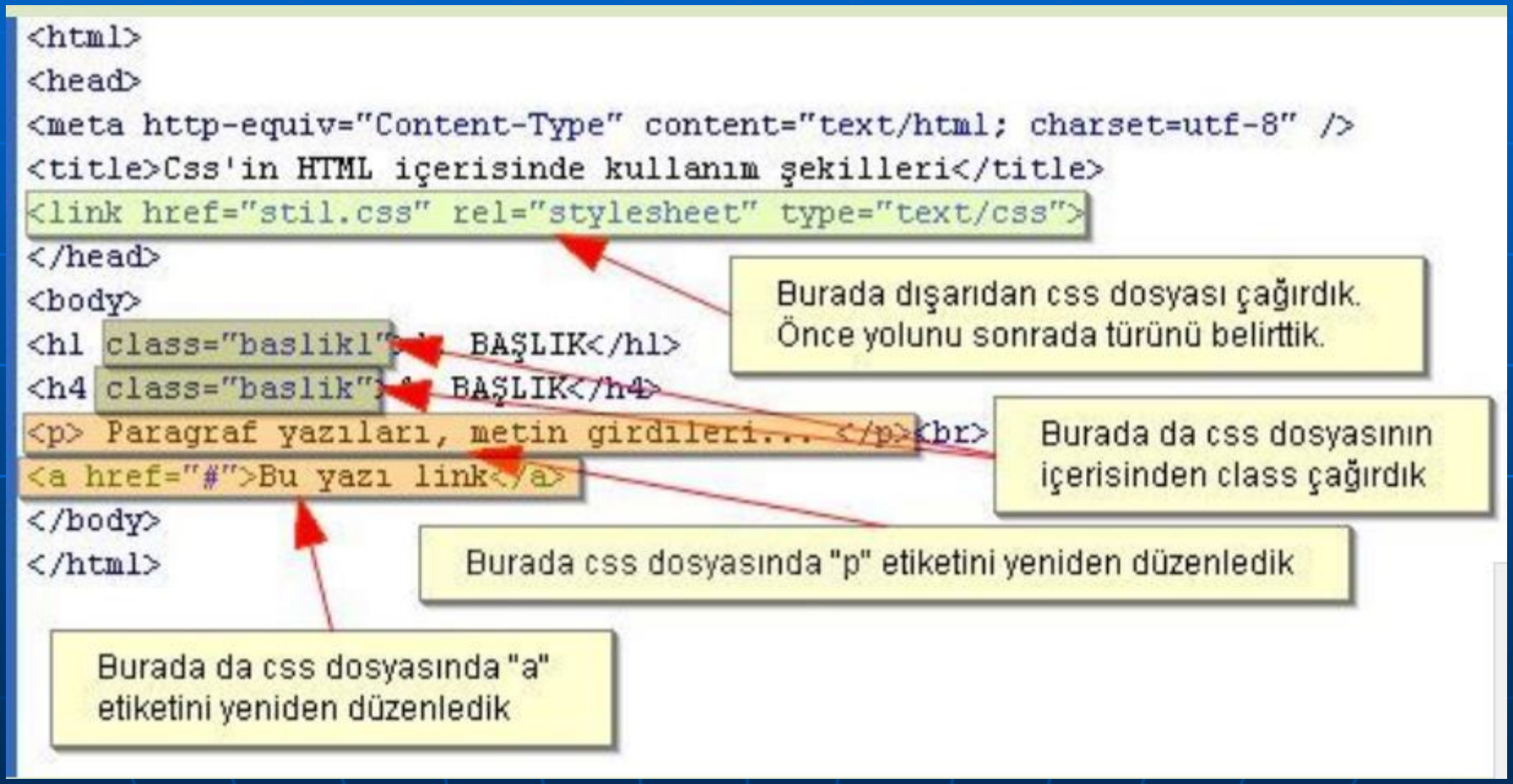
Burada yaptığımız stil özellikleri <head> </head> tagı içerisinde <style> </style> tagları arasında yaptığımız özellikleri tanımladık. ' .yazi ' adında bir isim vererek değerlerini belirttik.

Burada da yaptığımız stil özelliğini class adında çağırdık.

CSS Examples -2 (External CSS How to Define stil.css)



CSS Examples -2 (How to Use stil.css)




```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Css'in HTML içerisinde kullanım şekilleri</title>
<link href="stil.css" rel="stylesheet" type="text/css">
</head>
<body>
<h1 class="baslik1">  BAŞLIK</h1>
<h4 class="baslik">  BAŞLIK</h4>
<p> Paragraf yazıları, metin girdileri... </p><br>
<a href="#">Bu yazı link</a>
</body>
</html>
```

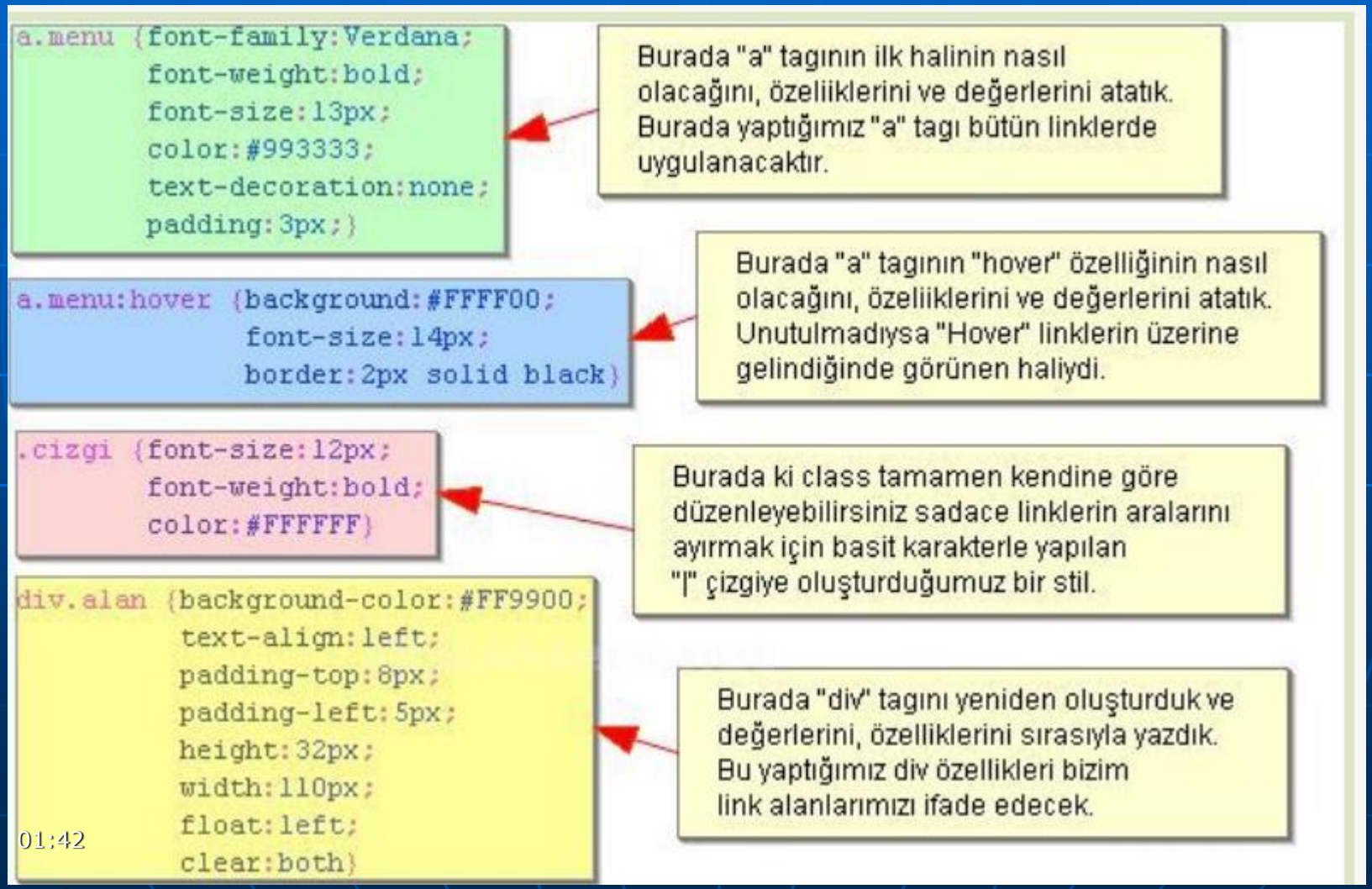
CSS Examples -3 (Internal CSS)

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Css'in HTML içerisinde kullanım şekilleri</title>
<style type="text/css">
h2.yesil { font-family:verdana;
           color:green; }
h2.kirmizi { font-family:tahoma;
             color:red; }
</style>
</head>
<body>
<h2 class="yesil" >CSS Kullanımına Örnek </h2>
<h2 class="kirmizi" >CSS Kullanımına Örnek </h2>
</body>
</html>
```

Burada "h2" etiketini iki farklı yönde kullandık class seçicisi ile "h2.yesil" ve "h2.kirmizi" adlı iki tane ayrı başlık oluşturduk.

class seçicisi sayesinde burada değişik bir başlık atatık.

CSS Examples -4 (External CSS How to Define stil.css)



CSS Examples -4 (How to Use stil.css)

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Css'in HTML içerisinde kullanım şekilleri</title>
<link href="stil.css" rel="stylesheet" type="text/css">
</head>
<body>
<div align="center" class="cizgi">
    <div class="alan"><a class="menu" href="#"> Ana Sayfa </a></div>
    <div class="alan"><a class="menu" href="#"> Hakkımızda </a></div>
    <div class="alan"><a class="menu" href="#"> Resimler </a></div>
    <div class="alan"><a class="menu" href="#"> Galeri </a></div>
    <div class="alan"><a class="menu" href="#"> İletişim </a></div>
</div>
</body>
</html>
```

Burada css dosyasından yaptığımız "a" özeliklerini çağırıyoruz

Burada css dosyasından yaptığımız "div" özeliklerini çağırıyoruz

01:42

CSS Examples -4
(The Display of the Page)

CSS Examples -5 (External CSS How to Define stil1.css)

```
a.menu { font-family:Verdana;
         font-weight:bold;
         font-size:13px;
         color:#993333;
         text-decoration:none;
         padding:3px;
         border:solid 3px }
```

Burada "menü" adında link özellikleri ayarladık. Yazı tipi - verdana, kalın, 13px, bir renk, menünün altındaki çizginin çıkmaması için "text-deceration" ifadesini "none" yaptık, 3px boşluk ve 3px kenarlık ekledik.

```
a.menu:hover { background:#000000;
               color:#fff000;
               font-weight:bold;
               padding:3px }
```

Burada linkin "hover" özelliğinin değerlerini atatık Bilindiği gibi "hover" üzerine gelince gözüken linkti. Bir arkaplan, yazı renki, yazı kalınlığı ve 3px lik boşluk kullandık.

```
a.menu:active { color:#00FFFF;
                text-decoration:none }
```

Burada menünün "active" özelliğin linke basıldığında karşımıza çıkan görüntüyü ifade etmektedir. Bir renk, ve altında çizgi çıkmamasını sağladık.

```
a.menu:visited { background-color:#EFEFEF;
                 color:#666666;
                 text-decoration:none }
```

Burada menünün "visited" yani linke basıldıktan sonra karşımıza çıkan görüntüyü ifade etmektedir. Bir arkaplan, bir renk, ve altında çizgi çıkmamasını sağladık.

01:42

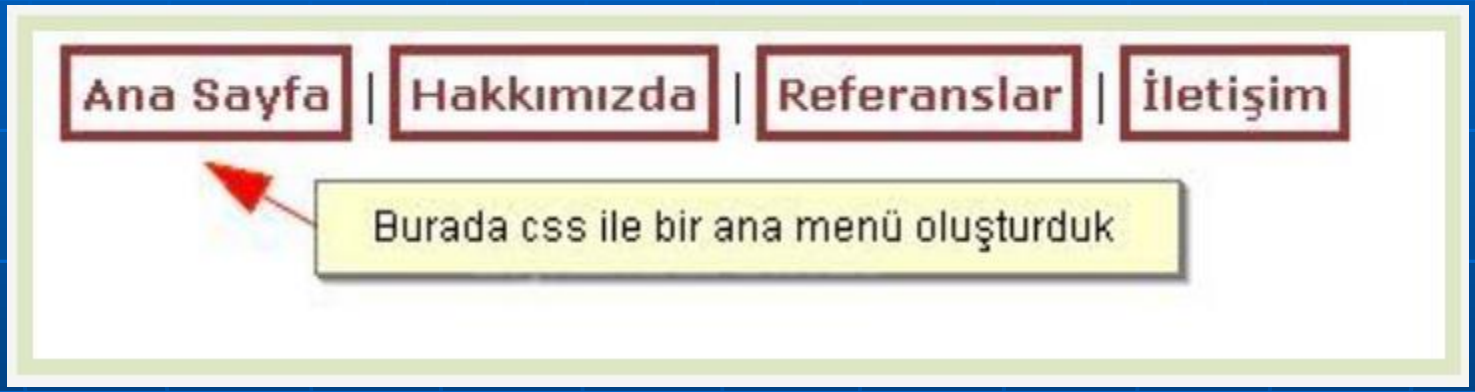
CSS Examples -5
(How to Use stil1.css)

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Css'in HTML içerisinde kullanım şekilleri</title>
<link href="still.css" rel="stylesheet" type="text/css">
</head>
<body>
<a href="#" class="menu">Ana Sayfa</a> |
<a href="#" class="menu">Hakkımızda</a> |
<a href="#" class="menu">Referanslar</a> |
<a href="#" class="menu">İletişim</a>
</body>
</html>
```

Dikkat edildiyse burada stil1.css adlı dosyayı html sayfamıza çağırıyoruz

Burada css dosyasından menu adlı class 'ımızı çağırdık.

CSS Examples -5
(The Display of the Page)



01:42

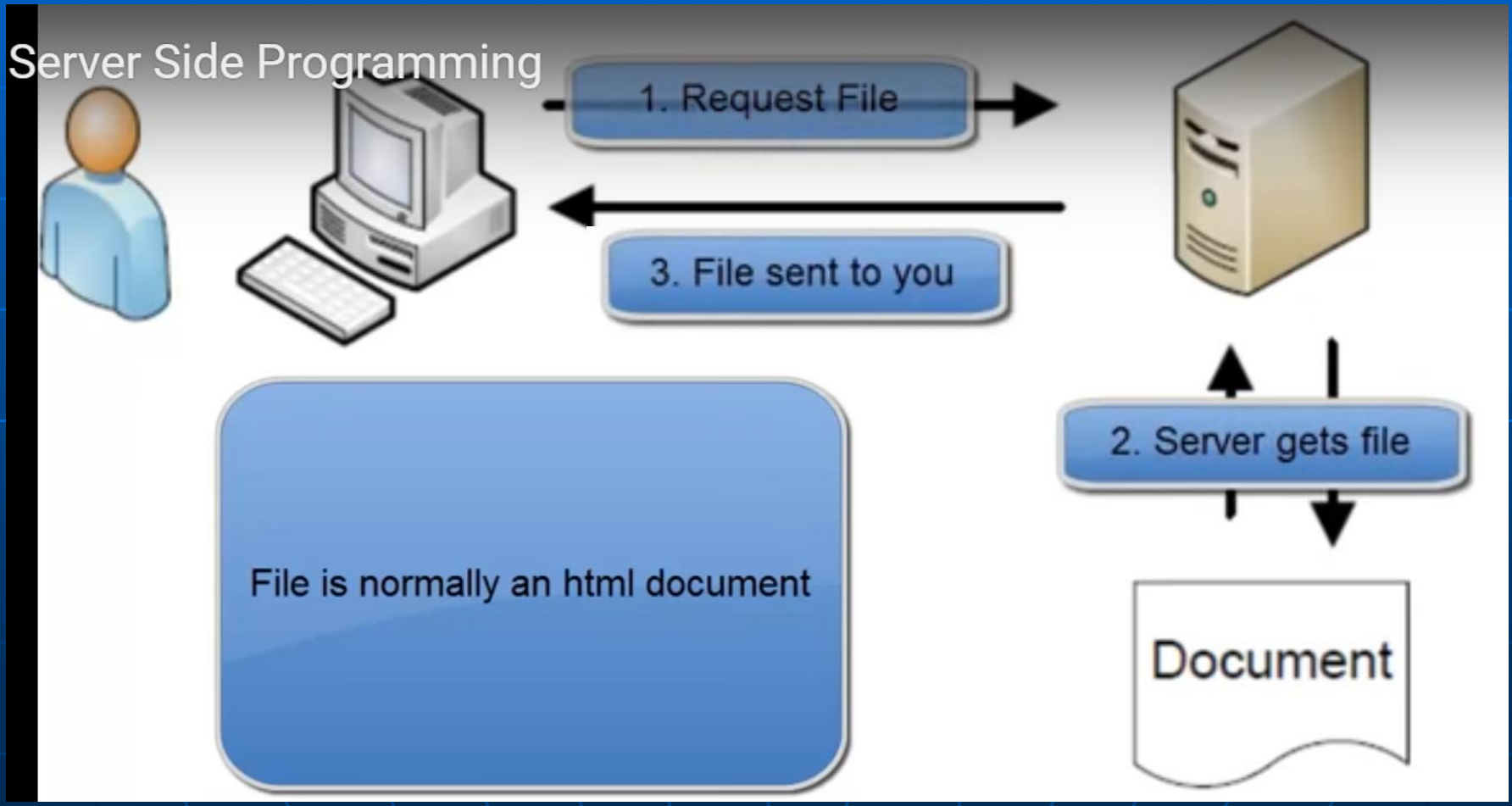
Web Applications
Client-Server Model

- Program that accesses the application is called the **client** (usually a browser)
- Program that provides the application is called the **server**

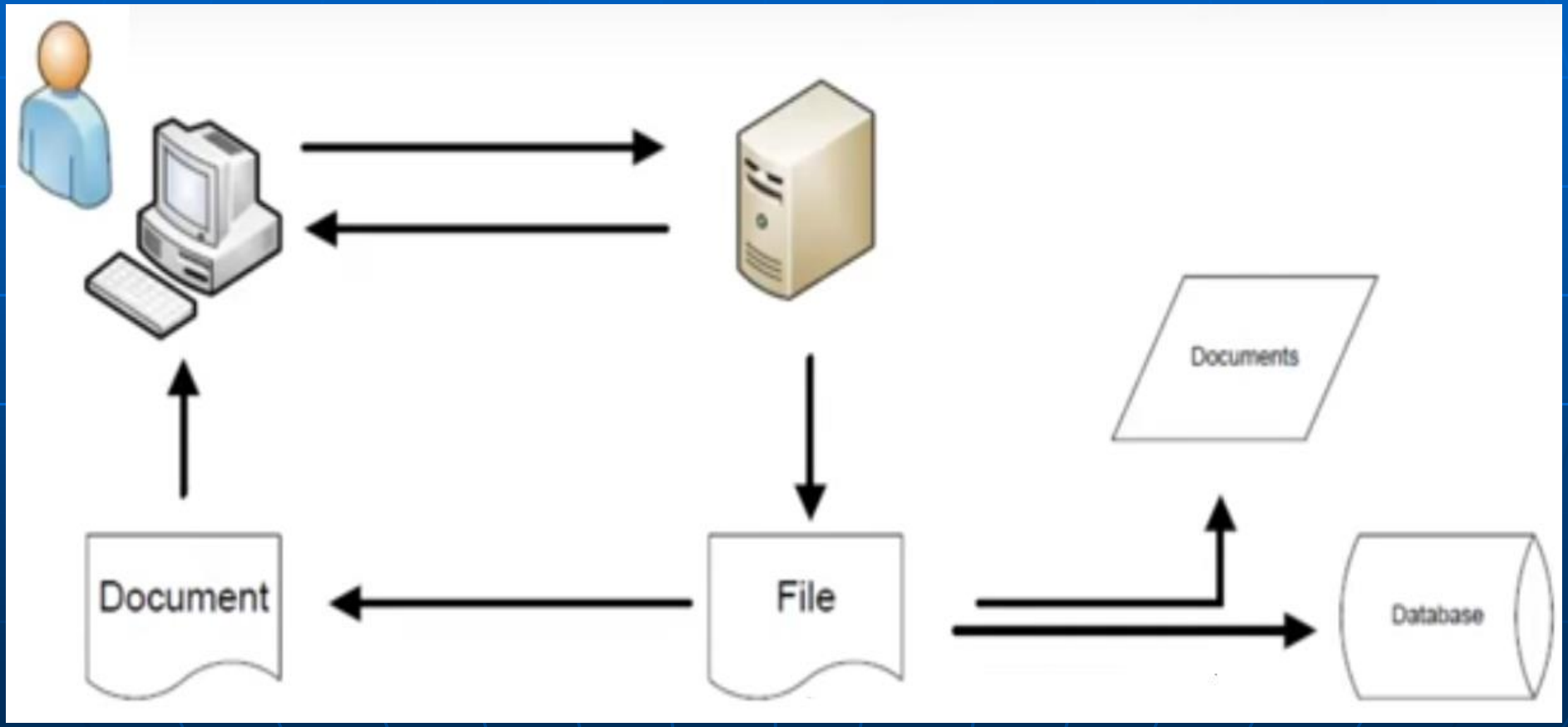
Two Meanings of the Server

- Software applications that can be accessed remotely by clients
- Computer hardware or virtual machine hosting such applications

Basics of Server Side Programming



Server Side Web Page Request



Static and Dynamic Content

- Static Content
  - Server transmits an existing file (text, image, etc.)
  - Client renders and displays the file
- Dynamic Content
  - Content is generated on-the-fly by programs running on the server and/or client

Server-Client Relation in Web Applications

- The machine where the Web site actually resides is called a Web server.
- When you send a request for a Web page by entering a Web site address, this request is sent to a Web Server. The Web server then sends the Web page to your browser.
- A Web server can manipulate the code within a Web page before sending it to your browser.

01:42

Server-Client Relation in Web Applications (continued)

- When the request is sent to a server to get a Web page, the server can actually execute a program instead.
- This program might be either Active Server Pages (ASP), PHP, C/C++, and ISAPI, Java Server Pages (JSP), Java Server Faces (JSF) and so on.

Server-Client Relation in Web Applications (Continued)

- Web server is separate from the Web browsers (client) that will use a Web site's pages.
- The Web server can be located anywhere in the world or even off the world! But the browser is generally on a machine you are using.

Web Applications

- Server Side Programming
- Client Side Programming

Server Side Programming

- The programs running on the Web Server are server side programs because they are on the side of the internet that the Web server is on.
- If code is executed on the Web server, it is considered server side code.
- Server side scripts start processing data generally when end user or client triggers some action.
- Server side looks for the request object sent by end user and processes it and presents the relevant data model.

Difference Between Client Side & Server Side Programming

- When programming a web application, it's important to know the difference between client-side programming and server-side programming.
- Client-side programming is run on the clients machine, creating some advantages and disadvantages. Even with client-side advantages, server-side programming is more secure and is preferred by most programmers.
- Server-side programming also has more options for languages than client-side.

Server Side Programming

- There are a lot of options for using server side scripts including some of the following frequently used applications:
- - Encrypting and Password Protection
- - Searching a Database on the Server
- - Processing an Online Form

Client Side Programming

- The browser being used to access the Web site is on the same side of the Web as you, the client side.
- If code is executed on your browser, it is considered client-side.
- There is code that you can use on the server and there is code that you can use on the client.
- HTML, Javascript, Flash files, ActiveX controls, Java applets, and a number of other technologies can be executed on the client side.

Client Side Programming

- Client-side programs run on the user's computer.
- An example of client-side programming is Javascript.
- Javascript can be used to run checks on form values and send alerts to the user's browser. The problem with client-side scripts is the limit of control and problems with operating systems and web browsers. Since programming a website involves users with several options of computer software, it's difficult for programmers to account for any bugs in the code or compatibility issues with browsers.

Client Side Programming (Continued)

- Because of their ability to react to your user's actions, these types of scripts can be used for many fun and useful applications such as:
- - Online Games
- - Form Checkers (those little help boxes that check the information that a user inputs into the fields of a form to make sure that they are accurate before the form is submitted)

Client-side Advantages

- ***Server Resources*** –

  Each client uses its own resources to execute the code on the webpage, hence saving resources for the provider.

  This is inherently the disadvantage with server-side scripting, where the server must use valuable resources to parse each page it sends out, possibly slowing down the web site.

Client-side Advantages (Continued)

- Scripts in the category of client side scripts are executed by, and run directly in, the user's browser itself.
- They are easily embedded into your web site using easy to use HTML code and because client side scripts are being ran in the browser they can easily and quickly respond to your user's actions which can make for a more interactive experience for your user when they visit your web site.
- By including code within a web page, a number of features can be added to a Web page without the need to send information to the Web Server which takes time.
- Tasks on the client side include data validation, special formatting and more.

01:42

Server Side Programming

- Server-side scripts run on the server. This reduces the amount of bugs or compatibility problems since the code runs on one server using one language and hosting software.
- Server-side programming can also be encrypted when users send form variables, protecting users against any hack attempts.
- Some examples of server-side programming languages are C#, VB.NET, and PHP.

Server-side Advantages

- ***Code Protection*** - Because the server side code is executed before the HTML is sent to the browser, your code is hidden.
- ***Browser Independent*** - Server-side code is browser independent. Because the HTML code returned by your browser is simple HTML, you do not need to worry about the version of browser, javascript etc. that the client is using.

Server-side Advantages (Continued)

- ***Application Updates*** - By operating on the server-side updating to new versions, fixing bugs, implementing code patches become a simple process of updating the server machine.
- Consider in opposition to this, a windows application running on the client - in order to update or make fixes, the client must download the latest version each time.
- Uses less of your visitors resources.

Server-side Advantages (Continued)

- Since server side scripts run on the server they generally have more flexibility than client side scripts since you have a greater range of files that they can access and more variety of scripting languages to choose from.
- Some examples of scripting languages that you can use with server side scripts include Java, PHP, Perl, Python, and VBScript among others.

Server-side Advantages (Continued)

- **Security**

  Server-side scripts are more secure than client-side.

  For instance, when a user accesses a bank account online, the server side-script communicates with the client using encryption.

  A client-side script is plain text and run on the client's browser. Any hacker can view the code and steal private information from the user's computer.