*Hypertext Markup Language*

* Language used to markup documents (i.e. web pages) in the WWW
  + Structure (and content)
  + Presentation (mostly deprecated in favor of style sheets)
* Versions

o HTML 2.0, 3.2, 4.0

* + HTML 4.01
    - Strict – deprecated
    - Transitional – still make use the deprecated
    - Frameset – using frames
  + HTML 5 (working draft)
* Head
  + title, base, link, meta, style, script
* Body
  + Grouping elements (div, span) o Headings (h1 – h6)
  + Paragraphs, line breaks, horizontal rules (p , br , hr) o Lists (ul, ol, li, dl, dt, dd, dir, menu)
  + Tables (table, th, tr, td, thead, tfoot, tbody, colgroup, col) o Structured text
    - Phrase elements (em, strong, dfn, code, samp, kbd, var, cite, abbr, acronym)
    - Quotations (blockquote, q)
    - Subscripts and superscripts (sub, sup)
    - Preformatted texts (pre)
  + Font styles and alignments (tt, i, b, big, small, strike, s, u, font, basefont, center) o Document changes (ins, del)
  + Links and anchors (a)
  + Objects, images, applets (object, img, param, applet) o Scripts (script, noscript)
  + Miscellaneous (address, bdo) o Frames, noframes, iframe
  + Frameset (for frameset DTD)

*Extensible Hypertext Markup Language*

* Reformulation of HTML in XML
* Versions
  + XHTML 1.0
  + XHTML 1.1
  + XHTML 1.2
  + XHTML 2.0
  + XHTML 5 (working draft)
* Basic differences between HTML and XHTML
  + All elements have beginning and ending tags
  + All elements are nested properly
  + Elements and attribute names are case sensitive (lowercase)
  + Attribute values are quoted)
  + Attribute values cannot be minimized
* XHTML Doc. Components
  + Document Type Declaration (DOCTYPES)
    - HTML 4.01 DOCTYPES
    - XHTML 1.0 DOCTYPES
  + Elements
    - Tags
    - Content (elements)
    - Attributes
      * id, class, title, alt, lang, intrinsic event attributes (e.g. onClick)
    - Character Entity References
      * &lt; &gt; &amp; &nbsp; &shy;
* XHTML Elements
  + Block-level and inline (aka text-level) elements
    - Block-level elements represent “larger” document structures and may contain inline & other block level elements
    - Inline elements contain only data and other inline elements
* Author Styles
  + External style sheets (recommended)
  + Embedded styles
  + Inline styles
* User Style
* User Agent Style (example default style sheet for HTML 4)

*Quirks*

* Older ways of rendering documents

*Standards*

* *Following the specification by W3C for HTML, CSS Cascading Style Sheets*
* Language used to specify the presentation aspects (i.e. layout and formatting of structurally marked up (e.g. HTML, XML, XHTML, SVG, etc.) document).
* Developed by *Hakon Wium Lie* (HTML Cascading Style Sheets / CHSS) and *Bert Bos* (Stream-based Style Sheet Proposal / SSP)
* Versions:
  + CSS 1, CSS 2, CSS 2.1, CSS 3 (structural semantic markup / modularizes CSS 2.1)
* CSS Rules
  + Selectors
    - Universal selector

\*{

}

* + - Type

p{

}

* + - Grouping

h1,h2,h3{

}

* + - ID

#nav{

}

* + - Class

.hide{

}

* + - Descendant

div.content em{

}

* + - Child

div.content > em{

}

* + - Adjacent sibling

h1 + p{

}

* + - Attribute acronym

div[class=note] {

}

* + - Pseudo classes
      * First child pseudo class (:first-child)

li:first-child{

}

* + - * Link pseudo class (:link, :visited)

a:link, a:visited{

}

* + - * Dynamic pseudo class (:hover, :active, :focus)

ul.menu > li{

}

ul.menu > li:hover{

}

* Declaration Block
  + Properties
    - Visual, aural, paged, interactive
    - Margins, borders, and padding
    - Colors and background
    - Fonts
    - Text formatting
    - Lists and tables
    - Positioning and dimensions
    - Miscellaneous
  + Values
    - Keywords
    - Numbers
    - Length

o Length units: em ex px in an mm pt pc

relative absolute

* Percentages
* Colors
* Strings
* URLs
* Counters
* Initial values, inherited values, !important rules
* Specified, computed, used, and actual values
  + By origin and importance
    - User important declarations
    - Author important declarations
    - Author normal declarations
  + By specificity
    - Inline styles (1) or not (0)
    - Number of ID attributes in selector
    - Number of other attributes and pseudo-classes in selector
    - Number of element names and pseudo-elements in selector
  + By order
* CSS comments /\* \*/

*Client-side web scripting*

* Allows programs (i.e. scripts) to be downloaded from a web server and executed in the client environment (e.g. browser)
* Common client-side scripting technologies o JavaScript, ECMAScript, Jscript o VBScript, ActionScript
  + Java Applets, ActiveX Controls, Flash Animations, Microsoft Silverlight, Adobe Integrated Runtime (Adobe AIR)
* Common uses:
  + Dynamic (X)HTML
    - Page embellishments, visual “effects”, content generation and manipulation, user interaction, document and page navigation
  + Client-side form data validation
  + Asynchronous content retrieval, RIA Technologies (e.g. AJAX).
* Common issues:
  + Browser support
    - No scripting support
    - Scripting disabled
    - Plugin availability
  + Version incompatibilities, non-standard implementations
  + Capabilities restrictions
    - Scripting languages are not for general-purpose programming
    - E.g. javascript is restricted by the sandbox execution model and the same origin policy
  + Security risks
    - Browser implementation defects (e.g. buffer overflows)
    - E.g. javascript, cross-site scripting (XSS) or cross-site request forgery (XSRF) issues
    - Malicious ActiveX controls

*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:
  + 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 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:
  + 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) o Booleans (true, false)
        + Strings (Single or Double quote delimited) o Undefined and Null
      * Composite (object) types
        + Core JavaScript Objects

Object, Number, Boolean, String, Date, Math, Global, RegExp, Error

Arrays (Array)

Functions (Function, Arguments)

* + - Keywords
* DOM Objects
  + Anchor, Applet, Attr, Comment, DOMException, DOMImplementation, DocumentFragment, Element, Event, Form, Image, Input, Layer, Link, Node, Option, Select, Style, Text, TextArea
    - 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 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
    - 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
    - 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
  + 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
  + 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
  + onreadystatechange
    - Stores a function to be called automatically each time readyState changes
  + readyState
    - 0 : request not initialized
    - 1 : server connection established
    - 2 : request received
    - 3 : processing request
    - 4 : request finished and response is ready
  + status

 200 : OK

* + - 404 : Not Found

*Document Object Model*

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