EE 461L:
BASIC WEB PROGRAMMING

DHTML

- Or, “Dynamic HTML”
- Combination of web technologies that support the creation of dynamic and interactive web pages
- **HTML** for creating text and image links and other “standard” page elements
- **CSS** style sheets for formatting text and html in addition to other style features such as positioning and layering
- **JavaScript** as a programming language that allows access to and dynamic control of individual properties in both the HTML and the CSS

Why Use DHTML?

- DHTML allows webpages to include all kinds of dynamic content
  - Animations
  - Pop up menus
  - Web page content from external data sources
  - Drag-and-drop capability into the webpage
  - …

HTML

Hyper Text Markup Language (HTML)

- An HTML file is a text file that contains **markup tags**
  - Markup tags instruct a browser on how the information in the page is organized
  - Markup tags are usually instructions on how the information should be displayed
  - HTML files must end with a .html (or .htm) extension

An example

```
<html>
   <!-- This is a comment -->
   <head>
      <title>EE461L Sample HTML Page</title>
      <!-- The title is just information; nothing in the header is displayed in the browser -->
   </head>

   <body bgcolor="green">
      This is just ordinary text.
      <i>This is italicized text.</i>
      <br>
      <p align="center">This is a centered paragraph</p>
      <a href="http://www.ece.utexas.edu/~meberlein">Visit my webpage!</a>
   </body>

</html>
```

- Most (not all, e.g., **
**) tags come in opening and closing pairs
- There are lots and lots of tags for all kinds of different things
  - You can look up tags you might want or need, e.g., http://www.w3schools.com/html/
- Some tags can include attributes
  - E.g., **<body bgcolor="green">** … **</body>**
  - Attributes always come in name value pairs as above

Elements and Tags

- Tags are labels you use to surround content
- Tag format: < ... stuff...>
- 2 kinds:
  - Opening and closing tags:
    - `<tag attribute="value" attribute="value"> content  </tag>`
    - Example: <html> ... </html>
    - Content goes between the two tags
  - Stand alone or empty tags:
    - `<tag attribute="value" attribute="value">`
    - Examples:
, <hr> (for line break, horizontal line)

Tags

- Beginning of your html file: <html>
- End of your file: </html>
- Header: <head> ... </head>
  - Document info
  - Contains title element
  - Reference to stylesheet
- Title: <title> ... </title>
  - Defines a title in the browser toolbar
  - Title displayed in search engine results
- Body: <body> ... </body>
  - Encloses content of document (what's displayed in browser)

More Tags

- h1, h2, h3, h4, h5, h6 used for headings
  - largest to smallest
  - h1: level 1, largest text
  - h6: level 6, smallest text
  - Example: <h1> The largest heading! </h1>
- For paragraphs
- stronger emphasis
- <small> For small text </small>
- <hr> for horizontal line
-
 for line break

Lists

- unordered list
- ordered list
- list items

Example:

```
<ol>
  <li> one... </li>
  <li> two... </li>
</ol>
```

HTML Attributes

- Opening tags can contain attributes
  - extra information
  - value in quotation marks

Syntax:

<tag attribute = "value"> ... </tag>

Links

- To other webpages:
  - tag: a is for anchor or hyperlink
  - attribute: href is for hypertext reference (the link's destination)

    Example:

     Here is the text for link

    Example:

     CNN News

    Displays like this:

    CNN News

id Attribute

- If you want to refer back to part of your webpage, add an id attribute

<h1 id = "lunch"> Something about lunch </h1>

Link back to lunch:

 Go to lunch

This scrolls the page back to the section with the lunch id.

Images

- If you want to add an image to your webpage:
- tag: <img>
- attributes: src, width, height, alt

```
<img src = "http://..." width = "100" height = "90">
```

- src – file location
  - Can be relative
- height, width: in pixels

```
<img src = "myphoto.gif" width = "80" height = "75"
alt = "alternative text for the image">
```

style Attribute

- Appearance of an element: color, font, size
- Use instead of attributes like bgcolor, which are deprecated in html5
- <li style=**"color:red"**> My item...
- <p style=**"color:green; background-color:wheat"**> ... </p>

- HTML Color Names

Tables

- Define a table: <table> Table tags </table>
- Table row: <tr> ... </tr>
- Table header cell: <th> header </th>
- Table data cell: <td> data cell </td>

```
<table>
    <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>
```

- An example

YOU TRY IT

Instructions

- Open a text editor
- Create a file called *foo*.html (where *foo* is whatever legal name you want to give your file)
- Put some html formatted text in it
  - You can use my example or create your own
- Save the file
- Open it in a web browser

- Extend it:
- Include a table with at least 3 rows
  - Make the rows have different background colors
- Add a link to a website

CSS

Cascading Style Sheets (CSS)

- Effectively define how to display HTML elements
- Promoting “separation of concerns”
  - Let the HTML define the “content” and the CSS define the “style”
  - Buys us **modularity**, **simplicity**, **readability**, **reusability**, **usability**
- **Syntax**
  - `selector { property: value }`
    - `Example: h1 {color: green; text-align: center; }`
    - `Example: p {font-family: verdana; font-size: 20px;}`
    - The selector is normally the HTML element or tag
    - The property is the attribute you wish to change
      - You can list more than one property, separated by semi-colons
    - Each property can take one or more values, separated by commas
    - This is the most common syntax; there are other options

Internal Style Sheets

- Why? I don't know, since you lose a lot of the benefits. But maybe your single document has a unique style, but you don't want to completely embed it in the HTML.
- Internal styles are defined in the head section using **<style>…</style>**
- Example

```
<html>
  <head>
    <style>
      hr {color: green}
      p {margin-left: 20px}
      body {background-color: yellow}
    </style>
  </head>
  <body> … </body>
</html>
```

External Style Sheets

- External style sheets really enable reusability across multiple pages
- Style sheet is “inserted” via a reference in the **<head>…</head>** section
- Something like:

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

Example: style.css

```
body {
     background-color: #CCFFFF;
      font-size: 2em;
}
p {
      color: red;
      text-align: center;
    }
a {
      color: blue;
    }
```

- Background color for body: aqua
- font-size: 2 times the current font
- Text in paragraphs: red, centered
- Text in anchors: blue

Another Webpage

```
<!DOCTYPE html>
<html>
  <head> <title> This is my title! </title>
        <link rel = "stylesheet" href="style.css">
  </head>
  <body>
  <h1> All About Me </h1>
  <p> Favorite Foods </p>
    <ol>
        <li> pesto </li>
        <li> sushi </li>
        <li> pancakes </li>
    </ol>
 <p> Favorite Websites </p>
    <ol>
        <li> <a href = "http://slashdot.org"> Slashdot </a> </li>
    </ol>
  </body>
</html>
```

Indicates HTML5 document, must go before <html> tag

Colors

- Predefined include: aqua, black, blue, fuchsia, gray, green, lime, navy, maroon, orange, olive, purple, teal, silver, white, gold, yellow
- Specify as:
  - **rgb(0, 255, 0)**
  - **rgb(0%, 100%, 0%)**
  - **#00ff00**
- Can select a color (for text) and a background color:

```
h1 {
    color: yellow;
    background-color: #CCFFFF;
}
```

Text Properties

- font-family
  - Values: arial, helvetica, serif, verdana
- font-size
  - Values: 20px, small, medium, large
- font-weight
  - Values: bold, normal, bolder, lighter
- font-style
  - Values: italic, normal
- text-transform
  - Values: capitalize, uppercase, lowercase
    - capitalize: capitalizes first letter of every word
    - uppercase: every letter capitalized

Text Alignment

- Property: text-align
  - Values: left, right, center

Example:

```
h1 {
        text-align: center;
}
```

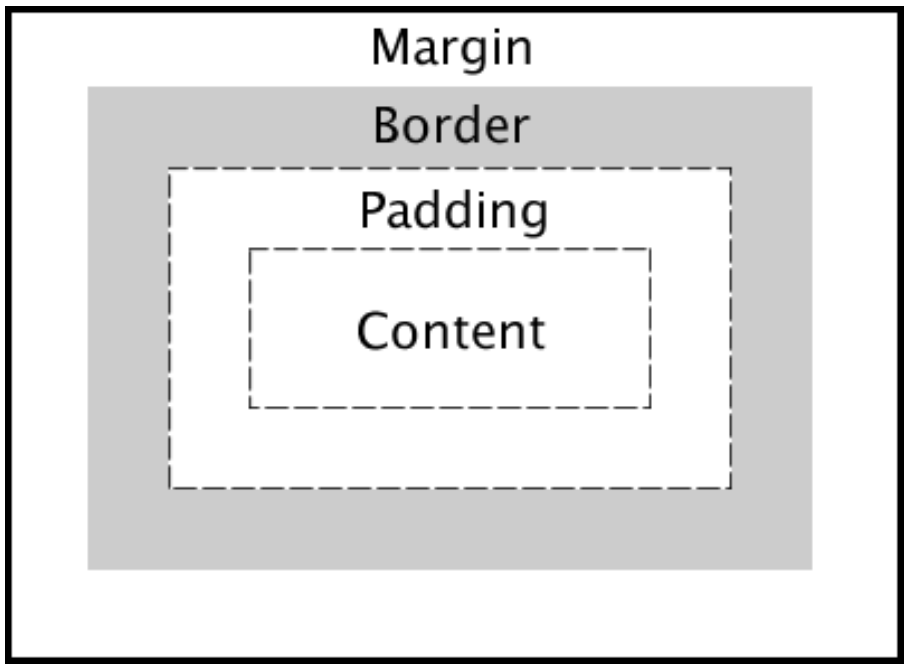
Margins, Borders, Padding, Oh My

- An element is surrounded by a padding box
  - Which is surrounded by a border box
    - Which is surrounded by a margin box

- Value of margin, padding: describes the width of the space around

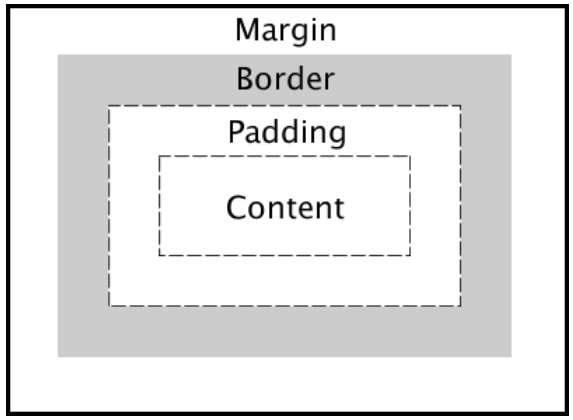
Example:

```
h2 {
   font-size: 1.5em;
   margin: 20px;
   padding: 40px;
}
```



Border Properties

- Property: border-style
  - Values: none (no border), dotted, dashed, solid, groove, ridge
- Property: border-color
- Property: border-width
  - Values: thin, medium, thick



- Other Properties: border-top-width, border-bottom-width, border-left-width, border-right-width

Border Example

```
h1 {
   border-style: dotted;
   border-width: 2px;
   border-left-width: 10px;
   border-right-width: 10px;
   border-color: teal;

}
```

Example

```
table {border-style: solid; color: #333333; border-width: 5px; border-color: #FF0000;
   width: 100%; height: 20%; text-align: center; border-collapse: collapse;
}
tr {
  border-style: dashed;
  border-width: 2px;
  border-color: #DD0000;
}
td{
  border-style: solid;
  border-width: 2px;
}
th {
  border-style: solid;
  border-width: 2px;
}
```

div Tag

- Group a section or division of html together using div tag
- Then specify appearance of that section with CSS

Example 1:

HTML file:

```
<div id="myLife">
  <p> Hello, World! </p>
  <h1> This is my header </h1>
</div>
```

CSS file:

```
#myLife  p {
      color: red;
}
#myLife  h1 {
     color: blue;
}
```

- All paragraphs in #myLife: red text
- All h1 headers in #myLife: blue text

Example 2:

```
<div style="color:#0000FF">
  <p> Hello... </p>
  <h2> A smaller heading </h2>
</div>
```

Example 1 (cont'd)

Hello, World!

This is my header

Background Image

- We've used background colors:
  - h1 {background-color: #6688EE;}
  - p {background-color: teal; }
- background-image specifies an image to use as the background of an element:
  - body {

    background-image: url(myPic.jpg);

    }
  - If you are using images that do not belong to you, make sure you are observing copyright restrictions.

class Selector

- So far: properties for elements of the same type:

```
p { color: blue; }
```

- If we don't want the same appearance for all paragraphs (or all elements of the same type): Use Classes and IDs
- We've already seen id attribute (used with div tag)
- Classes are more general: we can identify multiple elements with a class
  - ID only occurs once on page

.class Selector

.html file:

```
<h1 class="hello"> Hello #1 </h1>
<p> Hello #2 </p>
<p class="hello"> Hello #3 </p>
```

.css file:

```
.hello {
    color: green;
}
```

- Hello #1 and Hello #3 will both be green.
- **Note:** Put a **.** in front of the class name.
- Or only apply style to a specific HTML element by naming the HTML selector first:

  p.hello { color: #CCFFFF;}

  - Only paragraph elements with class "hello" will have the specified text color.

class Selector

```
<style type="text/css">
table.one {background-color: aqua; color:brown;
text-align: left; }
table.two {background-color: yellow; text-align: center;
color:black; }
</style>

<body>
<table class="one" width="100%">
...

<table class="two" width="70%">
...
```

class Selector Example

Item	Cost
Cheerios	$3.49
Milk	$2.79

Car	Price
FIAT 124 Spider	$23,820
Cadillac Escalade	$74,695

Grouping in CSS

- If you have several selectors with the same properties:

```
h1 { color: green;}
.hello {color: green;}
```

- You can consolidate by separating selectors with commas:

```
h1, .hello {color: green;}
```

Nesting in CSS

- Specify properties for selectors within other selectors:

.html file:

```
<div id="hello">
   <h1> Header 1 </h1>
   <p> Paragraph 1 </p>
</div>
```

.css file:

```
#hello  h1 {
   color: blue;
}
#hello  p {
   color: teal;
}
```

- The h1 inside ID hello is blue
- The p inside ID hello is teal

YOU TRY IT

Instructions

- Open a text editor
- Create a file called *bar.*css (where *bar* is whatever legal name you want to give your file)
- Put some css definitions in it. Change several different properties (at least five).
  - Go here for more ideas: http://www.w3schools.com/css/default.asp
- Save the file
- Note: html style attributes will override css rules in a .css file
- Add the css to your *foo*.html webpage

JAVASCRIPT

JavaScript

- JavaScript was designed to add interactivity to HTML webpages
  - More separation of concerns … now we have **content** (HTML), **style** (CSS), and **interaction** (JavaScript)
- JavaScript is a **scripting** language (which is effectively a lightweight programming language)
- JavaScript embedded in an HTML page connects through interfaces called **Document Object Models** (DOMs)
  - Allows interactivity and dynamic behavior
- JavaScript is **interpreted** (i.e., scripts execute without preliminary compilation)

Changing HTML Content with JavaScript: A Method and A Property

- `getElementById()` method
  - Finds an HTML element with a specified id
- `innerHTML` property
  - The content of an HTML element

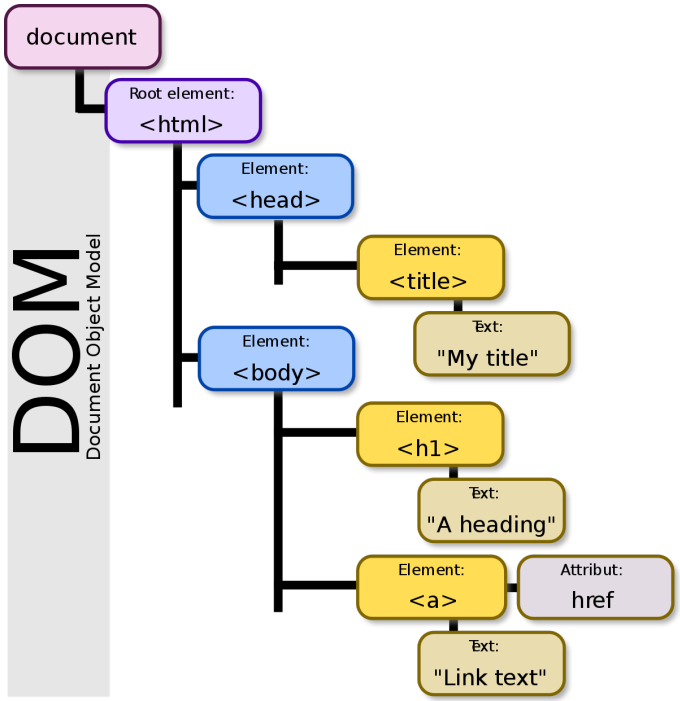
- **Example:** Get the element with id myID and change its content to hello

```
<p id="myID"> goodbye </p>
<script>
document.getElementById("myID").innerHTML = "hello";
</script>
```

SIDEBAR: THE DOM

Document Object Model

- The Document Object Model (DOM) is a representation of an HTML document at runtime as a collection of Javascript methods and objects
  - Allows a webpage to be queried and modified dynamically by Javascript
- Overall document structure:
  - window (global)
    - represents the browser's window
  - window.document
  - window.document.body



https://en.wikipedia.org/wiki/Document_Object_Model

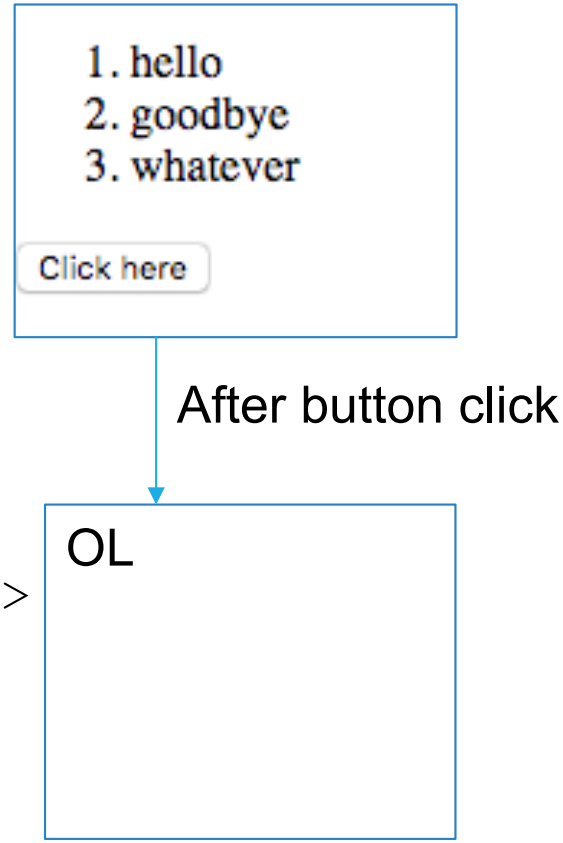
Nodes

- All major components of HTML (elements, raw text, etc.) have a common set of properties and methods
  - Key links: parentNode, nextSibling, previousSibling, firstChild, lastChild
  - `nodeName` property is the element type (in upper case: P, DIV, etc.)
  - Also `getAttribute`, `setAttribute`, etc. methods

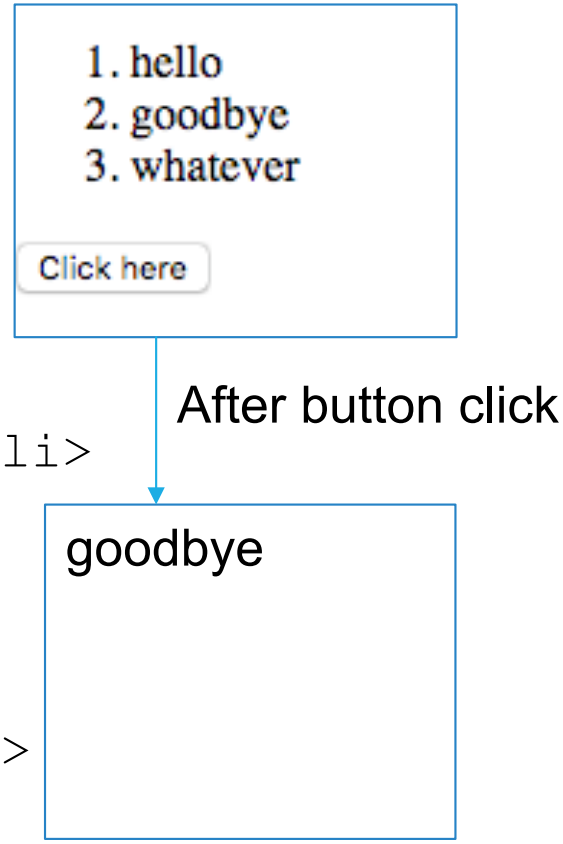
Example

```
<ol>
<li id="ListItem"> hello </li>
<li> goodbye </li>
<li> whatever </li>
</ol>

<button onclick="fun1()"> Click here </button>
<script>
function fun1() {
  var x =
     document.getElementById("ListItem").parentNode.nodeName;
  document.write(x);
}
```



Example




```
<ol>
<li id="ListItem"> hello </li><li> goodbye </li>
<li> whatever </li>
</ol>

<button onclick="fun1()"> Click here </button>
<script>
function fun1() {
  var x =
     document.getElementById("ListItem").nextSibling.innerHTML;
  document.write(x);
}
```

Example

```
<a id="linkThis" href="http://www.cnn.com"> CNN </a>

<button onclick="fun2()"> Click! </button>
<script>
    function fun2() {
    var x =
  document.getElementById("linkThis").getAttribute("href");
    document.getElementById("para").innerHTML = x;
}
</script>

<p id="para"></p>
```

Javascript and the DOM

- How does Javascript relate to an HTML document?
  - Approach #1: work down through the DOM hierarchy starting at the top:
    - `node = document.body.firstChild.nextSibling.firstChild;`
    - `node.setAttribute` ...
  - Approach #2 (usually better): include an **`id`** attribute in elements that will be referenced dynamically:
    - `<div id="div4"> ... </div>`
    - Call `document.getElementById("div4")` in code
    - You need to ensure that each `id` is unique within a page

Basic DOM Operations

- Change the content of an element:

```
element.innerHTML = "This text is <i> important </i>";
```

  - Replaces any existing content (but retains existing element attributes); causes node structure of DOM to change
- Change an image (e.g., toggle appearance on clicks):

```
img.src = "newImage.jpg"
```

- Make element visible or invisible (e.g., for expandable sections):
  - **Invisible:** `element.style.display = "none";`
  - **Visible:** `element.style.display = "";`
    - resets to default display property for element
  - **Display property values:**
    - block – element fills entire line
    - inline – allows content on element's left/right
    - none – element hidden – it takes up no space

Hint: Helpful for JavaScript exercise

Example

```
<html>
<body>
<img id="bestDog" src="BubblesClose.png"
                          height="400" width="400">
<button onclick="notFunny()"> Click Here </button>

<script>
   function notFunny() {
      document.getElementById("bestDog").src =
"https://upload.wikimedia.org/wikipedia/commons/thumb/3/3a/Cat03.jpg/1200px-
Cat03.jpg";
   }
</script>
</body>
</html>
```

More Basic Operations

- Change appearance of an element (e.g., highlight when the mouse passes over):
  - Change its class: `element.className = "active";`
  - Use separate CSS styles for each appearance
  - Generally a bad idea to modify the style directly in the element, e.g., `element.style.color = "#ff0000"`
  - Example
- Redirect to a new page:

```
window.location.href = "newPage.html"
```

- Create a new element and add it to an existing one:

```
element = document.createElement("P");
parent.insertBefore(element, sibling);
```

  - Alternative: `parent.appendChild(element);`
  - Example

More Basic Operations

- Simple dialog boxes:

```
alert("Please click to continue");
if(confirm("Are you sure you want to...?")) {
    ...
 }

name = prompt("Enter user name here: ");
```

Coordinates and Positioning

- Coordinates (for example, to position a pop-up menu next to an existing element):
  - The origin is at the upper left; y increases as you go down and x as you go across
  - Read location with element.offsetLeft, element.offsetTop

Positioning Elements

- Normally elements are positioned automatically by the browser as part of the document
- To pull an element out of document flow and position it explicitly:

```
element.style.position = "absolute";
element.style.left="40px";
element.style.top="10px";
```

- In this case, the element no longer occupies space in the document flow

BACK TO JAVASCRIPT

Embedding JavaScript in HTML

- Scripts can go in the body section of the HTML
  - These will be executed when the page loads
  - Example:

```
<html>
  <body>
    <script>
      document.write("Hello, World!")
    </script>
  </body>
</html>
```

  Will just display the text "Hello, World!" on the page when it loads.
- Scripts can go in the head section of the HTML
  - These can be **called** directly or can be triggered by an **event**

Example

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

<p id="example"> Fun with JavaScript! </p>

<button type="button"
onclick='document.getElementById("example").innerHTML
= "Woohoo!" '> Click me </button>
</body>
</html>
```

External JavaScript

- Scripts can be saved in external files and included in the HTML page
  - Save the scripts in an JavaScript file with the extension .js
  - Then, in the HTML file use the script tag to **src** the scripts:

    ```
    <script src="foo.js"> </script>
    ```

    - This tells the browser to go look in the foo.js file for any script definitions that are not found internally
    - This is nice because it allows you to reuse the JavaScript definitions across multiple web pages

An Example

```
<html>
  <body>
    <script>
      var d = new Date();
      var timeh = d.getHours();
      var timem = d.getMinutes();
      document.bgColor = "red";
      document.write("the time is: ");
      document.write(timeh);
      document.write(":");
      document.write(timem);
      if (timeh >= 12){
        document.write("PM");
      }else{
        document.write("AM");
      }
    </script>
  </body>
</html>
```

What does it do?

Does it work for you?

Another Example

```
<html>
  <head>
    <script>
      function message(){
        alert("Welcome guest!");
      }
    </script>
  </head>
  <body>
    <input type="button" value="View Message" onclick="message()" >
  </body>
</html>
```

What does it do?

Does it work for you?

One More Example

```
<html>
  <body>
    <h1 id="header">My Header</h1>
    <script>
      document.getElementById('header').style.color="red";
    </script>
    <p><b>Note:</b> It is the script that changes the style!</p>
  </body>
</html>
```

What does it do?

Does it work for you?

Ok, One More...

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

<script>
function stuff() {
    var x = document.getElementById("demo");
    x.style.color = "blue";
    x.innerHTML = "Changed It";
    x.style.textAlign = "center";
}
</script>
</head>
<body>
<p id="demo"> A paragraph </p>
<button onclick="stuff()"> Click It </button>
</body>
</html>
```

What does it do??

JavaScript: Changing Element Style

```
document.getElementById("demo").style.fontSize = "30px";
document.getElementById("demo").style.color = "red";
```

Hiding/Showing Elements

```
var x = document.getElementById("demo");

<!-- To hide the element -->
x.style.display = "none";

<!-- To make it visible again -->
x.style.display = "block";
```

Writing Data

1. Write to HTML element using innerHTML
   - specify element by ID with getElementById() method
   - define element's content with innerHTML property
   - document.getElementById("demo").innerHTML = "hi";
2. document.write() function
   - document.write("Hello!");
   - Don't use after HTML doc is loaded – deletes all HTML
3. alert box
   - <script> alert(5+1); </script>

YOU TRY IT

Instructions

- Open a text editor and an html file to edit
- Add a text input to the body of your page

```
<input type="text" id="text1">
```

- Create four JavaScript functions:
  - hide() should hide the text input
  - show() should show the text input
  - format() should change the background of the text field to green, the color of the text typed to red, the font size to 20, and the alignment to centered
  - reset() should reset the formatting to white background, black text, size 14 font, left alignment and ensure that the input text box is visible
- Create four buttons, one that calls each function upon being clicked

Submit

- Yup. This is graded. Turn in as part of tutorial.
  - The HTML file that resulted from the first exercise (it should include the table that you added). It should also input the .css file you created in the second step.
  - The CSS file you created in the second exercise. It should change at least five properties.
  - The HTML file you created in the third exercise that includes the four JavaScript functions and the four buttons.

HTTP

HTTP

- protocol used by browsers to communicate with web servers
- Request-response protocol, layered over TCP/IP sockets

Sample HTTP Request

```
GET /index.html HTTP/1.1
Host: www.example.com
User-Agent: Mozilla/5.0
Accept: text/xml, application/xml, application/xhtml+xml, text/html
Accept-Language: en-us
Accept-Charset: ISO-8859-1, utf-8
Connection: keep-alive
<blank line>
```

- First line contains method, URL, version number
  - GET method: reads information from server
  - POST method: uploads data from browser (e.g., form data), returns information from server; data in body of the request

Sample HTTP Response

```
HTTP/1.1 200 OK
Date: Mon, 29 Jan 2018 17:36:27 GMT
Server: Apache-Coyote/1.1
Content-Type: text/html; charset=UTF-8
Content-Length: 1846

<!DOCTYPE html PUBLIC ... >
<html ...>
...
</html>
```

- Line 1: protocol version, status code, textual explanation
- Headers have same general format as for requests
- Response body does not have to be HTML

Redirection

- Causes a browser to fetch a new URL in place of page initially requested
  - Set `Location` header to some other URL
  - Return a status of 307
  - Useful if information has moved

HTTPS

- Identical to HTTP except request and response messages are transmitted using SSL/TLS
  - HTTPS used automatically when URL begins with https:// rather than http://
  - Request and response message are sent in an encrypted fashion between browser and server
  - Network sniffers cannot access private data (e.g., passwords, credit card numbers)
  - Certificate exchange lets browser identify the servers it's communicating with

FORMS

Web Form

- Collection of elements, each of which has
  - name
  - value
  - user interface (text, textbox, password, radio, checkbox, etc.)
- Different Elements provide different ways for user to edit value
- Hidden fields can be used to submit info not explicitly entered by user

HTML Form

```
<form action="/product/update" method="post">
        Product: <input type="text" name="product"> <br>
        Price: <input type="text" name="cost" value="9.95"> <br>
<input type="submit" value="Submit">
</form>
```

- `<form>` element: overall container for form elements
  - action: URL to invoke on submission
  - method: which HTTP method to use for communicating with the server
    - defaults to GET, but POST is more appropriate
  - can have more than one form on a page

HTML Form

```
<form action="/product/update" method="post">
       Product: <input type="text" name="product"> <br>
       Price: <input type="text" name="cost" value="9.95"> <br>
<input type="submit" value="Submit">
</form>
```

- `<input>` elements: controls for entering data
  - type attribute: specifies which of several controls to use
  - name attribute: used to identify this particular value when posting to the server
  - value attribute: specified initial value for this text element

- `<input type="submit">` creates a button for submitting the form
  - value attribute specifies the button text

QUESTIONS?