Web Programming
10636316

Dr. Sufyan Samara
Computer Engineering Department
An Najah National University

1

Course outline

- Objective:
  - Provide the necessary knowledge for developing and administrating the applications running on World Wide Web.
- Course outline:
- **Introduction**
  - Background
  - Servers and protocols
  - Web hosting
  - Client/Server programming
  - Data exchange
- **HTML5**
  - The Document Model
  - Style Elements, Hyperlinking, and Interactivity
  - HTML5
  - XML and XHTML
- **CSS3**
  - Tags and Styles
  - Style Sheets
  - Properties
  - CSS Objects

2

Course outline – cont.

- **Client Side Scripting**
  - JavaScript
  - Overview of some libraries such as JQuery
  - Overview of other scripts such as TypeScript
  - Overview of some JavaScript toolkits and frameworks such as Bootstrap, Angular, React, Vue.js
  - Overview of some other toolkits and frameworks such as Flutter and Dart
- **Server Side Scripting**
  - PHP
  - Working with files, database, and security
- **Web Databases**
  - MySQL
  - Database Tools
- **AJAX**
- **XML**
- Overview of some Content Management Systems such as WordPress, Drupal, and Joomla
- Overview of some Model–View–Controller frameworks such as Laravel and CakePHP.
- **Textbook and references:**
  - Programming the World Wide Web by Robert W. Sebesta
  - W3Schools Online Web Tutorials

3

Grading

- Midterm exam 30%
- Project and HomeWorks and Report 30%
- Final exam 40%

4

Objectives

- Learn how to program using XHTML, HTML5, CSS3, JavaScript, PHP, MySQL, and AJAX.
- Provide a comprehensive introduction to the programming tools and skills required to build and maintain server sites on the web.
- Introduce some toolkits and frameworks
- XML.

5

Note

- The area of web programming is very wide
  - Many books can be found explaining just one web programming language or a tool
- Textbook:
  - Programming the world wide web, 8th edition, by Robert W. Sebesta
- Highly recommended reference
  - W3Schools Online Web Tutorials (https://www.w3schools.com/)
- Two exams, one project.
  - Midterm: 30%, Final: 40%, projects: 30%
- Tools
  - Open Source and free tools e.g. Visual Studio Code, sublime, notepad++.
  - Professional tools such as PhpStorm and Dreamweaver
  - We will use PhpStorm
  - For debugging we will use Xdebug

6

The internet

- Originated in the 1960s. Developed by the DoD
- A collection of computers connected together in a big network.
  - Using routers, switches, hubs, and many different servers and protocols.
  - A known widely used protocol is the TCP/IP protocol.
- Robust: provides a delivery mechanism in which a piece of information can take multiple routes from one point to another.
- Information are:
  - Requests for information or service
  - Responses containing that information or the result of a service
  - Informational responses (control messages), that is, reasons why the service or information could not be delivered
- Client/Server topology

7

Internet Protocol Addresses

- Every thing on the internet has to be uniquely identified.
  - A numeric address is used called the internet protocol address (IP).
    - A unique 32-bit number
      - About 4,3 billion addresses
    - Written as four 8-bit numbers, separated by periods.
    - A trend to use a new standard, IPv6, (128-bit)
    - Not user friendly.

8

Domain Names

- To access some information on one computer (the host machine) on the internet, you need its address.
- People are not good remembering numbers.
  - Map numeric addresses to names
  - Names begins with the host machine that are followed by the enclosing collection of machines are called domains.
    - edu, com, gov, ps, jo, de
    - Ex: eng.najah.edu
  - Names given to browsers have to be converted to numeric addresses
    - Domain Name Services (DNS)

9

World wide web

- On top of TCP/IP many applications limited protocols were developed
  - telnet: to allow a user to log in and access another computer.
  - File Transfer Protocol (ftp): to allow transferring of files among computers.
  - Usenet: electronic bulletin board.
  - mailto: to allow messages to be sent from one user to another.
- World Wide Web (www) or Web: originally to allow scientists to exchange documents.
  - Allow a user anywhere to search for and retrieve documents from databases on any number of different document-serving computers on the internet.
  - All document have been formed using hypertext.
    - A text containing links to other documents.
    - This developed to include hypermedia (pictures, sound, and video)

10

Web browsers

- Two computers communicating as Client/Server. (request/response)
- Document provided by web servers are requested by clients browsers.
- Although many protocols are supported by the web, the most common one is the Hypertext Transfer Protocol (HTTP).
- Most common browsers are: IE, Firefox, Safari,

11

Web Servers

- Handel client requests
- Each document on the server has a URL.
- All the communication between a Web server and a web browser is done based on a standard web protocol.
- When a web server runs, it informs the operating system that it is ready for any network request.
  - All network requests that meets the server communication protocol are send the the server.
  - The web server runs in the background
  - A web browser opens a network connection to a web server.
  - All HTTP commands include a URL.
    - Indicates the specifications of the host server machine
- Common Web servers are Apache and IIS.

12

Web Servers – cont.

- Each web server has a folder which contains the web documents.
  - The document is mapped to a URL
    - Example: C:\apache\httpd\myWebSite\index.html is mapped to http://www.myWebServer.com/myWebSite/index.html
- If a web server provides a document that is residing on a different computer, it is called a proxy server.
- Many web servers are now supporting other protocols than the HTTP such as ftp, gopher, news, and mailto.
  - In addition nearly all web servers supports interacting with database systems through Common Gateway Interface (CGI) programs and server-side scripts.

13

Uniform (or Universal) Resource Locators (URL)

- Used to identify documents (resources) on the internet.
- Formats: scheme : object-address
  - Scheme: communications protocol
    - http, ftp, gopher, telnet, file, mailto, and news
    - http: used to request and send eXtensible Hypertext Markup Language (XHTML) documents.
      - URL object-address: //fully-qualified-domain-name/path-to-document
      - A domain name has to include a port number
        - » For Http the default is 80, no need to be included unless it is something else.
    - file: using the file protocol to reference documents residing on the same computer the requester browser resides on.
      - file://path-to-document

14

URL – cont.

- No spaces allowed
  - Also semicolons, colons, and ampersands(&).
  - To include such disallowed characters, they must be encoded as a percent sign (%) followed by its ASCII hexadecimal code.

15

Multipurpose Internet Mail Extensions (MIME)

- Developed to specify the format of different kinds of documents to be sent via internet .
  - Depending on the format, the browser knows how to render it.
- MIME specifications have the form: type/subtype
  - Common types are: text, image, and video.
    - Common text subtypes are plain and html.
    - Common image subtypes are gif and jpeg
    - Common video subtypes are mpeg and quicktime.
    - Experimental subtypes begins with x- as in video/x-msvideo
      - How can a browser render an experimental type??
        - » An external program has to be provided, e.g. divx codec

16

The HTTP protocol

- Consists of two phases: the request and the response.
- Any communication (request or response) between browser and server consists of two parts: a header and a body.
- The header contains information about the communication.
- The body contains the data of the communication if there is any.

17

The Request Phase

- The general form
  - HTTP method domain part of the URL HTTP version
    - GET /myWebPage.html HTTP/1.1
    - Http methods include
      - GET returns the contents of the specified document
      - HEAD returns the header information for the specified document
      - POST Executes the specified document, using the enclosed data
      - PUT Replaces the specified document with the enclosed data
      - DELETE Deletes the specified document
      - The most common methods are GET and POST
  - Header fields: four categories
    - General: for general information such as date
    - Request: Included in request headers
      - One common request field is the accept field which specify a preference of the browser for the MIME type of the requested document e.g. Accept: text/plain or Accept: text/* for any.
    - Response: for response headers
    - Entity: Used in both request and response headers

18

The Request phase – cont.

- The general form – cont.
  - Blank line
    - The header of a request has to be followed by a blank line.
  - Message body
    - Not all requests has message body, for example requests that use the GET, HEAD, and DELETE methods do not have bodies.

19

The Response phase

- The general form
  - Status line: Includes
    - the http version used
    - A three-digit status code for the response, depending on the first digit it means
      - 1 informational
      - 2 success
      - 3 redirection
      - 4 client error, e.g.: 404 not found
      - 5 server error
    - A short textual explanation of the status code
    - Ex: HTTP/1.1 200 OK
  - Response header fields
  - Blank line
  - Response body

20

Security

- Consider sending a credit card number to a company, the security issues for this transaction are
  - Privacy: it must not be possible for the credit card number **to be stolen on its way** to the company's server
  - Integrity: it must not be possible for the credit card number **to be modified on its way** to the company's server
  - Authentication: it must be possible for both the purchaser and the seller to be certain of each other's **identity**.
  - Nonrepudiation: it must be possible to **prove legally** that the message was **actually sent and received**.
- Using encryption
  - Public-key encryption e.g. RSA

21

XHTML

- HTML is defined with the use of the Standard Generalized Markup Language (SGML)
- All web standards are managed by the World Wide Web Consortium (W3C)
  - W3.org
- HTML is the language of the World Wide Web
- XHTML is a formal redefinition of HTML under XML.
  - This makes the HTML code easier to validate, as it adheres more closely to the XML standard for data exchange.
  - The basic unit of definition used in XML and HTML is the tag
    - **<tag_name>Content inside tag</tag_name>**

22

The Document Model

- Refers to the collection of elements that make up a Web page
  - The core technology behind sharing information using standards such as XML and computer programs.
- Consists of layered containers
  - The physical HTML document (or file)
  - The document information (title, keywords, and so on)
  - The layout information (formatting and presentation)
  - The content items (text, hyperlinks, videos, pictures, add-ins, and so on)

23

Anatomy of the Document

- XHTML document must begin with an xml declaration element to identify that it is based on XML
  - <?xml version = “1.0” encoding = “utf-8”?>
- An HTML document consists of three parts:
  - Definition section: defines the content, recommended but can be left out
    - Contains the doc. Type, the HTML version, ..
    - <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN” "http://www.w3.org/TR/html4/loose.dtd">
      - HTML transitional is allowable under W3C but has features replaced by styles
      - The strict doctype
        - » <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 strict//EN” "http://www.w3.org/TR/html4/strict.dtd">
  - The HEAD: information to the browser
  - The BODY: all the content that is designed to be rendered
  - In HTML5, the definition section and the head section are mandatory.

24

The simplest HTML5 document template

```
<!DOCTYPE  html>
<html>
<head>
</head>
<body>
</body>
</html>
```

25

The contents of the HEAD Tag section

- The TITLE tag specifies what is displayed in the title bar of the browser
  - **<TITLE>My Title</TITLE>**
- The meta tags
  - Used by SEO (Search Engine Optimization)
  - Tells a search engine to index a page and give it some help in categorizing the information you're providing
    - **<meta name="keywords" content="HTML, book, tutorial" >**
    - **<meta name="description" content="course about WEB programming. . ." >**
    - **<meta name="author" content="Your name">**
    - **<meta name="copyright" content="2012">**
    - **<meta name="generator" content="the program you used">**
  - does not have an end tag
    - This is because all the information that you need is contained within the tag, in something called attributes.
- The client side scripts section
  - Now, these can also be put in the BODY of the document

26

The HTML BODY Tag

- Browsers may read all body before rendering, or render on the fly
  - Try to give as much information about the content in the beginning
    - Layout information
      - Each block of text contained between paragraph tags **<P> </P>**
      - E.g. images are rendered last, because they are to be downloaded
        - » Give size of images to reserve place

27

Hyperlinking

- The ability of linking content together
  - This can be a page, an image, a video, a sound, or a zip file containing a software release
- You will encounter two kinds of reference within an HTML page
  - Internal anchors: a reference within the current document, e.g. table of contents
  - External anchors: reference to external document

28

HTML Short Reference

- Anything in the document that is between chevrons (< and >) is considered to be an HTML tag
  - The robust parsing of most browsers will just ignore tags that are invalid or improperly formed.
- Comments
  - Between **<!--** **and** **-->**
- Tags and Attributes
  - **<tag attribute="value">content</tag>**
  - The tag tells the browser what kind of mark-up is being introduced, and the attributes can be used to change the exact rendering of the mark-up.
- Minimized Tags
  - A minimized tag is one where there is no start and end tag, just a single instance of the tag
  - **<tag>** or **<tag/>**
    - the second variation is the one preferred by the XHTML standard

29

Nesting and Overlapping

- Nesting means including tags inside other tags
  - **<tag_one>Content <tag_two>Other Content</tag_two> More Content</tag_one>**
- Overlapping means extending the inner tag past the outer tag
  - **<tag_one>Content <tag_two>Other Content</tag_one> More Content</tag_two>**
- Many tags can be nested, provided that the sense of the document is retained
  - one can specify italic text inside bold text in order to have bold, italic text
  - one can also nest containers such as tables inside other containers **(inside cells of tables, in the case of tables)**
- **<B>Bold <I>*Bold + Italics*</B>** *Italics only?***</I>**

30

Text Formatting

- The first three tags are the easiest. They typically have no attributes, and are used on their own, or nested, in order to render specific effects
  - The bold text tag is <B>
    - **This will be <B>in bold</B>**
  - The Italic text tag is <I>
    - **This will be <I>in italics</I>**
  - The underline text tag is <U>
    - **This will be <U>underlined</U>**
- You can use the FONT tag. However it has been deprecated by the addition of style sheets.
  - attributes that allow you to specify the font by the FACE attribute, color, and size of the text that you want to be displayed.

31

B, I, and U Example with nesting

```
<HTML>
<HEAD>
<TITLE>Bold, Italics and Underlined</TITLE>
</HEAD>
<BODY>
This is <B>bold text</B>.<BR /><BR />
This is <I>italicized text</I>.<BR /><BR />
This is <U>underlined text</U>.<BR /><BR />
This is <B>bold +<I>italicized
+<U>underlined</U></I></B>.
</BODY>
</HTML>
```

32

Font Example

```
<HTML>
<HEAD>
<TITLE>Times, Courier and Verdana Text</TITLE>
</HEAD>
<BODY>
<FONT FACE="Courier" SIZE="6">This is Courier text</FONT><BR /><BR />
<FONT FACE="Times" Color="#808080">This is Times text</FONT><BR /><BR />
<FONT FACE="Verdana" Color="Red">This is Verdana text</FONT><BR /><BR />
</BODY>
</HTML>
```

- The standard HTML colors are: Black, Silver, Gray, White, Maroon, Purple, Fuchsia, Red, Yellow, Olive, Green, Lime, Navy, Teal, Blue, Aqua
- One can specify colors using RGB hex value
  - You can use a tool called the Color Schemer, available at http://www.colorschemer.com/online.html

33

Pre – Formatting and emphasizing

- Pre formatting tag <PRE>

```
<PRE>
    This text
        is preformatted.
</PRE>
```

- Emphasizing tags <EM>
  - On most browsers this is same as italic
- and <STRONG>
  - On most browsers this is the same as bold

This text is <EM>emphasized</EM>
This text is <STRONG>strongly emphasized</STRONG>.

- Other tages

<CODE>—Computer code excerpts
<SAMP>—Sample output from programs
<KBD>—Text to be entered by users

34

The <abbr> tag (abbreviation)

- <abbr title="Hyper Text Markup Language">HTML</abbr>
- The HTML word will be viewed
- When the mouse hover over the HTML word a popup note will be shown viewing the content of the title attribute.

35

Flow Formatting

- Anything that changes the flow of the content around the page, as well as (optionally) specific color and style information
- The heading tag, heading 1 as the most important, and 6 as the least

<h1>Heading 1</h1>
<h2>Heading 2</h2>
<h3>Heading 3</h3>
<h4>Heading 4</h4>
<h5>Heading 5</h5>
<h6>Heading 6</h6>
This is normal text.

- Heading can take several tags such as ALINE
  - **<h1 align="RIGHT">Right Aligned Heading</h1>**

36

Paragraph <P>

```
<body>
<h1 align="LEFT">Heading</h1>
<p>This is a
paragraph of text.</p>
This is not inside a paragraph.
<p>This is another paragraph.</p>
</body>
```

- The ALIGN attribute with the paragraph tag.
  - This is deprecated
  - You can use the DIV align attribute

```
<div align="CENTER">
<p>This is centered.</p>
</div>
<p>This is not.</p>
```

- or to use three control tags:
  - **<CENTER>**
  - **<RIGHT>**
  - **<LEFT>**
  - However, only the CENTER tag is supported in the HTML standard and is part of the Transitional definition. In other words, under strict HTML 4.01, it should not be used.
- The <br/> tag force a break line in the document

37

lists of items

- Ordered list: numbered items (nominally 1, 2, 3, and so on, but can be changed)
  - Enclosing it in **
** and **
** tags
  - Each item must begin with an **<li>** tag; however, a closing tag is not required.
  - The TYPE attribute can be used to change the numbering scheme used, however it is deprecated.
    - ‘‘i’’ for roman numerals (i, ii, iii, iv, and so on)
    - “a” for lowercase letters (a, b, c, d, and so on).
  - To start indexing other than the default use the attribute start
    - <OL start=“5”>

```
<ol TYPE=“i">
<li>This is the first item</li>
<ol TYPE="a">
<li>This is sub-item one</li>
<li>This is sub-item two</li>
<li>etc. </li>
</ol>
</ol>
```

38

Unordered list <UL>

- There is simply a bullet or other illustrative element in front of the item rather than a number

```
<ul>
<li>This is the first item</li>
<ul>
<li>This is sub-item one</li>
<li>This is sub-item two</li>
<li>etc. </li>
</ul>
</ul>
```

- The **TYPE** attribute is used to change the ordering style for the ordered list
  - Three allowable values **DISC, SQUARE, and CIRCLE**

39

<UL> Types

```
<ul type="DISC">
<li>This is the first item
<ul type="SQUARE">
<li>This is sub-item one</li>
<li>This is sub-item two</li>
<li type="CIRCLE">This is a different sub-item bullet! </li>
</ul>
</ul>
```

40

Definition list

- A definition list is just a structured list that contains items and their definitions.
- The definition block is usually slightly indented from the item to which it refers, and when the line wraps, the indent is kept.
- You cannot change the aspect of it using the TYPE attribute. Instead, styles must be used to change the rendering.

```
<dl>
<dt>First item</dt>
<dd>First definition</dd>
<dt>Second item</dt>
<dd>Second definition, first part</dd>
<dd>Second definition, second part</dd>
</dl>
```

41

The Table <TABLE>

- consists of a start tag **<table>** and end tag **</table>**. Between these, it is build up from a series of rows, which have a start tag **<tr>** and an end tag **</tr>**.
- Inside table rows, there are a number of columns, or cells, each with a start tag **<td>** and end tag **</td>**.

```
<table BORDER="1">
<tr>
<td>Row 1, Cell 1</td>
<td>Row 1, Cell 2</td>
</tr>
<tr>
<td>Row 2, Cell 1</td>
<td>Row 2, Cell 2</td>
</tr>
</table>
```

- The BORDER attribute can be "0" to force the border to be invisible).

42

Tables cont.

- The WIDTH attribute: controls the table width
  - value for the WIDTH attribute can be presented as a percentage of the available block or as a discrete value, in pixels.

Ex: A table that has three columns, two of 25% and one of 50%, you would use HTML such as:

```
<table WIDTH="100%">
<tr>
<td WIDTH="25%">
</td>
</tr>
<tr>
<td WIDTH="50%">
</td>
</tr>
<tr>
<td WIDTH="25%">
</td>
</tr>
</table>
```

Note: if you want to center the table, you can enclose it in **<CENTER>** and **</CENTER>** tags.

43

Homework

- Use the <table> tage to to generate the webpage shown here. Note that you may need to use attributes like COLSPAN, WIDTH, and VALIGN, also other tags like <H1>, <BR/>, …

.

44

More on Tables

- BGCOLOR attribute is used to set the background of the table or a particular cell inside the table. deprecated
- The CELLSPACING and CELLPADDING attributes are used in the <TABLE> tag to set the space between cells (spacing) and the inside margins (padding).
- The **<TH>** tag allows you to specify that a cell contains heading information

```
<table>
<tr>
<th>Col 1</th>
<th>Col 2</th>
<th>Col 3</th>
</tr>
<tr>
<td>Cell 1</td>
<td>Cell 2</td>
<td>Cell 3</td>
</tr>
</table>
```

45

Document Linking

- The generic form for an external link (which will be in the index page, linking to other pages) is as follows:
- <A> is the anchor tag
  - **On the same page**
    - **<A HREF="#internal_ref_name>Link Text</A>**
  - **On the same server**
    - **<A HREF="target_page.html">Link Text</A>**
  - **On different server**
    - **<A HREF="http://www.mycom.com/target_folder/target_page.html">Link Text</A>**
    - **<A HREF="http://www.mycom.com/target_folder/target_page.html #internal_ref_name"> Link Text</A>**
- Links can be text, images, or multimedia.
- Text links typically appear as underlined, differently colored words on a page.
- Any image on a web page can be turned into a link.
- You can place <a> tags around video and audio content to make them links.

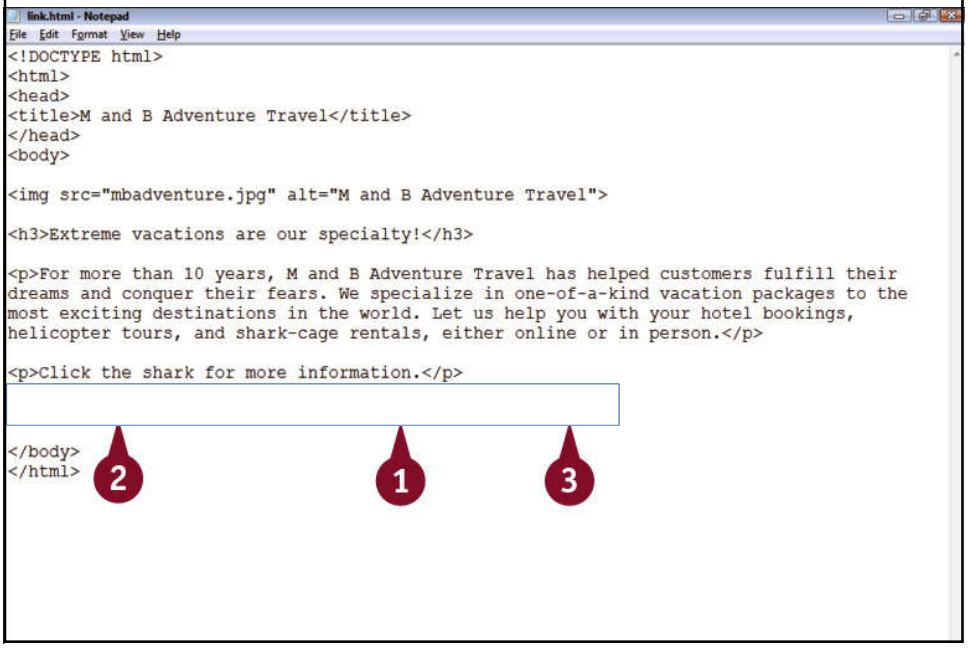
46

Absolute and Relative Links

- *Absolute links* use a complete web address, or *URL*, to point to a specific page on a specific web server:
  - Click here
- *Relative links* use shorthand to reference a page and do not specify the server. You generally use relative links to reference documents on the same website:
  - Click here
- You must include the prefix (http://) when referencing URLs in your HTML.
- If you are linking to a document that resides on a file transfer site, you use the FTP prefix (ftp://).
- If you want to create a link that opens an e-mail program, allowing a user to send an e-mail message, you use the MAILTO prefix (mailto:).
- There is also an encrypted version of the HTTP prefix, https://, which you use when linking to secure areas of websites, such as those involving payment transactions.

47

Insert an Image Link



48

Open a New Window with a Link

- You use a target attribute within the <a> link tag to open links in new windows. To make all the links on your page open in new windows, you can use the <base> tag.
- To make multiple links on a page open in the same window, use the same target value for the links, for example target= “new1”.
  - The first time a user clicks one of the links, a “new1” window opens. Subsequent link clicks open pages in the same window.

49

```
</head>
<body>

<img src="mbadventure.jpg" alt="M and B Adventure Travel">

<h3>Extreme vacations are our specialty!</h3>

<p>For more than 10 years, M and B 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 your hotel bookings,
helicopter tours, and shark-cage rentals, either online or in person.</p>

<p>Other online travel resources:</p>

<ul>
<li><a href="http://www.wiley.com"                    ">Wiley Publishing</a>: Find travel
guides and other books.</li>
<li><a href="http://www.wikipedia.org">W    ipedia</a>: Research travel destinations.</li>
<li><a href="http://www.google.com">Goo    /a>: Doscover other travel Web sites.</li>
</ul>

</body>
</html>
```

1

link-allnew.html - Notepad

File Edit Format View Help

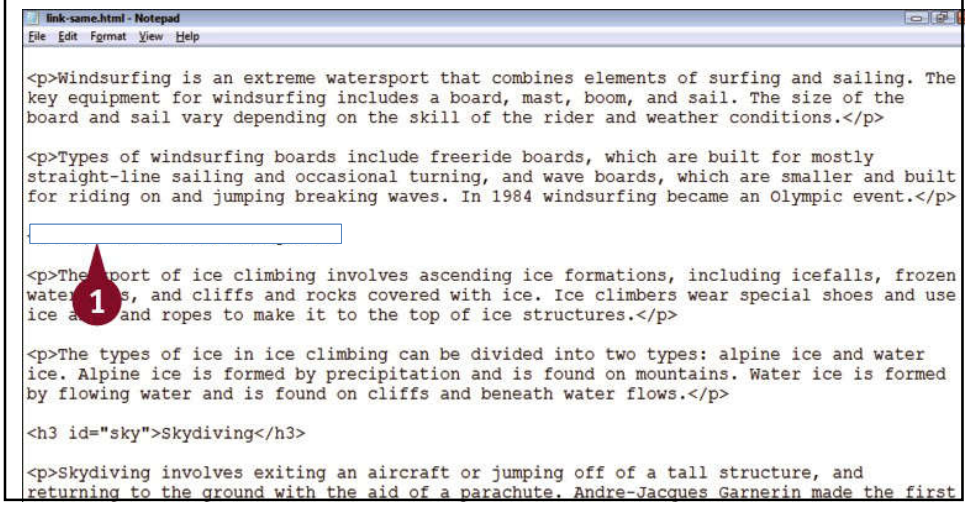
```
<!DOCTYPE html>
<html>
<head>
<title>M and B Adventure T    l</title>

</head>
<body>

<img src="mbadventure.jpg" alt="M and B Adventure Travel">

<h3>Extreme vacations are our specialty!</h3>

<p>For more than 10 years, M and B Adventure Travel has helped customers fulfill
dreams and conquer their fears. We specialize in one-of-a-kind vacation packages
```

1

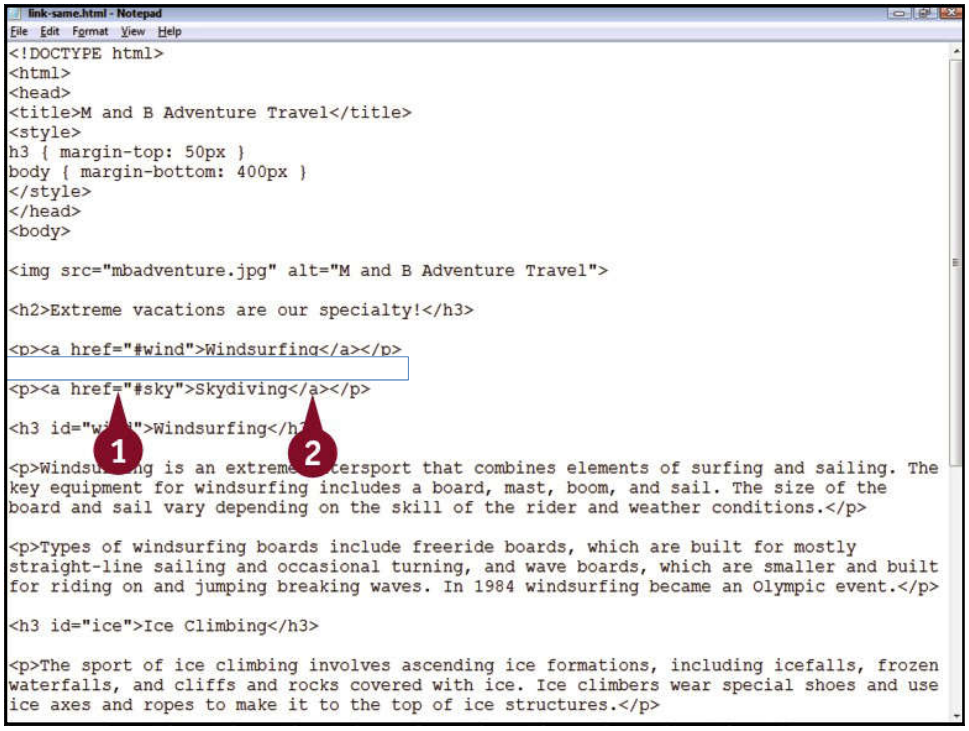
50

Link to an Area on the Same Page

- You must assign names to the areas to which you want to link
  - You can do this with the *id* attribute.



51

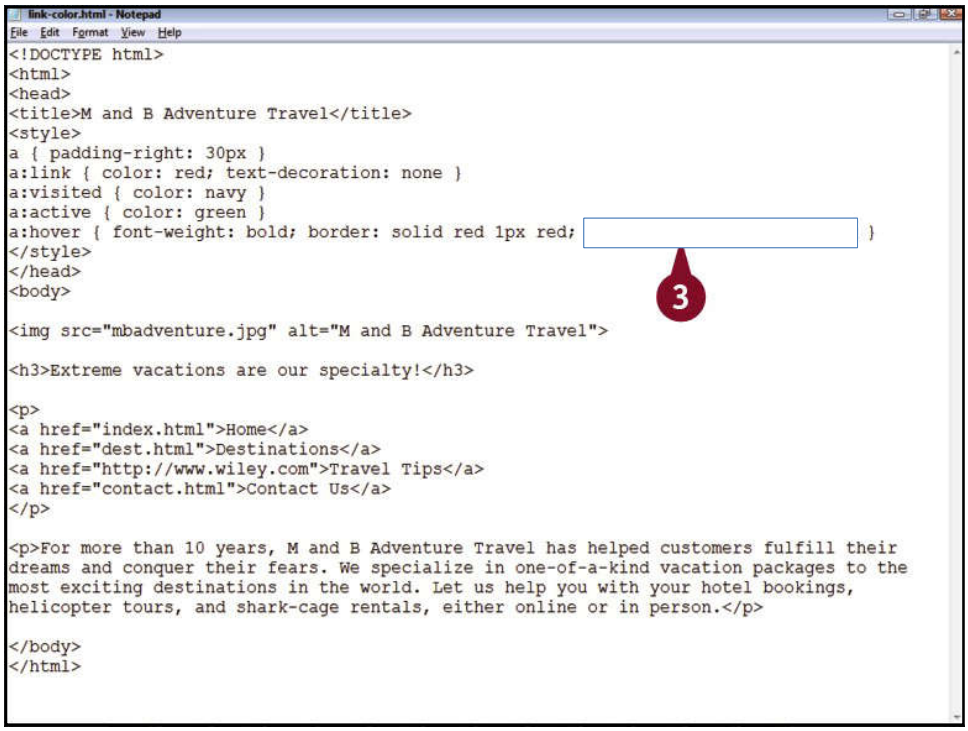


52

Change Link Colors

- You can control the appearance of links throughout your web pages using a style rule.
- You can change the color of unvisited, visited, and active links to make them match the theme of your website.
- You can also remove the default underlining that normally appears beneath a link using the *text-decoration* property.

53

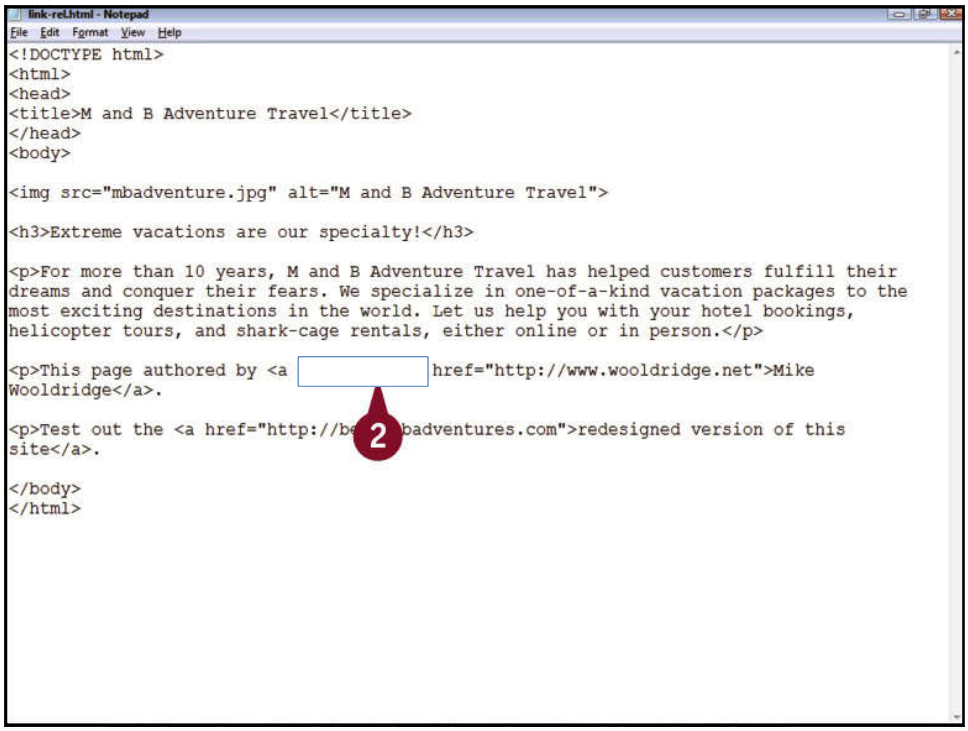


54

Define Link Relationships

- The rel attribute can mark the link as belonging to the previous or next document in a series, to the website of the author of the current document, or to a copyright license for the current document.
- You can also use the rel attributes to tell search engines to ignore a link when determining the ranking of the current page. Search engines often analyze the outgoing links on pages to help categorize the pages the links are on.

55



56

The values of the rel attributes in <a> tags

- The following are values that can be used with the rel attributes in <a> tags:
- Alternate: Link to an alternate representation of the current document
- Author: Link to the author of the current document
- External: Link to a document at a site that is external to the site of the current document
- first, prev, next, last: Links to other documents in a series of documents
- Help: Link to context-sensitive help
- License: Link to a copyright license for the current document
- Nofollow: Specifies that search engines should not consider the linked document for page-ranking purposes
- Noreferrer: Specifies that the browser should not send referrer information when the link is clicked
- Search: Link to a search page for the current document

57

Forms

- Input elements such as text fields, menus, and check boxes **collect data from a website user**.
- Web page forms have **three important par**ts: a <form> tag, form input elements, and a Submit button.
- All forms should include a Submit button for sending the data to a web server for processing.
- **After** the form data is **processed**, a **script typically sends a confirmation page back** to the browser window noting whether or not the form data was sent successfully.

58

Form tag <FORM>

- Typically, a form has **two attributes** filled out
  - ACTION: the location of the script that will process the form contents.
  - METHOD: indicates the method that will be used in sever side processing (GET or POST)
- The **<INPUT>** tag is used to **specify the elements** that are to be used to allow the users to enter data
  - accompanied by the NAME and TYPE attributes.
  - The **NAME** gives the name of the field that will be **used by the script on the server (or client)** to retrieve the data associated with the field.
  - The **TYPE** attribute can be one of a variety of different data entry types, of which
    - the most common are likely to be: TEXT, RADIO, CHECKBOX, and SUBMIT

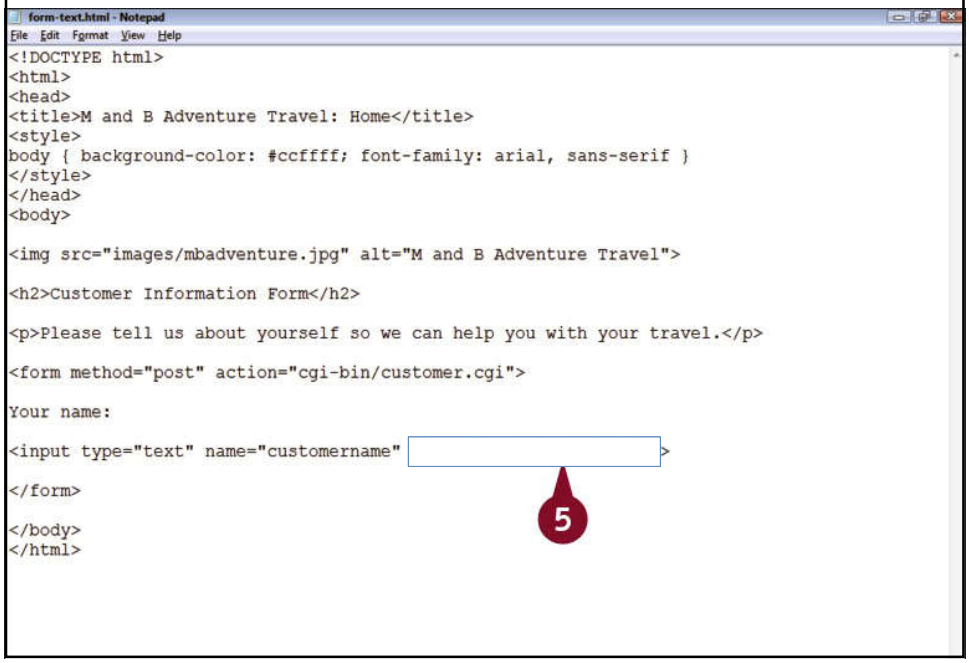
59

Control Types

- **Text Boxes**
  - **New** in HTML5, some text boxes can hold a specific type of text content such as an **e-mail address** or a **URL**.
- **Radio Buttons**
- **Check Boxes**
- **Menu**
  - New in HTML5, you can define **menus that hold time and date** information such as months, weeks, and days.
- **File Upload**
- **Range Slider is new** in HTML5
- **Date and Time Fields**
  - HTML5 introduces several new field types for presenting chronological information such as month, week, and time fields.
- **Submit Button**
- **Reset Button**

60

Textbox



61

Tips, Default value, and the password textbox

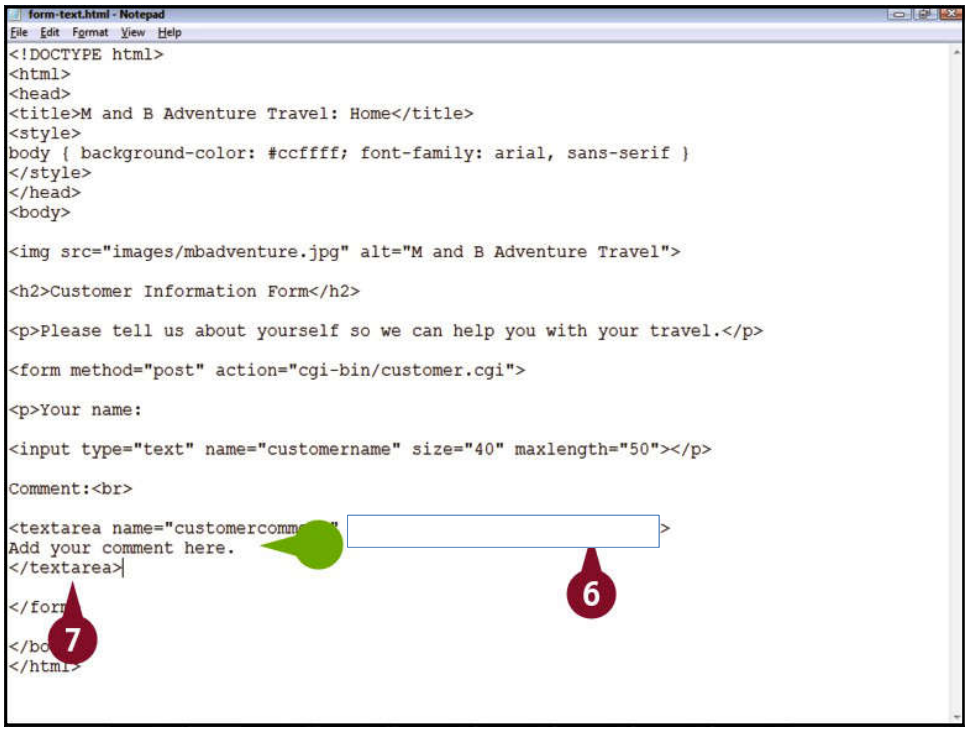
- To specify a **default**, you can add the **value attribute** to the <input> tag. For example:
  - <form method="post" action="/cgi-bin/feedback.pl">
  - <input type="text" name="fullname" value="Enter your first and last names">
  - </form>
- **Password text boxes** are similar to regular text boxes with one difference. Rather than displaying the characters that are typed, the input field displays the data as asterisks (*) or bullets (•). This prevents others from seeing the password text. To create a text box for password entry, you specify the password type in the <input> tag. For example:
  - <input type="password" name="textboxName" size="45">
- To create a text field that **accepts only numbers**
  - <input type="number" name="age" value="21" min="1" max="120" step="0.5">
- You can define a form input field of type hidden. For example:
  - <input type="hidden" name="specialcode" value="abc123">

62

Add a Large Text Area

- **Text area size** is measured in rows and columns, with the measurement based on the number of characters that can be displayed.
- **Wrap attribute**.
  - Use soft to wrap text within the text area but does not wrap text in the form results (text come after data processing).
  - Use hard to wraps text within both the text area and the form results.
  - Use off to turn off text wrapping, forcing users to create new lines of text as they type.
- If the user types more text than is visible in the text area, scroll bars appear at the side of the text box
- You can use the readonly attribute if you want to display default text in a text area and do not want users to move or edit the text.
  - This attribute does not require a value.

63



64

Add Check Boxes

- Note: The check box value does not appear on the form, it is used as the value to be assigned if the check box is checked.
- To automatically show the check box as selected use the checked attribute
  <input type="checkbox" Name="newsletter" Value="yes" checked>

form-checkboxes.html - Notepad

File Edit Format View Help

```
<h2>Customer Information Form</h2>

<p>Please tell us about yourself so we can help you with your travel.</p>

<form method="post" action="cgi-bin/customer.cgi">

<p>Your name:

<input type="text" name="customername" size="40" maxlength="50"></p>

<p>Comment:<br>

<textarea name="customercomment" rows="5" cols="60" wrap="hard">
Add your comment here.
</textarea></p>

Service:

</form>
```

4

5

65

Add Radio Buttons

- When radio buttons have different name attributes, the browser treats them as parts of different radio button sets.
- You can use the checked attribute to show one radio button in the group as selected by default.

form-radios.html - Notepad

File Edit Format View Help

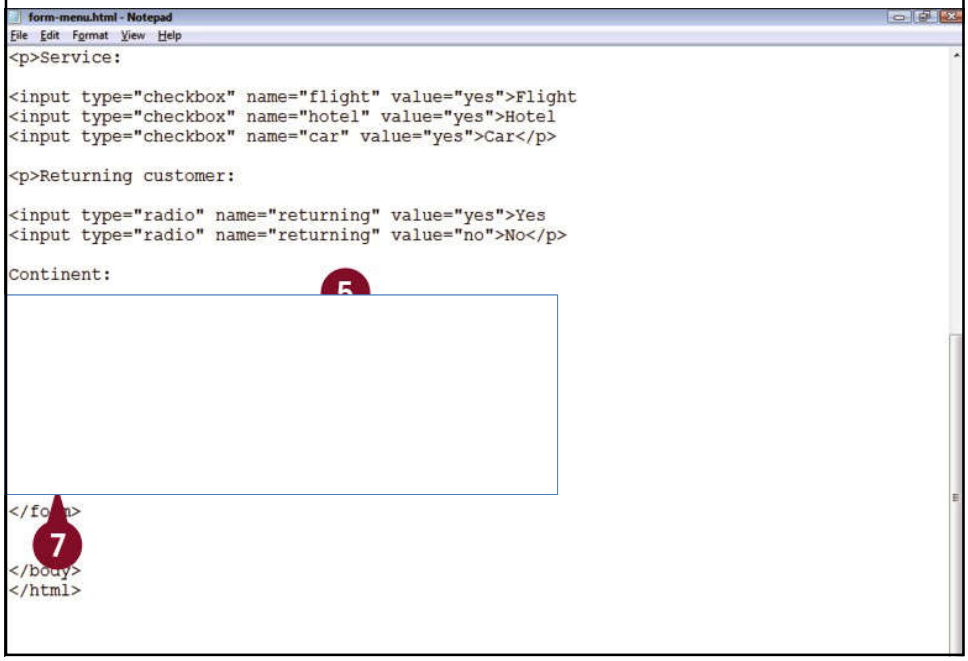
```
<p>Your name:

<input type="text" name="customername" size="40" maxlength="50"></p>

<p>Comment:<br>

<textarea name="customercomment" rows="5" cols="60" wrap="hard">
Add your comment here.
</textarea></p>

<p>Service:

<input type="checkbox" name="flight" value="yes">Flight
<input type="checkbox" name="hotel" value="yes">Hotel
<input type="checkbox" name="car" value="yes">Car</p>

Returning customer:

</form>


</body>
</html>
```

4

5

66

Add a (combo) List



67

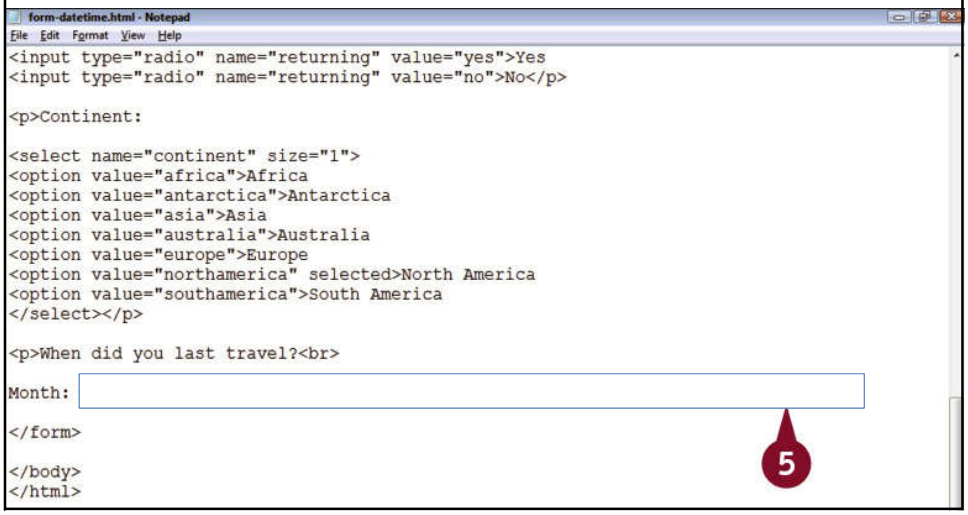
Combo/list

- If the value of the size attribute is greater than 1 the view will be a list (rectangular box) with scroll bars.
- To create a submenu, use the <optgroup> tag and the label attribute (note that not all browsers support the <optgroup> tag)
  - What is your favorite sport?
  - <select name="favoritesport">
  - <optgroup label="Summer">
  - <option value="Diving">Diving
  - <option value="Biking">Biking
  - </optgroup>
  - <optgroup label="Winter">
  - <option value="Skiing">Skiing
  - <option value="Ice Climbing">Ice Climbing
  - </optgroup>
  - </select>

68

Add a Date and Time Input

- For a month type, step values are in months.
- For a week type, step values are in weeks.
- For a time type, step values are in seconds.
  - The value 3600 would create increments one hour apart.



69

E-mail, URL, TEL and autocomplet

- To add an email text field
  - <input type="email" name="emailaddress" size="25" maxlength="256">
- To create an input field for a telephone number
  - <input type="tel" name="phone">
- To prevent a text box autocompleting from happening, you can set the autocomplete attribute to off. For example:
  - <input type="text" name="ssn" autocomplete="off">
- To add a URL text
  - <input type="url" name="emailaddress" size="25" maxlength="200">
  - checks if the text includes "://" with alphabetical text before and alphanumeric text or a period after.

70

Add a Range Slider

- <input type="range" min="100" max="5000" step="100" value="1000">
- To display the current value of the slider next to the slider as text do the following
  - <input type="range" min="100" max="5000" step="100" value="1000" onchange="document.getElementById('display').innerHTML=this.value">
  - 1000

71

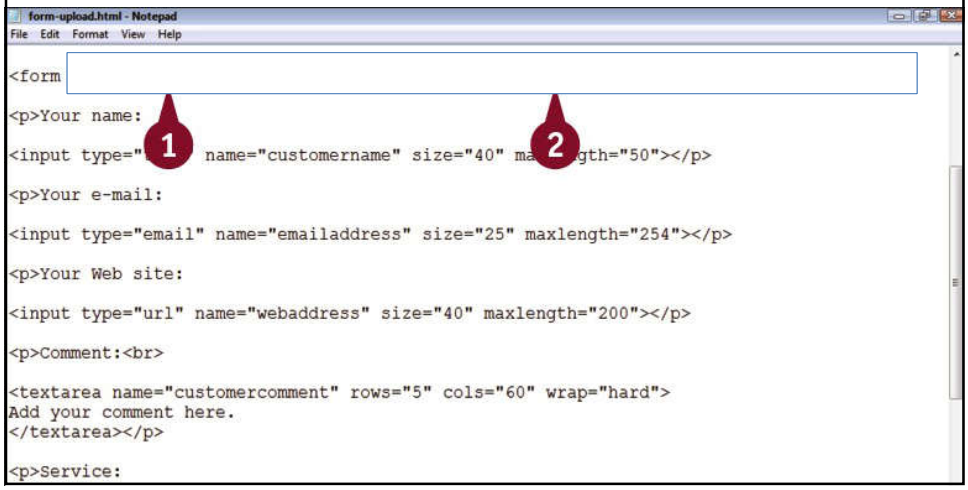
File upload – cont.

- To limit the accepted file type, you can use the accept attribute to list the files your server can process.
- You list the files by their MIME (Multipurpose Internet Mail Extensions) types. Your HTML code may look like this:
- <input type="file" name="userfiles" accept="image/gif, image/jpeg, image/png">

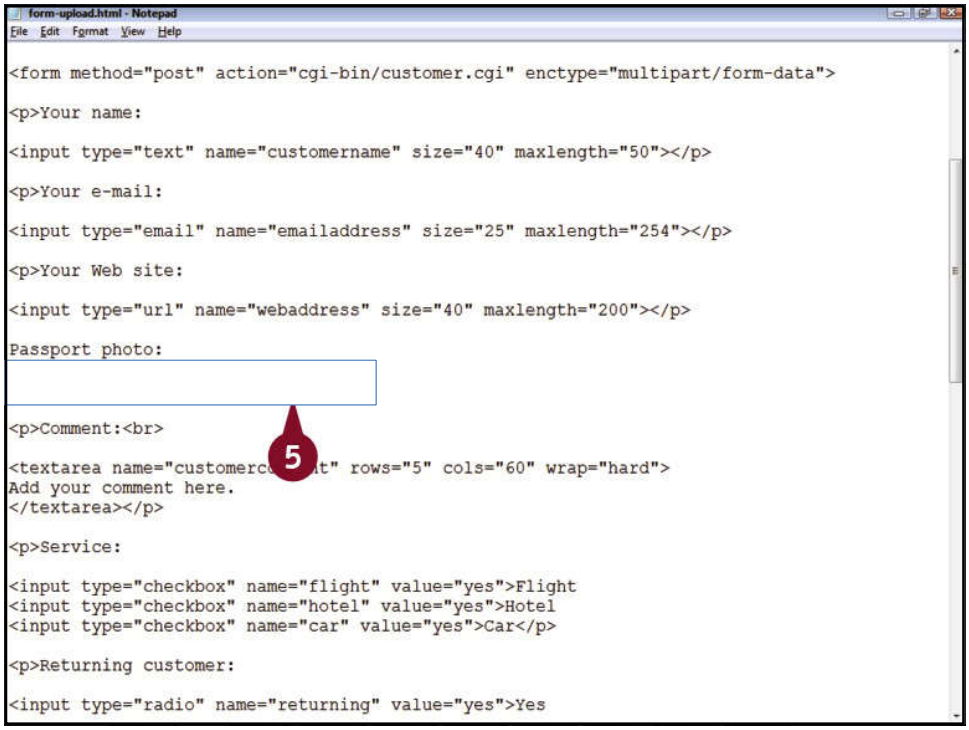
72

Add a File Upload

- To allow users to upload files to the server
- Works only with the post method in the <form> tag



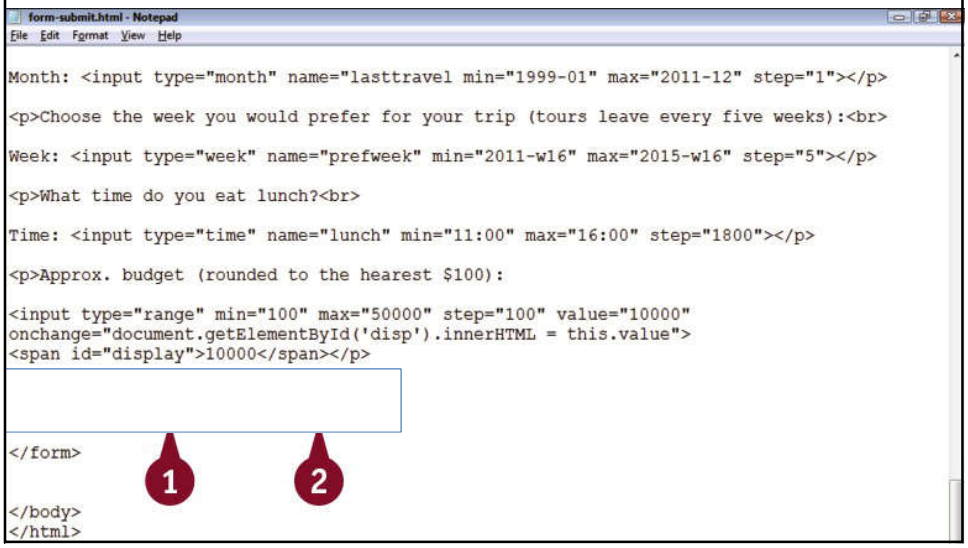
73



74

Submit and reset button

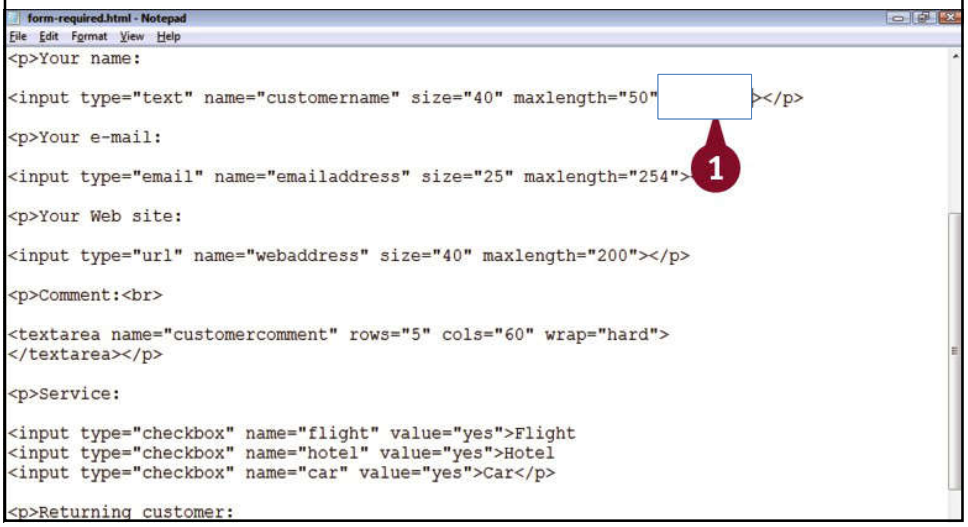
- The submit button allows users to send the data they enter into the form.
- The reset button allows users to clear the data they have entered.



75

Require a Field

- You can mark fields in your form as required using the required attribute.
- If you set a field to required, your users cannot submit the form until they have entered valid data in that field.
- The required attribute does not require a value associated with it.



76

Add a Placeholder

- You can add placeholder text to your form fields to add instructions.
- The text appears inside the form field when the form first appears. If a user clicks inside the field, the placeholder text disappears and the user can type in the field as usual.

77

Validate Input with a Pattern

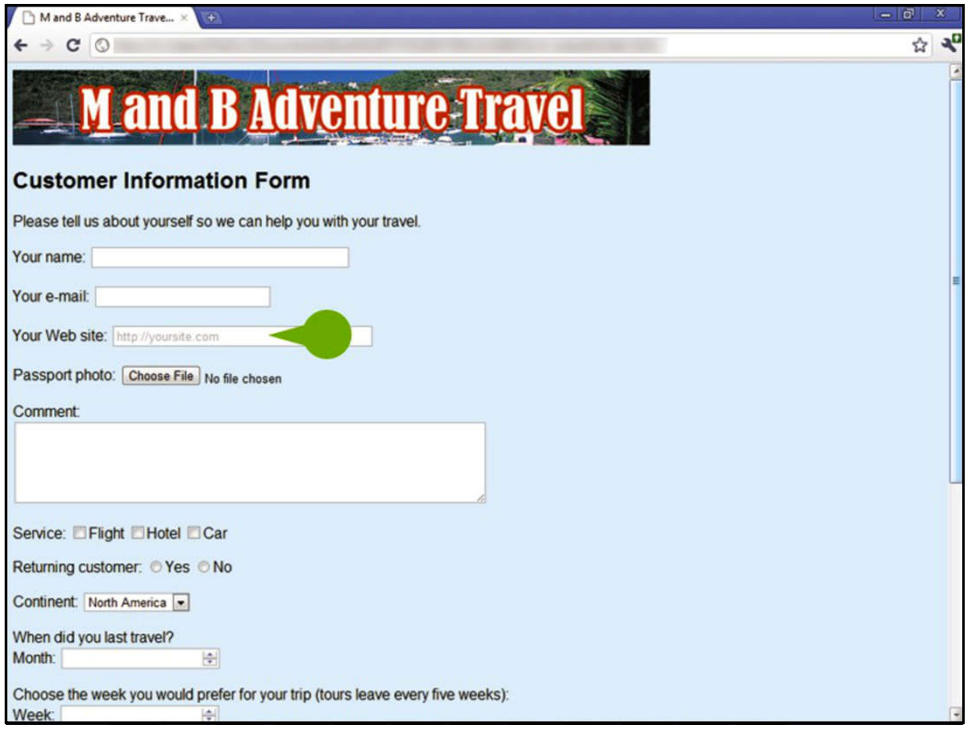
- You can add a pattern attribute to the <input> tag to perform additional validation.
- You create a pattern using a language known as *regular expressions*.
- The browser will check to see if the data in the field matches the defined pattern
- If there is no match, the browser aborts the form submission and displays an alert.
- For example, to Accept only Letters and Numbers: pattern=”[A-z0-9]*”

78

Some regular expressions

- . Any character, including letters, numbers, symbols, and whitespace.
- [a-z] Any lowercase letter.
- [A-Z] Any uppercase letter.
- [A-z] Any letter, lowercase or uppercase.
- [0-9] Any number.
- * Any number of instances of the character preceding it, including zero instances.
- + One or more instances of the character preceding it.
- ? One or zero instances of the character preceding it.
- {N} N instances of the character preceding it, where N is a positive number.

79



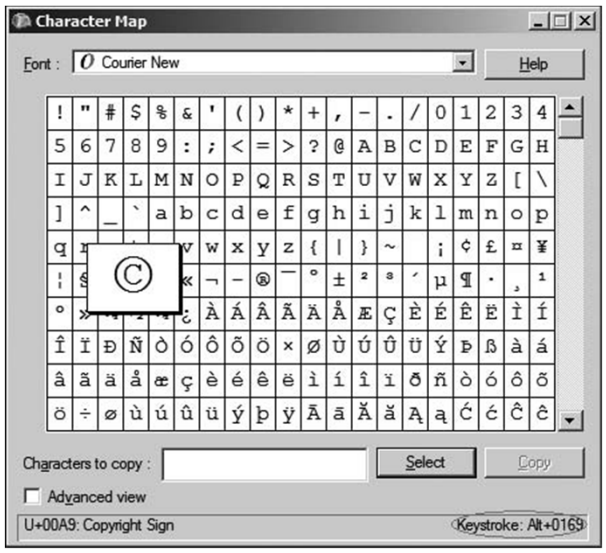
80

Common Symbol Codes

- Some symbols are used for coding in HTML such as <, >, and “.
- If you want these symbols to appear in the web page for the user to see, you need to use the &code instead.
  - < <
  - " “
  - ® ®
  - > >
  - © ©
  -   space
  - The complete list can be found on W3C
  - Other symbols can be inserted using the &#code, see the character map in windows
    - Start→Programs→Accessories→System Tools→Character Map

81

Character map



82

Drawing on Canvases

- A new <canvas> tag enables you to define a rectangular area on your page on which you can draw.
- You can use scripting commands to create shapes, draw straight and curved lines, apply color gradients, and even add images or parts of images within the area
- In the past, designers needed to create such visual content in a separate image editor or drawing program and then embed the result as an image.
- We will revisit Canvas after learning javascript

83

CSS and Styles

- Allow you to reflect the design you want on the content, but be able to change it on a whim without recoding everything from the ground up.
- Attach specific information pertaining to the rendering of HTML while remaining within XML and HTML guidelines.
- Very important when dealing with Content Management Systems (CMS).
- Done by creating named derivations then use these derivations as attributes in tages
- Styles are created using the **"text/css"** language
- You can introduce styles in three places
  - **Inline** (at the same time as the tag, such as <h1 style="color: purple;">. . .)
  - **Internally**, as part of the <head> of the document
  - **Externally**, in style sheets
    - External style sheets are a good idea if you need to be able to swap between styles
- Two flavors of style sheet
  - CSS adds style to existing content
  - XSL provides the possibility to transform the content into something different— CSS and HTML, XHTML, or XML.

84

Tags and Styles

- **Each object** (element) in the page **has properties**, and you refer to each one through the Document Model.
- Document Model can be seen as a tree, where the document is at the root, and each block element (** **, **<p>**, heading, table, list, and so on) can be a branch, or leaf, on that tree.
- Each element has properties
- Styles give access to all properties

85

Tags and Styles

- **CSS** documents are simple text files.
- Each tag in an HTML document can be altered with the application of a style
  - With CSS, you can specify that your heading be bright red, 36 pixels tall, and aligned to the right side of the page.
  - For example, you can change the meaning of the bold tag to use a different color or some other text decoration.
  - If you define the bold tag as being green and bold, that will be applied each time you use the **<b>** tag.
- The W3C actually defines a style sheet for XHTML 1.0, which gives the default style information for the tags that are defined in that standard.

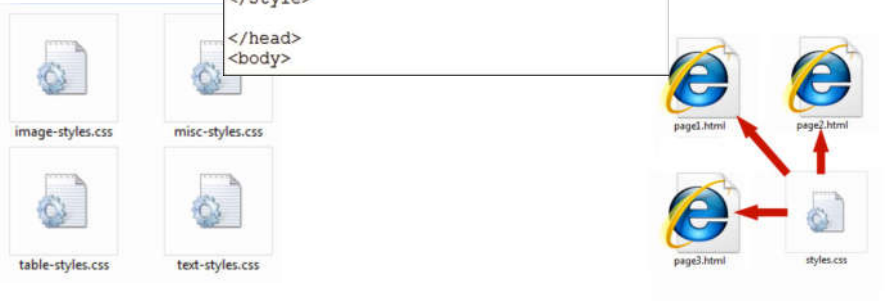
86

CSS

Style sheets are made up of rules, and each rule has two distinct parts: a selector and a declaration.

```
<!DOCTYPE html>
<html>
<head>
<title>M and B Adventure Travel</title>

<style>
body { color: black;
       background: white }
h2   { color: red }
p    { font-size: 10px;
       text-align: center }
</style>

</head>
<body>
```

The selector is h2 and {color: red} is the declaration



87

Internal CSS

internal.html - Notepad

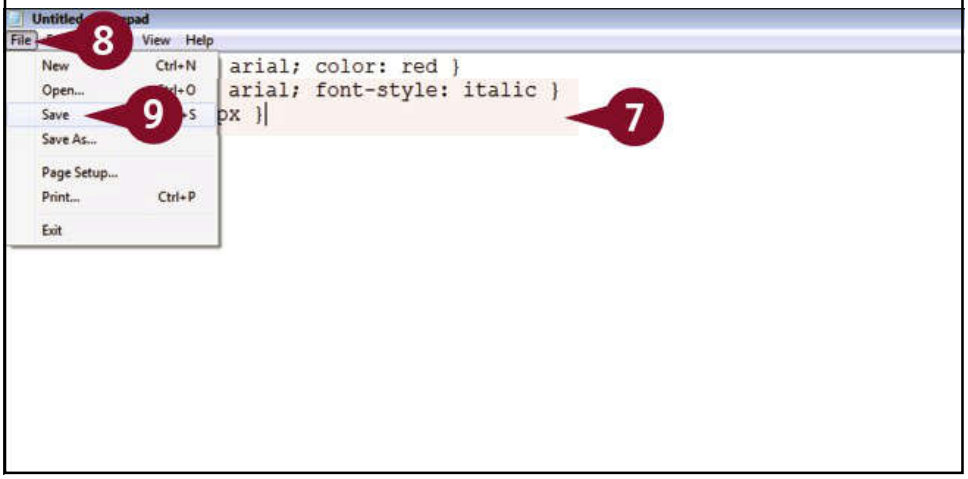
File Edit Format View Help

```
<!DOCTYPE html>
<html>
<head>
<title>M and B Adventure Travel</title>

</head>
<body>

<h1>M and B Adventure Travel</h1>

<h2>Extreme vacations are our specialty</h2>

<p>For more than 10 years, M and B 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 your hotel bookings,
helicopter tours, and shark-cage rentals, either online or in person</p>

</body>
</html>
```

88

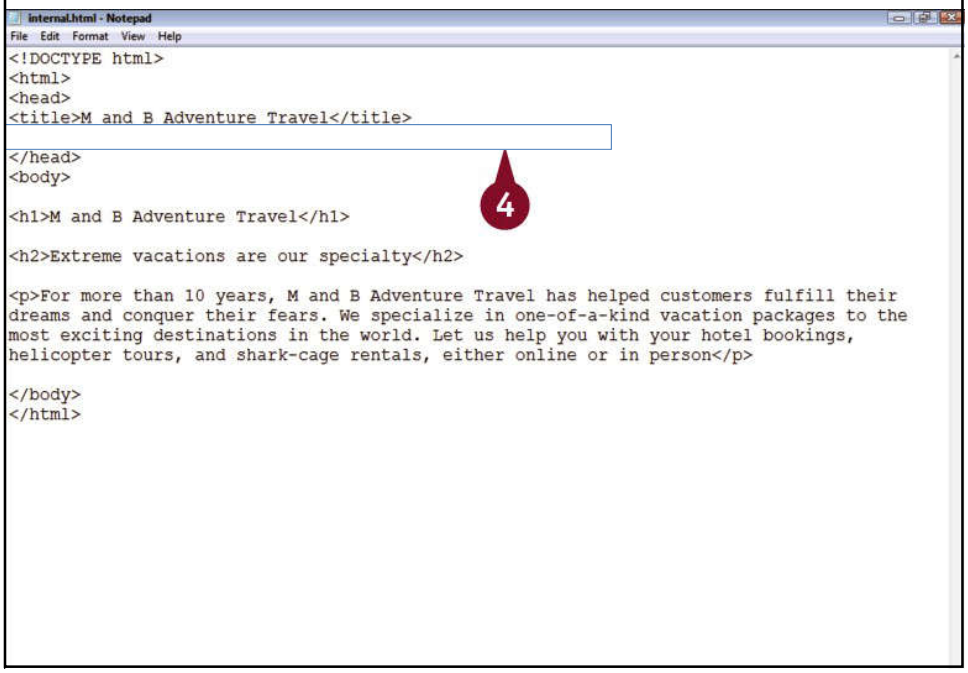
Create an External Style Sheet

- Save into css file e.g. style.css
- Use the <link> tag



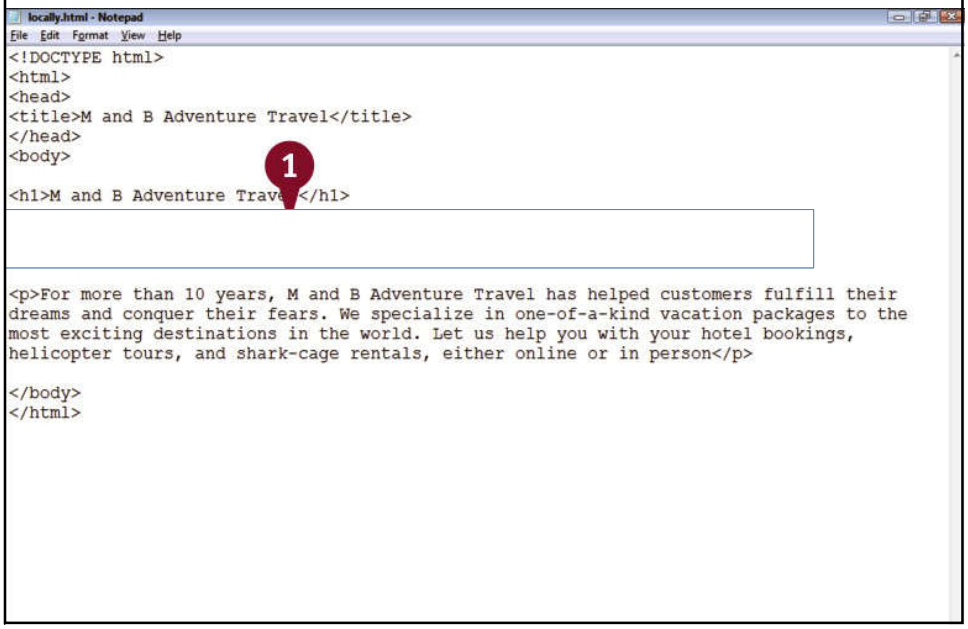
89

External CSS



90

Apply a Style Locally



91

CSS – Applying style to multiple tags or all HTML tags

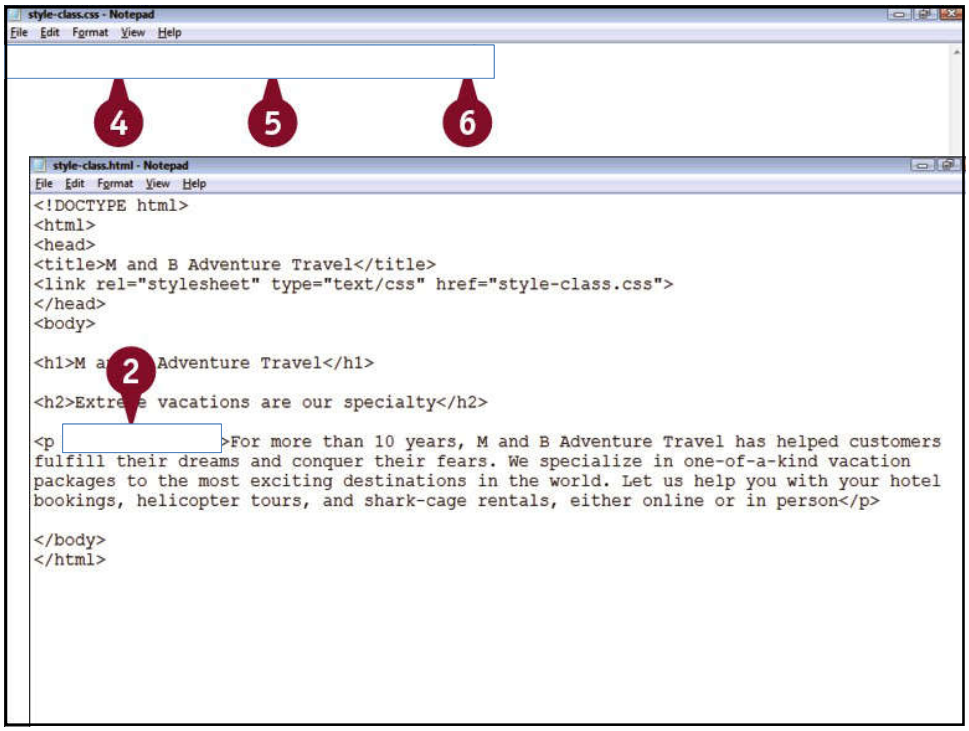
- You can separate a list of selectors by commas to **apply the same set of styles to all** of them:
  - h1, h2, h3 {color: red; font-size: 36px}
  - The style rule above turns all three types of headings on your page red and sizes them to 36 pixels. Writing style rules this way saves you time and makes your code more compact.
- You can use the **universal selector (*)** to apply styles to every HTML tag in your document:
  - * {padding: 0}
  - The style rule above removes the padding from all the elements on your page.
- Note that you can override such a rule by writing additional rules for specific tags.
  - Adding the following to your style sheet applies padding to h3 headings while, due to the previous rule, all other elements still have no padding:
    - h3 {padding: 10px}

92

Assign a Class

- You can create a CSS class to apply a style rule to specific **instances** of HTML tags in a page. For example, if you want the introductory paragraphs formatted differently from all the other paragraphs, you can create a class specifically for the introductory paragraphs. After you create the class and assign it using the "class" attribute, the browser applies the formatting to all the affected paragraphs.
- You can set up a class in an internal or external style sheet.
  - Ex: h1.my_heading { color: "blue"; }
    - **<h1 class="my_heading">H1 In Blue</h1>**

93



94

Generic class

- You can use a generic class to format more than one type of tag. For example, you might use a generic class to format both paragraphs and level 3 headings in a document. When defining a generic class, simply type a period followed by the class name and then your declaration. For example:
  - **.myclass {color: blue}**
- Do not type an HTML tag before the period. The following HTML examples apply the class to different tags:
  - ** some text **
  - **<h3 class="myclass">some other text</h3>**

95

Assign multiple classes to a tag

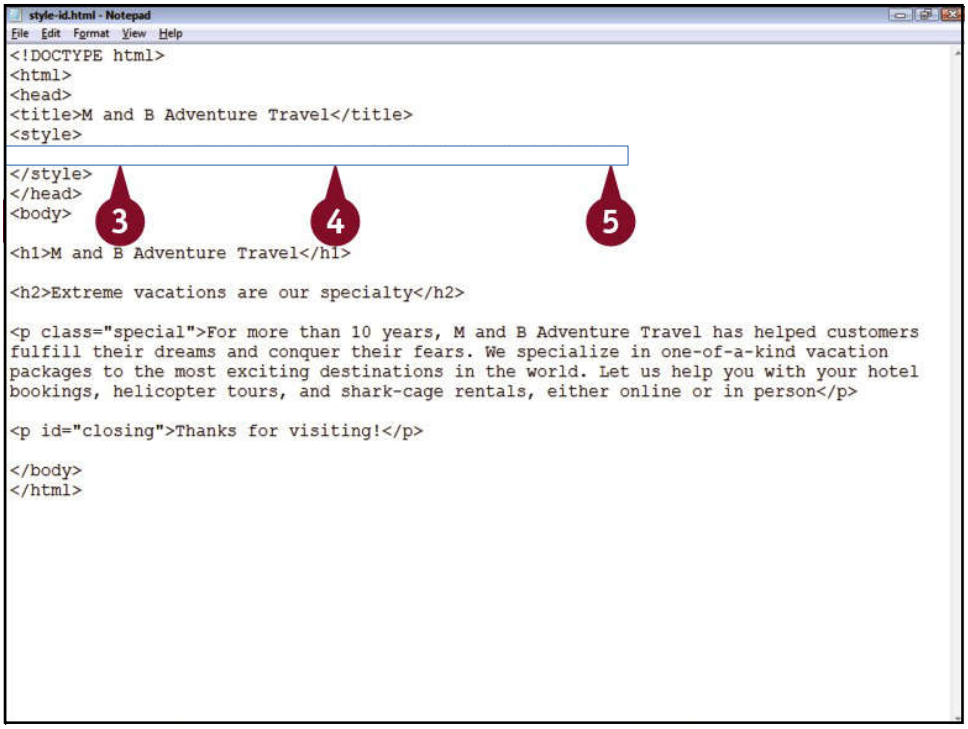
- You can assign multiple classes to a tag to add more than one set of styles to that tag. You separate the class names with spaces. For example:
  - `<p class="huge fancy">This is a standout sentence.</p>`
- The paragraph above would have the styles associated with both the `huge` class and the `fancy` class applied to it.

96

Apply a Style Using an ID

- You can apply an "`id`" attribute to an HTML tag on your page to give it a unique identifier.
- You can then apply styles to that HTML tag using a special CSS selector for that tag.
- Using the "`id`" attribute to apply styles is an alternative to using the "`class`" attribute.
- You can set your ID rules in an internal or external style sheet.

97

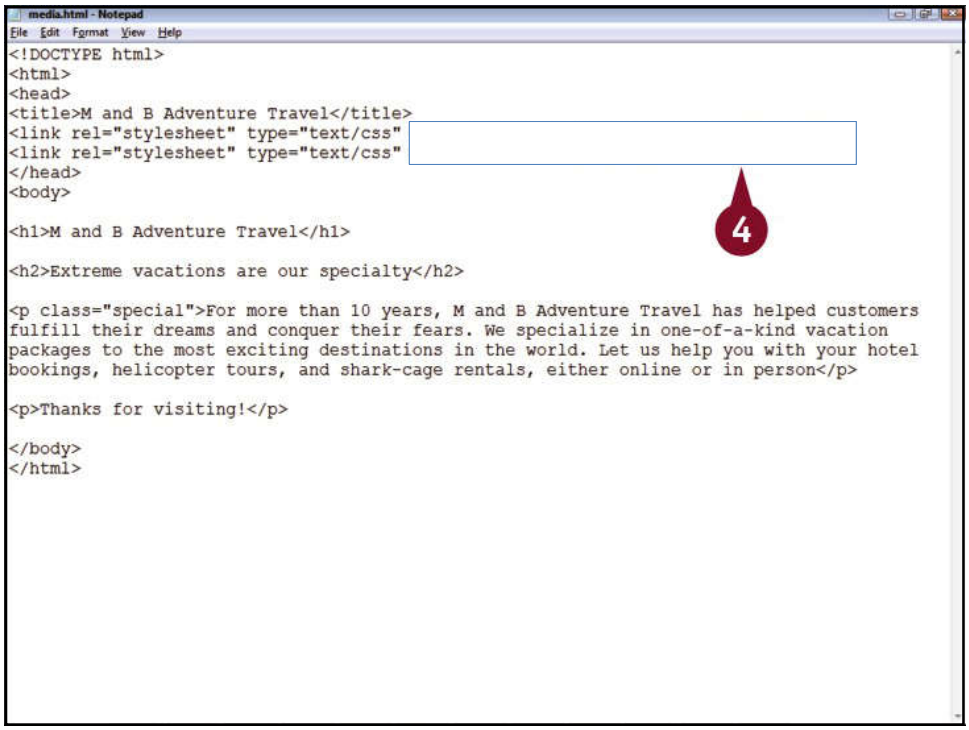


98

Link to Media-specific Style Sheets

- You can link to several style sheets in your HTML document and specify that different styles sheets be applied for different media using the `media` attribute.
- For example, one style sheet could be applied when the document is displayed on a computer screen, another style sheet could be applied when the document is printed, and yet another could be applied for viewing on a mobile phone or other handheld device.

99



100

The different media types available

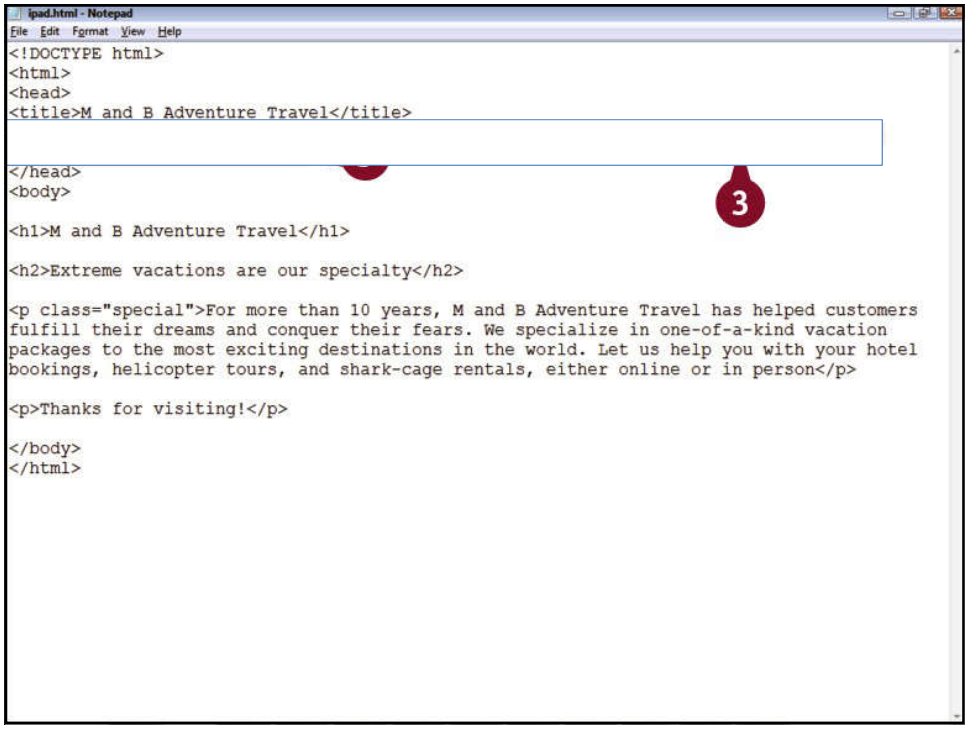
- **all**: For all devices
- **projection**: For projected presentations
- **braille**: For braille tactile feedback devices
- **screen**: For color computer screens
- **embossed**: For paged braille printers
- **speech**: For speech synthesizers
- **handheld**: For handheld devices such as mobile phones
- **tty**: For teletypes, terminals, and other media that use a fixed-pitch character grid
- **print**: For printed pages and for documents viewed in print preview mode
- **tv**: For televisions
- ***Note that some advanced mobile phones such as the iPhone and Android phones as well as the iPad do not respect the handheld media type***

101

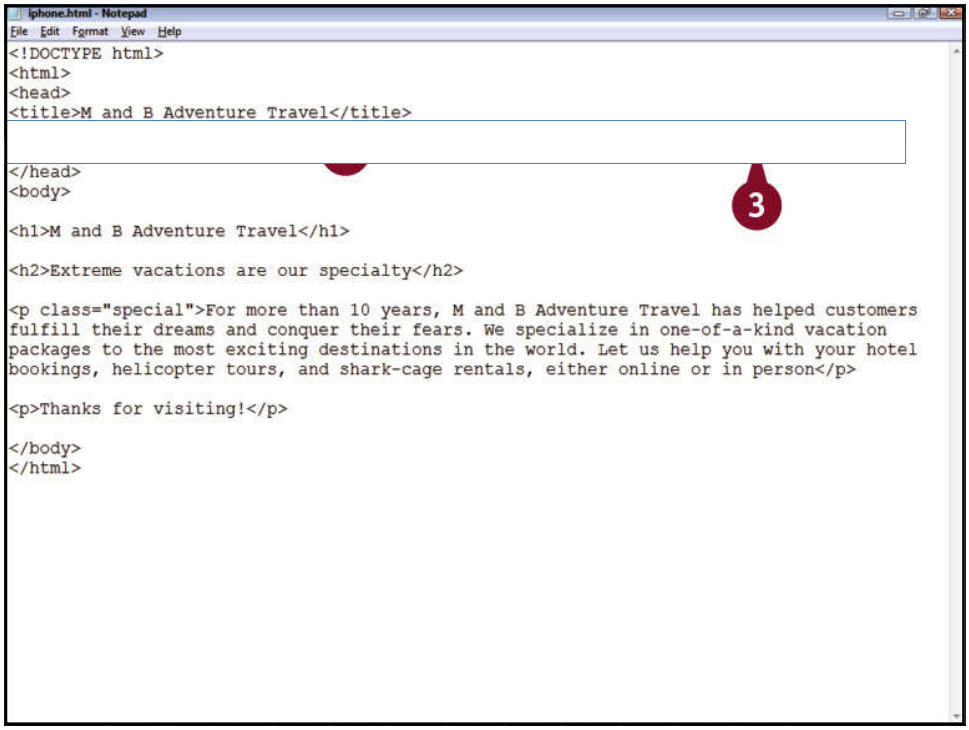
Link to Style Sheets for iPads, iPhones, and Android Phones

- The iPad, iPhone, and Android OS web browser **ignore** `handheld` links and instead behave more like browsers on traditional computers — they load style sheets specific to the `screen` media type.
- However, you can create style sheets specific to browsers on these devices by checking the maximum width of the device screen.
- The current iPad screen has a width of 1024 pixels on its longer side.
- The current iPhone and Android phone screens have a width of 480 pixels on their longer sides.

102


ipad.html - Notepad
File Edit Format View Help
<!DOCTYPE html>
<html>
<head>
<title>M and B Adventure Travel</title>
</head>
<body>
<h1>M and B Adventure Travel</h1>
<h2>Extreme vacations are our specialty</h2>
For more than 10 years, M and B 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 your hotel bookings, helicopter tours, and shark-cage rentals, either online or in person
Thanks for visiting!
</body>
</html>
3

103


iphone.html - Notepad
File Edit Format View Help
<!DOCTYPE html>
<html>
<head>
<title>M and B Adventure Travel</title>
</head>
<body>
<h1>M and B Adventure Travel</h1>
<h2>Extreme vacations are our specialty</h2>
For more than 10 years, M and B 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 your hotel bookings, helicopter tours, and shark-cage rentals, either online or in person
Thanks for visiting!
</body>
</html>
3

104

TIPs

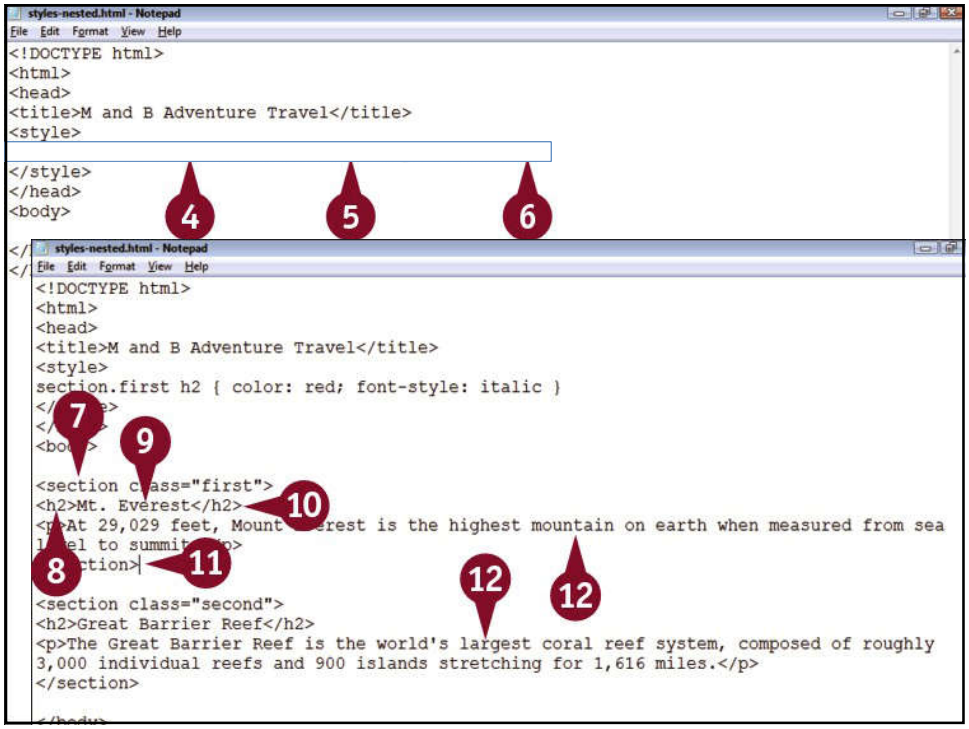
- **What should I keep in mind when optimizing a page for mobile devices such as the iPhone?**
- Key things to consider are that you are designing for a smaller screen**, navigation by touch**, and slower download speeds. You should consider **increasing the font size** to make text more legible and links easier to tap; turning off the display of larger images to reduce the page download time; and disabling CSS float styles to convert multicolumn layouts into single-column layouts, which can be easier to view on smaller screens
- **Is there a way to make a CSS rule specific to a media type?**
- Yes. The steps above describe loading an entire external style sheet based on the media type. To limit a specific CSS rule to a media type, you can use the `@media` directive and surround the rule with an additional set of brackets:
- `@media only screen and (max-device-width: 480px) { p.bigger {font-size: 16px;} }`
  - The CSS code above sets the font size for **paragraphs** displayed on iPhones.

105

Define Styles for Nested Tags

- You can set up style rules for your page based on how tags are nested inside other tags.
- For example, you can specify that a style rule be applied to a heading tag, but only when that heading is nested inside a certain type of section tag.
- You create such a style rule by specifying a sequence of tags or tag classes in the selector.
- The nested order of tags on your page must match the sequence of the tags in the selector for the style to be applied.
- Defining style rules this way enables you to efficiently apply styles to precise sections of your page.

106



107

TIPs

- **What is the terminology to describe the nested relationship of tags on a page?**
- We can use family-tree terminology. The outer tags are known as *ancestors* and the inner tags are known as *descendants*. In the example above, the `<h2>` tag must be a descendant of the `<section>` tag for the style to be applied.
- Tags directly next to each other in the hierarchy can be given more specific classification. The outer tag is a *parent* and the next tag immediately on the inside is a *child*.
- **How do I specify a parent-child relationship in my CSS rule?**
- You can specify that a style rule be applied only to the immediate descendant, or child, of a tag using a greater-than symbol (>) in the selector. The following applies a rule to a `<p>` tag that is directly inside of an `<article>` tag:
  - `article > p {font-size: 16px; color: green}`
  - This rule would apply to the following:
  - `<article><p>Hello, HTML5!</p></article>`
  - The rule *would not* apply to the following:
  - `<article><section><p>Hello, HTML5!</p></section></article>`

108

Derived Objects

With CSS you can change the flow, layout, position, size, and even adjust the parameters so that the resulting rendering of a tag looks like you want, but the behavior will remain the same, e.g. a <ul> will always behave as a list

109

SPAN and DIV

- Provide bulk formatting for a section of text
- The **<SPAN>** tag is used in conjunction with inline content. In other words, it cannot break across paragraphs
- The **<DIV>** (division) tag defines block content—one that can contain any other tag, including paragraphs and **<SPAN>** content.
  - Can be nested but do not do it.
    - Some Browsers may fail to render the content as you want.

110

The and Tags

- The ** ** is a block level element. In other words, in a Document Model tree it can be either a branch (containing sub-blocks) or a leaf (containing only text to be rendered)
  - A block defined with the ** ** tag can have specific borders, background, margins, text styles, and so on.
- The **** is an inline element. It doesn't have all the properties of a block level element like ** ** but can have specific background, font, and text decoration.
  - It also does not separate itself from other blocks
- The ** ** and **** tags can both be used to make classes.

```
<div class="sidebar_menu">
    <span class="sidebar_menu_item">Item One</span><br/>
</div>
```

111

Summary

- A style consists of three parts
  - The selector—The object, or part of the HTML document
  - The property—The attribute (property) of the object
  - The value—A valid value for that property
- The selector is created using the dot notation
  - tag.class { multiple properties; } or
  - tag.class {
    property: value;
    property: value;
    }
  - The class may not be required, so this is correct
    - tag { multiple properties; }
  - Can be hierarchical like applying style to the emphasis elements of the form tag
    - Form em{font-size:14pt}
- Style data {property:value} are always contained within braces.
- Ex: To set the boarder of a div tag, one can use
  - div.odd_borders { border-top-style: solid; border-bottom-style: dotted; }
  - The boarder has four sides which are border-top border-left border-bottom border-right
  - They can be set all together with the order of top right bottom left
    - div.odd_borders { border-style: solid solid dotted solid; }

112

Summary – cont.

- Inline

```
<html>
<head>Example Inline Styles</head>
<body>
<h1 style="text-decoration: underline overline; font-weight: bolder;">
Heading 1
</h1>
</body>
</html>
```

- Internally, as part of the **<head>** of the document

```
<head>
<title>Example Style Placement</title>
<style type="text/css">
<!-- style statements here -->
</style>
</head>
<body>
<!-- page content here -->
</body>
```

- Externally, in style sheets

113

Standard Selectors

- button //element
- #someId
- .someClass
- * //universal selector, avoid
- table td //descendent
- div > p //child
- div + h1 //subsequent adjacent sibling
- div ~ h1 //subsequent sibling
- a[href] //attribute
- a[href='http://localhost'] //equals
- a[href*='localhost'] //contains
- a[href^='https'] //starts with
- a[href$='.png'] //ends with
- a[data-bind~='css'] //contains - space separated

114

Layout

- Defines the position of **block-level tag** contents.
- Block-level tags place new lines before and after the content they enclose.
- The **<p>, <h1>, and <table>** tags are examples of block-level tags. So are the new semantic HTML5 tags such as **<section>, <article>, <header>, and <footer>**.

115

Set Width and Height for an Element

layout-width.html - Notepad

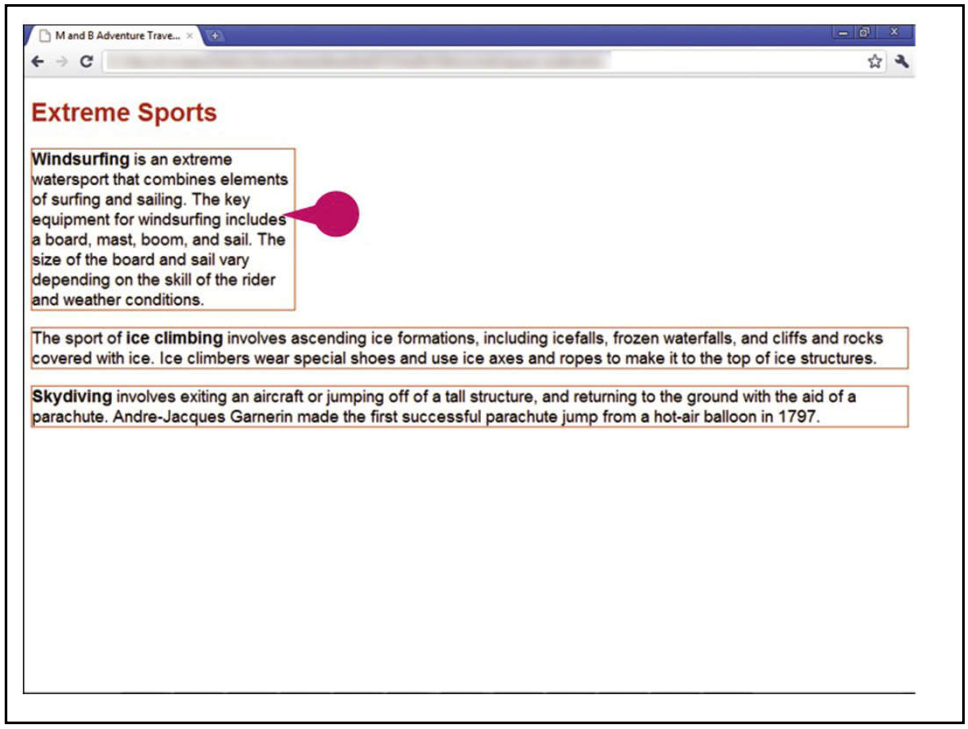
File Edit Format View Help

```
<!DOCTYPE html>
<html>
<head>
<title>M and B Adventure Travel: Extreme Sports</title>
<style>
h1 { font-family: Arial }
h2 { color: #cc3333 }
p { border: solid #cc3333 1px }

</style>
</head>
<body>

<h2>Extreme Sports</h2>

<p class="first"><b>Windsurfing</b> is an extreme watersport that combines elements of
surfing and sailing. The key equipment for windsurfing includes a board, mast, boom, and
sail. The size of the board and sail vary depending on the skill of the rider and weather
conditions.</p>

<p>The sport of <b>ice climbing</b> involves ascending ice formations, including icefalls,
frozen waterfalls, and cliffs and rocks covered with ice. Ice climbers wear special shoes
and use ice axes and ropes to make it to the top of ice structures. </p>

<p><b>Skydiving</b> involves exiting an aircraft or jumping off of a tall structure, and
returning to the ground with the aid of a parachute. Andre-Jacques Garnerin made the first
successful parachute jump from a hot-air balloon in 1797. </p>

</body>
</html>
```

- You can specify values in points (**pt**), pixels (**px**), millimeters (**mm**), centimeters (**cm**), inches (**in**), picas (**pc**)*, x-height (**ex**)**, or em space (**em**)***.

*1pc=(1/6)inche
**1 ex, is a relative measure of the font x character hight
***1 em, is a relative measure of the font M character width

116



117

Relative Size

layout-width.html - Notepad

File Edit Format View Help

```
<!DOCTYPE html>
<html>
<head>
<title>M and B Adventure Travel: Extreme Sports</title>
<style>
body { font-family: arial }
h2 { color: #cc3333 }
p { border: solid #cc3333 1px }

</style>
</head>
<body>

<h2>Extreme Sports</h2>

<p class="first"><b>Windsurfing</b> is an extreme watersport that combines elements of
surfing and sailing. The key equipment for windsurfing includes a board, mast, boom, and
sail. The size of the board and sail vary depending on the skill of the rider and weather
conditions.</p>

<p>The sport of <b>ice climbing</b> involves ascending ice formations, including icefalls,
frozen waterfalls, and cliffs and rocks covered with ice. Ice climbers wear special shoes
and use ice axes and ropes to make it to the top of ice structures. </p>

<p><b>Skydiving</b> involves exiting an aircraft or jumping off of a tall structure, and
returning to the ground with the aid of a parachute. Andre-Jacques Garnerin made the first
successful parachute jump from a hot-air balloon in 1797. </p>

</body>
</html>
```

1

2

- **The web browser displays the element with a width relative to the size of the enclosing box.**

118

What happens to text that extends outside a CSS box?

- You can control how text outside a box is handled using the overflow property.
  - **Setting** the property to visible causes the text to be rendered outside the box.
  - A hidden value hides the text outside the box.
  - Both scroll and auto values display scroll bars for viewing the content, if needed.
- You can assign the **overflow** property to the <p>, , and other block-level tags.

119

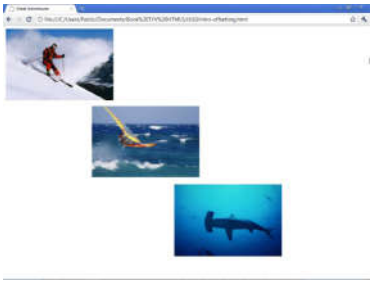
Positioning Content

- *You can place using different positioning*
- ***Relative* positioning** places content on the page **relative to the normal flow** of the other content on the page.
- ***Absolute* positioning** places content on **absolute points on the page relative to the containing block.**
- ***Fixed* positioning** places content **relative to the browser window and keeps it fixed as a user scrolls.**
- **The default is static**

120

Offsetting Content

- You can offset content on your web page from its normal position **using top, left, right, and bottom style sheet properties**.
- These are **ignored** when static positioning is used



121

Use Relative Positioning

- Place content relative to other content on the page.
- If you **offset a relatively positioned element** using the top, left, right, or bottom property, the element is offset relative to the point where it would normally begin.

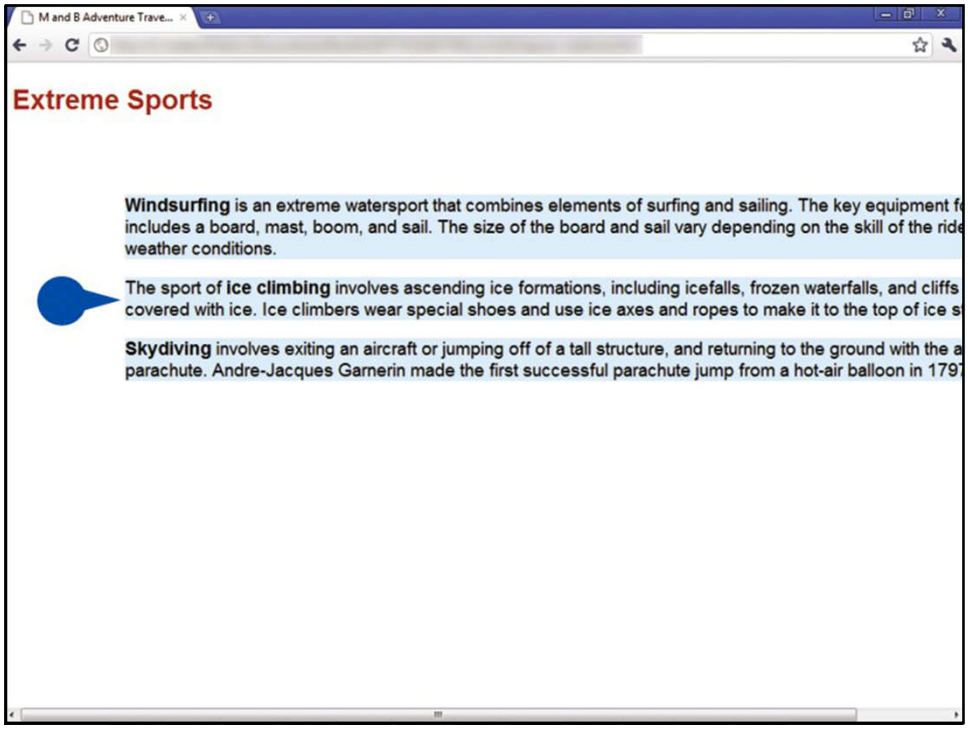
layout-relative.html - Notepad

File Edit Format View Help

```
<!DOCTYPE html>
<html>
<head>
<title>M and B Adventure Travel: Extreme Sports</title>
<style>
body { font-family: arial }
h2 { color: #cc3333 }

</style>
</head>
<body>

<h2>Extreme Sports</h2>

<p><b>Windsurfing</b> is an extreme watersport that combines elements of surfing and
sailing. The key equipment for windsurfing includes a board, mast, boom, and sail. The
size of the board and sail vary depending on the skill of the rider and weather
conditions.</p>
```

2

122



123

Use Absolute Positioning

- To place an element **at exact coordinates** on a page, **independent** of elements that came before it.
- The coordinates are determined relative to the box that encloses it.
- Absolute positioning removes an object from the normal flow of page content.
- Its size and position have no effect on the position of content that follows it.

124

layout-absolute.html - Notepad

File Edit Format View Help

```
<!DOCTYPE html>
<html>
<head>
<title>M and B Adventure Travel: Extreme Sports</title>
<style>
body { font-family: arial }
h2 { color: #cc3333 }
p { background-color: #ccffff; width: 600px }

</style>
</head>
<body>

<h2>Extreme Sports</h2>

<img src="windsurfer.jpg">

<p><b>Windsurfing</b> is an extreme watersport that combines elements of surfing and
sailing. The key equipment for windsurfing includes a board, mast, boom, and sail. The
size of the board and sail vary depending on the skill of the rider and weather
conditions.</p>

<p>The sport of <b>ice climbing</b> involves ascending ice formations, including icefalls,
frozen waterfalls, and cliffs and rocks covered with ice. Ice climbers wear special shoes
and use ice axes and ropes to make it to the top of ice structures. </p>

<p><b>Skydiving</b> involves exiting an aircraft or jumping off of a tall structure, and
returning to the ground with the aid of a parachute. Andre-Jacques Garnerin made the first
successful parachute jump from a hot-air balloon in 1797. </p>

</body>
```

3

4

125

M and B Adventure Trave...

Extreme Sports

Windsurfing is an extreme watersport that combines elements of surfing and sailing. The key equipment for windsurfing includes a board, mast, boom, and sail. The size of the board and sail vary depending on the skill of the rider and weather conditions.

The sport of **ice climbing** involves ascending ice formations, including icefalls, frozen waterfalls, and cliffs and rocks covered with ice. Ice climbers wear special shoes and use ice axes and ropes to make it to the top of ice structures.

Skydiving involves exiting an aircraft or jumping off of a tall structure, and returning to the ground with the aid of a parachute. Andre-Jacques Garnerin made the first successful parachute jump from a hot-air balloon in 1797.

126

Use Fixed Positioning

- You can apply *fixed* positioning to place an element at **exact coordinates on a page** and have it remain fixed while a viewer scrolls.
- Fixed positioning is **supported only in newer web browsers**.
- Some of the methods to keep navigation links visible as visitors view content on a long page.

127

layout-fixed.html - Notepad

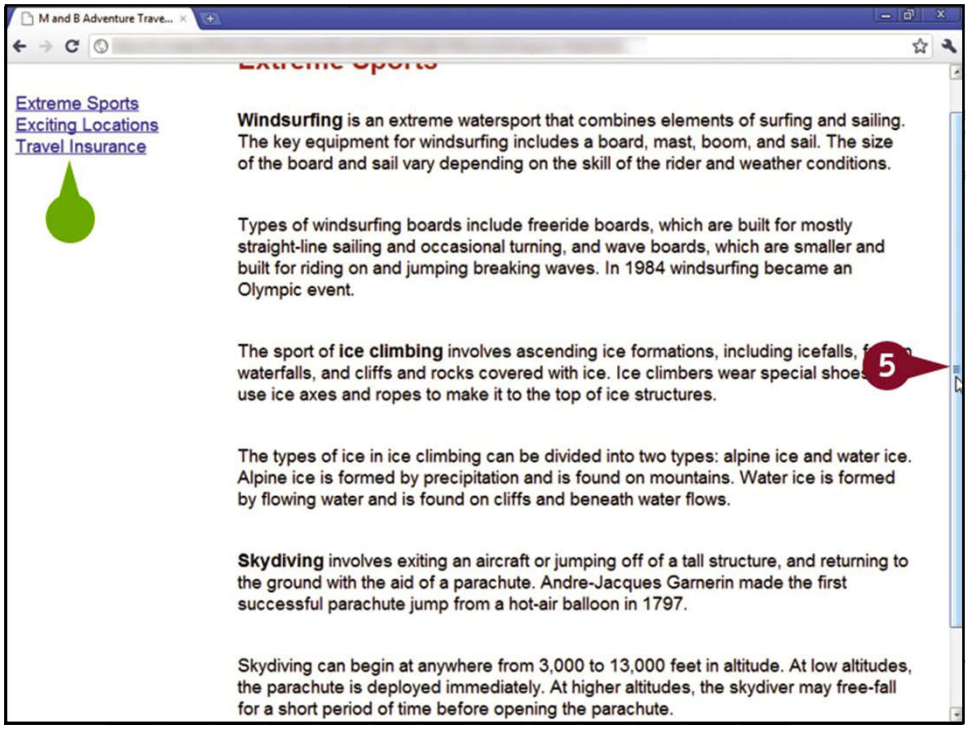
File Edit Format View Help

```
<!DOCTYPE html>
<html>
<head>
<title>M and B Adventure Travel: Extreme Sports</title>
<style>
body { font-family: arial }
h2 { color: #cc3333; position: relative; width: 600px; left: 200px }
p { position: relative; width: 600px; left: 200px; padding: 10px 0 }
ul { list-style-type: none; padding: 0 }

</style>
</head>
<body>

<ul>
<li><a href="sports.html">Extreme Sports</a></li>
<li><a href="locations.html">Exciting Locations</a></li>
<li><a href="insurance.html">Travel Insurance</a></li>
</ul>
</nav>

<h2>Extreme Sports</h2>

<p><b>Windsurfing</b> is an extreme watersport that combines elements of surfing and
sailing. The key equipment for windsurfing includes a board, mast, boom, and sail. The
size of the board and sail vary depending on the skill of the rider and weather
conditions.</p>

<p>Types of windsurfing boards include freeride boards, which are built for mostly
straight-line sailing and occasional turning, and wave boards, which are smaller and built
for riding on and jumping breaking waves. In 1984 windsurfing became an Olympic event.</p>
```

4

3

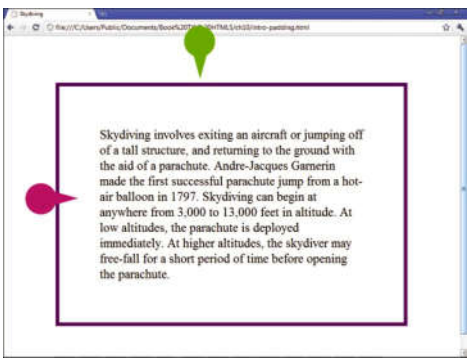
128



129

Padding and Margins

- Space **outside the edge of the box** is known as ***margin***, whereas space **inside the edge of the box** is called *padding*.
- **Style** sheets enable you to **control space** on the top, left, right, and bottom of the boxes independently.



130

Set Margins

layout-margins.html - Notepad

File Edit Format View Help

```
<!DOCTYPE html>
<html>
<head>
<title>M and B Adventure Travel: Extreme Sports</title>
<style>
body { font-family: arial; background-color: #ccffff }
h2 { color: #cc3333 }

</style>
</head>
<body>

<h2>Extreme Sports</h2>

<p><b>Windsurfing</b> is an extreme watersport that combines elements of surfing and
sailing. The key equipment for windsurfing includes a board, mast, boom, and sail. The
size of the board and sail vary depending on the skill of the rider and weather
conditions.</p>

<p>Types of windsurfing boards include freeride boards, which are built for mostly
straight-line sailing and occasional turning, and wave boards, which are smaller and built
for riding on and jumping breaking waves. In 1984 windsurfing became an Olympic event.</p>

<p>The sport of <b>ice climbing</b> involves ascending ice formations, including icefalls,
frozen waterfalls, and cliffs and rocks covered with ice. Ice climbers wear special shoes
and use ice axes and ropes to make it to the top of ice structures. </p>

<p>The types of ice in ice climbing can be divided into two types: alpine ice and water
ice. Alpine ice is formed by precipitation and is found on mountains. Water ice is formed
by flowing water and is found on cliffs and beneath water flows.</p>
```

1 2

131

M and B Adventure Trave...

Extreme Sports

Windsurfing is an extreme watersport that combines elements of surfing and sailing. The key equipment for windsurfing includes a board, mast, boom, and sail. The size of the board and sail vary depending on the skill of the rider and weather conditions.

Types of windsurfing boards include freeride boards, which are built for mostly straight-line sailing and occasional turning, and wave boards, which are smaller and built for riding on and jumping breaking waves. In 1984 windsurfing became an Olympic event.

The sport of **ice climbing** involves ascending ice formations, including icefalls, frozen waterfalls, and cliffs and rocks covered with ice. Ice climbers wear special shoes and use ice axes and ropes to make it to the top of ice structures.

The types of ice in ice climbing can be divided into two types: alpine ice and water ice. Alpine ice is formed by precipitation and is found on mountains. Water ice is formed by flowing water and is found on cliffs and beneath water flows.

Skydiving involves exiting an aircraft or jumping off of a tall structure, and returning to the ground with the aid of a parachute. Andre-Jacques Garnerin made the first successful parachute jump from a hot-air balloon in 1797.

Skydiving can begin at anywhere from 3,000 to 13,000 feet in altitude. At low altitudes, the parachute is deployed immediately. At higher altitudes, the skydiver may free-fall for a short period of time before opening the parachute.

132

Add Padding

layout-padding.html - Notepad

File Edit Format View Help

```
<!DOCTYPE html>
<html>
<head>
<title>M and B Adventure Travel: Extreme Sports</title>
<style>
body { font-family: arial; background-color: #ccffff }
h2 { color: #cc3333 }

</style>
</head>
<body>

<h2>Extreme Sports</h2>

<p><b>Windsurfing</b> is an extreme watersport that combines elements of surfing and
sailing. The key equipment for windsurfing includes a board, mast, boom, and sail. The
size of the board and sail vary depending on the skill of the rider and weather
conditions.</p>

<p>Types of windsurfing boards include freeride boards, which are built for mostly
straight-line sailing and occasional turning, and wave boards, which are smaller and built
for riding on and jumping breaking waves. In 1984 windsurfing became an Olympic event.</p>

<p>The sport of <b>ice climbing</b> involves ascending ice formations, including icefalls,
frozen waterfalls, and cliffs and rocks covered with ice. Ice climbers wear special shoes
and use ice axes and ropes to make it to the top of ice structures. </p>

<p>The types of ice in ice climbing can be divided into two types: alpine ice and water
ice. Alpine ice is formed by precipitation and is found on mountains. Water ice is formed
by flowing water and is found on cliffs and beneath water flows.</p>
```

1 2

133

M and B Adventure Trave...

Extreme Sports

Windsurfing is an extreme watersport that combines elements of surfing and sailing. The key equipment for windsurfing includes a board, mast, boom, and sail. The size of the board and sail vary depending on the skill of the rider and weather conditions.

Types of windsurfing boards include freeride boards, which are built for mostly straight-line sailing and occasional turning, and wave boards, which are smaller and built for riding on and jumping breaking waves. In 1984 windsurfing became an Olympic event.

The sport of **ice climbing** involves ascending ice formations, including icefalls, frozen waterfalls, and cliffs and rocks covered with ice. Ice climbers wear special shoes and use ice axes and ropes to make it to the top of ice structures.

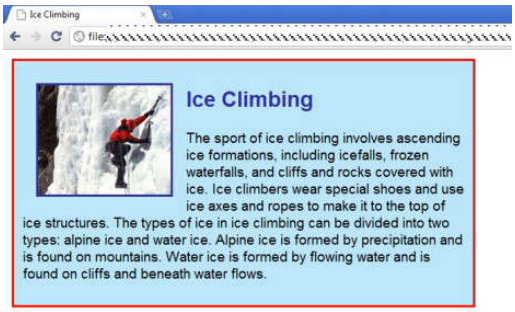
The types of ice in ice climbing can be divided into two types: alpine ice and water ice. Alpine ice is formed by precipitation and is found on mountains. Water ice is formed by flowing water and is found on cliffs and beneath water flows.

Skydiving involves exiting an aircraft or jumping off of a tall structure, and returning to the ground with the aid of a parachute. Andre-Jacques Garnerin made the first successful parachute jump from a hot-air balloon in 1797.

134

Floating Content

- The **float** CSS property takes a box out of the normal flow of your page and **moves it to the right or left side of the enclosing box**.
- **Content** that follows then **wraps around** the **floated element**.



135

Align Elements Horizontally

- The float property **does not work with** elements for which you have assigned an **absolute or fixed position**.

layout-float1.html - Notepad

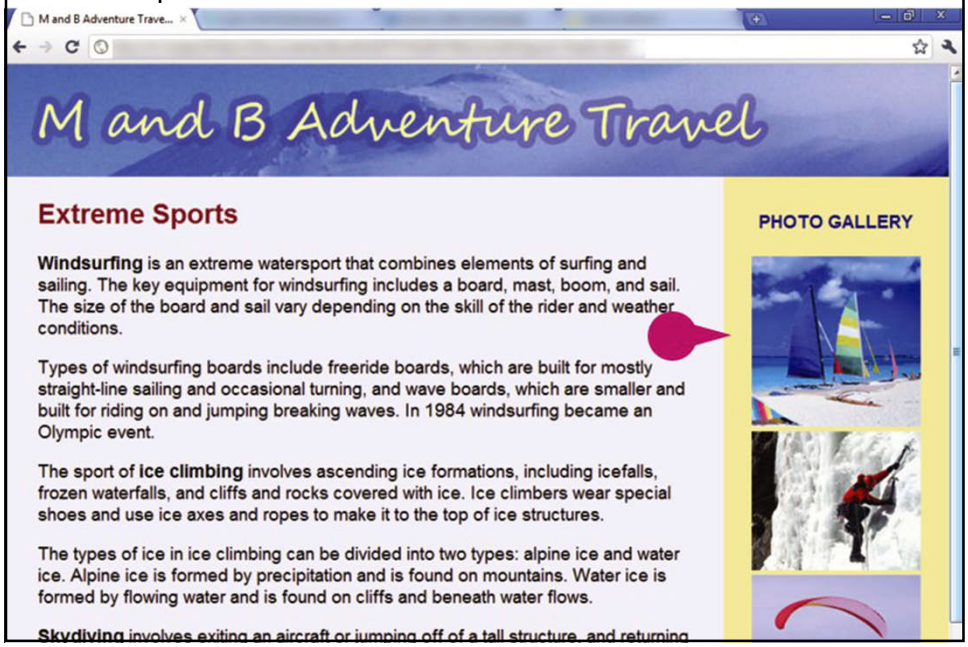
File Edit Format View Help

```
<!DOCTYPE html>
<html>
<head>
<title>M and B Adventure Travel: Extreme Sports</title>
<style>
body { font-family: arial; background-color: #eeeeff; margin: 0; padd    0 }
div.header { width: 100%; height: 100px }
h2 { color: #A11231 }
h2, p { padding: 0 30px }
aside { background-color: #F4E29F; width: 180px; padding: 10px          }
aside h4 { color: #1231A1; width: 180px; font-size: 16px; text-align: center; text-
transform: uppercase }
aside div { width: 180px; text-align: center }

</style>
</head>
<body>

<div class="header"><img src="mbadventure2.jpg"></div>

<aside class="right-align">
<h4>Photo Gallery</h4>
<div><img src="windsurf_sm.jpg"></div>
<div><img src="iceclimb_sm.jpg"></div>
<div><img src="parachute_sm.jpg"></div>
</aside>
```

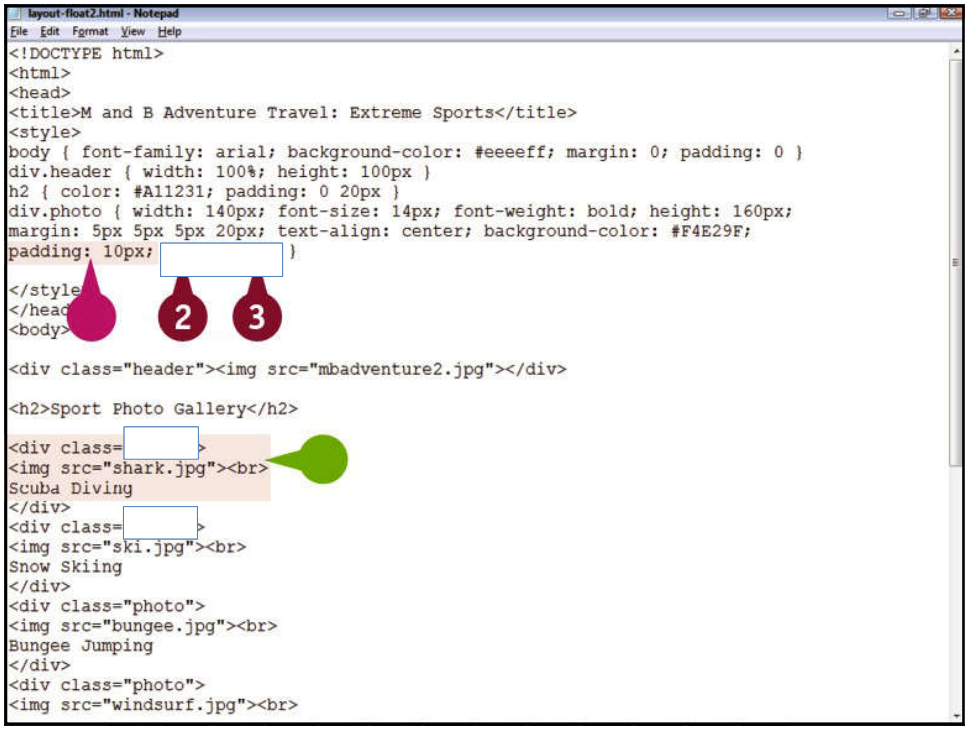
2 3

136

- Content that comes after the floated element in your HTML wraps around the other side.

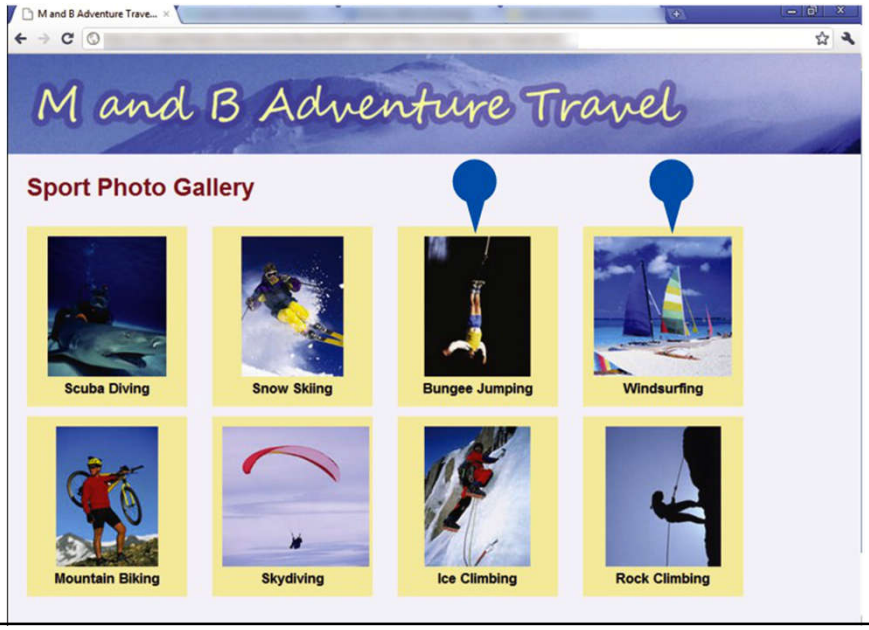


137



138

- By floating several smaller, rectangular elements to the same side, you can create a grid arrangement, which is useful for creating photo galleries.

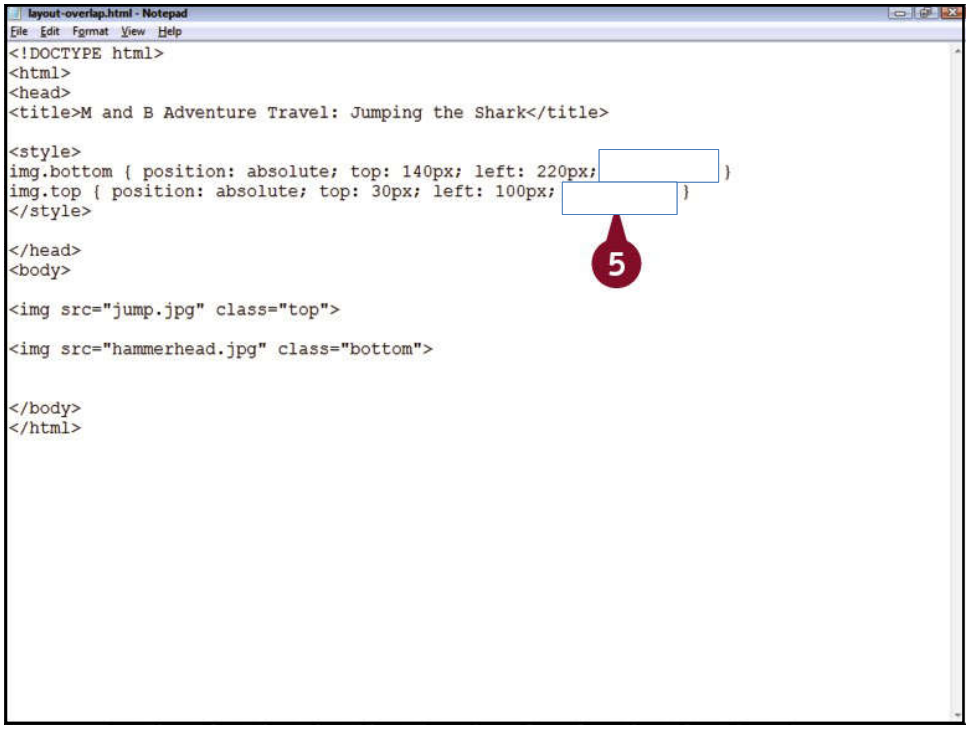


139

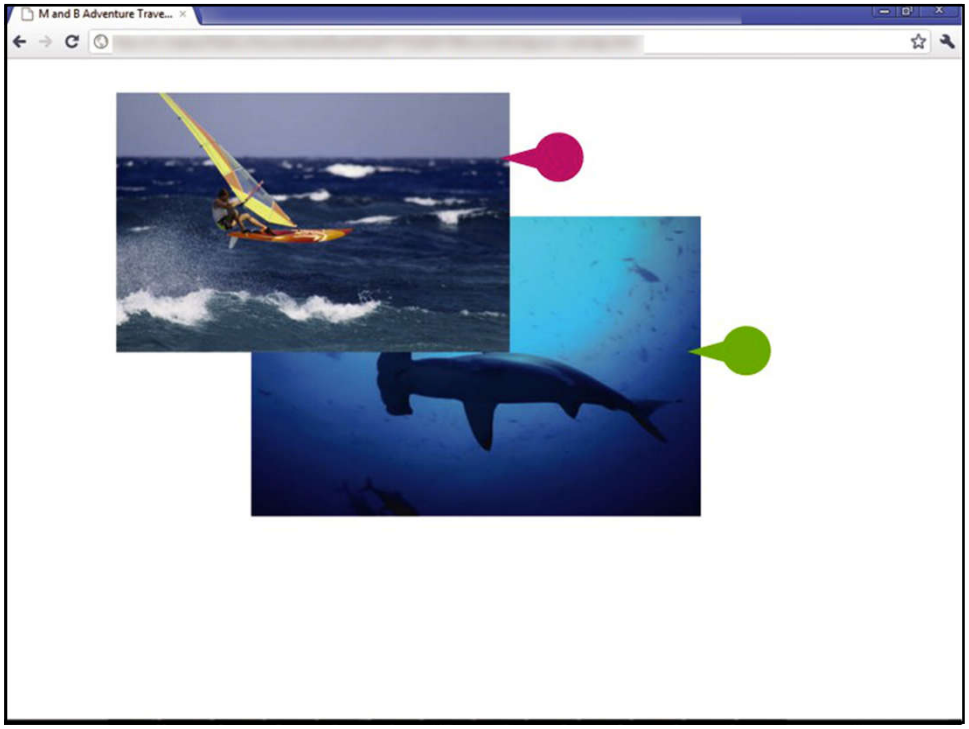
Control the Overlap of Elements

- You can use style sheets to overlap elements on your pages by **positioning them at similar coordinates.**
- You **control** the stack of them by adjusting the z-index property for each element.
- An element with a **higher z-index** value appears **above** an element with a lower z-index value.
- z-index values can be **positive**, **negative**, or **zero**.

140


layout-overlap.html - Notepad
File Edit Format View Help
<!DOCTYPE html>
<html>
<head>
<title>M and B Adventure Travel: Jumping the Shark</title>
<style>
img.bottom { position: absolute; top: 140px; left: 220px; }
img.top { position: absolute; top: 30px; left: 100px; }
</style>
</head>
<body>
<img src="jump.jpg" class="top">
<img src="hammerhead.jpg" class="bottom">
</body>
</html>
5

141


M and B Adventure Trave...

142

Transparency

- Transparency done by changing object opacity.
- However, different browsers recognize different style sheet commands for changing opacity.
  - For **Internet Explorer**, you can type **filter: alpha(opacity=?)** in your style rule, replacing ***?*** with a value from **0 to 100**.
  - **For other popular browsers**, you can type **opacity: ?** in your style rule, replacing ***?*** with a fractional number from **0.0 to 1.0**.
  - You can put both properties in a declaration to make the effect compatible with as many web browsers as possible.

143

Borders

- Borders can be set explicitly or as a group
- There are **three basic parameters** that each border can be set with
  - Style—For example, solid, dotted, and so on
  - Color—The color
  - Width—The width, in pixels, percentage, or relative measurement
- There are three ways that the parameters can be specified.
  - You **can group** a border by edge
    - border-top: style color width;
  - The **parameters can be grouped** for all edges
    - border-width: width;
    - border-style: solid;
    - border-color: red;
  - They can be **set explicitly for a specific edge-property combination**
    - border-top-width: 5px;
    - border-bottom-style: dotted;
    - border-right-color: blue;
    - border-left: solid red 10px;

144

CSS Objects and Properties Short Reference

- Background
  - can be used with practically any element, from divisions and table cells to the document body itself.
  - Forms
    - background-color : red;
    - background-color : #f00;
    - background-color : #ff0000;
  - Ex:
    - div.warning_text { background-color: red; }
    - It is also possible to specify a background image
      - background-image: url("http://www.server.com/myimage.jpg")
      - The image will **scroll with** the rest of the content, but it **can be fixed** to the viewport by specifying fixed as a value for the background-attachment property.
      - For example:
        - » body { background-image: url("http://www.server.com/myimage.jpg"); **background-attachment : fixed;**}

145

Background cont.

- The background-repeat property is used If the image is smaller than the allotted space (canvas), it is to make it repeat to cover the available space.
- To set this behavior, there are four possible settings, from which one may be chosen:
  - background-repeat : repeat repeat-x repeat-y no-repeat
  - If repeat is specified, the image is tiled throughout the background, starting from the center of the background. If repeat-x or repeat-y is specified, the image is repeated in either the horizontal or vertical axis, working from the inside out.
- To stretch an image use the CSS3 property, background-size
  - body {
  - background: url(bgimage.jpg) no-repeat;
  - background-size: 100%;
  - }
  - Or use width and height properties of the image with fixed position to size it

146

Text and Fonts

- Any text block or span can have its text and font properties set; this includes the div, p, table, and span elements.
  - The font, color, spacing, decoration (underlining, for example), line height, weight (bold, for example), and alignment (center, right, and so on) can all be specified using styles.
- **Text color**:
  - p.red_text { color: red; }
  - p.red_text { color: #F00; } <!-- #RGB -->
  - p.red_text { color: #ff0000; } <!-- #RRGGBB -->
- **Font style**:
  - font-style: normal, italic, or oblique
- **Font weight**:
  - font-weight: normal, bold, bolder, lighter
- **Font size**:
  - font-size: pt size or percentage
    - font-size: 14pt;
    - font-size: 200%:
- **Text decoration**:
  - text-decoration: underline, overline, line-through, none
  - This cannot combined with font compound property, see next slide
- **Text alignment**:
  - text-align: left, right, center, justify
- **Text indent**:
  - text-indent: number or percentage
- **Text transformation**:
  - text-transform: uppercase, capitalize, lowercase
    - Capitalized option only transforms the first letter where uppercase transforms all the letters of a word.

147

Text and Fonts – cont.

- To make sure that all the **lines are spaced with reference to the largest font** used in the entire paragraph, **and not** space each line **according to the size of the text contained within**, you must set the **line-height** property:
  - line-height: pt size, multiplier or percentage
    - The multiplier is a simple number, and, like the percentage, is calculated with reference to the font size of the element to which the style property is being set.
      - ** **
      - ** **
      - ** **
- **Font Family**:
  - font-family: name, generic name;
    - Using this form, you can identify an exact font, with a generic alternative:
    - font-family: Times New Roman, serif;
  - Generic names are
    - Serif—Times New Roman, Garamond, and so on
    - Sans-serif—Arial, Verdana, and so on
    - Cursive—Script, Corsiva, and so on
    - Fantasy—Anything goes. . .
    - Monospace—Courier New and so on
- The font property can also be specified as a compound property statement:
  - **font: style + variant + weight + size + line-height + family**
    - div.article:first-line {
    - font: normal small-caps bold 12pt
    - 14pt Times New Roman, serif;
    - }

148

styles.css - Notepad

File Edit Format View Help

```
p { font-size: 20px }
p { text-indent: 30px }
h1 { color: green }
h2 { color: #336699 }
body { font-family: arial, "gill sans" }
h1 { text-align: center }
h2 { text-transform: uppercase }
li { line-height: 2.0 }
h3 { letter-spacing: 6px }
span.highlight { background-color: #ffcc99 }
ul.horiz { list-style-type: none; padding: 0; float: none }
ul.horiz li { float: left; padding: 10px }
```

8 9 10 11

149

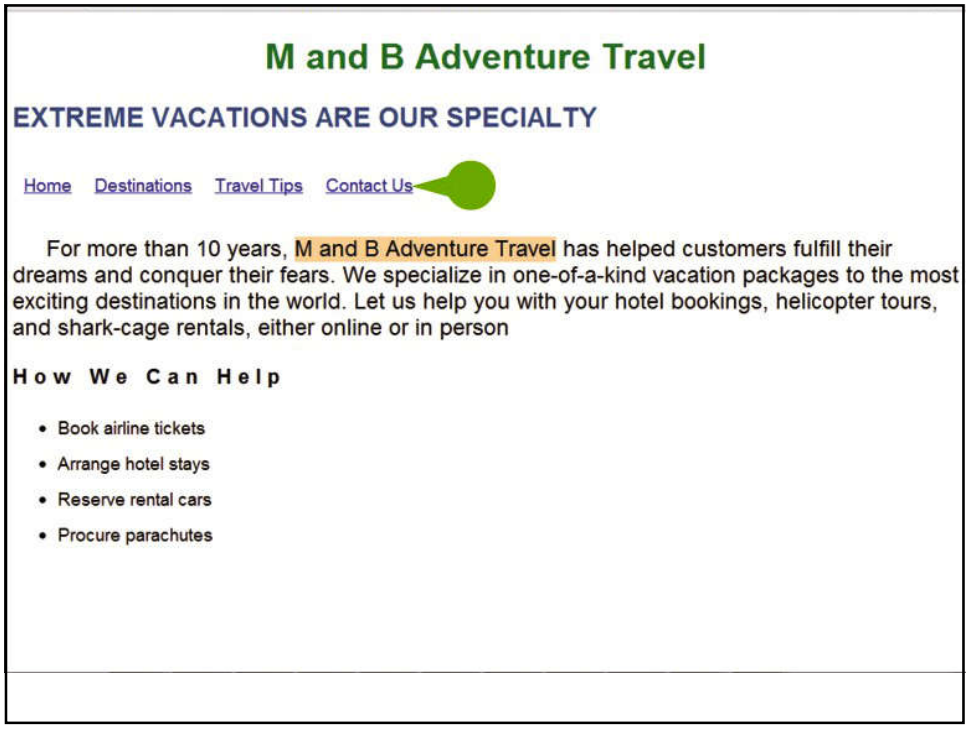
mbtravel.html - Notepad

File Edit Format View Help

```
<!DOCTYPE html>
<html>
<head>
<title>M and B Adventure Travel: Home</title>
<link rel="stylesheet" href="styles.css">
</head>
<body>

<h1>M and B Adventure Travel</h1>

<h2>Extreme vacations are our specialty</h2>

<ul class="horiz">
<li><a href="index.html">Home</a></li>
<li><a href="dest.html">Destinations</a></li>
<li><a href="tips.html">Travel Tips</a></li>
<li><a href="contact.html">Contact Us</a></li>
</ul>

<br style="clear: both">

<p>For more than 10 years, <span class="highlight">M and B Adventure Travel</span> 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 your hotel bookings, helicopter tours, and shark-cage rentals, either online or in
person</p>

<h3>How We Can Help</h3>

<ul>
<li>Book airline tickets</li>
<li>Arrange hotel stays</li>
<li>Reserve rental cars</li>
<li>Procure parachutes</li>
```

2 1 3

150



151

Image File Formats

- Although numerous file types are used for computer images, JPEG, GIF, and PNG are the three most popular types used on the web.
- Another option for displaying graphical content on your pages in HTML5 is by using the new <canvas> tag.
- **JPEG**
  - stands for ***Joint Photographic Experts Group***, supports **24-bit** color.
  - JPEG is not a good choice for solid-color artwork because it results in a **larger overall file size**, which translates to longer download times.
- **GIF**
  - stands for ***Graphics Interchange Format***, supports up to **256** colors.
  - If your image or graphic contains few colors and **not a lot of detail**, such as logos.
  - A single GIF file can also **store multiple images** and display them as an animation.
- **PNG**
  - Stands for ***Portable Network Graphics***, offers **rich color** support and **advanced compression** schemes, so it is a good choice for a variety of image types.
  - supports **24-bit** color but can also be saved with fewer colors.

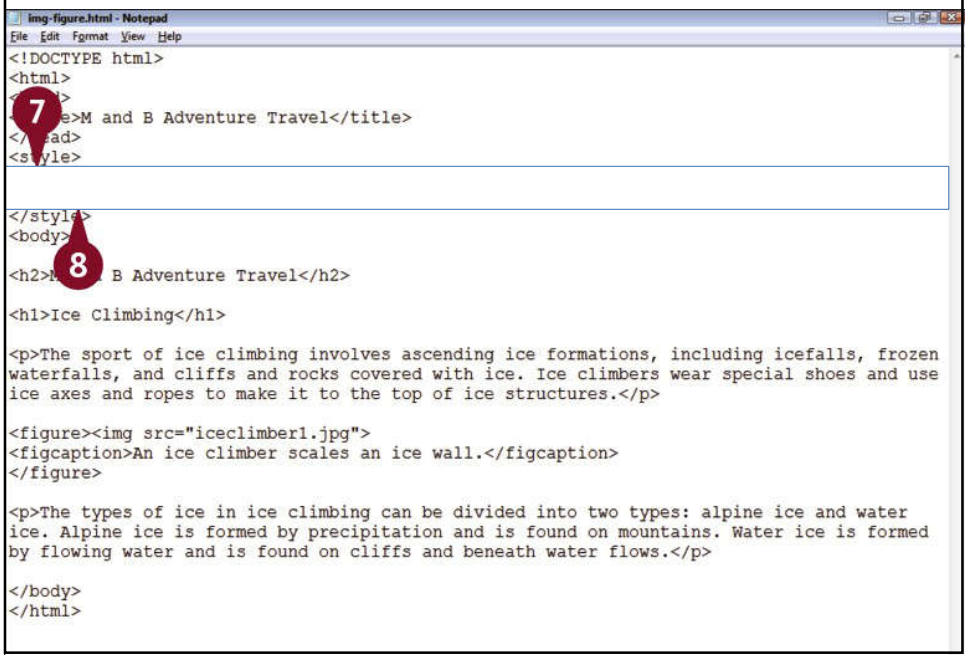
152

Images

- To insert an image into a web page
  - **<IMG SRC="image_name" WIDTH="width" HEIGHT="height" ALT="Image Text" title="The image title">**
  - One can use Width and height to resize the image as needed
  - The alternative attribute ALT used to display text description in case the image is not loaded. This also used by SEO.
  - **The title will be shown as pop up note when the mouse hovers over the image.**
  - Another attribute is the **BORDER** attribute.
    - Value "0" indicate no border.

153

Using Figures and captions



154

M and B Adventure Travel

Ice Climbing

The sport of ice climbing involves ascending ice formations, including icefalls, frozen waterfalls, and cliffs and rocks covered with ice. Ice climbers wear special shoes and use ice axes and ropes to make it to the top of ice structures.

An ice climber scales an ice wall.

The types of ice in ice climbing can be divided into two types: alpine ice and water ice. Alpine ice is formed by precipitation and is found on mountains. Water ice is formed by flowing water and is found on cliffs and beneath water flows.

155

Image map

- Consider the problem of having a picture where each part of it reference something else.
  - E.g. the world map, the user click on a country to get the information about it.

156

The USEMAP attribute and the <MAP> tag

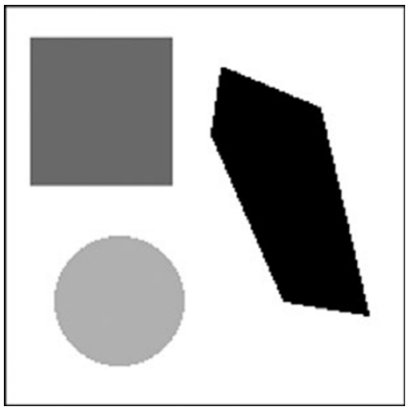
- To turn the image into a client side imagemap, you need to specify the USEMAP attribute and insert a reference to the <MAP> element that contains the specific information relating to the areas of the image that can be clicked.
- The areas of the map can be rectangular, circular, or irregularly shaped.
  - Need to know the coordinates.
  - Use MS Paint or any paint/photo editing program

157

Imagemap

- Rectangle 13,16 83,90 (Top-Left and Bottom-Right)
- Circle 58,147 32 (Center and Radius)
- Polygon 108,31 158,51 181,155 139,147 103,65

.



158

Imagemap example

```
<IMG SRC="imagemap.jpg" WIDTH="200" HEIGHT="200”
      USEMAP="#image_map">
<MAP NAME="image_map">
<AREA HREF="index.html" SHAPE="RECT" COORDS="13,16,83,90">
<AREA HREF="page2.html" SHAPE="CIRCLE" COORDS="58,147,32">
<AREA HREF="page3.html" SHAPE="POLY"
       COORDS="108,31,158,51,181,155,139,147,103,65,108,31">
</MAP>
```

- **The # sign is used for internal reference, e.g. USEMAP=“#image_map”**

.

159

Video and audio

- HTML5 introduces two new tags, <video> and <audio>, for embedding multimedia into web pages.
- You embed a file by providing a src value along with the tag, similar to how you embed an image file with the <img> tag.
- Attributes for the<video> and <audio> tags enable you to
  - show player controls,
  - make clips repeat,
  - preload content, and
  - provide alternative formats and sources of the media.
- Prior to HTML5, web designers embedded video and audio using complicated combinations of <embed> and <object> tags.
  - Those tags depended on web browsers having separate helper applications known as *plug-ins* installed to view the content.
- Browsers that support the HTML5 <video> and <audio> tags can play video and audio content *natively*, which means they do not need extra plug-ins.

160

Before you view the video

- You have to record it with as low size as possible (resolution and size conflict)
- Convert it to web-friendly formats
  - Many free and inexpensive converters
    - Ex: the open-source MiroVideo Converter www.mirovideoconverter.com.
    - For audio you can use AVS Audio Converter www.avs4you.com.

161

Formats in different browsers

- Support for video and audio file formats varies across browsers.

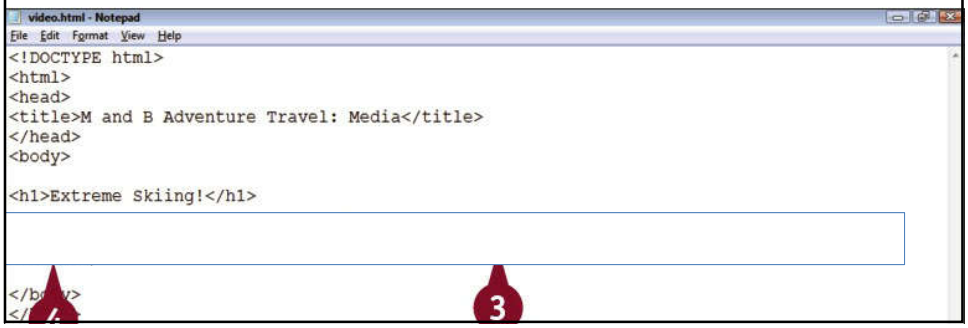
Browser	Video	Audio
Google Chrome 10	Ogg Theora (ogg, .ogv), H.264 (.m4v), WebM (.webm)	Vorbis (.ogg, .oga)
Mozilla Firefox 4	Theora (.ogg, .ogv), WebM (.webm)	Vorbis (.ogg, .oga)
Internet Explorer 9	H.264 (.m4v), WebM (.webm)	Vorbis (.ogg, .oga)
Apple Safari	H.264 (.m4v)	MP3 (.mp3)

- To make sure as many users can see your content as possible, you can offer clips in multiple formats at once.

162

Insert a Video File

- Use the <video> tag.
- The controls attribute allows the user to start and stop the video clip or control it in other ways.
- After the <video> tag, you can type alternative content that appears if the browser is unable to load the video file.
- You can add the autoplay attribute to the <video> tag to start playing it automatically. For example: <video src="myvid.ogg" autoplay></video>

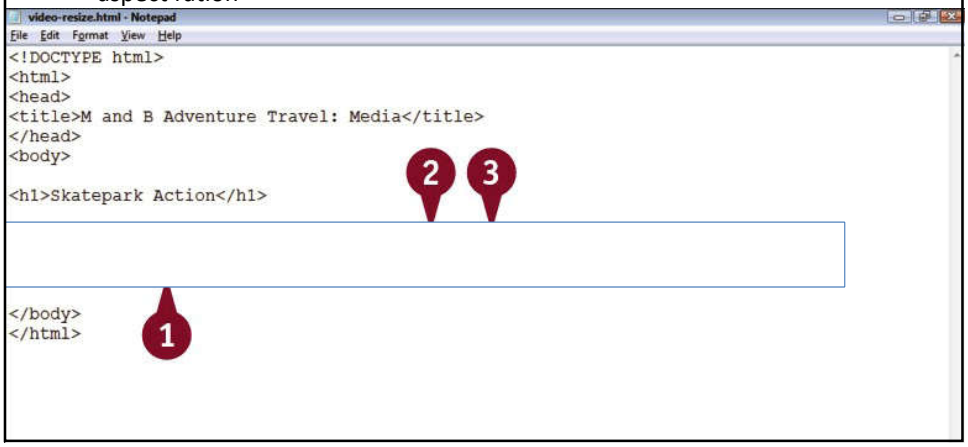


163

164

Resize a Video

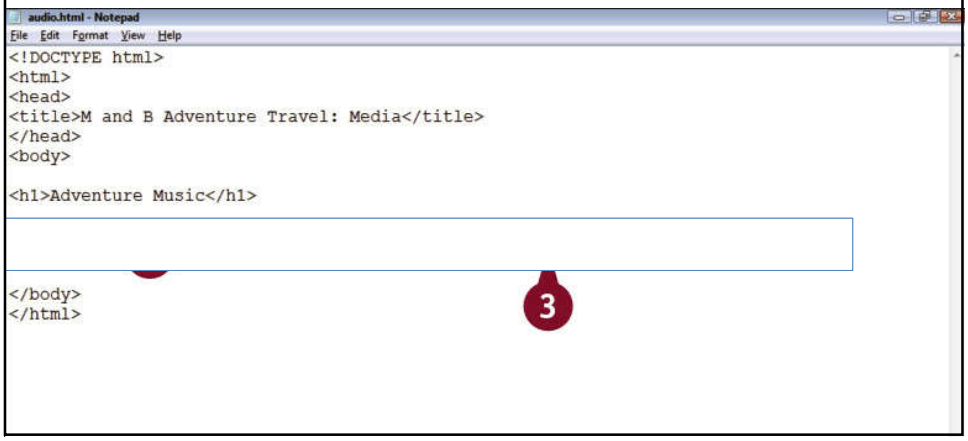
- You can add width and height attributes to change the dimensions of the video
- If you do not specify dimensions, the browser plays the video at the size at which it was saved.
- When specifying one dimension, the other is resized to the corresponding aspect ration



165

Insert an Audio File

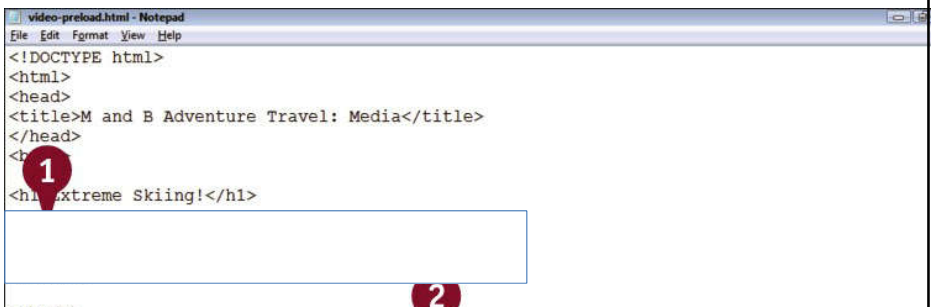
- You can add audio to your page, add controls, or play in the background.
- Controls may appear slightly different across different web browsers.
- You can add alternative text inside your <audio> tag that appears if the browser is unable to play the audio file.
- You can use the autoplay to play audio automatically Ex: in the background
  - <audio src="myaudio.ogg" autoplay></audio>



166

Preload Multimedia

- Preloading a multimedia allows a web browser to start downloading the multimedia file whether or not the user is actually playing it.
- This can give viewers a better experience
- The downside is that it wastes network resources if the user never plays the file.



167

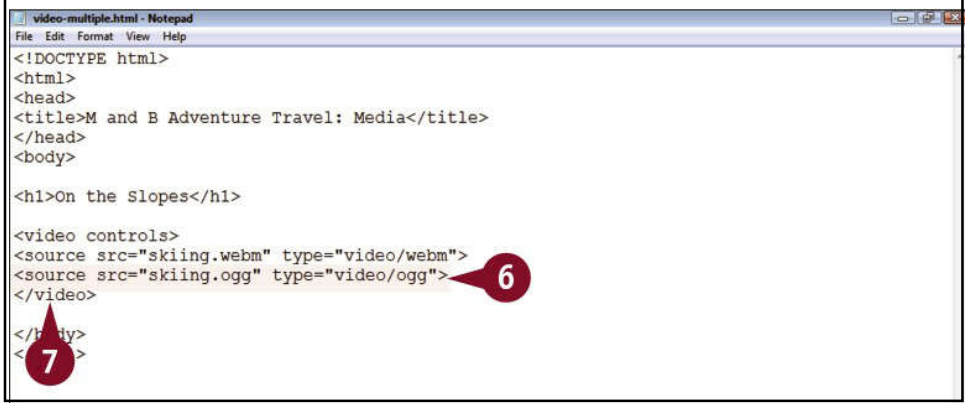
Loop Multimedia

- You can use the loop attribute in the <video> or <audio> tags to cause your clips to repeat after they finish.

168

Offer Multiple Sources

- You can associate your <video> tag with both a WebM video file (.webm) as well as an Ogg Theora video file (.ogg, .ogv) at the same time.
- HTML5-capable browsers attempt to load one source and, if it is not recognized, can try loading the next source.
- MIME stands for *Multipurpose Internet Mail Extensions*, many types has been added as MIME types in HTML5



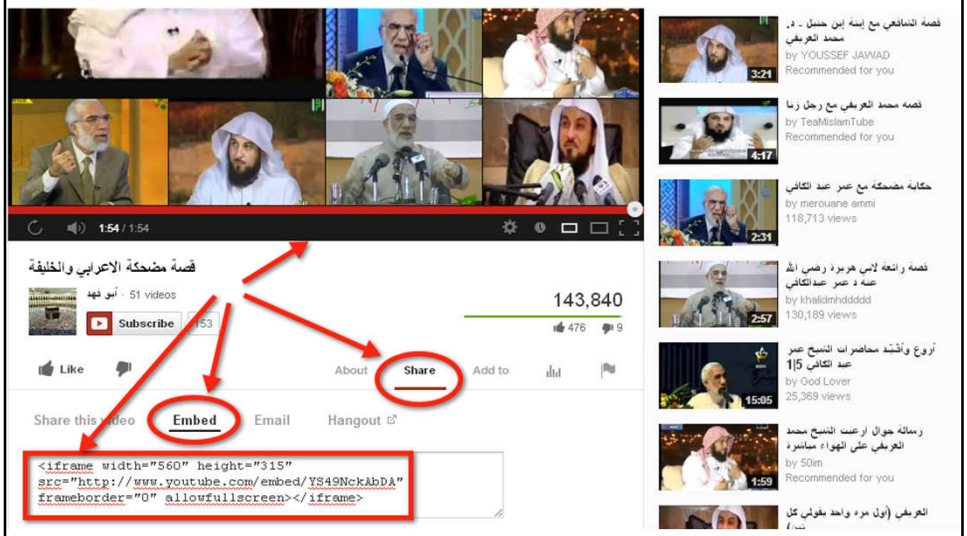
169

Support Older Browsers

- For video, you can include additional HTML tags with your <video>tag to support older web browsers.
- This is done by including both the <embed> tag and the <object> tag.
- They can also be used for the <audio> tag

170

Adding video from youtube



171

Youtube – cont.

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>M and B Adventure Travel: Islam</title>
</head>
<body>

<h1>Omar Abd Alkafi</h1>

<iframe width="560" height="315" src="http://www.youtube.com/embed/YS49NckAbDA"
frameborder="0" allowfullscreen></iframe>

</body></html>
```

172

Add a Meter Image

- The <meter> tag is new in HTML5. You set the range using the min and max attributes.

img-meter.html - Notepad

File Edit Format View Help

```
<!DOCTYPE html>
<html>
<head>
<title>M and B Adventure Travel</title>
</head>
<body>

<h2>Mountain Safety Test Results</h2>

Mary: <meter value="97" min="0" max="100" low="12" high="97" optimum="100">97
points</meter><br>
Joe: <meter value="12" min="0" max="100" low="12" high="97" optimum="100">12
points</meter>

</body>
</html>
```

7 8 9 10

The low attribute indicates the lowest realized value in the range.
The high attribute indicates the highest realized value in the range.
The optimum attribute indicates the range optimum.

173

Mountain Safety Test Results

John:
Mary:
Joe:

Display progress

- $0 <progress value=”850” max=”1000”>$850</progress> $1000

M and B Adventure Travel

Trip Fundraiser Progress

We've raised $850 so far!

$0 $1000

174

Pseudo Classes

- :link
- :visited
- :active
- :hover
- :focus
- :checked
- :lang(language)
- :not(selector)
- :first-child
- :last-child
- :nth-child(an+b)
- :nth-last-child(an+b)
- :only-child
- :only-of-type
- :first-of-type
- :last-of-type
- :target //ID at end of URL - e.g. 'http://a.com/index.html#test'

```
//examples
p:lang(en) {}
div p:nth-child(5) {}
div p:nth-child(odd) {}
div p:nth-child(even) {}
div p:nth-child(3n+2) {}
```

175

Pseudo-Elements

- There are elements that have no corresponding tag in HTML.
  - They might be **contained within a tag** or just be **outside the document model**.
  - They are, however, quite **browser-dependent**, and so I will cover only two of them here.
    - **The first line of a block (div, p, and so on)**
      - The first line might not be as long or short on all browsers
      - It is possible to make sure (using specific font sizes in points and bounding boxes in pixels) that the text takes the same space on every platform, but often this is not desirable.
        - » div.article::first-line { text-transform: capitalize; }
    - **The first letter of an inline or block element (span, block, p, and so on)**
      - If you assume that the very first letter should be given extra importance, you can add the following:
        - » div.article::first-letter { font-size: 14pt; }
    - div { width: 150px; margin:10px;}
    - div::first-line { text-transform: uppercase; }
    - div::first-letter { font-size: 200%; float:left;}

```
::before //can set content
::after //can set content

//examples
#myParagraph::after {
      color: #FFF;
      content: 'test content';
}
#myParagraph::first-letter {
      font-weight: bold;
}
```

176

Lists style

- They can have **various bullet types, borders, backgrounds, and layouts, as well as many variations of counted lists**, including special character sets and numbering schemes.
- **list-style-position** indicates where the bullets (or numbers, letters, and so on) should be placed in relation to the list item text and formatting box. It is specified as follows:
  - list-style-position: inside, outside

```
<ul style="list-style-position: inside;border: 1px solid black;">
<li>These bullets are</li>
<li>inside the box</li>
</ul>
<ul style="list-style-position: outside;border: 1px solid black;">
<li>These bullets are</li>
<li>outside the box</li>
</ul>
```

177

Lists style – cont.

- list-style-type
  -
    - list-style-type: decimal, alpha, roman
      - The alpha and roman are used in conjunction with the word lower- or upper- prefixes
  -
    - list-style-type: disc, circle, square
  - Using an image
    - list-style-image: url(http://mydomain.com/image_file.png)
  - You can group these into a single compound statement
    - list-style: type + position + image
      - **<ul style="list-style: disc inside url (http://domain.com/image/image_file.jpg;">**

178

CSS3 and CSS4!!

- New features in CSS3 include support for additional selectors, drop shadows, rounded corners, multiple backgrounds, animation, transparency, and much more.
- CSS3 is not finalized and yet to be standarized
- Browser vendors have implemented many of the new features in CSS3 using their own **"**prefixed**"** versions of a property.
  - For example, to transform an element in Firefox, you need to use the **-moz-transform** property; to do the same in WebKit-based browsers such as Safari and Google Chrome, you have to use **-webkit-transform**.
  - In some cases, you**'**ll need up to four lines of code for a single CSS property

179

CSS3 Colors

- You can use the red, green, blue, alpha RGBA() or the hue, saturation, light, alpha HSLA() methods
  - form {
  - ⋮
  - background: rgba(0,0,0,0.2) url(../images/bg-form.png) no-repeat bottom center;
  - }
- Red, green, blue takes values form 0 to 255
- Alpha takes a value from 0 to 1
- The hue, in degrees from 0 to 359. it represents a basic color, for example (240 = blue, and 300 = magenta)
- The saturation, as a percentage. It represents the intensity of the color

180

Rounded Corners: border-radius

- Safari, Chrome, Opera, IE9, and Firefox 4 support rounded corners
- Example:
  - -moz-border-radius: 25px;
  - border-radius: 25px;
  - border-top-left-radius: 5px;
  - border-top-right-radius: 10px;
  - border-bottom-right-radius: 15px;
  - border-bottom-left-radius: 40px;

181

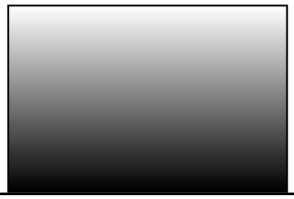
Drop Shadows

- Example
  - **-webkit-box-shadow: 2px 5px 0 0 rgba(72,72,72,1);**
  - **-moz-box-shadow: 2px 5px 0 0 rgba(72,72,72,1);**
  - **box-shadow: 2px 5px 0 0 rgba(72,72,72,1);**
- The first value is the horizontal offset**,** the second value is the vertical offset. A positive value will create a shadow to the right or down of the element respectively, a negative value to the left or up respectively.
- The third value, if included, is the blur distance of the shadow.
- The fourth value determines the spread distance of the shadow.
- The fifth is the color of the shadow
- For Text you can use the **text-shaow**

182

Linear Gradient

- background-image: linear-gradient(…);
- You specify the direction of the gradient, and then provide some color stops.
- For the direction, you can provide either the angle along which the gradient should proceed, or the side or corner from which it should start
  - background-image: -moz-linear-gradient(270deg, #FFF 0%, #000 100%);
  - background-image: -moz-linear-gradient(top, #FFF 0%, #000 100%);
  - background-image: -moz-linear-gradient(#FFF 0%, #000 100%);



183

Radial Gradients

- background-image: -moz-radial-gradient(30px 30px, #FFF, #000);
- The first argument is the center position, then the colors of the gradient
- Other arguments can be added

184

Animation

```
Element_selector{
    …
    animation-name: some_name;
    animation-duration: 5s;  /*animation duration time*/
    animation-timing-function: ease-in|linear|...;
    animation-iteration-count: infinite|integer_value|...;
    animation-direction: normal|reverse|alternate|...;
    animation-fill-mode: none|backwards|forwards|...;
    …
}
@keyframe some_name{
    %0{CSS properties}
    %...
    %50{CSS properties}
    %...
}
```

185

Transforms

- Supported in Firefox 3.5+, Opera 10.5, WebKit since 3.2 (Chrome 1), and even Internet Explorer 9.
- The CSS3 transform property lets you **translate**, **rotate**, **scale**, or **skew** any element on the page.

187

Translate example

- Let' s say we want to move the word " AD" over to the right when the user hovers over it

<h1>Put your AD up here</h1>

So we can write the following CSS3 code

```
h1:hover span {
color: #484848;
display:inline-block; /* to make an element as a block (needed for webkit browsers)*/
-webkit-transform: translateX(40px);
-moz-transform: translateX(40px);
-ms-transform: translateX(40px);
-o-transform:translateX(40px);
transform: translateX(40px);
}
```

188

Other transform functions

- The **translate(x,y)** function moves an element by x from the left, and y from the top.
- The scale(x,y) function scales an element by the defined factors horizontally and vertically, respectively.
- The rotate(angle in deg) function rotates an element around the point of origin
- The skew(x,y) function specifies a skew along the X and Y axes

189

Example

```
h1:hover span {
color: #484848;
-webkit-transform:rotate(10deg) translateX(40px) scale(1.5);
-moz-transform:rotate(10deg) translateX(40px) scale(1.5);
-ms-transform:rotate(10deg) translateX(40px) scale(1.5);
-o-transform:rotate(10deg) translateX(40px) scale(1.5);
transform:rotate(10deg) translateX(40px) scale(1.5);
}
```

190

More in CSS3

- You can control the translation duration by
  -webkit-transition-duration: 0.2s;
  -moz-transition-duration: 0.2s;
  -o-transition-duration: 0.2s;
  transition-duration: 0.2s;
- And much more
  – You can also work with animation in CSS3 and control frames and timeline!!

191

Client Side Scripting

- Generally speaking, when you include a client side script within the Web page (or as an external script), it is to perform one of the following functions:
  - Improve navigation
    - drop-down menus
  - Provide special effects
    - inline calendars or image selection (from thumbnail examples), which layer CSS (styles, such as div elements) on top of regular HTML.
  - Provide contextual highlighting
    - includes offering topic expansion or inline help as well as floating sign-up boxes or splash screens.
  - Validate forms or data
  - Provide dynamic content
    - the ability to switch between content without loading another page (that is, like a tab control), as well as retrieving and displaying information from the server based on data that the user is entering (AJAX)
  - Communicate within pages (inter-page communication)
    - Shopping cart or questionnaire (survey) platform for multi-page questionnaires.

192

Intro.

- Client side scripts are executed by the client (the browser) as if they are extensions to the HTML page that is being displayed
  - **browser** must have **support** for the scripts
    - **Build-in** scripts: JavaScript, HTML, XML
    - Add-ins: Adobe Flash player, Microsoft Silverlight, VRML, and so on.
- Scripts can be **included inline** or **externally** loaded as separate files. The Web server can serve scripts in the same way that it can serve image files and external style sheets—they are just another reference to be loaded as part of the page.
- **Before executing scripts**, the **browser** generally tries to **make sure that it has all the information it needs**.

193

Inline Scripts

- Placed in the head section of the HTML document and will be accessible for all the HTML document.
  - **<script>**
  - **// Code lines go here…**
  - **</script>**
  - This can also be placed inside the body tag as well.
- Inside the script tag you place functions and variables.
- Usually, the type of script should be mentioned in the script tag: <script type="text/JavaScript">
- scripts can be used as actions in, for example, form controls.
  - This makes use of the events that are part of the HTML DOM and defined by the W3C.

194

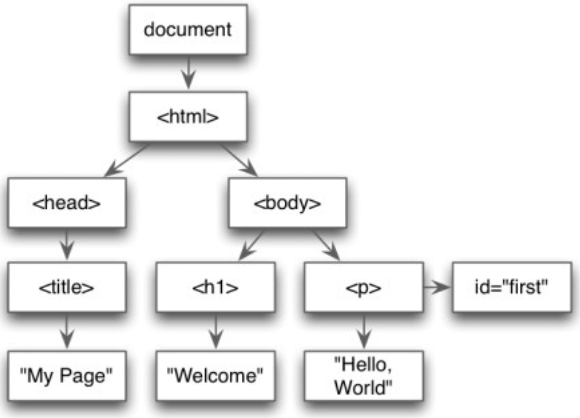
External Scripts

- Can be shared across multiple Web pages.
- In addition, there is a certain amount of hiding of the script that takes place because it is on a server and not directly in the page.
  - However, scripts are never going to be safe unless you take additional steps to protect them by using a tool that makes them unreadable.
    - The use of an interpreter or virtual machine such as in java applets.
    - Or using on-the-fly encryption algorithm would be better off encoded

195

DOM or the Document Object Model

- **JavaScript** can access and change your **elements** on a page using DOM.
- DOM is hierarchal representation of HTML5 **tags.**



196

Introduction to JavaScript

- Can be used: **inline, external, mixed**
- An **inline** script must be contained within a matching pair of **<script>** tags, either in the head or the body section
  - **<script language="JavaScript1.1">**
    - **// Code lines go here…**
  - **</script>**
- The script tag may sometimes omitted when JS is used **directly as event handler**
  - **<input type="submit" value="Submit" onClick="document.myForm.name.style = 'background-color:red'">**
  - **Back**

197

External JavaScript documents

- External JavaScript documents **must have the extension .js** and must be specified by **the Web server** as a type of **document that it can serve**.
- **Configuration section** is required so the correct MIME type is passed to the browser; without it, the browser might not know how to process the included file.
- To include an external script
  - **<script src="my_functions.js" type="text/javascript">**
  - **The type attribute is often omitted**, because it is assumed that the server will supply the correct MIME type to the browser. However, because this is not guaranteed, I prefer to give the browser all the information that it might need in advance.

198

Example

```
<!DOCTYPE html>
<html>
<head>
<title>JavaScript</title>

</head>
<body>

<h2 id="tochange">Extreme Sports</h2>

<input type="text" id="new">
```

199

The <noscript> Tag

- It is necessary to do two things when using JavaScript
  - **Hide the script in HTML comments**
  - **Provide alternative rendering instructions**
- A **browser that does not process JavaScript** (or the **<script>** tag) will not be able to do anything more than ignore the information that it does not understand. So you just put those commands in HTML comments, thus:

```
<script language="JavaScript1.1">
<!- - Use HTML comments to hide the code
// JavaScript commands go here...
// ... this line ends the hiding here - ->
</script>
```

- The second approach is to display some other content that does not rely on scripts to be displayed by the browser.

```
<noscript>
Your browser does not seem to support JavaScript!<br/>
(Perhaps you need to enable it?)
</noscript>
```

- It is becoming less and less necessary to do either of these options but it is still a good practice

200

Remarks

- JavaScript rarely fails noisily.
  - the **only clue** that you have that it is not correctly written is that **it fails to do anything**.
  - You need **to use the reporting function** of your browser.
    - In **Internet Explorer**, this is indicated by a little error flag in the bottom left of the status bar. Double click to access it.
    - In **Firefox**, you need to access the **error console** via the menu.

201

JavaScript and events

- Normally added to handle events
- Event handlers can be defined using HTML5 tag attributes or within your JavaScript code.

```
<!DOCTYPE html>
<html>
<head>
<title>JavaScript</title>
<style>
body { background-color: #ffeecc; font-family: arial, sans-
serif }
</style>
</head>
<body>

<img src="dog.png" >

<img src="cat.png" >

<img src="bird.png" >

</body>
</html>
```

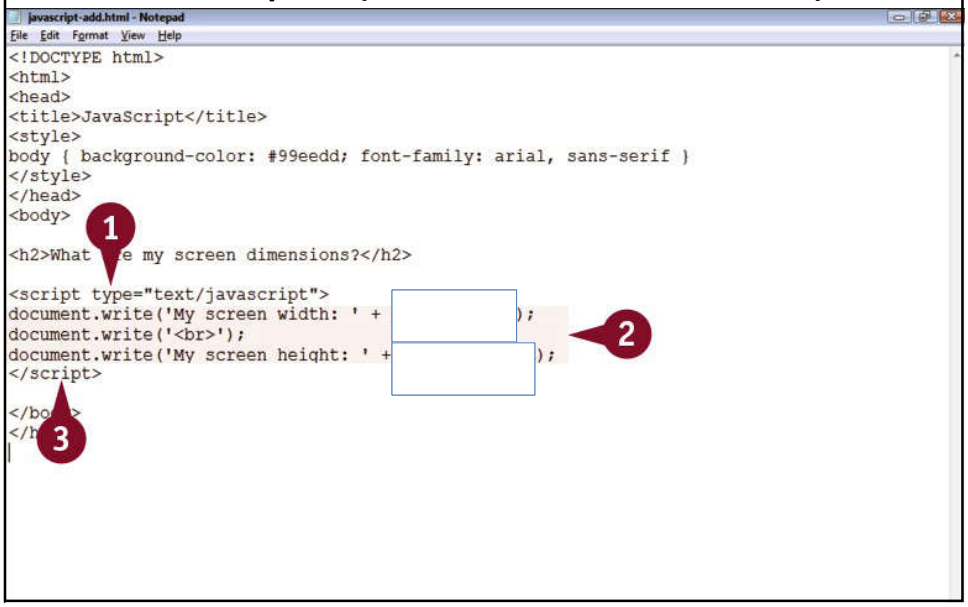
202

Remarks – cont.

- **You can't put quotes inside quotes**.
  - When it is necessary, **use a single quote** inside the double quotes.
    - **<input type="button" onClick="document.src = 'new_page.html';">**
- All statements must **end** with a semicolon (;).
- In JavaScript, **comments** are usually contained after // and inside /* */.
- Names of things in JavaScript **cannot start** with numbers or other special characters (although they can start with underscores, _, if you so wish).
  - JavaScript names **cannot contain spaces**
- JavaScript is also **not type-sensitive**.
  - JavaScript converts values by itself based on the target type or chooses the most appropriate type for a literal provided by the programmer.

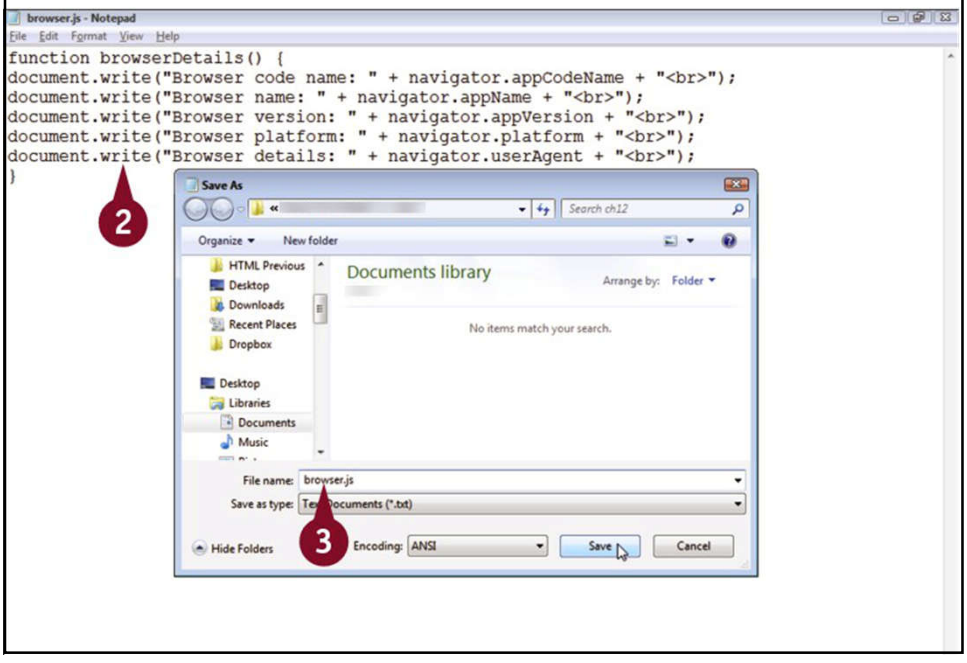
203

Example (screen dimensions)



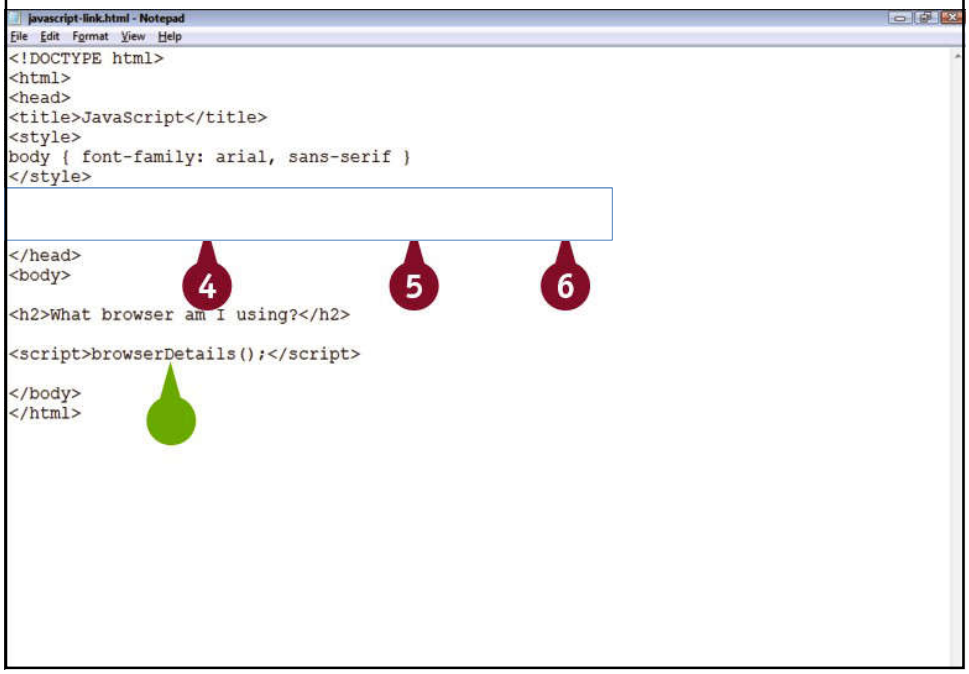
204

Link to external JS file



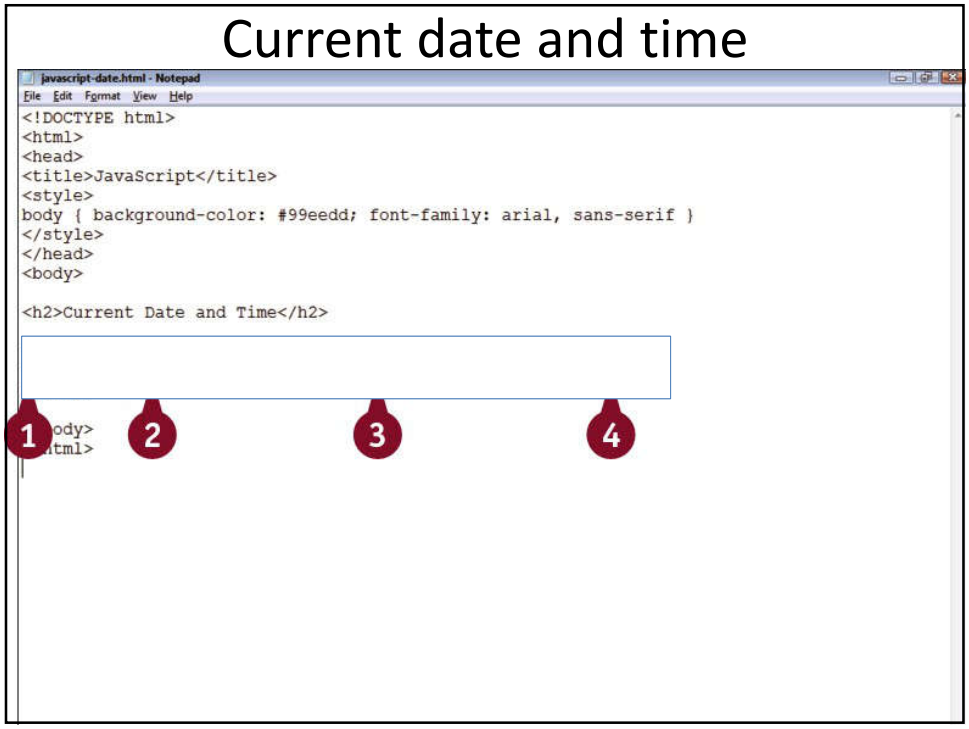
205

Link to external JS – cont.



206

Current date and time



207

Alert Message Box

javascript-alert.html - Notepad

File Edit Format View Help

```
<!DOCTYPE html>
<html>
<head>
<title>JavaScript</title>
<link rel="stylesheet" type="text/css" href="styles.css">
</head>

<section id="main-content">

<header>
<img src="mbadve e.png" alt="M and B Adventure Travel">
<p>Extreme vacations are our specialty!</p>
</header>

<nav>
<ul>
<li><a href="index.html">Home</a></li>
<li><a href="dest.html">Destinations</a></li>
<li><a href="tips.html">Travel Tips</a></li>
<li><a href="contact.html">Contact Us</a></li>
</ul>
</nav>

<br style="clear: both">

<p>For more than 10 years, M and B 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 your hotel bookings,
helicopter tours, and shark-cage rentals, either online or in person.</p>
```

1

208

Pop-up window

javascript-popup.html - Notepad

File Edit Format View Help

```
<nav>
<ul>
<li><a href="index.html">Home</a></li>
<li><a href="dest.html">Destinations</a></li>
<li><a href="tips.html">Travel Tips</a></li>
<li><a href="contact.html">Contact Us</a></li>
</ul>
</nav>

<br style="clear: both">

<p>

</p>

<p>For more than 10 years, M and B 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 your hotel bookings,
helicopter tours, and shark-cage rentals, either online or in person.</p>

<section id="example-images">
<img src="shark.jpg">
<img src="parachute.jpg">
<img src="bike.jpg">
</section>

<footer>
<small>Copyright &copy; 2011 M and B Adventure Travel. All rights reserved.</small>
</footer>
```

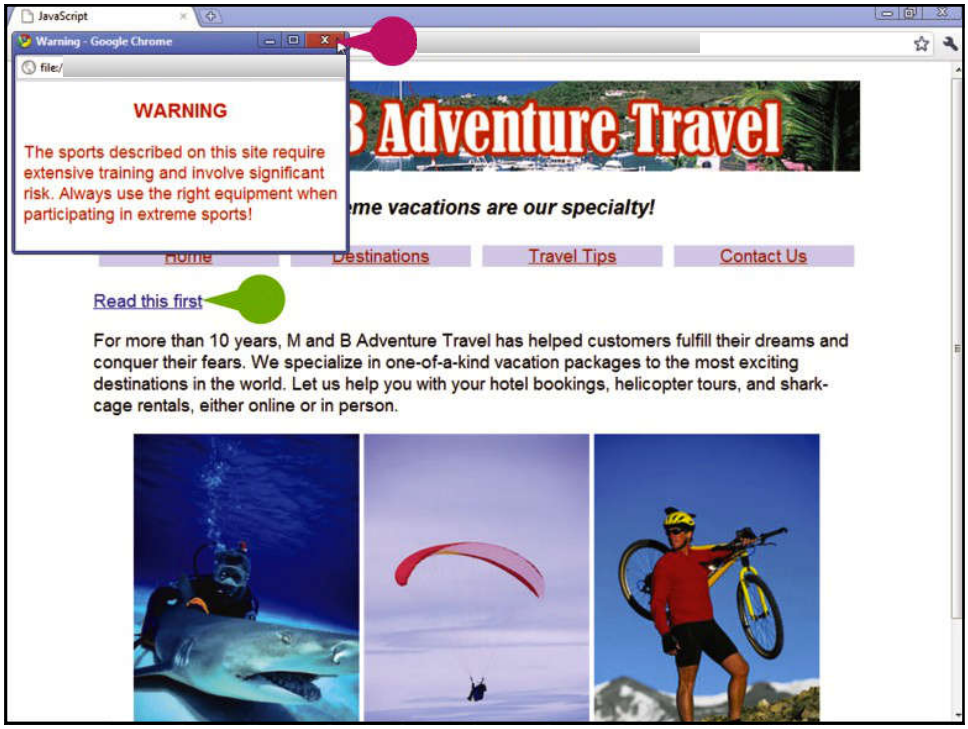
5

6

This **disables the normal link behavior** so you can use the tag for a pop-up window.

You can add the top=val1, left=val2 to position your window to the desired location

209



210

Create an Image Rollover Effect

- When the user positions the mouse pointer over the image on the web page, the image is replaced with a different one.
- When the user moves the mouse pointer off the image, the original image returns.
- Example: Images displayed in black and white, when mouse is over an image it is colored

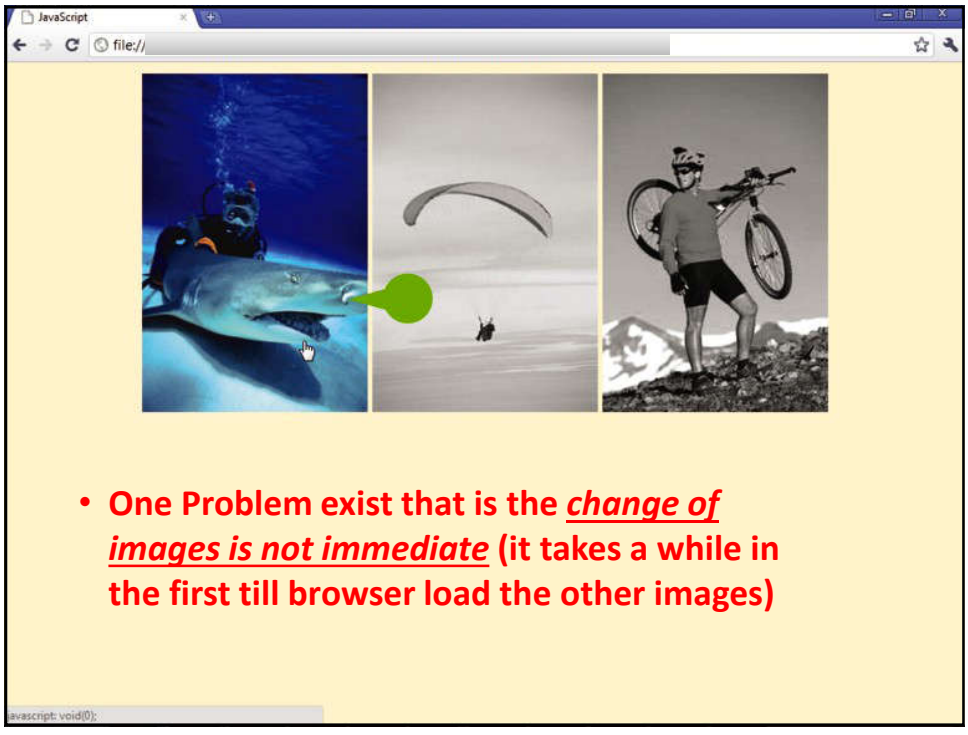
211

javascript-image.html - Notepad

File Edit Format View Help

```
<!DOCTYPE html>
<html>
<head>
<title>JavaScript</title>
<style>
body { background-color: #ffeecc; font-family: arial, sans-serif }
section#example-images { text-align: center; margin: 10px }
</style>
</head>
<body>

<section id="example-images">

<img src="parachute_bw.jpg">

<img src="bike_bw.jpg">

</section>

</body>
</html>
```

212



213

How can I make my rollover effects more immediate?

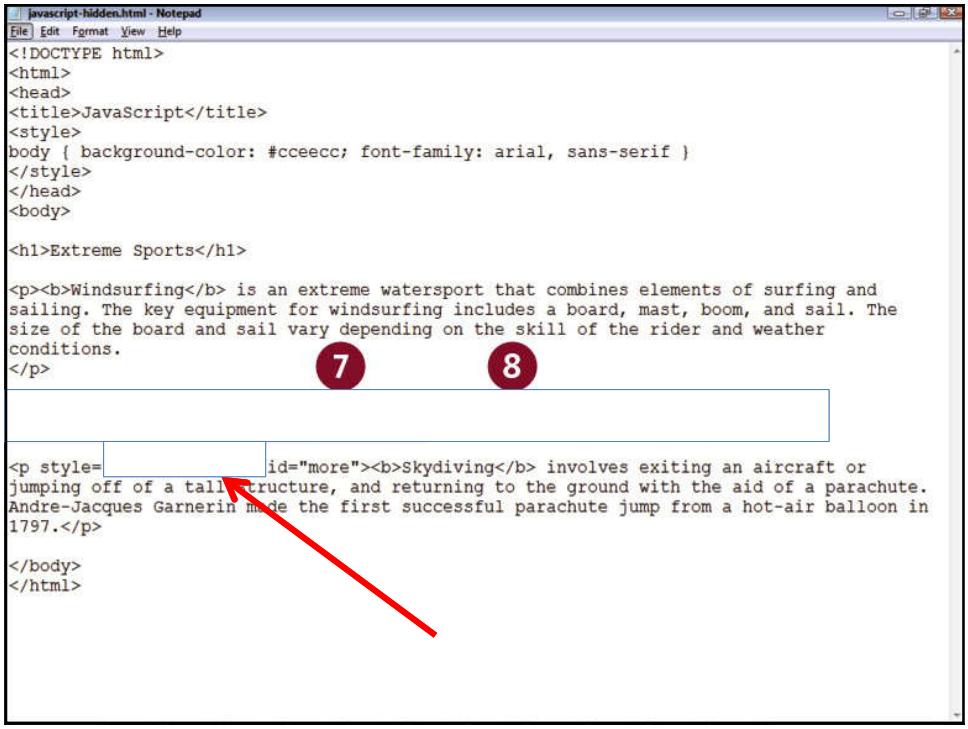
- You can instruct the browser to preload the rollover image using JavaScript.
- That way, the browser does not have to download the rollover image when the mouse pointer rolls over the original image.
- You can preload an image by adding the following script inside the <head> tag of your HTML:
- <script type="text/javascript">?=new Image(h,w); ?.src="imagepath"</script>
  - Replace *?* with an identifier for the image,
  - Replace *h,w* with height and width values for the image, and
  - Replace *imagepath* with the path to the image.

214

Show a Hidden Element

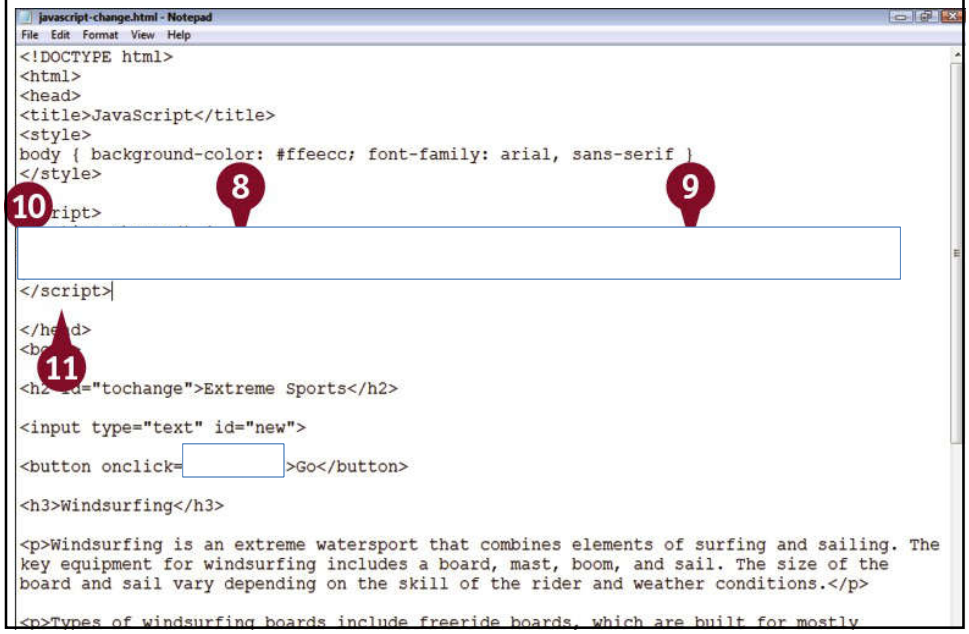
- You can hide the element by setting its CSS display property to none.
- With JavaScript, you can change that property to block or inline, which exposes the element on the page.
- The visibility property is **similar to the display** property.
- You can set visibility to hidden to hide content on your page. **However, that content, though invisible, still takes up space on the page.**
- When the display property is set to none, the affected content does not take up space on the page.

215

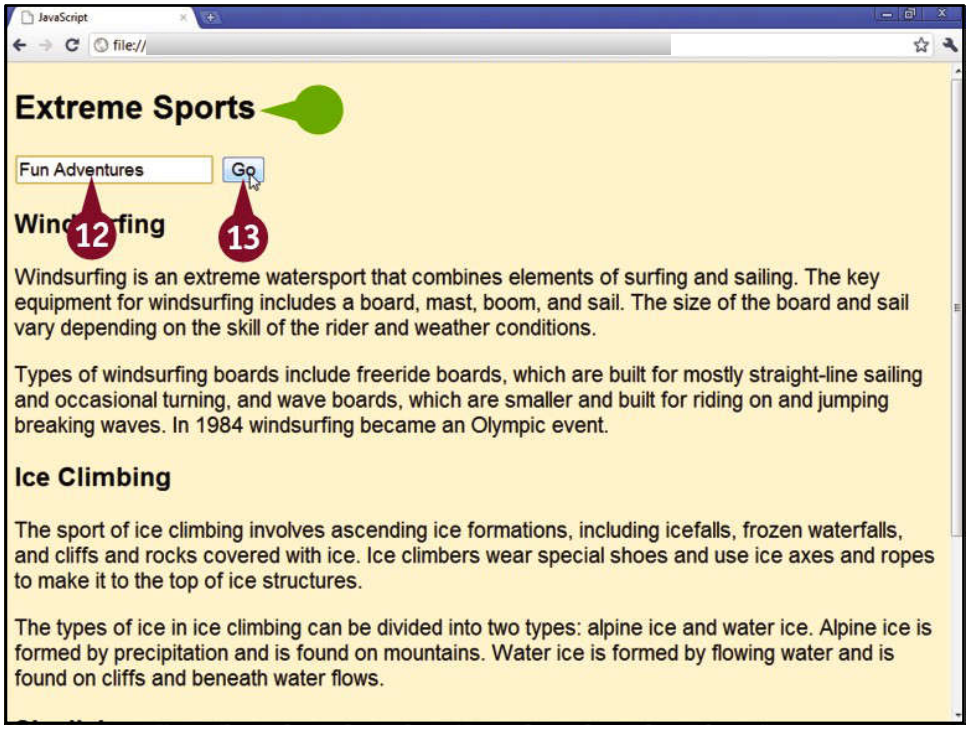


216

Change Page Content



217



218

Setting debugging tools

- Check your php version using phpinfo()
- Download the prober Xdebug version
  - https://odan.github.io/2020/12/03/xampp-xdebug-setup-php8.html
- Install xdebug and configure php.ini
  - Move the downloaded dll file to: C:\xampp\php\ext
  - Rename the dll file to: php_xdebug.dll
  - Open the file C:\xampp\php\php.ini with Notepad++
  - Disable output buffering: output_buffering = Off
  - Scroll down to the [XDebug] section (or create it) and copy/paste these lines:

```
[XDebug]
zend_extension=xdebug
xdebug.mode=debug
xdebug.start_with_request=trigger
```

  - Restart Apache

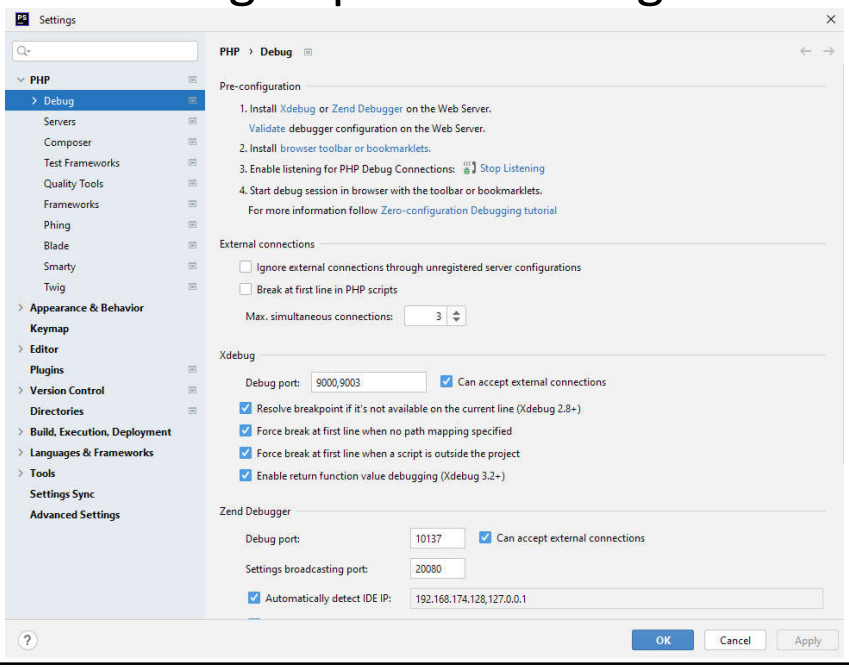
219

Setting debugging tools – cont.

- JavaScript debugging only work with chrome, so we will use chrome.
- Download xdebug helper browser extension for PhpStorm
  - https://www.jetbrains.com/help/phpstorm/browser-debugging-extensions.html
- Now we need to configure PhpStorm

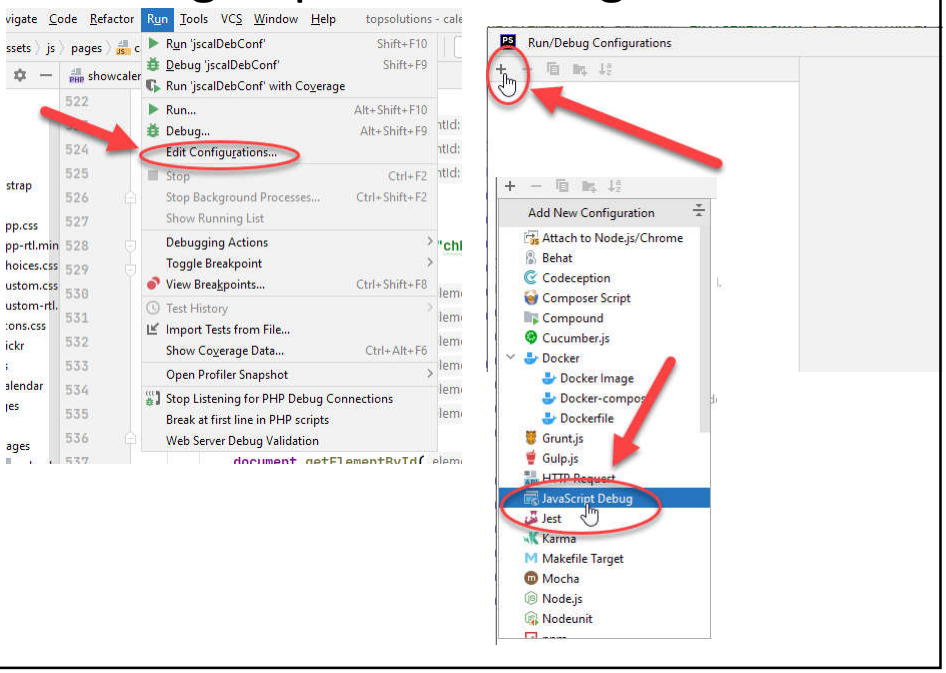
220

Xdebug PhpStorm configure



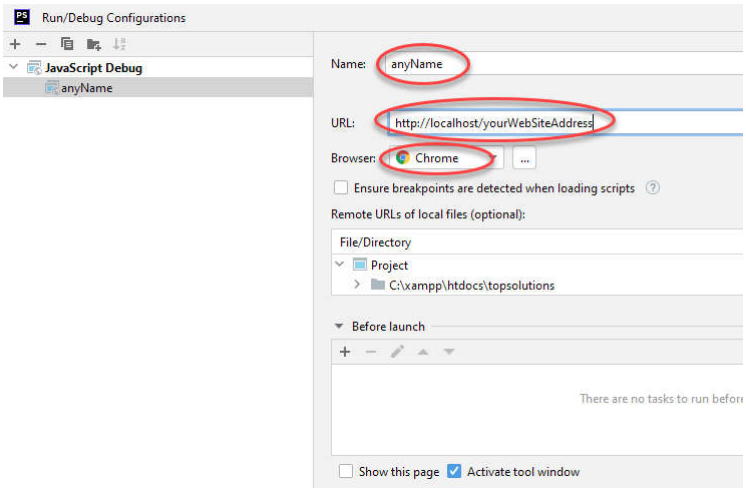
221

Xdebug PhpStorm configure – cont.



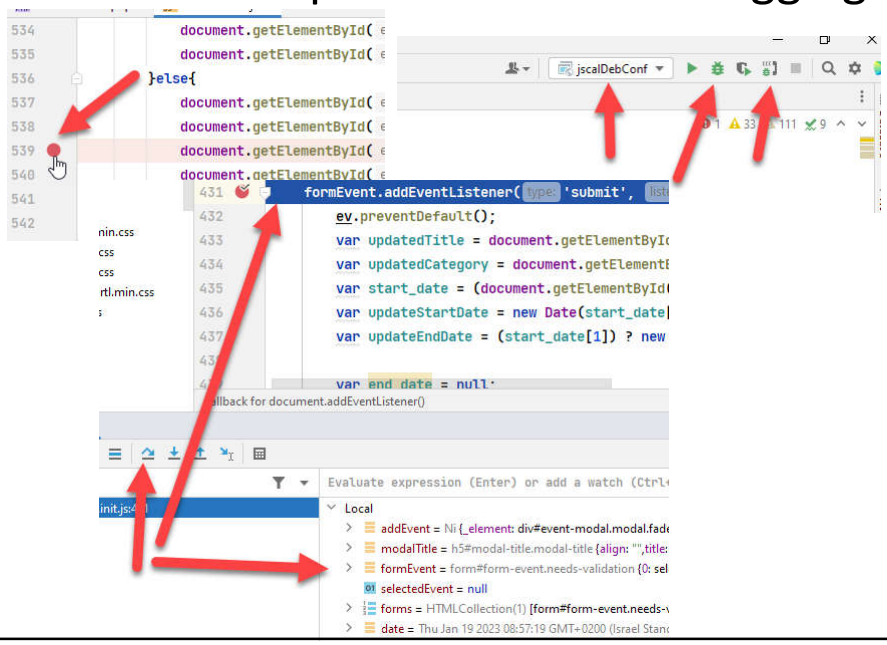
222

Xdebug PhpStorm configure – cont.



223

Set a break point and start debugging



224

Variables and Types

- In JavaScript, there are only **four data types** that you really need to be concerned with:
  - null, undefined—nothing, empty, and undefined (these are not the same)
  - Numbers—Integers or floating point numbers
  - Boolean values—true or false
  - Character strings—"a" or "a string"

225

Variable Types – cont.

- A variable is **defined** either **implicitly** or **explicitly**
  - Implicitly
    - JavaScript engine will try to determine the data types from the values that it is passed
      - Both of the following are string types
        - » myString = "a string"; myString = "a string equals " + 42;
      - Both of the following are number types
        - » myString = 42 + 42;
    - A variable is usually assigned the data type of the first piece of data to be stored in it.
    - **Do not try to mix literals or variables**, it is better to ensure that your code does not assume anything.
  - Explicitly with the var keyword
    - var myString = ""; // an empty string

226

Arrays

- An array **can contain any data type** that can be defined in JavaScript,
  - including the objects that make up an HTML page.
- They can also be mixed, meaning that different types of data can be stored next to each other in the array.
  - But you have to remember which array element holds which data type!!

227

Arrays – cont.

- An array can be declared by
  - Either filling it with values or
  - Reserving a certain number of items in advance.
- Example
  - var myArray=new Array(10);
  - An array is indexed zero-based.
    - myArray[0] = "banana"; // This is the first element
  - Array can be declared and initialized at the same time
    - var myFruits = new Array ("apple", "orange", "banana");
  - Multidimensional arrays are also possible
    - var myTable[10][10]; // 10 x 10 array
    - myTable[0][1] = 3;
    - myTable[1][1] = "three"; // [0][1] still contains 3

228

Operators

- Operators can be categorized depending on their function
  - Assignment—Assigns a value to a variable
    - x = "42"; // Put a string literal in x
    - x /= 2; // x = x / 2 (division)
    - x -= 7; // x = x - 7
  - Comparison—Compares one or more values
    - == , != , >, >=, <, <=
  - Mathematical—Performs simple mathematical operations
    - **+**, , **/**, and *****.

229

Operators – cont.

- Logical—For use with Boolean (true and false) values
  - &&, ||, and !
    - !(x = = y) || !(a = = b);
- String—Specifically for use with strings
  - +
    - **"a " + "b"; // returns "a b"**
- The new and delete operators
  - var myAddress = new String ("some text");
  - var myArrayOfNumbers = new Array (1, 2, 3, 4, 5);
  - delete (myAddress);
  - delete myArrayOfNumbers[0]; // empty array element 0 (the first one) and set its value to undefined.

230

Operators – cont.

- The conditional operator, ?
  - condition ? if_true : if_false
    - (y == 3) ? x = 0 : x = 1;
- The "***this*** " operator is special in that it **returns the object that is currently being referenced**. This will usually be in the context of an event handler coupled with an HTML object.
  - **<form action="submit.php" onClick="validate_form(this);" method="post">**
- The operator "*void*" is used when something must be evaluated, but no value is returned or processed

231

Functions

- Functions are stored either
  - In the head of the HTML document
  - In a separate .js file
- The reason for this is to be sure that all the functions are available when the HTML document objects are instantiated (loaded and rendered) by the browser.
- There is a set of useful built-in functions, but the vast majority of work in JavaScript is done via the methods associated with built-in objects and not standalone functions.
- Functions can be called from anywhere in the HTML document by using the *"script"* tag or by assigning them to event handlers.

232

Calling Functions

- All the parameters are passed by value
  - myFunction (myString, myNumber); // myString and myNumber passed by value
- When an object is passed (such as an array), it is passed by reference

Built-In Functions:

- Conversion functions "*Number*" and "*String*" (convert string to number and number to string respectively)
  - myDate = new Date (); // make a new date object, set to Today
  - myDateString = String (myDate); // convert to a string
- The "eval" function allows you to evaluate a string as if it were a piece of JavaScript code.
  - myNumber = "(3>1)?(1 + 3):(3-1)"; // myNumber contains "(3>1)?(1 + 3):(3-1)"
  - myNumber = eval (myNumber); // myNumber now contains 4
- The "escape" and "unescape" functions allow you to encode a URL by escaping the special characters.
  - mySearchString = escape("first second third");
    // mySearchString contains "first%20second%20third

233

User-Defined Functions

- A function (name and parameters) **should not be identical to another function.**
- To define a function
  - **function myFunction (myParameter1, myParameter2) {**
    // My code statements
    }
- A function **can also return a value**, which can be used for ongoing processing. An example of a function that returns a value is as follows:
  - function myFunction (myParameter1, myParameter2) {
    // My code statements
    return myReturnValue;
    }
  - The type of myReturnValue can be anything that JavaScript supports, even objects.
- As long as the function has been defined before it is called, it can be called from inside another function.
- A function can also be defined as recursive; in other words, it can call itself.
  - **function my_fact (n) {**
    if (n <= 1) return 1; // is n less than or equal to 1?
    else return (n * my_fact(n-1)); // if not call itself
    }

234

Using Variable Arguments

- It is perfectly possible to define a function that does not have exactly one named parameter for each piece of information that it needs to process. Instead it has only one named parameter, and the rest are implied and can be accessed from the function code implicitly.
- Parameters that are passed and not named can be accessed through the arguments property of the function.
  - **function myFunction (parameter_0) {**
    return parameter_0 + this.arguments[1];
    }
  - This example assumes that the function will be passed more than one argument, otherwise arguments[1] will be undefined.
- You can obtain the number of arguments by using the length property of the arguments collection
  - **for (var arg = 0; arg < this.arguments.length; arg++) {**
    // process this.arguments[arg] here
    }

235

Strings and String Processing

- Two types of string
  - **Literals**
    - var myString_1 = "My String";
  - **Objects**
    - var myString_2 = new String ("My String");
- you use “typeof” to determine what kind of type these are
    - typeof myString_1; // returns string
    - typeof myString_2; // returns object
- In reality, because JavaScript converts between a value and an object, there is no difference between the two when it comes to practical matters of their application.

236

Example

```
<script type=“text/javascript”>
    var outText = new String("Bold");
    document.write(outText.bold());
    var evalString = "outText.italics()";
    evalString = "document.write(" + evalString + ");";
    eval (evalString);
</script>
```

.

237

Conditional Processing

- if .. else

```
if ( a = = "my string " + myNumberAsString ) {
    // do this if a is true
} else {
    // do this if a is false
}
```

- switch

```
switch ( colorNumber ) {  // the condition can be any testable data type
case 0 :
    colorString = "#ff0000";
    break;
case 1 :
    colorString = "#0000ff";
    break;
default:
    colorString = "#ffffff";
}
```

238

Loops

- The for loop
  - for (var counter = 0; counter < total_iterations; counter++) {
    // statements to execute
    }
- The do . . . while loop

```
var counter = 0;
do {
    // statements to execute
    counter++;
} while ( counter < total_iterations );
```

- The while loop

```
var counter = 0;
while ( counter < total_iterations ){
    // statements to execute
    counter++;
}
```

239

The break and continue Keywords

```
var counter = 0;
while ( counter < total_iterations ){
    // statements to execute
    counter++;
    if ( counter * 2 > target_value )
        break; // leave the loop
}
```

- /***
  **/

```
for ( var counter = 0; counter < total_iterations; counter++ ) {
    if ( counter % 2 = = 0 )
        continue; // Only execute the following on odd
iterations
    // statements to execute
}
```

240

The JavaScript Web Function Library

- JavaScript has interfaces to HTML objects and a collection of functions that are useful in Web programming based around the document container mode.
- These are often mirrors of the objects contained within the DOM.

241

Events

- Each event that can be attached with any HTML object (usually the body, img, a, and form tags) has the general form:
  - <tag onEvent="code">
- The code can be anything from a call to a function (recommended) to a single line of JavaScript.

```
<script>
function event_handler() {
    // function code here
}
</script>
<body onKeyPress="event_handler();">
```

- **In this example there are no access to event properties, to have such access:**

```
<body>
<script>
function event_handler(e) {
// function has access to event properties
}
document.onKeyPress = event_handler(event);
</script>
</body>
```

242

Events – cont.

- **onAbort** This event occurs when the user interrupts the loading of a page. The OnAbort event **should only be trapped once** for the HTML document as a whole. For example it is used with img tags.
  - **<img src="myimage.gif" onAbort="if (!confirm('Stop loading document?')) { history.go(0); }">**
  - This snippet will ask if the users want to stop loading the document. If they click No on the confirm dialog box that appears, the document will be reloaded.
- **onBlur** This event occurs when an element in a page or when the window containing a page **loses focus**—that is, the user moves away from it by using the Tab button. It only applies to interactive form elements and the window object.

243

Events – cont.

- **onChange** This event occurs when the content of an interactive form element changes. This only applies to those elements that have selectable or user entered values.
  - For example, the Google Suggest service traps this event in order to populate the list of suggested search terms. This is used as follows:
  - **<input type="text" name="keywords" onChange="updateKeyList(this.value);">**
- **onClick** This event occurs when an object on a document is clicked—that is, a form button (including checkboxes and radio buttons) or a link. Because a click is a combination of a mouseDown and mouseUp event, clicks can be trapped separately with their own handlers. An example of handling the onClick event might be:
  - **<a onClick="this.myForm.submit();"> <img src="submit.gif" alt="Submit">**

244

Events – cont.

- **onDblClick** This event is the same as the onClick event, except that it is triggered when the user double-clicks a control.
- **onFocus** This event is the opposite of onBlur, that is, it is triggered when the focus moves to a control on a from.
- **onKeyDown**, **onKeyPress**, and **onKeyUp** These three events are triggered in response to key presses, usually trapped by a document object. However, they are also listed for event handlers for links, images, and text area input controls. They use a number of so-called event properties.

245

Events – cont.

- **onLoad and onUnload**
  - The onLoad event is triggered when a document has finished loading.
  - It is usually handled in the body tag, thus: <body onLoad="my_function();”>
    - Ex: When page is loaded into frame.
      - <body onLoad="my_function();” onUnload="my_cleanup_function();">
- **onMouseDown** and **onMouseUp**
  - The combination of these two events causes an onMouseClick event to be triggered. The onMouseUp event can be trapped as well as the onMouseClick event for a given object.
  - Handling these events is useful in allowing drag-and-drop style positioning of elements on a page.

246

Events – cont.

- **onMouseMove**
  - This event is triggered when the mouse is moved over a browser window.
- **onMouseOut** and **onMouseOver**
  - The onMouseOut event is triggered when the mouse is moved off an image area (that is, a client side imagemap) or link, and
  - an **onMouseOver** event is triggered when the mouse first enters the area or hovers over a link.
    - **<img src="out.bmp” onMouseOver="this.src=’in.bmp’;” onMouseOut="this.src=’out.bmp’;">**

247

Example

```
<html>
<head>
<script type="text/javascript">
function bigImg(x)
{
    [          ]
}
function normalImg(x)
{
    [          ]
}
</script>
</head>
<body>
[                                                    ]

<p>The function bigImg() is triggered when the user moves the mouse pointer over the image. This function enlarges the image.</p>
<p>The function normalImg() is triggered when the mouse pointer is moved out of the image. That function changes the height and width of the image back to normal.</p>
</body>
</html>
```

248

Events – cont.

- **onReset** and **onSubmit** The onReset event is triggered when the user clicks the Reset button of a form, and the onSubmit event is triggered when the user clicks the Submit button. These events are handled in the form tag:
  - <form method="post" onSubmit="validate_form();" onReset="reset_form();">
- **onSelect** This event is triggered when the content of a text field is selected. This is useful in providing dynamic forms and support to users when they fill them out.
  - For example, a function could be written to populate a container of city names, so the users can choose their city from the list.

249

Example

```
<html>
<head>
<script type="text/javascript">
function showMsg()
{
      alert("You have selected some of the text!");
}
</script>
</head>
<body>

Select text: <input type="text" value="Hello world!"
onselect="showMsg()" />

</body>
</html>
```

250

Event object – cont.

Methods

Method	Description
initEvent()	Specifies the event type, whether or not the event can bubble, whether or not the event's default action can be prevented
preventDefault()	To cancel the event if it is cancelable, meaning that any default action normally taken by the implementation as a result of the event will not occur
stopPropagation()	To prevent further propagation of an event during event flow

EventTarget Object

Methods

Method	Description
addEventListener()	Allows the registration of event listeners on the event target (IE8 = attachEvent())
dispatchEvent()	Allows to send the event to the subscribed event listeners (IE8 = fireEvent())
removeEventListener()	Allows the removal of event listeners on the event target (IE8 = detachEvent())

251

Event properties – cont.

Event Object Property	Description
SrcElement	The element that fired the event
type	Type of event
returnValue	Determines whether the event is cancelled
clientX	Mouse pointer X coordinate relative to window
clientY	Mouse pointer Y coordinate relative to window
offsetX	Mouse pointer X coordinate relative to element that fired the event
offsetY	Mouse pointer Y coordinate relative to element that fired the event
button	Any mouse buttons that are pressed
altKey	True if the alt key was also pressed
ctrlKey	True if the ctrl key was also pressed
shiftKey	True if the shift key was also pressed
keyCode	Returns UniCode value of key pressed

252

Event properties

- The **cancelable** property:
  - Returns whether or not an event can have its default action prevented
- The **currentTarget** property
  - Returns the element whose event listeners triggered the event
- The timeStamp property
  - Returns the time (in milliseconds relative to the epoch) at which the event was created

253

Example

```
<html>
<head>

<script type="text/javascript">
function getEventTrigger(event)
  {
  var x=event.currentTarget;
  alert("The id of the triggered element: "
  + x.id);
  }
</script>
</head>

<body >

<p id="p1" onmousedown="getEventTrigger(event)">
Click on this paragraph. An alert box will
show which element triggered the event.</p>

</body>
</html>
```

254

Example

```
<html>
<head>
<script type="text/javascript">
function show_coords(event)
  {
  var x=event.clientX
  var y=event.clientY
  alert("X coords: " + x + ", Y coords: " + y)
  }
</script>
</head>

<body onmousedown="show_coords(event)">

<p>Click in the document. An alert box will alert
the x and y coordinates of the mouse pointer.</p>

</body>
</html>
```

255

Example

```
<html>
<head>
<script type="text/javascript">
function showTimestamp(event)
  {
  var minutes = 1000*60
  var x=event.timeStamp;
  alert(x/minutes)
  }
</script>
</head>

<body onmousedown="showTimestamp(event)">

<p>Click in the document. An alert
box will alert the timestamp.</p>

</body>
</html>
```

256

The document Object

- The **document** Object
  - Has some handy methods
    - document.open();
    - document.write(HTML to write)
    - document.close();
  - **<script>document.close(); document.open(); document.write();</script>**
    - The **.open** and **.close** methods are superfluous for cross-platform browser development because not all browsers react in the same way.
    - In most cases calls to these methods are additive.
  - The document object has the following arrays:
    - forms—An array of all the forms
    - images—An array of all the images
    - links—An array of all the link locations in the document

257

The document object – cont.

- In addition, it contains the following useful properties:
  - domain—The domain hosting the document
  - cookie—Any cookies associated with it
  - width and height—The width and height, in pixels
  - title—The title, as displayed in the title bar
  - URL—The full URL of the document
- The arrays are zero-based like all arrays.
- In addition it is possible to identify forms by their name,
  - provided that forms have been defined with a name="myForm" attribute
  - document.forms["MyForm"]
- Generally, the properties should only be read and not modified
  - Others can be modified:
    - **<input type="button" value="Change Image" onClick="document.buttons[0].src='new.gif';">**
- Most objects in the document model also have a **visible** property.
  - Can be used for making controls in a form available (or not) depending on the options already chosen.
    - Ex: make an entire ** ** appear and disappear, like a message or pop-up box.

258

The window Object

- Has many properties and is the top-level object in hierarchy.
- There may be many windows at the same level
  - frames inside a frameset.
- The **frames** array property gives access to any frame that might be contained within the window.
  - The **length** property indicates the number of frames for and the **parent** property to traverse to parent.
- The **location** property contains the address (URL) of the currently loaded document in the window (or frame).
- The **status** property contains a value that is displayed in the status bar (usually at the bottom of the screen) and this can be updated to change the message being displayed.

259

The window Object – cont.

- The **back** and **forward** methods load the previous and next URLs in the history list associated with the window.
- The **close** method can be used to close the window.
- The **home** method loads the URL that is specified in the user's preferences.
  - **<input type="button" value="Back" onClick="window.back();">**
  - **<input type="button" value="Home" onClick="window.home();">**
  - **<input type="button" value="Fwd" onClick="window.forward();">**
- The **print** method opens the standard operating system's Print dialog box and prints the window contents.
- The **stop** method halts the current download, which is useful if something might take a long time to complete and you would like to allow the users to stop the process.

260

Programmatic HTML Generation

- Using the String object, it is possible to convert a string literal to a piece of HTML code.
- Usually used with the document.write method to generate the inline HTML.
  - however, the actual HTML is not visible when the user selects View Source, only the JavaScript that has generated the HTML will be
- Ex:
  - var myString = "anchor text";
  - document.write(myString.anchor("anchor_name"));
  - This example is the equivalent of the following HTML code:
    - anchor text
  - By the same token, the link method is used as follows:
    - var myString = "url text";
    - document.write(myString.link("url"));
  - This is rendered by the browser as if the HTML had been:
    - url text

261

Programmatic HTML Generation

- There are also three methods used to create text of different font sizes:
  - big—Makes text that is bigger than normal
  - small—Makes text that is smaller than normal
  - fontsize (n)—Makes text of a specific font size
  - Ex:
    - document.write("Big Text".big());
    - document.write("Small Text".small());
    - document.write("Really Big Text".fontsize(72));
- Of course, string variables could have been used in these examples instead of literals.
- Also there are some methods used to create fonts of different properties:
  - fixed—Chooses a fixed font (like Courier)
  - fontcolor (color)—Specifies the color of the font, where color is either #rrggbb or color name, following the Web standards

262

Useful Top-Level Methods and Properties

- Attached to top-level objects (such as the **window** object)
  - These methods can be called without the **window** reference, as it is assumed.
- Message Boxes
  - JavaScript offers three possibilities—**alert**, **confirm**, and **prompt**.
- The **alert** method displays a box alerting the user, with a custom string and a single OK button. For example:
  - alert("Alert! You pressed the alert button...");

263

Useful Top-Level Methods and Properties

- The confirm method displays a confirmation box from which the users can select OK or Cancel. If they click OK, confirm returns true; otherwise, it returns false. For example:
  - theResult = confirm("Do you really want to do this?");
- The prompt method displays a box in which the users can type some text. If they then click OK, the text is returned; otherwise, null is returned. For example:
  - theResult = prompt("Please enter some text...");
- If more complex interaction is required, you can use the window.open method to open a new window containing a form from which information can be obtained. Or, a custom div can be displayed containing the relevant HTML.

264

265

Interval Processing

- JavaScript provides the possibility to execute a function after a certain length of time has elapsed in one of two ways:
  - Once every x milliseconds (interval)
    - activated by using the **setInterval** method and
    - stopped using the **clearInterval** method.
  - After x milliseconds and never again (timeout)
    - Activated by using the **setTimeout** method and
    - Stopped using the **clearTimeout** method.
- The setInterval method can be used in one of two ways: with an expression or with a function plus an argument list.
  - For example, the following two usages are identical:
    - var timerID=setInterval("my_function (1, 2, 3, 4);", 25)
    - var timerID=setInterval(my_function, 25, 1, 2, 3, 4)
  - To cancel the interval timer, use the clearInterval method with the timer identifier:
    - clearInterval (timerID);
- setTimeout and clearTimeout arguments are the same as the setInterval and clearInterval

266

Accessing Cookies

- Cookies are small pieces of information that are stored as name, value, expiration tuples.
- The expiration date is optional
- The exact format of the cookie data is up to the programmer to define.
- JavaScript provides access to the cookies through the **document.cookie** property.
- cookies need to be encoded (using the **escape** and **unescape** functions) to render unsupported characters (such as spaces) as data using their hexadecimal equivalents.

267

268

Cookies – example

- For example, a space is replaced by **%20** with the call to
  - This is then replaced by a space value when given to the **unescape** function.
- So, to set a cookie, it is necessary to build a cookie string (**document.cookie**) with three parts—
  - the name (not encoded),
  - the value (encoded),
  - and the expiry date (not encoded and also **optional**).
- The cookie string can then be set in the document properties, but is stored by the browser. The complete solution is as follows:
  - **document.cookie = name + ”=“ + escape(value) + ”; expires=" + expiry_date; + ";"**
- To get the expiry date, you simply convert the **Date** object value to a UTC (Coordinated Universal Time) string using the **.toUTCString()** of the **Date** object. For example:
  - **now = new Date();**
  - **expiry_date = now.toUTCString();**
- If expiry date not required, the cookie becomes a pair with just the name and encoded value.

269

Cookies Example

```
function createCookie(name,value,days) {
        if (days) {
                var date = new Date();
                date.setTime(date.getTime()+(days*24*60*60*1000)); //days converted to milliseconds
                var expires = "; expires="+date.toUTCString();
        }
        else
                var expires = ”;”;

}

function readCookie(name) {
        var nameEQ = name + "=";
        var ca =
        for(var i=0;i < ca.length;i++) {
                var c = ca[i];
                if (c.indexOf(nameEQ) == 0) return unescape(c.substring(nameEQ.length,c.length));
        }
        return null;
}

function eraseCookie(name) {
        createCookie(name,"",-1);
}
```

270

Cookies Example

```
function setCookie(cname, cvalue, exdays) {
  var d = new Date();
  d.setTime(d.getTime() + (exdays*24*60*60*1000));
  var expires = "expires="+d.toUTCString();
  document.cookie = cname + "=" + cvalue + "; " + expires;
}

function getCookie(cname) {
  var name = cname + "=";
  var ca = document.cookie.split(';');
  for(var i=0; i<ca.length; i++) {
    var c = ca[i];
    while (c.charAt(0)==' ') c = c.substring(1);
    if (c.indexOf(name) == 0) return c.substring(name.length, c.length);
  }
  return "";
}

function deleteCookie(name) {
        setCookie(name,"",-1);
}
```

271

Using Cookie example

```
function checkCookie() {
    var user = getCookie("username");
    if (user != "") {
        alert("Welcome again " + user);
    } else {
        user = prompt("Please enter your name:", "");
        if (user != "" && user != null) {
            setCookie("username", user, 365);
        }
    }
}
```

272

Query Strings

- Some URLs that are the result of a form submission contain data that has been submitted and that might be useful to extract from the URL
- The location.search property accesses the portion of the URL after the ? (question mark), which is known as the query string.
- The value returned includes the **?**, so the first operation is to discard it.
- Using the **indexOf** method of the string object, it is possible to extract the values for use in the JavaScript code. This could also be true of a form that is passed back to the same document for processing, but the query string is likely to be more complex.
- For example, consider the following HTML code:
  - **<form>**
  - **<input type="text" value="" name="textfield">**
  - **<input type="Submit" value="Submit!">**
  - **</form>**
  - This submits the form back to the source page with the URL. If you had entered Banana in the text field, the URL would look something like this:
    - file:///test.html?textfield=Banana

273

Query string example

```
function querySt(Key) {
    if(window.location.search.length==0)
        return null;
    QSwithoutQuestion = window.location.search.substring(1);
    QSArray = QSwithoutQuestion.split("&");
    for (i=0;i<QSArray.length;i++) {
        NameValue = QSArray[i].split("=");
        if (NameValue[0] == Key) {
            return NameValue[1];
        }
    }
}
```

274

Client side storage

- Cookies
  - Up to 4KB
  - Originally to create states between client and server side
  - Accessible from server side and client side
  - Send to server with every HTTP request
  - Only stores Strings
  - Consists of a mandatory part which is the key-value pairs
  - Can have other optional parts
    - expires=date;
    - path=path;
      - Cookie is valid only in the defined path. If empty the cookie is valid for any directory (website structure)
    - domain=domain;
      - Cookie is valid only in the defined domain.
    - secure;
      - Cookie is valid or it is send only with https

275

Client side storage – cont.

- LocalStorage
  - Up to 10MB (depends on the browser)
  - Never expires unless the user explicitly deletes the data
  - Accessible from client side only
  - Accessible from any window or any tab
  - Plaintext only. Not secure
  - localStorage.setItem()
- SessionStorage
  - Up to 5MB
  - Expires when the tab is closed
  - Accessible from client side only
  - Accessible from the current tab (private to tab)
- Web SQL
- IndexedDB
- Offline Storage

276

277

Core JavaScript Objects

Number Object Properties

Property	Description
constructor	Returns the function that created the Number object's prototype
MAX_VALUE	Returns the largest number possible in JavaScript
MIN_VALUE	Returns the smallest number possible in JavaScript
NEGATIVE_INFINITY	Represents negative infinity (returned on overflow)
POSITIVE_INFINITY	Represents infinity (returned on overflow)
prototype	Allows you to add properties and methods to an object

Number Object Methods

Method	Description
toExponential(x)	Converts a number into an exponential notation
toFixed(x)	Formats a number with x numbers of digits after the decimal point
toPrecision(x)	Formats a number to x length
toString()	Converts a Number object to a string
valueOf()	Returns the primitive value of a Number object

278

String

String Object Properties

Property	Description
constructor	Returns the function that created the String object's prototype
length	Returns the length of a string
prototype	Allows you to add properties and methods to an object

279

String Object Methods

Method	Description
charAt()	Returns the character at the specified index
charCodeAt()	Returns the Unicode of the character at the specified index
concat()	Joins two or more strings, and returns a copy of the joined strings
fromCharCode()	Converts Unicode values to characters
indexOf()	Returns the position of the first found occurrence of a specified value in a string
lastIndexOf()	Returns the position of the last found occurrence of a specified value in a string
match()	Searches for a match between a regular expression and a string, and returns the matches
replace()	Searches for a match between a substring (or regular expression) and a string, and replaces the matched substring with a new substring
search()	Searches for a match between a regular expression and a string, and returns the position of the match
slice()	Extracts a part of a string and returns a new string
split()	Splits a string into an array of substrings
substr()	Extracts the characters from a string, beginning at a specified start position, and through the specified number of character
substring()	Extracts the characters from a string, between two specified indices
toLowerCase()	Converts a string to lowercase letters
toUpperCase()	Converts a string to uppercase letters
valueOf()	Returns the primitive value of a String object

280

String example

- string.charAt (n)
- string.slice (start, end)
- string.substr (start, length)
- string.split (separator, limit)
- var queryString = window.location.search.substr (1);
  - // everything except the ?
- var queryArray = queryString.split ("&");
  - // split by the & character
- string.concat(string_1, string_2, ... string_n)
  - var myString1 = "String1";
  - var myStringN = myString1.concat("2", "3");
- var firstThat = myString1.indexOf ("g1"); // returns 5
- var lastThat = myStringN.lastIndexOf ("2"); // returns 7
- var myString2 = myStringN.replace("123", "1234");
- string.toUpperCase()
- string.toLowerCase()

281

String HTML Wrapper Methods

The HTML wrapper methods return the string wrapped inside the appropriate HTML tag.

Method	Description
anchor()	Creates an anchor
big()	Displays a string using a big font
blink()	Displays a blinking string
bold()	Displays a string in bold
fixed()	Displays a string using a fixed-pitch font
fontcolor()	Displays a string using a specified color
fontsize()	Displays a string using a specified size
italics()	Displays a string in italic
link()	Displays a string as a hyperlink
small()	Displays a string using a small font
strike()	Displays a string with a strikethrough
sub()	Displays a string as subscript text
sup()	Displays a string as superscript text

282

Array Object Properties

Property	Description
constructor	Returns the function that created the Array object's prototype
length	Sets or returns the number of elements in an array
prototype	Allows you to add properties and methods to an object

Array Object Methods

Method	Description
concat()	Joins two or more arrays, and returns a copy of the joined arrays
join()	Joins all elements of an array into a string
pop()	Removes the last element of an array, and returns that element
push()	Adds new elements to the end of an array, and returns the new length
reverse()	Reverses the order of the elements in an array
shift()	Removes the first element of an array, and returns that element
slice()	Selects a part of an array, and returns the new array
sort()	Sorts the elements of an array
splice()	Adds/Removes elements from an array
toString()	Converts an array to a string, and returns the result
unshift()	Adds new elements to the beginning of an array, and returns the new length
valueOf()	Returns the primitive value of an array

283

Array Examples

- var myNumbers = new Array (42, 84, 12);
- var myStrings = new Array ("forty-two", "eighty-four", "twelve");
- var myNewArray = myNumbers.concat(myStrings);
- //[42, 84, 12, "forty-two", "eighty-four", "twelve"]
- var myString = myNewArray.join(", "); // default separator is comma
  - turns an array into a string, separated with a user-defined or default separator.
  - //"42, 84, 12, forty-two, eighty-four, twelve"
- var newArray = myNewArray.slice (0, 3);
  - // newArray contains [42, 84, 12, "forty-two"]
- array.splice (startAt, toRemove, toAdd1, toAdd2, ... toAddN)
  - In this definition, startAt gives the index at which to begin the splice. Similarly, toRemove gives the number of elements that should be removed
  - var newArray = new Array ( );
  - newArray.splice (0, 0, "1", "two", 3); //["1", "two", 3]
  - newArray.splice (1, 2, "twelve", "four"); //["1", "twelve", "four"]

284

Array Examples – cont.

- The **sort** method allows the programmer to provide a compare function. This enables the default alphabetical sorting behavior of JavaScript to be overridden.
- If provided, the compare function must take two parameters and return one of the following values:
  - -1 If the first parameter is less than the second
  - 0 If the two parameters are equal
  - 1 If the first parameter is greater than the second

```
function compare ( a, b ) {
    if (a < b) {
        return -1;
    }
    if (a > b) {
        return 1;
    }
    return 0;
}
var myArray = new Array ( 1, 5, -3 );
myArray.sort ( compare );
```

285

Math Object Properties

Property	Description
E	Returns Euler's number (approx. 2.718)
LN2	Returns the natural logarithm of 2 (approx. 0.693)
LN10	Returns the natural logarithm of 10 (approx. 2.302)
LOG2E	Returns the base-2 logarithm of E (approx. 1.442)
LOG10E	Returns the base-10 logarithm of E (approx. 0.434)
PI	Returns PI (approx. 3.14159)
SQRT1_2	Returns the square root of 1/2 (approx. 0.707)
SQRT2	Returns the square root of 2 (approx. 1.414)

286

Math Object Methods

Method	Description
abs(x)	Returns the absolute value of x
acos(x)	Returns the arccosine of x, in radians
asin(x)	Returns the arcsine of x, in radians
atan(x)	Returns the arctangent of x as a numeric value between -PI/2 and PI/2 radians
atan2(y,x)	Returns the arctangent of the quotient of its arguments
ceil(x)	Returns x, rounded upwards to the nearest integer
cos(x)	Returns the cosine of x (x is in radians)
exp(x)	Returns the value of E^x
floor(x)	Returns x, rounded downwards to the nearest integer
log(x)	Returns the natural logarithm (base E) of x
max(x,y,z,...,n)	Returns the number with the highest value
min(x,y,z,...,n)	Returns the number with the lowest value
pow(x,y)	Returns the value of x to the power of y
random()	Returns a random number between 0 and 1
round(x)	Rounds x to the nearest integer
sin(x)	Returns the sine of x (x is in radians)
sqrt(x)	Returns the square root of x
tan(x)	Returns the tangent of an angle

287

JS – Example (Custom message box)

- Can be done by opening a new window, containing some HTML that defines the message that you would like to display. However, this has problems, such as pop-ups being blocked by the browser, the page opening in a new browser window or Tab, and so on.
- The solution is to make a piece of HTML that can be layered on top of the content and hidden or shown at will.

288

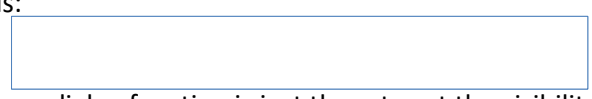
Custom MB

```
<div id="msgBox" style=" width:250px;height:150px;visibility:hidden;left:150;top:150;">
    <table style="background-color:#000080;" width=100% height=100% cellpadding=5>
        <tr height=15px>
            <td style="color:#ffffff;">
                <b>Message Box Title</b>
            </td>
        </tr>
        <tr>
            <td style="background-color:#ffffff;" align="center" valign="center">
                <b>Message Box Text</b><br/>
                <table>
                    <tr>
                        <td>
                            <input type="button" value=" OK " onClick="close_dialog('OK');">
                        </td>
                        <td> </td>
                        <td>
                            <input type="button" value="Cancel" onClick="close_dialog('Cancel');">
                        </td>
                    </tr>
                </table>
            </td>
        </tr>
    </table>
</div>
```

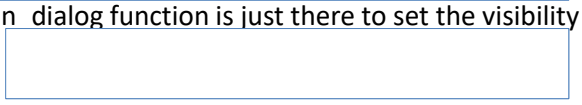
289

Custom MB – cont.

- The whole dialog box is contained inside a div tag, which is set to visibility:hidden so that it cannot be seen initially.
- The close_dialog function, triggered by the onClick events of the two buttons, is just there to set the visibility back to hidden. It looks like this:



- The open_dialog function is just there to set the visibility to visible:



- To test the code, you need to place a button, with appropriate JavaScript code, to make the **div** visible.
  - 

290

- to add an overlay that blanks out the
- original page when the pop-up message box is shown.
- The overlay is defined as follows:
  -

291

Display a Calculation

- You can use the new HTML5 <output> tag to display the result of a calculation on your page.
- The tag works with one or more form <input> tags that accept numeric values from the user.
- The <output> tag enables you to perform mathematical calculations for your users and easily display the results on the page.

292

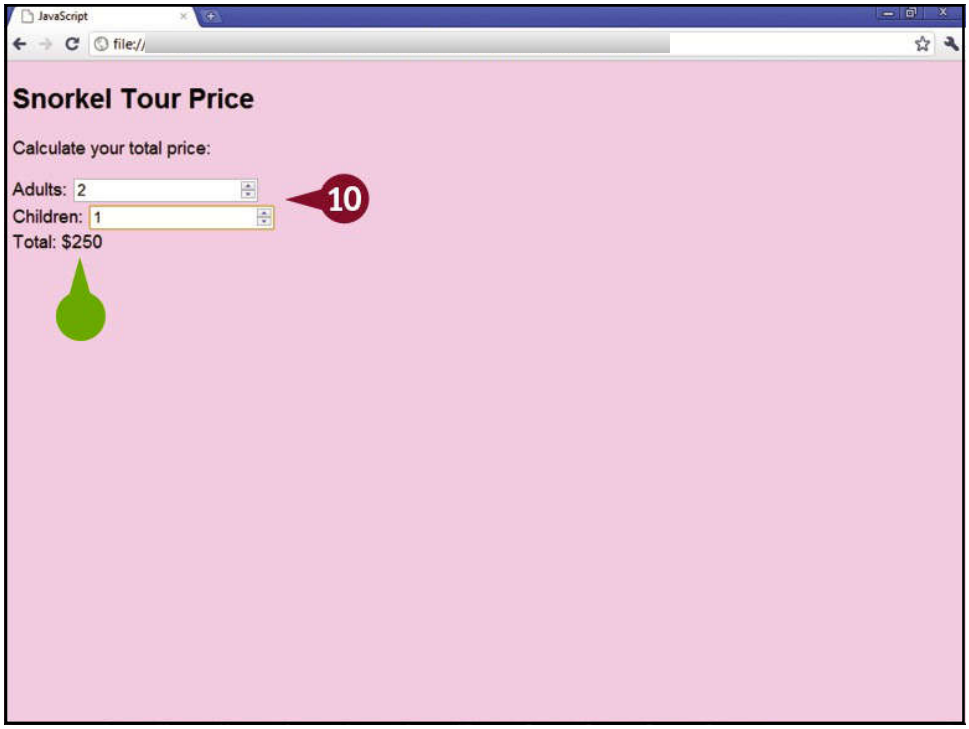
javascript-output.html - Notepad

File Edit Format View Help

```
<!DOCTYPE html>
<html>
<head>
<title>JavaScript</title>
<style>
body { background-color: #ffccff; font-family: arial, sans-serif }
</style>
</head>
<body>

<h2>Snorkel Tou ice</h2>

<p>Calculate you total price:</p>

<form oninput="total.value = adult.valueAsNumber * 100 + child.valueAsNumber * 50">
Adults: <input type="number" step="1" name="adult"><br>
Children: <input type="number" step="1" name="child"><br>
Total: $<output name="total"></output>
</form>

</body>
</html>
```

8 9

293

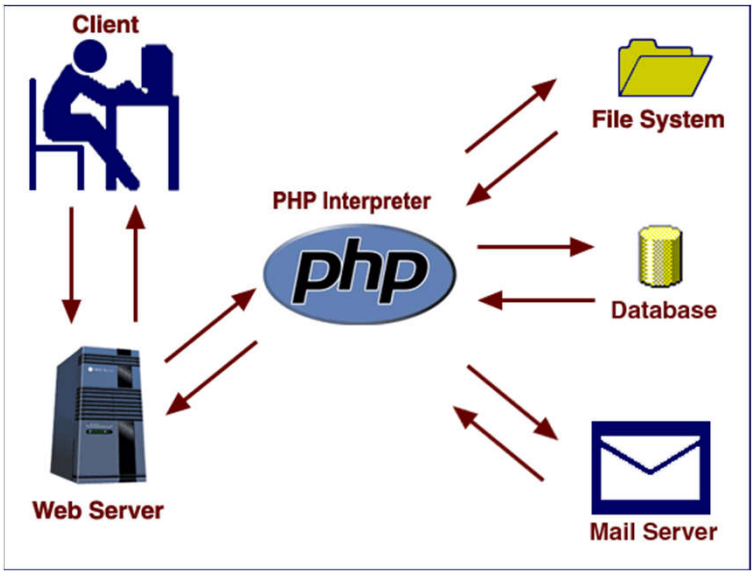


294

Server Side Scripting

295

PHP



296

Intro to PHP

- These PHP scripts are written in plain text, making them easy to edit and upload to the server. Although they can be compiled this is not common for user-created scripts.
- Like JavaScript, PHP can be contained inline within an HTML document, or it can be referenced from an HTML document, usually through embedded PHP code. HTML and PHP can be freely mixed so long as the conventions are followed.
- The difference is that all pages that contain PHP code should have the extension .php, so that the system can send them to the php interpreter.
  - In addition, sometimes they need to have their permissions changed so that they are classified as executable.

297

What if host does not support PhP

- One of the three would happen
  - The PHP code might be identified by the server as text and returned to the browser as a page of text, thereby exposing the PHP code.
  - The server returns an error page, as it is not sure what to do with the PHP file, having only been set up to serve HTML, images, and some other forms of files.
  - The server treats the PHP code as a proprietary object, and the browser starts to download it by asking the user to specify the location for the .php file.
- None of these is desirable, need to make configurations

298

Inline PHP

- Can be written in many different styles
- The correct way to break out of HTML and introduce some PHP is as follows:
  - <body>
  - This is HTML.
  - <?php
  - // PHP code here
  - ?>
  - This is also HTML.
  - </body>
- Ex:
  - <?php
  - echo "Bold Text";
  - ?>

299

External PHP

- The easiest way to reference an external PHP file is through the include function, and is usually inserted in a block of PHP code. For example:
- <?php
- include 'my_module.php';
- ?>

300

Other styles

- Short style
  - <? echo '<p>Order processed.</p>'; ?>
  - This tag style is the simplest and follows the style of a Standard Generalized Markup Language (SGML) processing instruction. To use this type of tag—which is the shortest to type—you either need to enable the short_open_tag setting in your config file or compile PHP with short tags enabled. The use of this style is not recommended because it will not work in many environments as it is no longer enabled by default.
- SCRIPT style
  - <script language='php'> echo ' Order processed. '; </script>
  - This tag style is the longest and will be familiar if you've used JavaScript or VBScript. You might use it if you're using an HTML editor that gives you problems with the other tag styles.
- ASP style
  - <% echo '<p>Order processed.</p>'; %>
  - This tag style is the same as used in Active Server Pages (ASP) or ASP.NET. You can use it if you have enabled the asp_tags configuration setting. You probably have no reason to use this style of tag unless you are using an editor that is geared toward ASP or ASP.NET. Note that, by default, this tag style is disabled.

301

Notes on PHP

- Whitespace are ignored by php unless it is enforced by directives such as \n.
- Comments are the same as in C/C++, in addition
  - It supports shell one line comment style using #

302

Example (form processing)

```
<form action="processorder.php" method="post">
<table border="0">
        <tr bgcolor="cccccc">
                <td width="150">Item</td>
                <td width="15">Quantity</td>
        </tr>
        <tr>
                <td>Keyboards</td>
                <td align="center"><input type="text" name="KBqty" size="3" maxlength="3"/></td>
        </tr>
        <tr>
                <td>CDs</td>
                <td align="center"><input type="text" name="CDqty" size="3" maxlength="3"/></td>
        </tr>
        <tr>
                <td>LCDs</td>
                 <td align="center"><input type="text" name="LCDqty" size="3" maxlength="3" /></td>
        </tr>
        <tr>
                <td align="center"><input type="submit" value="Submit Order"/></td>
        </tr>
</table>
</form>
```

303

Process order (example – cont.)

- processorder.php

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; ">
<title>Insert title here</title>
</head>
  <body>
  <?php
    echo "your order has been processed";
    ?>
  </body>
</html>
```

304

Adding Dynamic Content

- The main reason for using a server-side scripting language is to be able to provide dynamic content to a site's users,
- Updating processorder.php
  - <?php
  - echo " Order processed at ";
  - echo date('H:i, jS F Y');
  - echo " ";
  - ?>
  - This can be also writen as one line using the '.' operator
    - <?php
    - echo " Order processed at ".date('H:i, jS F Y')." ";
    - ?>

305

Using the date() Function

- Expect a formatting string
  - Representing the style of output you would like.
  - H is the hour in a 24-hour format with leading zeros where required,
  - i is the minutes with a leading zero where required,
  - j is the day of the month without a leading zero,
  - S represents the ordinal suffix (in this case th), and
  - F is the full name of the month.
- It will be revisited later

306

PHP page processing

- Whenever PHP processes a page, it checks for URL and form variables, uploaded files, applicable cookies, web server, and environment variables.
- These are then directly accessible in the following arrays: $_GET, $_POST, $_FILES, $_COOKIE, $_SERVER, and $_ENV.

307

Accessing Form Variables

- The point of using the order form is to collect customers' orders.
- Getting the details of what the customers typed is easy in PHP but the exact method depends on
  - the version of PHP you are using and
  - setting in your php.ini file.

308

Accessing form variables – cont.

- Variable names in PHP start with a dollar sign ($).
- One can access each form field as a PHP variable whose name relates to the name of the form field
  - Can be accessed via variables in three ways
    - **$CDqty // short style**
    - **$_POST['CDqty']; // medium style**
    - **$HTTP_POST_VARS['CDqty'] // long style**

309

Accessing form variables – cont.

- Short style (**$CDqty**) is convenient but requires the **register_globals** configuration setting be turned on.
- For security reasons, this setting is turned off by default.
- This style makes it easy to make errors that could make your code insecure, which is why it is no longer the recommended approach. It would be a bad idea to use this style in a new code as the option is likely to disappear in PHP6.

310

Accessing form variables – cont.

- Medium style (**$_POST['CDqty']**) is the **recommended** approach. If you create short versions of variable names, based on the medium style it is not a security issue and instead is simply on ease-of-use issue.
- Long style (**$HTTP_POST_VARS['CDqty']**) is, however, **deprecated** and is therefore likely to be removed in the long term. This style used to be the most portable but can now be disabled via the **register_long_arrays** configuration directive, which improves performance.
  - This requires the global keyword to be accessed, see vars scope, see later slides

311

Why not to use short style

- No obvious distinction between variables that you have created and untrusted variables that have come directly from users.
- If you are not careful to give all your own variables a starting value, your scripts' users can pass variables and values as form variables that will be mixed with your own.

312

$_POST, $_GET, and $_REQUEST

- Medium style involves **retrieving form variables** from one of the arrays $_POST , $_GET , or $_REQUEST.
- One of the $_GET or $_POST arrays holds the details of all the form variables.
  - Depends on whether the method used to submit the form **was GET or POST** , respectively.
- A combination of all data submitted via GET or POST is also available through $_REQUEST.

313

Accessing form variables – cont.

- Returning to processorder.php file we can add
  - <?php
  - $KBqty=$_POST['KBqty'];
  - $CDqty=$_POST['CDqty'];
  - $LCDqty=$_POST['LCDqty'];
  - ?>
- Because this code will not produce any output, placing it above or below the **<html>** and other HTML tags that start your page makes no difference
- Generally place such blocks at the start of the script to make them easy to find.
- To see the value, one can use **echo $KBqty.' KBs
';**
  - The dot operator is used to concatenate string like the plus operator in javascript
  - One can use single or double quotations for string
  - In single quotations nothing altered, **text is output as is**.
  - In double quotations **variables are replaced by their values**
    - echo "$KBqty KBs ";
    - echo $KBqty.' KBs';

314

Identifiers

- Names of variables, classes, or functions
  - Identifiers can be of any length and can consist of letters, numbers, and underscores.
  - Identifiers cannot begin with a digit.
  - In PHP, identifiers are case sensitive. $tireqty is not the same as $TireQty .
    - Function names are an exception to this rule; their names can be used in any case.
  - A variable can have the same name as a function. This usage is confusing, however, and should be avoided. Also, you cannot create a function with the same name as another function.

315

Variable Types

- PHP supports the following basic data types:
  - Integer—Used for whole numbers
  - Float (also called double)—Used for real numbers
  - String—Used for strings of characters
  - Boolean—Used for true or false values
  - Array—Used to store multiple data items
  - Object—Used for storing instances of classes
  - Two special types are also available: NULL and resource.
- Variables that have not been given a value, have been unset
- Certain built-in functions (such as database functions) return variables that have the type resource.
- PHP changes the variable type according to what is stored in it at any given time.

316

Variable Variables

- Variable variables enable you to change the name of a variable dynamically.
- A variable variable works by using the value of one variable as the name of another.
  - For example, you could set
    - **$varname = 'CDqty';**
    - You can then use **$$varname** in place of **$CDqty** .
    - For example, you can set the value of
    - **$CDqty** as follows:
      - **$$varname = 5;**
      - This is exactly equivalent to
        - » **$CDqty = 5;**

317

Declaring and Using Constants

- You can define constants using the define function:
  - define('KBPRICE', 100);
  - define('CDPRICE', 10);
  - define('LCDPRICE', 4);
- One important difference between constants and variables is that when you refer to a constant, it does not have a dollar sign in front of it.
  - echo TIREPRICE;
- As well as the constants you define, PHP sets a large number of its own.
  - An easy way to obtain an overview of them is to run the phpinfo() function:
    - phpinfo();
- This function provides a list of PHP's predefined variables and constants, among other useful information.
- One other difference between variables and constants is that constants can store only boolean, integer, float, or string data.
  - These types are collectively known as scalar values.

318

Variable Scope

- The six basic scope rules in PHP are as follows:
  - Built-in superglobal variables are visible everywhere within a script.
  - Constants, once declared, are always visible globally; that is, they can be used inside and outside functions.
  - Global variables declared in a script are visible throughout that script, but not inside functions .
  - Variables inside functions that are declared as global refer to the global variables of the same name.
  - Variables created inside functions and declared as static are invisible from outside the function but keep their value between one execution of the function and the next.
    - Trying to assign values to these variables which are the result of expressions will cause a parse error.
  - Variables created inside functions are local to the function and cease to exist when the function terminates.

319

Superglobals

- The complete list of superglobals is as follows:
  - **$GLOBALS** —An array of all global variables (Like the **global** keyword, this allows you to access global variables inside a function—for example, as
  - **$GLOBALS[‘myvariable’]** .)
  - **$_SERVER** —An array of server environment variables
  - **$_GET** —An array of variables passed to the script via the **GET** method
  - **$_POST** —An array of variables passed to the script via the **POST** method
  - **$_COOKIE** —An array of cookie variables
  - **$_FILES** —An array of variables related to file uploads
  - **$_ENV** —An array of environment variables
  - **$_REQUEST** —An array of all user input including the contents of input including
  - **$_GET** , **$_POST** , and **$_COOKIE** (but not including **$_FILES** since PHP 4.3.0)
  - **$_SESSION** —An array of session variables

320

Scope – cont.

```
<?php
$a = 1;
include 'b.inc';
?>

/*$a is visible to b.inc script but not to functions
inside the b.inc*/
```

321

Examples

```
<?php
$a = 1; /* global scope */
function test()
{
   echo $a; /* reference to local scope variable */
}
test();   //nothing will be printed
?>
////////////////////////////EX.2
<?php
$a = 1;
$b = 2;
function Sum()
{
   global $a, $b;   //now the global scope is referenced
   $b = $a + $b;   //$b has 3
}
Sum();
echo $b;
?>
```

- Using ***global*** keyword outside a function is not an error. It can be used if the file is included from inside a function.

322

Ex3 – another way to access globals

```
<?php
$a = 1;
$b = 2;
function Sum()
{
   $GLOBALS['b'] = $GLOBALS['a'] + $GLOBALS['b'];
}
Sum();
echo $b;
?>
```

323

Operators

- Arithmetic operators

Operator	Name	Example
+	Addition	$a + $b
-	Subtraction	$a - $b
*	Multiplication	$a * $b
/	Division	$a / $b
%	Modulus	$a % $b

- String operators
  - . Concatenation $a . $b
- Combined Assignment operators

Operator	Use	Equivalent To
+=	$a += $b	$a = $a + $b
-=	$a -= $b	$a = $a - $b
*=	$a *= $b	$a = $a * $b
/=	$a /= $b	$a = $a / $b
%=	$a %= $b	$a = $a % $b
.=	$a .= $b	$a = $a . $b

324

Reference Operator &

- **$a = 5;**
- **$b = $a;**
- **$a = 7; // $b will still be 5**
- **However,**
- **$a = 5;**
- **$b = &$a;**
- **$a = 7; // $a and $b are now both 7**
- Both **$a** and **$b** point to the same piece of memory. You can change this by unsetting one of them as follows:
  - **unset($a);**
- Unsetting does not change the value of **$b** (7) but does break the link between **$a** and the value 7 stored in memory.

325

Comparison Operators

Operator	Name	Use
==	Equals	$a == $b
===	Identical(equal and same type)	$a === $b
!=	Not equal	$a != $b
!==	Not identical	$a !== $b
<>	Not equal (comparison operator)	$a <> $b
<	Less than	$a < $b
>	Greater than (comparison operator)	$a > $b
<=	Less than or equal to	$a <= $b
>=	Greater than or equal to	$a >= $b

326

Logical Operators

Operator	Name	Use	Result
!	NOT	!$b	Returns true if $b is false and vice versa
&&	AND	$a && $b	Returns true if both $a and $b are true; otherwise false
||	OR	$a || $b	Returns true if either $a or $b or both are true; otherwise false
and	AND	$a and $b	Same as &&, but with lower precedence
or	OR	$a or $b	Same as ||, but with lower precedence
xor	XOR	$a x or $b	Returns true if either $a or $b is true, and false if they are both true or both false

327

Other operators

- The ternary operator (?:)
  - condition ? value if true : value if false
- The error suppression operator (**@**)
  - can be used in front of any expression, for example,
    - **$a = @(57/0);**
    - Without the @ operator, this line generates a divide-by-zero warning. With the operator included, the error is suppressed.
  - If you are suppressing warnings in this way, you need to write some error handling code to check when a warning has occurred. If you have PHP set up with the track_errors feature enabled in php.ini , the error message will be stored in the global variable $php_errormsg .

328

Other operators

- The Execution Operator (a pair of backticks (``))
  - PHP attempts to execute whatever is contained between the backticks as a command at the server's command line. The value of the expression is the output of the command.
- Ex: on a Windows server, you can use
  - **$out = `dir c:`;**
  - **echo '<pre>'.$out.'</pre>';**
- The instanceof operator
  - class sampleClass{};
  - $myObject = new sampleClass();
  - if ($myObject instanceof sampleClass)
  - echo "myObject is an instance of sampleClass";
- Working Out the Form Totals

329

Working Out the Form Totals

– Check the php example code

330

Using Variable Functions

- Used to test the variables
- is_array()
  - Checks whether the variable is an array.
- is_double(), is_float(), is_real()
  - (All the same function) Checks whether the variable is a float.
- is_long(), is_int(), is_integer()
  - (All the same function) Checks whether the variable is an integer.
- is_string()
  - Checks whether the variable is a string.
- is_bool()
  - Checks whether the variable is a boolean.
- is_object()
  - Checks whether the variable is an object.
- is_resource()
  - Checks whether the variable is a resource.
- is_null()
  - Checks whether the variable is null.
- is_scalar()
  - Checks whether the variable is a scalar, that is, an integer, boolean, string, or float.

331

Using Variable Functions – cont.

- is_numeric()
  - Checks whether the variable is any kind of number or a numeric string.
- is_callable()
  - Checks whether the variable is the name of a valid function.
- bool isset(mixed var);[;mixed var[,...]])
  - whether a variable is set or not.
- void unset(mixed var);[;mixed var[,...]])
  - unsets a variable
- bool empty(mixed var);
  - checks to see whether a variable exists and has a nonempty, nonzero value; it returns true or false accordingly.
- string gettype(mixed var)
  - return the type of the variable
- bool settype(mixed var, string type)
  - sets the type of a variable

332

Example

- Ex:
  - **$a = 56;**
  - **echo gettype($a).'
'; //int**
  - **settype($a, 'double');**
  - **echo gettype($a).'
';**
  - **echo 'isset($CDqty): '.isset($CDqty).'
';**
  - **echo 'isset($nothere): '.isset($nothere).'
';**
  - **echo 'empty($CDqty): '.empty($CDqty).'
';**
  - **echo 'empty($nothere): '.empty($nothere).'
';**

333

Reinterpreting Variables

- Use type casting like in C/C++
- Using functions
  - The following three functions can be useful for this task:
    - int intval(mixed var[, int base]);
    - float floatval(mixed var);
    - string strval(mixed var);
  - Each accepts a variable as input and returns the variable's value converted to the appropriate type.
  - The intval() function also allows you to specify the base for conversion when the variable to be converted is a string. (This way, you can convert, for example, hexadecimal strings to integers.)

334

Conditional statements

- **if … elseif … else** statement

```
if ($CDqty < 10) {
    $discount = 0;
} elseif (($CDqty >= 10) && ($CDqty <= 49)) {
    $discount = 5;
} elseif (($CDqty >= 50) && ($CDqty <= 99)) {
    $discount = 10;
} else {
    $discount = 15;
}
```

335

Conditional statement – cont.

- **switch** Statements
  - In a switch statement, the condition can take any number of different values, as long as it evaluates to a simple type (integer, string, or float).
  - You need to provide a case statement to handle each value you want to react to and, optionally, a default case to handle any that you do not provide a specific case statement for.
  - Ex:

```
switch($find) {
        case "a”:
                echo "<p>Regular customer.</p>";
                break;
        case "b”:
                echo "<p>Customer referred by TV advert.</p>";
                break;
        case "c”:
                echo "<p>Customer referred by phone directory.</p>";
                break;
        case "d”:
                echo "<p>Customer referred by word of mouth.</p>";
                break;
        default:
                echo "<p>We do not know how this customer found us.</p>";
                break;
}
```

336

Iterations

- **while** Loops

```
Ex:
$num = 1;
while ($num <= 5 ){
    echo $num.”<br />”;
    $num++;
}
```

- **do...while** Loops

```
$num = 100;
do{
    echo $num."<br />";
}while ($num < 1 ) ;
```

337

Iterations – cont.

- **for** and **foreach** Loops
  - If, for example, we have form fields with names such as name1 , name2 , name3 , and so on, we can process them like this:

```
for ($i=1; $i <= $numnames; $i++){
    $temp= "name$i";
    echo $$temp.'<br />'; // or whatever processing you want to do
}
```

  - By dynamically creating the names of the variables, you can access each of the fields in
  - **foreach** designed to work with arrays, to be seen later.

338

Example: Shipping cost

```
<html>
<body>
<table border="0" cellpadding="3">
    <tr>
        <td bgcolor="#CCCCCC" align="center">Distance</td>
        <td bgcolor="#CCCCCC" align="center">Cost</td>
    </tr>
<?php
    $distance = 50;
    while ($distance <= 250) {
    echo "<tr>
        <td align=\"right\">".$distance."</td>
        <td align=\"right\">".($distance / 10)."</td>
        </tr>\n";
    $distance += 50;
    }
?>
</table>
</body>
</html>
```

339

Example: Shipping cost – html equivalent

```
<html>
<body>
<table border="0" cellpadding="3">
        <tr>
                <td bgcolor="#CCCCCC" align="center">Distance</td>
                <td bgcolor="#CCCCCC" align="center">Cost</td>
        </tr>
        <tr>
                <td align="right">50</td>
                <td align="right">5</td>
        </tr>
        <tr>
                <td align="right">100</td>
                <td align="right">10</td>
        </tr>
        <tr>
                <td align="right">150</td>
                <td align="right">15</td>
        </tr>
        <tr>
                <td align="right">200</td>
                <td align="right">20</td>
        </tr>
        <tr>
                <td align="right">250</td>
                <td align="right">25</td>
        </tr>
</table>
</body>
</html>
```

340

Breaking out of control

- **break**: besides switch it can be used to stop executing a loop, the execution of the script will continue at the next line of the script after the loop.
- **continue**: can be used to jump to the next loop iteration.
- **exit**: can be used to finish executing the entire PHP script. For example:

```
if($totalqty == 0){
        echo "You did not order anything on the previous page!<br />";
        exit;
}
```

341

Alternative Control Structure Syntax

- Other than do…while loop, all the other statements have an alternative style
- It consists of replacing the opening brace (**{**) with a colon (**:**) and the closing brace with a new keyword, which will be **endif** , **endswitch** , **endwhile** , **endfor** , or **endforeach**, depending on which control structure is being used.
- Ex:

```
if ($totalqty == 0) :
    echo "You did not order anything on the previous page!<br />";
    exit;
endif;
```

342

Arrays

- Numerically Indexed Arrays
  - indices start at zero by default
  - **$products = array('CD', 'KB', 'LCD');**
  - range() function can be used to automatically create the array of an ascending sequence
    - **$numbers = range(1,10);** //An array called numbers with //elements ranging from 1 to 10
    - **$odds = range(1, 10, 2);** // an optional third parameter that //allows setting the step size between values.
    - **$letters = range('a', 'z');** // can also be used with characters

343

Arrays – cont.

- Accessing Array Contents
  - Type **$products[0]** , **$products[1]** , and **$products[2]** to use the contents of the **$products** array.
  - change array elements' contents by using the = operator.
    - **$products[0] = 'CDs';**
  - We can use loops

```
for ($i = 0; $i<3; $i++) {
        echo $products[$i]." ";
}

Or

foreach ($products as $current) {
        echo $current." ";
}
```

- To add a new element—'Laptops' —to the end of the array, giving a total of four elements:
  - $products[3] = 'Laptops';

344

Arrays with Different Indices

- The following code creates an array with product names as keys and prices as values:
  - **$prices = array('CD'=>100, 'KB'=>10, 'LCD'=>4);**
- We can access the contents using the variable name prices array as $prices['CD'] , $prices['KB'] , and $prices['LCD'] .
- The following code creates the same $prices array
  - $prices = array('CD'=>100);
  - $prices['KB'] = 10;
  - $prices['LCD'] = 4;
  - Or directly as:
    - $prices['CD'] = 100;
    - $prices['KB'] = 10;
    - $prices['LCD'] = 4;

345

- Using Loops

346

Using Loops

- Cannot use a simple counter in a for loop to work with the array
- The foreach loop can be used exactly as it has been used in the previous example, or by incorporating the keys as

```
foreach ($prices as $key => $value) {
        echo $key." – ".$value."<br />";
}
```

- The each() function can also be used as
  - This function returns the current element in an array and makes the next element the current one.

```
while ($element = each($prices)) {
        echo $element['key'];
        echo " – ";
        echo $element['value'];
        echo "<br />";
}
```

  - When you call each() , it gives you an array with four values and the four indices to the array locations. The locations key and 0 contain the key of the current element, and the locations value and 1 contain the value of the current element.
  - If you want to use the array twice in the same script, you need to set the current element back to the start of the array using the function reset() .
    - reset($prices);

- Or using the list() function as

```
while (list($product, $price) = each($prices)) {
        echo "$product - $price<br />";
}
```

347

Multidimensional Arrays

- **$products = array(array('CDS', 'CDs', 100),**
  array('KBS', 'Keyboards', 10),
  array('LCD', 'LCDs', 4));
- **Or**

```
for ($row = 0; $row < 3; $row++) {
for ($column = 0; $column < 3; $column++) {
echo '|'.$products[$row][$column];
}
echo '|<br />';
}
```

```
$products = array( array( 'Code' => 'CDS',
                          'Description' => 'CDs',
                          'Price' => 100
                              ),
                   array( 'Code' => 'KBS',
                          'Description' => 'Keyboards',
                          'Price' => 10
                              ),
                   array( 'Code' => 'LCDS',
                          'Description' => 'LCDs',
                          'Price' =>4
                              )
                   );
for ( $row = 0; $row < 3; $row++){
while ( list( $key, $value ) = each( $products[$row])){
echo "|$value";
}
echo '|<br />';
}
```

348

Sorting Arrays

- **Using the sort() function**
  - **$products = array('CD', 'KBs', 'LCD');**
  - **sort($products);**
  - The array will be sorted into ascending alphabetical
- When using an array with descriptive keys to store items and their prices, we use the function
  - asort() to order the array according to the value of each element.
  - ksort() to sort the array by key rather than value.
  - **$prices = array('Tires'=>100, 'Oil'=>10, 'Spark Plugs'=>4);**
  - **asort($prices);**
- The rsort() function sorts a single-dimensional numerically indexed array into descending order.
- The arsort() function sorts a one-dimensional array into descending order using the value of each element.
- The krsort() function sorts a one-dimensional array into descending order using the key of each element.
- The function usort() sorts an array depending on a custom user defined function

```
function compare($x, $y) {
        if ($x[1] == $y[1]) {
        return 0;
} else if ($x[1] < $y[1]) {
        return -1;
} else {
        return 1;
    }
}
usort($products, 'compare');
```

- Similar to asort() , uasort() used when custom sorting a non-numerically indexed array by value.
- Similar to ksort() , uksort() should be used when custom sorting a non-numerically indexed array by key.

349

Other useful array related functions

- The count() and the sizeof() functions return how many elements are in an array.
- The array_count_values($array) function counts how many times each unique value occurs in the array named $array.

```
$arr = array(4, 5, 1, 2, 3, 1, 2, 1);
$ac = array_count_values($arr);
/*creates an array called $ac that contains
Key          Value
4        1
5        1
1        3
2        2
3        1*/
```

- An Array has an internal pointer.
  - Can be controlled using the current(), next(), each(), prev(), reset(), and end()

350

Useful array related functions

- When creating a new array, the current pointer is initialized to point to the first element in the array.
  - Calling current($array_name) or pos($array_name) directly after array initialization returns the first element.
- Calling either next() or each() advances the pointer forward one element.
  - Calling each($array_name) returns the current element before advancing the pointer.
  - Calling next($array_name) advances the pointer and then returns the new current element.
- The prev() function moves the current pointer back one and then returns the new current element.
- The reset() function returns the pointer to the first element in the array.
- The end($array_name) function sends the pointer to the end of the array.
- The function explode() split up each line so that you can apply some processing and formatting before printing. The output from this script is
- The explode function has the following prototype:
  - array explode(string separator, string string [, int limit])
- The array_count_values() function is more complex. If you call array_count_values($array) , this function counts how many times each unique value occurs in the array named $array .

351

The array_walk() function

- Used to apply a function to each element in an array
  - bool array_walk(array arr, string func, [mixed userdata])
  - Ex1:

```
function my_print($value){
    echo "$value<br />";
}
array_walk($array, 'my_print');
```

  - Ex2:

```
function my_multiply(&$value, $key, $factor){
    $value *= $factor;
}
array_walk(&$array, 'my_multiply', 3);
```

352

Object-Oriented PHP

```
class classname
{
    public $attribute1;
    public $attribute2;
    function __construct($param)
    {
        $this->$attribute1=$param;
        $attribute2=$param;
        echo "Constructor called with parameter ".$param."<br />";
    }
    function __destruct(){}
    function operation1(){}
    function operation2($param1, $param2){}
}
$a = new classname("First");      //prints:Constructor called with parameter First
$b = new classname("Second");     //prints:Constructor called with parameter Second
$c = new classname();             //prints:Constructor called with parameter
```

353

Accessor functions

```
class classname
{
        public $attribute;
}
$a = new classname();
$a->attribute = "value";
echo $a->attribute;          //bad practice, use accessor functions
################################################################
class classname
{
        public $attribute1;
        public $attribute2;
        function __get($name)
        {
                if( ($name == "attribute1"))
                        return $this->$attribute1;
                else
                        return $this->$attribute2;
        }
        function __set ($name, $value)
        {
                if( ($name="attribute1") && ($value >= 0) && ($value <= 100) )
                        $this->attribute = $value;
        }
}

$a=new classname();
$a->$attribute1 = 5;        // will call the __set function
$b=$a->$attribute2;         //will call the __get function
```

354

Inheritance and interface

```
class A
{
        public $attribute1;
        function operation1(){}
}
class B extends A
{
        public $attribute2;
        function operation2(){}
}
```

- PHP does not support multiple inheritance.
- ##

```
interface Displayable
{
        function display();
}
class webPage implements Displayable
{
        function display(){ …}
}
```

- We have to implement all interface functions

355

Static, constants, and final

```
class Math
{
	static function squared($input)
	{
		return $input*$input;
	}
}

echo Math::squared(8);
###################################
interface a
{
	const b = 'Interface constant';
}
echo a::b;
###################################
```

- Prevent method from being override

```
class A
{
	public $attribute = "default value";
	final function operation()
	{
		echo "Something<br />";
		echo "The value of \$attribute is ". $this->attribute."<br />";
	}
}
```

- Prevent class from being inherited

```
final class A{...}
```

356

Example, home.php

357

Setting Cookies

- Use setcookie()
- <?php
- setcookie('flavor','chocolate chip');
- ?>
- Cookies are sent with the HTTP headers, so if you're not using output buffering, setcookie( ) must be called before any output is generated.

358

Storing and Retrieving Data

- We can store data in two basic ways: in
  - flat files or in a
  - database.
  - By a flat file , we mean a simple text file.

359

Querying a Database from the Web

- In any script used to access a database from the Web, you follow some basic steps:
  - 1. Check and filter data coming from the user.
  - 2. Set up a connection to the appropriate database.
  - 3. Query the database.
  - 4. Retrieve the results.
  - 5. Present the results back to the user.

360

361

Connecting to DB

- The PHP library for connecting to MySQL is called mysqli (the i stands for improved).
- In PHP, you can use either an object-oriented or procedural syntax.
  - @ $db = new mysqli('localhost', 'bookorama', 'bookorama123', 'books');
    - » Returns an object
  - @ $db = mysqli_connect('localhost', 'bookorama', 'bookorama123', 'books');
    - » Returns a variable of type resource
    - » You will need to pass this resource in to all the other mysqli functions. This is very similar to the way the file-handling functions, such as fopen() , work.
- In either method (OOP or function), the result of an attempt for a connection can be checked for errors as follows:
  - if (mysqli_connect_errno()) {
  - echo 'Error: Could not connect to database. Please try again later.';
  - exit;
  - }
- Number of simultaneous connections to MySQL is configured through max_connections parameter in the file my.conf.
- Another related Apache parameter is the MaxClient in the httpd.conf file. This tells the server to reject new connection requests that exceeds MaxClient instead of allowing machine resources to be completely used up at busy times or when software has crashed.

362

Connecting to DB

- It is a good practice to have your connection account info in a separated file also directory.
- Connecting to a mysql DB is done using the function mysqli()
- <?php
- // this is dbconnect.php
- $db_server = 'localhost';
- $db_user_name = ’ahmad';
- $db_password = ’password';
- $db_name = 'somedb’;
- ?>

- <?php
- include('../code/dbconnect.php);
- $conn = @new mysqli($db_server, $db_user_name, $db_password,
- $db_name);
- // etc
- ?>

363

Using a DB and retrieving data from a DB

- When you are using MySQL, you need to tell it which database you plan to use.
- The database to use is specified as a parameter to the mysqli constructor or the mysqli_connect() function.
- If you want to change the default database, you can do so with the mysqli_select_db() function.
- It can be accessed as either
  - $db->select_db(dbname)
  - or as
    - mysqli_select_db(db_resource, db_name)
- To actually perform the query, you can use the mysqli_query() function. Before doing this, however, it's a good idea to set up the query you want to run:
  - $query = "select * from books where ".$searchtype." like '%".$searchterm."%'";
- You can now run the query:
  - **$result = $db->query($query);**
  - Or, if you want to use the procedural interface, you use
    - **$result = mysqli_query($db, $query);**

364

Retrieving the Query Results

- To retrieve number of rows returned
  - Using the object-oriented approach,
    - $num_results = $result->num_rows;
  - When you use a procedural approach, the
    - $num_results = mysqli_num_rows($result);
- Then one can iterate through the resulted rows
  - for ($i=0; $i <$num_results; $i++) {
  - // process results
  - }
- In each iteration of this loop, you call $result->fetch_assoc() (or mysqli_fetch_assoc()).
  - This function takes each row from the result set and returns the row as an array, with each key an attribute name and each value the corresponding value in the array:
    - $row = $result->fetch_assoc();
  - Or you can use a procedural approach:
    - $row = mysqli_fetch_assoc($result);
  - The attribute values are listed in each of the array values **$row[0]** , **$row[1]** , and so on.
  - (The **mysqli_fetch_array()** function allows you to fetch a row as either or both kinds of array.)
- You could also fetch a row into an object with the **mysqli_fetch_object()** function:
  - **$row = $result->fetch_object();**
  - or
    - **$row = mysqli_fetch_object($result);**
  - You can then access each of the attributes via **$row->title** , **$row->author** , and so on.

365

Disconnecting from the Database

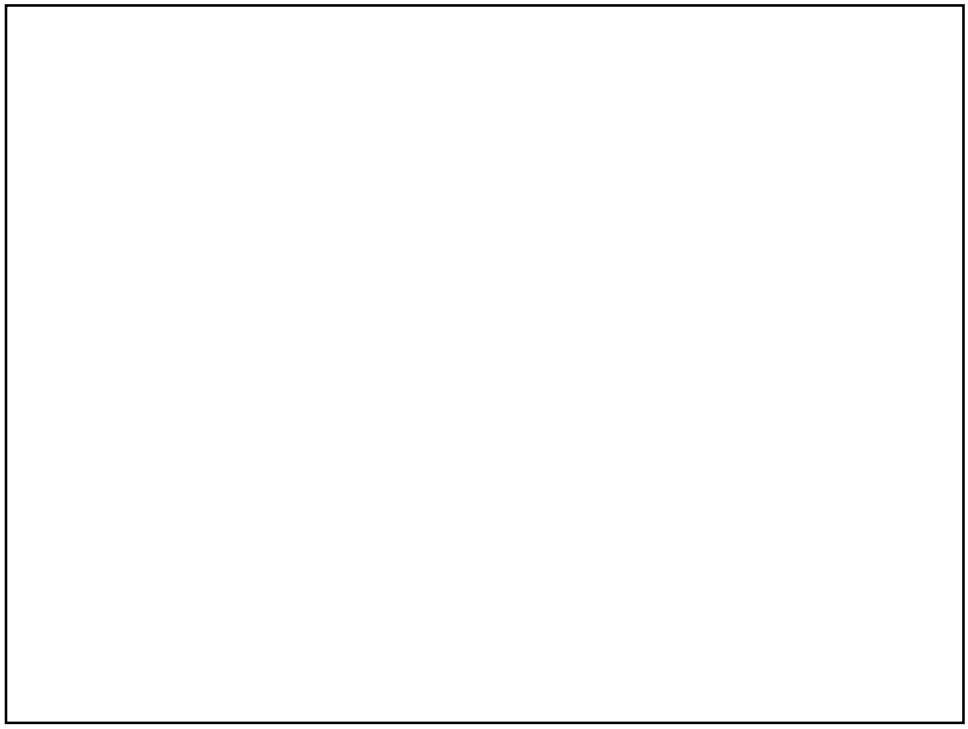
- You can free up your result set by calling either
  - **$result->free();**
  - or
  - **mysqli_free_result($result);**
- You can then use
  - **$db->close();**
  - or
  - **mysqli_close($db);**
  - to close a database connection.
  - Using this command isn't strictly necessary because the connection will be closed when a script finishes execution anyway.

366

Putting New Information in the Database

- Similar to getting items out of the DB.
  - Same basic steps:
    - make a connection,
    - send a query,
      - In this case, the query you send is an INSERT rather than a SELECT.
    - and check the results.
- However, we first need to get the data to be inserted.
  - This can be done using a simple html page, see next slide.

367



368

newbook.html

```
<html>
	<head>
		<title>Book-Store- New Book Entry</title>
	</head>
	<body>
		<h1>Book-Store- New Book Entry</h1>
		<form action="insert_book.php" method="post">
			<table border="0">
				<tr>
					<td>ISBN</td>
					<td><input type="text" name="isbn" maxlength="13" size="13"></td>
				</tr>
				<tr>
					<td>Author</td>
					<td> <input type="text" name="author" maxlength="30" size="30"></td>
				</tr>
				<tr>
					<td>Title</td>
					<td> <input type="text" name="title" maxlength="60" size="30"></td>
				</tr>
				<tr>
					<td>Price $</td>
					<td><input type="text" name="price" maxlength="7" size="7"></td>
				</tr>
				<tr>
					<td colspan="2"><input type="submit" value="Register"></td>
				</tr>
			</table>
		</form>
	</body>
</html>
```

369

insert_book.php

```
<html>
	<head>
		<title>Book Entry Results</title>
	</head>
	<body>
		<h1>Book Entry Results</h1>
	<?php
		// create short variable names
		$isbn=$_POST['isbn'];
		$author=$_POST['author'];
		$title=$_POST['title'];
		$price=$_POST['price'];
		if (!$isbn || !$author || !$title || !$price) {
			echo "You have not entered all the required details.<br />"
				."Please go back and try again.";
			exit;
		}
		if (!get_magic_quotes_gpc()) {
			$isbn = addslashes($isbn); //Returns a string with backslashes before characters that need to be quoted in database queries etc.
						//These characters are single quote ('), double quote ("), backslash (\) and NUL (the NULL byte).
						// for example when adding the string to insert the name O'reilly
			$author = addslashes($author);
			$title = addslashes($title);
			$price = doubleval($price);
		}
		@ $db = new mysqli('localhost', 'dbUsername', 'dbpass', 'dbname');
		if (mysqli_connect_errno()) {
			echo "Error: Could not connect to database. Please try again later.";
			exit;
		}
		$query = "insert into books values('".$isbn."', '".$author."', '".$title."', '".$price."')";
		$result = $db->query($query);
		if ($result) {
			echo $db->affected_rows." book inserted into database."; //affected_rows returns the number of rows affected by the insert
										//use it in INSERT, UPDATE, and DELETE statements
		} else {
			echo "An error has occurred. The item was not added.";
		}
		$db->close();
?>
</body>
</html>
```

370

Using Prepared Statements

- The mysqli library supports the use of prepared statements
  - Useful for speeding up execution when you are performing large numbers of the same query with different data.
  - They also protect against SQL injection-style attacks.
- Done by sending a template of the query you want to execute to MySQL and then send the data separately
- For example, the previous insert query could be
  - **$query = "insert into books values(?, ?, ?, ?)";**
  - **$stmt = $db->prepare($query);**
  - **$stmt->bind_param("sssd", $isbn, $author, $title, $price);**
  - **$stmt->execute();**
  - **echo $stmt->affected_rows.' book inserted into database.';**
  - **$stmt->close();**
  - The purpose of bind_param() is to tell PHP which variables should be substituted for the question marks.
  - The first parameter is a format string.
  - The value you are passing here (“sssd”) means that the four parameters are a string, a string, a string, and a double, respectively.
  - Other possible characters in the format string are i for integer and b for blob (Binary Large OBject such as images in a database).
  - After this parameter, you should list the same number of variables as you have question marks in your statement. They will be substituted in this order.

371

Using Prepared Statements

- As well as binding parameters, you can bind results.
- For SELECT type queries, you can use $stmt->bind_result() (or mysqli_stmt_bind_result()) to provide a list of variables that you would like the result columns to be filled into. Each time you call $stmt->fetch() (or mysqli_stmt_fetch()), column values from the next row in the result set are filled into these bound variables. For example, in the book search script you looked at earlier, you could use $stmt->bind_result($isbn, $author, $title, $price); to bind these four variables to the four columns that will be returned from the query.
- After calling $stmt->execute(); you can call $stmt->fetch(); in the loop.
- Each time this is called, it fetches the next result row into the four bound variables.
- You can also use mysqli_stmt_bind_param() and mysqli_stmt_bind_result() in the same script.

372

Using Other PHP-Database Interfaces

- PHP supports libraries for connecting to a large number of databases, including Oracle, Microsoft SQL Server, and PostgreSQL.
- In general, the principles of connecting to and querying any of these databases are much the same.
- The individual function names vary, and different databases have slightly different functionality, but if you can connect to MySQL, you should be able to easily adapt your knowledge to any of the others.
- If you want to use a database that doesn't have a specific library available in PHP, you can use the generic ODBC functions. ODBC, which stands for Open Database

373

Sessions

- "HTTP is a stateless protocol."
  - This means that the protocol has no built-in way of maintaining state between two transactions.
  - When a user requests one page, followed by another, HTTP does not provide a way for you to tell that both requests came from the same user.
- The idea of session control is to be able to track a user during a single session on a website.
- Sessions in PHP are driven by a unique session ID, a cryptographically random number.
  - It works by creating a unique identification(UID) number for each visitor and storing variables based on this ID. This helps to prevent two users' data from getting confused with one another when visiting the same webpage.

374

Sessions – cont.

- Before using sessions, start it
  - session_start(); // start up your PHP session!
    - This registers the user's session with the server, allow you to start saving user information and assign a UID (unique identification number) for that user's session.
- When you want to store user data in a session use the $_SESSION
  - $_SESSION['views'] = 1; // store session data
  - echo "Pageviews = ". $_SESSION['views']; //retrieve data
- To check if a session already have a value, use isset. Use unset to clear it
  - if(isset($_SESSION['views']))
  - $_SESSION['views'] = $_SESSION['views']+ 1;
  - else
  - $_SESSION['views'] = 1;
- we can also completely destroy the session entirely by calling the *session_destroy* function.
  - session_destroy();

375

Authentications

- There are two good ways for authentication:
  - HTTP basic authentication
    - Basic authentication overcomes the caching problem, but the browser still sends the password to the server with every request
  - Sessions
    - Session control overcomes both of these problems.

376

Using Basic Authentication

- You can trigger basic authentication using
  - PHP or
  - Using mechanisms built into your web server
- It is supported by web browsers and requested form scripts.
- Basic authentication transmits a user's name and password in plain text, so it is not very secure.
- When combining basic authentication with SSL and digital certificates, all parts of a web transaction can be protected by strong security

377

Basic authentication using PHP

- PHP scripts are generally cross-platform, but using basic authentication relies on environment variables
  – set by the server.
- For an HTTP authentication script to run on Apache using PHP as an Apache module or on IIS using PHP as an ISAPI module, it needs to detect the server type and behave slightly differently.

378

PHP authentication – cont.

```
<?php
// if we are using IIS, we need to set
// $_SERVER['PHP_AUTH_USER'] and
// $_SERVER['PHP_AUTH_PW']
if ((substr($_SERVER['SERVER_SOFTWARE'], 0, 9) == 'Microsoft') &&
(!isset($_SERVER['PHP_AUTH_USER'])) &&
(!isset($_SERVER['PHP_AUTH_PW'])) &&
(substr($_SERVER['HTTP_AUTHORIZATION'], 0, 6) == 'Basic ')
) {
        list($_SERVER['PHP_AUTH_USER'], $_SERVER['PHP_AUTH_PW']) =
        explode(':', base64_decode(substr($_SERVER['HTTP_AUTHORIZATION'], 6)));
}
// Replace this if statement with a database query or similar
if (($_SERVER['PHP_AUTH_USER'] != 'user') ||
($_SERVER['PHP_AUTH_PW'] != 'pass')) {
        // visitor has not yet given details, or their
        // name and password combination are not correct
        header('WWW-Authenticate: Basic realm="Realm-Name"');
        if (substr($_SERVER['SERVER_SOFTWARE'], 0, 9) == 'Microsoft') {
        header('Status: 401 Unauthorized');
        } else {
                header('HTTP/1.0 401 Unauthorized');
        }
        echo "<h1>Go Away!</h1>
        <p>You are not authorized to view this resource.</p>";
} else {
        // visitor has provided correct details
        echo "<h1>Here it is!</h1>
        <p>I bet you are glad you can see this secret page.</p>";
}
?>
```

379

Using Basic Authentication with Apache's **.htaccess** Files

- The Apache web server contains a number of different authentication modules that can be used to decide the validity of data entered by a user.
- The easiest to use is **mod_auth** , which compares name-password pairs to lines in a text file on the server.
- For that we need to create two separate HTML files: one for the content and one for the rejection page.
  - content.html— Sample Content
    - <html><body>
    - <h1>Here it is!</h1>
    - I bet you are glad you can see this secret page.
    - </body></html>
  - rejection.html—Sample 401 Error Page
    - <html><body>
    - <h1>Go Away!</h1>
    - You are not authorized to view this resource.
    - </body></html>

380

Using Basic Authentication with Apache's - cont.

- In addition a file with the name **.htaccess** is needed to be created and stored in the folder containing your website
  - This will control accesses to files and any subdirectories in its directory.
  - An **.htaccess** File Can Set Many Apache Configuration Settings, Including Activating Authentication
  - Ex:
    - **ErrorDocument 401 /rejection.html**
    - **AuthUserFile /home/.htpass**
    - **AuthGroupFile /dev/null**
    - **AuthName "Realm-Name"**
    - **AuthType Basic**
    - **require valid-user**
- **.htpass**— The Password File Stores Usernames and Each User's Encrypted Password
  - **user1:0nRp9M80GS7zM**
  - **user2:nC13sOTOhp.ow**
  - **user3:yjQMCPWjXFTzU**
  - **user4:LOmlMEi/hAme2**

381

Apache basic authentication – cont.

- The **/dev/null** is a special file on Unix systems that is guaranteed to be null, however, it is possible to specify that only authorized users who fall into specific groups may access resources.
- Like the PHP example, to use HTTP authentication, you need to name the realm as follows:
  - **AuthName "Realm-Name"**
  - You can choose any realm name you prefer, but this name will be shown to your visitors
- Each line in the **.htpass** file contains a username, a colon, and that user's encrypted password.
  - To create it, you use a small program called htpasswd that comes in the Apache distribution.
- The htpasswd program is used in one of the following ways:
  - htpasswd [-cmdps] passwordfile username
  - or
  - htpasswd -b[cmdps] passwordfile username password
  - Using -c tells htpasswd to create the file.
- The optional **m , d , p ,** or **s** switches can be used if you want to specify which encryption algorithm (including no encryption) you would like to use.
- The **b** switch tells the program to expect the password as a parameter rather than prompt for it.
  - htpasswd -bc /home/book/.htpass user1 pass1
  - htpasswd -b /home/book/.htpass user2 pass2
  - htpasswd -b /home/book/.htpass user4 pass3
  - htpasswd -b /home/book/.htpass user4 pass4

382

Configuring Session Control

- There is a set of configuration options for sessions that you can set in your php.ini file.
- Ex:

Option Name	Default	Effect
session.auto_start	0	**(disabled) Automatically starts sessions.**
session.cache_expire	180	**Sets time-to-live for cached session pages, in minutes.**

383

Ex: Implementing Authentication with Session Control

- This example consists of three files.
  - Login.php
  - Mainpage.php
  - Logout.php
- We will assume the use of MySQL that have the database auth containing a table holding the authorized users for our example website.
- This database can be created using the following SQL script.

```
use auth;
create table authorized_users (name varchar(20),
                               password varchar(40),
                               primary key (name)
                              );

insert into authorized_users values ('username',
                               'password');

insert into authorized_users values ('testuser',
                               sha1('password'));

grant select on auth.* to 'webauth' identified by 'webauth';

flush privileges;
```

384

Encrypting Passwords

- Storing the passwords as plain text is an unnecessary risk.
- A one-way hashing algorithm can provide better security with very little extra effort.
- PHP provides a number of one-way hash functions.
  - The oldest and least secure is the Unix Crypt algorithm, provided by the function *crypt()* .
  - The Message Digest 5 (MD5) algorithm, implemented in the function *md5()*, is stronger.
  - Stronger yet is the Secure Hash Algorithm 1 (SHA-1.) which provides a strong, one-way cryptographic hash function.
    - The prototype for this function is
      - *string sha1 (string str [, bool raw_output])*
        - » Given the string *str* , the function will return a pseudo-random 40-character string.
        - » If you set *raw_output* to be *true* , you will instead get a 20-character string of binary data.

385

Storing and Retrieving

- Writing data to a file requires three steps:
  - Open the file. If the file doesn't already exist, you need to create it.
  - Write the data to the file.
  - Close the file.
- Similarly, reading data from a file takes three steps:
  - Open the file. If you cannot open the file (for example, if it doesn't exist), you need to recognize this and exit gracefully.
  - Read data from the file.
  - Close the file.
- When you want to read data from a file, you have many choices about how much of the file to read at a time.

386

Opening a File

- Use the fopen() function.
  - We need to specify how we intend to use it. This is known as the **file mode**.
- Three main things can be done using file mode
  - Whether to open a file for **reading only**, for **writing only**, or for both **reading and writing**.
  - In writing, whether to **overwrite** any existing contents of a file or **append** new data to the end of the file
  - Specify whether to write **binary** or **text** files.

387

File modes

Mode	Mode Name	Meaning
r	Read	Open the file for reading, beginning from the start of the file.
r+	Read	Open the file for reading and writing, beginning from the start of the file.
w	Write	Open the file for writing, beginning from the start of the file. If the file already exists, delete the existing contents. If it does not exist, try to create it.
w+	Write	Open the file for writing and reading, beginning from the start of the file. If the file already exists, delete the existing contents. If it does not exist, try to create it.
x	Cautious write	Open the file for writing, beginning from the start of the file. If the file already exists, it will not be opened, fopen() will return false, and PHP will generate a warning.
x+	Cautious write	Open the file for writing and reading, beginning from the start of the file. If the file already exists, it will not be opened, fopen() will return false, and PHP will generate a warning.
a	Append	Open the file for appending (writing) only, starting from the end of the existing contents, if any. If it does not exist, try to create it.
a+	Append	Open the file for appending (writing) and reading, starting from the end of the existing contents, if any. If it does not exist, try to create it.
b	Binary	Used in conjunction with one of the other modes. You might want to use this mode if your file system differentiates between binary and text files. Windows systems differentiate; Unix systems do not. The PHP developers recommend you always use this option for maximum portability. It is the default mode.
t	Text	Used in conjunction with one of the other modes. This mode is an option only in Windows systems. It is not recommended except before you have ported your code to work with the b option.

388

Using fopen() to Open a File

- Assuming we want to save the computer shop order to a file, we can open this file for writing with the following:
  - **$fp = fopen("$DOCUMENT_ROOT/../orders/orders.txt", 'w');**
- fopen() expects two, three, or four parameters.
  - Usually, we use two.
  - The first parameter should be the file we want to open.
    - The document root is accessed using the PHP built-in variable $_SERVER
    - $DOCUMENT_ROOT=$_SERVER['DOCUMENT_ROOT'];
    - This variable points at the base of the document tree on your web server.
    - We use the .. to point to "the parent directory of the document root directory."
    - It is a good practice, for security reasons, to store sensitive data outside the director root. we do not want this file to be web accessible except through the interface that we provide.
    - If no path is specified, the file will be created or looked for in the same directory as the script itself.

389

Using fopen() – cont.

- The second argument is used for specifying mode.
  - 'w' allows only one order to be stored in the file. Each time a new order is taken, it overwrites the previous order
  - A better choice would be 'ab'
- The third parameter of fopen() is optional.
  - You can use it if you want to search the include_path (set in php.ini) for a file.
  - If you want to do this, set this parameter to **true**. If you tell PHP to search the **include_path** , you do not need to provide a directory name or path:
    - **$fp = fopen('orders.txt', 'ab', true);**
- The fourth parameter is also optional. The fopen() function allows filenames to be prefixed with a protocol (such as http://) and opened at a remote location.
  - You don't need this if you specify the protocol in the file name, see next slide.
- If fopen() opens the file successfully, a resource that is effectively a handle or pointer to the file is returned and should be stored in a variable—in this case, $fp. You use this variable to access the file when you actually want to read from or write to it.

390

Opening Files Through FTP or HTTP

- We can open files remotely with fopen() by specifying the protocol in the forth parameter
  - This capability can be disabled by turning off the allow_url_fopen directive in the php.ini file.
- If the filename you use begins with ftp:// , a passive mode FTP connection will be opened to the server you specify and a pointer to the start of the file will be returned.
- If the filename you use begins with http:// , an HTTP connection will be opened to the server you specify and a pointer to the response will be returned. When using HTTP mode with older versions of PHP, you must specify trailing slashes on directory names, as shown in the following:
  - http://www.example.com/
  - $file = fopen ("http://www.example.com/", "r");
  - not
  - http://www.example.com
- FTP example: $handle = **fopen**("**ftp**://user:pass@DomainName.com/filename.txt", "w");
- Remember that the domain names in your URL are not case sensitive, but the path and filename might be.
- Important: you need to have permissions to open the file you need.

391

Dealing with errors

- If the call to fopen() fails, (e.g. you don't have permissions to access file), the function will return false .
- You can deal with the error in a more user-friendly way by suppressing PHP's error message and giving your own:

```
@ $fp = fopen("$DOCUMENT_ROOT/../orders/orders.txt", 'ab');
if (!$fp){
    echo "<p><strong> Your order could not be processed at this time. "
        .Please try again later.</strong></p></body></html>";
    exit;
}
```

- The @ symbol in front of the call to fopen() tells PHP to suppress any errors resulting from the function call. Usually, it's a good idea to know when things go wrong, but in this case we're going to deal with that problem elsewhere. You can also write this line as follows:
  - $fp = @fopen("$DOCUMENT_ROOT/../orders/orders.txt", 'a');

392

Writing to a File

- We can use either of the functions fwrite() (file write) or fputs() (file put string);
- fputs() is an alias to fwrite(). You call fwrite() in the following way:
  - fwrite($fp, $outputstring);
  - Writes the string in outputstring to file fp
- Alternatively we can use
  - int file_put_contents (string filename, string data [, int flags [, resource context]])
  - This function writes the string contained in data to the file named in filename without any need for an fopen() (or fclose()) function call. This function is new in PHP5, and is a matched pair for file_get_contents()
  - The flags and context optional parameters are most commonly used when writing to remote files using, for example, HTTP or FTP.

393

Writing to a file – cont.

- The function fwrite() takes three parameters
  - The third one is optional.
  - The prototype for fwrite() is
    - int fwrite (resource handle, string string [, int length])
  - The third parameter, length, is the maximum number of bytes to write. If this parameter is supplied, fwrite() will write string to the file pointed to by handle until it reaches the end of string or has written length bytes, whichever comes first.
  - Ex: fwrite($fp, $outputstring, strlen($outputstring));
  - Third parameter commonly used when writing in binary mode because it helps avoid some cross-platform compatibility issues.

394

File Formats

- Depending on who is using the stored data file.
- Better to use a standard, such as XML
- Or invent one for your self, but you have to follow it and announce it to others.
  - **$outputstring = $date."\t".$CDqty." CDs \t".$KBqty." KBs\t".$LCDqty." LCDs\t\$".$totalamount."\t". $address."\n";**

- **Closing a File**
  - When you finish using a file close it with
    - fclose($fp);
    - Returns true if the file was successfully closed or false if it wasn't.

395

Locking sensitive files for exclusive use

- When dealing with sensitive data such as customer orders, it is good practice to have at one time only one instance that can modify the file, for that, we can use the function
  - bool **flock** (resource *$handle* , int *$operation* [, int *&$wouldblock*]) //wouldblock is not supported in windows
  - Operation can be one of
    - **LOCK_SH** to acquire a shared lock (reader).
    - **LOCK_EX** to acquire an exclusive lock (writer).
    - **LOCK_UN** to release a lock (shared or exclusive).
- Ex: flock($fp, LOCK_EX); //should be unlocked in ver. 5.3.2, other wise the close would unlock
  - flock($fp, LOCK_UN);

396

Reading from a File

- Knowing When to Stop: feof()
  - while (!feof($fp))
- Reading a Line at a Time: fgets(), fgetss(), and fgetcsv()
  - Ex:
    - $order= fgets($fp, 999);
      - Reads from file until it encounters a **newline character (\n), encounters an EOF , or has read 998 bytes from the file**.
    - string fgetss(resource fp, int length, string [allowable_tags]);
      - Same as fgets but it strips out any html/php tags other than the ones specified by the allowable tags string
      - Used for security reasons to eliminate any threats from text coming from untrusted sources
    - array fgetcsv (resource fp, int length [, string delimiter [, string enclosure]])
      - This function breaks up lines of files when you have used a delimiting character, such as the tab character (as we suggested earlier) or a comma (as commonly used by spreadsheets and other applications).
      - **$order = fgetcsv($fp, 100, "\t");**
      - The enclosure parameter specifies what each field in a line is surrounded by. If not specified, it defaults to " (a double quotation mark).

397

Reading from a file – cont.

- Reading the Whole File: readfile(), fpassthru(), and file()
- **int readfile(string filename, [int use_include_path[, resource context]]);**
  - A call to the readfile() function opens the file, echoes the content to standard output (the browser), and then closes the file.
  - The optional context parameter is used only when files are opened remotely via, for example, HTTP
- **fpassthru(),** dumps the contents of the file from the pointer's position onward to standard output. It closes the file when it is finished.
  - **$fp = fopen("$DOCUMENT_ROOT/../orders/orders.txt", 'rb');**
  - **fpassthru($fp);**
- The **file()** function is identical to readfile() except that instead of echoing the file to standard output, it turns it into an array. We would call it using
  - **$filearray = file($DOCUMENT_ROOT/../orders/orders.txt");**
  - Each line of the file is stored in a separate element of the array
- The **file_get_contents()** function is identical to readfile() except that it returns the content of the file as a string instead of outputting it to the browser.

398

Reading from a file – cont.

- Reading a Character: fgetc()

```
while (!feof($fp)){
    $char = fgetc($fp);
    if (!feof($fp))
        echo ($char=="\n" ? "<br />": $char);
    }
}
```

  - This code reads a single character at a time from the file using fgetc() and stores it in $char
  - It also does a little processing to replace the text end-of-line characters (\n) with HTML line breaks (
).
  - fgetc() returns the EOF character. You need to test feof() again after you've read the character because you don't want to echo the EOF to the browser.
- Reading an Arbitrary Length: fread()
  - string fread(resource fp, int length);
  - Reads up to length bytes or to the end of the file, whichever comes first.

399

try{}catch(){}

```
<?php
try {
  throw new Exception("A terrible error has occurred", 42);
}
catch (Exception $e) {
  echo "Exception ". $e->getCode(). ": ". $e->getMessage()."<br />”." in ".
$e->getFile(). " on line ". $e->getLine(). "<br />";
}
?>
```

##
##########################

400

The Exception Class

- PHP has a built-in class called Exception.
- The constructor takes two parameters, an error message and an error code.
- In addition to the constructor, this class comes with the following built-in methods:
  - getCode()—Returns the code as passed to the constructor
  - getMessage()—Returns the message as passed to the constructor
  - getFile()—Returns the full path to the code file where the exception was raised
  - getLine()—Returns the line number in the code file where the exception was raised
  - getTrace()—Returns an array containing a backtrace where the exception was raised
  - getTraceAsString()—Returns the same information as getTrace, formatted as a string
  - __toString()—Allows you to simply echo an Exception object, giving all the information from the above methods

401

Using Other Useful File Functions

- To check whether a file exists without actually opening it.
  - if (file_exists("$DOCUMENT_ROOT/../orders/orders.txt"))
- Determining How Big a File Is: filesize()
  - **$fp = fopen("$DOCUMENT_ROOT/../orders/orders.txt", 'rb');**
  - **echo nl2br(fread($fp, filesize("$DOCUMENT_ROOT/../orders/orders.txt")));**
  - **fclose($fp);**
  - The **nl2br()** function converts the **\n** characters in the output to HTML line breaks (**
**).
- Deleting a File: unlink()
  - **unlink("$DOCUMENT_ROOT/../orders/orders.txt");**
  - Returns false if the file could not be deleted. This situation typically occurs if the permissions on the file are insufficient or if the file does not exist.

402

Other Useful File Functions – cont.

- Navigating Inside a File: rewind(), fseek(), and ftell()
  - The **rewind($fp)** function resets the file pointer to the beginning of the file. The **ftell($fp)** function reports how far into the file the pointer is in bytes.
  - The function fseek() to set the file pointer to some point within the file. Its prototype is
    - **int fseek (resource fp, int offset [, int whence])**
    - sets the file pointer fp at a point starting from whence and moving offset bytes into the file.
    - The optional whence parameter defaults to the value SEEK_SET , which is effectively the start of the file. The other possible values are SEEK_CUR (the current location of the file pointer) and SEEK_END (the end of the file).

403

Example: Interacting with the File System and the Server

- Before looking into file uploading example, it is important to note that the php.ini file has five directives that control how PHP will work with file uploading

Directive	Description	Default Value
file_uploads	Controls whether HTTP file uploads are allowed. Values are On or Off.	On
upload_tmp_dir	Indicates the directory where uploaded files will temporarily be stored while they are waiting to be processed. If this value is not set, the system default will be used.	NULL
upload_max_file size	Controls the maximum allowed size for uploaded files. If a file is larger than this value, PHP will write a 0 byte placeholder file instead.	2M
post_max_size	Controls the maximum size of POST data that PHP will accept.This value must be greater than the value for the upload_max_filesize directive, since it is the size for all of the post data, including any files to be uploaded.	8M

404

HTML for File Upload

```
<html>
    <head>
        <title>Administration - upload new files</title>
    </head>
    <body>
        <h1>Upload new news files</h1>
        <form action="upload.php" method="post" enctype="multipart/form-data"/>
        <div>
            <input type="hidden" name="MAX_FILE_SIZE" value="1000000" />
            <label for="userfile">Upload a file:</label>
            <input type="file" name="userfile" id="userfile"/>
            <input type="submit" value="Send File"/>
        </div>
        </form>
    </body>
</html>
```

405

HTML for File Upload – cont.

- In the **<form>** tag, you must set the attribute **enctype="multipart/form-data"** to let the server know that a file is coming along with the regular information.
- You must have a form field that sets the maximum size file that can be uploaded. This is a hidden field and is shown here as
  - **<input type="hidden" name="MAX_FILE_SIZE" value=" 1000000">**
  - Note that the **MAX_FILE_SIZE** form field is optional, as this value can also be set server-side. However, if used in the form, the name of this form field must be **MAX_FILE_SIZE**. The value is the maximum size (in bytes) of files you will allow people to upload. Here, we set this field to 1,000,000 bytes (roughly one megabyte).You may like to make it bigger or smaller for your application.
- You need an input of type **file**, shown here as
  - **<input type="file" name="userfile" id="userfile"/>**
- You can choose whatever name you like for the file, but you should keep it in mind because you will use this name to access your file from the receiving PHP script.

406

Writing the PHP to Deal with the File

- When the file is uploaded, it briefly goes into the temporary directory that is specified in your php.ini upload_tmp_dir
- The data you need to handle in your PHP script is stored in the superglobal array $_FILES .
- The entries in $_FILES will be stored with the name of the <file> tag from HTML form. In our case it is named userfile, so the array will have the following contents:
  - The value stored in $_FILES['userfile']['tmp_name'] is the place where the file has been temporarily stored on the web server.
  - The value stored in $_FILES['userfile']['name'] is the file's name on the user's system.
  - The value stored in $_FILES['userfile']['size'] is the size of the file in bytes.
  - The value stored in $_FILES['userfile']['type'] is the MIME type of the file—for example, text/plain or image/gif .
  - The value stored in $_FILES['userfile']['error'] will give you any error codes associated with the file upload.This functionality was added at PHP 4.2.0.
- We should save the file from the temporary directory (move, copy, or rename), otherwise, it will be deleted when the script ends.

407

Php file for uploading

```
<html>
	<head>
		<title>Uploading...</title>
	</head>
	<body>
		<h1>Uploading file...</h1>
		<?php
			if ($_FILES[‘userfile’][‘error’] > 0)
			{
				echo ‘Problem: ‘;
				switch ($_FILES[‘userfile’][‘error’])
				{
					case 1: echo ‘File exceeded upload_max_filesize’;
						break;
					case 2: echo ‘File exceeded max_file_size’;
						break;
					case 3: echo ‘File only partially uploaded’;
						break;
					case 4: echo ‘No file uploaded’;
						break;
					case 6: echo 'Cannot upload file: No temp directory specified';
						break;
					case 7: echo 'Upload failed: Cannot write to disk';
						break;
				}
				exit;
			}
```

408

```
			// Does the file have the right MIME type?
			if ($_FILES[‘userfile’][‘type’] != ‘text/plain’)
			{
				echo ‘Problem: file is not plain text’;
				exit;
			}
			// put the file where we’d like it
			$upfile = ‘/uploads/’.$_FILES[‘userfile’][‘name’] ;
			if (is_uploaded_file($_FILES[‘userfile’][‘tmp_name’]))
			{
				if (!move_uploaded_file($_FILES[‘userfile’][‘tmp_name’], $upfile))
				{
					echo ‘Problem: Could not move file to destination directory’;
					exit;
				}
			}
			else
			{
				echo ‘Problem: Possible file upload attack. Filename: ‘;
				echo $_FILES[‘userfile’][‘name’];
				exit;
			}
			echo ‘File uploaded successfully<br><br>’;
			// remove possible HTML and PHP tags from the file's contents
			$contents = file_get_contents($upfile);
			$contents = strip_tags($contents);
			file_put_contents($_FILES['userfile']['name'], $contents);
			// show what was uploaded
			echo '<p>Preview of uploaded file contents:<br/><hr/>';
			echo nl2br($contents);
			echo ‘<br/><hr/>’;
		?>
	</body>
</html>
```

409

Uploading – error checking

- For checking the error code returned in $_FILES['userfile']['error'], A constant is also associated with each of the codes. These are:
  - UPLOAD_ERROR_OK , value 0, means no error occurred.
  - UPLOAD_ERR_INI_SIZE , value 1, means that the size of the uploaded file exceeds the maximum value specified in your php.ini file with the upload_max_filesize directive.
  - UPLOAD_ERR_FORM_SIZE , value 2, means that the size of the uploaded file exceeds the maximum value specified in the HTML form in the MAX_FILE_SIZE element.
  - UPLOAD_ERR_PARTIAL , value 3, means that the file was only partially uploaded.
  - UPLOAD_ERR_NO_FILE , value 4, means that no file was uploaded.
  - UPLOAD_ERR_NO_TMP_DIR , value 6, means that no temporary directory is specified in the php.ini (introduced in PHP 5.0.3).
  - UPLOAD_ERR_CANT_WRITE, value 7, means that writing the file to disk failed (introduced in PHP 5.1.0).

410

Useful functions and security issues

- Use the is_uploaded_file() and move_uploaded_file() functions to make sure that the file you are processing has actually been uploaded and is not a local file.
- Use the basename() function to modify the names of incoming files.
  - This function will strip off any directory paths that are passed in as part of the filename, which is a common attack that is used to place a file in a different directory on your server.

```
<?php
$path = "/home/httpd/html/index.php";
$file1 = basename($path);
$file2 = basename($path, ".php");
print $file1 . "<br/>"; // the value of $file1 is "index.php"
print $file2 . "<br/>"; // the value of $file2 is "index"
```

411

Useful functions and security issues

- The function opendir() opens a directory for reading. Its use is similar to the use of fopen() for reading from files. Instead of passing it a filename, you should pass it a directory name:
  - $dir = opendir($current_dir);
- When the directory is open, you can read a filename from it by calling readdir($dir)
  - while(false !== ($file = readdir($dir)))
  - When finished reading from a directory, call closedir($dir)

412

Using Directory – cont.

- As an alternative to these functions, you can use the dir class provided by PHP.
- It has the properties handle and path, and the methods read(), close(), and rewind(), which perform identically to the nonclass alternatives.

```
<?php
    $dir = dir("/uploads/");
    echo "<p>Handle is $dir->handle</p>";
    echo "<p>Upload directory is $dir->path</p>";
    echo '<p>Directory Listing:</p><ul>';
    while(false !== ($file = $dir->read()))
        //strip out the two entries of . and ..
        if($file != "." && $file != "..")
        {
            echo "<li>$file</li>";
        }
    echo '</ul>';
    $dir->close();
?>
```

413

Using Directory – cont.

- The function **scandir**() was introduced in PHP 5. It store the filenames in an array and sort them in alphabetical order, either ascending or descending.

```
$files1 = scandir($dir);
$files2 = scandir($dir, 1);
```

- The disk_free_space($path) function returns the number of bytes free on the disk (Windows) or the file system (Unix) on which the directory is located.
- Creating and removing directories

```
$oldumask = umask(0);  //change umask. The inverted umask is ANDed with permissions of the mkdir
mkdir("c:\\tmp\\testing", 0777);  //create directory with permisions of (0777 AND invert(umask))
umask($oldumask);
rmdir("c:\\tmp\\testing");  //remove directory
```

414

XML Introduction

- SGML is a meta-markup language
  - Developed in the early 1980s; ISO std. In 1986
- HTML was developed using SGML in the early 1990s - specifically for Web documents
  - *Two problems with HTML*:
    1. Fixed set of tags and attributes
       - User cannot define new tags or attributes
       - So, the given tags must fit every kind of document, and the tags cannot connote any particular meaning
    2. There are few restrictions on arrangement or order of tag appearance in a document
- One solution to the first of these problems:
  Let each group of users define their own tags (with implied meanings)
  (i.e., design their own "HTML"s using SGML)

415

XML Introduction (continued)

- ***Problem with using SGML***:
  - **It's too large and complex to use, and it is very difficult to build a parser for it**
- ***A better solution*: Define a lite version of SGML**
- **XML is not a replacement for HTML**
  - **HTML is a markup language used to describe the layout of any kind of information**
  - **XML is a meta-markup language that can be used to define markup languages that can define the meaning of specific kinds of information**
- **XML is a very simple and universal way of storing and transferring data of any kind**
- **XML does not predefine any tags**
- **XML has no hidden specifications**
- **All documents described with an XML-derived markup language can be parsed with a single parser**

416

XML Introduction (continued)

- We will refer to an XML-based markup language as a *tag set*
- Strictly speaking, a tag set is an *XML application*, but that terminology can be confusing
- An *XML processor* is a program that parses XML documents and provides the parts to an application
- A document that uses an XML-based markup language is an *XML document*

7.2 The Syntax of XML

- The syntax of XML is in two distinct levels:
  1. The general low-level rules that apply to all XML documents
  2. For a particular XML tag set, either a document type definition (DTD) or an XML schema

417

The Syntax of XML (continued)

- *General XML Syntax*
- XML documents consist of:
  1. data elements
  2. markup declarations (instructions for the XML parser)
  3. processing instructions (for the application program that is processing the data in the document)

- All XML documents begin with an XML declaration:

```
<?xml version = "1.0" encoding = "utf-8"?>
```

- *XML names*:
  - Must begin with a letter or an underscore
  - They can include digits, hyphens, and periods
  - There is no length limitation
  - They are case sensitive (unlike HTML names)

418

The Syntax of XML (continued)

- *Syntax rules for XML*: same as those of XHTML
  - Every XML document defines a single root element, whose opening tag must appear as the first line of the document
- An XML document that follows all of these rules is *well formed*

```
<?xml version = "1.0" encoding = "utf-8" ?>
<ad>
  <year> 1960 </year>
  <make> Cessna </make>
  <model> Centurian </model>
  <color> Yellow with white trim </color>
  <location>
    <city> Gulfport </city>
    <state> Mississippi </state>
  </location>
</ad>
```

419

7.2 The Syntax of XML (continued)

- Attributes are not used in XML the way they are in HTML

  - In XML, you often define a new nested tag to provide more info about the content of a tag

  - Nested tags are better than attributes, because attributes cannot describe structure and the structural complexity may grow

  - Attributes should always be used to identify numbers or names of elements (like HTML `id` and `name` attributes)

420

7.2 The Syntax of XML (continued)

```
<!-- A tag with one attribute -->
<patient name = "Samira Sami Sameer">
   ...
</patient>

<!-- A tag with one nested tag -->
<patient>
   <name> Samira Sami Sameer </name>
   ...
</patient>

<!-- A tag with one nested tag, which contains
      three nested tags -->
<patient>
   <name>
      <first> Samira </first>
      <middle> Sami </middle>
      <last> Sameer </last>
   </name>
   ...
</patient>
```

421

XML Document Structure

- **An XML document often uses two auxiliary files:**
  - **One to specify the structural syntactic rules**
  - **One to provide a style specification**

- **An XML document has a single root element, but often consists of one or more entities**
  - **Entities range from a single special character to a book chapter**
  - **An XML document has one *document entity***

- ***Reasons for entity structure*:**
  1. **Large documents are easier to manage**
  2. **Repeated entities need not be literally repeated**
  3. **Binary entities can only be referenced in the document entities (XML is all text!)**

422

XML Document Structure (continued)

- When the XML parser encounters a reference to a non-binary entity, the entity is merged in
- *Entity names*:
  - No length limitation
  - Must begin with a letter, a dash, or a colon
  - Can include letters, digits, periods, dashes, underscores, or colons

- A reference to an entity has the form:
  `&entity_name;`

- Predefined entities (as in XHTML):

<	`<`
>	`>`
&	`&`
"	`&quot;`
'	`&apos;`

423

XML Document Structure (continued)

- If several predefined entities must appear near each other in a document, it is better to avoid using entity references
  - Character data section

```
<![CDATA[ content ]]>
```

e.g., instead of

```
Start &gt; &gt; &gt; &gt; HERE
                  &lt; &lt; &lt; &lt;
```

use

```
<![CDATA[Start >>>> HERE <<<<]]>
```

- If the CDATA content has an entity reference, it is taken literally

424

Well formed XML confirms with XML general rules

- XML general syntax rules, like:
  - it must begin with the XML declaration
  - it must have one unique root element
  - start-tags must have matching end-tags
  - elements are case sensitive
  - all elements must be closed
  - all elements must be properly nested
  - all attribute values must be quoted
  - entities must be used for special characters
- Even if documents are well-formed they can still contain errors, and those errors can have serious consequences.

425

7.4 Document Type Definitions

- A DTD is a set of structural rules called *declarations*
  - These rules specify a set of elements, along with how and where they can appear in a document
- Purpose: provide a standard form for a collection of XML documents and define a markup language for them
- The DTD for a document can be internal or external
- All of the declarations of a DTD are enclosed in the block of a `DOCTYPE` markup declaration
- DTD declarations have the form:
  `<!keyword … >`
- There are four possible declaration keywords:
  `ELEMENT`, `ATTLIST`, `ENTITY`, and `NOTATION`

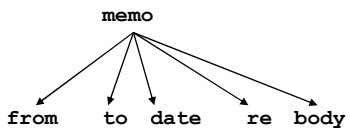
426

Document Type Definitions (continued)

- *Declaring Elements*
- An element declaration specifies the name of an element, and the element's structure
  - If the element is a leaf node of the document tree, its structure is in terms of characters
  - If it is an internal node, its structure is a list of children elements (either leaf or internal nodes)

  - General form:
    `<!ELEMENT` element_name `(`list of child names`)>`
  e.g.,

```
<!ELEMENT memo (from, to, date, re, body)>
```



427

Document Type Definitions (continued)

- *Declaring Elements* (continued)
  - Child elements can have modifiers, `+`, `*`, `?`
    e.g.,

```
<!ELEMENT person
        (parent+, age, spouse?, sibling*)>
```

  - Leaf nodes specify data types, most often `PCDATA`, which is an acronym for *p*arsable character *data*
    - Data type could also be `EMPTY` (no content) and `ANY` (can have any content)
    - Example of a leaf declaration:

```
<!ELEMENT name (#PCDATA)>
```

- *Declaring Attributes*
  - General form:

```
<!ATTLIST  el_name  at_name  at_type  [default]>
```

428

Document Type Definitions (continued)

- *Declaring Attributes* (continued)
  - *Attribute types*: there are ten different types, but we will consider only `CDATA`
  - *Default values*:
    a value
    `#FIXED` value (every element will have this value),
    `#REQUIRED` (every instance of the element must have a value specified), or
    `#IMPLIED` (no default value and need not specify a value)

```
<!ATTLIST car doors CDATA "4">
<!ATTLIST car engine_type CDATA #REQUIRED>
<!ATTLIST car price CDATA #IMPLIED>
<!ATTLIST car make CDATA #FIXED "Ford">

<car doors = "2" engine_type = "V8">
 ...
</car>
```

429

Document Type Definitions (continued)

- *Declaring Entities*
  - *Two kinds*:
    - A *general entity* can be referenced anywhere in the content of an XML document
    - A *parameter entity* can be referenced only in a markup declaration

- *General form of declaration*:
  `<!ENTITY [%] entity_name "entity_value">`
  e.g., `<!ENTITY jfk "John Fitzgerald Kennedy">`
  - A reference: `&jfk;`
  - If the entity value is longer than a line, define it in a separate file (an *external text entity*)

  `<!ENTITY entity_name SYSTEM "file_location">`

→ SHOW `planes.dtd`

430

7.4 Document Type Definitions (continued)

- *XML Parsers*
  - Always check for well formedness
  - Some check for validity, relative to a given DTD
    - Called *validating XML parsers*
  - You can download a validating XML parser from:
    `http://xml.apache.org/xerces-j/index.html`

- Internal DTDs

```
<!DOCTYPE root_name [
  …
]>
```

- External DTDs

`<!DOCTYPE` *XML_doc_root_name* `SYSTEM` *"DTD_file_name"*`>`

431

XML Schema

- Like DTD, an XML Schema describes the structure of an XML document.
  - But more powerful with many added features.
- The XML Schema language is also referred to as XML Schema Definition (XSD)
- The purpose of an XML Schema is to define the legal building blocks of an XML document:
  - the elements and attributes that can appear in a document
  - the number of (and order of) child elements
  - data types for elements and attributes
  - default and fixed values for elements and attributes

432

XML Schemas

- *Problems with DTDs*:
  - Syntax is different from XML - cannot be parsed with an XML parser.
  - DTDs do not allow specification of particular kinds of data
- XML Schema has *Two purposes*:
  - Specify the structure of its instance XML documents
  - Specify the data type of every element and attribute of its instance XML documents
- An XML schema is an XML document, so it can be parsed with an XML parser.
- The content of a specific element can be required to be any one of 44 different data types.
- The user can define new types with constraints **on existing data** types.
  - For example, a numeric data value can be required to have exactly seven digits.

433

XML Schema features

- Supports Data Types, this allows to
  - describe allowable document content
  - validate the correctness of data
  - define data facets (restrictions on data)
  - define data patterns (data formats)
  - convert data between different data types
- Uses XML Syntax, this allows to
  - Use same parser for schema and xml document
  - Manipulate schema with XML DOM
  - Reuse schema in other schemas

434

XML documents

- XML documents can have a reference to a DTD or to an XML Schema.
- Look at this simple XML document called "note.xml " with no definition reference

```
<?xml version="1.0"?>
<note>
  <to>Ahmad</to>
  <from>Sami</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

- The following example is a DTD file called "note.dtd" that defines the elements of the XML document above ("note.xml") and to the right is the note.xml file after adding a reference to the DTD file.:

```
<!ELEMENT note (to, from, heading, body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
```

```
<?xml version="1.0"?>

<!DOCTYPE note SYSTEM
"https://www.w3schools.com/xml/note.dtd">

<note>
  <to>Ahmad</to>
  <from>Sami</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

435

XML documents – cont.

- The following example on the left is an XML Schema file called "note.xsd" that defines the elements of the XML document above ("note.xml"):
- To the right is the XML document with a reference to an XML Schema

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="https://eng.najah.edu/webclass"
xmlns="https://eng.najah.edu/webclass"
elementFormDefault="qualified">

<xs:element name="note">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="to" type="xs:string"/>
      <xs:element name="from" type="xs:string"/>
      <xs:element name="heading" type="xs:string"/>
      <xs:element name="body" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

</xs:schema>
```

```
<?xml version="1.0"?>

<note
xmlns="https://eng.najah.edu/webclass"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="https://eng.najah.edu/webclass note.xsd">
  <to>Ahmad</to>
  <from>Sami</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

436

note.xml and note.xsd

- The note element is a **complex type** because it contains other elements.
- The other elements (to, from, heading, body) are **simple types** because they do not contain other elements. You will learn more about simple and complex types in the following chapters.

437

XSD - The <schema> Element

- The <schema> element is the root element of every XML Schema.
- The <schema> element may contain some attributes. A schema declaration often looks something like this:

```
<?xml version="1.0"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace=" https://eng.najah.edu/webclass "
xmlns=" https://eng.najah.edu/webclass "
elementFormDefault="qualified">
...
...
</xs:schema>
```

- xmlns:xs="http://www.w3.org/2001/XMLSchema"
  - indicates that the elements and data types used in the schema come from the "http://www.w3.org/2001/XMLSchema" namespace.
  - It also specifies that the elements and data types that come from the "http://www.w3.org/2001/XMLSchema" namespace should be prefixed with xs:
- targetNamespace="https://eng.najah.edu/webclass"
  - indicates that the elements defined by this schema (note, to, from, heading, body.) come from the "https://eng.najah.edu/webclass" namespace.
- xmlns="https://eng.najah.edu/webclass"
  - indicates that the default namespace is "https://eng.najah.edu/webclass".
- elementFormDefault="qualified“
  - indicates that any elements used by the XML instance document which were declared in this schema must be namespace qualified. Means all attributes and nested elements come from this same name space.

438

Referencing a Schema in an XML Document

- This XML document has a reference to an XML Schema

```
<?xml version="1.0"?>

<note xmlns="https://eng.najah.edu/webclass"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="https://eng.najah.edu/webclass note.xsd">

<to>Ahmad</to>
<from>Sami</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

- xmlns="https://eng.najah.edu/webclass"
  - specifies the default namespace declaration. This declaration tells the schema-validator that all the elements used in this XML document are declared in the "https://eng.najah.edu/webclass" namespace.
- xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  - This to indicate that the following xml document uses an xml schema instance namespace.
- xsi:schemaLocation=" https://eng.najah.edu/webclass note.xsd"
  - you can use the schemaLocation attribute. This attribute has two values, separated by a space. The first value is the namespace to use. The second value is the location of the XML schema to use for that namespace

439

XSD Simple Elements

- A simple element is an XML element that contains only Data. It cannot contain any other elements or attributes.
- The Data can be of many different types. It can be one of the types included in the XML Schema definition (boolean, string, date, etc.), or it can be a custom type that you can define yourself.
- You can also add restrictions (facets) to a data type in order to limit its content, or you can specify that the data has to match a specific pattern.
- The syntax for defining a simple element is:
  - `<xs:element name="xxx" type="yyy"/>`
  - where xxx is the name of the element and yyy is the data type of the element.
- XML Schema has a lot of built-in data types. The most common types are:
  - xs:string
  - xs:decimal
  - xs:integer
  - xs:boolean
  - xs:date
  - xs:time

Example XML elements

```
<lastname>Sultan</lastname>
<applicationnumber>22</applicationnumber>
<dateborn>2000-07-22</dateborn>
```

Example XML elements XML schema definition

```
<xs:element name="lastname" type="xs:string"/>
<xs:element name="applicationnumber" type="xs:integer"/>
<xs:element name="dateborn" type="xs:date"/>
```

440

Default and Fixed Values for Simple Elements

- Simple elements may have a default value OR a fixed value specified.
  - A default value is automatically assigned to the element when no other value is specified.

```
<xs:element name="color" type="xs:string" default="red"/>
```

  - A fixed value is also automatically assigned to the element, and you cannot specify another value.

```
<xs:element name="color" type="xs:string" fixed="red"/>
```

441

XSD Attributes

- Simple elements cannot have attributes. If an element has attributes, it is considered to be of a complex type. But the attribute itself is always declared as a simple type.
- The syntax for defining an attribute is
  - `<xs:attribute name="xxx" type="yyy"/>`
  - where xxx is the name of the attribute and yyy specifies the data type of the attribute.
- XML Schema has a lot of built-in data types. The most common types are:
  - xs:string
  - xs:decimal
  - xs:integer
  - xs:boolean
  - xs:date
  - xs:time

XML element with an attribute

```
<lastname lang="EN">Smith</lastname>
```

the corresponding attribute definition

```
<xs:attribute name="lang" type="xs:string"/>
```

442

Default, Fixed, Optional and Required Attributes

- A default value is automatically assigned to the attribute when no other value is specified.
  - `<xs:attribute name="lang" type="xs:string" default="EN"/>`
- A fixed value is also automatically assigned to the attribute, and you cannot specify another value.
  - `<xs:attribute name="lang" type="xs:string" fixed="EN"/>`
- Attributes are optional by default. To specify that the attribute is required, use the "use" attribute:
  - `<xs:attribute name="lang" type="xs:string" use="required"/>`
- If an XML element is of type "xs:date" and contains a string like "Hello World", the element will not validate.
- With XML Schemas, you can also add your own restrictions to your XML elements and attributes. These restrictions are called facets.

443

XSD Restrictions/Facets

- With XML Schemas, you can also add your own restrictions to your XML elements and attributes. These restrictions are called facets.

Restrictions on Values

```
<xs:element name="age">
  <xs:simpleType>
    <xs:restriction base="xs:integer">
      <xs:minInclusive value="0"/>
      <xs:maxInclusive value="120"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Restrictions on a Set of Values

To limit the content of an XML element to a set of acceptable values, we would use the enumeration constraint.

```
<xs:element name="car">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="Audi"/>
      <xs:enumeration value="Golf"/>
      <xs:enumeration value="BMW"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

```
<xs:element name="car" type="carType"/>

<xs:simpleType name="carType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Audi"/>
    <xs:enumeration value="Golf"/>
    <xs:enumeration value="BMW"/>
  </xs:restriction>
</xs:simpleType>
```

Note: In this case the type "carType" can be used by other elements because it is not a part of the "car" element.

444

XSD Restrictions/Facets – cont.

445

XSD Restrictions/Facets – cont.

The following defines an element called "initials" with a restriction. The only acceptable value is THREE of the LOWERCASE OR UPPERCASE letters from a to z

```
<xs:element name="initials">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="[a-zA-Z][a-zA-Z][a-zA-Z]"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

The example below defines an element called "letter" with a restriction. The acceptable value is zero or more occurrences of lowercase letters from a to z. You can use + to indicate one or more. You can also use {} to define an exact number of occurrences.

```
<xs:element name="letter">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="([a-z])*"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

The next example defines an element called "gender" with a restriction. The only acceptable value is male OR female

```
<xs:element name="gender">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="male|female"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

446

XSD Restrictions/Facets – cont.

This example defines an element called "address" with a restriction. The whiteSpace constraint is set to "preserve", which means that the XML processor WILL NOT remove any white space characters

```
<xs:element name="address">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:whiteSpace value="preserve"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

The whiteSpace constraint is set to "replace", which means that the XML processor WILL REPLACE all white space characters (line feeds, tabs, spaces, and carriage returns) with spaces. You may also set it to "collapse", which means that the XML processor WILL REMOVE all white space characters (line feeds, tabs, spaces, carriage returns are replaced with spaces, leading and trailing spaces are removed, and multiple spaces are reduced to a single space)

```
<xs:element name="address">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:whiteSpace value="replace"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

447

XSD Restrictions/Facets – cont.

You can use any of the following restrictions, you can find more examples on these on https://www.w3schools.com/

Constraint	Description
enumeration	Defines a list of acceptable values
fractionDigits	Specifies the maximum number of decimal places allowed. Must be equal to or greater than zero
length	Specifies the exact number of characters or list items allowed. Must be equal to or greater than zero
maxExclusive	Specifies the upper bounds for numeric values (the value must be less than this value)
maxInclusive	Specifies the upper bounds for numeric values (the value must be less than or equal to this value)
maxLength	Specifies the maximum number of characters or list items allowed. Must be equal to or greater than zero
minExclusive	Specifies the lower bounds for numeric values (the value must be greater than this value)
minInclusive	Specifies the lower bounds for numeric values (the value must be greater than or equal to this value)
minLength	Specifies the minimum number of characters or list items allowed. Must be equal to or greater than zero
pattern	Defines the exact sequence of characters that are acceptable
totalDigits	Specifies the exact number of digits allowed. Must be greater than zero
whiteSpace	Specifies how white space (line feeds, tabs, spaces, and carriage returns) is handled

448

XSD Complex Elements

- A complex element is an XML element that contains other elements and/or attributes.
- There are four kinds of complex elements:
  - empty elements
    - `<product pid="1345"/>`
  - elements that contain only other elements
    - ```
      <employee>
        <firstname>Ahmad</firstname>
        <lastname>Sami</lastname>
      </employee>
      ```
  - elements that contain only text
    - `<food type="dessert">Ice cream</food>`
  - elements that contain both other elements and text
    - ```
      <description>
      It happened on <date lang="english">03.03.99</date> ....
      </description>
      ```
  - **Note**: Each of these elements may contain attributes as well!

449

XSD Complex Elements – cont.

- Consider the following element
  - <employee>
    <firstname>Ahmad</firstname>
    <lastname>Smith</lastname>
    </employee>
- We can define a complex element in an XML Schema two different ways
  - The "employee" element can be declared directly by naming the element

```
<xs:element name="employee">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="firstname" type="xs:string"/>
      <xs:element name="lastname" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

    - The <sequence> indicator means that the child elements must appear in the same order as they are declared.
  - The "employee" element can have a type attribute that refers to the name of the complex type to use

```
<xs:element name="employee" type="personinfo"/>

<xs:complexType name="personinfo">
  <xs:sequence>
    <xs:element name="firstname" type="xs:string"/>
    <xs:element name="lastname" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

If you use the method described above, several elements can refer to the same complex type, like this

```
<xs:element name="employee" type="personinfo"/>
<xs:element name="student" type="personinfo"/>
<xs:element name="member" type="personinfo"/>

<xs:complexType name="personinfo">
  <xs:sequence>
    <xs:element name="firstname" type="xs:string"/>
    <xs:element name="lastname" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

450

XSD Complex Elements – cont.

- You can also base a complex type on an existing complex type and add some elements, like this

```
<xs:element name="employee" type="fullpersoninfo"/>

<xs:complexType name="personinfo">
  <xs:sequence>
    <xs:element name="firstname" type="xs:string"/>
    <xs:element name="lastname" type="xs:string"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="fullpersoninfo">
  <xs:complexContent>
    <xs:extension base="personinfo">
      <xs:sequence>
        <xs:element name="address" type="xs:string"/>
        <xs:element name="city" type="xs:string"/>
        <xs:element name="country" type="xs:string"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

451

Complex Empty Elements

- An empty XML element
  - `<product prodid="1345" />`
- The "product" element above has no content at all. To define a type with no content, we must define a type that allows elements in its content, but we do not actually declare any elements
-

```
<xs:element name="product">
  <xs:complexType>
    <xs:complexContent>
      <xs:restriction base="xs:integer">
        <xs:attribute name="prodid" type="xs:positiveInteger"/>
      </xs:restriction>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
```

It is possible to declare the "product" element more compactly, like this

```
<xs:element name="product">
  <xs:complexType>
    <xs:attribute name="prodid" type="xs:positiveInteger"/>
  </xs:complexType>
</xs:element>
```

452

XSD Text-Only Elements

- <shoesize country="france">35</shoesize>
-

```
<xs:element name="shoesize">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:integer">
        <xs:attribute name="country" type="xs:string" />
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
```

-

453

Complex Types with Mixed Content

- An XML element, "letter", that contains both text and other elements:

```
<letter>
  Dear Mr. <name>Ahmad Sami</name>.
  Your order <orderid>1032</orderid>
  will be shipped on <shipdate>2022-07-13</shipdate>.
</letter>
```

- The following schema declares the "letter" element:

```
<xs:element name="letter">
  <xs:complexType mixed="true">
    <xs:sequence>
      <xs:element name="name" type="xs:string"/>
      <xs:element name="orderid" type="xs:positiveInteger"/>
      <xs:element name="shipdate" type="xs:date"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

- **Note:** To enable character data to appear between the child-elements of "letter", the mixed attribute must be set to "true". The <xs:sequence> tag means that the elements defined (name, orderid and shipdate) must appear in that order inside a "letter" element.

454

XSD Indicators

- We can control HOW elements are to be used in documents with indicators.
- Order indicators:
  - All
  - Choice
  - Sequence
- Occurrence indicators:
  - maxOccurs
  - minOccurs
- Group indicators:
  - Group name
  - attributeGroup name

```
<xs:element name="person">
  <xs:complexType>
    <xs:choice>
      <xs:element name="employee" type="employee"/>
      <xs:element name="member" type="member"/>
    </xs:choice>
  </xs:complexType>
</xs:element>
```

```
<xs:element name="person">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="full_name" type="xs:string"/>
      <xs:element name="child_name" type="xs:string"
      maxOccurs="10" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

455

```
<xs:group name="persongroup">
  <xs:sequence>
    <xs:element name="firstname" type="xs:string"/>
    <xs:element name="lastname" type="xs:string"/>
    <xs:element name="birthday" type="xs:date"/>
  </xs:sequence>
</xs:group>

<xs:element name="person" type="personinfo"/>

<xs:complexType name="personinfo">
  <xs:sequence>
    <xs:group ref="persongroup"/>
    <xs:element name="country" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

```
<xs:attributeGroup name="personattrgroup">
  <xs:attribute name="firstname" type="xs:string"/>
  <xs:attribute name="lastname" type="xs:string"/>
  <xs:attribute name="birthday" type="xs:date"/>
</xs:attributeGroup>

<xs:element name="person">
  <xs:complexType>
    <xs:attributeGroup ref="personattrgroup"/>
  </xs:complexType>
</xs:element>
```

456

Data Types

- Data Type Categories
  - Simple (strings only, no attributes and no nested elements)
  - Complex (can have attributes and nested elements)
- XMLS defines over 44 data types:
  - 19 Primitive: string, Boolean, float, …
  - 25 Derived: byte, decimal, positiveInteger, …
- User-defined (derived) data types
  - *specify constraints on an existing type (the base type)*
- Constraints are given in terms of facets (aspects or features)
  - Example: the integer primitive data type has eight possible facets: totalDigits, maxInclusive, maxExclusive,minInclusive, min Exclusive, pattern, enumeration, and whitespace.

466

Data Types – cont.

- Elements in a DTD are all global.
  - Each has a unique name and is defined exactly once.
- Data declarations in an XML schema can be either local or global.
- A *local declaration* is a declaration that appears inside an element
  - A locally declared element is visible only in that element.
  - local elements with the same name can appear in any number of different elements with no interference among them.
- A *global declaration* is a declaration that appears as a child of the schema element.
  - Visible in the whole schema in which they are declared.

467

Simple Types

- Elements are defined in an XML schema with the element tag, which is from the XMLSchema namespace.
  - Normally prefixed with xsd
- An element xml schema tag includes the name attribute and, for a simple element declaration, the type attribute, which specifies the type of content allowed in the element.
  - <xsd:element name = “engine” type = “xsd:string” />
- An instance of the schema in which the engine element is defined could have the following element:
  - <engine> inline six cylinder fuel injected </engine>
- An element can be constant or given a default value with the fixed and the default attributes respectively.

```
<xsd:element name = "engine"   type = "xsd:string"
             default = "fuel injected V-6"  />

<xsd:element name = "plane"   type = "xsd:string"
             fixed = "single wing"  />
```

468

User-defined data types

- Defined in a **simpleType** element, using facets specified in the content of a restriction element
- Facet values are specified with the value attribute

```
<xsd:simpleType name = "middleName">
  <xsd:restriction base = "xsd:string" >
    <xsd:maxLength value = "20" />
  </xsd:restriction>
</xsd:simpleType>
```

```
<xsd:simpleType name = "phoneNumber">
  <xsd:restriction base = "xsd:decimal">
    <xsd:precision value = "7" />
  </xsd:restriction>
</xsd:simpleType>
```

469

Complex Types

- Complex types are defined with the complexType tag.
- The elements that nested inside an element-only element complex type must be contained in an ordered group, an unordered group, a choice, or a named group.
- The sequence element is used to contain an ordered group of elements
- Ex:

```
<xsd:complexType name = "sports_car">
  <xsd:sequence>
    <xsd:element name = "make"  type = "xsd:string" />
    <xsd:element name = "model"  type = "xsd:string" />
    <xsd:element name = "engine"  type = "xsd:string" />
    <xsd:element name = "year"  type = "xsd:decimal" />
  </xsd:sequence>
</xsd:complexType>
```

470

Complex types – cont.

- An unorderd group is defined in an all element.
- Elements in all and sequence groups can include the minOccurs and maxOccurs attributes to specify the numbers of occurrences.
- Possible values for min/maxOccurs is nonnegative integer or the keyword unbounded

471

Example

```
<?xml version = "1.0" encoding = "utf-8"?>

<!-- planes.xsd
     A simple schema for planes.xml
     -->
<xsd:schema
  xmlns:xsd = "http://www.w3.org/2001/XMLSchema"
  targetNamespace = "http://cs.uccs.edu/planeSchema"
  xmlns = "http://cs.uccs.edu/planeSchema"
  elementFormDefault = "qualified">

  <xsd:element name = "planes">
    <xsd:complexType>
      <xsd:all>
        <xsd:element name = "make"
                     type = "xsd:string"
                     minOccurs = "1"
                     maxOccurs = "unbounded" />
      </xsd:all>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

472

Example – cont.

```
<?xml version = "1.0" encoding = "utf-8"?>

<!-- planes1.xml
     A simple XML document for illustrating a schema
     The schema is in planes.xsd
     -->
<planes
  xmlns = "http://cs.uccs.edu/planeSchema"
  xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation = "http://cs.uccs.edu/planeSchema
                        planes.xsd">
    <make> Cessna </make>
    <make> Piper </make>
    <make> Beechcraft </make>
</planes>
```

473

Complex types with derived types

- You can define an element that is used inside a complex element, outside it with a reference to it.
- Example:

```
<xsd:element name = "year">
  <xsd:simpleType>
    <xsd:restriction base = "xsd:decimal">
      <xsd:minInclusive value = "1900" />
      <xsd:maxInclusive value = "2007" />
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>

<xsd:complexType name = "sports_car">
  <xsd:sequence>
    <xsd:element name = "make"  type = "xsd:string" />
    <xsd:element name = "model"  type = "xsd:string" />
    <xsd:element name = "engine"  type = "xsd:string" />
    <xsd:element ref = "year" />
  </xsd:sequence>
</xsd:complexType>
```

474

Validating Instances of Schemas

- Many programs
- Example xsv : XML Schema Validatorhttp://www.w3.org/XML/Schema#XSV
  - If the schema and the instance document are available on the Web, xsv can be used online
  - Note: If the schema is incorrect (bad format), xsv reports that it can find the schema

475

Displaying Raw XML Documents

- There is no presentation information in an XML document
- An XML browser should have a default style sheet for an XML document that does not specify one
- You get a stylized listing of the XML

476

Displaying XML Documents with CSS

- A CSS style sheet for an XML document is just a list of its tags and associated styles
- The connection of an XML document and its style sheet is made through an xml-stylesheet processing instruction
  - <?xml-stylesheet type = "text/css" href = "mydoc.css"?>
- CSS example
  - <!-- planes.css - a style sheet for the planes.xml document -->
  - ad { display: block; margin-top: 15px; color: blue;}
  - year, make, model { color: red; font-size: 16pt;}
  - color {display: block; margin-left: 20px; font-size: 12pt;}
  - description {display: block; margin-left: 20px; font-size: 12pt;}
  - seller { display: block; margin-left: 15px; font-size: 14pt;}
  - location {display: block; margin-left: 40px; }
  - city {font-size: 12pt;}
  - state {font-size: 12pt;}

477

The eXtensible Stylesheet Language (XSL)

- A recommendation for defining the presentation and transformations of XML documents.
- It consists of three related standards:
  - XSL Transformations (XSLT),
  - XML Path Language (XPath),
  - and XSL Formatting Objects (XSL-FO).
- XSLT uses style sheets to specify transformations
- An XSLT processor merges an XML document into an XSLT style sheet
  - This merging is a template-driven process
- An XSLT style sheet can specify page layout, page orientation, writing direction, margins, page numbering, etc.
- The processing instruction we used for connecting a CSS style sheet to an XML document is used to connect an XSLT style sheet to an XML document
  - <?xml-stylesheet type = "text/xsl" href = "XSLT style sheet"?>

478

Overview of Ajax

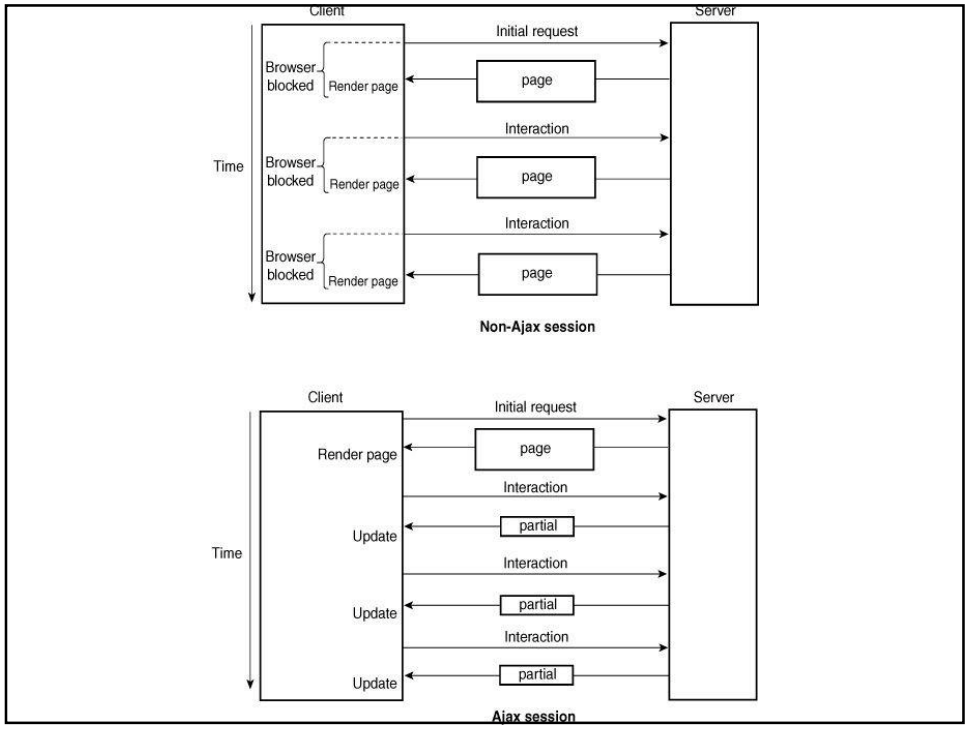
- *History*
  - Possibility began with the nonstandard iframe element, which appeared in IE4 and Netscape 4
  - An iframe element could be made invisible and could be used to send asynchronous requests
  - Microsoft introduced XmlDocument and XMLHTML ActiveX objects in IE5 – for asynchronous requests
  - A similar object is now supported by all current browsers
- Two events ignited widespread interest in Ajax:
  - The appearance of Google Maps and Google Mail
- Jesse James Garrett named the new Technology Ajax
- Goal of Ajax is to provide Web-based applications with responsiveness approaching that of desk-top applications

479

Overview of Ajax (continued)

- Specific kind of Web applications that benefit from Ajax are those that have frequent interactions between the client and the server
- Goals are achieved with two different approaches:
  - Client requests are handled asynchronously
  - Only small parts of the current document are updated
- Ajax does not use any new programming languages or markup languages
  - Client side: JavaScript, XML, XHTML, DOM, CSS
  - Server side: any (PHP, servlets, ASP.NET, etc.)

480



481

How to implement AJAX

- Ajax can be implemented in several different ways.
  - It can be implemented with just the basic tools, including JavaScript on the client (browser), the XMLHttpRequest object, and virtually any server-side software, using text, XHTML, or XML to transmit data.
  - Another way to implement Ajax is with the help of a client-side toolkit, such as **Dojo or Prototype**. There are also server-side tools, such as **DWR and GWT**.
  - There also are frameworks for implementing applications that use Ajax, such as **Adobe Flex, ASP.NET Ajax, JavaServer Faces, and Rails.**
- For security reasons, Ajax requests using XMLHttpRequest can be made only to the server and site that provided the document in which the request originated.

482

Example – zip-code states

- A sales page in which a costumer fills in his area zip-code, the AJAX tries to auto fill his country and state
- The application: Helps the user fill a form
- The form gathers client information; asks for the zip code before the names of the city and state
- As soon as the zip code is entered, the application sends a request to the server, which looks up the city and state for the given zip code and returns them to the form
- Uses JavaScript to put the city and state names in the form
- Uses PHP on the server to look up the city and state
- The form Must reference the JavaScript code file in its head
- Must register an event handler on the blur event of the zip code text box
- → SHOW popcornA.html

483

Example – zip-code states – cont.

- Two functions are required by the application:
  - The blur handler
  - A function to handle the response
- The Request Phase (The blur handler)
  - The communication to the server for the asynchronous request must be made through the XMLHttpRequest object, so one must be created
    - var xhr = new XMLHttpRequest();
  - When the server receives an asynchronous request, it sends a sequence of notices, called callbacks, to the browser (0, …, 4)
  - Only the last one is of interest, 4, which indicates that the response is complete
  - The response function is what is called in the callbacks
  - The response function must be registered on the onreadystatechange property of the XHR object
    - xhr.onreadystatechange = receivePlace;

484

Example – cont – the response phase

- The Response Document
- We will use a simple hash of zip codes and names of cities and states, so this will be very simple
- The response data is produced with a print statement
- → SHOW getCityState.php
- The Receiver Phase
  - A JavaScript function with no parameters Fetch the server response (text), split it into its two parts (city and state), and set the corresponding text boxes to those values
  - The receiver function must be able to access the XHR
  - If it is global, it would be accessible, but it could be corrupted by simultaneous requests and responses
  - The alternative is to register the actual code of the receiver, rather than its name

485

Example – cont – the reciever phase

- Actions of the receiver function:
  - Put all actions in the then clause of a selector that checks to see if readyState is 4
  - Get the response value from the responseText property of the XHR object
  - Split it into its two parts
  - Set the values of the city and state text boxes
- → SHOW popcornA.js
- Cross-Browser Support
  - What we have works with FX2 and IE7, but not IE5 and IE6
  - IE5 and IE6 support an ActiveXObject named Microsoft.XMLHTTP
  - xhr = new ActiveXObject("Microsoft.XMLHTTP");
  - → SHOW getPlace2.js

486

AJAX and response types

- Server can response to AJAX request using many document forms
  - XHTML
  - XML
  - JSON
  - Plain text

487

Return Document Forms – XHTML

- Most common approach is to place an empty div element in the original document
- The innerHTML property of the div element is assigned the new content

```
<div id = "replaceable_list">
  <h2> Najah University Champion/Runnerup – basketball</h2>
    <ul>
      <li> Red eagles</li>
      <li> Rockies </li>
    </ul>
</div>
```

- Now, if the user selects a different sport, say football, the XHTML response fragment could have the following:

```
<h2> Najah University Champion/Runnerup – football </h2>
<ul>
  <li> Giants </li>
  <li> Patriots </li>
</ul>
```

488

Return Document Forms – XHTML

- Now, the returned fragment can be inserted in the div element with

```
var divDom = document.getElementById("replaceable_list");
divDom.innerHTML = xhr.responseText;
```

- The disadvantage of using XHTML for the return document is it works well only if markup is what is wanted.
- However, oftentimes, it is data that is returned, in which case it must be parsed out of the XHTML

489

Return Document Forms – XML

- For the previous example, the following would be returned:

  <header> Najah University Champion/Runnerup – football</header>
  <list_item> Giants </list_item>
  <list_item> Patriots </list_item>
- Problem: the XML returned must also be parsed
  - Two approaches:
    - Use the DOM binding parsing methods

      Two disadvantages:

      i. Writing the parsing code is tedious

      ii. Support for DOM parsing methods is a bit inconsistent over various browsers
    - Use **XSLT** style sheets For the *example*,

490

Return Document Forms – XML – XSLT

```
<xsl:stylesheet version = "1.0"
  xmlns:xsl = "http://www.w3.org/1999/XSL/Transform"
  xmlns = "http://www.w3.org/1999/xhtml" >
  <xsl:template match = "/">
  <h2> <xsl:value-of select = "header" /></h2>
     <br />
     <br />
     <ul>
       <xsl:for-each select = "list_item">
         <li> <xsl:value-of select = "list_item"/>
           <br />
         </li>
       </xsl:for-each>
     </ul>
  </xsl:template>
</xsl:stylesheet>
```

491

Return Document Forms – JSON

- *JavaScript Object Notation (JSON)*
- Part of the JavaScript standard, 3rd edition
- A method of representing objects as strings, using two structures
  - A. Collections of name/value pairs
  - B. Arrays of values

492

Return Document Forms – JSON

- Example:

```
{"employees" :
  [
    {"name" : ”Ahmad, Sami", "address" :
      "1222 Faisal St"},
    {"name" : ”Noor, Najeeb", "address" :
      "332 Main road"},
    {"name" : ”Muhanad, Dapas", "address" :
      "222 Alquds Street"}
  ]
}
```

- This object consists of one property/value pair, whose value is an array of three objects, each with two property/value pairs
- Array element access can be used to retrieve the data elements
  - `var address2 = myObj.employees[1].address;`
  - puts "`332 Main road`" in `address2`
- JSON objects are returned in `responseText`
- How to get the object, `myObj`?

493

Return Document Forms – JSON

- The object could be obtained by running eval on the response string
- This is dangerous, because the response string could have malicious code
- It is safer to get and use a JSON parser

```
var response = xhr.responseText;
var myObj = JSON.parse(response);
```

- *JSON has at least three advantages over XML*
  1. JSON representations are smaller
  2. parse is much faster than manual parsing or using XSLT
  3. parse is much easier than manual parsing or using XSLT
- XML is better if the returned data is going to be integrated with the original document – use XSLT

494

Return Document Forms – JSON

- Example return document:

```
{"top_two":
   [
     {"sport": "football", "team": "Giants"},
     {"sport": "football", "team": "Patriots"},
   ]
 }
```

- The processing to put it in the XHTML document:

```
var myObj = JSON.parse(response);
document.write("<h2> Najah Uni. Champion/Runnerup" + myObj.top_two[0].sport + "</h2>");
document.write("<ul> <li>" +  myObj.top_two[0].team + "</li>");
document.write("<li>" + myObj.top_two[1].team  + "</li></ul>");
```

495

Ajax Toolkits

- There are many toolkits to help build Ajax applications, for both server-side and client-side
- Client-side toolkits:
- 1. *Dojo*
  - A free JavaScript library of modules, for Ajax and other parts of Web site software
  - Provides commonly needed code and hides the differences among browsers
  - We will use only one function, `bind`, which creates an XHR object and builds an Ajax request
  - `bind` is part of the `io` module
  - To gain access to Dojo module, if `dojo.js` is in the `dojo` subdirectory of where the markup resides

```
<script type = "text/javascript"
  src = "dojo/dojo.js">
</script>
```

496

Ajax Toolkits – cont.

- The bind function takes a single literal object parameter
- a list of property/value pairs, separated by commas and delimited by braces properties are separated from their values by colons
- The parameter must include url and load properties
- The value of the url property is the URL of the server
- The value of the load property is an anonymous function that uses the returned data
- It also should have method, error , and mimetype properties
- The getPlace function, rewritten with Dojo's bind:
- → SHOW dojo.io.bind

497

10.4 Ajax Toolkits (continued)

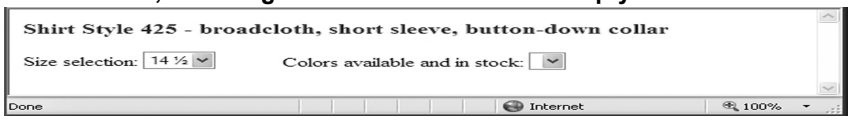
1. *Dojo* (continued)

- An example – ordering a shirt on-line

- After the user selects a size, present the user with the colors in that size that are now in stock

- Use Ajax to get the colors for the chosen size

- The original document is for one particular style of shirt, including a menu for sizes and an empty



→ **SHOW** `shirtstyles.css`

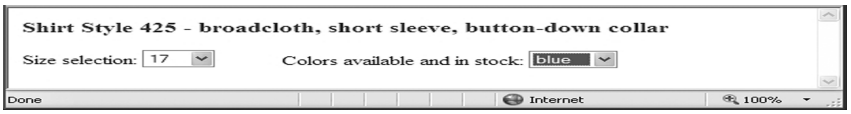
498

10.4 Ajax Toolkits (continued)

1. *Dojo* (continued)
   - The required JavaScript must define two functions
   A. `buildMenu` – the callback function to build the menu of colors
      - Get the DOM address of the empty select
      - If it is not the first request, set `options` property to zero
      - Split the returned value (a string of colors separated by commas and spaces)
      - Build the `Options` of the menu and add them to the menu with `add`
        - The second parameter to `add` is browser-dependent; for IE, it is `-1`; for others, it is `null`

   B. `getColors` – a wrapper function that calls `bind` to create the Ajax request

→ SHOW `shirt.js`



499

10.4 Ajax Toolkits (continued)

2. *Prototype*
   - A toolkit that extends JavaScript and provides tools for Ajax applications
   - Includes a large number of functions and abbreviations of commonly needed JavaScript code

```
$("name") is an abbreviation for
document.getElementById("name")
```

   - In Prototype, all of the Ajax functionality is encapsulated in the `Ajax` object
   - A request is created by creating an object of `Ajax.Request` type, sending the parameters to the constructor
     - The first parameter is the URL of the server
     - The second parameter is a literal object with the other required information:

       - `method` – `"get"` or `"post"`
       - `parameters` – what to attach to the `get`
       - `onSuccess` – the anonymous callback function to handle the return
       - `onFailure` – the anonymous callback function for failure
   - → SHOW the `Ajax.request` object creation

500

10.5 Security and Ajax

***- Issues*:**

1. **In many cases, Ajax developers put security code in the client code, but it also must be included in the server code, because intruders can change the code on the client**
2. **Non-Ajax applications often have just one or only a few server-side sources of responses, but Ajax applications often have many server-side programs that produce small amounts of data. This increases the attack surface of the whole application.**
3. **Cross-site scripting – servers providing JavaScript code as an Ajax response. Such code could be modified by an intruder before it is run on the client**
   - **All such code must be scanned before it is interpreted**
   - **Intruder code could also come to the client from text boxes used to collect return data**
     - **It could include script tags with malicious code**

501

Drag and Drop Example

- Register an Event listener
  - addEventListener(), takes three arguments
    - The name of the event
    - The function that would handle the event
    - If the event is bubble (how it is propagate (top down or bottom up (the default used in all browsers) so put it always false))
- Event names are:
  - Source
    - dragstart
    - drag
    - dragend
  - Destination
    - dragenter
    - dragleave
    - dragover *****affects the drop event
    - drop

502

Canvas

- The <canvas> tag enables you to define a blank drawing area on your web page.
- You can then draw shapes, lines, and text, insert images, create gradients, and do other interesting things using JavaScript.
- The tag accepts width and height attributes that define the size of the canvas.
- You also need to add an id attribute so your scripts can draw on the canvas.
- Next step in all canvas tasks is setting up the drawing context. This prepares your JavaScript code to access all the instructions it needs for adding shapes, lines, and other elements to your canvas.
  - Done using getContext method
- Then you need to choose a color and an outline width for drawing.

503

```
<!DOCTYPE html>
<html>
<head>
<title>HTML5 Canvases</title>
<style>
canvas#mycanvas { border: solid gray 1px }
</style>
</head>
<body>

<script>

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

504

Should the script be placed after canvas?

- To successfully draw on your canvas using JavaScript, your HTML5 canvas has to be loaded prior to your script code running.
- You can do this by placing your script after your canvas on your page, or place it in a JavaScript function which you can then call using the onload attribute in the <body> tag.

canvas-onload.html - Notepad

File Edit Format View Help

```
<!DOCTYPE html>
<html>
<head>
<title>HTML5 Canvas</title>
<style>
canvas#mycanvas { border: solid gray 1px }
</style>

<script>
function drawCanvas() {
var canv = document.getElementById('mycanvas');
var ctx = canv.getContext('2d');
}
</script>

</head>
<body onload="drawCanvas();">

<canvas id="mycanvas" width="600" height="400"></canvas>

</body>
</html>
```

505

Draw Rectangles

canvas-rect.html - Notepad

File Edit Format View Help

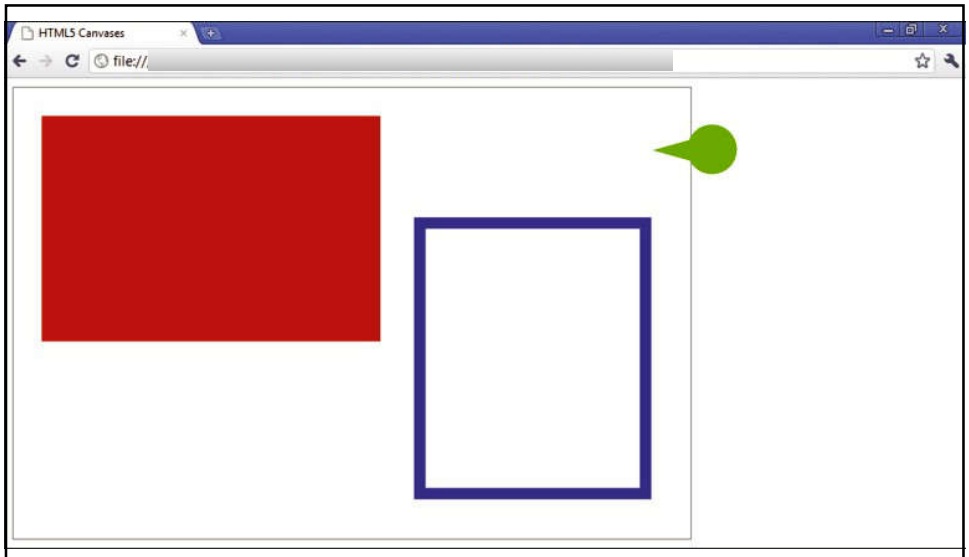
```
<!DOCTYPE html>
<html>
<head>
<title>HTML5 Canvases</title>
<style>
canvas#mycanvas { border: solid gray 1px }
</style>
</head>
<body>

<canvas id="mycanvas" width="600"

<script>
var cnv = document.getElementById
var ctx = cnv.getContext('2d');

</body>
</html>
```

canvas-rect.html - Notepad

File Edit Format View Help

```
<!DOCTYPE html>
<html>
<head>
<title>HTML5 Canvases</title>
<style>
canvas#mycanvas { border: solid gray 1px }
</style>
</head>
<body>

<canvas id="mycanvas" width="600" height="400"></canvas>

<script>
var cnv = document.getElementById('mycanvas');
var ctx = cnv.getContext('2d');
ctx.fillStyle = 'red';
ctx.fillRect(25, 25, 300, 200);

</body>
</html>
```

506

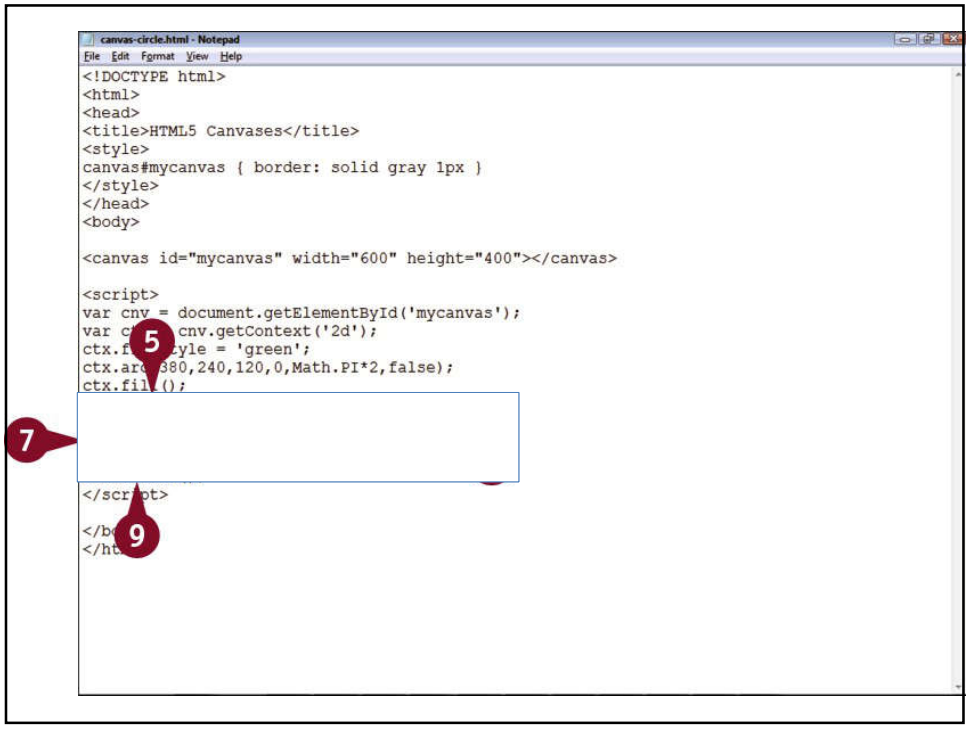


507

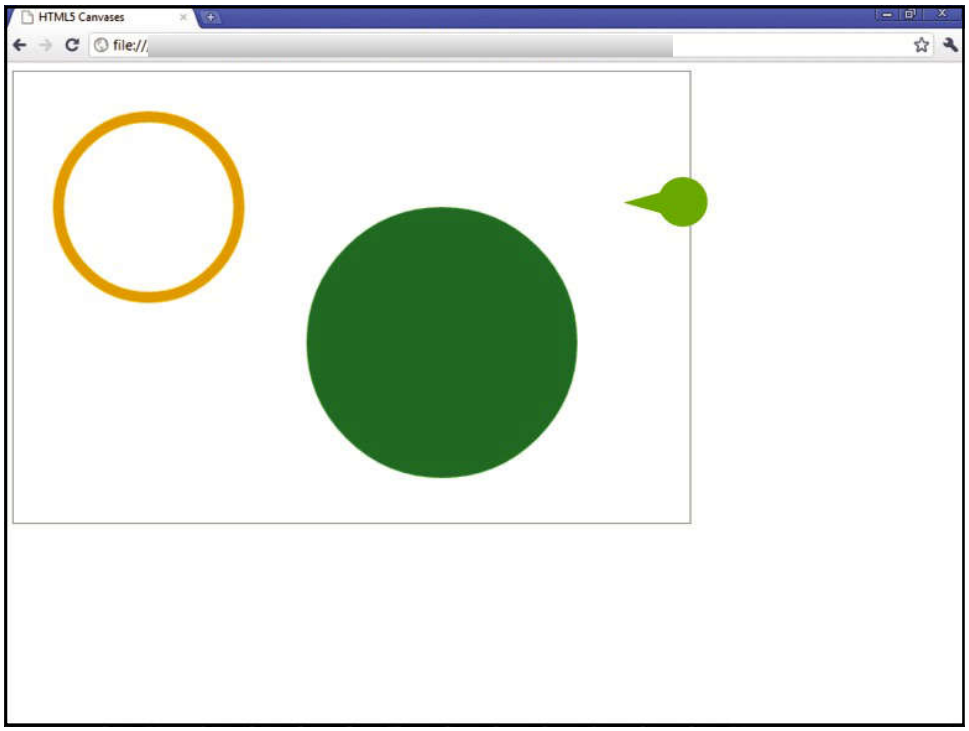
Draw Circles

- For creating a circle,
  - you define a center coordinate
  - a radius
  - you also define how far along the circumference of the circle to draw.
    - You define this value in radians, and to draw a complete circle this value is two times the value of pi.
- Use arc(number, number, number ,start_angle,end_angle, boolean);
  - The first two numbers define the coordinates of the circle center and
  - The third number defines the radius of the circle in pixels.
  - To draw a circle set the starting angle to 0 and the ending angle to Math.PI*2.
  - A true value draws the circle in a clockwise direction, and false draws it in a counterclockwise direction.

508


canvas-circle.html - Notepad
File Edit Format View Help
<!DOCTYPE html>
<html>
<head>
<title>HTML5 Canvases</title>
<style>
canvas#mycanvas { border: solid gray 1px }
</style>
</head>
<body>
<canvas id="mycanvas" width="600" height="400"></canvas>
<script>
var cnv = document.getElementById('mycanvas');
cnv.getContext('2d');
ctx.arc(380,240,120,0,Math.PI*2,false);
ctx.fill();
</script>
5
7
9

509


HTML5 Canvases
file://

510

Draw Lines

- You create a line by specifying the coordinate at which to start the line using the moveTo method.
- Then you specify an ending point with the lineTo method.
- Calling the stroke method draws the line.

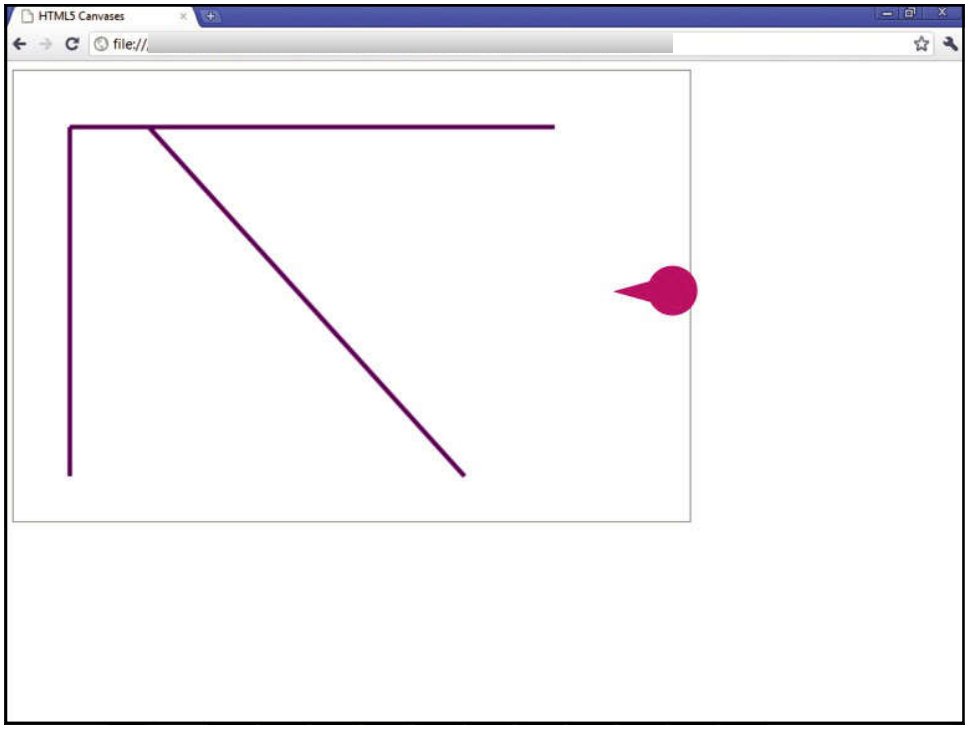
511

canvas-lines.html - Notepad

File Edit Format View Help

```
<!DOCTYPE html>
<html>
<head>
<title>HTML5 Canvases</title>
<style>
canvas#mycanvas { border: solid gray 1px }
</style>
</head>
<body>

<canvas id="mycanvas" width="600" height="400"></canvas>

<script>
var cnv = document.getElementById('mycanvas');
var ctx = cnv.getContext('2d');
</script>

</body>
</html>
```

512



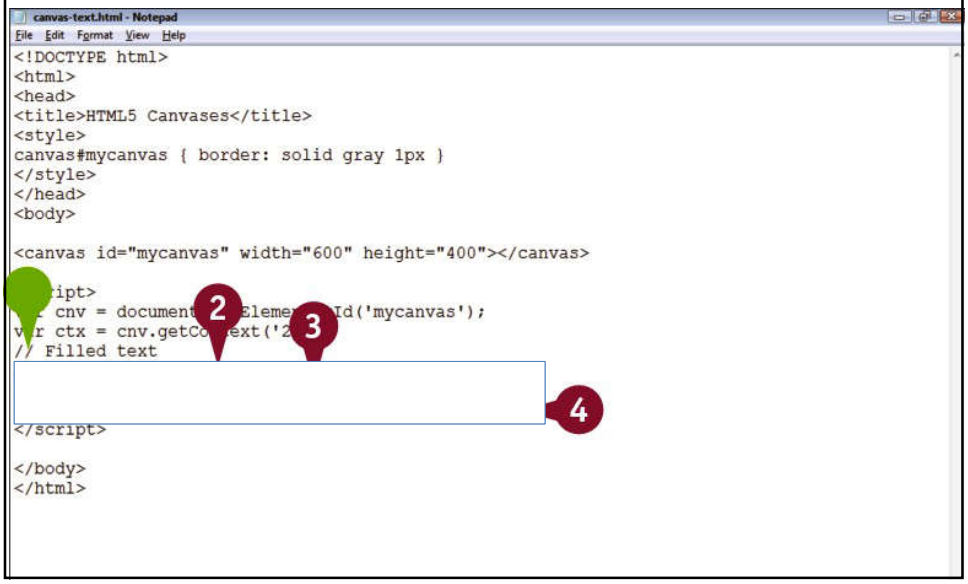
513

Add Text

- You can set a font family and size using the font method in your script.
- You can create filled text using the fillText method or outlined text using the strokeText method.
- To add extra styling to your text you can rotate it. For details, see ?Rotate Canvas Content.?

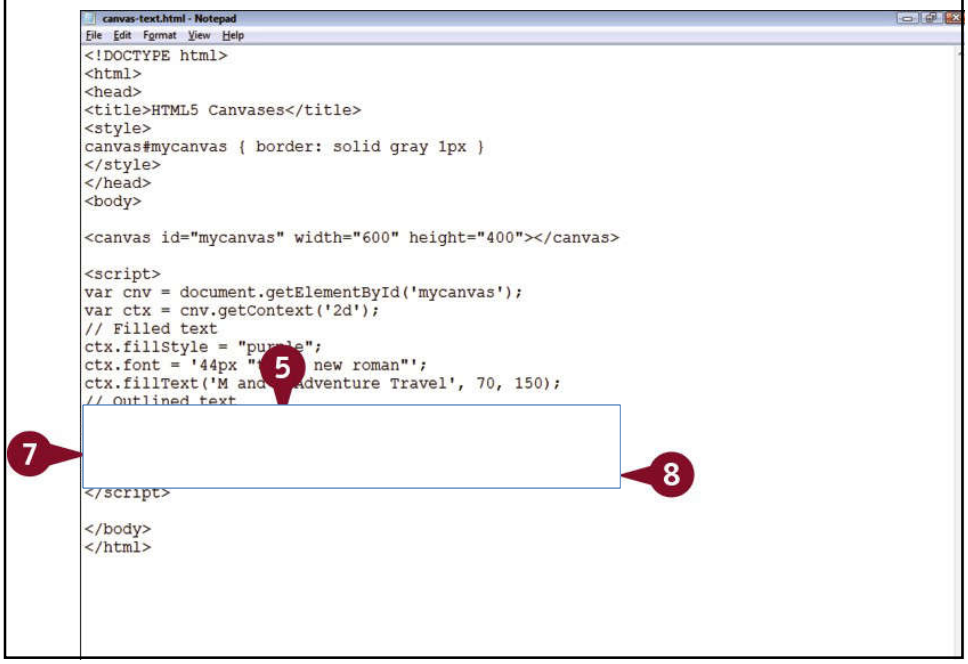
514

Draw Filled Text

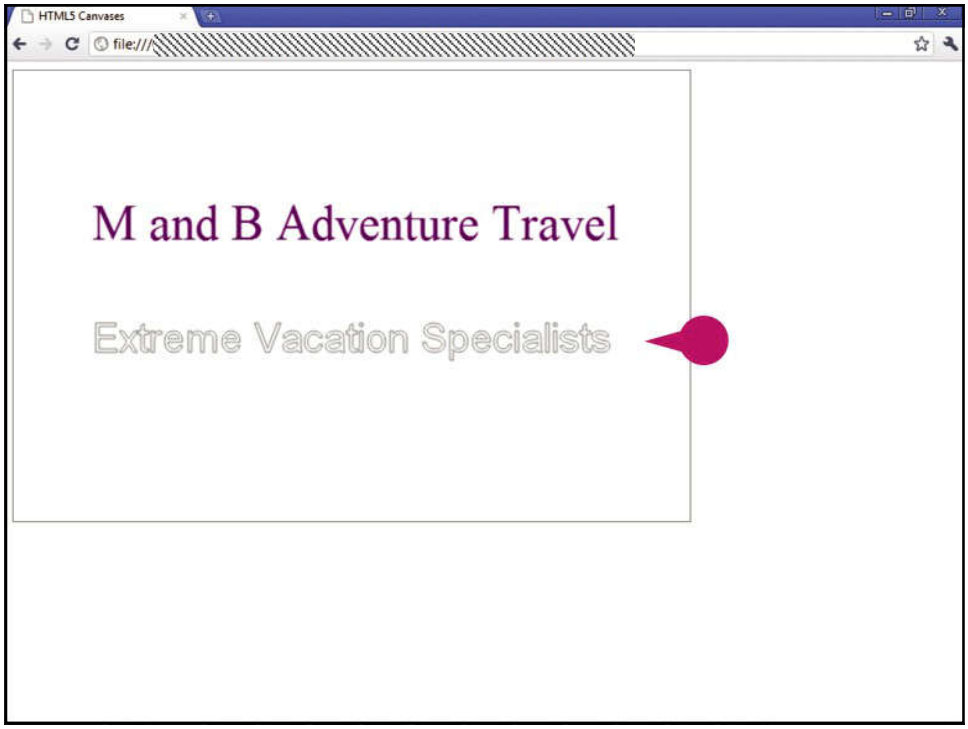


515

Draw Outlined Text



516

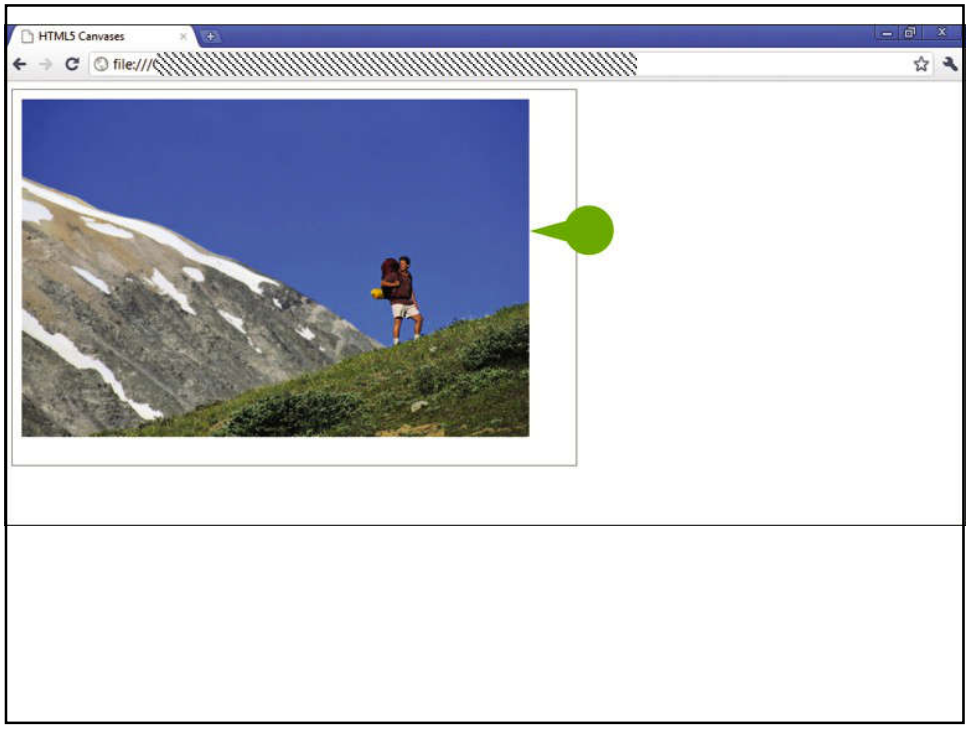


517

Add an Image

- The image must first be loaded into the browser.
- Script execution is delayed until after page loading using the onload property.

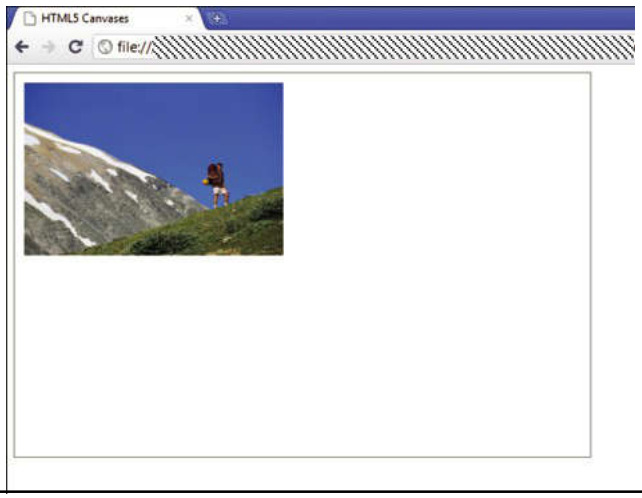
518



519

Scaling

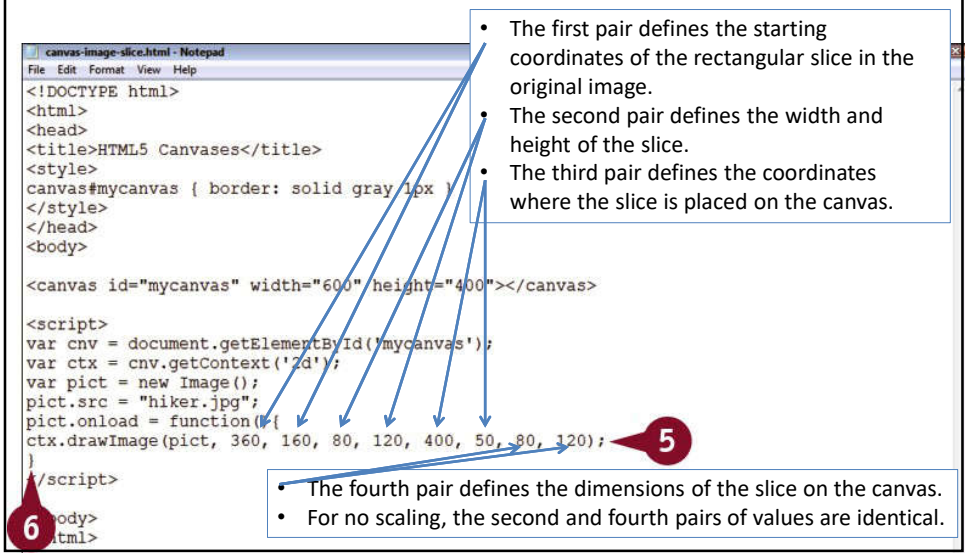
- The drawImage method accepts two additional values:
  - The first value defines the width of the resulting image, and the second defines the height.
  - ctx.drawImage(pict, 10, 10, pict.width*0.5, pict.height*0.5);



520

Slice an Image

- It enables you to place the part of the image that interests you on canva, or place several parts of the same image by repeating the slicing.

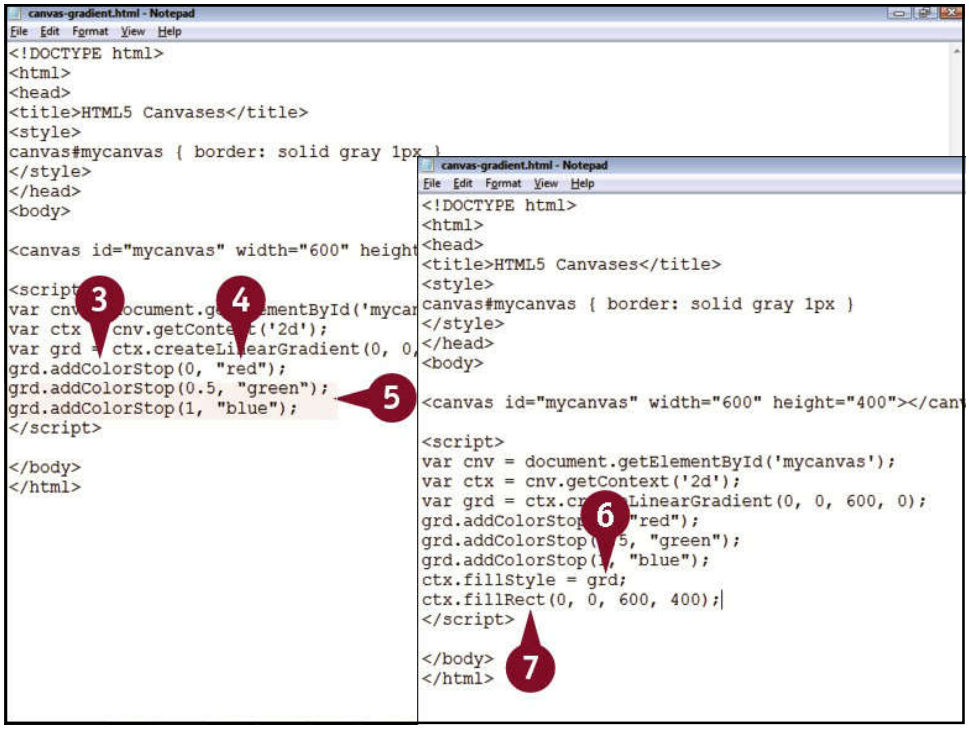


521

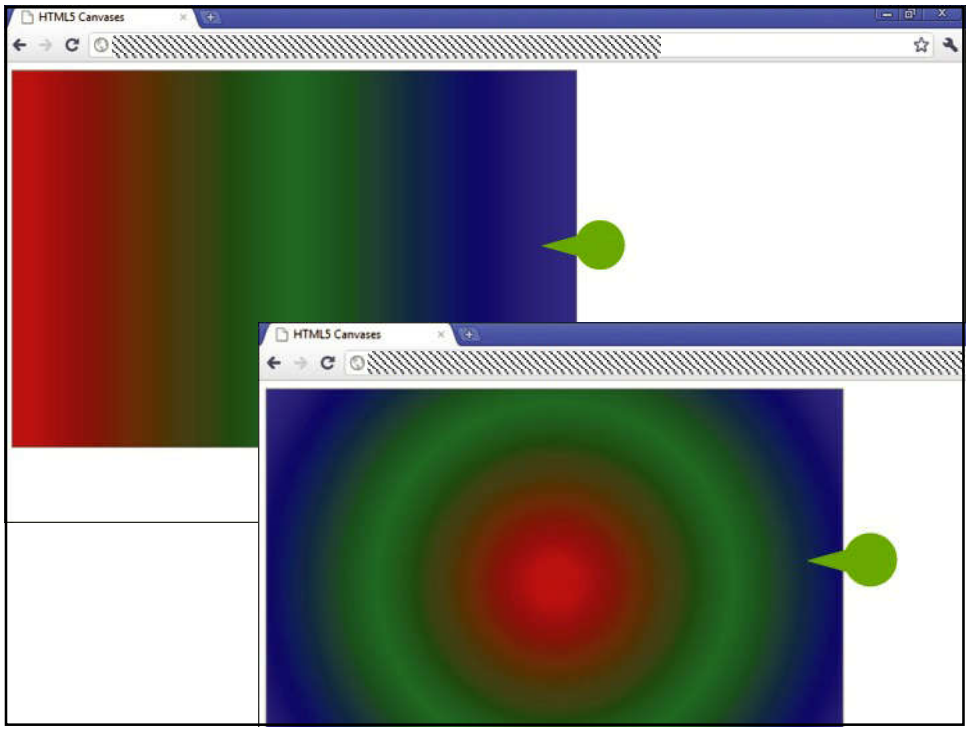
Add a Gradient

- Add a gradual transition from one color to another with linear and or radial.
- Has at least two color stops defining the start and the stop of the color transitions.
  - Additional color stops can be added
  - grd=ctx.createLinearGradient(x_{start},y_{start},x_{end},y_{end});
  - grd.addColorStop(0…1, color),
    - 0…1: a number from 0 to 1. This defines where along the line the color becomes solid, with values at 0 and 1 being at the ends of the gradient.
  - ctx.fillStyle = grd; to load the gradient.
  - ctx.fillRect(starting coordinates, ending coordinates); to draw the gradient rectangle.
- For radial, we use
  - grd=ctx.createRadialGradient(300, 200, 10, 300, 200, 360);
    - Arguments are definition of an inner and an outer circle. To define each circle, specify a center coordinate followed by a radius.

522


canvas-gradient.html - Notepad
File Edit Format View Help
<!DOCTYPE html>
<html>
<head>
<title>HTML5 Canvases</title>
<style>
canvas#mycanvas { border: solid gray 1px }
</style>
</head>
<body>
<canvas id="mycanvas" width="600" height="400"></canvas>
<script>
var cnv = document.getElementById('mycanvas');
var ctx = cnv.getContext('2d');
var grd = ctx.createLinearGradient(0, 0, 600, 0);
grd.addColorStop(0, "red");
grd.addColorStop(0.5, "green");
grd.addColorStop(1, "blue");
ctx.fillStyle = grd;
ctx.fillRect(0, 0, 600, 400);
</script>
</body>
</html>
3
4
5
6
7

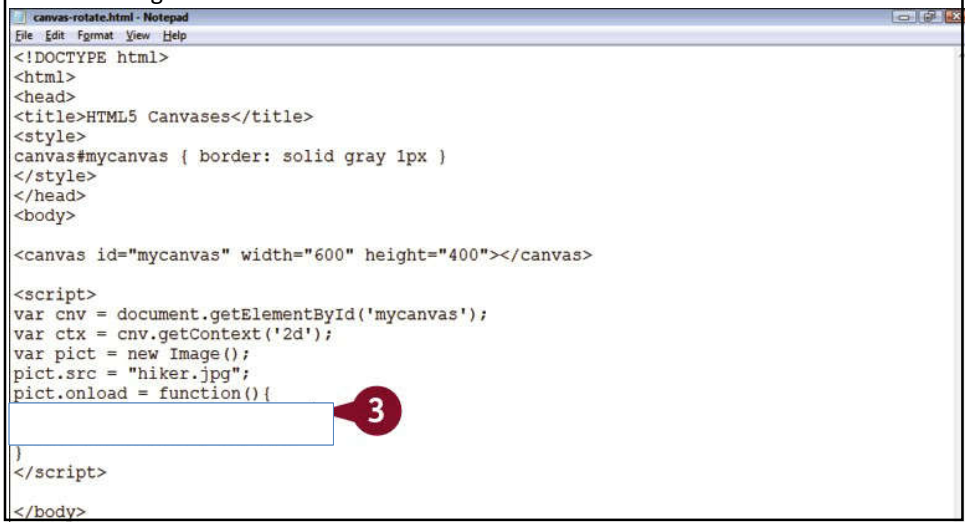
523


HTML5 Canvases
HTML5 Canvases

524

Rotate Canvas Content

- You can rotate shapes, images, text, and other elements on a canvas.
- The rotation occurs relative to the starting coordinate for an object.
- Use ctx.rotate(? * Math.PI/180);, replacing ? with the amount of rotation in degrees.



525

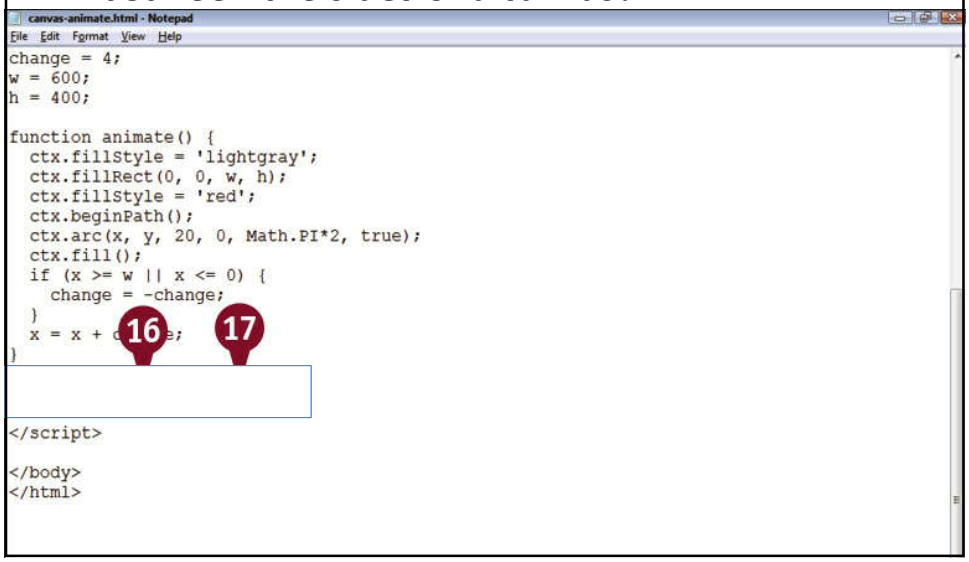
Additive rotation

- Happens when you perform more than one rotation on a page.
- That means if you make a 30-degree rotation, draw an object, make a 40-degree rotation, and draw another object, the second object appears rotated 70 degrees, which is probably not what you want.
- You can avoid this by saving the canvas context prior to performing any canvas methods.
- Then you perform the restore method after drawing the element, thereby resetting rotation. For example, the following rotates the "Rotate30" text 30 degrees and the "Rotate40" text 40 degrees:
- ctx.save();
- ctx.font = '40px "Times New Roman"';
- ctx.rotate(30 * Math.PI / 180);
- ctx.fillText('Rotate30', 200, 10);
- ctx.restore();
- ctx.save();
- ctx.font = '50px Verdana';
- ctx.rotate(40 * Math.PI / 180);
- ctx.fillText('Rotate40', 160, 100);
- ctx.restore();

526

Animation

- How to make a ball bounce back and forth between the sides of a canvas?



527