JavaScript

Script Notes

HTML

- Create objects on a web page
- Attributes can further define properties - href of an anchor describes a link, class attribute describes a css class, etc...

CSS

- Do not create objects
- Style properties that define appearance
- Classes like hover allow for some interactivity

JS

- Creates interactivity
- Makes pages dynamic after they load
- implement logic and analysis
- Process asynchronous data - exchange information with other web pages
- Can act as a bridge between
    - Page/User
    - Html and Server
    - HTML and CSS
- Can manipulate objects
- Creates objects
- Stores data in vars and arrays

- Calculates and can have logical branching
- Can have events and delayed execution

Common Uses of JS

- Validate data in forms
- Respond to user interaction with objects
- Change objects, stylesheets, images
- Load data from a server and display it on a page
- Create dynamic menus

JS Workflow

- What do you want to do, when do you want to do it, and sometimes how do you want to do it
- DOM: Document Object Model — drilling down through object tree to access objects vs named objects
- Objects: Anything from the entire document to a text input field
    - Three aspects:
        - Properties: Features of an object; set through CSS and sometimes HTML
        - Methods: Actions that an object can perform such as writing content to a page or submitting a form; these do something (like round())
        - Events/event handlers: Events that objects can undergo and that can be used as triggers
- Placed in the script block in the head or in any script block that is inline
    - Might set up variables in head, generate alerts in body, etc...
- Can be triggered immediately, when an event occurs, or when it is called (function)

- Variables are "containers" in which you can store information (such as text and numbers), objects from a page (such as images and text boxes), or delayed code called functions (routines you write that are stored in that variable but not "executed" until you "call" that variable.)
- Variables need to be "created" with the var command.
- Variables can be "read" (data/objects retrieved from them) and "written" (store new data/objects into them)

The "arguments" of a method or function are pieces of data (none, one or more) that you pass along to the routine. For instance, when you call the alert method you need to tell the program WHAT to display in the alert. So you pass the "message" as the sole argument in an alert call. The window.close() method does not have any argumetns (nothing else you need to tell it about closing a window), so the parentheses are always empty when you call that method. But regardless of whether a method (or function) has any arguments, you must always provide or represent that format through parentheses.

- Methods are pre-written routines (or "mini programs") already stored in Javascript.
- Functions are routines that YOU write. You must define them in a script block BEFORE you call them.

http://iyawebdev.com/component.cfm?compjoinid=6048&course=acad275

Objects:

- Objects can be anything from entire document to small text input field.
- Three aspects to objects:

1. Properties
2. Methods
3. Events/ Event Handlers

- Common objects include
  - window (the browser window)
  - document (the overall html document)
  - document.body (the visual page/content of the page)
- **Functions:**
  - Functions are variables that contain not data (text or string) but rather a set of JS code to be "run".
  - Functions can be set up to receive data (called parameters of the function) in which case when you call that function you pass data to it (called arguments). Functions can also return data out (with a return command).
  - Functions are most efficient to use when you want to execute a set of code many times. Or when you want to write out a long set of code in a central location that you can activate with a simple name (rather than having 100 lines of code inside an event handler, for instance).

Function

- Small set of code to be run/executed
  - Can be passed arguments

Function Parameters: When you define/create/store a function, you have to decide whether to not it needs "input" parameters -- whether the function "expects" to be handed data when it is called, and what "function variables" names to store each piece of data (for use in the function). Function parameters are representedby variable names in the function parentheses. A function with no paremeters still needs **()** but they are just left empty when

defining/writing the function. Functions parameters are called "local/function variables", and exist only within the body of the function.

Function Arguments: When you call/activate a function, you have to decide whether to pass data *into* the function. Data to be passed into a function are called arguments, and they are put inside the **()** when you call the function. Even if there is no data to be passed into a function, you always have to include the parentheses (even if they are empty).

```
function(){}
//Unnamed functions will run instantly at that point in your code
```

```
fadeinpage = function(){}
//Named functions can be activated by calling fadeinpage(), which has been stored in memor
y
```

```
swapImage = function(imgname, newsrc){}
//Functions can have inputs that, everytime you call the program, it will pass two pieces
 of information
```

Basics

```
document.querySelector("container");
//allows you to target specific objects with JS

//document. simply targets everything
```

```
document.querySelector("").addEventListener("", function(){

});

//adds an event listener to a specific object, which allows you to triggerevents
```

```
this.

//allows you to manipulate the properties of something that has already been selected; wor
ks in event listeners, for example

//specifically targets the object that caused the trigger

document.querySelector("#idname").addEventListener("click", function(){
  this.value //would return the value of #idname
});
```

```
//if you want, you can create your own attribute and then retrieve it with JS
//for this, you would have to use .getAttribute("nameOfAttribute")

document.querySelector("#caption").innerHTML = this.getAttribute("caption");
```

```
alert("") //would alert the user

prompt("") //would open up and alert with a textbox; returns whatever hte user typed
var promptEntered = prompt"Enter name");

navigator.appName //web browser name
navigator.appVersion //version of web browswer

window.close()
```

```
window //object that represents the web browser (two popular methods are .open() and .clos
e()
```

```
document //object that represents HTML file/page
```

```
console.log() //pushes data to the debug conso
```

```
click
mouseover
mouseout
change //detects when a user changes a value
```

```
//used when you want to add a click function after you say .addEventListener("click", func
tion())
```

```
parseFloat() //attempts to convert a piece of data into a num

//number has to come first

variables1 = parseFloat(variable1)
```

```
document.write() //insert text and html into the current page at that location

//generally inline
```

```
object.property //read or set an HTML property of an object

//for example
("#iframe1").src = ""; //can read src but also change it

document.querySelector("#textBox").value=""; //reads only traditional attributes

<input type="text" value="test" myAttribute="test">
<img src="test.png" anupamAttribute="test">
```

```
object.getAttribute("") //get the attribute from the HTML tag

//put the attribute name in the quotes
```

```
document.querySelector("#image").style.width //used to get various CSS properties

object.style.lineHeight //returns line height of object

//important - uses camel case
```

```
Math.random()
//Generates a number between 0 and 1

//multiply by the length of the array
```

```
Math.floor()
//Takes a number and cuts off everything after the decimal
```

```
document.documentElement.clientWidth //returns the width of the browser
```

Variables and Functions

- Variables can contain:
  - Simple data: like a number or a text string
  - An entire object: For instance, if you stored an image from the page into a variable, that variable contains ALL of the information about the image, from its html src (file path) to all of its style properties/settings.
  - A function: You can store a function (a mini program/routine) to be "called" or "activated" later.
  - Array data: Instead of one piece of data, arrays contain a series of values. So for example instead of one (text) color a series of colors, or instead of one image filepaths a series of image filenames.
  - Other: specialized types like boolean (true/false) and undefined

```
//declaring a variable

var newColor = "color name";

var newColor = prompt("Please enter a color");
```

```
//declaring a function

//when declaring a function you have to decide if it has input parameters; data passed is
 called an argument

var linkAlert = function(){
  //this is what is inside of my function
}
linkAlert();

var linkAlert = function(passd1, passd2){
```

```
  alert(passd1 + passd2);
  var myVar = 5;
}

var test = 5;

linkAlert(test, document.querySelector("#textbox").value);
```

- Global Variables: Variables that are created by the var command outside of the functions are global, which means they can be accessed from any blocks or files on the page
    - Read/manipulated anywhere
    - Exist in the memory of browser when the page is loaded and are lost when a new doc is loaded
- Local Variables: Set up without var or initialized in routines (functions) specifically for use in that function
    - Cannot be read outside the defining routine - when the functions finishes, the data is lost
- Object values: Vale properties of objects that can be manipulated like variables

```
//setting variables
//use the = sign to set

var i = "New Page Name";

//can also add

i += "Page 2";
```

```
//comparing variables is done with conditionals, which use == instead of ==

if (i == 5){
  //then do this
} else if (i==6) {
  //if this secondary if is true, execute this code
  //only executes if the previous part was false
} else {
//if nothing above is true, then do this
}
```

```
if (x > y || x == y){
  //will execute if x is greater than y OR x is equal to Y
}
```

```
if (x>y && y > 10){
  //will execute if x is greater than y AND y is greater than 10
}
```

```
if (x != y){
  //will execute if x does not equal y
}
```

Properties of Objects

```
//you can edit official HTML properties with JS

object.src = //allows you to set or get the src of an image, depending on if there is an e
qual sign or not

object.style.CSSCommandInCamelCase = //allows you to set or get the css value of an objec
t, depending on if there is an equal sign or not

object.innerHTML = //allows you to set or get the HTML of an object, depending on if there
is an equal sign or not
```

Building Strings

```
//you can add strings together as you would expect

var addedString = "My " + "Name";
```

```
//if you want to use special characters in a string, you need to use an escape character o
r single quotes
```

```
var addedString = 'Don\'t' + 'Touch'
var addedString = "Don't" + "Touch"
```

Outputting Content to a Page

```
//insert content with document.write()
//this method outputs text directly into the HTML code of a page at the exact location tha
t it is encountered

<html>
<p>Hello</p>

<script>
  document.write("<strong>Patrick</strong>");

<script>

<p>How are you</p>

//Output: Hello Patrick How are you

//Strong tag did not appear in the page; it is HTML being written, not text
```

```
.innerHTML //allows you to replace or read the HTML of a div

//can be used as .innerHTML += to add to the code in a div

//you could also create placeholder objects and edit the src or attributes later
```

Ok, so what techniques/approaches do we have for building "dynamic" elements in pages? There are four common approaches:

- Insert code/objects in-line to your html with ***document.write()***
- **Create placeholder objects for each one you intend to set (image, h3, caption) and then after page load (or near end) *edit* the objects** (with Javascript or jQuery) to their new/real content. I.e. set new src of images, innerHTML of containers, etc.
- **Create a ph div/container, that is empty, and after page load fill it with new html of all the objects**. I.e. create dynamic blocks of html insert then as the

new innerHTML of the ph div.

- **Leave empty space where the object will be located, then after page load insert new objects into the pages DOM (Domain Object Model)**. We will explore this fourth approach *later* in the course.

Arrays

- Gives us the ability to store multiple pieces of data under one variable
- Sets up rows that start from 0
- The row number is called the index

```
//declare and array and fill it

var favoriteColors = new Array;
favoriteColors[0] = "blue";
favoriteColor[1] = "red";
```

```
//declare an array and fill it
var favoriteColors = ["blue", "red"];
```

```
//declare and array and fill it

var favoriteColors = new Array;

favoriteColors.push("blue");
favoriteColors.push("red");
```

- Parallel Array: The idea that you have a number of related pieces of data you want to store in an array
- Allows you to relate certain pieces of data together by having them share the same index
- Not related programmatically, only conceptually

Common Functions/Commands

```
object.length

//returns the length of a string (number of characters)

//returns the number of items in an array
```

```
.toUpperCase() and .toLowerCase()

//transform a string to all uppercase or all lowercase

//allows you to compare two strings neutrally
```

```
.indexOf("Search String", starting position)

//searches for the position of a string inside of a string

//does not require the second part after the comma to search the whole string

//returns the position in the string OR -1 if not found
```

```
keyPress //used to update every time a key is typed
```

```
.charAt(num of char location)

//function for a string that returns the value of the character at a certain passed locati
on in the string

//can convert to string with the toString() method
```

```
.substring(index1, index);
//returns all characters in a string between these two

//does not search through the second value - SO it will include index1, but will not inclu
de index

var myString = word;
myString.substring(0,2) //this will give me back wo
```

```
/man/ //searches for man - replace the "" with / /

//g suffix represents more than one occurence, otherwise it will only find the first one
//i suffix means case insensitive

//usually paired with the replace method

//regular expressions
```

```
object.replace(regularexpression, replacementString)
```

Loops

- Takes one set of code and runs it over and over under conditions are met
- For loops
  - Create a variable and give it a value
  - Set up the condition that determines whether a loop continues to repeat; evaluated at every point in the loop
  - Write a command that is run at the end of every loop (increasing the value)

```
for (var i = 1; i < 5; i = i + 1){

}

//the break command can be used to break out of a loop if neccesary, for example if trigge
red by an if statement

var array = ["test", "test2"]
for(var i = 0; i < array.length; i++){
  alert(array[i]);
//array[0]
//array[1]
}
```

```
var findSport = function(sporttofind) {
  for(i=0;i<4;i=i+1)  {
    if(sporttofind.toUpperCase() == sportname[i].toUpperCase()) {
           alert('The sport ' + sportname[i] + ' is played with a ' + sportobject[i] +
              ' and you often hear the phrase  ' + sportphrase[i]);
```

```
          }
  }
}

//compares a passed variable to every point in the array using a for loop
```

Libraries

- People write libraries of code that can be easily called
- To add a script:
  - Use an src attribute to link the scripting source file
    - Can be placed in head or body, but it is best practice to put them as close to the end body tag as possible
  - Sometimes also need to build a script block that initializes the elements you want to use from the library on; this will initialize the elements you want to use the libraries on
  - Include the plug ins external CSS file in the head
  - Edit your page's HTML to include certain objects or classes, ID, etc...
- Script order matters
  - If you need a library, you need to call the library before you can use the plug in