Intro to JavaScript

JavaScript

JavaScript is a programming language that only runs in a browser in an .html document.

If your document isn’t an .html document, JavaScript won’t run.

And it won’t run independently of a browser.

JavaScript is used for:

- Programming (such as calculations)
- Doing more than one thing with a given mouse-click or keystroke
- Manipulating the DOM (Document Object Model) of already opened web pages, without requiring the page to refresh

JavaScript

JavaScript always appears between the <script> and </script> elements:

```
<script>
    var horses = "palaminos" + " " + "apaloosas"
</script>
```

- The <script> can appear in the <head> element or in the <body> element. It generally appears in the <body> element when the result of the is going to be written into the <body> of the document.
- JavaScript can be placed inline anywhere in the <body>, but is usually placed at the end of the document just before the </body> element — to make it easy to find and so it doesn’t slow down the loading of the page.
- Scripts can also be placed in a linked document, using <script src="javascript.js"></script>, like external CSS.

Syntax

Syntax

HTML is not case-sensitive:

<!DOCTYPE html> == <!doctype html>

JavaScript is VERY case sensitive:

var CamelCase ≠ var Camelcase ≠ var camelcase

(This is a good place to start looking when debugging.)

Syntax

White Space

JavaScript ignores white space, much like HTML does.

You can place white space (spaces, paragraph returns) wherever you want to make your coding more readable.

But, like with css, adding space within variables and functions where they don't belong will break your code.

onMouseOver() ≠ on Mouse Over()

Syntax

Naming

Names of variables and functions must start with either a letter, an underscore (_) or a $ sign. Almost always they will start with a letter.

Names of functions and variables can’t contain spaces.

Acceptable ways of handling this is to use **underlines**:

this_Is_My_Function

or use **camelCase**:

thisIsMyFunction

where the first letter is lower case, and the beginning letter of each word is capitalized with no spaces between the words.

Names are best when they explain the variable or function.

functionA is not as useful as **numberOfHorses**

Syntax

Statements

Statements end in semicolons, like an attribute in CSS:

```
var test1 = 1;
var test2 = 2;
if (test1 == test2) {
    alert("test1 = test2");
} else {
    alert("test1 ≠ test2");
};
```

Syntax

Comments

this_Is_My_Function //Comment goes to end of line

thisIsMyFunction   /* Comment can go for multiple lines,
In this case could discuss camelCasing in depth… */

Note that JavaScript commenting are the same as
CSS commenting.

Writing JavaScript

Basic Scripts

```
<!DOCTYPE html>
<html lang="en-US">
    <head>
        <meta charset="utf-8">
        <title>My First JavaScript</title>
    </head>
    <body>
        <script>
            var hello = "Hello, world!"; // var = variable
            var name = "My name is Professor Nettles.";
            document.write(hello + " " + name);
        </script>
    </body>
</html>
```

Basic Scripts

We can also add html coding to our script within the quotes:

```
<script>
    var hello = "<p>Hello, world!</p>";
    var name = "<p>My name is Professor Nettles.</p>";
    document.write(hello + name);
</script>
```

Basic Scripts

Variables in JavaScript are "loosely typed"; in other words, you put the data in a variable, and JavaScript figures out what it is. Types of info that can be contained in a JavaScript variable are:

- **Numeric** — integer or floating decimal: var horse=5;
- **String** — text within quotes: var horse="palomino"
- **Boolean** — either TRUE or FALSE

```
var test = 0 == 1  // test = false
```

Special content:

```
var horse = null;    // null — no content
var horse = ""; // defined but no content
```

Basic Scripts

Here is a simple numeric script:

```
<script>
    var cellphones = 150;
    var houses = 20;
    document.write(cellphones/houses);
</script>
```

Combining variables like this is called "concantination."

Basic Scripts

Here is a combined numeric and string script:

```
<script>
    var cellphones = 150;
    var houses = 20;
    prefix = "There are ";
    suffix = " cellphones in each house.";
    document.write(prefix + cellphones/houses + suffix);
</script>
```

Again, this is an example of concatination.

It's best to name all the variables at the beginning of your script, even if you don't give them initial values.

Basic Scripts

Something to remember about variables in JavaScript is that they act just like cascading style sheets (css); you can assign different values to a variable in an equation, and the last value will be the value of the variable:

```
<script>
    var cellphones = 150;
        document.write(cellphones + "<br>");
    var cellphones = 20;
        document.write(cellphones + "<br>");
    var cellphones = "No one has a telephone any more.";
        document.write(cellphones);
</script>
```

The variable cellphones is finally equal to "No one has a telephone any more."

Variable vs. Function

Conditional Statements

Decision-making vs Looping

Decision-making statements include:

- if
- if/else
- if/else if
- switch

Looping statements include:

- while
- for

If

Sample if script:

```
<script>
    var age=22;
    if (age>21) {
        document.print("My age is " + age);
        alert("Let's Party!");
    }
</script>
```

if/else

Sample if/else script:

```
<script>
    var age=21;
    if (age>=21) {
        document.print("I am over age 21");
        alert("Let's party!");
    } else {
        document.print("I am under age 21");
        alert("Time to do my homework!");
    };
</script>
```

if/else if

Sample if/else if script:

```
<script>
    var age = eval(prompt("What is your age? ",""));
    if (age<21) {
        document.write("I'm " + age + "; I'm too young to party!");
    } else if (age==21) {
        document.write("I'm " + age + "; let's party!");
    } else {  //default
        document.write("I'm " + age + "; I'm too mature to party!");
    }
</script>
```

Loops

Loops

Loops allow the script to move through a statement multiple times until a counting variable tells it to stop.

while

Sample while loop:

```
<script>
    var i=0;   //initialize loop counter
    while(i<10) {
        document.write(i + "<br/>");
        i++;   // (i++ is equivalent to i = i + 1)
    }
</script>
```

for

Sample for loop:

```
<script>
    for(var i=0; i<10; i++ ) {
    document.write(i + "<br/>");
}
    //for each instance that this is true, do this;
      stop when it is no longer true
    //var 1=0 creates an initial variable with value 0
    //for as long as i<10, run the loop
    //i++ adds 1 to i each time it loops
</script>
```

Now, what would you have to do to get this
to count down from 10 to 0?

Affecting HTML

Adding output to HTML objects

We've been using:

```
    document.write(variable to be written);
```

to output the results of our javascript equations.

We can also target specific parts of our html to place the results.
Here is an example:

```
    <p id="demo"></p>
    <script>
        for(var i=0; i<10; i++ ) {
        document.getElementById("demo").innerHTML=(i + "<br/>");
            }
    </script>
```

Notice in this case we only get "9". This is because this method overwrites each time it is executed.

Adding output to HTML objects

Notice in this case we only get "9". This is because this method overwrites the previous text each time it is executed.

To keep all the numbers, we can change it to:

```
<p id="demo"></p>
<script>
    for(var i=0; i<10; i++ ) {
    document.getElementById("demo").innerHTML+=(i + "<br/>");
        }
</script>
```

Notice the "+=" instead of the "+".

(This isn't always the best way, but works well in this instance.)

Functions

Functions

Functions are self-contained units of code designed to accomplish a specific task.

They can be used over and over again, so you don't have to type them over and over again.

Functions aren't executed until called, and return a value.

Variables vs. Functions

What we have been working with are **Variables**:

```
var name = "Deane Nettles;
var title = "professor";
var output = name + ", " + title;
document.write(output);
```

However, if you want to use that function over again without retyping it, you can use it to create a **Function.**

```
function output(name, title) {  //names function, order of array
    var comma = ", "
    return name + comma + title; }
document.write(output ("Deane Nettles", "professor"));
```

(I am using document.write to keep the code short.)

Functions

Here's a function from a previous script.

```
<head>
    <script> // functions are usually placed in the head
            function forLoop() { // names function
                                 // calculation in curly brackets
                for(var i=0; i<10; i++ ) {
                    document.write(i + "<br/>");
                }
            }  // end of function
    </script>
</head>
<body>
    <script>
        forLoop(); // since script is in the head, it doesn't
                // automatically get called; this script calls function
    </script>
</body>
```

Functions

```
<head>
    <style>
        #soundslike {outline:1px; padding:40px; font-size:60px; float:left;}
    </style>
       <script>
         function duck() {
                  var walksLike = "duck";
                        var soundsLike = document.getElementById("soundslike");
                  if (walksLike == "dog") {
                        soundsLike.innerHTML = "Woof! Woof!";
                  } else if (walksLike == "duck") {
                        soundsLike.innerHTML = "Quack, Quack";
                  } else {
                        soundsLike.innerHTML = "Crickets...";
                  }
            }
       </script>
</head>
<body>
      <div id="soundslike"></div>
      <script>
            duck();
      </script>
</body>
```

Functions (mobile app)

```
<head>
<meta charset="utf-8">
<script>
    function mileage(miles, gas){
        var result = miles/gas;
        return result;
    }
</script>
</head>
<body>
    <script>
        var miles = eval (prompt ("How many miles did you drive? ",""));
        var gas = eval (prompt ("How much gas did you use? ",""));
        var rate = mileage (miles, gas);
        document.write ("You drove " + miles + " miles and used " +
           gas + " gallons of gas,<br>so your mileage is " + rate +
           " miles per gallon.");
    </script>
</body>
```

Functions using button onclick

head

```
<script>
    function sing () {
        var drink = "Energy Drink";
        var lyrics= "";
        var cans = eval(prompt("How many cans? ",""));
        var header = "<h1>" + cans + " Bottles of " + drink + "</h1>";
        while (cans > 0) {
            lyrics = "<p>" + lyrics + cans + " cans of  " + drink + " on the wall, <br>";
            lyrics = lyrics + cans + " cans of  " + drink + ";<br>";
            lyrics = lyrics + "Take one down, and pass it around,<br>";
            if (cans > 1) {
                lyrics = lyrics + (cans - 1) + " cans of  " + drink + " on the wall!</p>";
            }
            else {
            lyrics = lyrics + "No more cans of  " + drink + " on the wall!</p>";
            }
            cans = cans - 1;
        }
        document.getElementById("header").innerHTML=header;
        document.getElementById("lyrics").innerHTML=lyrics;
    }
</script>
<body>
       <button onclick="sing()" style="font-size:16px;">Click me</button>
       <div id="header"></div>
       <div id="lyrics"></div>
</body>
```