# Session 1: INTRODUCTION TO JAVASCRIPT WEB DEVELOPMENT

## JavaScript vs. Java

Javascript is NOT JAVA!!! JAVA and C# are both HIGH LEVEL LANGUAGES WHICH DEMAND A COMPILER TO COMPILE CODE INTO RUNTIME CODE. Javascript is more of an 'interpreted' language which is much more fluid and doesn't demand a heavyweight compiler but is executed more as a 'script' would be executed, line by line. The Javascript 'runtime' engine will run within the browser for example Chrome has a V8 engine which is also used to power NodeJS which executes Javascript both in the browser but also running on the operating system itself natively.

## JavaScript Evolution

Created 1995

Brendan Eich from Netscape/Mozilla

ECMAScript is the 'official name' for Javascript

2005 : explosion with 1) AJAX 2) jQuery

HTML5

NodeJS driving forward a completely new era of Javascript

Current version : ECMAScript 6

## Security Overview

Javascript as such on the CLIENT is NOT SECURE! You CANNOT TRUST CLIENT-SIDE JAVASCRIPT AS IT COULD BE HACKED OR CHANGED AT ANY TIME. So form validation etc will have to be performed both on the client but again on the server just to make sure data is safe. For example in particular with ‘code injection’ attacks where form data is submitted containing <script>..</script> or other eg SQL commands.

## JavaScript Versions and Browser Support

Please see <https://en.wikipedia.org/wiki/ECMAScript#Versions> for details of Javascript versions, which are officially called ECMAScript versions now.

Browser support can be a very technical and detailed matter with the best website on the planet to confirm if a feature is supported in a given browser being <http://CanIUse.com/> where you can search for an individual feature to check how it is supported – this goes for HTML, CSS and Javascript.

## JavaScript Engines

These are the browser engines which run Javascript.

* Check out this official guide to browser engines at <https://en.wikipedia.org/wiki/Web_browser_engine> and a comparison of engines at <https://en.wikipedia.org/wiki/Comparison_of_web_browser_engines>
* Chrome : Blink which is a fork of Webkit which Chrome used to use and is officially the most popular browser engine
* Opera : Blink like Chrome
* Gecko : Mozilla Firefox
* Internet Explorer : Trident
* Edge : EdgeHTML
* Safari : WebKit

# Session 2: LANGUAGE STRUCTURE

## Character Set

For web pages, HTML, CSS and Javascript the default character set is UTF-8 which provides a 2-byte code which can represent any character from any language on the planet, plus some symbols too, which means that in theory any language can be represented in computer form.

ASCII was the primitive one-byte representation of all basic English letters and characters, for example see this page here on ASCII 🡺 <http://www.ascii-code.com/> and also here on W3Schools at <http://www.w3schools.com/html/html_charset.asp>

UTF-8 is also called UNICODE. There is also a version called UTF-16 which is a character set used in many operating systems.

HTML pages : to use a non-standard character set the code would simply be <meta charset="ISO-8859-1">

## Case Sensitivity

HTML is CASE INsEnSiTiVe although <lower\_case> tags should always be used, for everything!

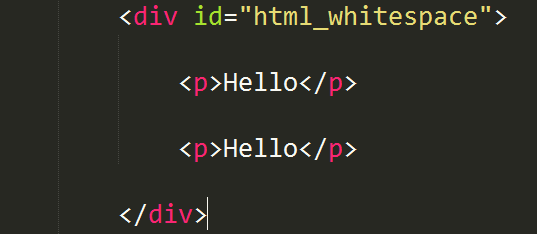
CSS is CASE INsEnSiTiVe

Javascript is CASE SENSITIVE so you must be EXTRA CAREFUL for example document.getElementById(“x”)

## Layout and Whitespace

Standard programming practice can and should be used to lay out good code for example:

* Declare variables at the top of the code
* Use comments
* White space is not normally an issue but be aware it can cause issues for example if parsing DOM elements then spaces (from your code) can be counted as DOM elements even though they might not be visible or have any effect on the page. They may influence (1) CSS layout 🡺 for example firstchild might be applying to a ‘space’ element (2) arrays of multiple elements eg document.forms.myForm.length 🡺 may be affected by non-visible DOM elements
  + See <http://www.w3schools.com/jsref/tryit.asp?filename=tryjsref_node_childnodes> for details
  + See HTML Whitespace lab also in index.htm



* + The above picture has 5 DOM elements (two paragraphs and three spaces)

## JavaScript Literals

A ‘literal’ is a fixed string that is no longer treated as a variable and cannot be changed without altering the string directly eg by adding or removing characters. Advanced discussion on literals can be found at <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Grammar_and_types> .

## Identifiers

Javascript identifiers are used to name variables and functions

First character of every Javascript variable/function must be

Letter of the alphabet (NOT A NUMBER)

$

\_

Note : Numbers are not allowed to start variables, neither are other special characters

Other letters may be the above (letter, $ and \_) plus numbers so \_123 is valid.

camelCase tends to be used in Javascript.

Please see discussion on Javascript syntax at <http://www.w3schools.com/js/js_syntax.asp>

## Reserved Words

Javascript’s reserved word list is as follows

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
| abstract | arguments | boolean | break | byte |
| case | catch | char | class\* | const |
| continue | debugger | default | delete | do |
| double | else | enum\* | eval | export\* |
| extends\* | FALSE | final | finally | float |
| for | function | goto | if | implements |
| import\* | in | instanceof | int | interface |
| let | long | native | new | null |
| package | private | protected | public | return |
| short | static | super\* | switch | synchronized |
| this | throw | throws | transient | TRUE |
| try | typeof | var | void | volatile |
| while | with | yield |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| alert | all | anchor | anchors | area |
| assign | blur | button | checkbox | clearInterval |
| clearTimeout | clientInformation | close | closed | confirm |
| constructor | crypto | decodeURI | decodeURIComponent | defaultStatus |
| document | element | elements | embed | embeds |
| encodeURI | encodeURIComponent | escape | event | fileUpload |
| focus | form | forms | frame | innerHeight |
| innerWidth | layer | layers | link | location |
| mimeTypes | navigate | navigator | frames | frameRate |
| hidden | history | image | images | offscreenBuffering |
| open | opener | option | outerHeight | outerWidth |
| packages | pageXOffset | pageYOffset | parent | parseFloat |
| parseInt | password | pkcs11 | plugin | prompt |
| propertyIsEnum | radio | reset | screenX | screenY |
| scroll | secure | select | self | setInterval |
| setTimeout | status | submit | taint | text |
| textarea | top | unescape | untaint | window |

## Operator Precedence

Basic BODMAS precedence will take place eg

Brackets first so 3+2=5, multiply by 4 squared = 16 to give 80 then add 21 divided by 7 = 3 to give 83

## Good practice and tips

For example see Google guide to Javascript here

<https://google.github.io/styleguide/javascriptguide.xml>

# Session 3: DATATYPES AND VALUES

## Numbers

var x=1;

## Strings

var x = “some string”;

## String Methods

Strings can be joined together using ‘+’ symbol or string1.concat(string2)

Strings can also be treated as arrays of characters and dissected as such.

myString.length will return the length of the string

## Booleans

var x = true; var y = false;

## JavaScript Functions

Var y = function(){};

Function y(){}

Note that () causes the function to be called straight away. This can be used to good effect in some callback functions where the function can be named without the brackets, thus is not executed until runtime.

See W3Schools <http://www.w3schools.com/js/js_functions.asp> and Mozilla for discussion on functions <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Functions> .

The arrow notation can also be used to define a function

() => {}

Param => expression when only one parameter is used

## Return

Each function can only return one object so if multiple objects are required to be returned from a function then a JSON object will have to be created and returned

Function x(){

Var y = { a:1,b:2}

Return y

}

## The this Keyword

Default refers to the ‘window’ object so console.log(this===window) will return true by default

This is useful inside an object

Var myObject= {

A:1

B:2

C:function(){this.A + this.B}

}

This can be useful when used with constructors

Function myClass(a,b){

This.a=a;this.b=b;

}

Var myNewObject = new myClass(1,2) 🡺 creates new object with myNewObject.a=1 .b=2

‘this’ object can be set manually using the call(), apply() and bind() methods

myFunction.call(object,param1,param2) will run myFunction but whatever object is passed in to the function can also be used to reference ‘this’ within the function.

myFunction.apply(object,[param1,param2])

var newFunction = myFunction.bind(object) adds a permanent binding

More detail available at <https://developer.mozilla.org/en/docs/Web/JavaScript/Reference/Operators/this> and <http://javascriptissexy.com/understand-javascripts-this-with-clarity-and-master-it/>

## Objects Introduction

Objects are declared using the JSON format

var myObject = { a:1, b:”mystring” }

Objects can hold any other data type within it, even other objects and arrays.

See <http://www.w3schools.com/js/js_objects.asp> for a basic guide to Javascript objects.

Once created object data is accessed using key-value pairs eg

myObject.a=1

myObject.b=”mystring”

## Arrays Introduction

Arrays are declared using

Var myArray = [1,2,3,”hi”];

Alternatively (not recommended) you can also use

Var myArray = new Array(1,2,3,”hi”);

Once created the array elements are accessed using a numeric index where zero [0] is the first item.

myArray[0] = 1

myArray[3] = “hi”

Javascript arrays can hold different data types

Please see <http://www.w3schools.com/js/js_arrays.asp> for basic discussion on arrays.

## Type Conversion

Javascript contains the following data types

* + string
  + number
  + boolean
  + function
  + object
    - object
    - array
    - date
  + null can assign eg to a string
  + undefined var x; once declared x is undefined until value is assigned

typeof() returns the type of your object

String(x) converts number x to string

x.toString converts number x to string

x.toExponential to power of 10

x.toFixed(2) fixed number of decimal places

x.toPrecision(3) string with 3 significant figures

Number(“1.23”) converts string to number

parseFloat(“1.23”) returns 1.23 decimal number

parseInt(“1.23”) returns 1 as integer

More detailed discussion at <http://www.w3schools.com/js/js_type_conversion.asp>

## Primitive Datatype Wrappers

A primitive is a DATA TYPE THAT IS NOT AN OBJECT AND ALSO HAS NO METHODS. They also are IMMUTABLE so cannot change.

Objects are collections of properties. Properties can reference objects or primitives. Primitives are VALUES ONLY.

Javascript primitive data types

* string
* number
* Boolean true or false
* Null Null
* Undefined Undefined

See <https://developer.mozilla.org/en-US/docs/Glossary/Primitive> for discussion on primitives

And <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures> on discussion on data types

A wrapper can be used to create a new object eg var Twelve = new Number(12) will create a valid number.

Avoid though as unpredictable results eg var myBoolean = new Boolean(false) will return true

# Session 4: VARIABLES, OPERATORS AND EXPRESSIONS

## JavaScript Variables

var x = 1;

If used within a function the SCOPE is LOCAL.

If used outside a function the SCOPE is GLOBAL

let x = 1; can be used inside a function block and the value is enclosed within that block only (cannot be accessed from outside)

Variables are ‘hoisted’ so put at the front of the code, no matter where you declare them.

For efficiency declare multiline eg var a=1,b=2;

Declare var x;

Initialize x=1;

const x = 1; value cannot be changed

In general let is now preferred over var because the intention of scope is clear. But const can also be used – see interesting discussion at <https://medium.com/javascript-scene/javascript-es6-var-let-or-const-ba58b8dcde75#.gx4l9m2jr> where const is recommended for all variables that are not intended to change.

## 

Var summary

Var declares all variables;

Let is a newer alternative and can be used to generate cleaner code

Const can be used for variables whose value is never intended to change

Var outside function => global scope

Var inside function => local scope

## JavaScript Expressions

* Expression PRODUCES A VALUE LIKE myvar or 1+2
* Statement performs an action like looping through a variable many times

See <http://www.2ality.com/2012/09/expressions-vs-statements.html> for a more detailed discussion

## Arithmetic Operators

+ / -

++ / -- add / subtract one

% remainder after division 10%3=1

\*\* to the power of eg 2\*\*3 = 2x2x2 = 8

## Relational Operators

Comparing one item to another and outputting a Boolean true or false depending if relationship is valid or not.

Var myObject={a:1,b:2}

“a” in myObject returns true

## String Operators

+ will concatenate strings

## Logical Operators

&& AND false && false 🡺 false

|| OR true || false 🡺 true

! NOT !(true) 🡺 false

## Bitwise Operators

Act on binary numbers with each individual binary bit being acted upon in a certain way

AND & 1&1=1 but 1&0=0

OR | 1&0=1 but 0&0=0

See more detail at <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Expressions_and_Operators#Bitwise_operators>

## Assignment Operators

These assign values

= a=6; assigns value 6 to a

+= a=a+6

Details on all assignment operators at <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Expressions_and_Operators>

## The global Object

DOM object operates with the global object being called the ‘Window’ object. All other objects are children of this.

With variables they can be declared outside of any function and this makes them available to the global namespace ie the root Window object.

var a=7; initialize a global object

window.a => will return 7

Note that by wrongly declaring an object without var will also make it a global object. This can be prevented by adding the syntax ‘using strict’ at the start of your Javascript file to force a stricter version of Javascript which prevents such errors from happening unnoticed.

# Session 5: JAVASCRIPT CONTROL STRUCTURES

## If/Else

if {}

else if {}

else {}

## For/For In

for (var i=0;i<10;i++){ // i }

var myObject={a:1,b:2}, item;

for(item in myObject){ console.log(myObject[item])}

## While

var x=0;

while(x<10){//do this;x++;}

## Do..While

var x=0;

do{//do this;x++}

while(x<10);

## Switch

var x = 3;

switch(x){

case 1://do this;break;

case 2: //do this;break;

case 3: // do this ; break;

default: // do this;

}

## Break and Continue

for (var i=0;i<10;i++){

If(i=7){break;} out of loop

If(i=3){continue;} break out of current loop and start next one straight away

}

See <http://www.w3schools.com/js/js_break.asp> for clarification

## JavaScript Exception Handling

try {

Myundefinedfunction(); // will generate a system exception

Throw(‘an error’) // manually throw exception

}

catch(exception e){

Console.log(e); // also e.data, also e.type

}

finally{}

See showcase/js\_35\_throw\_exception.htm as an example of manually throwing an exception and catching it

# Session 6: OBJECTS AND ARRAYS

## Object

## Common Properties and Methods

## Dates, Times and Timers

var d = new Date(); // standard way of creating a new date object

Displaying dates

d.getDay() Returns 0-6 where 0 is a Sunday and 6 is a Saturday

d.getDate() Returns 1-31 calendar date

d.getMonth Returns 0-11

d.getFullYear Returns eg 2016

d.getHours/Minutes/Seconds/Milliseconds()

d.getTimeZoneOffset Returns time zone difference away from GMT

d.getTime Returns milliseconds since 1 Jan 1970

d.set…. Sets time

See <http://www.w3schools.com/jsref/jsref_obj_date.asp> also <http://www.w3schools.com/js/js_dates.asp> for fuller discussion

See Mozilla reference at <https://developer.mozilla.org/en/docs/Web/JavaScript/Reference/Global_Objects/Date>

## Classes and Prototyping

Javascript can do classes but they are not fully supported in all browsers. Javascript itself it not an OOP so does not support classes natively. In order to create classes Javascript uses functions to do the same thing.

Declare a class

var myClass = function(){}

Instantiate a class

var myNewObject = new myClass()

Declare a class with properties

var myClass=function(x,y){

this.x=x;

this.y=y;

}

Instantiate : var myNewObject = new myClass(1,2);

Class Methods

Can be added in the class declaration body but better for memory usage not to do this but to delcare afterwards as a new prototype object. Prototypes can be used to extend any Javascript object and not just classes.

myClass.prototype.myMethod = function(){ console.log (this.x) }

Call the method using myNewObject.myMethod() which in this case will log this.x which is 1 to the console.

Please see discussion at <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Introduction_to_Object-Oriented_JavaScript> for further details including inheritance.

Static Methods can be used which like in regular OOP are only valid when called as part of the class and not as part of an instantiated object. See <https://developer.mozilla.org/en/docs/Web/JavaScript/Reference/Classes> for details.

Extends keyword also exists to either extend a class or provide Parent-Child inheritance features similar to regular OOP inheritance. More details again at <https://developer.mozilla.org/en/docs/Web/JavaScript/Reference/Classes>

## Arrays

## Array Methods

Size of array

myArray.length;

Add and remove items from an array

myArray.push(x) add to end

myArray.pop() remove item from end

myArray.unshift(x) add to start

myArray.shift() remove (shift) item from the start of the array

Output array items as a string

myArray.join()

myArray.sort()

myArray.reverse()

myArray.indexOf(‘hi’) Search the array for this element and return position of hit

myArray.forEach() Call a function for each array element

myArray.forEach(function(item,index,array){ } //index=0,1,2 etc, item will be value at each index.

# Session 7: JAVASCRIPT FUNCTIONS

## Defining Functions

var y = function(){}

() => {}

function y(){}

## Invoking Functions

y don’t call the function right now

y() call and invoke the function right now

## Functions as Data

Functions can be treated just the same as any other object, and if they return data objects that data is the same also

## Anonymous Functions

Named functions have traceability for example when doing performance measurements we can observe the names of functions which are consuming processing resources. However it is not always best to have named functions; in some instances it may be the best thing to use anonymous functions for example when performaing a loop over and over again. Anonymous functions are just created and called in the same way as other functions

\*\*

## Passing Arguments to Functions

var y = function(a,b){}

var z = y(1,2)

# Session 8: INTRODUCTION TO REGULAR EXPRESSIONS

## The RegEx Object

Regular expressions permit us to do input validation with specific validation types catering for every conceivable different type of text and number string including special characters.

Regular Expressions come natively in HTML5 now with the input tag having also a ‘pattern’ tag with the syntax

<input type=”text” pattern="[A-Za-z]{3}">

Please see <http://www.w3schools.com/tags/tryit.asp?filename=tryhtml5_input_pattern> for a working demo.

## Methods and Usage

As well as the native HTML usage of regular expressions Javascript also permits internal checking of field values against a given Regular Expression. For these purposes we have a new data type called Regular Expression whose value takes the form / / where in between the slashes are whatever validation is required.

The syntax would be var myRegExp =/pattern/ where pattern consists of a series of letters and characters to represent the pattern to check against.

The simplest method would be to check for one string within the input and this would be coded

var myRegExp = /mystring/

This would find one match but to find multiple matches the /g can be added ie

var myRegExp = /mystring/g

See details at <http://www.w3schools.com/jsref/jsref_obj_regexp.asp>

## Patterns Matching Examples

Please see sites <https://regexone.com/> and <http://regexr.com/> for good places to do this dynamically online before building your own offline.

Simple validations would include

One number [0-9]

\d

One letter [a-zA-Z]

Two letters [0-9]{2}

Two letters and two numbers [a-zA-Z]{2}[0-9]{2}

nn-AA where nn is 2 numbers and AA is 2 letters [a-zA-Z]{2}-[0-9]{2}

Alphabetic or numeric input [a-zA-Z0-9]

\w

Alphabetic, numeric including symbols but not spaces \S (capital S)

Space \s

Match only at the start of the expression ^ eg ^[0-9]{4} matches 4 numbers at start only

Match only at the end of the expression $ eg [0-9]{4}$ matches last 4 numbers at end only

Match multiple characters (one or more) + eg \w+ will match multiple letters

Match zero or more \* b\w\* will match b, be and brown

Match 3 or more {3,}

Match 0 or 1 so item becomes optional ? colou?r will match color and colour as u optional

Match OR |

## Regular Expressions Tools

As mentioned above please see the sites <https://regexone.com/> and <http://regexr.com/> to get started with some great examples of Regular Expressions.

# Session 9: BROWSER-BASED JAVASCRIPT

## The Window Object

The Window Object is the root object in the DOM. All elements are sub-elements from this root object.

The DOM hierarchy goes

Window => Document => HTML => BODY => DIV => CONTENT

Documents can be referenced by their position in the DOM for example form elements can be referenced

Document.myForm.childNodes[1] etc if necessary although normally this would not be required. However if pages are going to be generated dynamically it might be very important to be able to count and iterate through the elements of a DOM object using this method.

Details of all methods supported by the window object are to be found here <http://www.w3schools.com/jsref/obj_window.asp>

## JavaScript Event Handling

Events are the driving force behind the success of today’s Javascript.

Events can be defined in two ways.

Firstly the straightforward way you are probably used to:

<input type=submit onclick=”fn\_run\_this\_validation()” />

Other simple events like this would be onclick, ondblclick, onmouseover, onmouseout, onkeydown, onkeyup, onkeypress, onload, onchange. A full list of ‘on’ Javascript events can be found here <http://www.w3schools.com/jsref/dom_obj_event.asp> and some full examples would be found here <http://www.w3schools.com/js/js_events_examples.asp>

However it is possible to add ‘event listeners’ in code outside of the HTML. This is, in general, considered better practice so it would be best practice to write in the HTML

<input id=”submit\_form” type=”submit” value=”submit”/>

Then in Javascript or jQuery code add an event listener which would have the syntax

document.getElementById(‘x’).addEventListener(“click”,function(){ // do this });

Notice that ‘onclick’ has now become ‘click’ where the difference will be ‘onclick’ is the event handler and ‘click’ is the event to listen for.

## JavaScript Form Handling

In order to handle a form correctly the <submit> button has to have the following syntax

<submit type=”submit” onsubmit=”return fnsubmit()”>

The return command gives the form handling code (Javascript) the ability to return either true or false and let the form submit (return true) or not (return false). This allows form validation to take place and to stop the submission of the form when the form is in error.

The function code would be a standard function with the syntax

function fnsubmit(){

var returnBoolean = false;

// code to validate form; if valid then set returnBoolean = true;

return returnBoolean;

}

This will permit the form code to return either true or false.

## The Document Object Model

Elements can be selected, inserted, changed and appended amongst other things, into the DOM.

For example to create an element one would use the syntax

document.createElement . . .

To append an existing item into the DOM one could use appendChild where one would have to select an item first, then append the new item to the selected item.

When selecting items in the DOM items can be selected individually with an ID or multiple items can be selected eg by className (document.getElementsByClassName) or for example by tagName (document.getElementsByTagName);

## Cookies

## Alternative Storage Options

# Session 10: JAVASCRIPT TOOLS

## Profiling JavaScript

Javascript can be ‘profiled’ to view the slow parts of the page.

For example Chrome, by hitting F12 and then choosing the ‘Timeline’ tab brings up this tool, already built-in.

Links for reference are

<https://developers.google.com/web/tools/chrome-devtools/evaluate-performance/timeline-tool#make-a-recording>

<https://developers.google.com/web/tools/chrome-devtools/evaluate-performance/timeline-tool#profile-js>

To record a timeline hit F12 => Timeline => CTL E to start and stop the recording. To avoid ‘noise’ from plugins (Chrome extensions) either use Incognito mode or create a new user profile (either new user or completely erase all existing Chrome settings). Keep the recording as short as possible.

Once recorded, the timeline may be examined by zooming in and out to the desired degree of complexity in order to view which code has been executing and taking up time.

You can select one item in order to get more detail below.

Firebug in Firefox has similar tools <https://developer.mozilla.org/en-US/docs/Tools/Performance>

## Firebug

Firebug is the equivalent of F12 for Chrome and has to be enabled for Firefox before it can be used.

<http://getfirebug.com/>

Details on how to use firebug here <https://getfirebug.com/whatisfirebug>

Details on profiling here <https://getfirebug.com/javascript> - click on Console then Profile to activate profiling, then click on Profile again to stop profiling and display a report on which functions have been called how many times etc.

## Fiddler

Fiddler is a paid performance tool but has a free offering which may be downloaded if you provide your email address.

Fiddler provides an active display of all web calls being made to and from your computer, with a detailed view of activity and protocols etc.

Install from <http://www.telerik.com/> and choose ‘Fiddler’ in the Products menu.

## YSlow

YSlow as the name suggests is a tool for finding out why your page is slow!

Checking out the github repo at <https://github.com/marcelduran/yslow/issues> it is not clear if this tool is still under active development or not (probably not).

However the tool runs and gives some amazing suggestions based on its analysis of your code which has run, for example combining CSS stylesheets, using CDN networks to deliver content, tips to make content cacheable etc

## Other JavaScript Tools

<http://jqueryui.com> provides some great ideas for providing jQuery features with minimal coding effort

Google Web Toolkit compiles Javascript from Java but has the advantage of optimising the Javascript for all browsers. Info and download the SDK at <http://www.gwtproject.org/overview.html?csw=1>

General overview of critical Javascript tools that a professional web developer today would be getting involved with are listed here <https://www.sitepoint.com/essential-tools-libraries-modern-javascript-developers/>

# Session 11: JAVASCRIPT EXTRAS

## NodeJS

NodeJS can be used to execute Javascript both on the server and on the client, thus removing the need for a further language to get involved. This ‘native’ processing of Javascript requests between a client and a server can allow for a great speed increase of background delivery, as node processes can take up a very small memory footprint. Node is extremely popular today and has taken off in a big way. Huge libraries available completely open source under MIT licence are available for free download at <https://www.npmjs.com/> . Source code for NodeJS is found at <https://nodejs.org/en/> and may be downloaded and installed here. Once installed, any node application can be run just by typing ‘Node <myapp.js>.

## AJAX Asynchronous Javascript and XML

Is the traditional way to update a web page without fully refreshing the page.

## Working with Structured Data

The web has defaulted to using first XML as the default structural way of passing data across the web, and also using PHP to execute MYSQL calls (or C# to execute SQL calls) but subsequently has levelled out using JSON as the default playing ground for sending data around the web. If in doubt JSON should be used as it also is the default data structure in the newer No-SQL databases such as MongoDB. These databases can cope with higher throughput of data than traditional SQL databases.

# JavaScript Libraries/Frameworks

## Intro

Javascript frameworks undergird SPA Single Page Applications which are kept in memory unlike traditional HTML applications which often are ‘stateless’.

## Angular.JS

Created 2009 by Google

Extends HTML attributes using Angular ‘directives’

Two-way data binding is at the core of Angular. When data is changed / entered on a form this goes through to the model, and vice-versa.

## React.JS

This is a competitor to Angular.js

React is behind Instagram and Facebook and is developed mostly by Facebook.

Released 2013.

Embodies a ‘Virtual DOM’

Focuses on the View part of MVC

Flux and Redux are part of the React suite of tools

## Ember.JS

Released 2011

MIT License

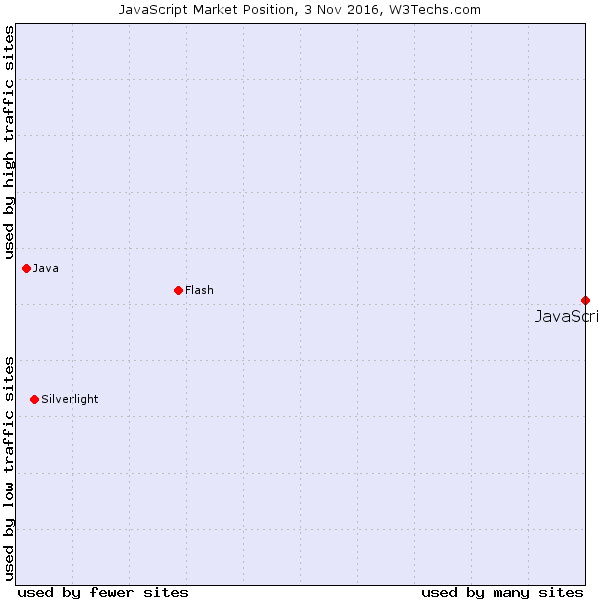
Created fastboot.js module

Could session 12 (Using JQUERY) be replaced with one that looks more at frameworks to help structure JavaScript (e.g. Model – View – View Model MVVM) as opposed to a library of useful functions. Possibly giving a quick overview of Angular, React or something similar to show the team how they work?”

Use of Javascript In The World Today

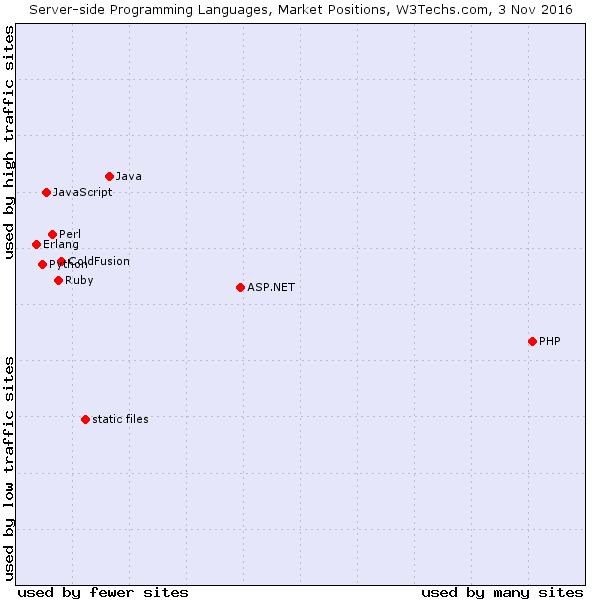
Javascript is in nearly every web page and has superceded all other web languages.

<https://w3techs.com/diagram/market_technology/cp-javascript>



Javascript on the SERVER still has a long way to go, but is GAINING GROUND FAST!

<https://w3techs.com/diagram/market/programming_language/10>



# Appendix

## Direct Web Remoting (DWR)

Not in active development; best perhaps avoided unless essential eg make a legacy Java application talk to a web page. Permits a server to generate Javascript code from Java and send to browser for rendering. Also permits the client to make calls to server which is running Java from a client which is running Javascript. The mechanism underlying the calls is AJAX.

Technology may be downloaded here <http://directwebremoting.org/dwr/index.html> .

## AHAH

AJAH is a subset of AJAX with reduce capabilities attempting to do the same thing but instead of using XML as the carrier it uses HTML. By including a file called ahah.js in your project which is only 20 lines long you can have a simpler form of AJAX working.

Links for further investigation

<http://www.openjs.com/articles/ajax/ahah_asynchronous_html_over_http/>

<http://microformats.org/wiki/rest/ahah>

<http://www.xfront.com/microformats/AHAH.html>

Example at

<http://www.xfront.com/microformats/examples/ahah/example01/Waldorf-Astoria-Factsheet.html>

Terms Discussed

IF..ELSE IF ..ELSE

DO.,.WHILE

WHILE

TRY.CATCH.FINALLY

UNDEFINED

STRING

CONST

VAR

LET

MODULAR

GLOBAL

LOCAL

% MODULUS

JSON

JQUERY

FOR

FOREACH

SWITCH

CASE

OR

AND

XOR

NOT

&&

||

BREAK

CONTINUE

DEFAULT

KEY

ITEM

VALUE

ARRAY

OBJECT

LENGTH

DATE

DATETIME

POP

PUSH

BOOLEAN

SLICE

SPLICE