Web Technology and Standards

Dr. Alawami

Agenda

- Web Accessibility Standard
- Document Object Model(DOM)
- Bootstrap
- Forms
- Web Scripting

WAI Checklist

- See https://www.w3.org/WAI/, but only required to provide:
  - Provide a text equivalent for every non-text element (e.g., via "alt", "longdesc", or in element content). This includes: images, graphical representations of text and programmatic objects, ascii art, frames, scripts, etc.
  - Ensure that all information conveyed with color is also available without color, for example from context or markup.
  - If you use tables identify row and column headers.
  - If you use frames title each frame to facilitate frame identification and navigation.
  - Organize documents so they may be read without style sheets.
  - Use the clearest and simplest language appropriate for a site's content.
  - If you use images and image maps provide redundant text links.
  - If you use applets and scripts ensure that pages are usable when scripts, applets, or other programmatic objects are turned off or not supported.

Bootstrap

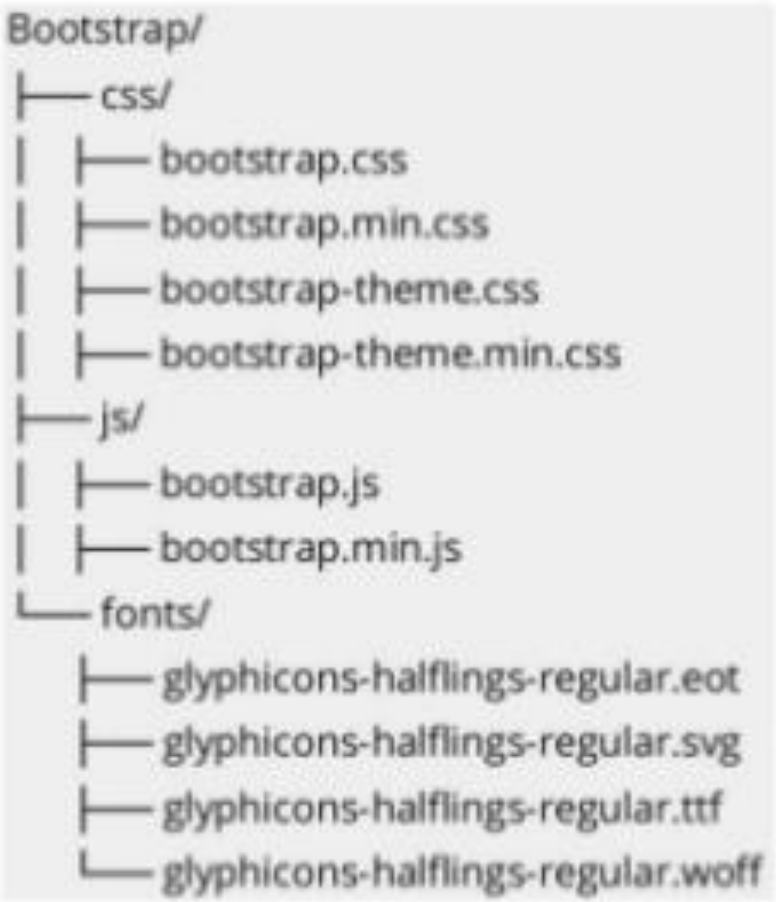
Bootstrap

- Powerful front-end responsive framework
- Developed using :HTML, CSS, JavaScript
- Feature
  - Multidevice support
  - CSS resetting
  - Good looking UI
  - Mobile first mindset
  - Themes

Why Bootstrap?

- Ease of use and Quick to learn
- Speed of development
- Consistent design and common component
- Compatibility across browser
- Responsive framework
- Mobile support

File structure



How to use it?

- Download
  - https://getbootstrap.com/

```
<!DOCTYPE html>
<html>
  <head>
    <title>Page title</title>
    <link href="//netdna.bootstrapcdn.com/bootstrap/3.0.0/css/bootstrap.min.css" rel="stylesheet">
    <script src="//netdna.bootstrapcdn.com/bootstrap/3.0.0/js/bootstrap.min.js"></script>
    <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0, user-
scalable=no">
  </head>
  <body>
    <!-- page content goes here -->
  </body>
</html>
```

Starter template

```
<!DOCTYPE html>
<html lang="en">
 <head>
<!-- Required meta tags -->
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
 <!-- Bootstrap CSS -->
 <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-alpha.6/css/bootstrap.min.css" integrity="sha384-
rwoIResjU2yc3z8GV/NPeZWAv56rSmLldC3R/AZzGRnGxQQKnKkoFVhFQhNUwEyJ" crossorigin="anonymous">
 </head>
<body>
<h1>
Hello, world!</h1>
<!-- jQuery first, then Tether, then Bootstrap JS. -->
<script src="https://code.jquery.com/jquery-3.1.1.slim.min.js" integrity="sha384-A7FZj7v+d/sdmMqp/nOQwliLvUsJfDHW+k9Omg/a/EheAdgtzNs3hpfag6Ed950n"
crossorigin="anonymous"></script>
 <script src="https://cdnjs.cloudflare.com/ajax/libs/tether/1.4.0/js/tether.min.js" integrity="sha384-
DztdAPBWPRXSA/3eYEEUWrWCy7G5KFbe8fFjk5JAIxUYHKkDx6Qin1DkWx51bBrb" crossorigin="anonymous">
</script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-alpha.6/js/bootstrap.min.js" integrity="sha384-
vBWWzlZJ8ea9aCX4pEW3rVHjgjt7zpkNpZk+02D9phzyeVkE+jo0ieGizqPLForn" crossorigin="anonymous">
</script>
</body>
</html>
```

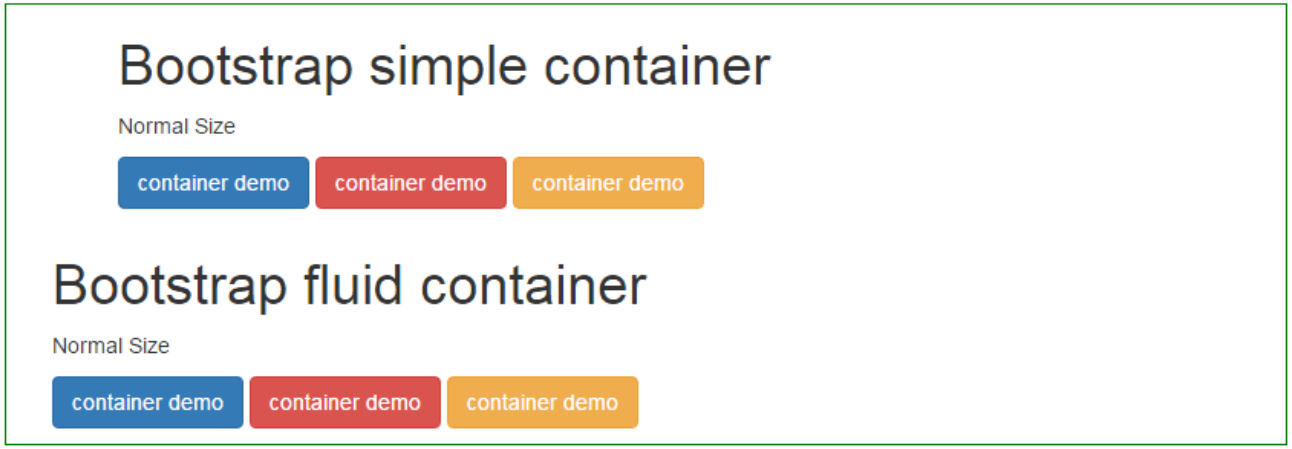
Bootstrap classes

- Utilizing Grid layout
- A set of classes that you need to get familiar with.

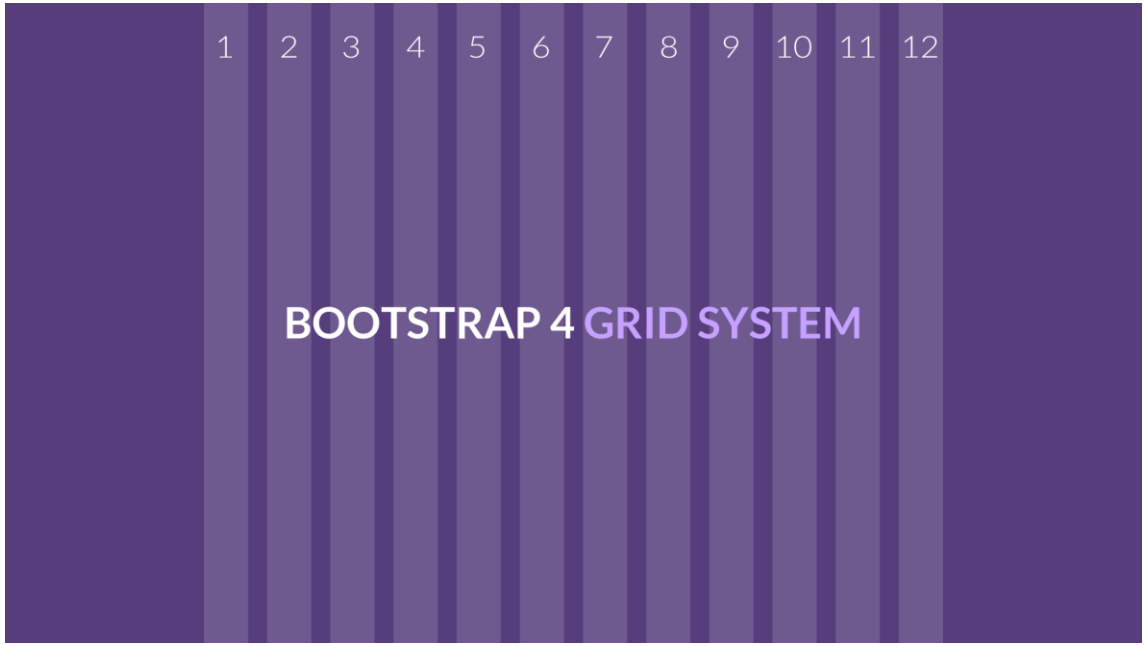
https://v4-alpha.getbootstrap.com/examples/

Containers

- Require container to house the grid system and your page content
- Two types of container
  - Fixed width container
  - Fluid container



Grid Layout

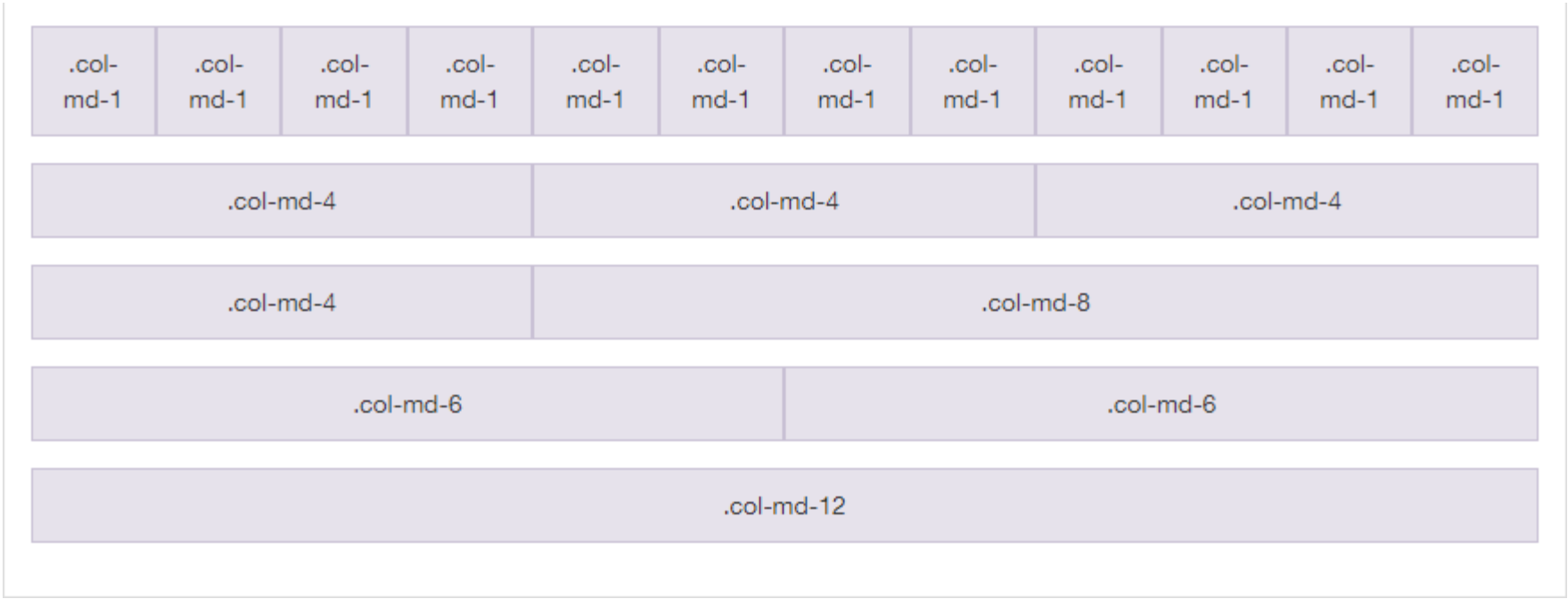


Sizes in Bootstrap

4 Sizes of Bootstrap Grid

Size Name	Screen Size	CSS Class
Extra Small Devices (Phone)	0 - 767 px	.col-xs-1 ~ .col-xs-12
Small Devices (Tablet)	768 - 991 px	.col-sm-1 ~ .col-sm-12
Medium Devices (Desktop)	992 - 1219 px	.col-md-1 ~ .col-md-12
Large Devices (Large screen desktop)	1200px +	.col-lg-1 ~ .col-lg-12

Grid Layout


.col-md-1
.col-md-1
.col-md-1
.col-md-1
.col-md-1
.col-md-1
.col-md-1
.col-md-1
.col-md-1
.col-md-1
.col-md-1
.col-md-1
.col-md-4
.col-md-4
.col-md-4
.col-md-4
.col-md-8
.col-md-6
.col-md-6
.col-md-12

Basic structure

```
<div class="container">
  <div class="row">
    <div class="col-*-*"></div>
    <div class="col-*-*"></div>
  </div>
  <div class="row">
    <div class="col-*-*"></div>
    <div class="col-*-*"></div>
    <div class="col-*-*"></div>
  </div>
  <div class="row">
    ...
  </div>
</div>
```

Grid example

- https://www.w3schools.com/bootstrap/bootstrap_grid_examples.asp

Nesting column

```
<div class="row">
<div class="col-sm-9">
            Level 1: .col-sm-9
            <div class="row">
                        <div class="col-xs-8 col-sm-6"> Level 2: .col-xs-8 .col-sm-6 </div>
                        <div class="col-xs-4 col-sm-6"> Level 2: .col-xs-4 .col-sm-6 </div>
             </div>
 </div>
 </div>
```

Image shape classes

- Classes:
  - img-rounded
  - img-circle
  - img-thumbnail
  - img-responsive: adjust image size accordingly

Rounded Corners:

Circle:

Thumbnail:

Alert


```
<div class="alert alert-success">
  <strong>Success!</strong> Indicates a successful or positive action.
</div>

<div class="alert alert-info">
  <strong>Info!</strong> Indicates a neutral informative change or action.
</div>

<div class="alert alert-warning">
  <strong>Warning!</strong> Indicates a warning that might need attention.
</div>

<div class="alert alert-danger">
  <strong>Danger!</strong> Indicates a dangerous or potentially negative action.
</div>
```

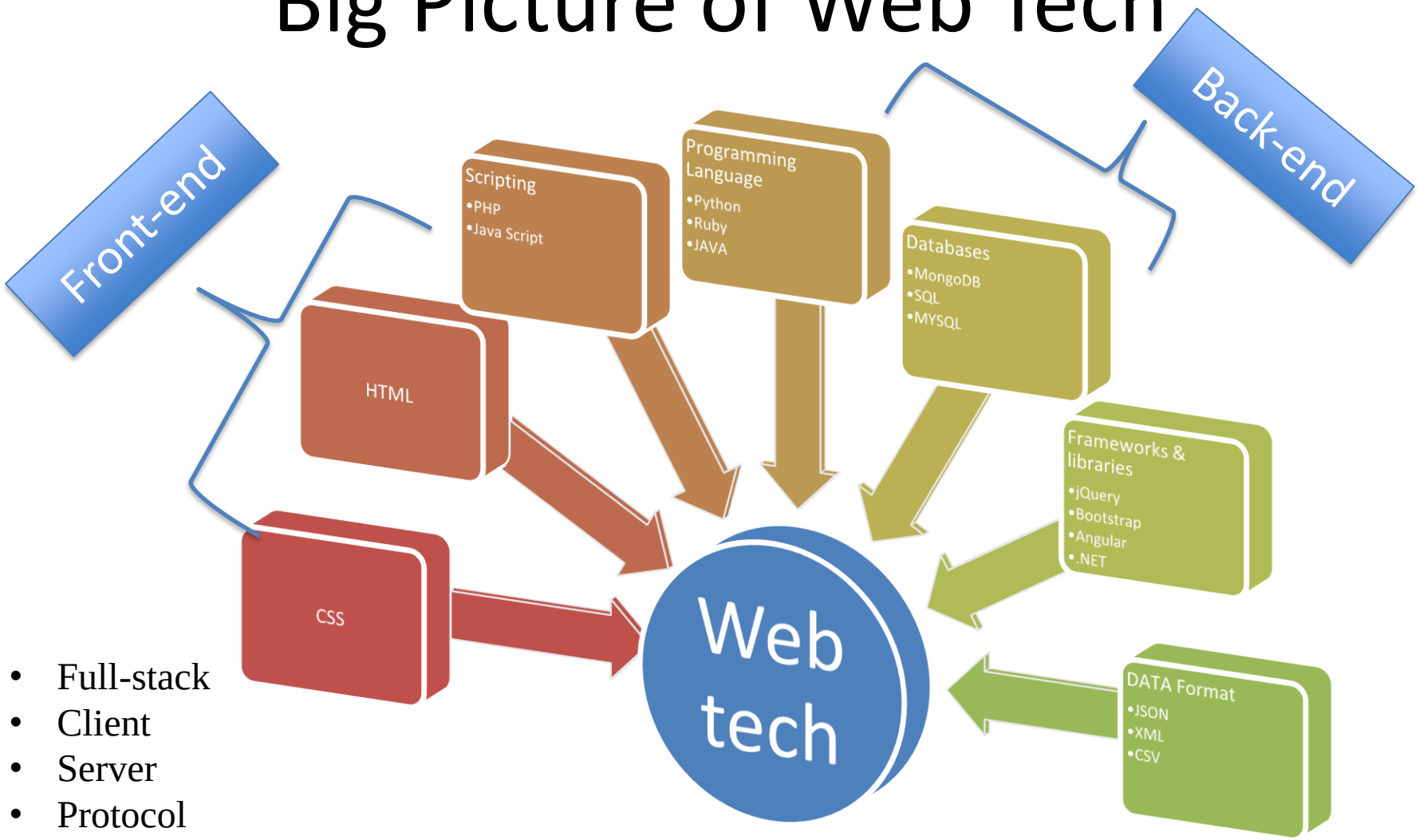
Quick references

- https://www.w3schools.com/bootstrap/bootstrap_tables.asp
- https://getbootstrap.com/docs/4.0/getting-started/introduction/

Demo

- Responsive Design
- https://www.w3schools.com/bootstrap/bootstrap_get_started.asp

Big Picture of Web Tech



- Full-stack
- Client
- Server
- Protocol
- API

https://differential.com/insights/14-technologies-every-web-developer-should-be-able-to-explain/

HTML FORMS

HTML Forms and CGI

- To make pages more dynamic, the Common Gateway Interface (CGI) was defined
- CGI defines the rules for passing data to and running and application of the server
- “Forms” are to pass data to a CGI program
- The server, takes the data and gives it to the program which it runs.
- The program processes the data and returns the results to the – most commonly an HTML doc

Forms Construction

- A form is an element in the body of an HTML document.
- A form element has two attributes – method and action
  - The method specifies which http protocol will be used
  - The action specifies the program that will process the data
- A form will have one or more inputs elements

A Sample Form

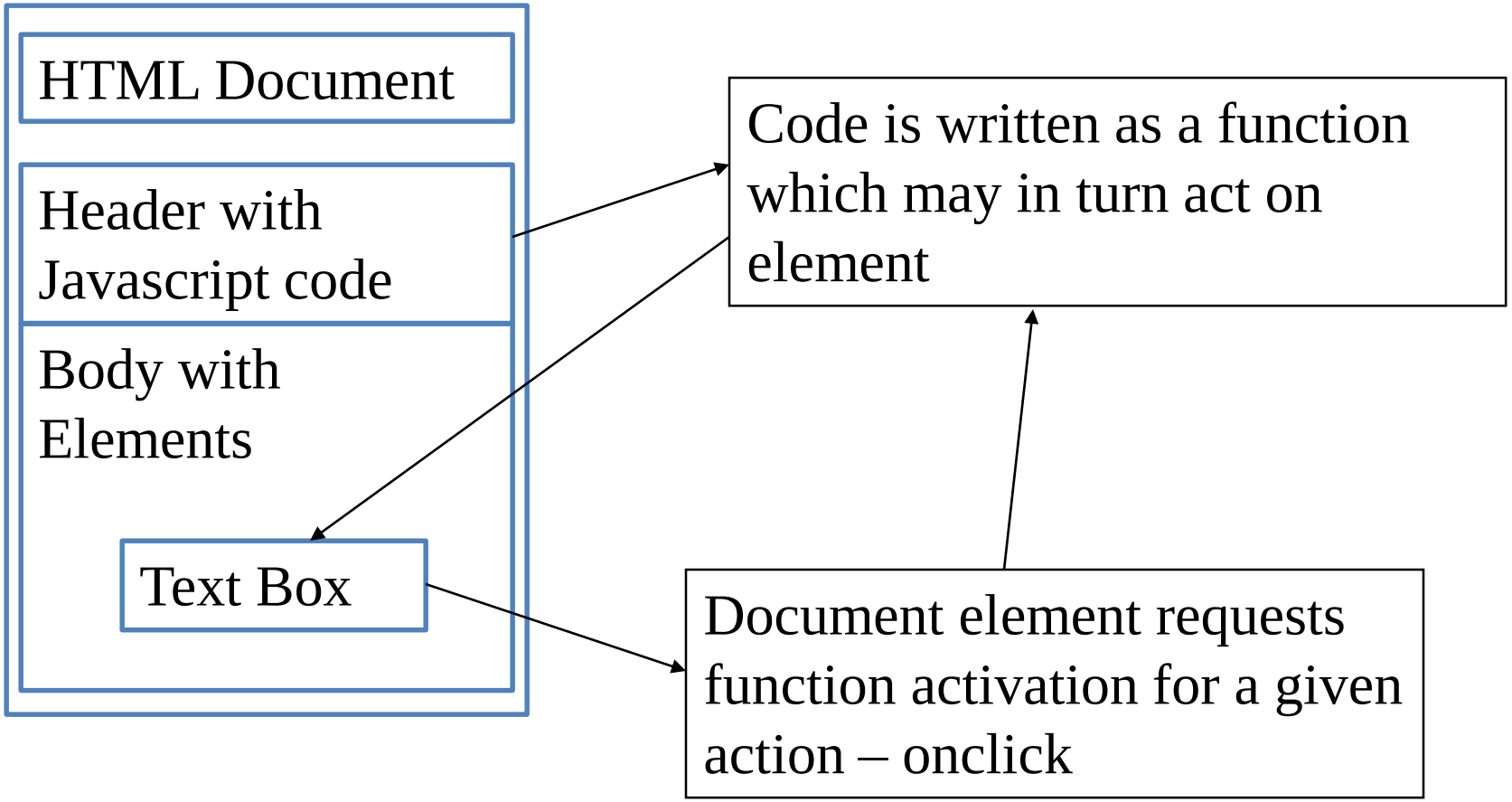
```
<from
    method = "POST"
    action = "http://augment.sis.pitt.edu/cgi-bin/form.cgi">

<p>name:
    <input type="text" size = "40"  maxlength ="80"
    name ="name" value     = "anonymous“ /></p>
<p>subject:
    <input type="text" size = "40" maxlength ="80"
    name ="subject" value = "none"/></p>
<p>
    <input type = "submit" name = "ssc" value = "send comment"/>
    <input type = "reset"  name = "clr" value = "clear omment">/></p>
</form>
```

Some Simple JavaScript

- All of these would be between script tags
  - Last modification date
    - var theDate = "“; theDate = document.lastModified;
    - document.writeln("This document was last modified “+theDate);
  - Day of the week
    - var theDate = Date(); var Day = theDate.getDay();
    - document.writeln("This document was last modified “+Day);
  - Random Saying
    - var sayings = [“string one”, “string two”, “etc”];
    - var select = Math.floor(Math.random()*3;
    - document.writeln(sayings[select]);

JavaScript in a Nutshell

Javascript Conceptually



What Javascript Is and Is Not

- JavaScript is
  - an interpreted loosely-typed object-based language
  - event driven, embedded into HTML, and dependent upon a simplified DOM
- JavaScript is not
  - - simplified Java -- the two languages have disjoint sets of capabilities
  - - simple -- mastery of JavaScript requires advanced programming skills

What JavaScript Can and Can't Do

- JavaScript can:
  - Control document appearance and content
  - Control the browser
  - Interact with the user
  - Read and write client state with cookies
  - Interact with applets
  - Manipulate embedded images
- JavaScript "can't":
  - Directly produce graphical displays
  - Read or write files
  - Establish network connections
  - Support any kind of multithreading

JavaScript Basics

Syntax Basics

- JavaScript is case-sensitive
- JavaScript ignores whitespace between “tokens”
- Semi-colons are “optional”
- Comments
  - C++ style (i.e. //)
  - C - style (i.e. /* */)
- Identifiers, or “A name used to refer to something else”
  - First character must be a letter or an underscore (_) or ($)
- Variables are names associated with a data value.
  - JavaScript is an untyped language (i = 2, sum = ++i)

Data Types and Data Type Wrappers

- Primitive Data Types
  - **Boolean** are true / false values only
  - **Functions** are code that may be executed multiple times
  - **Objects** are named pieces of data that have a collection of properties
  - **Arrays** are indexed collection of data values
  - **Null** indicates “no value”
  - **Undefined** returned when an variable doesn’t exist
- Data Type Wrappers
  - Each primitive datatype (number, string, etc.) has a corresponding object type defined for it.
  - Object Wrappers contain the same data value but also define properties and methods to manipulate the data values.
  - Wrappers are created as transient objects

Operators

- The three classes of operators are
  - binary (+, -, *, /, etc.)
  - unary (-3, +62, etc.)
  - ternary (?:)
- A couple useful operators
  - The Conditional (?:)

    greeting = “hello” + ((name != null) ? name : “there”);
  - typeof(i)

    (typeof (value) != “string”) ? ““” + value + ““” : value
  - Object Creation Operator (new)

    o = new Object; c = new rectangle(3,5,2,1);
  - The delete operator (sets object value to null)

    Delete o;

Strings

- A series of characters enclosed in double quotes.
- JavaScript has many built-in string operations.
  - concatenation msg = “Hello, “ + “world”;
  - length last_char= s.char(s.length -1);
  - substring sub = s.substring(0,4)
  - indexOf i = s.indexOf(‘a’);
  - charAt i = s.charAt(s.length-1);

Conditional Statements

```
-if(name == null)
      name = “John Doe”
-if((address == null) || (address == ““))
{
      address = “undefined”;
      alert(“Please provide a mailing address”);
}
-if(name == null)
    name=“John Doe”
else
    document.write(name)
```

Loop Statements

-while(count < 10){

document.write(count);

count++; }

-for (count=0; count<10; count++)

document.write(count);

-for (prop in MyObject)

document.write(“name: “ + prop ”; value: “ + MyObject[prop], “
“);

Client-Side Program Structure

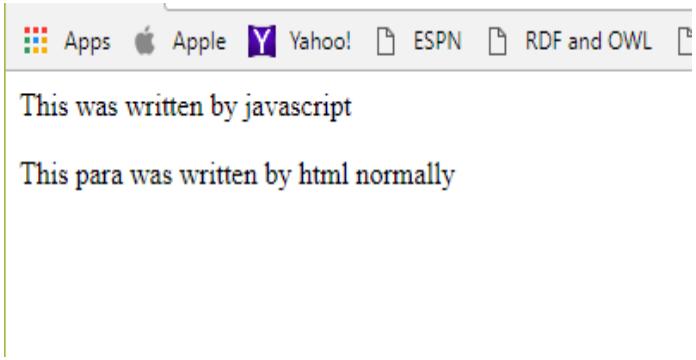
Client-Side Program Structure

- It is preferable to put JavaScript in an external file
  <script type="text/javascript“ src="scripts/valid.js“ />
- Techniques for embedding JavaScript code in HTML:
  <SCRIPT language = “Javascript”>
  JavaScript code
  </SCRIPT>.

- A single HTML file may contain more than one pair of (non-overlapping) <SCRIPT> tag pairs
  - JavaScript statements are executed in the order they appear.
  - Javascript functions are executed when invoked

- Different scripts on the same page are part of the same program. Context scope is the HTML page, not the script block

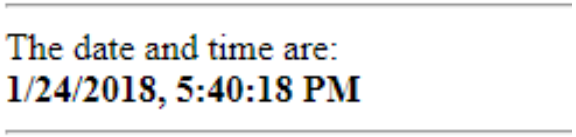
Embedded Direct Execution




```
<HTML>
<HEAD>
<TITLE>Javascript Test File #1</TITLE>
</HEAD>
<BODY>
   <SCRIPT language="JavaScript">
   <!-- this makes the program an html comment
   document.write("<P>This was written by javascript</P>");
   // javascript comment to end html comment -->
   </SCRIPT>
   <NOSCRIPT>
   <P>If you see this,
   there is no java scripting on this machine</P>
   </NOSCRIPT>
<P>This para was written by html normally</P>
</BODY>
</HTML>
```

Embedded using Functions

```
<HTML>
<HEAD>
<TITLE>Today's Date</TITLE>
    <SCRIPT LANGUAGE="JavaScript">
    // Define functions for later use
    function print_todays_date()
    {
      var d = new Date(); // today's date and time
      document.write(d.toLocaleString());
    }
    </SCRIPT>
</HEAD>
<BODY>
<HR>The date and time are:<BR><b>
    <SCRIPT LANGUAGE="JavaScript">
    // call the function defined above
    print_todays_date();
    </SCRIPT>
</B><HR>
</BODY>
</HTML>
```



Execution of JavaScript Programs

- Scripts
  - In order of appearance as part of the browsers HTML parsing process.
- Functions
  - Execute when called
  - Are frequently used as event handlers which allow for asynchronous execution
  - Can be defined to manipulate elements that are not yet defined

Debugging Javascript

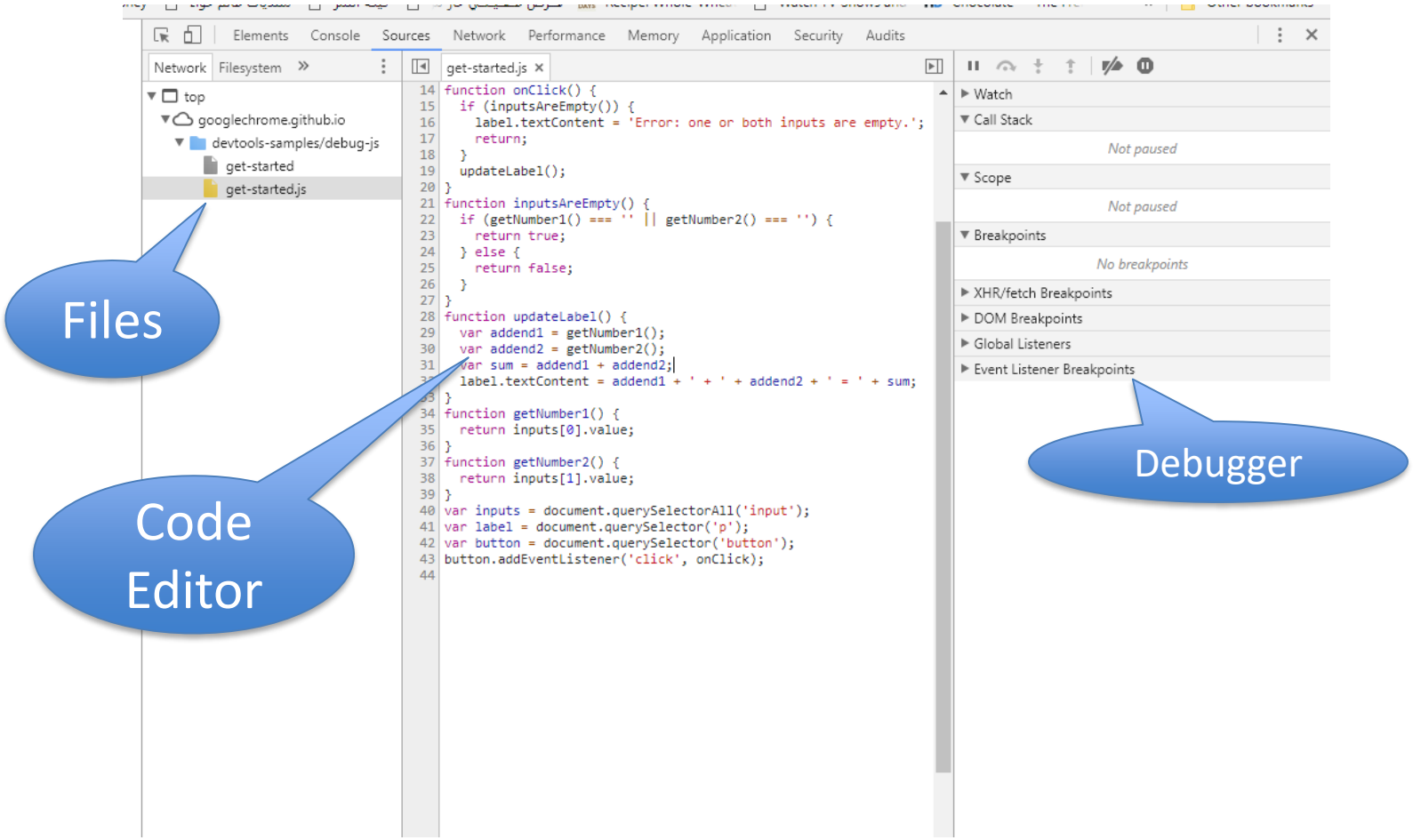
In Goggle Chrome (developer tool, console)

- In Firefox,
  - Firebug is integrated into Firefox developer(No plug-in)

console.log() => Your new best friend

- Breakpoints
- Conditional Breakpoints
- JavaScript debugger
- The inspector HTML source, computed style, events, layout and the DOM

Chrome Debugger



JavaScript
Objects and Events
(DOM Level 0)

DOM

- The Document Object Model (DOM)
  - programming interface for HTML and XML documents
- Allow programs to change the document structure, style, and content
- DOM represent documents as nodes and objects
- Follow W3CDOM standard

Basic Objects

The browser object hierarchy (for Navigator)

- window
  - history
  - location
  - document
    - anchor (<A>'s)
    - link (<A>'s and <AREAS>'s -- imagemaps)
    - image
    - form
      - button
      - checkbox, radio, select,
      - text, textarea
      - hidden, password,
      - reset, submit

Dynamic HTML

- JavaScript can change all the HTML elements in the page
- JavaScript can change all the HTML attributes in the page
- JavaScript can change all the CSS styles in the page
- JavaScript can remove existing HTML elements and attributes
- JavaScript can add new HTML elements and attributes
- JavaScript can react to all existing HTML events in the page
- JavaScript can create new HTML events in the page

JavaScript and Events

- Events occur when a user interacts with the HTML file (which defines the “user-interface”)
- JavaScript extends HTML with the events:
  - onClick, onFocus, onBlur, onChange, onMouseOver
- Event Handlers are normally written as functions

```
<input type text name =“t0”
Value =“” onChange=“validate(this)”>
```

- They can be written as direct attribute changes

```
<input type =“text” name =“t1”
Value = “” onChange=“this.value=‘not so fast’”>
```

Browser Object

- The Browser Object
  - Navigator provides version and configuration information about the browser.
    - appName
    - appVersion
    - userAgent
    - appCodename

Window Object

- Window objects have the following properties
  - closed, default status, length, name, opener, parent, self, status
- Window objects have the following methods
  - alert(string), confirm(string), prompt(string, input default);
  - blur(), focus()
  - scroll(x,y);
  - ID=setTimeOut(expression,msec) -- does expression after msec
  - clearTimeOut(ID) -- clears the timer associated with ID
  - open (arguments) opens a new window
  - eval(string) -- evals string as if it were java script.
- Winow objects have the following events
  - onBlur, onFocus
  - onLoad, onUnload
  - onError

Location and History

- Location
  - The location object has the following properties
    - href, protocol, host, hostname, port, path, hash, search,
  - The Location object only has one method
    - assign(string) changes the href
- History
  - The history object has the following properties
    - current, length, previous, next
  - The history object has the following methods
    - back(), forward(), go(num), and go(string)

Document Object

- The Document Object has the following properties
  - alinkColor,linkColor,vlinkColor
  - bgColor, fgColor
  - cookie, domain, lastModified, referrer, title, URL
- The Document object has the following methods
  - close()
  - open() opens document for writing
  - write and writeln

Component Arrays

- Several component arrays are defined for the DOM
- The arrays can be accessed by number or name
- The following arrays are defined for documents
  - anchors arguments
  - elements forms
  - frames history
  - images links
  - embeds applets
  - mimeTypes options
  - plugins
- events for links, area, and anchor objects include
  - onClick, onMouseOver,onMouseOut

The write() Method

- The write method of the document object is used to dynamically generate web page content.
  - using write() in current window will overwrite currently displayed content!
- For use in a currently open document, open() must be called explicitly

```
<SCRIPT>
   parent.frames[0].document.open();
   parent.frames[0].document.write
      (“<HR>Hello from your sibling frame!<HR>”);
   parent.frames[0].document.close();
</SCRIPT>
```

Two Little Scripts using Document Properties

- The document property “lastModified” is a date-time string that specifies the last modification

```
< SCRIPT >
    var theDate = ""
    theDate = document.lastModified
    document.write("This document was last modified  ");
    document.writeln(theDate);
</ SCRIPT >
```

- The referrer is the URL of the document that contained the link that brought the user here

```
<SCRIPT>
   if (document.referrer != null)
      {document.write(“Glad to see you came from:  ");
          document.writeln(document.referrer);}
</SCRIPT>
```

IS 2560: JavaScript and the DOM

Graduate Program Information Science and Technology
School of Information Sciences
University of Pittsburgh

Overview

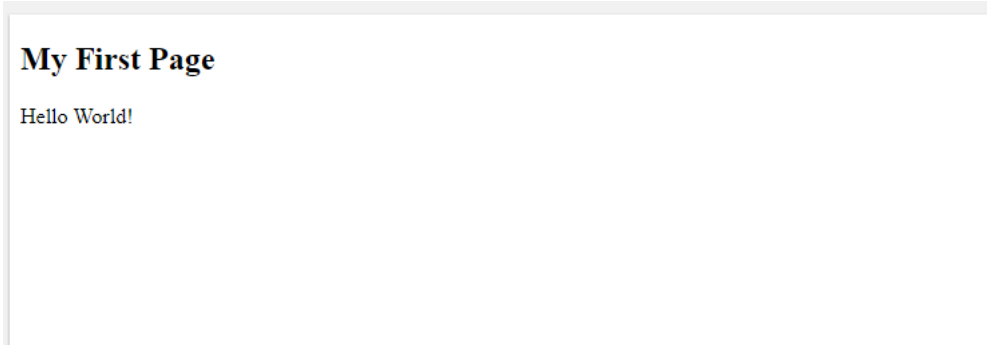
- Javascript and the DOM
- Node Types in the DOM
- Iteration over Nodelists
- Document Methods
- Node Element Methods

JavaScript and the DOM

- JavaScript includes much more extensive support for manipulating DOM objects
- It is possible to access nodes in the tree a variety of ways, and then once accessed grab hold of other nodes
- Common way to access a node are:
  - document. **getElementsByTagName**(“p”)
  - document.getElementByID(“id3”)
- Once a node or set of nodes is obtained, you can check its type or value – see next slide
  - Keep in mind that there may be multiple text nodes around an element – extra white space
- A good online reference is:
  - http://www.w3schools.com/dom/

Hello World

```
<!DOCTYPE html>
<html>
<body>
<h2>My First Page</h2>
<p id="demo"></p>
<script>
document.getElementById("demo").innerHTML = "Hello World!";
</script>
</body>
</html>
```



Source: W3schools.com

Node Type Return Values

Node type	nodeName	nodeValue
Document	#document	null
DocumentFragment	#document fragment	null
DocumentType	doctype name	null
EntityReference	entity reference name	null
Element	element name	null
Attr	attribute name	attribute value
ProcessingInstruction	target	content of node
Comment	#comment	comment text
Text	#text	content of node
CDATASection	#cdata-section	content of node
Entity	entity name	null
Notation	notation name	null

Testing Node Type

```
var x = document.getElementById("myP").nodeType;
```

- If the node is an element node, the nodeType property will return 1.
- If the node is an attribute node, the nodeType property will return 2.
- If the node is a text node, the nodeType property will return 3.
- If the node is a comment node, the nodeType property will return 8.

Node Properties

- Examples of DOM node properties include:
  - x.nodeName - the name of x
  - x.nodeValue - the value of x
  - x.parentNode - the parent node of x
  - x.childNodes - the child nodes of x
  - x.attributes - the attributes nodes of x
- Keep in mind that some properties are null for some nodes and that the return value may be a node or a node set.

Iterating over a Nodelist

- Using getElementsbyTagName

```
x=document.getElementsByTagName("title");
for (i=0;i<x.length;i++)
  {
      document.write(x[i].childNodes[0].nodeValue);
      document.write("<br />");
  }
```

Iterating over a Nodelist

- When using childnodes
  - x=document.getElementById(“XXX”).childNodes;
  - for (i=0;i<x.length;i++)
  - {
  - if (x[i].nodeType==1)
  - {//Process only element nodes (type 1)
  - document.write(x[i].nodeName);
  - document.write("
");
  - }
  - }
- Note that node types equate to numbers
- 1=ELEMENT_NODE, 2=ATTRIBUTE_NODE, 3=TEXT_NODE, 4=CDATA_SECTION_NODE, 5=ENTITY_REFERENCE_NODE, 6=ENTITY_NODE,7=PROCESSING_INSTRUCTION_NODE, 8=COMMENT_NODE, 9=DOCUMENT_NODE

Document Node Methods

- We have already seen to document node methods:
  - getElementById(id)
  - getElementsByTagName()
- There are several other methods including:
  - createComment()
  - createElement()
  - createTextNode()
  - createAttribute(name)

Element Node Methods

- The element node methods – which are extensive – include a variety of DOM construction functions:
  - appendChild()
  - cloneNode()
  - hasChildNodes()
  - insertBefore()
  - removeChild()
  - replaceChild()
  - getAttribute()
  - removeAttribute()
  - setAttribute()

Using Classes and Attributes

```
<style>
.o{ color: black; } .i {color: red;}
.d{ text-decoration: line-through; background-color: #ffdddd;}
</style>
<script type="text/javascript" language="JavaScript">
function select(obj){
 id = obj.getAttribute("class");
 if (id == "y")
        {alert("Great Job, you're done with this one");
        obj.parentNode.previousSibling.setAttribute("class", "d");}
 else
 {alert("Whoops, try again when ready");
        obj.parentNode.previousSibling.setAttribute("class", "i");}
}
</script>
</head><body>
<center><h1>Sample Quiz</h1></center>
<h3 class = "o">This is a question</h3><ul>
<li class = "y" onclick=select(this)>Answer A</li>
<li class = "n" onclick=select(this)>Answer B</li>
```

Using Data Input

```
function begin() {
var student = prompt("Who is the student?", "");
var adviser = prompt("Who is the Adviser?", "");
var now = new Date();
var tnode1=document.createTextNode("Prepared for "+ student + " by "+ adviser);
var enode1=document.createElement("br");
var tnode2=document.createTextNode("
("+now.getMonth()+"/"+now.getDate()+"/"+(now.getYear()+1900)+
")");
fnode=document.getElementById("for");
fnode.appendChild(tnode1);
fnode.appendChild(enode1);
fnode.appendChild(tnode2);}
```

Cloning and Appending Nodes

```
function select(obj){
id = obj.getAttribute("id");
if (id < 2300 || id == 12130)
        {carea_elem = document.getElementById("F");
        count_elem = document.getElementById("fc");
        }
newchild = obj.cloneNode(true);
tn=newchild.firstChild;
txt = tn.nodeValue.substr(0,11);
tn.data=txt;
newchild.appendChild(tn);
newchild.setAttribute("onclick", "deselect(this)");
obj.setAttribute("class", "selected");
newchild.setAttribute("class", "chosen");
carea_elem.appendChild(newchild);
count = carea_elem.childNodes.length
ctnode=document.createTextNode(count);
count_elem.removeChild(count_elem.childNodes[0]);
count_elem.appendChild(ctnode);
}
```

Demo MSIScourse

IS 2560: Select JavaScript Features

Graduate Program Information Science and Technology
School of Information Sciences
University of Pittsburgh

Image Manipulation

Image Replacement

- The Image.src Property
  - This property is read/write.
  - Can be used to make the browser load a new image in the same space as the one currently displayed.
  - New image must be the same size as the current image
- Image Caching
  - Creating an off-screen image forces it to cache.
  - Caching dramatically speeds the loading of an image.
  - To replace an image set the src property of the desired onscreen image to the URL of the desired image

Image Event Handlers

- Both the <IMG> tag and the Image() constructor have an onLoad() event handler.
  - invoked when the image is completely loaded.
  - Use to automatically start animations.
- onError
  - invoked if an error occurs during an image load
- onAbort
  - invoked if the user aborts an image load.
  - example: clicking the “stop” button.
- For any image ONE and only one of these handlers will be called

Image Animation (IA) Script

```
<IMG SRC="images/0.gif" NAME="animation"
<SCRIPT>
var im_array= new Array(10);
var slide=0;
var timeout_id = null;

for(var i=0; i<10; i++) {
   images[i] = new Image();
   images[i].src = "images/" + i + ".gif";  }

function animate(){
   document.images[0].src = im_array[frame].src;
   slide = (slide+1)%10;
   timeout_id = setTimeout("animate()", 250); }

</SCRIPT>
```

Form to Control "IA" Script

```
<HTML><HEAD></HEAD>
<BODY>
<IMG NAME="animation"SRC=""/>
<FORM>
   <INPUT TYPE=button Value="Start"
  onClick="if(timeout_id == null) animate()">
   <INPUT TYPE=button Value="Stop"
   onClick="if(timeout_id) clearTimeout(timeout_id);
    timeout_id = null;">
</FORM>
</BODY>
</HTML>
```

Demo

- Inspecting DOM element

Assignment 2

- Courseweb

For next week

- Reading
  - Chapter 1,3,4 of Angular js in action
  - https://www.tutorialspoint.com/angularjs/angularjs_mvc_architecture.htm
  - https://www.w3schools.com/jquery/
- We will cover JQuery, JSON,AJAX (hopefully Angular too)
- Wrap up your assignment 1(DUE TONIGHT)
- Assignment 2 (Due 10/1)