Intro

- **JavaScript is premier client-side scripting language used in Web development**
  - **Note especially**
    - **Client side**
    - **Focus on web development**
    - **Scripting**
- **Part of the client-side ‘triangle’ consisting of (X)HTML, CSS and of course JavaScript**
  - **Manipulation of mark-up and style via the *document object model* or DOM**

First Look at JavaScript - Helloworld

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>JavaScript Hello World</title>
<meta http-equiv="content-type" content="text/html;
   charset=ISO-8859-1" />
</head>
<body>
<h1>First JavaScript</h1>
<hr />
<script type="text/javascript">
  document.write("Hello World from JavaScript!");
</script>
</body>
</html>
```

Helloworld Deconstructed

- **<script> tag used to delimit the script code from the HTML**
  - **The script tag causes the browser's JavaScript interpreter to be invoked, the script run and any output produced**
  - **The browser is considered the "host" environment**
    - **There are other hosts for JavaScript and its variants**

- **The demo also shows how the script can write back out to the document in this case using the document.write( ) method**

Helloworld Deconstructed

- **The interplay between (X)HTML and JavaScript can be tricky at first**

```
<script type="text/javascript">
// Careful on tag and script intermixture
<strong>
  document.write("Hello World from JavaScript!");
</strong>
</script>
```

- **Instead you would do**

```
<script type="text/javascript">
 document.write("<strong>Hello World from    JavaScript!</strong>");
</script>
```

- **or even**

```
<strong>
<script type="text/javascript">
 document.write("Hello World from JavaScript! ");
</script>
</strong>
```

Using the <script> Tag

- You can use as many **<script>** tags as you like in both the **<head>** and **<body>** and they are executed sequentially.

- ```
  <h1>Ready start</h1>
  <script language="Javascript" type="text/javascript">
      alert("First Script Ran");
  </script>
  <h2>Running…</h2>
  <script language="Javascript" type="text/javascript">
      alert("Second Script Ran");
  </script>
  <h2>Keep running</h2>
  <script language="Javascript" type="text/javascript">
      alert("Third Script Ran");
  </script>
  </h1>Stop!</h1>
  </body>
  ```
  ```

<script> Tag in the <head>

- Given top-down read (and execution) often script is found in the <head> of an (X)HTML document

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>JavaScript in the Head</title>
<meta http-equiv="content-type" content="text/html;charset=ISO-8859-1" />
<script type="text/javascript">
   function alertTest() {
    alert("Danger! Danger! JavaScript Ahead");
   }
</script>
</head>
<body>
<h2>Script in the Head</h2>
<hr />
<script type="text/javascript">
   alertTest();
</script>
</body>
</html>
```

Script masking and <noscript>

- Script Hiding using HTML and JavaScript comments
  - <script type="text/javascript">
    <!--
      put your JavaScript here
    //-->
    </script>
  - Avoids printing script onscreen in non-script aware browsers

- **<noscript>** Element
  - Useful to provide alternative rendering in browsers that have script off or don't support script
  - <noscript>
      Either your browser does not support JavaScript or it is currently disabled.
    </noscript>

  - Next example shows a great way to keep non-JavaScript aware users out of your site

Script masking and <noscript>

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>JavaScript Masked</title>
<meta http-equiv="content-type" content="text/html; charset=ISO-8859-1" />
</head>
<body>
<script type="text/javascript">
<!--
 document.write("Congratulations! If you see this you have
   JavaScript.");
//-->
</script>
<noscript>
 <h1 class="errorMsg">JavaScript required</h1>
 <p>Read how to <a href="/errors/noscript.html">rectify this
   problem</a></p>
</noscript>
</body>
</html>
```

Meta Refresh Trick with <noscript>

- Change the <head> to contain a meta refresh to automatically redirect the user to an error page if the script is off

- Copy this into every page into your site and you can improve the chances users have script on

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Needs JavaScript</title>
<meta http-equiv="content-type" content="text/html; charset=ISO-8859-1" />
<noscript>
    <meta http-equiv="Refresh" content="0;URL=/errors/noscript.html">
</noscript>
</head>
```

- Downsides
  - Consider non-script aware bots
  - Won't validate

Event Handlers

- **(X)HTML** defines a set of event handler attributes related to JavaScript events such as **onclick**, **onmouseover**, etc. which you can bind JavaScript statements to.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>JavaScript Events</title>
<meta http-equiv="content-type" content="text/html; charset=ISO-8859-1" />
</head>
<body onload="alert('page loaded');">
<form action="#" method="get">
 <div id="formfields">
   <input type="button" value="press me" onclick="alert('You pressed my
   button!');" />
 </div>
</form>
<p><a href="http://www.yahoo.com" onmouseover="alert('hi');">Yahoo!</a></p>
</body>
</html>
```

Linked Scripts

- Like linked style sheets you can store JavaScript code in a separate file and reference it
  - Use a .js file
  - Contains only JavaScript
  - Store these files like images in a common directory in your site (e.g. /scripts)
  - Linked scripts can be cached and “clean up” (X)HTML documents
  - Linked scripts do have problems under some browsers

Linked Script Example

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Linked Script</title>
<meta http-equiv="content-type" content="text/html; charset=ISO-
   8859-1" />
<script type="text/javascript" src="danger.js"></script>
</head>
<body>
<form action="#" method="get" id="form1">
 <div id="formfields">
   <input type="button" name="button1" id="button1"
          value="press me" onclick="alertTest();" />
 </div>
</form>
</body>
</html>
```

Linked Script Example Contd.

- In file danger.js you would have simply

```
function alertTest( )
{
  alert("Danger! Danger!");
}
```

JavaScript, (X)HTML, and CSS Link

- JavaScript very much relies on markup and CSS in browsers, in fact it manipulates objects that are created by the **correct** use of tags and style properties

- For example, the **document** object contains objects and collections corresponding to many of the tags in the (X)HTML document.
  - `document.forms[ ], document.images[ ], document.links[ ],` etc.
  - We can always jump directly to the object using something like `document.getElementById( )` under a DOM compliant browser

Basic Features Contd.

- Whitespace
  - Whitespace is generally ignored in JavaScript statements and between JavaScript statements but not always consider
    - x = x + 1 same as x =x + 1
    - s = typeof x; is same as s=typeof x but it not the same as s=typeofx; or s= type of x;
  - Return character can cause havoc
  - Given white space support by JavaScript some developers favor “crunching”

Basic Features Contd.

- Statements
  - A script is made up of individual statements
  - JavaScript statements are terminated by returns or semi-colons (;)
  - So `x = x+1;` `alert(x);` same as `x = x+1` `alert(x)`
  - Prefer to use semi-colons because if you reduce returns you run into problems `x=x+1 alert(x)` throws an error while `x=x+1;alert(x);` does not.

Blocks

- To group together statements we can create a block using curly braces { }. In some sense this creates one large statement

- Blocks are used with functions as well as larger decision structures like if statements

```
function add(x,y)
    {
      var result = x+y;
       return result;
     }
```

```
if (x > 10)
   {
     x= 0;
      y = 10;
       }
```

Variables

- Variables store data in a program
- The name of a variable should be unique well formed identifier starting with a letter and followed by letters or digits
- Variable names should not contain special characters or white space
- Variable names should be well considered
  - X versus sum
  - Some rules of programming might not follow on the Web?

Variables Contd.

- Define a variable using the var statement
  - `var x;`
- If undefined a variable will be defined on its first use
- Variables can be assigned at declaration time
  - `var x = 5;`
- Commas can be used to define many variables at once
  - `var x, y = 5, z;`

Arrays

- Access arrays by index value
  - `var myArray = new Array(4)`
  - `myArray[3] = "Hello";`
- Arrays in JavaScript are 0 based given
  - `var myArray2 = ["Thomas", true, 3, -47];`
  - myArray2[0] is “Thomas”, myArray[1] is true and so on
  - Given new Array(4) you have an array with an index running from 0 – 3

Expressions and Operators

- Make expressions using operators in JavaScript
- Basic Arithmetic
  - + (addition), - (subtraction/unary negation), / (division), * (multiplication), % (modulus)
- Increment decrement
  - ++ (add one) -- (subtract one)
- Comparison
  - >, <, >=, <= , != (inequality), == (equality), === (type equality)
- Logical
  - && (and) || (or) ! (not)

More Operators

- Bitwise operators (&, |, ^)
  - Not commonly used in JavaScript except maybe cookies?
- String Operator
  - + serves both as addition and string concatenation
  - document.write("JavaScript" + " is " + " great! ");
  - You should get familiar with this use of +
- Be aware of operator precedence
  - Use parenthesis liberally to force evaluations
  - var x = 4 + 5 * 8 versus x = (4+5) * 8

Flow Control

- Basic program execution control handled in JavaScript using the **if** statement
- **if** (*expression*) *true-case* or **if** (*expression*) *true-case;* else *false-case;*

```
if (x > 10)
    alert("x bigger than 10");
  else
    alert("x smaller than 10");
```

More on If Statements

- You can use **{ }** with **if** statements to execute program blocks rather than single statements

```
if (x > 10)
    {
       alert("X is bigger than 10");
       alert("Yes it really is bigger");
     }
```

- Be careful with **;**'s and **if** statements

```
if (x > 10);
    alert("I am always run!? ");
```

Switch Statements

- **If** statements can get messy so you might consider using a **switch** statement instead

- **switch** (*condition*)
  {
  case (value) : *statement(s)*
  break;
  …
  default: *statement(s)*;
  }

- The **switch** statement is not supported by very old JavaScript aware browsers (pre-JavaScript 1.2), but today this is not such an important issue

Switch Example

```
var x=3;
    switch (x)
    {
      case 1: alert('x is 1');
              break;
      case 2: alert('x is 2');
              break;
      case 3: alert('x is 3');
              break;
      case 4: alert('x is 4');
              break;
      default: alert('x is not 1, 2, 3 or 4');
    }
```

Loops

- JavaScript supports three types of loops: **while**, **do/while**, and **for**
- Syntax of while:

  while(*condition***)**
  statement(s)

- Example:

```
var x=0;
    while (x < 10)
    {
      document.write(x);
      document.write("<br />");
      x = x + 1;
    }
    document.write("Done");
```

For Loop

- The most compact loop format is the **for** loop which initializes, checks, and increments/decrements all in a single statement

```
for (x=0; x < 10; x++)
   {
     document.write(x);
   }
```

- With all loops we need to exercise some care to avoid infinite loops. See example

For/In Loop

- One special form of the for loop is useful with looking at the properties of an object. This is the **for/in** loop.

```
for (var aProp in window)
    {
      document.write(aProp)
      document.write("<br />");
    }
```

- We will find this construct useful later on when looking at what we can do with a particular object we are using

Loop Control

- We can control the execution of loops with two statements: **break** and **continue**
- **break** jumps out of a loop (one level of braces)
- **continue** returns to the loop increment

```
 var x=0;
while (x < 10)
{
   x = x + 1;
   if (x == 3)
     continue;

   document.write("x = "+x);
   if (x == 5)
     break;
}
document.write("Loop done");
```

Functions

- Functions are useful to segment code and create a set of statements that will be used over and over again The basic syntax is

  function *name*(*parameter list*)
  {
  function statement(s)
  return;
  }

- For example

```
function add(x, y)
    {
      var sum = x + y;
      return sum;
    }
```

Functions Contd.

- We can then invoke a function using the function name with ( )'s

```
var result = add(2, 3);
```

- We can also pass variable values as well as literals

```
var a = 3, b=5;
var result;
result = add(a,b);
```

- Variables are passed to function by value so you must use return to send things back.
- You can return a value or not from a function and you can have as many return statements as you like

Input/Output in JavaScript

- Special dialog forms
  - Alert
    - alert("Hey there JavaScript coder! ");
  - Confirm
    - if (confirm('Do you like cheese?')
      alert("Cheese lover");
      else
      alert("Cheese hater");
  - Prompts
    - var theirname = prompt("What's your name? ", " ");

Input/Output in JavaScript Contd.

- Writing to the HTML document
  - document.write()
  - document.writeln()

- Writing should be done before or as the document loads.
- In traditional JavaScript the document is static after that, though with the DOM everything is rewritable
- Since we are writing to an (X)HTML document you may write out tags and you will have to consider the white space handling rules of (X)HTML

Comments and Formatting

- When writing JavaScript you may want to include a comment
  - `/* This is a multiple line style comment */`
  - `// This is a single line comment`
- You also may want to format your script for nice reading or you may want to crunch it for fast download?