*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: JavaScipt 1.8.1, ECMAScript 5 [ECMA-262 5th Edition]
* Common version: JavaScript 1.5, Jscript 5.5, ECMAScript v3 [ECMA-262 3rd Edition]
* JavaScript Frameworks:
  1. script.aculo.us, jQuery, MooTools, Prototype, Dojo Toolkit, etc.
* Linked/Embedded in web pages using the <script> element
  1. Linked:

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

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

<script type=”text/javascript”>

<!--hide script from non-JavaScript browsers…

/\* 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:

1. Paradigm:
   * Object-oriented (prototype-based), functional, imperative scripting language o Java-/C-like syntax
   * Implicit semicolon insertion for statement termination
   * Identifiers are alphanumeric, \_, and $ characters
   * Single-line (//) or block (/\*\*/) comments
2. Type system and variable scoping rules:
   * Dynamic (aka loose or weak) typing
   * Global (aka top-level) or local scopes
   * Data types:
     + Primitive types
   1. Numbers (decimal, hexadecimal notation)
   2. Booleans (true, false)
   3. Strings (Single or Double quote delimited)
   4. Undefined and Null

* Composite (object) types
  1. Core JavaScript Objects
     + Object, Number, Boolean, String, Date, Math, Global, RegExp, Error
     + Arrays (Array)
     + Functions (Function, Arguments)

1. 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
  1. Used to declare global/local scoped variables
* if-else
  1. Condition expressions having values of 0, “”, null, and undefined evaluate to false
* switch-case-default-break
  1. allows any expression type to be used as the switch expression
  2. case labels may be of different types
  3. case labels may be expressions
  4. case execution falls-through, unless terminated by a break
* while, do-while, for, for-in, break, continue
  1. for while and do-while, false condition expressions similar to if-else
  2. for-in used for property enumeration
  3. allows labeled break/continue
* try-catch-finally, throw
  1. throw and catch can handle any expression type
* function, return
  1. JavaScript functions are similar to Java methods 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 undefined 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
  2. Functions can be invoked as global functions (i.e. as methods of the Global object) as methods of specific objects, or as object constructors
* with
  1. 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)

*Asynchronous JavaScript and XML (AJAX)*

* Synchronous Request

1. Recommended for few small requests. JavaScript will not continue to execute, until the server response is ready. If the server is busy or slow, the application will hang or stop.

var xhr = new XMLHttpRequest();

xhr.open(“GET”, “somepage.txt”, false);

xhr.send(null);

if(xhr.status==200){

var str = xhr.responseText;

}

* Asynchronous Request

1. JavaScript does not have to wait for the server response. It can execute other scripts while waiting for server response, and deal with the response when the response is ready.

var xhr = new XMLHttpRequest();

xhr.open(“POST”, “products.php”, true); xhr.setRequestHeader(“Content-Type”, “application/x-www-form-urlencoded”);

xhr.onreadystatechange = function(){

if(this.readyState == 4 && this.status == 200){

var hdrDate = this.getResponseHeader(“Date”);

var xml = this.responseXML;

}

}

xhr.send(“productID=1234&catID=5678”);

* Cross-Browser and Old Browser Support

var xhr;

if(typeof XMLHttpRequest != “undefined”){ xhr = new XMLHttpRequest();

}else if (window.ActiveXObject){

var activeXIDs = [“MSXML2.XmlHttp.5.0”,

“MSXML2.XmlHttp.4.0”,

“MSXML2.XmlHttp.3.0”,

“MSXML2.XmlHttp”,

“Microsoft.XmlHttp”];

for(var i=0; i<activeXIDs.length;i++){

try{

xhr = new ActiveXObject(activeXIDs[i]);

}catch(e){}

}

}

* Three important properties of the XMLHttpRequest Object

o onreadystatechange

* Stores a function to be called automatically each time readyState changes

1. readyState
   * 0 : request not initialized
   * 1 : server connection established
   * 2 : request received
   * 3 : processing request
   * 4 : request finished and response is ready
2. status
   * 200 : OK
   * 404 : Not Found

*Document Object Model*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Document Object Collections | | o | referrer |
|  | o | anchors[] | o | title |
|  | o | forms[] | o | URL |
|  | o | images[] | Document Object Methods | |
|  | o | links[] | o | close() |
|  | Document Object Properties | | o | getElementById() |
|  | o | cookie | o | getElementsByName() |
|  | o | documentMode | o | getElementsByTagName() |
|  | o | domain | o | open() |
|  | o | lastModified | o | write() |
|  | o | readyState | o | writeln() |

*Java EE*

* Java EE is an open, standards-based development and deployment platform for creating distributed, transactional, reliable, secure, multi-tiered, web-based, server-centric, component-based enterprise applications
* Java EE Application Model

1. Java programming language, Java Virtual Machine (JVM) o Java EE Components
   * Java EE Clients
     + Application Clients, Applets (embedded in web clients)
   * Web Components
     + Servlets, JavaServer Pages (JSP), JavaServer Faces (JSF)
   * Enterprise JavaBeans (EJBM) o Java EE Containers
   * Client containers, web container, EJB container o Java EE Server

* Java EE Web Application

1. Collection of resources installed under a specific subset of the URL namespace of a web application server compliant with the Java

EE Specification (e.g. Apache‟s Tomcat, Apache‟s Geronimo, Sun Microsystems‟ Glassfish, IBM‟s WebSphere, etc.)

1. Resources
   * Static resources: web pages, images, stylesheets, etc. *(serves as is)*
   * Dynamic resources: servlets, JSPs
   * Miscellaneous resources: business object classes (e.g. Java Beans, EJB), support libraries, etc. o XML-formatted descriptor and configuration files
   * web.xml, application.xml, context.xml, etc.
   1. Organized into a standard hierarchical structure and typically packaged and deployed as WAR or EAR files

* Java EE APIs
  1. Enterprise JavaBeans Technology

1. Java Servlet Technology o Java Server Pages

o Java Server Pages Standard Tag Library o Java Server Faces

o Java Msg Service API o Java Transaction API o JavaMail API

o JavaBeans Activation Framework o Java API for XML Processing