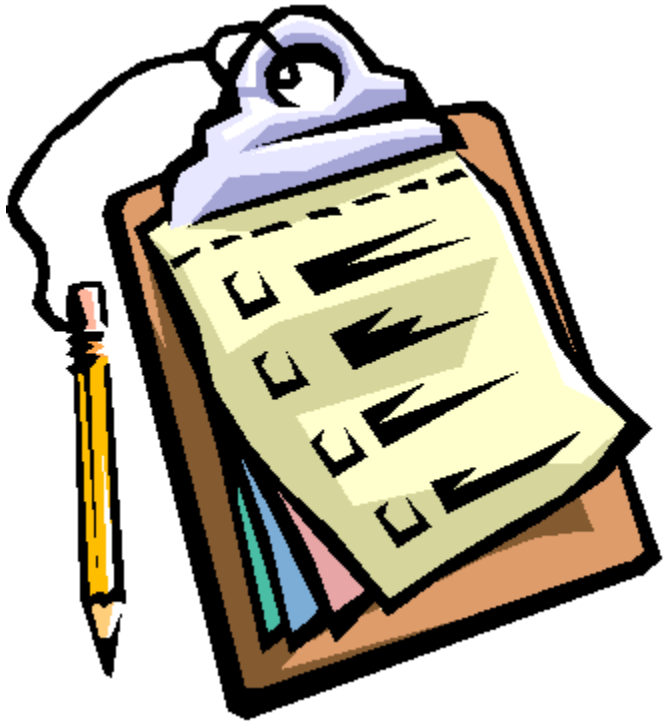
INTERNET ENGINEERING

Client-side Web Programming Basics

Sadegh Aliakbary

Agenda

- HTML Language
- CSS
- JavaScript



HTML

Building Websites

- HTML (HyperText Markup Language)
- The dominate language of the internet
- Describes a webpage's content
- Controlled by the World Wide Web Consortium (W3C)
- In practice, affected by Browser compatibility

Terms

- HTML Document
  - A **plain text** file, that can be created using:
    - a text editor (Notepad in Windows, or vim on Linux)
    - a Web page editor (Dreamweaver, …)
- DHTML (Dynamic HTML)
  - a combination of **HTML**, **CSS**, and **JavaScript**
- XHTML (E**x**tensible HTML)
  - HTML defined as an XML application

Tag Syntax

- HTML Markup tags: Special codes that tell the Web browser how to display the HTML document
- HTML tags include:

1. A start tag
2. An optional attribute name/value pair
3. Content affected by the tag (optional)
4. A closing tag

```
<tag-name attribute-name="attribute-value">
    ... Content ...
</tag-name>
```

Markup Tag

- Tells the Web browser the format of the text
- Surrounded by < >
- Examples:
  - paragraph tag: <p>

Markup Tag

- In pairs: start tag and end tag (closing tag)
- Example:
  - start tag:
  - end tag:
- Placement of start and end tags
  - Example:

This is a paragraph.

element content

Document Syntax

- HTML documents are made up of a hierarchical "tree" of tags structuring content.

- Correct hierarchy:

```
<p> The following text is <b> bold </b> </p>
```

- Incorrect hierarchy:

```
<p> The following text is <b> bold </p> </b>
```

Tags That Do Not Have Element Content

Examples:

- line break:
</br>

  can be written as:

- image tag: <img></img>

  can be written as: <img />

Attributes of a Tag

- To specify properties of the element
- In name-value pairs like this: name = "value"
- Example:
  - id attribute:

    This is a paragraph.
- Placed inside the start tag

Hello World in HTML

Hello World Example

```
<!DOCTYPE html>
<html>
    <head>
        <title> Hello World Page </title>
    </head>
    <body>
        <p> Hello SBU Students! </p>
        <p> This is our first HTML page </p>
    </body>
</html>
```

Hello World Example

```
<!DOCTYPE html>
<html>
      <head>
            <title> Hello World Page </title>
      </head>
      <body>
            <p> Hello SBU Students! </p>
            <p> This is our first HTML page </p>
      </body>
</html>
```

- The `html` tags enclose all of the page's content.

Hello World Example

```
<!DOCTYPE html>
<html>
      <head>
            <title> Hello World Page </title>
      </head>
      <body>
            <p> Hello SBU Students! </p>
            <p> This is our first HTML page </p>
      </body>
</html>
```

- The `head` tags contain extra information about the webpage.

Hello World Example

```
<!DOCTYPE html>
<html>
      <head>
            <title> Hello World Page </title>
      </head>
      <body>
            <p> Hello SBU Students! </p>
            <p> This is our first HTML page </p>
      </body>
</html>
```

- The `title` tags tell the browser the page's name; often the title is used by the browser to label the window or tab.

Hello World Example

```
<!DOCTYPE html>
<html>
    <head>
        <title> Hello World Page </title>
    </head>
    <body>
        <p> Hello SBU Students! </p>
        <p> This is our first HTML page </p>
    </body>
</html>
```

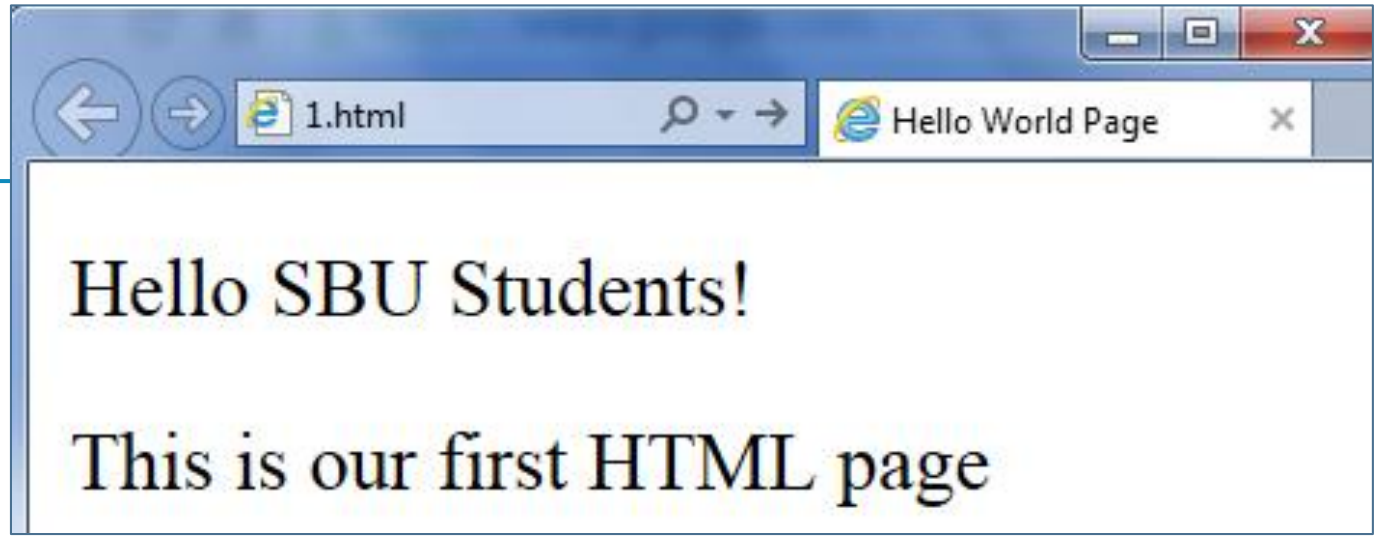
- The body tags contain the content displayed on the webpage.

Hello World Example

```
<!DOCTYPE html>
<html>
      <head>
            <title> Hello World Page </title>
      </head>
      <body>
            <p> Hello SBU Students! </p>
            <p> This is our first HTML page </p>
      </body>
</html>
```

- The p tags surround paragraphs of text.

Hello World Example




```
<!DOCTYPE html>
<html>
    <head>
        <title> Hello World Page </title>
    </head>
    <body>
        <p> Hello SBU Students! </p>
        <p> This is our first HTML page </p>
    </body>
</html>
```

HTML Tags

Common Tags

- Headers

  `<h1>, <h2>, <h3>, ...`

- Italics

  `<i>`

- Boldface

  `<b>`

- Line breaks (singleton tag)

  `<br>`

- Comments (singleton tag)

  `<!-- ... -->`

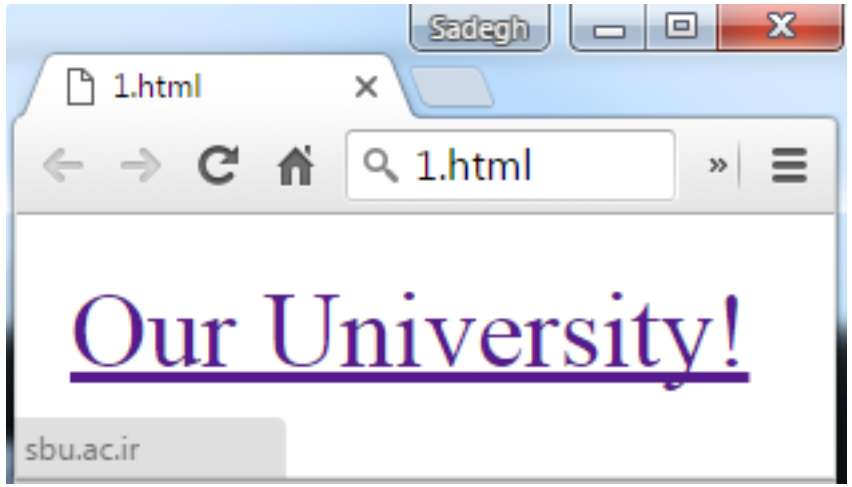
Note

- HTML has no variables or commands
  - HTML is not a programming language
  - It is a markup language, like LaTex, XML, …
- HTML is merely a way of formatting a document
- Intended to be platform- and device-independent
- HTML tags are not case-sensitive

Links

The link tag:

```
<a href="http://sbu.ac.ir">
    Our University!
</a>
```



The `href` attribute:

- Use full page address with "`http://`" to access other website's pages
- Use only the page's filename to access other html pages saved in the same directory

Images

The image tag:

```
<img src="sbu.jpg">
```

The `src` attribute:

- Use full page address with "`http://`" to access other website's pages
- Use only the page's filename to access other html pages saved in the same directory

<ALT> Graphic Tag

- <ALT> option.
- Example: <img src="pencil.gif" ALT="pencil graphic">
- Benefits?
  - A description
  - Unable-to-Load situations
  - Search Engines

pencil graphic

Barack Obama - Timeline | Amit Agarwal | ctrlq.org

https://www.facebook.com/photo.php?fbid=10152922145268183&set=t.500808182&type=3&theater

Elements Network Sources Timeline

How Facebook sees this photograph?

```
▼<div class="_2-sx" style="width: 975px; height: 443px;">
    <img class="spotlight" alt="Image may contain: 26 people, people
    smiling, people sitting, people standing and indoor" aria-busy="false"
    src="https://scontent-mrs.xx.fbcdn.net/v/t31.0-8/
    11026265_10152922145268183_7094814000304971490_o.jpg?
    oh=c909302fd8c5bfd30fd76df08d4051f1&oe=58E25645"> == $0
  </div>
```

#facebook body #photos_snowlift div div div div div div.stage div._2-sx img.spotlight

Styles »

:hov .cls +

element.style { }

5YrNpy2AQJa.(

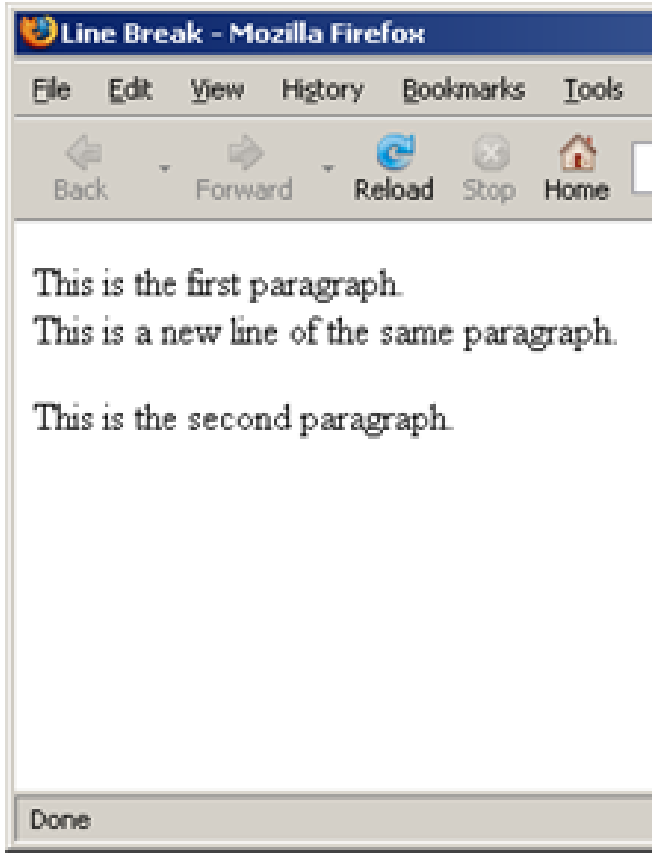
Exercise

- Create your Homepage web page
  - Use links and images
- “View source” for some web pages
- Use software tools for creating and editing web pages

Line Break:

Example:

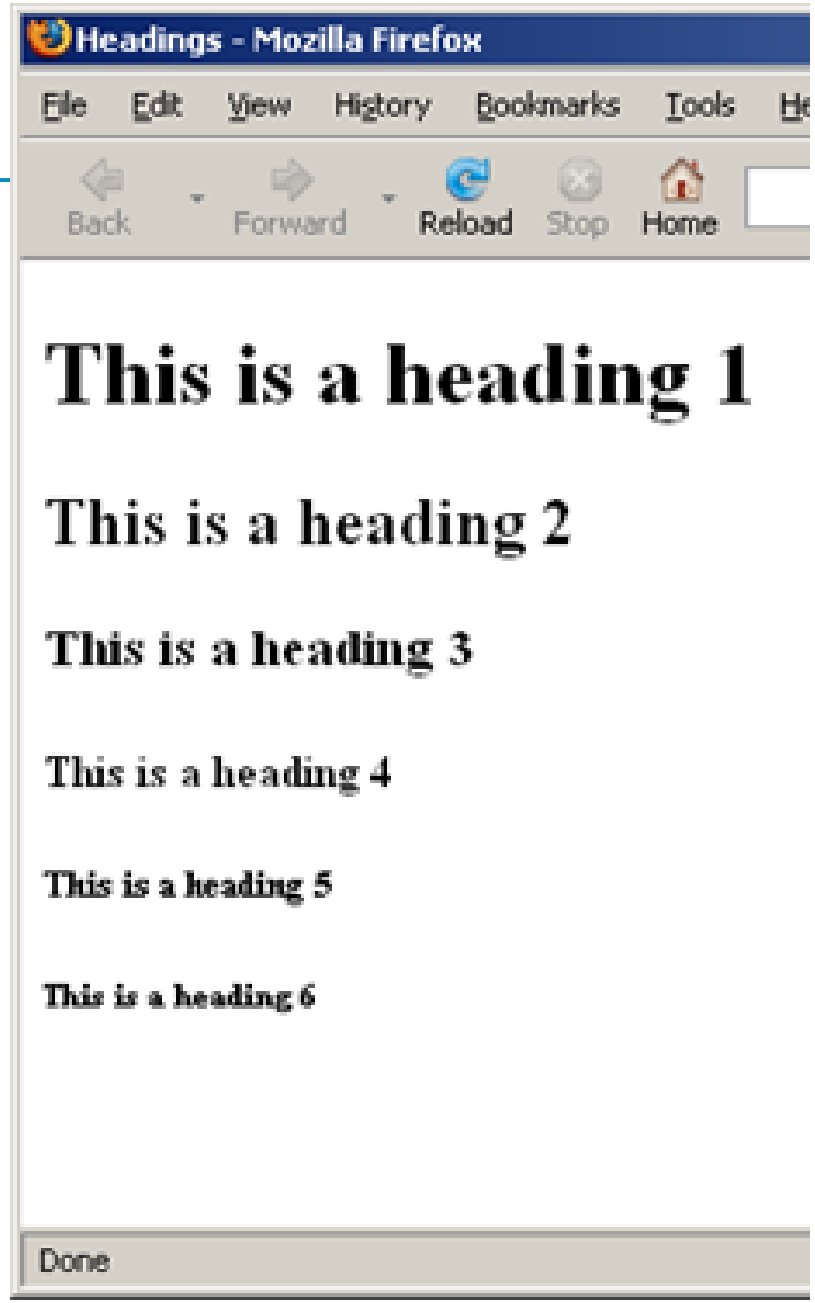
```
<p>This is the first paragraph.<br />This is a new line of the same
same paragraph.</p>
<p>This is the second paragraph.</p>
```



Headings: <h1> - <h6>

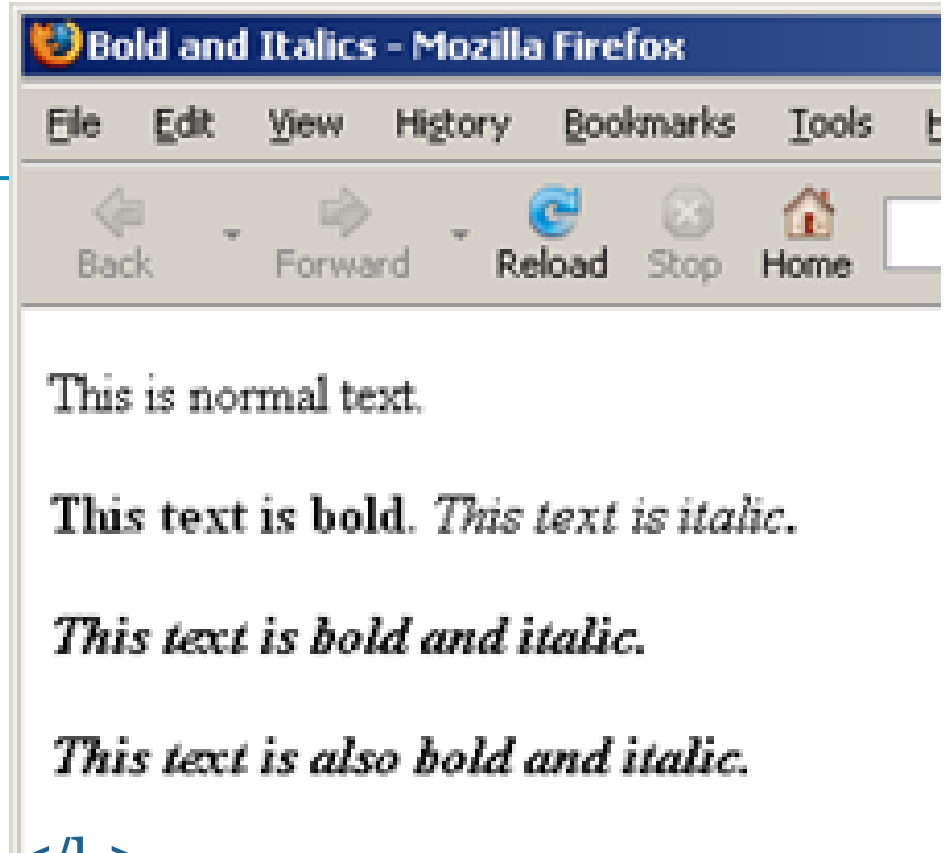
Example:

```
<h1>This is a heading 1</h1>
<h2>This is a heading 2</h2>
<h3>This is a heading 3</h3>
<h4>This is a heading 4</h4>
<h5>This is a heading 5</h5>
<h6>This is a heading 6</h6>
```



Bold and Italics

```
<p>This is normal text.</p>
<p>
<b>This text is bold. </b>
<i>This text is italic.</i>
</p>
<p>
<b><i>This text is bold and italic.</i></b>
</p>
<p>
<i><b>This text is also bold and italic.</b></i>
</p>
```

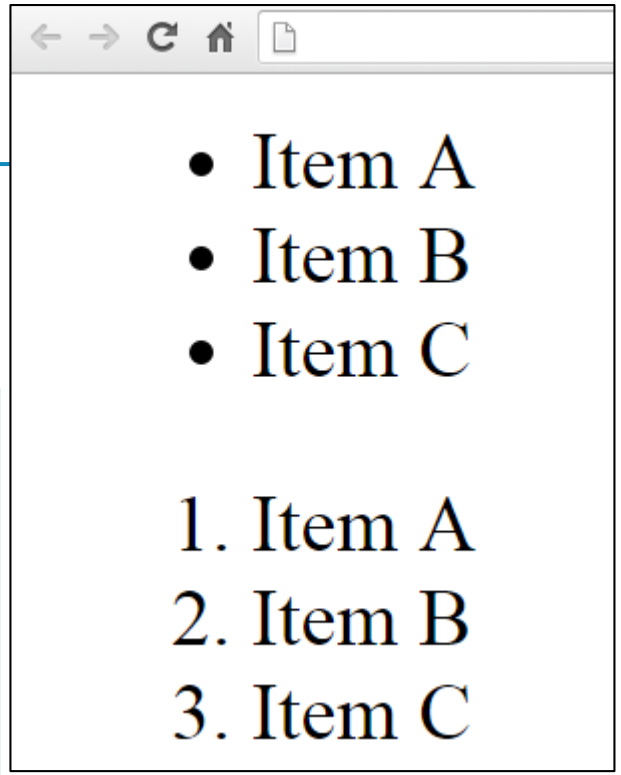


List

- Ordered list:
- Unordered list:
- List item:

Example:

```
<ul>
  <li>Item A</li>
  <li>Item B</li>
  <li>Item C</li>
</ul>
<ol>
  <li>Item A</li>
  <li>Item B</li>
  <li>Item C</li>
</ol>
```



Table

- Table: <table></table>
- Table row: <tr></tr>
- Table data: <td></td>
- Table heading: <th></th>

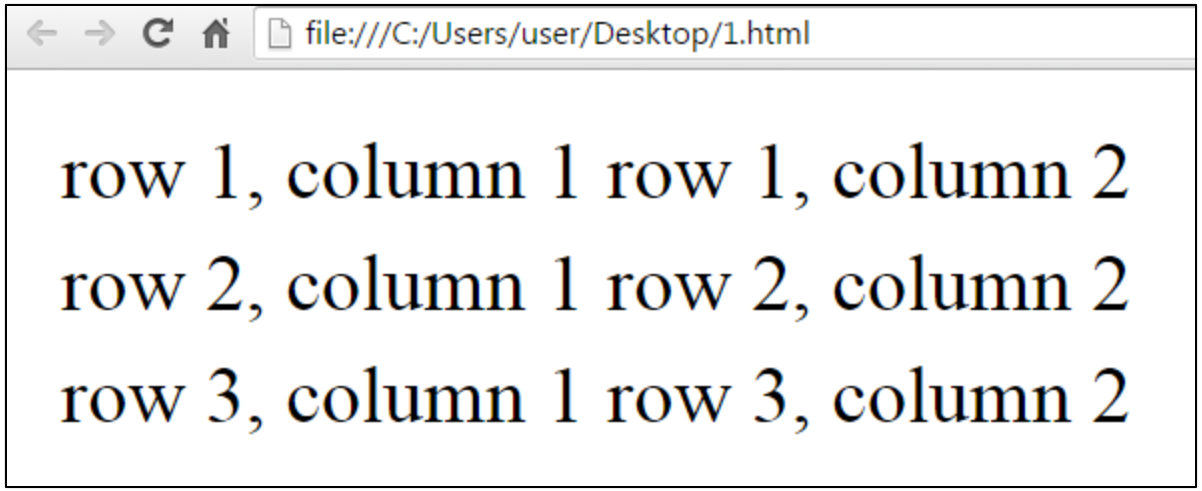
Table example

```
<table border=1>
<tr>
<th>Name</th><th>ID</th>
</tr>
<tr>
<td>Eric</td><td>123</td>
</tr>
<tr>
<td>RPI</td><td>456</td>
</tr>
</table>
```

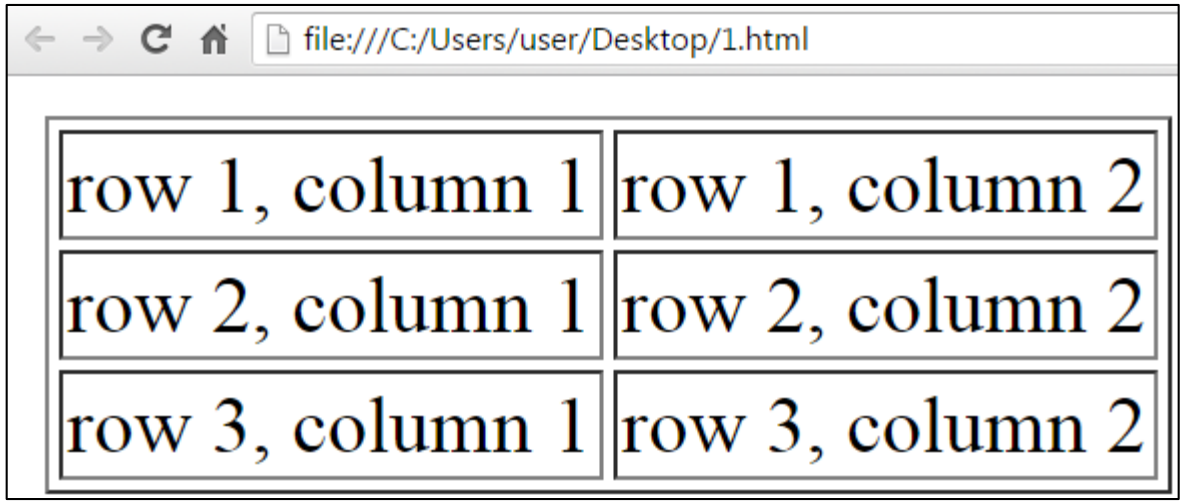
Name	ID
Eric	123
RPI	456

Example: A table <u>without</u> a border

```
<table>
  <tr>
    <td> row 1, column 1</td>
    <td> row 1, column 2</td>
  </tr>
  <tr>
    <td> row 2, column 1</td>
    <td> row 2, column 2</td>
  </tr>
  <tr>
    <td> row 3, column 1</td>
    <td> row 3, column 2</td>
  </tr>
</table>
```

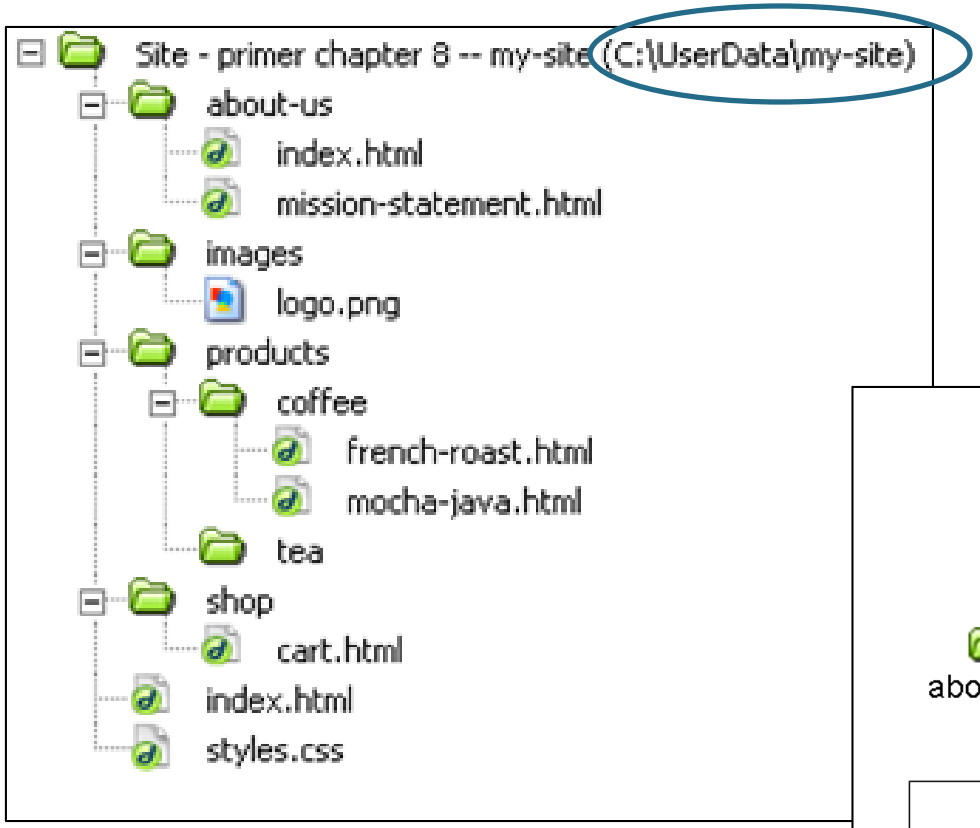


Example: A table with a border



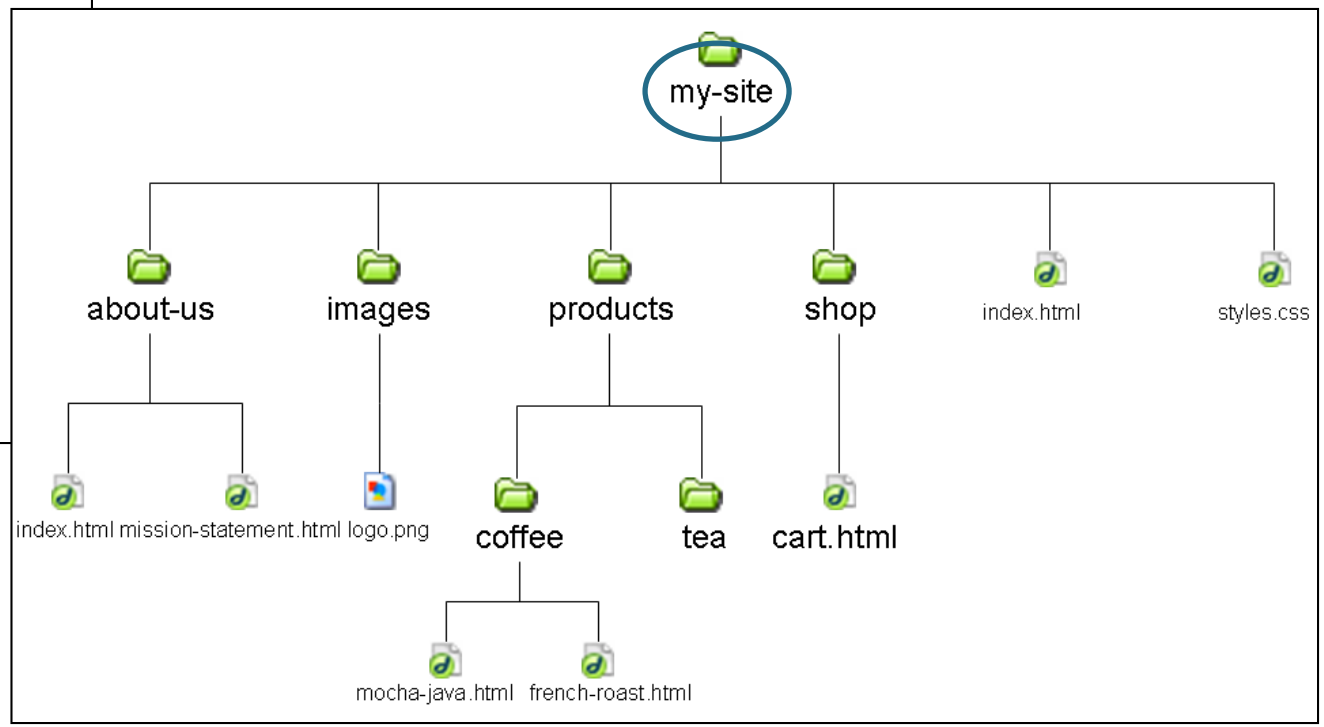

```
<table border="1">
  <tr>
    <td> row 1, column 1</td>
    <td> row 1, column 2</td>
  </tr>
  <tr>
    <td> row 2, column 1</td>
    <td> row 2, column 2</td>
  </tr>
  <tr>
    <td> row 3, column 1</td>
    <td> row 3, column 2</td>
  </tr>
</table>
```

Example Folder Structure of a Site

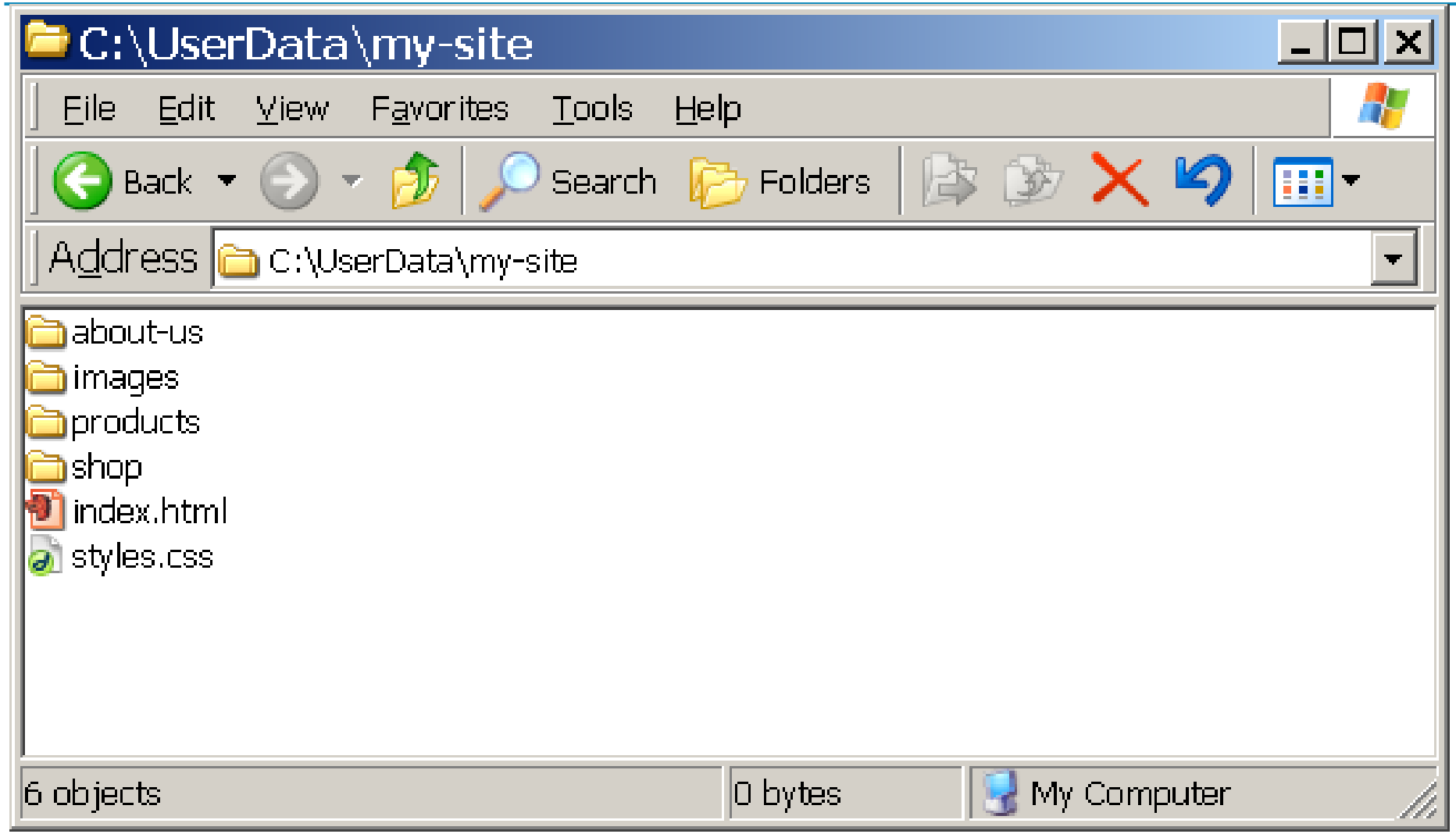


Root folder

Root folder



Navigating Folders



Character Entities

- Reserved characters in HTML must be replaced with character entities: **&entity_name;**
- **`<`** ➔ ‘less than’ ➔ <
- **`>`** ➔ ‘greater than’ ➔ >
- **`&`** ➔ ‘ampersand’ ➔ &
- **`"`** ➔ ‘quotes’ ➔ “
- **` `** ➔ ‘non-breaking white space’
  - A non-breaking space is a space that will not break into a new line.

Example

```
<p>Word1          Word2</p>
<p>Word1 Word2</p>
<p>Word1   
   Word2</p>
<br/>
<br/>
<p>A&lt;B &amp; C&gt;D
and &quot;X&quot; is a char</p>
```

Word1 Word2

Word1 Word2

Word1 Word2

A<B & C>D and "X" is a char

HTML Forms

Form Tag <form> … </form>

- Used to send user-specified information to server
  - The server then sends back its response, a new HTML document
- The form tag itself needs at least 2 attributes:

1- The "**action**" attribute

- The program on the server that the form's contents are sent to
- It processes the information and returns the response document

2- The "**method**" attribute.

- Specifies the **HTTP method** of the HTTP request
- Generally use method="post" method

Example

```
<form action="/books/save.jsp" method="post"> ...</form>
```

- The form sends name=value pairs to the server
- "name" and "value" are both specified within each form element.

Form Inputs

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

First name:

Last name:

```
<form>
  <input type="radio" name="gender" value="male" checked> Male<br>
  <input type="radio" name="gender" value="female"> Female<br>
</form>
```

Male

Female

Submit Form Elements

- "**Submit**" button: sends the form to the server.

  <input type="submit" value="button caption">

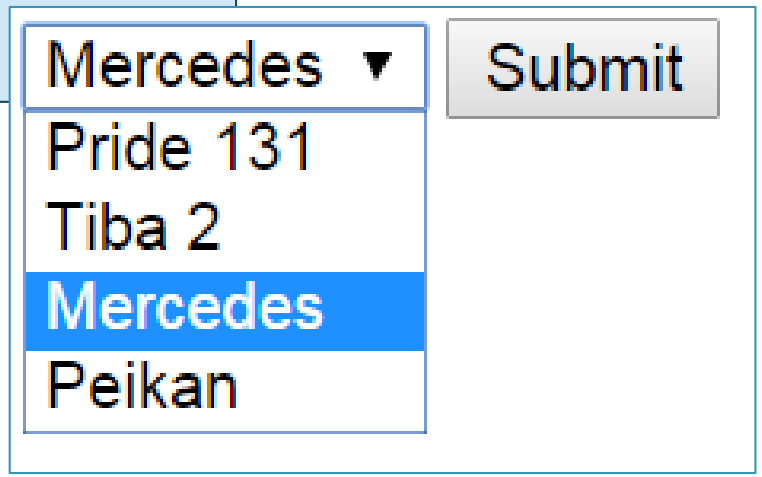
- Is "**Submit**" the only way to send data to the server?
  - No
  - AJAX

```
<form action="action_page.php">
  First name:<br>
  <input type="text" name="firstname" value="Mickey">
  <br>Last name:<br>
  <input type="text" name="lastname" value="Mouse"><br><br>
  <input type="submit" value="Submit">
</form>
```

First name:
Mickey
Last name:
Mouse
Submit

Select Box (Dropdown List)

```
<select name="car">
  <option value="Pride">Pride 131</option>
  <option value="Tiba">Tiba 2</option>
  <option value="Mercedes">Mercedes</option>
  <option value="Peikan">Peikan</option>
</select>
```



Other Form Elements

- Button
- Email
- Hidden
- Password
- Reset
- Checkbox
- File
- …

Quiz: What does this HTML do?

```
<html>

<body>

<form action="http://www.bing.com/search" method="GET">
<input name="q" />
<input type="submit"  value="Search"/>
</form>

</body>

</html>
```

But Generally we use method="post" ,
And the action is a relative path (no http)

Example

```
<form action="login.php" method="post">
  Username: <input type="text" name="username">
  <br>Password:<input type="password" name="password">
  <br><input type="submit" value="Login">
</form>
```

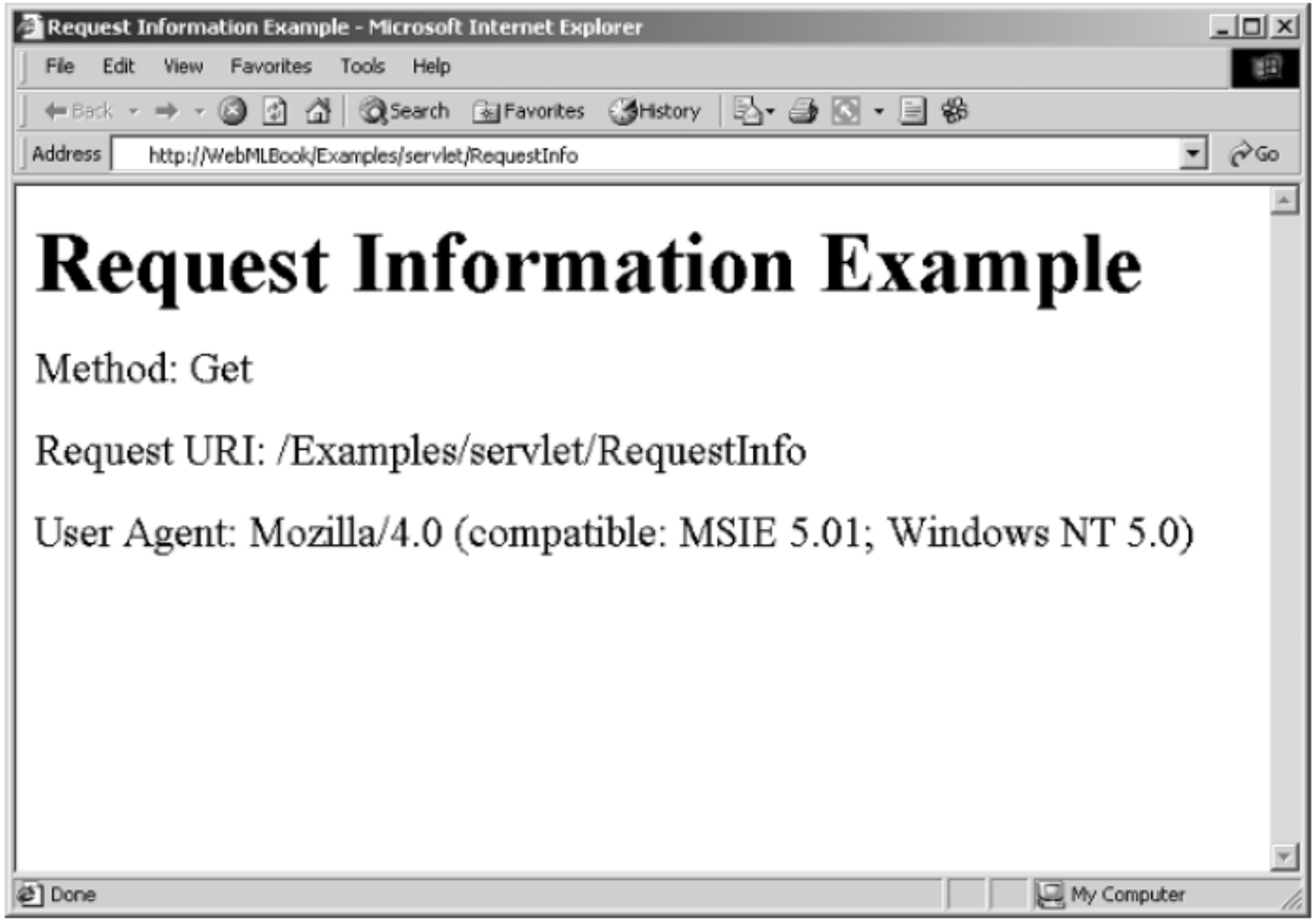
Username: ali

Password: ••••••••••

Login

Server-side Scripts

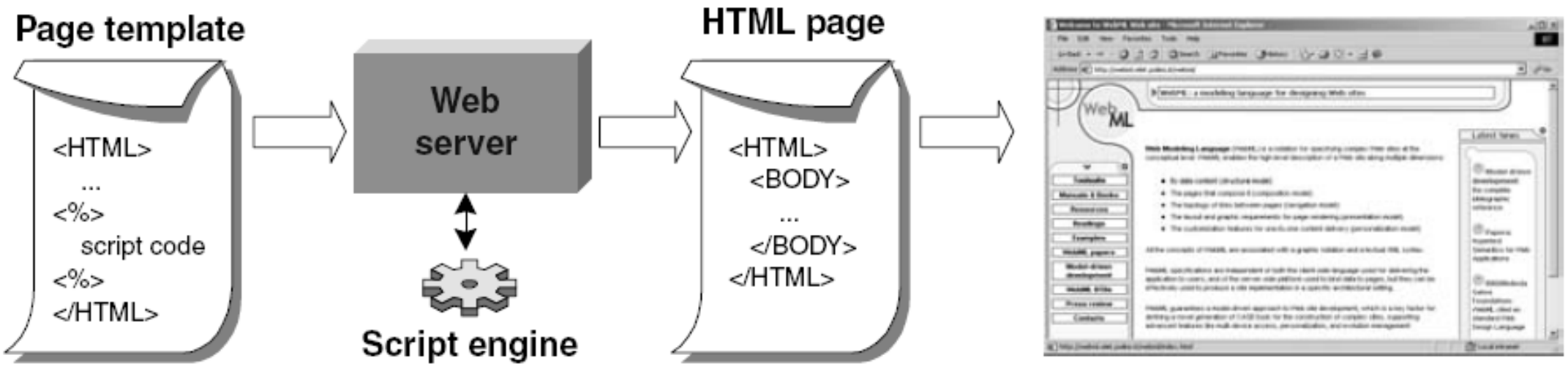
```
1:  import java.io.*;
2:  import javax.servlet.*;
3:  import javax.servlet.http.*;
4:
5:  public class RequestInfo extends HttpServlet {
6:
7:    public void doGet(HttpServletRequest request,
8:                      HttpServletResponse response)
9:                      throws IOException, ServletException {
10:     response.setContentType("text/html");
11:     PrintWriter out = response.getWriter();
12:     out.println("<HTML>");
13:     out.println("<HEAD>");
14:     out.println("<TITLE>Request Information Example</TITLE>");
15:     out.println("</HEAD>");
16:     out.println("<BODY>");
17:     out.println("<H3>Request Information Example</H3>");
18:     out.println("Method: " + request.getMethod());
19:     out.println("<BR>");
20:     out.println("Request URI:  " + request.getRequestURI());
21:     out.println("<BR>");
22:     out.println("User Agent:" + request.getHeader("User-Agent"));
23:     out.println("</BODY>");
24:     out.println("</HTML>");
25:   }
26: }
```


Request Information Example - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://WebMLBook/Examples/servlet/RequestInfo
Go
Request Information Example
Method: Get
Request URI: /Examples/servlet/RequestInfo
User Agent: Mozilla/4.0 (compatible: MSIE 5.01; Windows NT 5.0)
Done
My Computer

Server-Side Scripting

- Writing a server side program (e.g. servlet)
  - requires programming skills
- Programming is applied:
  - not only to calculate the dynamic portions of the page
  - but also the fixed textual content and HTML markup
- The definition of the static content :
  - Requires only HTML skills
  - And could be delegated to the graphic designer
- **Server-side scripting:**
  - Inserting into an HTML page template some programming instructions
  - that are **executed by a server program**
  - to calculate the dynamic parts of the page

Server-Side Scripting



Server-side scripting

- Various incarnations of the server-side scripting approach are available,
  - *Active Server Pages (ASP) language by Microsoft,*
  - *Java Server Pages (JSP) language by Javasoft,*
  - *PHP open language*

ASP Example (Ancient!)

```
<HTML>
<HEAD>
<TITLE>Request Information Example</TITLE>
</HEAD>
<BODY>
<H3>Request Information Example</H3>
Method: <%Response.Write(Request.ServerVariables("request_method"))%>
<BR>
Request URI: <%Response.Write(Request.ServerVariables("URL"))%>
<BR>
User Agent: <%Response.Write(Request.ServerVariables("http_user_agent"))%>
</BODY>
</HTML>
```

CSS

CSS Outline

- Introduction
- Style Types
- Selectors
- Cascading
  - Inheritance
- CSS3

Introduction

- XHTML: Structure of web pages
  - Almost all XHTML tags have their own presentation
    - Traditionally, HTML is used for document structure & *presentation* (physical formatting)

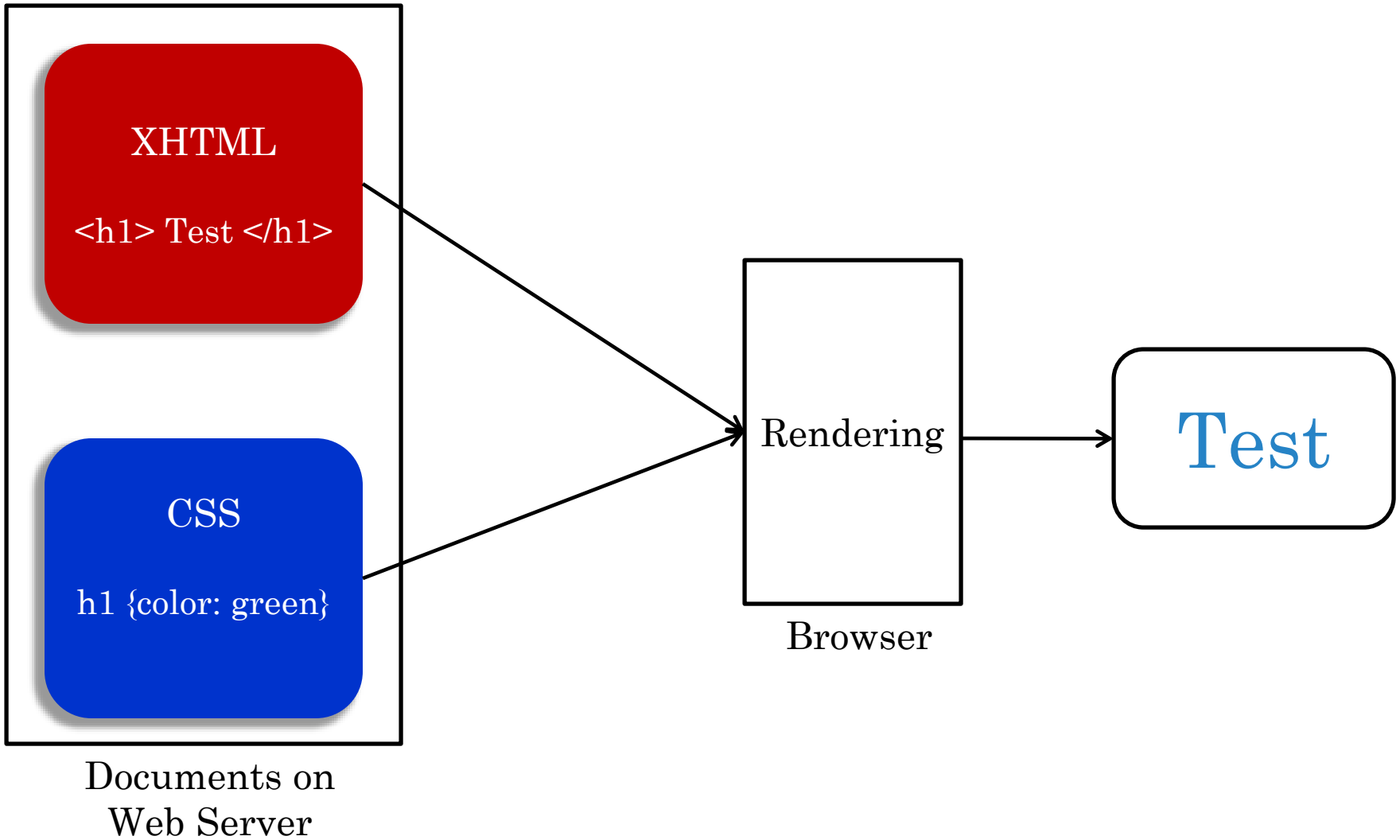
```
<i><font size=10 color=red> test </font></i>
```

- No reusability
  - Tremendous repeated formatting tags
- Structure and Formatting mixed up mess

Introduction (cont'd)

- (in XHTML) Separation between
  - Content & Structure
  - Presentation
- Content & Structure are specified by XHTML tags
- Cascading Style Sheet (CSS) defines the presentation and style
  - How elements are presented in web pages!!
  - Override the default presentation of elements
    - If presentation is not specified, browser uses its own default presentation for HTML elements

Introduction (cont'd)



Introduction to CSS

- Created by Hakon Wium Lie of MIT in 1994
- The W3C standards (CSS 1,2,3) to control visual presentation of web pages
- Uses a different syntax than HTML/XHTML
  - A different language
- Works with the common visual browsers
- Greatly simplifies visual design, site management and content maintenance

Håkon Wium Lie

- Chief technical officer of the **Opera** Software company
  - as of 2016
- Co-creator of the CSS web standard (**1994**)
- At the time, Lie was working with Tim Berners-Lee at CERN
- CSS1 released in 1996
- Bert Bos was influential
  - He became co-author of CSS1 and is regarded as co-creator of CSS

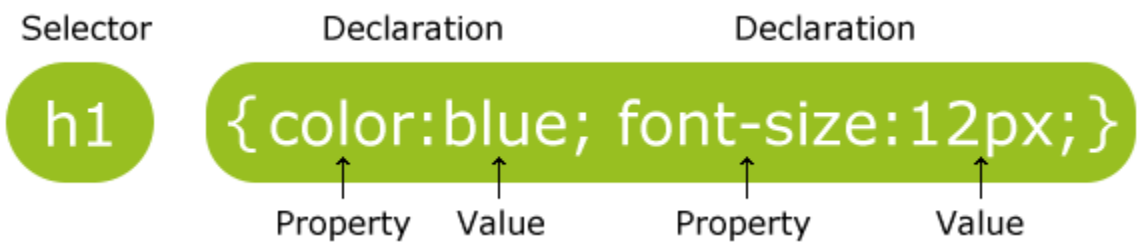
Introduction (cont'd)

- Advantages
  - Reuse styling rules in multiple pages
    - Make easier site management
    - Saves time
  - More control over layout
  - Styles + JavaScript → dynamic presentation
  - Multiple presentation of the same content
    - Media dependent presentation

CSS Styles

- Styles are defined by *style rules*
  - Selector + Declaration block (property + value)
- Structure of style rules

```
Selector1 {
    Property1: Value1;
    Property2: Value2;
    …
    /* This is comment */
}
```



Selectors

- Tag based
- Class based
- ID based
- Attribute based
- …

Sample Properties

- Font:
  - font-family
  - font-size
  - font-style
- Text:
  - text-align
  - color
  - letter-spacing
  - word-spacing
- Background:
  - background-color
  - background-image

Sample Properties (cont'd)

- Border:
  - border-color
  - border-style
  - border-width
- Position: bottom, left, right, …
- Table
  - spacing
  - color
  - alignment
- List
  - style
- Complete list in www.w3.org/style/css/

Values

- Values depend on property
- Predefined keywords
  - left, right, italic, none, …
- Color
  - red, green, blue, …
  - #XXYYZZ where 00 <= XX,YY,ZZ <=FF
  - rgb(XX, YY, ZZ) where 0 <= XX,YY,ZZ <=255
  - …
- Size & Length & Font-size
  - cm, in, mm, pt, px, %,, …

Inline Styles

- Add styles to each tag within HTML file!!!
  - Used to format a single HTML element
  - Selector is implicitly specified
  - Style is given as an attribute

```
<h1 style="color:red; font-family: sans-serif">
  Test Heading 1</h1>
```

- Element based
  - Hard to update
  - Violates structure-style separation

Internal (embed) Styles

- A style is used in the entire HTML file
- Used to control style of elements in a single web page (e.g., all h1)

```
<head>
  <style type="text/css">
      h1 {color: red;
          font-family: sans-serif;}
  </style>
</head>
```

External Styles

- A text file containing style rules
- Used to control style in multiple web pages
- Example
  - A text document with .css extension contains

```
h1, h2, h3, h4, h5, h6 {
    color: #FF0000;
    font-family: sans-serif;
}
```

External Styles (cont'd)

- External style file is used in XHTML web page through linking it to the web page

```
<head>
<title>External CSS</title>
<link href="external_css.css" rel="stylesheet"
 type="text/css" />
</head>
```

External Styles Advantages

- Avoid repeating styles for each page
  - It is easier to update whole site
- CSS can be cached independent of HTML
- Different style sheets can be attached to the same document
  - Personalized & Customization & Media dependent
- A style sheet can import and use styles from other style sheets
- Many-to-Many relationship between HTMLs and CSSs

Media Depended Presentation

- Web page presentation can be depended on media

```
<style type="text/css" media="screen">
<link rel="stylesheet" href="style.css" media="all">
```

- Available media types
  - `all`
  - `screen`
  - `print`
  - `speech`
  - ...

Media Depended Presentation

index.html | blue_format.css

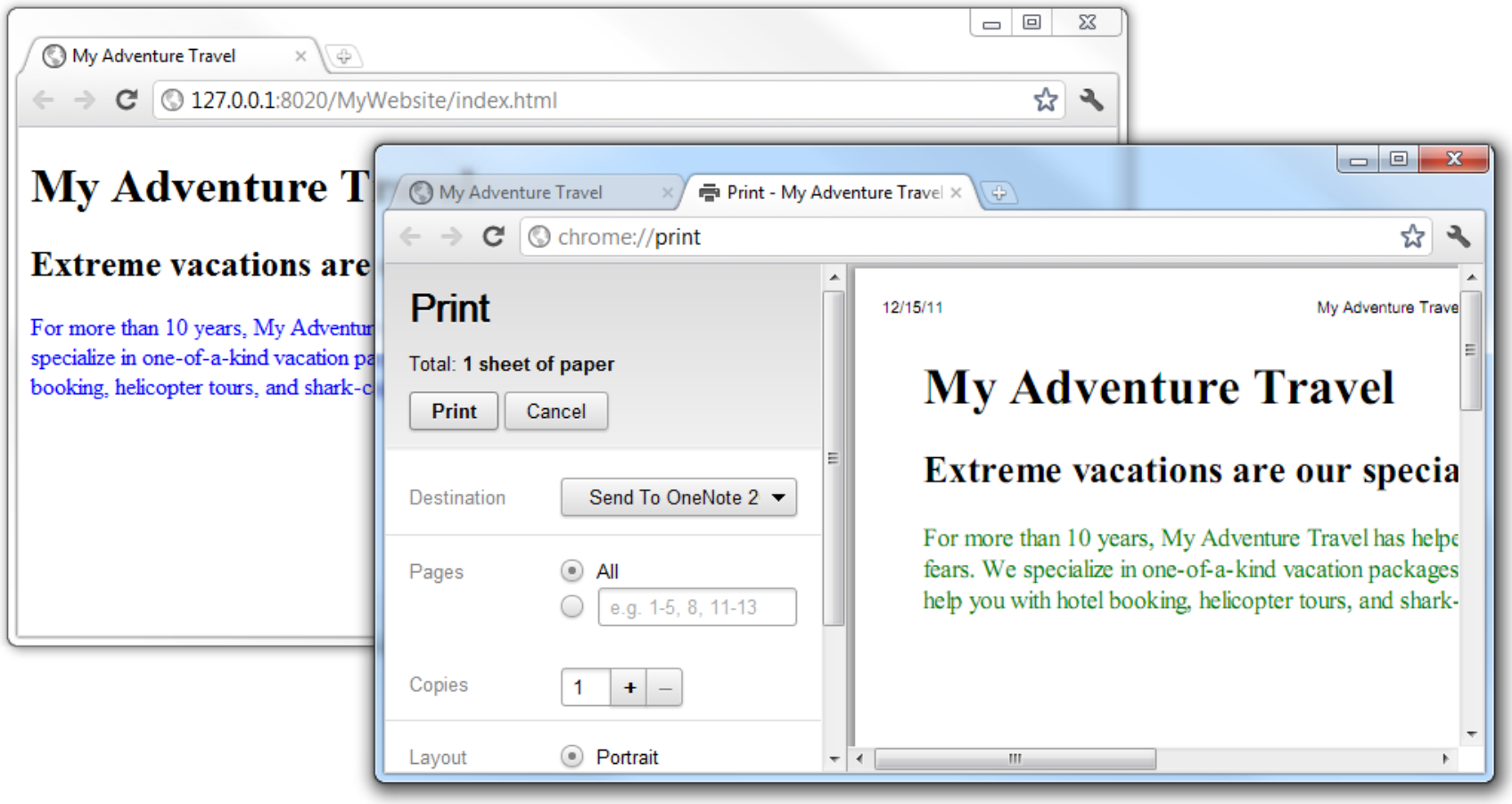
```
1 p {color: blue}
2
```

green_format.css

```
1 p {color: green}
2
```

```
<!DOCTYPE html>
<html>
    <head>
        <title>My Adventure Travel</title>
        <link rel="stylesheet" type="text/css" href="green_format.css" media="print" />
        <link rel="stylesheet" type="text/css" href="blue_format.css" media="screen" />
    </head>
    <body>
        <h1 class="myClass1">My Adventure Travel</h1>
        <h2>Extreme vacations are our specialty</h2>
        <p>For more than 10 years, My Adventure Travel has helped customers fulfill their
dreams and conquer their fears. We specialize in one-of-a-kind vacation packages to the
most exciting destinations in the world. Let us help you with hotel booking, helicopter
tours, and shark-cage rentals, either online or in person.</p>
    </body>
</html>
```

Media Depended Presentation



Selectors

- *: universal selector (everything)
- XHTML Tags
- Attribute based
  - Special attributes
    - Class based: per-class style
    - ID based: per-element styles
    - In combination with tag names
- DOM tree based
  - Child & Descendant, Sibling, …
- Pseudo-class & Pseudo-element

We may not need all these selectors at the beginning; however, they are powerful tools that simplify complex CSS rules

Element Selector

```
<head>
  <style type="text/css">
      * {color: blue}
      h1 {text-decoration: underline}
      p {font-style: italic}
  </style>
</head>


<body>
<h1> Testing Element Selector </h1>
<p> This is a paragraph </p>
</body>
```

id Selector

- Assign ID to elements

```
<h2 id="blue_heading"> This is blue heading </h2>
<h2 id="red_heading"> This is red heading </h2>
```

- Define style for each ID

```
#blue_heading {color:blue;}
#red_heading {color:red;}
```

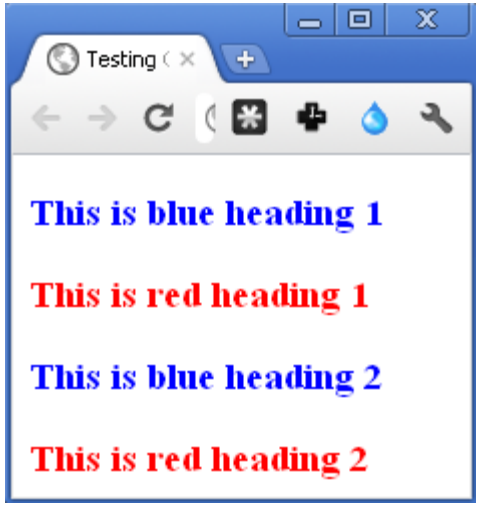
class Selector

- Assign class to element

```
<h3 class="blue_heading"> This is blue heading 1</h3>
<h3 class="red_heading"> This is red heading 1</h3>
<h3 class="blue_heading"> This is blue heading 2</h3>
<h3 class="red_heading"> This is red heading 2</h3>
```

- Define style of each class

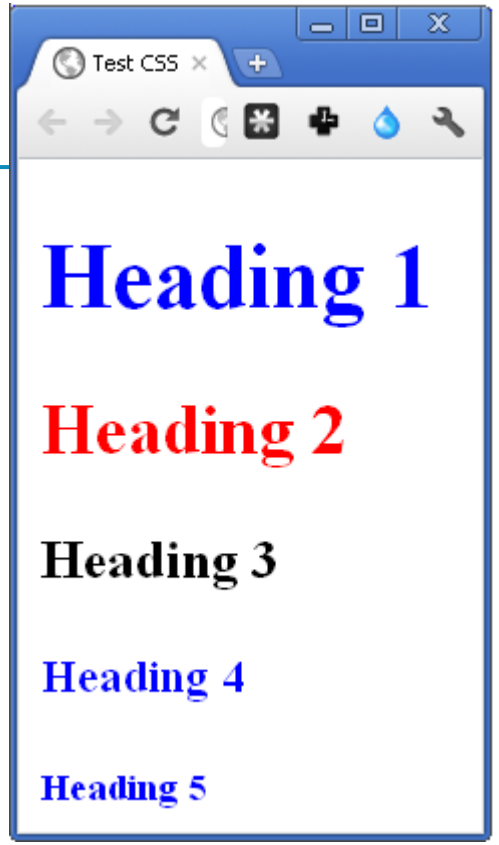
```
.blue_heading {color:blue;}
.red_heading {color:red;}
```



Combined Element & Class Selector

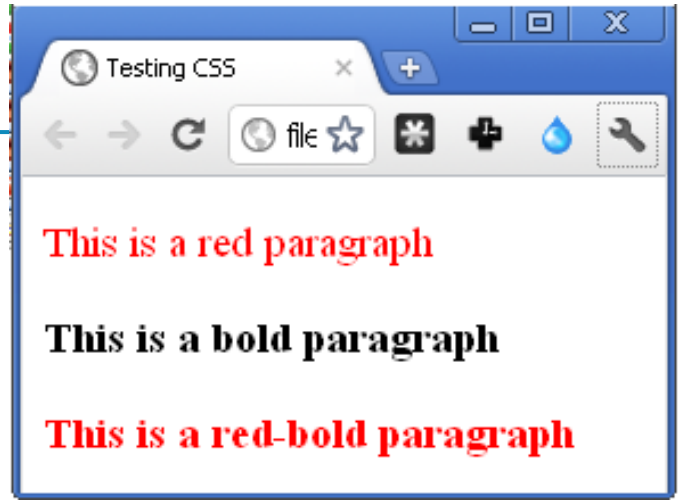
```
<style type="text/css">
  .all {color: blue;}
  h2.all {color: red;}
  h3.all {color: black;}
</style>

<h1 class="all"> Heading 1 </h1>
<h2 class="all"> Heading 2 </h2>
<h3 class="all"> Heading 3 </h3>
<h4 class="all"> Heading 4 </h4>
<h5 class="all"> Heading 5 </h5>
```



- Note:
  h2.all is more specific than .all, so it applies late
- The order of .all and h2.all is not important

Multiple Classes




```
<style type="text/css">
  .red {color:red;}
  .bold {font-weight:bold;}
</style>
```

If two selectors have the same specificity, the last one to be declared wins

```
<body>
  <p class="red"> This is a red paragraph </p>
  <p class="bold"> This is a bold paragraph  </p>
  <p class="red  bold"> This is a red-bold paragraph </p>
</body>
```

and in CSS

- XHTML:

```
<div class="green_color">
<p>
This is a green paragraph using div. But, <span id="red_color"> this is
  red </span> using span.
</p>
</div>
```

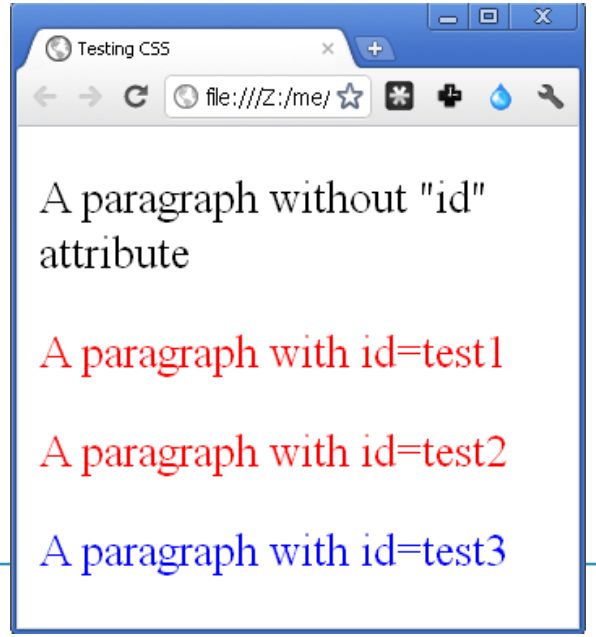
- CSS:

```
.green_color {
  color:green;
   font-weight:bold;}
#red_color {color:red;}
```

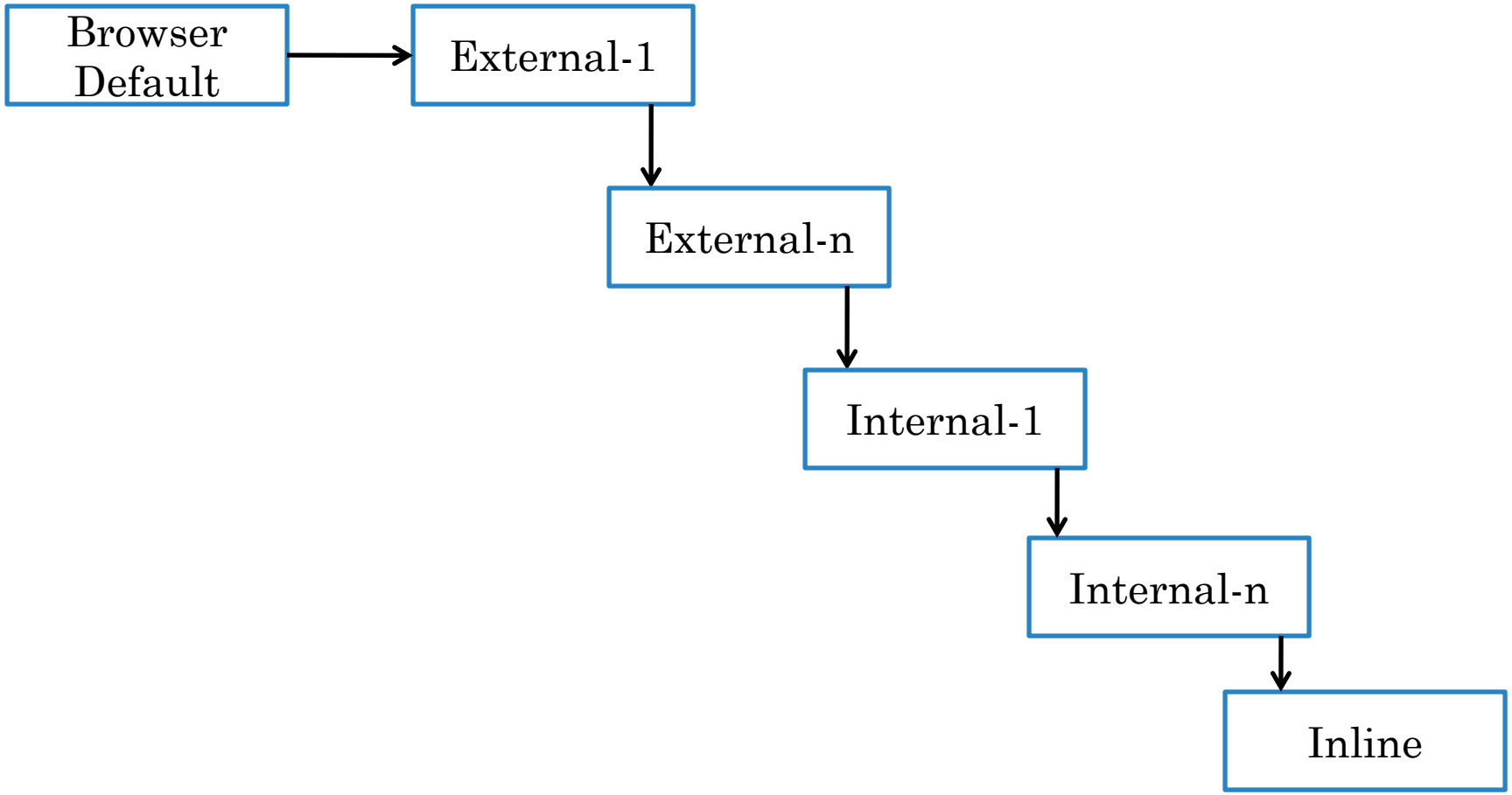
Attribute Selector

```
<p>  A paragraph without "id" attribute </p>
<p myid="test1"> A paragraph with id=test1 </p>
<p myid="test2"> A paragraph with id=test2 </p>
<p myid="test3"> A paragraph with id=test3 </p>
```

```
p[myid] {color:red}
p[myid="test3"] {color:blue}
```



Cascading

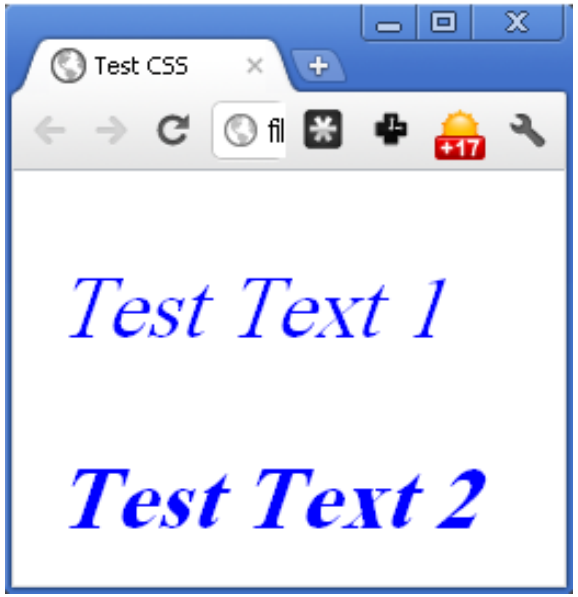

Browser Default
External-1
External-n
Internal-1
Internal-n
Inline

Cascading (cont'd)

test.css: `p {font-style: italic;}`

XHTML:

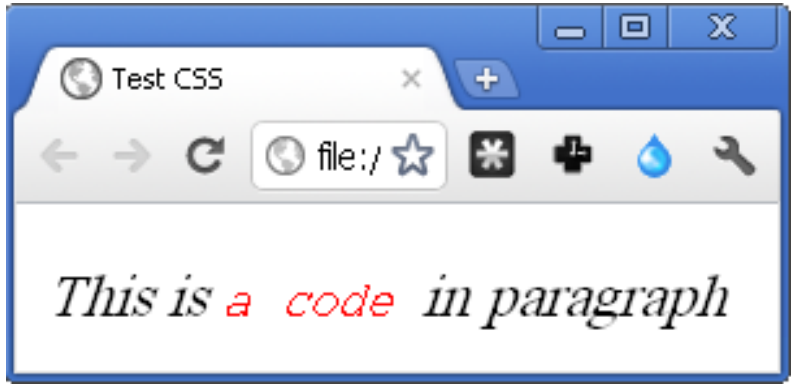
```
<head>
  <link href="test.css"  rel="stylesheet" type="text/css" />
  <style type="text/css">
      p {color: blue;}
  </style>
</head>
<body>
<p> Test Text 1</p>
<p style="font-weight: bold;"> Test Text 2 </p>
</body>
```



Inheritance

- XHTML document is interpreted as a tree
  - DOM tree
- Children inherit some styles from their parent
  - Not all properties, e.g. **`border`**

```
<style type="text/css">
  p {font-style: italic;}
  code {color: red;}
</style>
```




```
<p> This is <code> a code </code> in paragraph </p>
```

Conflicting Styles & Overriding

- What happen if multiple style rules specify different values for the same property of an element?
  - External style: `p {color: red}`
  - Internal style: `p {color: blue}`
  - They are conflicting
- What is the final rule?
  - It depends on
    - Rule types, order, …
  - Specified by the overriding algorithm

Overriding

- In general,
  - Priority 1: more specific
    - Inline > ID > Class > Element (Tag) > Div/Span
  - Priority 2: more closer to the element if they are the same selector (e.g., both are id selector)
    - Inline > Internal
    - Inline > External
    - Internal <?> External
      - *The style that comes after the other*

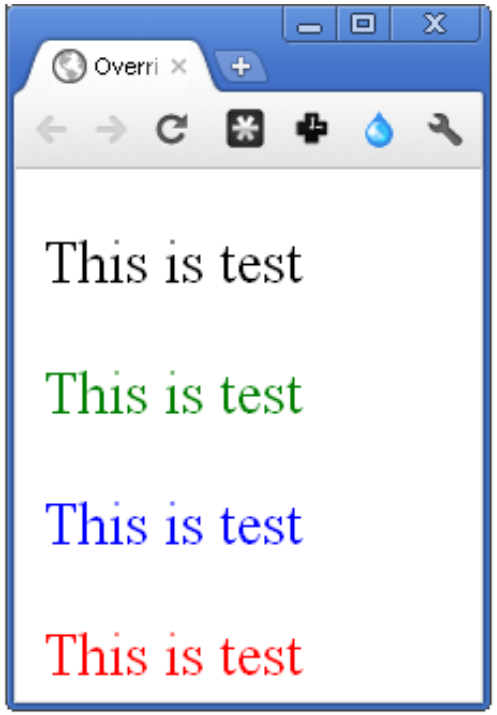
styles override more general styles

- Children's style overrides parent's style

Overriding (cont'd)

```
<style type="text/css">
  p{color: red;}
  p.c{color: blue;}
  p#i{color: green;}
</style>

<p class="c" id="i"
 style="color: black;"> This is test </p>
<p class="c" id="i"> This is test  </p>
<p class="c"> This is test </p>
<p> This is test </p>
```



Overriding (cont'd)

- To prevent overriding, add `!important`

```
h1{font-style: italic;
color: blue !important}

<h1> Heading 1 </h1>
<h1 style="font-style: normal;
color=red">Heading 1</h1>
```

CSS3

- W3C is working on CSS3
  - The next standard of CSS
- CSS3 is split up into modules
  - More than 30 modules
  - Old specification has been split into smaller pieces, and new ones are also added
- The CSS3 specification is still under development
  - Many of the new CSS3 properties have been implemented in modern browsers

(Sample) CSS3 Modules

- 2D Transforms
  - Manipulation of content in two dimensions, such as rotating, scaling, and skewing objects
- 3D Transforms
  - Extends 2D Transforms to manipulate elements in a three-dimensional space
- Animations
  - To modify CSS property values over time, such as position or color, to create animated layouts

(Sample) CSS3 Modules

- Backgrounds and Border
  - Multiple backgrounds, a variety of background properties for positioning and sizing, new border properties with images and shadows
- Web Fonts
  - Defines downloadable fonts
- Multi-column Layout
  - Defines how to flow text into many columns

CSS3 Examples

- Text shadowing

```
h1 {text-shadow: 5px 5px 5px #FF0000;}
```

Text-shadow effect!

- Web font

```
<style type="text/css">
@font-face{
  font-family: myFirstFont;
  src: url('sansation_light.ttf')
}

div {font-family:myFirstFont;}
</style>

<div>Using the new font</div>
```

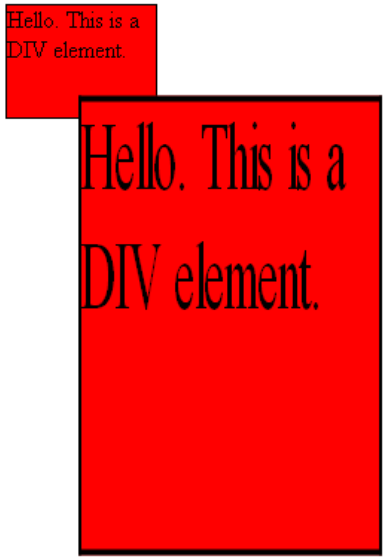
CSS3 Examples: 2D Transformation

```
<div>Hello. This is a DIV element.</div>

<div id="div2">Hello. This is a DIV element.</div>
```

- `transform:rotate(30deg);`

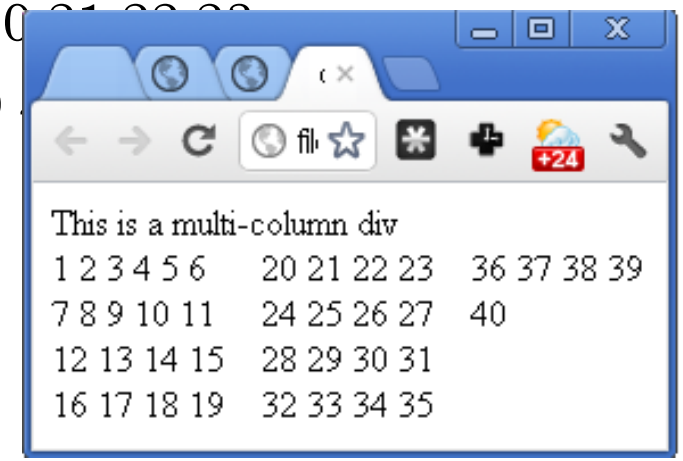
- `transform:scale(2,4);`



CSS3 Examples: Multi-Column

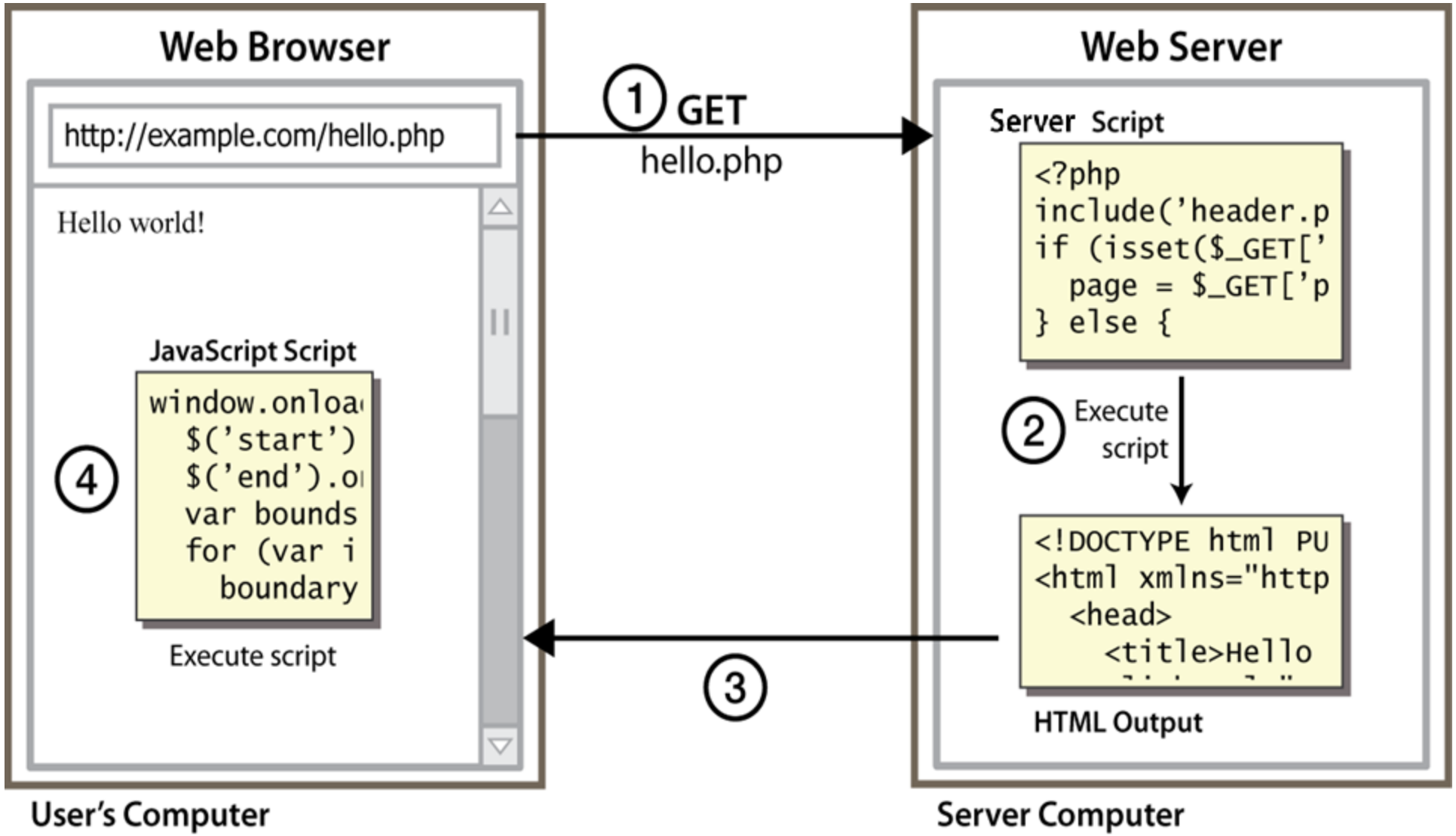
```
<style type="text/css">
.newspaper {column-count:3;}
</style>

<body>
This is a multi-column div.
<div class="newspaper">
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39
</div>
</body>
```



JavaScript

Client Side Scripting



Why use client-side programming?

Server-side programming (e.g. PHP) already allows us to create dynamic web pages.

Why also use client-side scripting?

- client-side scripting (JavaScript) benefits:
  - **usability**: can modify a page without having to post back to the server (faster UI)
  - **efficiency**: can make small, quick changes to page without waiting for server
  - **event-driven**: can respond to user actions like clicks and key presses

So, why use server-side programming?!

- **Access** to server's private **data**
  - client can't see source code
- **power**: can write files, open connections to servers, connect to databases, ...
  - Processing power, memory, ...
- **compatibility**: not subject to browser compatibility issues

JavaScript History

- JavaScript created in **1995** by Brendan Eich, then working at **Netscape** and now of **Mozilla**
- JavaScript was not always known as JavaScript:
  - the original name was **Mocha**
  - the name was changed to **LiveScript**
  - then the name **JavaScript** was adopted
  - (Java was very popular around then)

JavaScript, Today

- A completely new and exciting cycle of evolution
- With new developments such as the **Nodejs** platform
  - allowing us to use JavaScript on the **server-side**
- And HTML5 APIs
  - to control user **media**
  - open up **web-sockets** for always-on communication
  - get data on geographical location and device features
  - …

JavaScript, Today (cont'd)

- Many client side frameworks are based on JavaScript
  - jQuery
  - Bootstrap
  - AngularJS
  - React

What is Javascript?

- A lightweight programming language
- A "**scripting language**"
  - used to make web pages interactive
  - insert **dynamic text** into HTML (ex: user name)
  - **react to events** (ex: page load user click)
  - get information about a **user's computer** (eg: browser type)
  - perform calculations on **user's data** (ex: form **validation**)

What is Javascript?

- A web standard
  - but not supported identically by all browsers
  - The differences (among browsers) are being smaller
- NOT related to Java!
  - other than by name and some syntactic similarities

Javascript vs Java

- **interpreted**, not compiled
- more **relaxed syntax** and rules
  - **fewer** and "looser" **data types**
  - variables **don't need to be declared**
  - errors often **silent** (few exceptions)
- **key construct** is the **function** rather than the class
  - "**first-class**" functions are used in many situations
- contained within a web page and integrates with its HTML/CSS content

JavaScrip

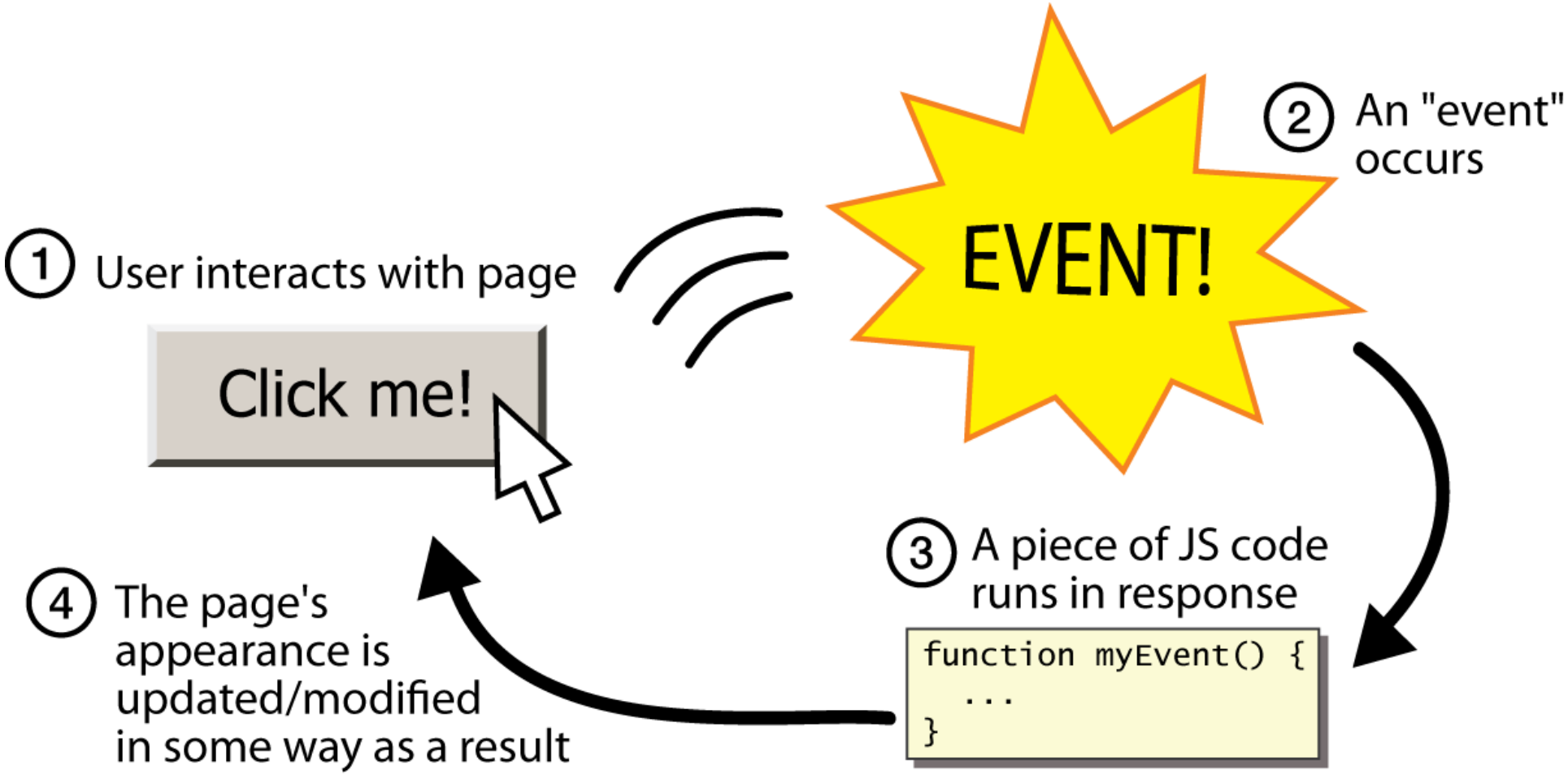
- Similar to other scripting languages (like PHP):
  - interpreted, not compiled
  - **relaxed about syntax**, rules, and types
  - **case-sensitive**
  - built-in **regular expressions** for powerful text processing
- JS is **object-oriented**: noun.verb()
- JS focuses on user interfaces and interacting with a document
- JS code runs on the client's browser
  - Recently, more applications, even server-side (e.g., Node.js)

Linking to a JavaScript file: `script`

```
<script src="filename" type="text/javascript"></script>
```
HTML

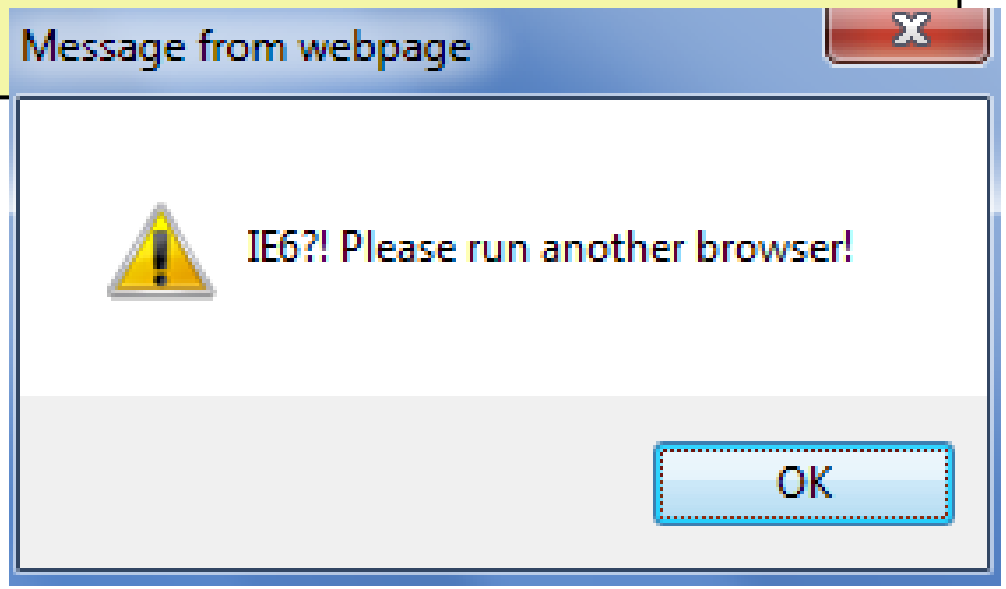
- script tag should be placed in HTML page's **head**
- script code is stored in a separate **.js** file
- JS code can be placed directly in the HTML file (like CSS)
  - but this is bad style
    - separate content, presentation, and behavior)

Event-driven programming



A JavaScript statement: `alert`

```
alert("IE6?! Please run another browser!");
```



- a JS command
- pops up a dialog box with a message
- Nowadays, popup are **not common**
- An alert window is **disruptive and annoying**
- A better user experience: have the message **appear on the page**…

Event-driven programming

- Programs used to start with a main method
- JavaScript programs instead **wait** for user actions called **events** and respond to them
- event-driven programming: writing programs driven by [user] events
- Let's write a page with a clickable button that pops up a "Hello, World" window...

Buttons

```
<button>Click me!</button>
```
HTML

- button's text appears inside tag; can also contain images
- To make a responsive button or other UI control:
  1. choose the control (e.g. button) and event (e.g. mouse 1. click) of interest
  2. write a JavaScript function to run when the event occurs
  3. attach the function to the event on the control

JavaScript functions

```
function name() {
   statement ;
   statement ;
   ...
   statement ;
}
```

```
function myFunction() {
    alert("Hello!");
    alert("How are you?");
}
```

- the above could be the contents of example.js linked to our HTML page

- statements placed into functions can be evaluated in response to user events

```
<button onclick="myFunction();">
Click me!</button>
```

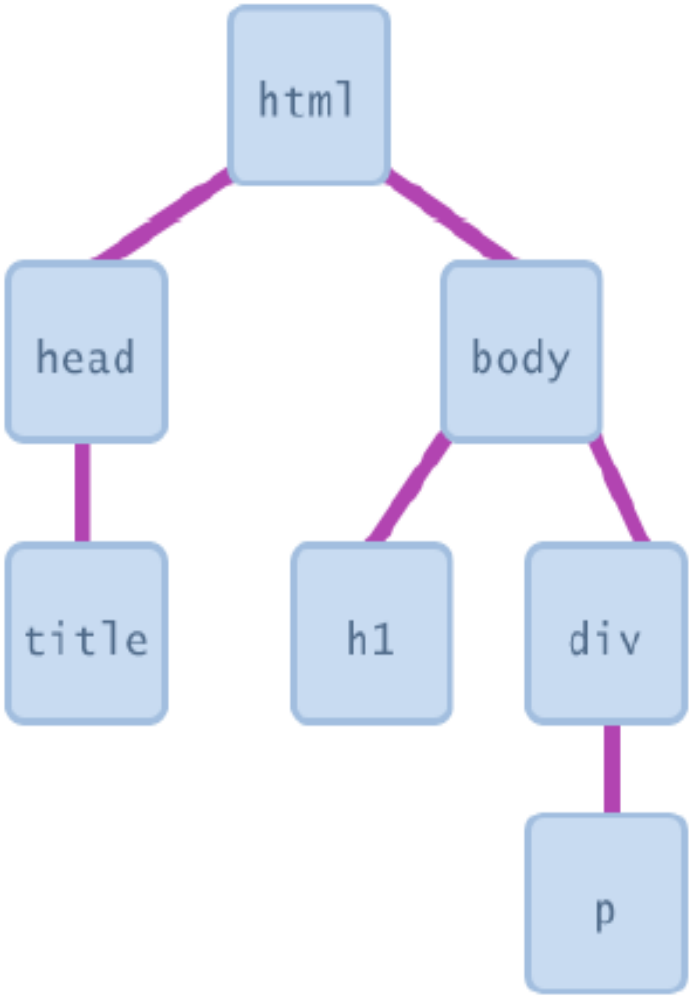
Event handlers

```
<button onclick="myFunction();">Click me!</button>
```

- JavaScript functions can be set as event handlers
  - when you interact with the element, the function will execute
- onclick is just one of many event HTML attributes we'll use
  - onmouseover, onmouseout, onmousedown, onmouseup, …
  - onkeydown, onkeypress, …

Document Object Model (DOM)

- Most JS code manipulates elements on an HTML page
- we can examine elements' **state**
  - e.g. see whether a box is checked
- we can **change state**
  - e.g. insert some new text into a div
- we can **change styles**
  - e.g. make a paragraph red



DOM element objects

HTML

```
<p>
  Look at this octopus:
  <img src="octopus.jpg" alt="an octopus" id="icon01" />
  Cute, huh?
</p>
```

DOM Element Object

Property	Value
tagName	"IMG"
src	"octopus.jpg"
alt	"an octopus"
id	"icon01"

JavaScript

```
var icon = document.getElementById("icon01");
icon.src = "kitty.gif";
```

Accessing elements: document.getElementById

```
var name = document.getElementById("id");
```
JS

```
<button onclick="changeText();">Click me!</button>
<span id="output">replace me</span>
<input id="textbox" type="text" />
```
HTML

```
function changeText() {
    var span = document.getElementById("output");
    span.innerHTML = "You Replaced Me!";
    span.className = "MyClass";

    var textBox = document.getElementById("textbox");
    textbox.value = "Hello!!";
    textbox.style.color = "red";
}
```
JS

Accessing elements: `document.getElementById`

- **document.getElementById** returns the DOM object for an element with a given id
- can change the text
  - inside most elements by setting the **innerHTML** property
  - in form controls by setting the **value** property

Changing element style: `element.style`

Attribute	Property or style object
color	color
padding	padding
background-color	backgroundColor
border-top-width	borderTopWidth
Font size	fontSize
Font famiy	fontFamily

Example

```
function changeText() {
    //grab or initialize text here

    // font styles added by JS:
    text.style.fontSize = "13pt";
    text.style.fontFamily = "Comic Sans MS";
    text.style.color = "red"; // or pink?
}
```
JS

More Javascript Syntax

Variables

```
var name = expression;
```
JS

```
var clientName = "Connie Client";
var age = 32;
var weight = 127.4;
```
JS

- variables are declared with the **var** keyword (case sensitive)
- types are not specified, but JS does have types ("loosely typed")
  - **string, number, boolean, object, function, null, undefined**
  - Three types of ***objects***: **Object, Date, Array**
  - can find out a variable's type by calling `typeof`

Number type

```
var enrollment = 99;
var medianGrade = 2.8;
var credits = 5 + 4 + (2 * 3);
```
JS

- integers and real numbers are the same type (no int vs. double)
- same operators: + - * / % ++ -- = += -= *= /= %=
- similar precedence to Java
- many operators auto-convert types: "2" * 3 is 6

Comments (same as Java)

```
// single-line comment
/* multi-line comment */
```

- identical to Java's comment syntax
- recall: 4 comment syntaxes
  - HTML: <!-- comment -->
  - CSS/JS: /* comment */
  - Java/JS: // comment

Math object

```
var rand1to10 = Math.floor(Math.random() * 10 + 1);
var three = Math.floor(Math.PI);
```

- methods: `abs, ceil, cos, floor, log, max, min, pow, random, round, sin, sqrt, tan`
- properties: `E, PI`

Special values: null and undefined

```
var ned = null;
var benson = 9;
// at this point in the code,
// ned is null
// benson's 9
// caroline is undefined
```

- `undefined` : has not been declared, does not exist
- `null` : exists, but was specifically assigned an empty or null value

Logical operators

- > < >= <= && || ! == != === !==
- most logical operators automatically convert types:
  - 5 < "7" is true
  - 42 == 42.0 is true
  - "5.0" == 5 is true
- === and !==
  - strict equality tests; checks both type and value
  - "5.0" === 5 is false

if/else statement (same as Java)

```
if (condition) {
     statements;
} else if (condition) {
     statements;
} else {
     statements;
}
```

- identical structure to Java's if/else statement
- JavaScript allows almost anything as a condition

Boolean type

```
var iLike190M = true;
if ("web devevelopment is great") { /* true */ }
if (0) { /* false */ }
```

- any value can be used as a Boolean
  - "falsey" values: 0, 0.0, NaN, "", null, and undefined
  - "truthy" values: anything else
- converting a value into a Boolean explicitly:
  - `var boolValue = Boolean(otherValue);`
  - `var boolValue = ! (otherValue);`

for loop (same as Java)

```
var sum = 0;
for (var i = 0; i < 100; i++) {
     sum = sum + i;
}
```

```
var s1 = "hello";
var s2 = "";
for (var i = 0; i < s.length; i++) {
     s2 += s1.charAt(i) + s1.charAt(i);
}
// s2 stores "hheellllloo"
```

while loops (same as Java)

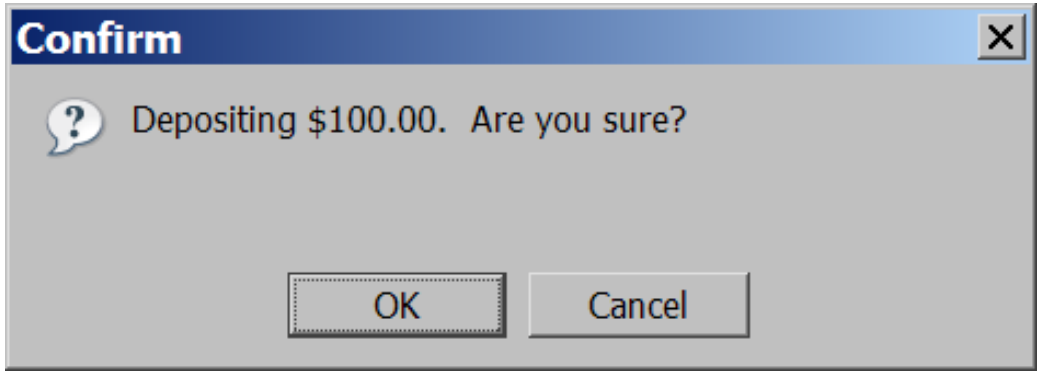
```
while (condition) {
     statements;
}
```

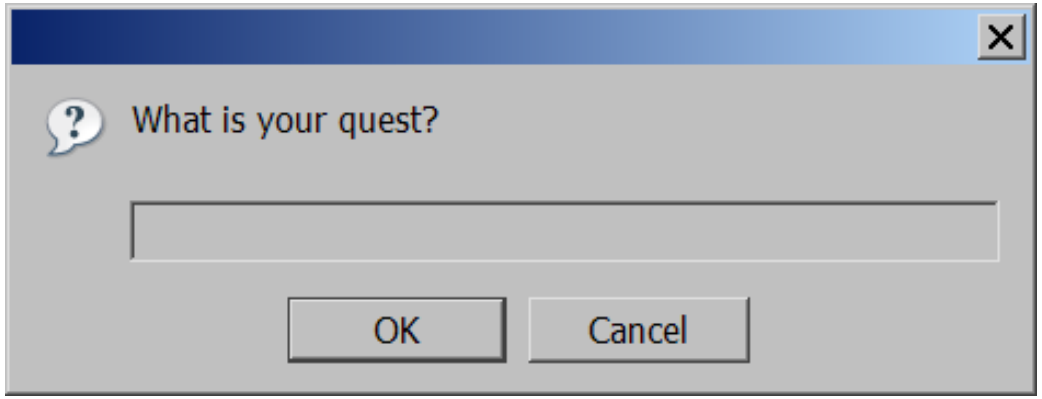
```
do {
   statements;
} while (condition);
```

- break and continue keywords also behave as in Java

Popup boxes

```
alert("message"); // message
confirm("message"); // returns true or false
prompt("message"); // returns user input string
```





Arrays

```
var name = []; // empty array
var name = [value, value, ..., value]; // pre-filled
name[index] = value; // store element
```

```
var ducks = ["Huey", "Dewey", "Louie"];
var stooges = []; // stooges.length is 0
stooges[0] = "Larry"; // stooges.length is 1
stooges[1] = "Moe"; // stooges.length is 2
stooges[4] = "Curly"; // stooges.length is 5
stooges[4] = "Shemp"; // stooges.length is 5
```

Array methods

```
var a = ["Stef", "Jason"]; // Stef, Jason
a.push("Brian"); // Stef, Jason, Brian
a.unshift("Kelly"); // Kelly, Stef, Jason, Brian
a.pop(); // Kelly, Stef, Jason
a.shift(); // Stef, Jason
a.sort(); // Jason, Stef
```

- array serves as many data structures: list, queue, stack, ...
- methods: `pop/push, shift/unshift, join, reverse, sort, toString`
  - push and pop add / remove from back
  - unshift and shift add / remove from front
  - shift and pop return the element that is removed

String type

```
var s = "Connie Client";
var fName = s.substring(0, s.indexOf(" ")); // "Connie"
var len = s.length; // 13
var s2 = 'Melvin Merchant';
```

- methods: **`charAt`**, **`indexOf`**, **`lastIndexOf`**, **`replace`**, **`split`**, **`substring`**, **`toLowerCase`**, **`toUpperCase`**
- length property (not a method as in Java)
- concatenation with + :
  - 1 + 1 is 2, but "1" + 1 is "11“
  - But "2" * 3 is 6

More about `String`

- accessing the letters of a String:
- escape sequences behave as in Java: \' \" \& \n \t \\
- converting between numbers and Strings:

```
var count = 10;
var s1 = "" + count; // "10"
var s2 = count + " bananas, ah ah ah!";
// "10 bananas, ah ah ah!"
var n1 = parseInt("42 is the answer"); // 42
var n2 = parseFloat("booyah"); // NaN
```

```
var firstLetter = s[0]; // fails in IE
var firstLetter = s.charAt(0); // does work in IE
var lastLetter = s.charAt(s.length - 1);
```

Splitting strings: split and join

```
var s = "the quick brown fox";
var a = s.split(" ");// ["the", "quick", "brown", "fox"]
a.reverse(); // ["fox", "brown", "quick", "the"]
s = a.join("!"); // "fox!brown!quick!the"
```

- split breaks apart a string into an array using a delimiter
  - can also be used with regular expressions (seen later)
- join merges an array into a single string, placing a delimiter between them

Conclusion

References

- W3Schools Online Web Tutorials
  http://www.w3schools.com/
- TutorialsPoint
  www.**tutorialspoint**.com/
- Internet Engineering course, Amirkabir University of Technology, Dr. Bahador Bakhshi
  http://ceit.aut.ac.ir/~bakhshis/
- Web Programming Step by Step by Marty Stepp, Jessica Miller, and Victoria Kirst

The End