CodeActually
Dr. Cindy Royal
Texas State University
School of Journalism and Mass Communication

Introduction to Programming for Communicators

- JavaScript is a scripting language commonly implemented as part of a web browser in order to create enhanced user interfaces and dynamic websites.
- JavaScript is not the same as Java.
- Can be run client side or server-side (Node.js – a JavaScript environment).
- Foundation for code libraries like JQuery.

For the first series of exercises, we'll be practicing in Chrome's console. This is an environment that let's you run code and see the result. Later, we will be implementing JavaScript in html pages.

In Chrome, choose View, Developer Tools and click on the Console tab. Or Ctrl-click on the screen and choose Inspect Element.

Later, we'll be using a text or html editor and a browser for some of the exercises.

Your First Program
Write the string "Hello World!" to your document

```
console.log("Hello World!");
```

Don't worry if you see "undefined" in the console after the code returns. The console is attempting to determine the type that the code returns, and in most cases in our exercises that is "undefined."

Objects, Properties and Methods
For this exercise, we will mostly be using the console object to write code to the console.

```
console.log(“info goes here”);
```

Later we will use the document.write method to write to the html document. We do this by calling the document object and the write method.

Basic JavaScript Syntax
End lines with semicolon ;
Put arguments/parameters in parentheses ( )
Period (.) between object and method or property.
Strings in quotation marks "

Anything can be an object in JavaScript – any data type, as you will see as we progress through the exercises. Unlike other languages, you do not have to declare a type when instantiating a variable.

Some objects are built-in (i.e. document, window), some can be defined.

Data Types
String, Number, Boolean, Array, Object, Null, Undefined - programming languages have syntax and functions that work with a variety of data types.

We will mainly be concerned with strings and numbers, later adding booleans and arrays.

Length
Length is a string property that returns the length of the string.

```
    console.log("Hello World!".length);
```

This returns a number that is the length of that string, including the space.

Concatenation
Use concatenation to join strings. The empty space between the quotation marks (" ") represents a space. The plus sign is the concatenation operator.

```
    console.log("Hello" + " " + "Cindy!");
```

Variables
Use variables to store numbers and strings. The var prefix is optional.

```
    var name = "Cindy";
    console.log("Hello" + " " + name);
```

and

```
    var firstname = "Cindy";
    var lastname = "Royal";
    console.log(firstname + " " + lastname);
```

Math and numbers
Operators and Math

Within your script tag, try the following. You will see the different operators for math functions.

```
     console.log(3+4);
     console.log (3-4);
     console.log (3*4);
     console.log (3/4);
     console.log(3%4);
```

The “%” operator is the modulo that shows the remainder of division calculation.

Substrings
I’m using these scripts to see what my JLo name would be, finding the first character of my first name and the first two characters of my last name.

```
     var firstname = "Cindy";
     var lastname = "Royal";
     console.log(firstname.substring(0,1) +
     lastname.substring(0,2));
```

or

```
     var first = "Cindy".substring(0,1);
     var last = "Royal".substring(0,2);
     console.log(first + last);
```

Notice that we used a combination of variables and the substring method in two different ways to achieve the same result. Often in programming, there are multiple ways to solve a problem.

Alerts and Prompts

Alert - a general warning

```
     alert("Danger");
```

Confirm - answer yes or no before you can continue.

```
     confirm("Are you sure?");
```

Prompt - answer anything - creates variable

```
prompt("What is your name?");


var name = prompt("What is your name?");
console.log("Hello " + name);
```

Booleans and if statements
Use booleans to test if something is true or false. Combine with an if/else statement to define different outcomes. Notice that the if statements actions are delineated with curly braces. You can also test for mathematical statements like greater than >, less than < or equal to ==.

```
name="Martin";
if(name=="Cindy"){
console.log("Yes");
}
else {
console.log("No");
}
```

or

```
x=2;
if(x>1) {
console.log("Yes");
}
else {
console.log("No");
}
```

Comparisons (= or ==)
A single equal sign (=) indicates assignment, usually used to assign a value to a variable. If you are doing a test for comparison, then you use 2 equal signs (==).

```
2+2==4;

var name="Cindy";
if(name=="Cindy";)...
```

Functions
Functions allow you to define an operation and then use it later. Notice the statements in the function are enclosed in curly braces. Then we call the function using its name to execute it. There are two different ways to write a function.

```
var hello = function () {
var name = prompt("What is your name?");
console.log("Hello " + name);
}

hello();
```

or

```
function hello() {
var name = prompt("What is your name?");
console.log("Hello " + name);

}

hello();
```

Arguments
You can add arguments in the parentheses of a function to pass information into it.

```
     var hello = function (a) {
     console.log("Hello " + a);
     }

     hello("Cindy");
```

or

```
     function hello(a) {
     console.log("Hello " + a);
     }
     hello("Cindy");
```

Loops
Loops allow for iteration through a cycle.

The **while loop** is a statement that uses a variable as initializer, condition and iterator.

```
     var i=0;  // initializer
     while (i<5) { //condition
     console.log("I am writing this five times");
     i++; // iterator
     }
```

Writing JavaScript in an HTML File
Notice that the console doesn't write the same line over 5 times. It shows a number after the statement. To actually see this execute five times, this is a good time to switch over to writing code to a document. Open a text editor like Text Wrangler. We will structure a full html page later, but for now, simply write the code below. Save your files with .html extension (any name is fine, something like javascript.html). Open the html file that includes the JavaScript in the browser.

```
     <script>
          var i=0;
          while (i<5) {
          document.write("I am writing this five times");
          i++;
          }
     </script>
```

Notice that we are now using the document.write method instead of console.log, which writes the string to the html document so it will display in the browser.

This writes the string five times, but it doesn't use a break to put each one on a different line. Modify the document.write line to include an html break, like this:

```
document.write("I am writing this five times <br />");
```

Loops, continued
The **for loop** reduces some of the coding by having three statements in the parentheses, separated by a semicolon; - an initializer, a condition to test and an iterator.

```
<script>
for (var i=0; i<4; i++)
{
document.write("I am writing this 4 times<br />");
}
</script>
```

Or if you want to get fancy, you can add the iterator to have a different number in each line.

```
<script>
     for (var i=0; i<4; i++)
     {
     document.write("This is time " + i + "<br />");
     }
</script>
```

Arrays
Arrays allow you to store a collection of information in a variable and then access it via its index number. Index numbers start with 0. Enclose elements of an array in square brackets.

```
<script>
     var favGroups = new Array("Old 97s", "Spoon",
     "Wilco");
     document.write(favGroups[1]);
</script>
```

or use the square bracket notation.

```
<script>
     var favGroups = ["Old 97s", "Spoon", "Wilco"];
     document.write(favGroups[1]);
</script>
```

This accesses the item in the array in location 1. Notice that it isn't the first item in the array. Arrays start with 0, so to access the first item us favGroups[0].

You can use a loop to iterate through an array. Concatenate a break so the items appear on a different line. Your condition uses the length property of an array to determine how many items are in it.

```
<script>
     var i=0;
     var favGroups = [“Old 97s", "Spoon", "Wilco"];
     while(i<favGroups.length){
     document.write(favGroups[i] + "<br />");
     i++;
     }
</script>
```

Notice how we can add html items as a string. We are beginning to integrate html and JavaScript for interactivity.

Objects

Objects are similar to arrays except they store items in key/value pairs rather than accessing them by their index number. Instead of using square brackets, you use curly braces for objects.

```
<script>
     var bands;

     bands = {
     name: "Old 97s",
     city: "Dallas",
     genre: "alt-country"
     };

     document.write(bands.name + ", " + bands.city + ", " +
     bands.genre);

</script>
```

We can use an array in conjunction with an object to add multiple elements to the object.

```
<script>
     var bands;

     bands = [
     {
     name: "Old 97s",
     city: "Dallas",
     genre: "alt-country"
     },

     {
     name: "Spoon",
     city: "Austin",
     genre: "Indie Rock"
     },
     {
     name: "Wilco",
     city: "Chicago",
     genre: "alt-country"
     }

     ];
document.write(bands[1].name + ", " + bands[1].city + ", "
+ bands[1].genre);

</script>
```

And we can loop through an object just like we did with an array. Your condition uses the length property of an object to determine how many items are in it.

```
<script>
     var bands;

     bands = [
     {
     name: "Old 97s",
     city: "Dallas",
     genre: "alt-country"
     },
     {
     name: "Spoon",
     city: "Austin",
     genre: "Indie Rock"
     },
     {
     name: "Wilco",
     city: "Chicago",
     genre: "alt-country"
     }
     ];

for(var i=0; i<bands.length; i++) {
document.write(bands[i].name + ", " + bands[i].city + ", "
+ bands[i].genre + "<br />");

}
</script>
```

You will sometimes see data embedded in objects in JavaScript referred to as JSON (JavaScript Object Notation).

Using JavaScript in an html document
getElementById(); is a magical method that you can use to access parts of a Web page. The function below, when called, changes the innerHTML of the element with the id "first". This is very powerful when combined with forms to ask a user for input to change something on a page.

Let's properly structure the document now to include the HTML5 doctype declaration, too.

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

     <p>Hello <span id="first">Cindy</span></p>

<script>
function nameChange() {

     document.getElementById('first').innerHTML = 'Jon';
}

     nameChange();
</script>

</body>
</html>
```

We will go into more detail with getElementById(); in the exercise.

Events

An advanced use of JavaScript in an html document has to do with mouse events. An event is an interaction – onclick, onload, onmouseover, onmouseout.

Simple:

```
<!DOCTYPE html>
<html>
<head>
<title>My JavaScript</title>
</head>
<body>
<h1 onclick="this.innerHTML='Good work!'">Click on this
text!</h1>
</body>
</html>
```

This “this” operator tells the code to operate on the current element.

In a function:

```
<!DOCTYPE html>
<html>
<head>
<script>
     function changetext()
     {
     document.getElementById("new").innerHTML="You're a
     pro!";
     }
</script>
</head>
<body>
<h1 id="new" onclick="changetext()">Click on this
text!</h1>
</body>
</html>
```

You will use events combined with getElementById() to create interactivity on your site.

More advanced use of getElementById() with form
This form requests your name, then it stores that info in a variable and presents it in the message below on the page.

```
<!DOCTYPE html>
<html>
<head>
<title>Fun With Forms</title>
</head>
<body>

<form  id="form1" action=" "/>

<p>Name: <input type="text" id="newname" /></p>

<input type="submit" value="Submit">

</form>

<p>Hi <strong><span id="yourname">"Your Name
Here"</span></strong>. Welcome to the site.</p>
<script>
document.getElementById("form1").onsubmit=function() {
    var newName = document.forms["form1"]["newname"].value;
    document.getElementById("yourname").style.color = "red";
    document.getElementById("yourname").innerHTML = newName;
    return false; // required to not refresh the page
  }
</script>
</body>
</html>
```

This code changes the innerHTML of the span tag in the html document to the text from the form. It also changes the color. We'll get more practice with JavaScript in the charting exercise.

Using Javascript with an external script file
You can also organize your scripts in an external file (just like we do with CSS files) that allows you to reference them across files. This is helpful for storing functions.

script.html
```
<html>
<head>
<script type="text/javascript" src="script.js"></script>
</head>
<body>
<div id="helloMessage">
</div>

<script>
     writeMessage();
</script>
</body>
</html>
```

script.js
```
function writeMessage() {
     document.getElementById("helloMessage").innerHTML =
     "Hello, world! from an external script";
     }
```

Comments
Developers use comments to leave notes in the code, provide instructions for other developers or to provide clues to certain techniques. It helps to organize code. Comments can also be used to temporarily remove and test code without having to delete it.

With the script tag.
```
// This is a comment
/* This is a multi-line
comment */
```

Note: basic html comments are handled by <!--   -->. This would be used in html documents outside the <script> tag.

Side Note:
The document object is part of the Document Object Model (DOM). The DOM is an application programming interface (API) associated with the page that allows you to access and change things with JavaScript, as we did with getElementById(). There

are other objects associated with the DOM. Some objects are browser objects (i.e. window). More on this here: http://www.w3schools.com/jsref/default.asp

Objects can also have properties. Properties don't have arguments (and therefore parentheses), but are used to return specific information about the object.
i.e.

```
	document.title;
```

This returns the title of the document. You still need a method to tell it what to do with what it returns.

```
	document.write(document.title);
```

Now you have a basic understanding of programming concepts. There are many ways to go from here. You can learn more about coding Web pages with HTML/CSS, so that you can integrate interactive concepts. Or you can move into the world of JQuery to learn how to take advantage of coding libraries.

We will move into an exercise that creates an interactive chart with HTML and Javascript. You will get practice using these techniques in an actual example.