What is JavaScript?

- JavaScript was designed to **add interactivity to HTML pages**
- JavaScript is usually **embedded directly into HTML pages**
- JavaScript is an **interpreted** language (means that scripts execute without preliminary compilation)

What can a JavaScript Do?

- **JavaScript gives HTML designers a programming tool** - JavaScript is a scripting language with a very simple syntax.
- **JavaScript can put dynamic text into an HTML page** – You can use JavaScript to add html code into your html page, e.g., `document.write("<h1>" + name + "</h1>")`
- **JavaScript can react to events** – You can make JavaScripts that execute when something happens, like when a page has loaded or when a user clicks on something.
- **JavaScript can read and write HTML elements** - A JavaScript can read and change the content of HTML.
- **JavaScript can be used to validate data** - A JavaScript can be used to validate form data before it is submitted to a server (e.g., make sure you’ve typed in a 7-digit). This saves the server from extra processing.
- **JavaScript can be used to detect the visitor's browser** - A JavaScript can be used to detect the visitor's browser, and - depending on the browser - load another page specifically designed for that browser.
- **JavaScript can be used to create cookies** - A JavaScript can be used to store and retrieve information on the visitor's computer.

How to Put a JavaScript Into an HTML Page

```html
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html lang="en-US" xml:lang="en-US" xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>JavaScript Guidelines</title>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
</head>
<body>

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

</body>
</html>

Note: If we had not entered the `<script>` tag, the browser would have treated the `document.write("Hello World!")` command as pure text, and just written the entire line on the page.
Example Explained

To insert a JavaScript into an HTML page, we use the `<script>` tag. Inside the `<script>` tag we use the `type` attribute to define the scripting language.

So, the `<script type="text/javascript">` and `</script>` tells the browser where the JavaScript starts and ends:

```html
<html>
<body>
<script type="text/javascript">
...  
</script>
</body>
</html>
```

The word `document.write` is a standard JavaScript command for writing output to a page. Document is an object in JavaScript. There are a bunch of objects in JavaScript, like forms, windows, etc. Each object has functions (or actions) associated with it. The document object has a function called “write” that writes to the document.

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. In this case the browser will write Hello World! to the page:

**HTML Comments to Handle Simple Browsers**

Browsers that do not support JavaScript will just display the JavaScript as page content.

To prevent this, and as a part of the JavaScript standard, use the HTML comment tag.

Add `<!--` before the first JavaScript statement, and a `-->` (end of comment) after the last JavaScript statement.

```html
<html>
<body>
<script type="text/javascript">
<!--
document.write("Hello World!");
//-->
</script>
</body>
</html>
```

`//` is the JavaScript comment symbol. This prevents JavaScript from executing the `-->` tag.
**Where to Put the JavaScript**

JavaScripts inside the `<body> </body>` tags in a page will be executed immediately when the page loads into the browser. (this is done like the example above).

Sometimes we want it to execute when a user triggers an event (for instance, we may want the script to execute only when we click on a button in a form or put our mouse pointer over a picture). In this case, we may put the scripts in between the `<head></head>` tags, and then call the script in the body tag.

**Scripts in the head section:** These scripts load ahead of time, so that when you call them, you don’t have to worry about them not having downloaded from the internet yet.

```html
<html>
<head>
<script type="text/javascript">
....
</script>
</head>
</html>
```

**Scripts in the body section:** Executed when the page loads into the browser.

**You can have Scripts in both the body and the head section:**

```html
<html>
<head>
<script type="text/javascript">
....
</script>
</head>
<body>
<script type="text/javascript">
....
</script>
</body>
</html>
```

**JavaScript is Case Sensitive**

JavaScript is case sensitive - therefore watch your capitalization closely when you write JavaScript statements, create or call variables, objects and functions.

**JavaScript Statements**

A JavaScript statement is a command to the browser. The purpose of the command is to tell the browser what to do.

This JavaScript statement tells the browser to write "Hello Dolly" to the web page:

```javascript
document.write("Hello Dolly");
```
It is normal to add a semicolon at the end of each executable statement. This is a good programming practice, and most often you will see this in JavaScript examples on the web.

However, the semicolon is optional and the browser is supposed to interpret the end of the line (a carriage return) as the end of the statement.

Note: Using semicolons makes it possible to write multiple statements on one line.

**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.

**Example**

```
<script type="text/javascript">
document.write("<h1>This is a header</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 using { and }.

The purpose is to make sequence of statements execute together. We use these { } to block off code usually in loops or in conditional (if) statements. In other words, the browser reads and executes statements in order from top to bottom unless we have a loop or a conditional statement. So in those cases we need to group the statements that we want executed in a possible different order to be blocked off, and we do that using { and }.

**Example**

```
<script type="text/javascript">
if (new == "true")
{
    document.write("<h1>This is a header</h1>");
    document.write("<p>This is a paragraph</p>");
    document.write("<p>This is another paragraph</p>");
}
else {
    document.write("<h1>Old Pro Section</h1>";
    document.write("<p>We already know the basics. </p>");
}
</script>
```
Note: When using { and }, it is customary to indent all statements between the { and }. I indent 3 spaces, some people indent 5 spaces – how much you indent is up to you. Indenting makes it a lot easier to see exactly which statements are included within a block. There will be more examples of indenting later.

**JavaScript Comments**

Comments are text that we DON’T want the browser to execute. So we put special symbols around that text so the browser knows which sections we don’t want executed. We usually put comments around plain old English statements that we’re using to remind ourselves of why we did something or what particular code should be doing (or possibly of code we’ll need to add later. Sometimes we put comments around code we think isn’t working currently to make sure the resto of our code is working. There are two ways to specify comments:

Single line comments start with //

```html
<!-- This will write a header:
document.write("<h1>This is a header</h1>");
// This will write two paragraphs:
document.write("<p>This is a paragraph</p>");
document.write("<p>This is another paragraph</p>");
</script>
```

**Multi-Line Comments**

```html
/* The code below will write one header and two paragraphs */
document.write("<h1>This is a header</h1>");
document.write("<p>This is a paragraph</p>");
document.write("<p>This is another paragraph</p>");
</script>
/* and end with */.
```

**Using Comments to Prevent Execution**

In this example the comments is used to prevent the execution of multiple code lines:

```html
/*
document.write("<h1>This is a header</h1>");
document.write("<p>This is a paragraph</p>");
document.write("<p>This is another paragraph</p>");
*/
</script>
```
Using Comments at the End of a Line

```javascript
// This will write "Hello"
// This will write "Dolly"
</script>

JavaScript Variables

Variables are "containers" for storing information.

Do You Remember Algebra From School?

Do you remember algebra from school? x=5, y=6, z=x+y

a letter (like x) could be used to hold a value (like 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)

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.

Example

```html
<html><head></head>
<body>
<script type="text/javascript">
var firstname;
firstname="Fred";
document.write(firstname + "<br/>");

firstname="Wilma";
document.write(firstname + "<br/>");
</script>
</body>
</html>
```
Declaring (Creating) JavaScript Variables

Creating variables is referred to as "declaring" variables.

declare variables with the **var statement**:

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

After the declaration shown above, the variables are empty (they have no values yet).

You can assign values to the variables when you declare them:

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

After the execution of the statements above, the variable `x` will hold the value `5`, and `carname` will hold the value `Volvo`.

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

Assigning Values to Undeclared JavaScript Variables

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

These statements:

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

have the same effect as:

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

JavaScript Operators

As with algebra, you can do arithmetic operations with JavaScript variables:

```javascript
y=x-5;
z=y+5;
```

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:

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
<th>Example</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>Addition</td>
<td>x=y+2</td>
<td>x=7</td>
</tr>
<tr>
<td>-</td>
<td>Subtraction</td>
<td>x=y-2</td>
<td>x=3</td>
</tr>
<tr>
<td>*</td>
<td>Multiplication</td>
<td>x=y*2</td>
<td>x=10</td>
</tr>
<tr>
<td>/</td>
<td>Division</td>
<td>x=y/2</td>
<td>x=2.5</td>
</tr>
<tr>
<td>%</td>
<td>Modulus (division remainder)</td>
<td>x=y%2</td>
<td>x=1</td>
</tr>
<tr>
<td>++</td>
<td>Increment</td>
<td>x=++y</td>
<td>x=6</td>
</tr>
<tr>
<td>--</td>
<td>Decrement</td>
<td>x=--y</td>
<td>x=4</td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>Operator</th>
<th>Example</th>
<th>Same As</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>x=y</td>
<td></td>
<td>x=5</td>
</tr>
<tr>
<td>+=</td>
<td>x+=y</td>
<td>x=x+y</td>
<td>x=15</td>
</tr>
<tr>
<td>-=</td>
<td>x-=y</td>
<td>x=x-y</td>
<td>x=5</td>
</tr>
<tr>
<td>*=</td>
<td>x*=y</td>
<td>x=x*y</td>
<td>x=50</td>
</tr>
<tr>
<td>/=</td>
<td>x/=y</td>
<td>x=x/y</td>
<td>x=2</td>
</tr>
<tr>
<td>%=</td>
<td>x%=y</td>
<td>x=x%y</td>
<td>x=0</td>
</tr>
</tbody>
</table>
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.

```javascript
txt1="What a very";
txt2="nice day";
txt3=txt1+txt2;
```

After the execution of the statements above, the variable txt3 contains "What a very nice day".

To add a space between the two strings, insert a space into one of the strings:

```javascript
txt1="What a very ";
txt2="nice day";
txt3=txt1+txt2;
```

or insert a space into the expression:

```javascript
txt1="What a very";
txt2="nice day";
txt3=txt1+" +txt2;
```

After the execution of the statements above, the variable txt3 contains:

"What a very nice day"

---

Adding Strings and Numbers

Look at these examples:

**Example**

```javascript
x=5+5;
document.write(x);
```

```javascript
x="5"+"5";
document.write(x);
```

```javascript
x=5+"5";
document.write(x);
```

```javascript
x="5"+5;
document.write(x);
```
The rule is:

If you add a number and a string, the result will be a string.

**JavaScript Comparison and Logical Operators**

Comparison and Logical operators are used to test for true or false.

**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:

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>==</td>
<td>is equal to</td>
<td>( x==8 ) is false</td>
</tr>
<tr>
<td>!=</td>
<td>is not equal</td>
<td>( x!=8 ) is true</td>
</tr>
<tr>
<td>&gt;</td>
<td>is greater than</td>
<td>( x&gt;8 ) is false</td>
</tr>
<tr>
<td>&lt;</td>
<td>is less than</td>
<td>( x&lt;8 ) is true</td>
</tr>
<tr>
<td>&gt;=</td>
<td>is greater than or equal to</td>
<td>( x&gt;=8 ) is false</td>
</tr>
<tr>
<td>&lt;=</td>
<td>is less than or equal to</td>
<td>( x&lt;=8 ) is true</td>
</tr>
</tbody>
</table>

Note the == operator. This is different from the = assignment operator. With =, we are assigning something a value. For instance, numb = 5; says the variable numb now holds the value 5, or we are setting the value inside of numb to 5. numb == 5 says something very different. It says, is the value inside of numb equal to the value 5? It returns yes (true) if the value inside of numb is equal to 5 and no (false) if the value in numb is not equal to 5. It does not set the value inside of numb to 5. Be careful about this – it is a common mistake to say something like,

```javascript
var numb = 10;
if (numb = 5) {
    document.write("numb is, in fact, 5");
}
else {
    document.write("numb is not 5");
}
```

In this example, the statement `document.write("numb is not 5");` will not get executed because the value in numb gets set to the value of 5 in the statement, numb = 5. Be careful of this!
How Can it be Used

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

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

Conditional Statements

Conditional statements in JavaScript are used to perform different actions based on different conditions.

Very often when you write code, you want to perform different actions for different decisions. You can use conditional statements in your code to do this.

In JavaScript we have the following conditional statements:

- **if statement** - use this statement if you want to execute some code only if a specified condition is true
- **if...else statement** - use this statement if you want to execute some code if the condition is true and another code if the condition is false
- **if...else if...else statement** - use this statement if you want to select one of many blocks of code to be executed
- **switch statement** - use this statement if you want to select one of many blocks of code to be executed

If Statement

You should use the if statement if you want to execute some code only if a specified condition is true.

**Syntax**

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

Note that if is written in lowercase letters. Using uppercase letters (IF) will generate a JavaScript error!

**Example 1**

```javascript
<script type="text/javascript">
//Write a "Good morning" greeting if
//the time is less than 10
var d=new Date();
var time=d.getHours();
```
if (time<10)
{
    document.write("<b>Good morning</b>");
}
</script>

Example 2
<script type="text/javascript">
//Write "Lunch-time!" if the time is 11
var d=new Date();
var time=d.getHours();

if (time==11)
{
    document.write("<b>Lunch-time!</b>");
}
</script>

Note: Remember, when comparing variables you must always use two equals signs next to each other (==)!

If...else Statement

If you want to execute some code if a condition is true and another code if the condition is not true, use the if....else statement.

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

Example
<script type="text/javascript">
//If the time is less than 10,
//you will get a "Good morning" greeting.
//Otherwise you will get a "Good day" greeting.
var d = new Date();
var time = d.getHours();

if (time < 10)
{
    document.write("Good morning!");
}
else
{
    document.write("Good day!");
}
</script>
If...else if...else Statement

You should use the if....else if...else statement if you want to select one of many sets of lines to execute.

Syntax

```javascript
if (condition1)
{
    code to be executed if condition1 is true
}
else if (condition2)
{
    code to be executed if condition2 is true
}
else
{
    code to be executed if condition1 and condition2 are not true
}
```

Example

```javascript
<script type="text/javascript">
var d = new Date();
var time = d.getHours();
if (time<10)
{
    document.write("<b>Good morning</b>");
}
else if (time>10 && time<16)
{
    document.write("<b>Good day</b>");
}
else
{
    document.write("<b>Hello World!</b>");
}
</script>
```

Or (for browser validation!!!)

```javascript
if (navigator.appName == 'Microsoft Internet Explorer')
{
    Alert('IE
' + navigator.userAgent);
}
else if (navigator.appName == 'Netscape')
{
    Alert('Netscape
' + navigator.userAgent);
} Else {
    Alert('Other Browser');
}
/* What we have here is an object called navigator, and the object has a property (like weight, height, etc.). In this case the property is the name of the application. So we can check to see if the navigator’s application name is “Microsoft Internet Explorer” or if it is “Netscape” or if it is something else. Here we just print out which browser is being used. However, we could modify the html code based on the browser. */
Alert Box

Alert dialog box – pops on the screen to display a message and stays on the screen until someone clicks the OK button.

Display an alert dialog box by calling the alert function and passing it the text to be displayed.

Syntax:

```javascript
alert("This is an alert box");
```

Confirm Box

A confirm box is often used if you want the user to verify or accept something.

When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed.

If the user clicks "OK", the box returns true. If the user clicks "Cancel", the box returns false.

Syntax:

```javascript
confirm("SomeText");
```

Example:

```html
<html>
<head>
<script type="text/javascript">
function disp_confirm()
{
    var r=confirm("Do you want chocolate?");
    if (r==true)
    {
        document.write("You pressed OK!");
    }
    else
    {
        document.write("You pressed Cancel!");
    }
}
</script>
</head>
<body>

<input type="button" onclick="disp_confirm()" value="Display a confirm box" />

</body>
</html>
```

/*Note: In this example, we put a button in the body. When you click on the body, you call the javascript that is located in the head section. In this case we’re using a JavaScript FUNCTION. We’ll talk more about functions shortly.*/
Prompt Box

A prompt box is an easy way for you to ask user to enter info into your javascript.

When a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed after entering an input value.

If the user clicks "OK" the box returns the input value. If the user clicks "Cancel" the box returns null.

Syntax:

```javascript
prompt("somertext","defaultvalue");
```

To get value typed in:

```javascript
Var holder;
Holder = prompt('enter number', '3');
```

Example:

```html
<html>
<head>
</head>
<body>
<script type="text/javascript">
<!--
Var userID
Var password
userID = prompt('enter user id', ' '); password = prompt(' enter password', ' '); if (userID == 'SpongeBob') {
  if (password == 'squarepants') {
    alert('Login valid: Hi ' + userID);
  } else {
    alert('Login Invalid');
  }
} //--></script>
</body>
</html>
```
Example2:

```html
<html>
<head>
<script type="text/javascript">
function disp_prompt()
{
  var name=prompt("Please enter your name","Harry Potter");
  if (name!=null & & name!="")
  {
    document.write("Hello " + name + ": How are you today?";)
  }
}
</script>
</head>
<body>
<input type="button" onclick="disp_prompt()" value="Display a prompt box" />
</body>
</html>
```

Assignment1: You’re running a store that sells 2 items: dog collars and dog treats. If the user wants to buy dog treats, you need to ask how many treats the user wants to buy. The cost of treats is a dollar per treat. If the user orders more than 30 treats, they get a five dollar discount.

If the user wants to buy a collar, you need to ask them what size they want. A small collar is $10, a medium is $12, and a large is $14.

After you’ve asked for the info and calculated the results, you must confirm the user to what they’ve ordered and the final cost.

**JavaScript Switch Statement**

If you have a lot of conditions, you can use a switch statement instead of an if…elseif… statement.

**Syntax**

```javascript
switch(n)
{
  case 1:
    execute code block 1
    break;
  case 2:
    execute code block 2
    break;
  default:
    code to be executed if n is different from case 1 and 2
}
```
First we have an expression $n$ (most often a variable), that is evaluated once. The value of the expression is then compared with the values for each case in the structure. If there is a match, the block of code associated with that case is executed. Use `break` to prevent the code from running into the next case automatically.

**Example**

```javascript
<script type="text/javascript">
// You will receive a different greeting based on what day it is. Note that Sunday=0,
// Monday=1, Tuesday=2, etc.
var d=new Date();
theDay=d.getDay();
switch (theDay)
{
    case 5:
        document.write("Finally Friday");
        break;
    case 6:
        document.write("Super Saturday");
        break;
    case 0:
        document.write("Sleepy Sunday");
        break;
    default:
        document.write("I'm looking forward to this weekend!");
}
</script>
```

Note with the switch statement we put the { and } around the ENTIRE switch statement but not around the individual cases. In the above example, we compare the value inside theDay with each value after the word case. So first we see if the value in theDay is equal to 5. If it is, we execute everything under the case statement until we hit either the break statement (which will kick execution to the first line after }, or until we hit the next case. If the value in theDay does not equal 5, we’ll compare it to the next case, which is 6. We’ll do this for each case and we’ll execute the one(s) in which we have a (with theDay and the case).