Javascript

* Developed circa 1995 by *Brendan Eich* at Netscape Communications as the scripting language for the Netscape Navigator Browser
* Formerly called Mocha, then LiveScript, then JavaScript
* Standardized by ECMA International as ECMAScript
* Latest version: JavaScript 1.8.1, ECMAScript 5 [ECMA-262 5TH Edition]
* Common version: JavaScript 1.5, Jscript 5.5, ECMAScript v3 [ECMA-262 3rd Edition]
* JavaSscript Frameworks:
* script.aculo.us, jQuery, MooTools, Prototype, Dojo Toolkit, etc.
* Linked/Embedded in web pages using the <script> element
* Linked:

<script type=”text/javascript” src=”scripts.js” /> </script>

* Embedded (either in the <head> or the < body> element)

<script type=”text/javascript”>

<!—hide script from non-JavaScript browers…

/\* script code goes here \*/

// end of script hiding … 🡪

</script>

<noscript>

…content to show when scripting not available

</noscript>

* JavaScript + DOM/BOM + CSS + (X)HTML = DHTML
* JavaScript code in (X)HTML pages can be executed “on the fly” as the document is rendered (i.e. code outside of functions executes as the <script> element is encountered); in most cases though, JavaScript code is executed in response to document events (e.g. clicking a page element>.
* Basic language features:
* Paradigm:
* Object-oriented (prototype-based), functional, imperative scripting language
* Java-/C-like syntax
* Implicit semicolon insertion for statement termination
* Identifiers are alphanumeric,\_, and $ characters
* Single-line (//) or block (/\*\*/) comments
* Type system and variable scoping rules:
* Dynamic (aka loose or weak) typing
* Global (aka top-level) or local scopes
* Data types
* Primitive types
* Numbers (decimal, hexadecimal notation)
* Booleans (true, false)
* Strings (Single or Double quote delimited)
* Undefined and Null
* Composite (object) types
* Core JavaScript Objects
* Object, Number, Boolean, String, Date, Math, Global, RegExp, Error
* Arrays (Array)
* Functions (Function, Arguments)
* DOM Objects
* Anchor, Applet, Attr, Comment, DOMException, DOMImplementation,

DocumentFragment, Element, Event, Form, Image, Input, Layer, Link, Node, Option, Select, Style, Text, TextArea

* Keywords
* break, case, catch, continue, default, delete, do else, finally, for, function, if, in, instanceof, new, return, switch, this, throw, try, typeof, var, void, with
* Reserved words (currently unused)
* abstract, Boolean, byte, char, class, const, debugger, float
* Statements and control structures
* var
* Used to declare global/local scoped variables
* If-else
* Condition expressions having values of 0, “”, null, and undefined evaluate to false
* Switch-case-default-break
* allows any expression type to be used as the switch expression
* case labels may be of different types
* case labels may be expressions
* case execution falls-through, unless terminated by a break
* while, do-while, for, for-in, break, continue
* for while and do-while, false condition expressions similar to if-else
* for-in used for property enumeration
* allows labeled break/continue
* try-catch-finally, throw
* throw and catch can handle any expression type
* function, return
* JavaScript functions are similar to Java method except for the following differences:
* No return value type is specified, and return is optional within the function body
* Functions may return a value on one invocation and not return a value (i.e. have an unidentified return value) on another invocation
* Functions may return different types of values on different invocations
* Function parameters are dynamically typed
* Functions can be invoked with an arbitrary number of arguments, regardless of the actual parameters specified in the function definition (the Arguments object can be used to access unnamed arguments passed to the function invocation)
* Functions are first-class objects
* Functions can be invoked as global functins (i.e. as methods of the Global object) as

methods of specific objects or as object constructors

* with
* Used to access object properties without having to explicitly qualify the property with the object name (serves as shorthand notation for accessing object properties, at the expense of program readability)