UNOBTRUSIVE JAVASCRIPT

CAN MAKE

HTML

DO MAGICAL THINGS

WITHOUT GOING & BLOOMIN' WELL RUINING

THE EXPERIENCE

FOR THOSE LESS FORTUNATE

Web Technology

Lecture 10: Introduction to JavaScript

What is JavaScript

- JavaScript's official name is ECMAScript.
- ECMAScript is developed and maintained by the ECMA organization.
- Everyone can use JavaScript without purchasing a license
- JavaScript was designed to add interactivity to HTML pages
- Is scripting language (a lightweight programming language)
- JavaScript is an interpreted language (means that scripts execute without preliminary compilation)

Are Java and JavaScript the same?

- NO!
- Java and JavaScript are two completely different languages in both concept and design!
- Java (was developed by Sun Microsystems now Oracle) is a powerful and much more complex programming language - in the same category as C and C++.

What can JavaScript do?

- JavaScript gives HTML designers a programming tool.
- JavaScript can put dynamic text into an HTML.
- JavaScript can react to events.
- JavaScript can read and write HTML elements.
- JavaScript can be used to validate data.
- JavaScript can be used to detect the visitor's browser.
- JavaScript can be used to create cookies.

JavaScript into an HTML page

- To insert a JavaScript into an HTML page, we use the `<script>` tag.
- So, the `<script>` and `</script>` tells where the JavaScript starts and ends.
- The `document.write` command is a standard JavaScript command for writing output to a page.
- By entering the `document.write` command between the `<script>` and `</script>` tags, the browser will recognize it as a JavaScript command and execute the code line.

```
<html>
 <body>
  <script>
     document.write("<h1>Hello World!</h1>");
  </script>
 </body>
</html>
```

Where to Put the JavaScript

- JavaScript in a page will be executed immediately while the page loads into the browser.
- This is not always what we want.
- Sometimes we want to execute a script when a page loads, or at a later event, such as when a user clicks a button.
- We insert the script inside a function.

Scripts in `<head>`

- Scripts to be executed when they are called, or when an event is triggered, are placed in functions.
- Put your functions in the head section, this way they are all in one place, and they do not interfere with page content.

```
<html>
<head>
<script>
function message()
{
alert("This alert box was
called");
}
</script>
</head>

<body onload="message()">
</body>
</html>
```

Scripts in `<body>`

- If you don't want your script to be placed inside a function, or if your script should write page content, it should be placed in the body section.

```
<html>
<head>
</head>

<body>
 <script>
 document.write("This message is
written by JavaScript");
 </script>
</body>

</html>
```

Scripts in `<head>` and `<body>`

- You can place an unlimited number of scripts in your document, so you can have scripts in both the body and the head section.

```
<html>
<head>
<script>
function message()
{ alert("This alert box was
called "); }
</script>
</head>
<body onload="message()">
<script>
document.write("Hello!");
</script>
</body>
</html>
```

Using External JavaScript

- If you want to run the same JavaScript on several pages, without having to write the same script on every page, you can write a JavaScript in an external file.
- Save the external JavaScript file with a .js file extension.
- To use the external script, point to the .js file in the "src" attribute of the `<script>` tag:

```
<html>
<head>
    <script src="scriptname.js"></script>
</head>
</html>
```

JavaScript Statements

- Unlike HTML, JavaScript is case sensitive - therefore watch your capitalization closely when you write JavaScript statements.
- A JavaScript statement is a command. The purpose of the command is to tell the browser what to do.
- This JavaScript statement tells the browser to write "Hello" to the web page: `document.write("Hello");`
- It is normal to add a semicolon at the end of each executable statement.

JavaScript Code

- JavaScript code (or just JavaScript) is a sequence of JavaScript statements.
- Each statement is executed by the browser in the sequence they are written.
- This example will write a heading and two paragraphs to a web page:

```
<script>
   document.write("<h1>This is a heading</h1>");
   document.write("<p>This is a paragraph.</p>");
   document.write("<p>This is another
                                paragraph.</p>");
</script>
```

JavaScript Blocks

- JavaScript statements can be grouped together in blocks.
- Blocks start with a left curly bracket {, and ends with a right curly bracket }.
- The purpose of a block is to make the sequence of statements execute together.
- This example will write a heading and a paragraph to a web page:

```
<script>
{
   document.write("<h1>This is a heading</h1>");
   document.write("<p>This is a paragraph.</p>");
}
</script>
```

JavaScript Variables

- Do you remember algebra from school? x=5, y=6, z=x+y
- Do you remember that a letter (x) could be used to hold a value (5), and that you could use the information above to calculate the value of z to be 11?
- These letters are called **variables**, and variables can be used to hold values (x=5) or expressions (z=x+y).

JavaScript Variables

- As with algebra, JavaScript variables are used to hold values or expressions.
- A variable can have a short name, like **x**, or a more descriptive name, like **carname**.
- Rules for JavaScript variable names:
  - Variable names are case sensitive (y and Y are two different variables)
  - Variable names must begin with a letter or the underscore character
- **Note:** Because JavaScript is case-sensitive, variable names are case-sensitive.

Declaring JavaScript Variables

- Creating variables in JavaScript is most often referred to as "declaring" variables.
- You can declare JavaScript variables with the **`var`** keyword:

```
var x;
var carname;
```

- After the declaration shown above, the variables are empty (they have no values yet). However, you can also assign values to the variables when you declare them:

```
var x=5;
var carname="Volvo";
```

- **Note:** When you assign a text value to a variable, use quotes around the value.

Assigning Values to Variables

- If you assign values to variables that have not yet been declared, the variables will automatically be declared.
- These statements:

```
x=5;
carname="Volvo";
```

- have the same effect as:

```
var x=5;
var carname="Volvo";
```

JavaScript Operators

- The assignment operator = is used to assign values to JavaScript variables.
- The arithmetic operator + is used to add values together.

```
y=5;
z=2;
x=y+z;
```

- The value of x, after the execution of the statements above is 7.

JavaScript Arithmetic Operators

- Arithmetic operators are used to perform arithmetic between variables and/or values.
- Given that **y=5**, the table below explains the arithmetic operators:

Operator	Description	Example	Result
`+`	Addition	`x=y+2`	`x=7`
`-`	Subtraction	`x=y-2`	`x=3`
`*`	Multiplication	`x=y*2`	`x=10`
`/`	Division	`x=y/2`	`x=2.5`
`%`	Modulus (division remainder)	`x=y%2`	`x=1`
`++`	Increment	`x=++y`	`x=6`
`--`	Decrement	`x=--y`	`x=4`

JavaScript Assignment Operators

- Assignment operators are used to assign values to JavaScript variables.
- Given that **x=10** and **y=5**, the table below explains the assignment operators:

Operator	Example	Same As	Result
=	x=y		x=5
+=	x+=y	x=x+y	x=15
-=	x-=y	x=x-y	x=5
*=	x*=y	x=x*y	x=50
/=	x/=y	x=x/y	x=2
%=	x%=y	x=x%y	x=0

The + Operator Used on Strings

- The + operator can also be used to add string variables or text values together.
- To add two or more string variables together, use the + operator.
```
txt1="What a very";
txt2="nice day";
txt3=txt1+txt2;
```
After the execution of the statements above, the variable `txt3` contains `"What a verynice day"`.
- To add a space between the two strings, insert a space into one of the strings:
```
txt1="What a very ";
txt2="nice day";
txt3=txt1+txt2;
```

Adding Strings to Numbers

- The rule is: If you add a number and a string, the result will be a string!

```
x=5+5;
document.write(x); shows: "10"

x="5"+"5";
document.write(x);  shows: "55"

x=5 + “is the number!";
document.write(x);    shows: “5 is the number”
```

Operator Precedence

- JavaScript Operator Precedence is similar to the Mathematical Operator Precedence and Associativity.
- The multiplication operator `*` has a higher precedence than `/` and the addition operator `+`, so the multiplication is performed before the addition.

  `4 + 3 * 2` (shows *10)*

  `(4 + 3) * 2` (shows 14)

  `(4 / 2) * 2` (shows 4)
- The `+` and `–` operator is "left associative", meaning that it is evaluated left to right, so the numeric subtraction is performed first.

  `8 - 6 + "some text"` (shows 2some text)

Comparison Operators

- Comparison operators are used in logical statements to determine equality or difference between variables or values.
- Given that **x=5**, the table below explains the comparison operators:

Operator	Description	Example
`==`	is equal to	`x==8` is false
`===`	is exactly equal to (value and type)	`x===5` is true `x==="5"` is false
`!=`	is not equal	`x!=8` is true
`>`	is greater than	`x>8` is false
`<`	is less than	`x<8` is true
`>=`	is greater than or equal to	`x>=8` is false
`<=`	is less than or equal to	`x<=8` is true

How Can it be Used

- Comparison operators can be used in conditional statements to compare values and take action depending on the result:

```
if (age<18)
  document.write("Too young");
```

What is the correct JavaScript syntax to display "Hello World" on the browser?

A. `"Hello World"`
B. `document.write("Hello World")`
C. `response.write("Hello World")`
D. `("Hello World")`

Logical Operators

- Logical operators are used to determine the logic between variables or values.
- Given that **x=6 and y=3**, the table below explains the logical operators:

Operator	Description	Example
&&	and	`(x < 10 && y > 1)` is true
||	or	`(x==5 || y==5)` is false
!	not	`!(x==y)` is true

Conditional Operator

- JavaScript also contains a conditional operator that assigns a value to a variable based on some condition:

```
variablename=(condition)?value1:value2
```

- For example:

```
greeting=(visitor=="M")?"Dear Member":"Dear ";
```

- If the variable **visitor** has the value of "M", then the variable **greeting** will be assigned the value "Dear Member" else it will be assigned "Dear".

Conditional Statements

- Very often when you write code, you want to perform different actions for different decisions.
- In JavaScript we have the following conditional statements:
  - `if` statement - use this statement to execute some code only if a specified condition is true.
  - `if...else` statement - use this statement to execute some code if the condition is true and another code if the condition is false.

If Statement

- Use the if statement to execute some code only if a specified condition is true.

```
 if (condition)
 {
     code to be executed if condition is true
  }
```

- For Example:

```
var d=new Date();
var time=d.getHours();

if (time<10)
{
  document.write("<b>Good morning</b>");
}
```

If...else Statement

- Use the if statement to execute some code only if a specified condition is true.

```
 if (condition)
    { code to be executed if condition is true }
 else
    { code to be executed if condition is not true }
```

- For Example: http://jsfiddle.net/edo77uk/4u358/

```
if (time < 10)
{
  document.write("Good morning!");
}
else
{
  document.write("Good day!");
}
```

JavaScript Loops

- Often when you write code, you want the same block of code to run over and over again in a row.
- In JavaScript, there are two different kind of loops:
  - `for` - loops through a block of code a specified number of times
  - `while` - loops through a block of code while a specified condition is true

The for Loop

- The for loop is used when you know in advance how many times the script should run.

```
for (var=startval;var<=endval;var=var+inc)
{
        code to be executed
}
```

The for Loop - Example

- The example below defines a loop that starts with `i=0`. The loop will continue to run as long as `i` is less than, or equal to 5. `i` will increase by 1 each time the loop runs.
- **Note:** The increment parameter could also be negative, and the `<=` could be any comparing statement.

```
var i=0;
for (i=0;i<=5;i++)
{
   document.write("The number is " + i);
   document.write("<br />");
}
```

The while Loop

- The while loop loops through a block of code while a specified condition is true.

```
while (var<=endvalue)
{
    code to be executed
}
```

- **Note:** The <= could be any comparing operator.

The while loop - Example

- The example below defines a loop that starts with `i=0`. The loop will continue to run as long as `i` is less than, or equal to 5. `i` will increase by 1 each time the loop runs:

```
var i=0;
while (i<=5)
{
   document.write("The number is " + i);
   document.write("<br />");
    i++;
}
```

http://jsfiddle.net/edo77uk/xSyKS/