Internet Programming

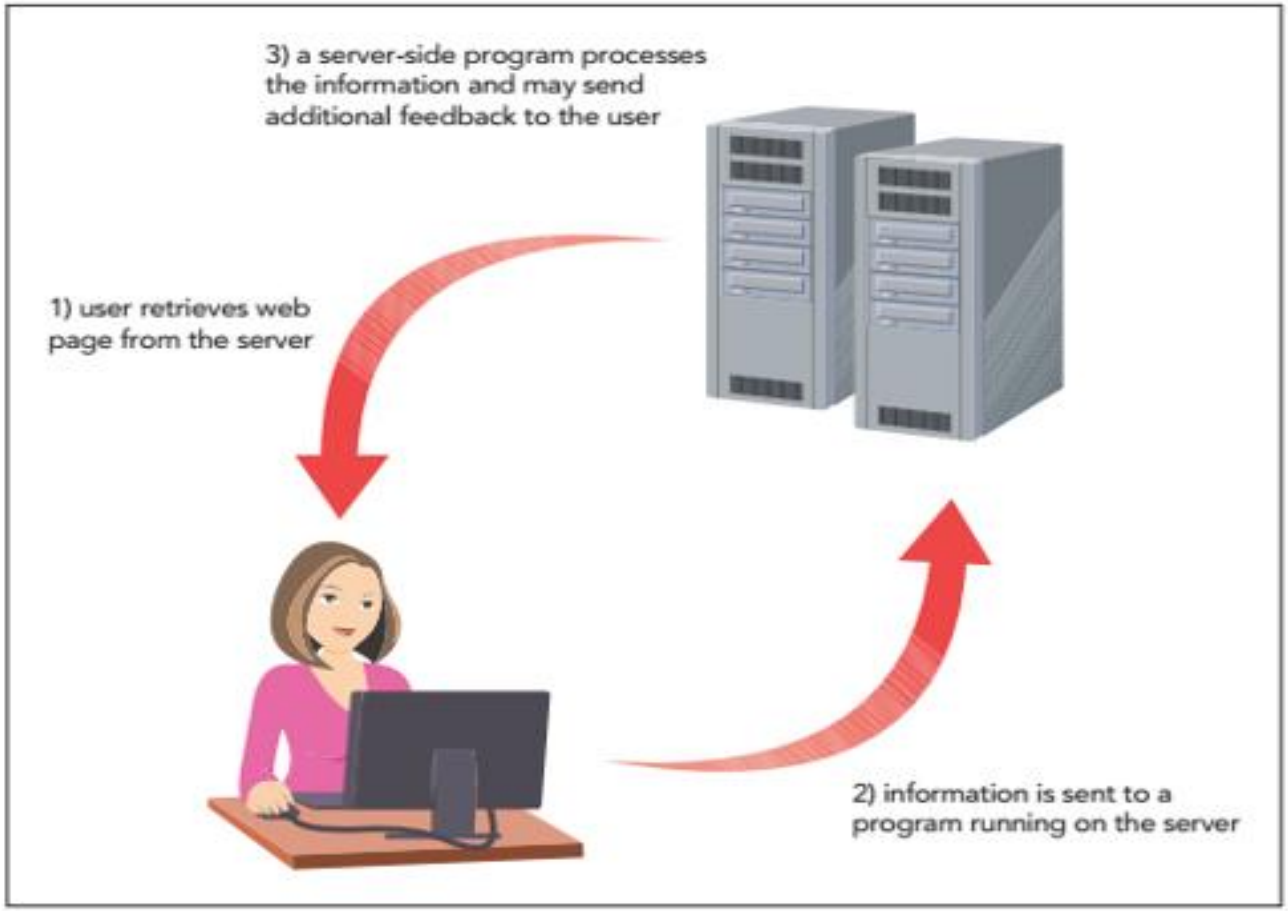
JavaScript Introduction

Dan Brandon, Ph.D., PMP

Server Side Programming

- **Server-side programming**: Program code runs from the server hosting the website
- Advantage
  - Connects a server to an online database containing information not directly accessible to end users
- Disadvantages
  - Use server resources and requires Internet access
  - Long delays in cases of system over-load

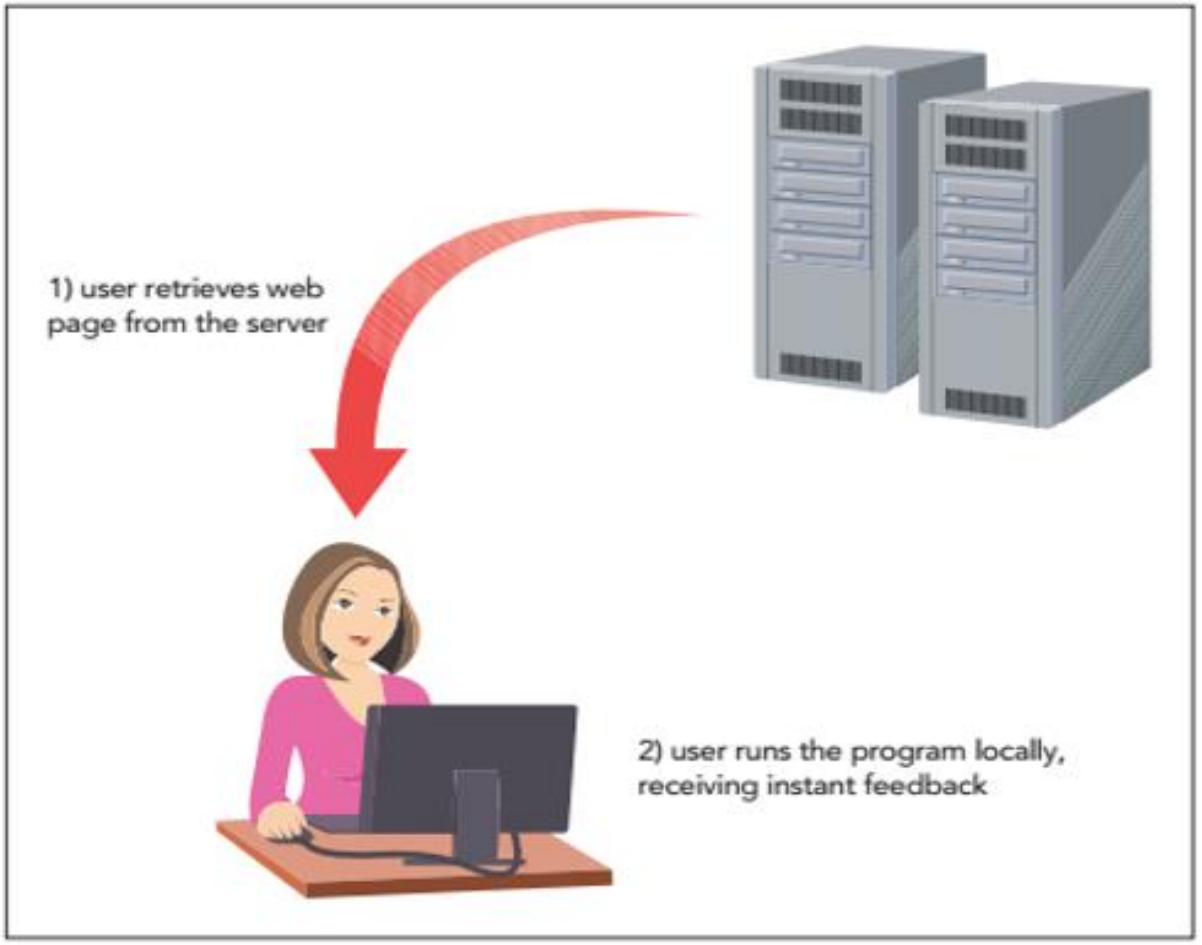
Server Side Programming (con't)



Client Side Programming

- **Client-side programming**: Programs run on the user's computer using downloaded scripts with HTML and CSS files
- Distributes load to avoid overloading of program-related requests
- Client-side programs can never replace server-side programming

Client Side Programming (con't)



JavaScript vs Java

- JavaScript
  - Limited capabilities
  - **Originally client side (only works within Browsers), now also a server version**
  - Interpreted → no compiler
- Java
  - Full object oriented language
  - Applications, Applets (inside browsers), Servlets
  - Compiled via Java Virtual Machines
  - Applets downloaded separately (not part of HTML document)

HTML/CSS/JavaScript

HTML
Structure

CSS
Appearance

JavaScript
Action

JavaScript Uses

- Generating or modifying HTML and/or CSS dynamically (dynamic HTML)
- Validating fields in an HTML form
- Reporting usage statistics
- Obtaining local user input
- Allowing the user to make choices to invoke various actions
- Showing properly windowed messages and warnings
- Local form processing and other client processing

JavaScript (JS) Tags

- JS start with:
  - <script language="JavaScript">
- And end with:
  - </script>
- In between beginning and ending tag is the script code, which is case sensitive
- This is an embedded script, as opposed to an external scrip discussed later
  - The statements of the script code are very much like C/C++/Java, and a semicolon is typically used to terminate each statement (and declaration), although the semicolon is not required

JavaScript (JS) Tags (con't)

- Embedded JS can be place anywhere in an HTML document, but normally in HEAD section
- C/Java "like" syntax
- Interpreter type execution (no compiler)
  - There are now compiled JavaScript languages available
- Even though Microsoft calls their version of ECMAScript "Jscript", you still use "JavaScript" for the language parameter
- Most browsers have "JavaScript" as the default scripting language, but best to specify it; IE/Edge also supports VBScript

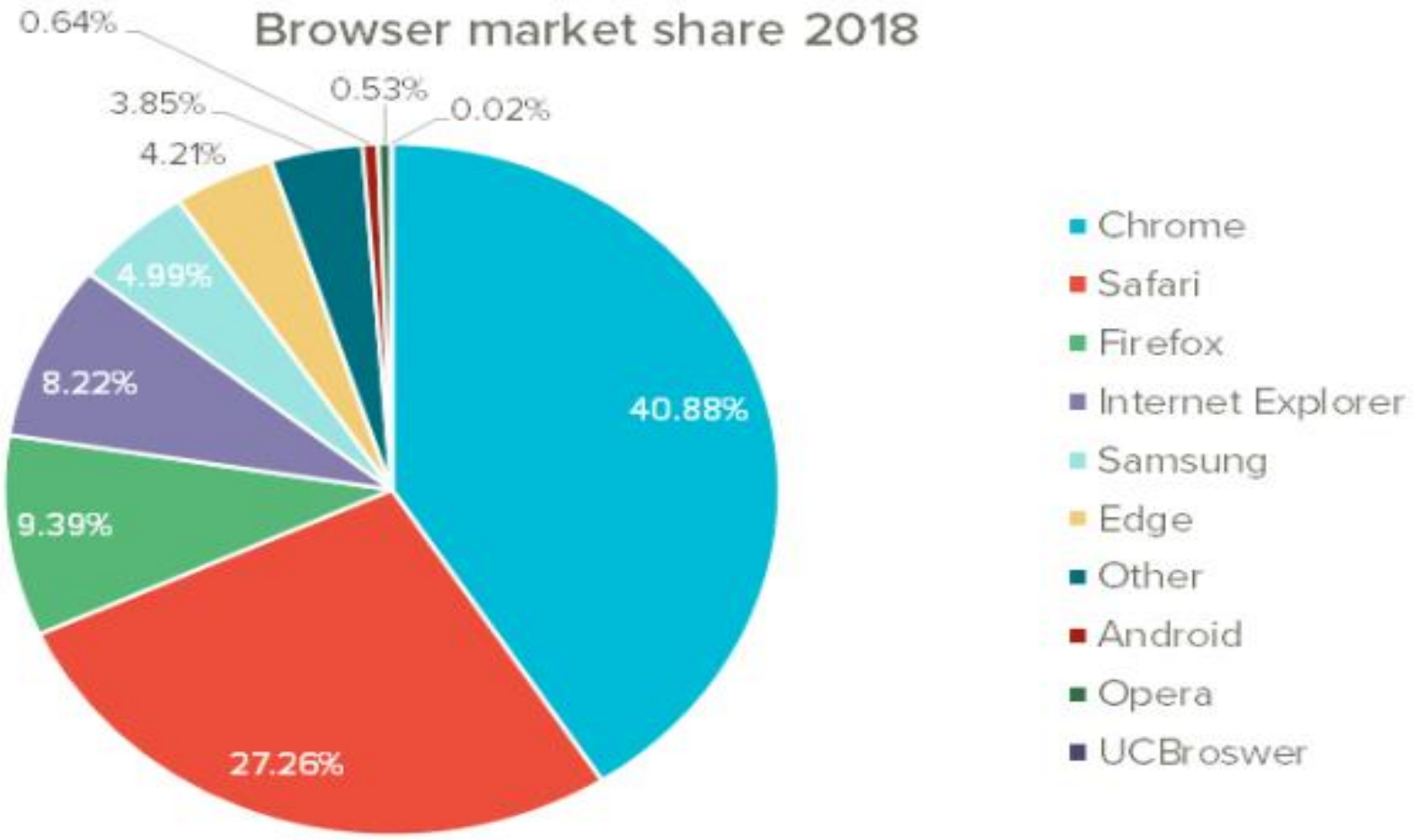
JavaScript (JS) Tags (con't)

- When a browser encounters a script, it immediately stops loading the page and begins loading and then processing the script commands
- The `async` and `defer` attributes can be added to `script` element to modify its sequence of processing
  - The `async` attribute tells a browser to parse the HTML and JavaScript code together
  - The `defer` attribute defers script processing until after the page has been completely parsed and loaded
  - The `async` and `defer` attributes are ignored for embedded scripts

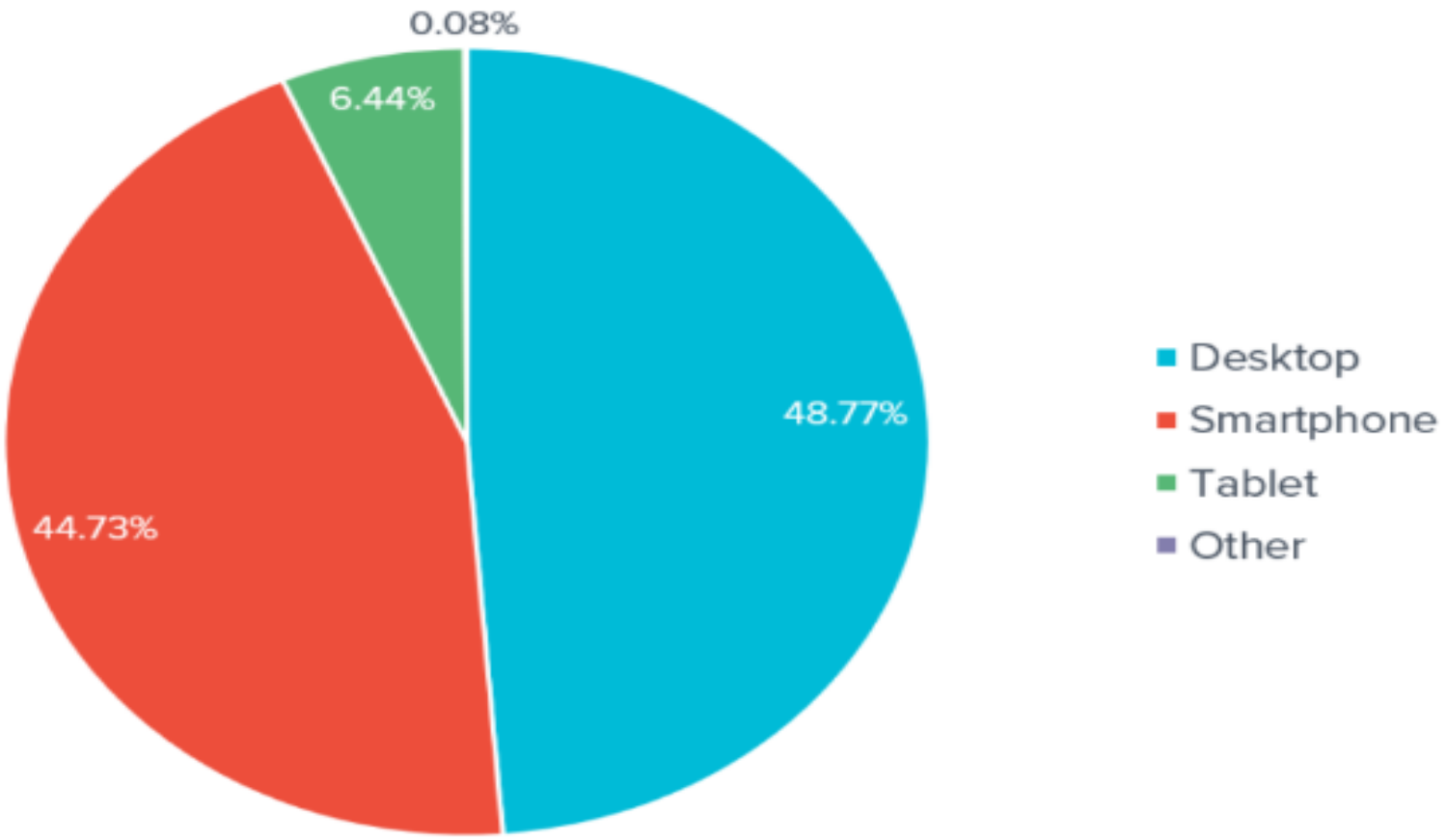
Browsers not Supporting JS

- To avoid the problems with very old browsers that do not support JS, you can put the JS code within comments:
  - <SCRIPT LANGUAGE="JavaScript">
  - <!--
  - ...script...
  - // -->
  - </SCRIPT>
- There are some incompatibilities with differing browsers and versions thereof

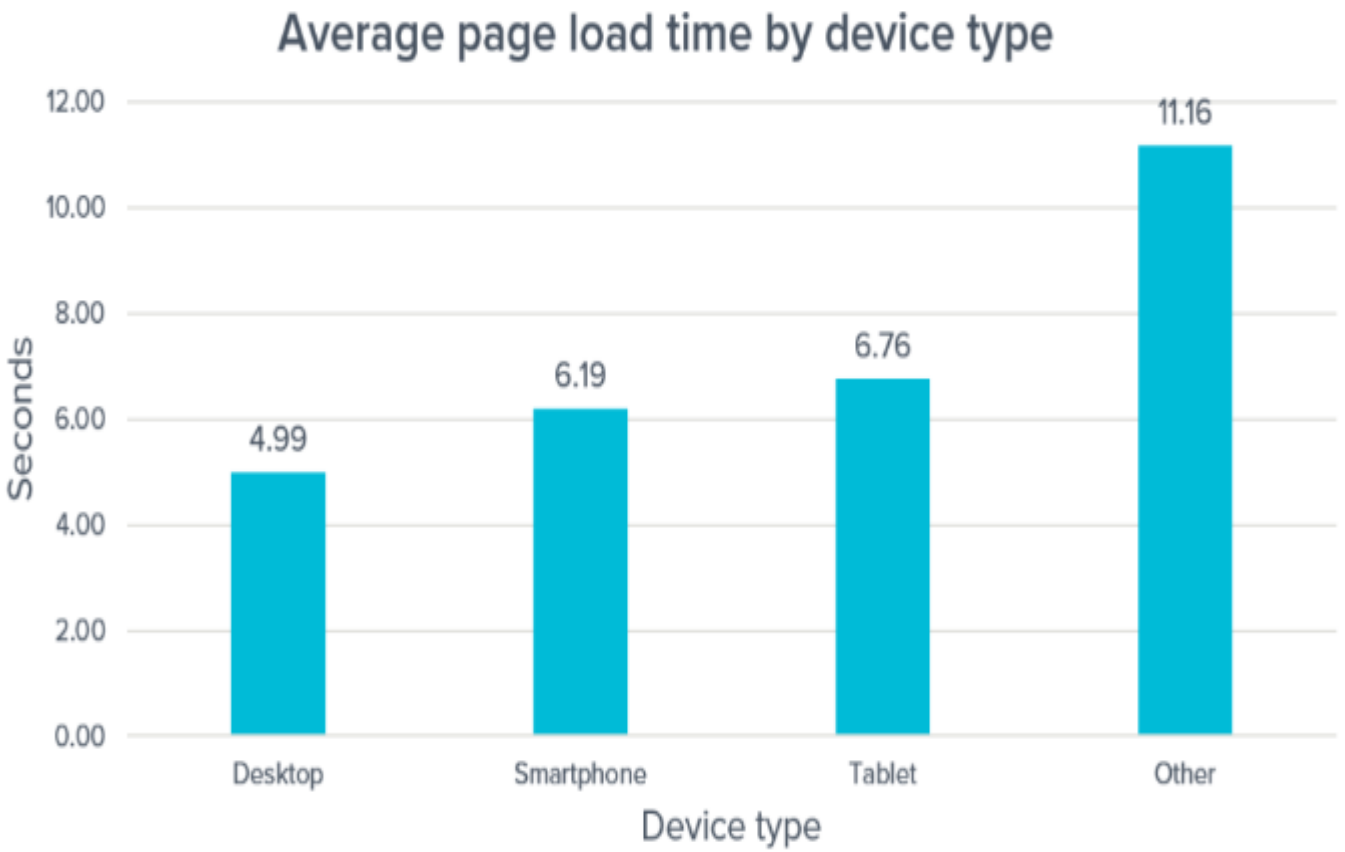
Browser Market



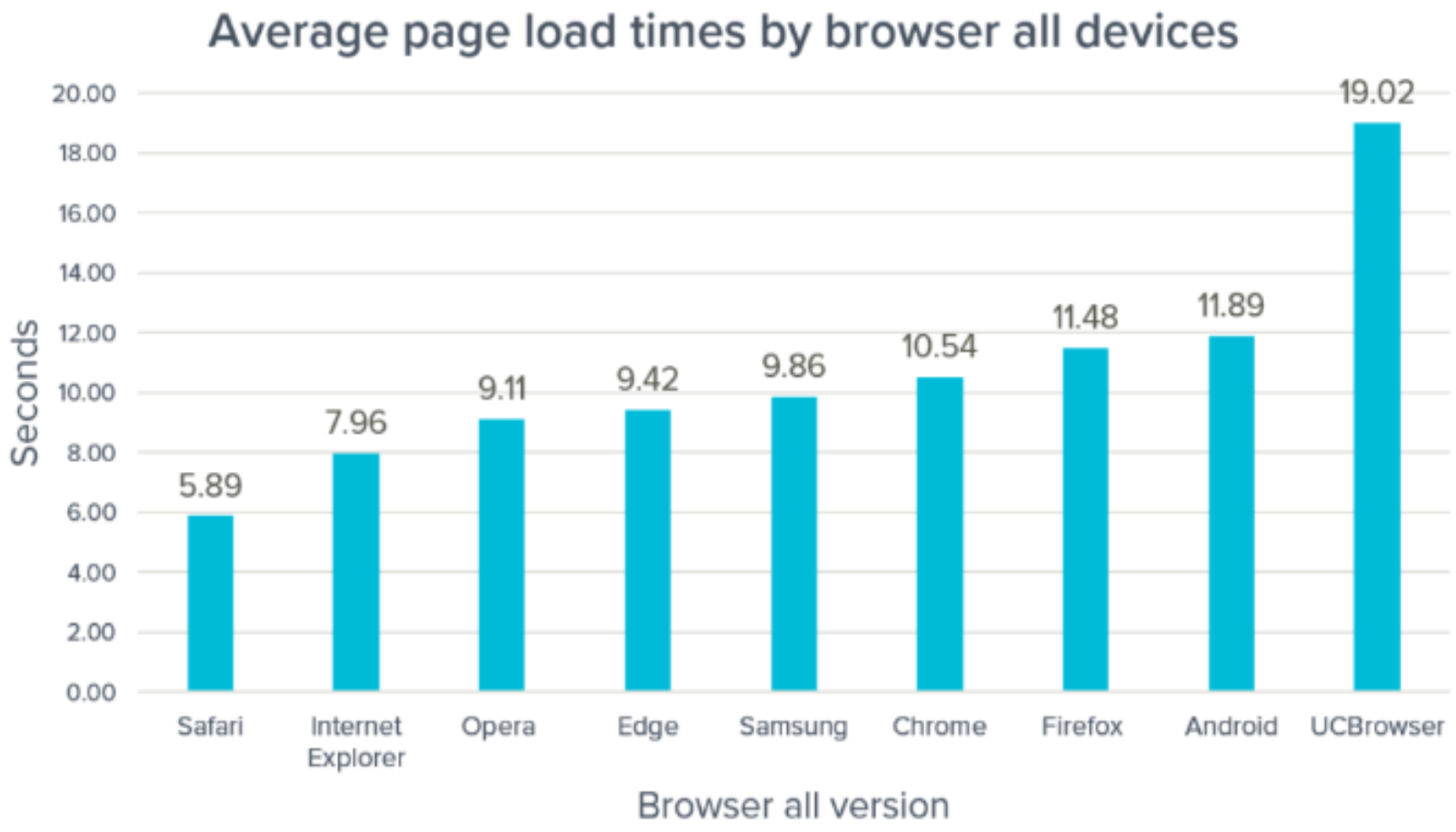
Web Device Usage



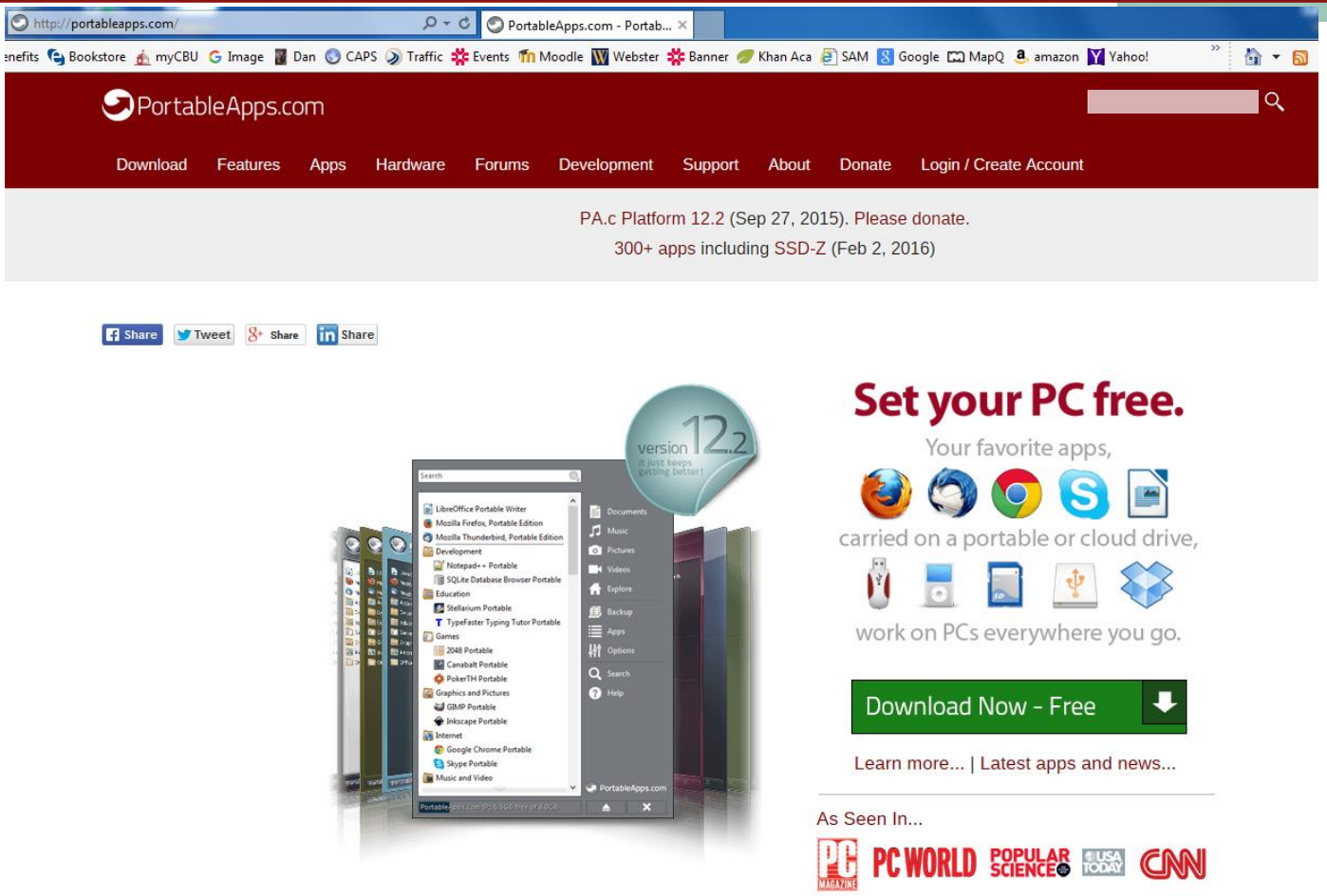
Page Load Times by Device



Page Load by Browser



Testing in Multiple Browsers



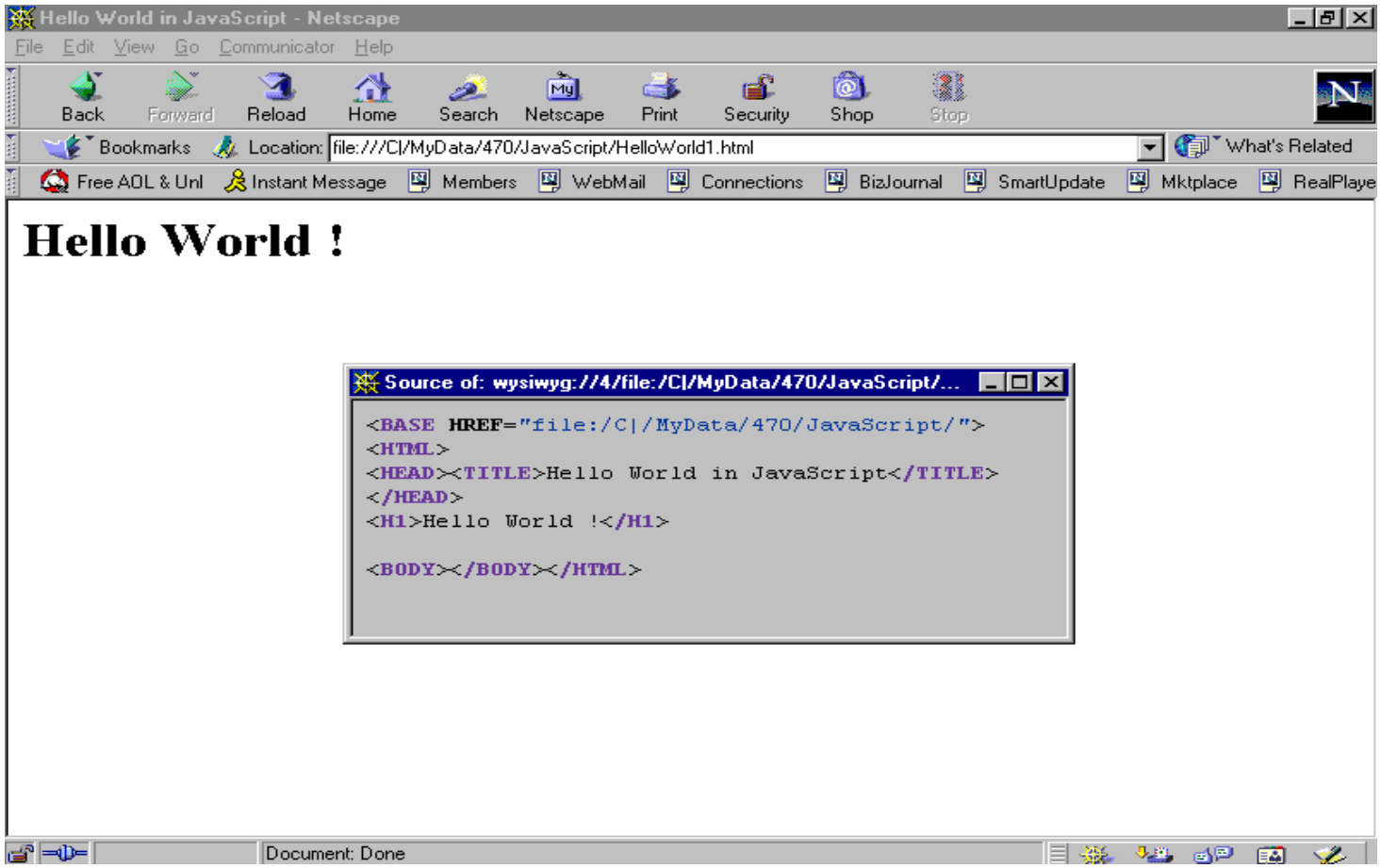
Hello World in JavaScript [HelloWorld1.html]

[writeln is a function of the document object discussed later;
and although "document" is optional in some browsers, it is required in some]

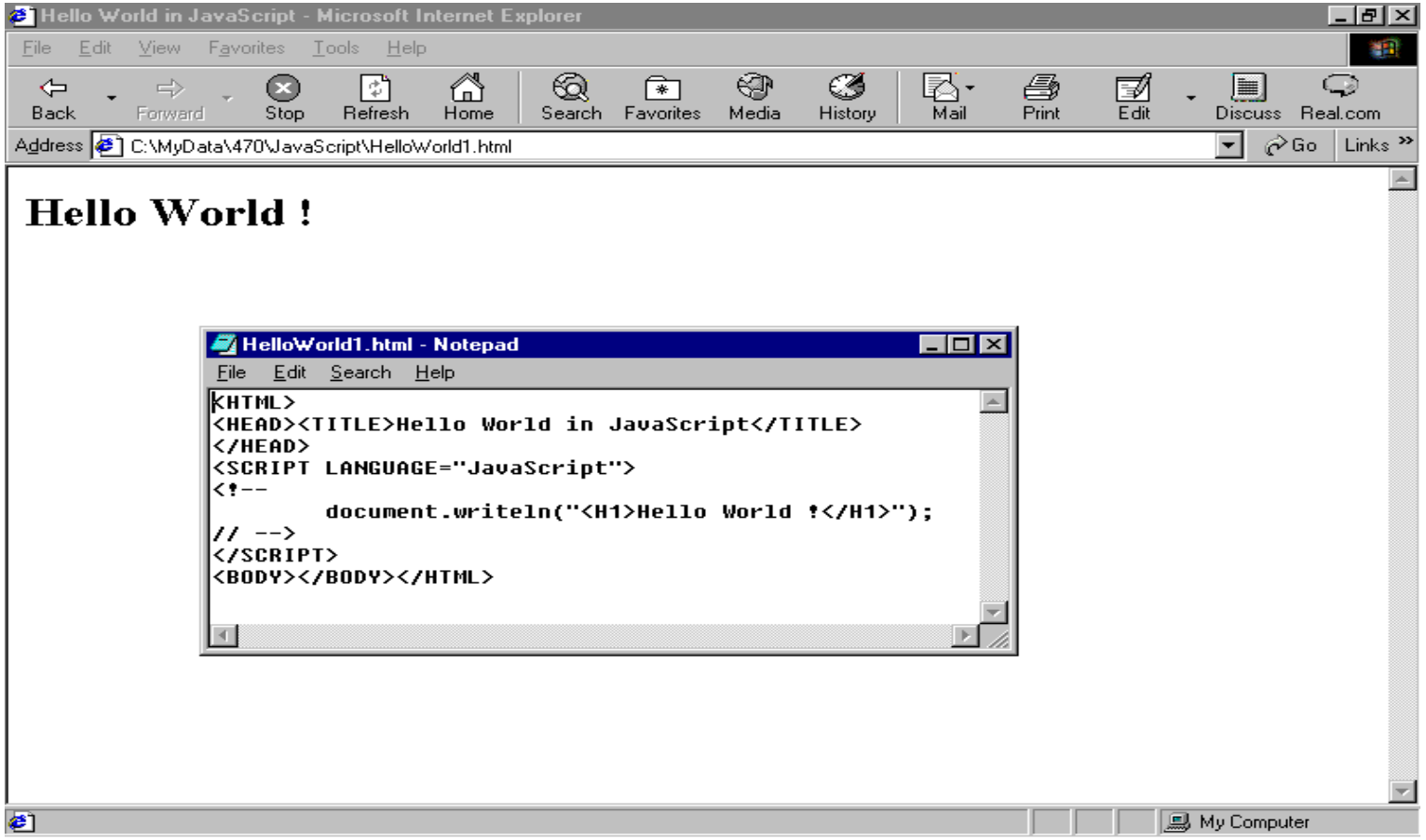
```
<HTML>
<HEAD><TITLE>Hello World in JavaScript</TITLE>
</HEAD>
<SCRIPT LANGUAGE="JavaScript">
<!--
      document.writeln("<H1>Hello World !</H1>");
// -->
</SCRIPT>
<BODY></BODY></HTML>
```

Will write "Hello World" in web page, try it out !
Note that it writes HTML code in the "document" !

Older Browser's "view source" shows dynamically written HTML



Newer Browser “view source” shows original HTML/Javascript

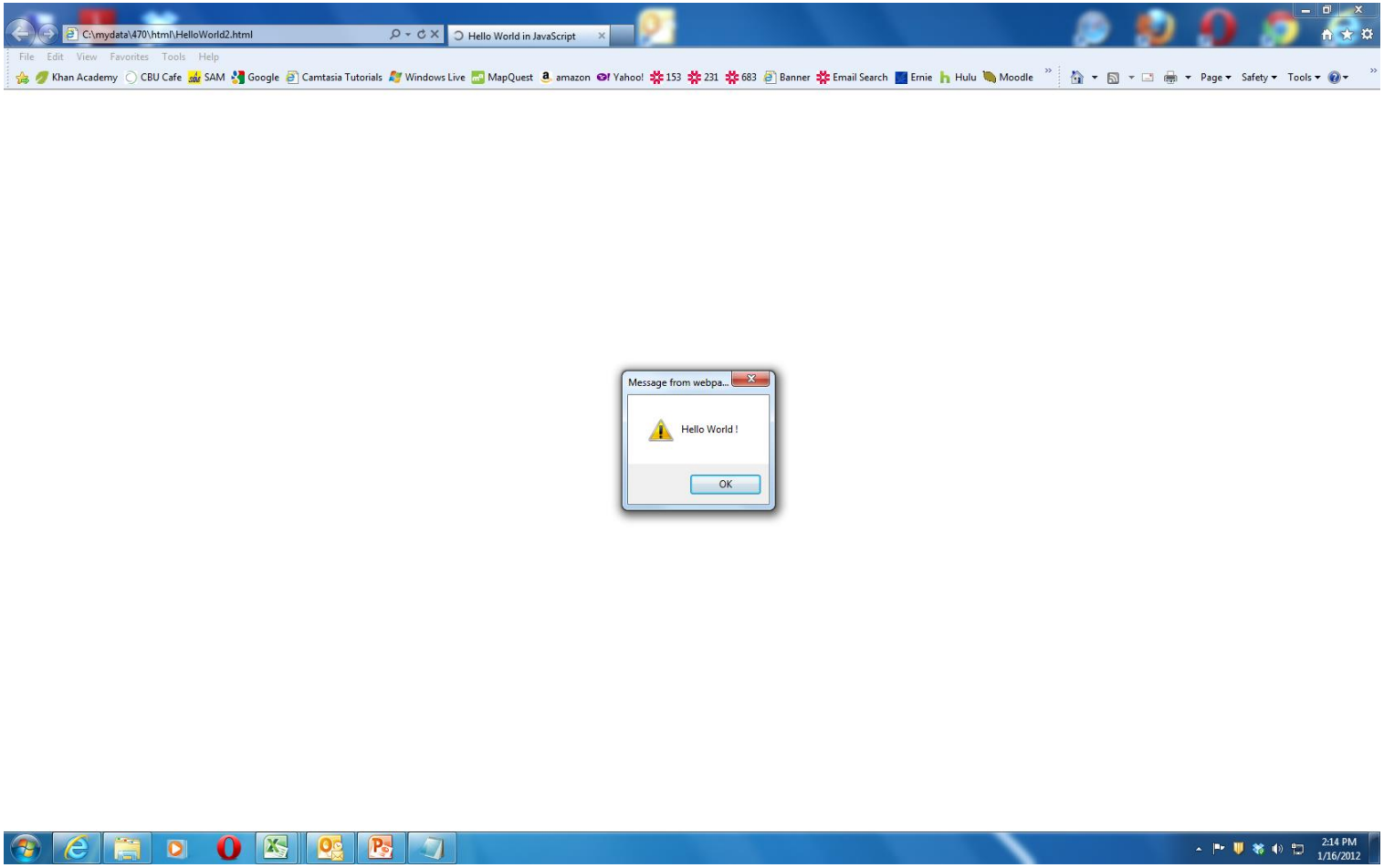


Using Alert Function [HelloWorld2.html]
[alert is a function of the window object]

- <HTML>
- <HEAD><TITLE>Hello World in JavaScript</TITLE>
- </HEAD>
- <SCRIPT LANGUAGE="JavaScript">
- <!--
- window.alert("Hello World !");
- // -->
- </SCRIPT>
- <BODY></BODY></HTML>

Plain text is written in the alert window ! Can also use console.log(), but requires “developer tools” to be open in most browsers.

Alert (con't)


C:\mydata\470\html\HelloWorld2.html
Hello World in JavaScript
File Edit View Favorites Tools Help
Khan Academy CBU Cafe SAM Google Camtasia Tutorials Windows Live MapQuest amazon Yahoo! 153 231 683 Banner Email Search Ernie Hulu Moodle
Page Safety Tools
Message from webpa...
Hello World !
OK
2:14 PM
1/16/2012

Errors

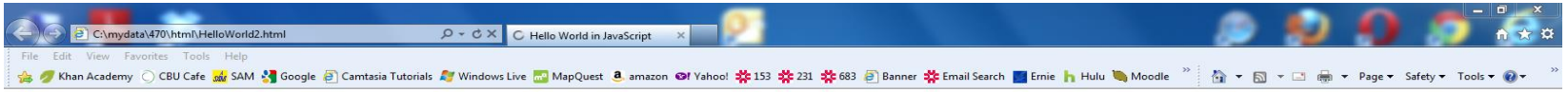
- JavaScript is case sensitive
- Extra white space between commands is ignored
- Line breaks placed within the name of a JavaScript command or a quoted text string cause an error
- Types of errors:
  - **Load-time errors** – occur when a script is first loaded by a browser
  - **Run-time errors** – occur during execution of a script without syntax errors
  - **Logical errors** – are free from syntax and executable mistakes but result in an incorrect output

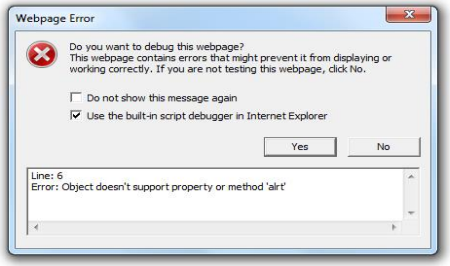
Errors (con't)

- Errors in your JS may be shown in a dialog box when the JS is loaded
  - May need to enable script debugger (see next slide)
  - In your Hello World example, misspell alert, and then open the html file to see what happens
  - Next, leave off the final SCRIPT tag and see what happens

Try these out !

Error's (con't)


C:\mydata\470\html\HelloWorld2.html
Hello World in JavaScript
File Edit View Favorites Tools Help
Khan Academy
CBU Cafe
SAM
Google
Camtasia Tutorials
Windows Live
MapQuest
amazon
Yahoo!
153
231
683
Banner
Email Search
Ernie
Hulu
Moodle
Page
Safety
Tools

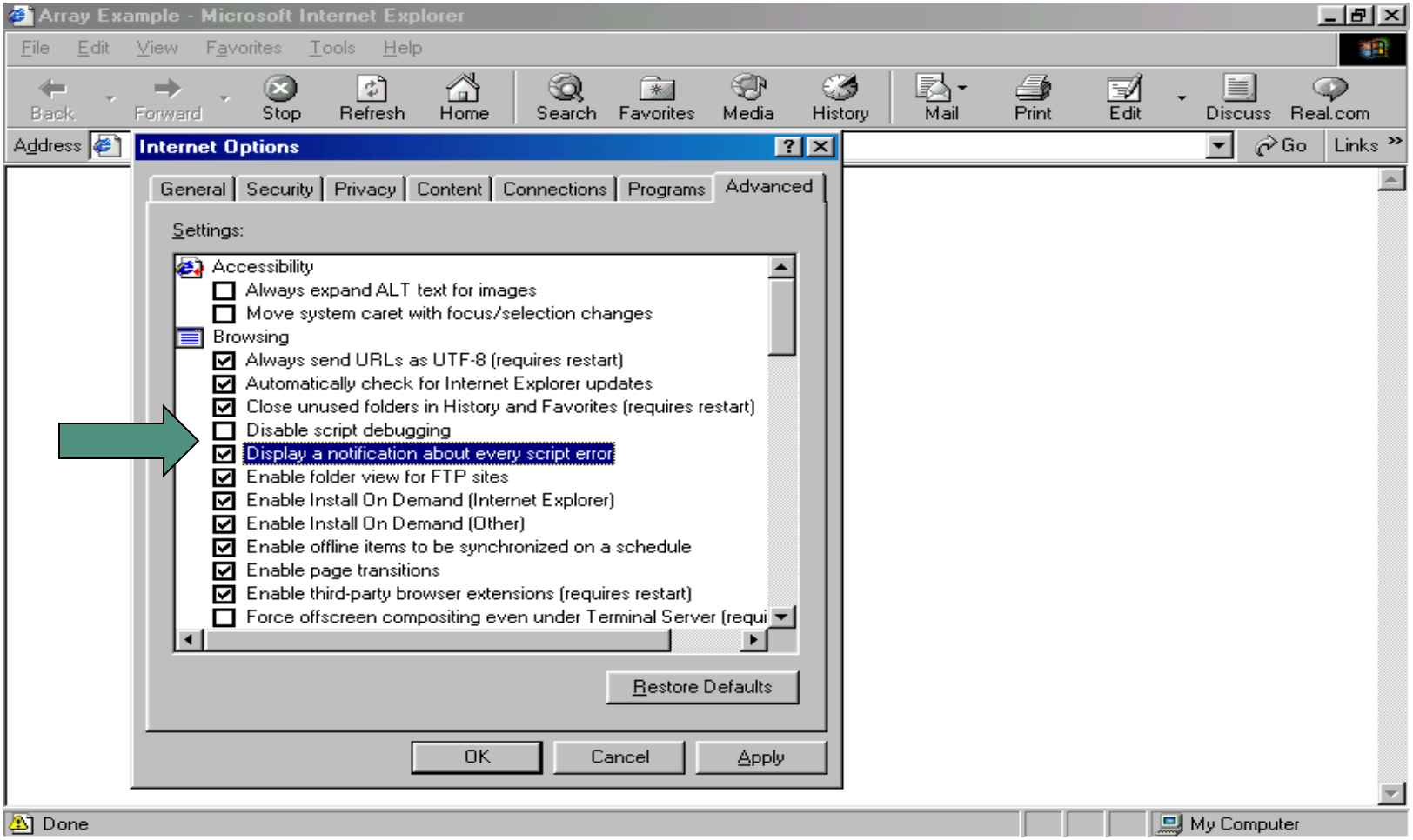

Webpage Error
Do you want to debug this webpage?
This webpage contains errors that might prevent it from displaying or working correctly. If you are not testing this webpage, click No.
Do not show this message again
Use the built-in script debugger in Internet Explorer
Yes
No
Line: 6
Error: Object doesn't support property or method 'alrt'


2:17 PM
1/16/2012

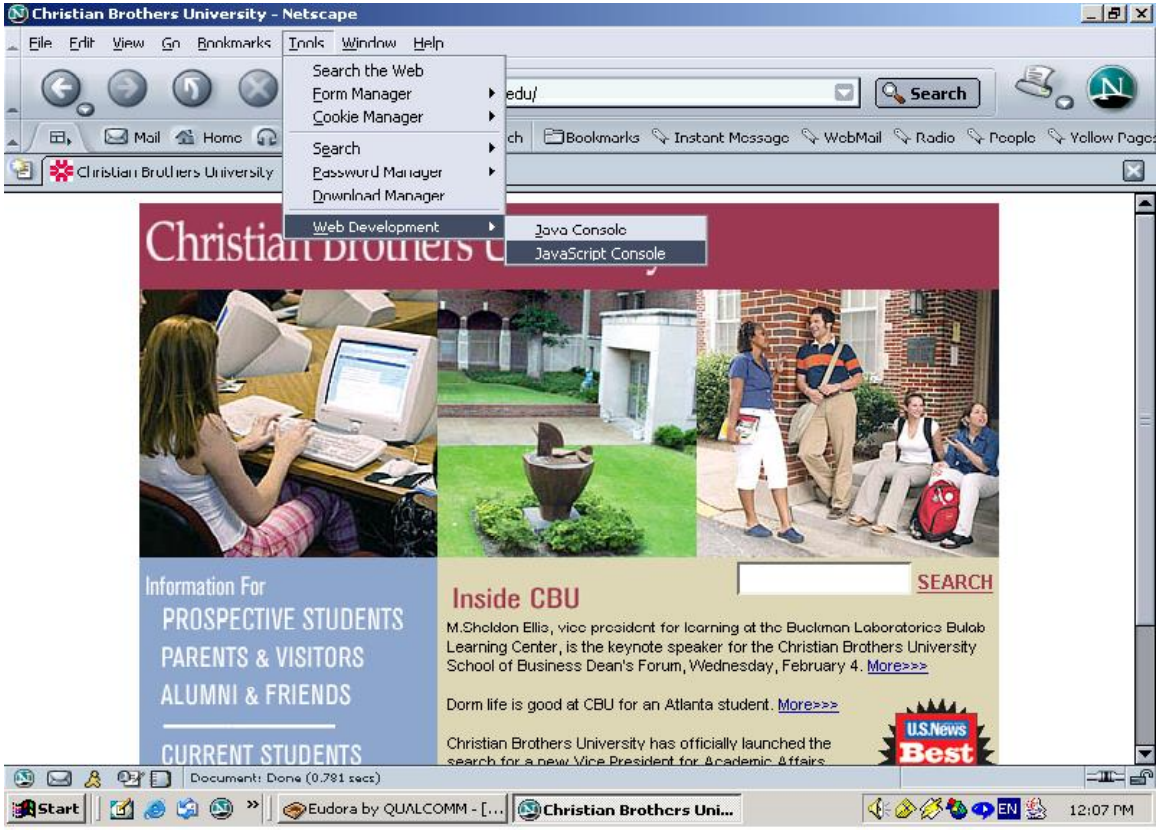
Error's (con't)

- Not all errors will be reported, and those that are reported may be reported on the wrong line or more errors will be reported that occurred
- Your browser may have a JavaScript debugger enabled – try the above scenarios in other browsers !
- In some browsers, there is a shortcut to open a debugging tool (F12)
- Also in some browsers, the tools can also be opened by selecting Developer Tools from the browser menu

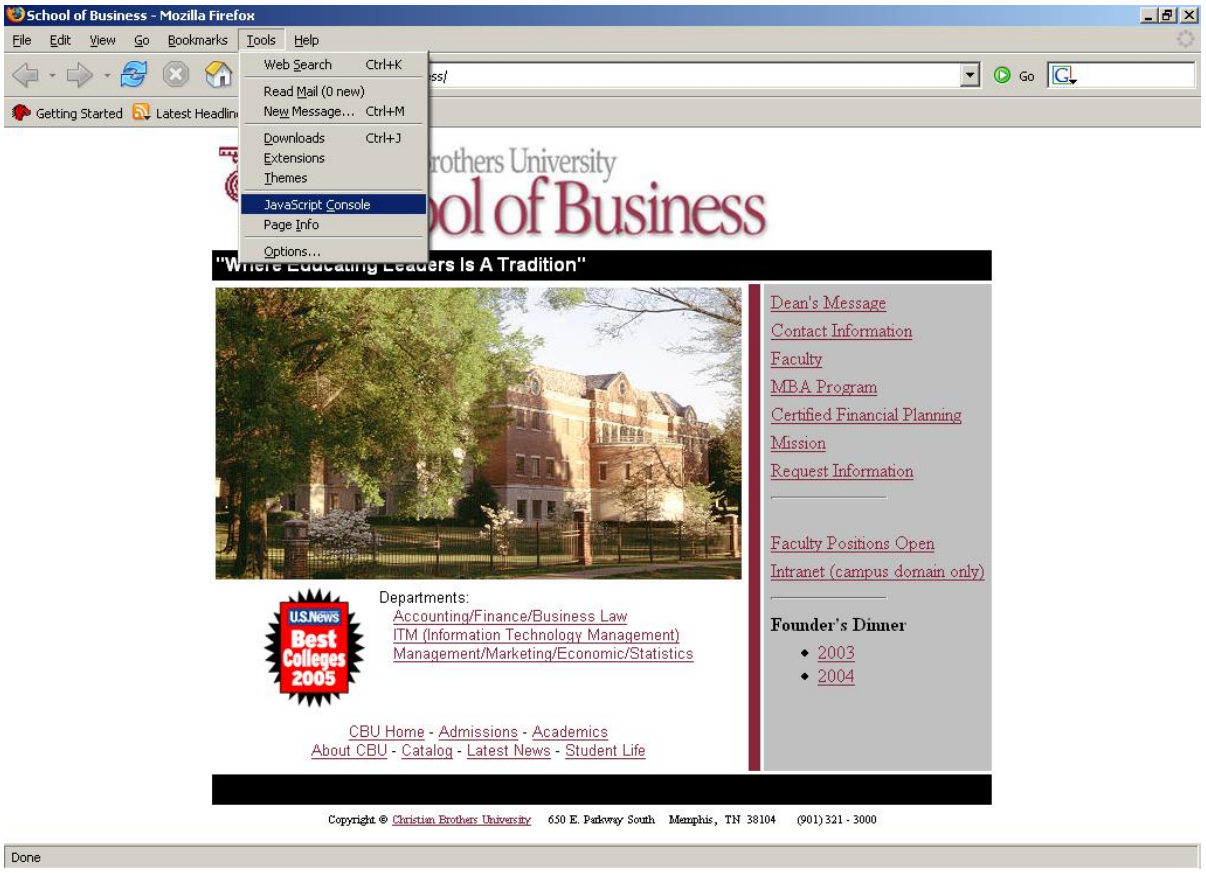
Setting Error Reporting in IE



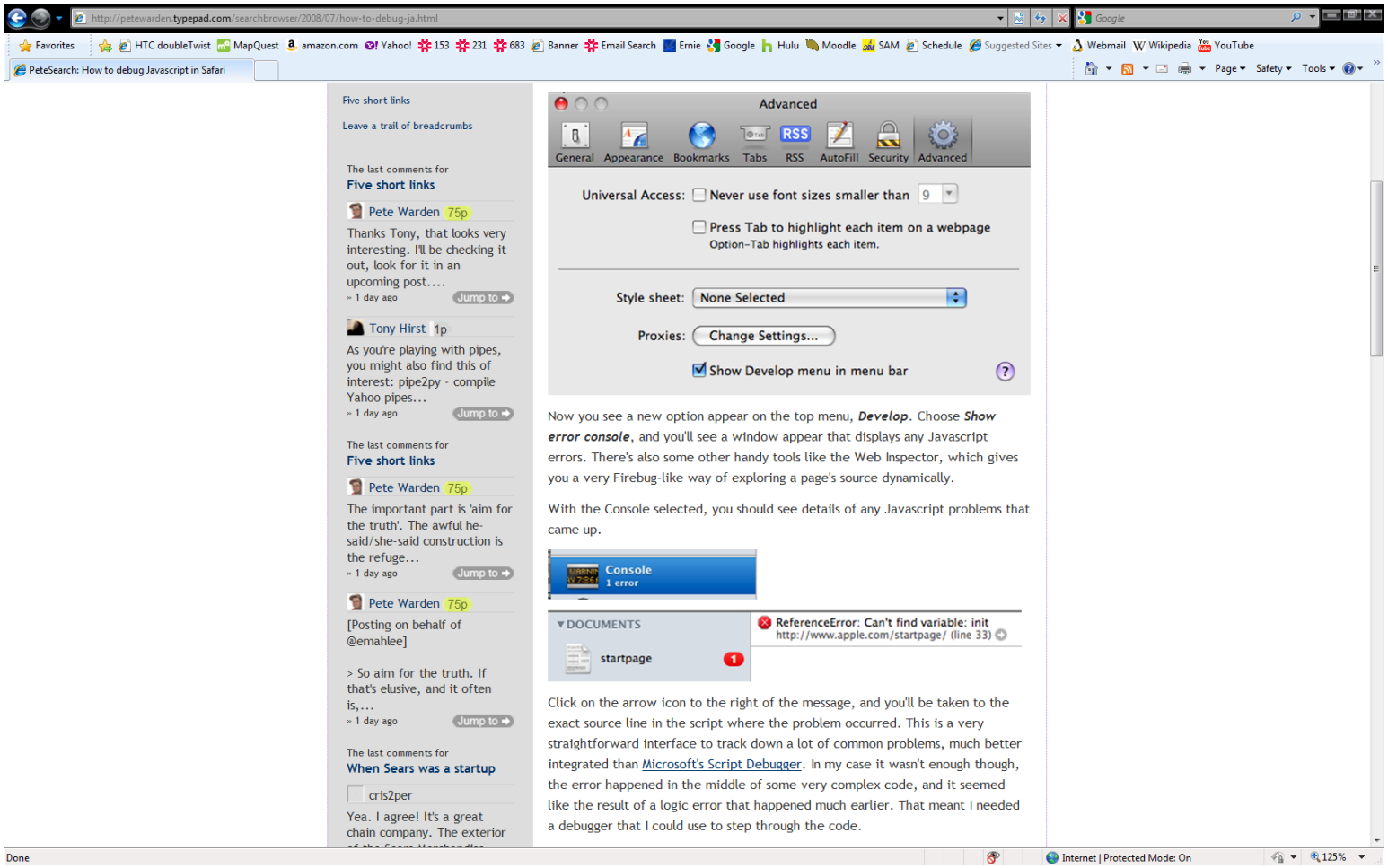
Setting Error Reporting in Nav



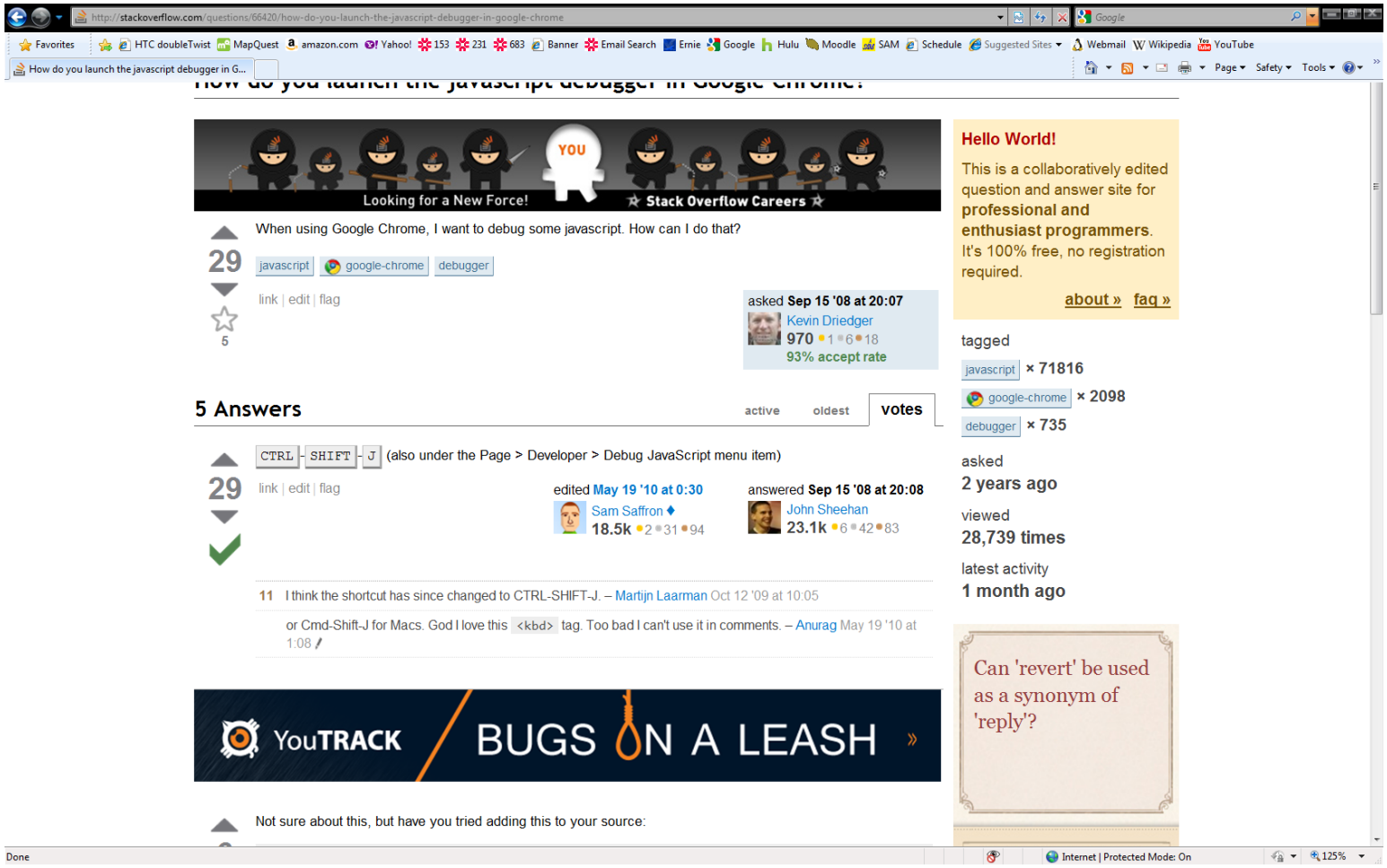
JavaScript Console in FireFox



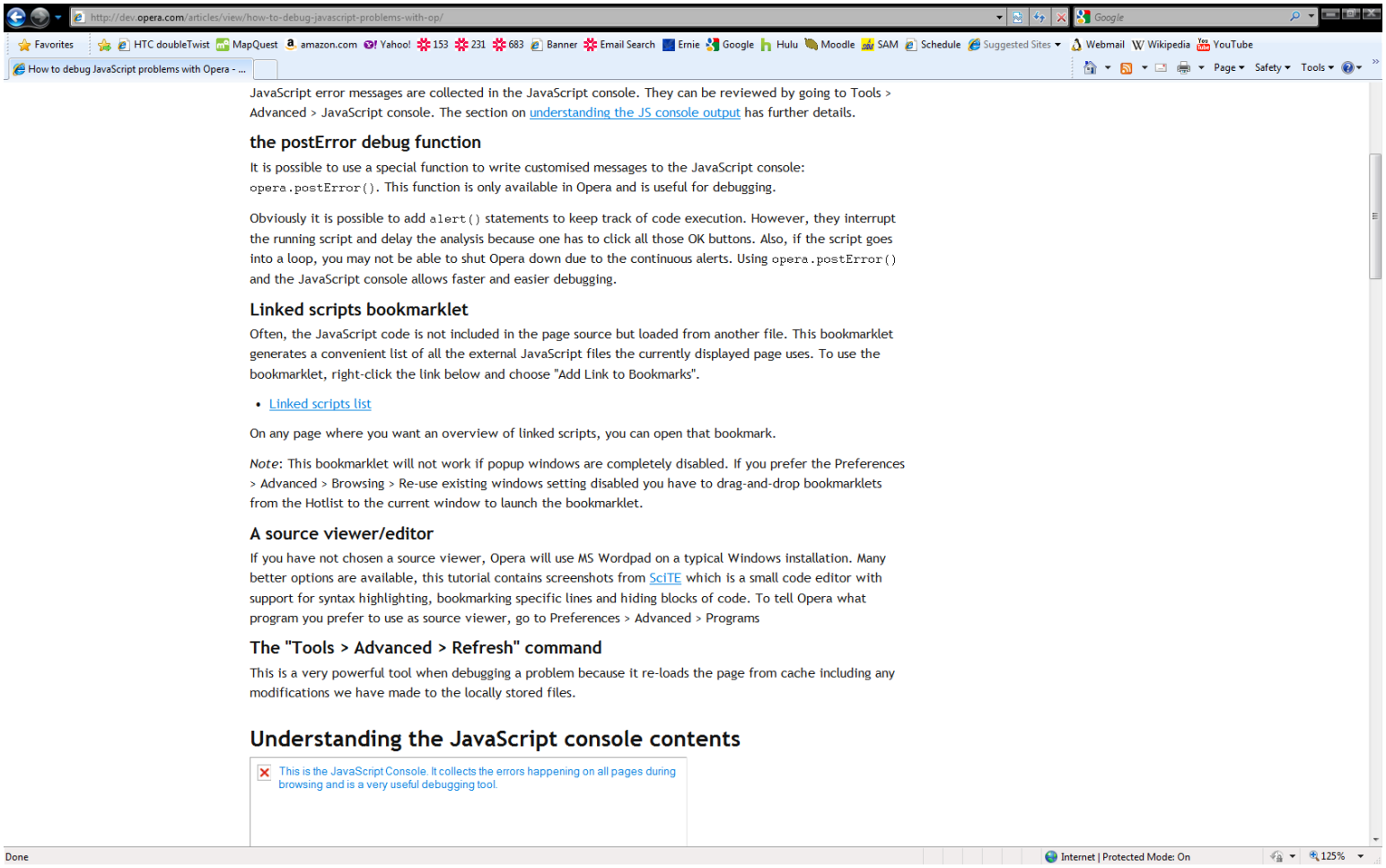
JS Debugging in Safari



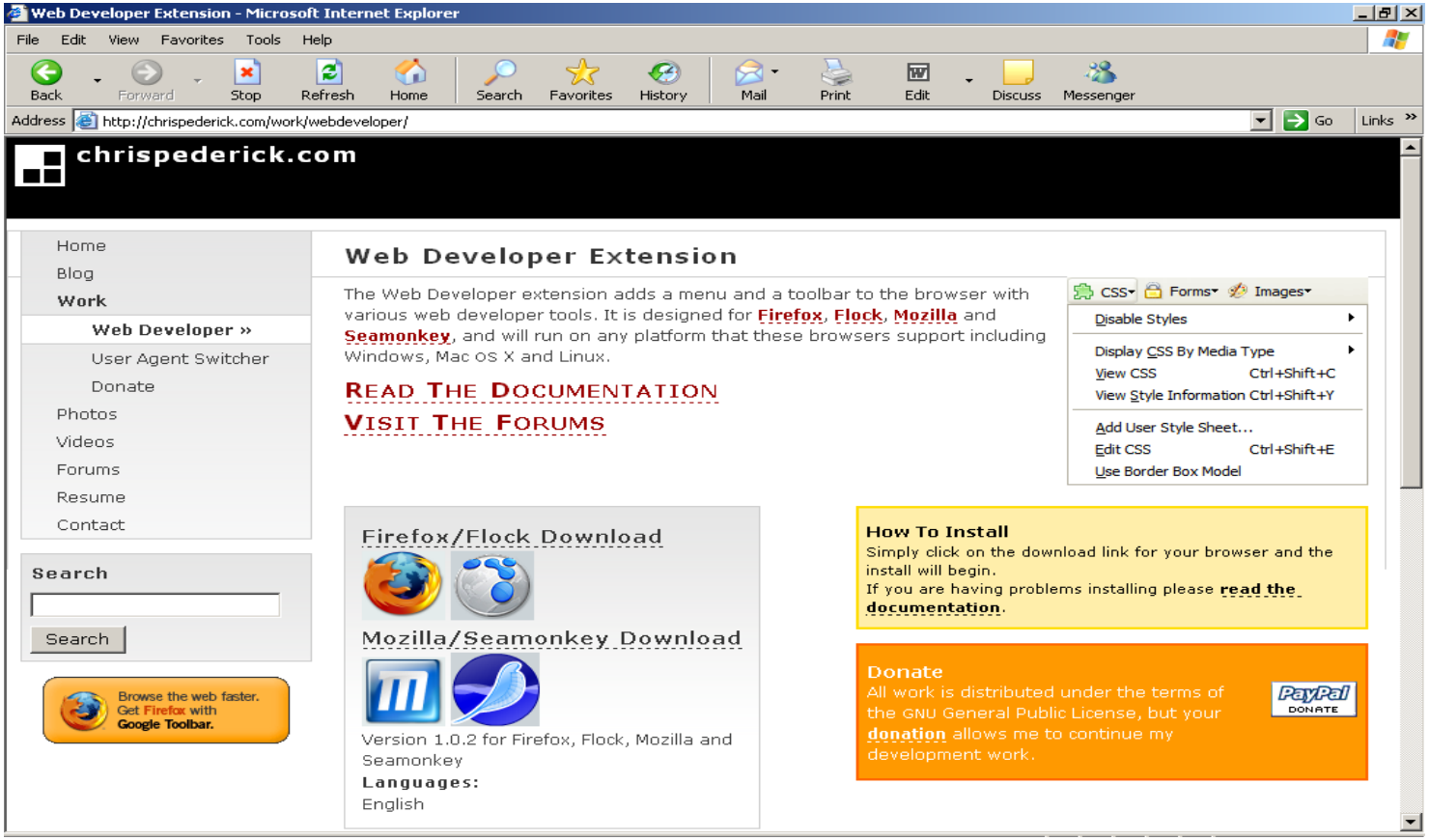
JS Debugging in Chrome



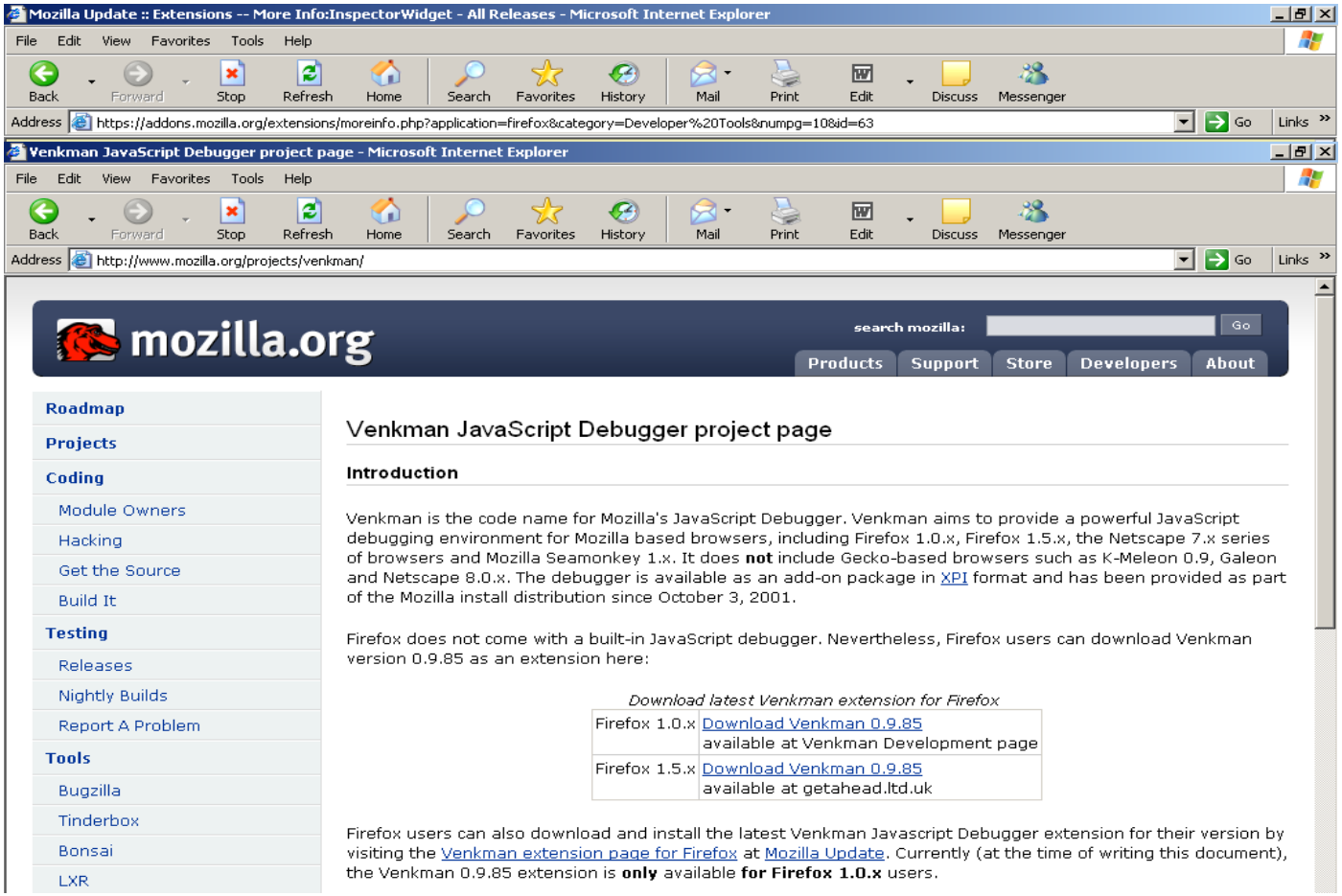
JS Debugging in Opera



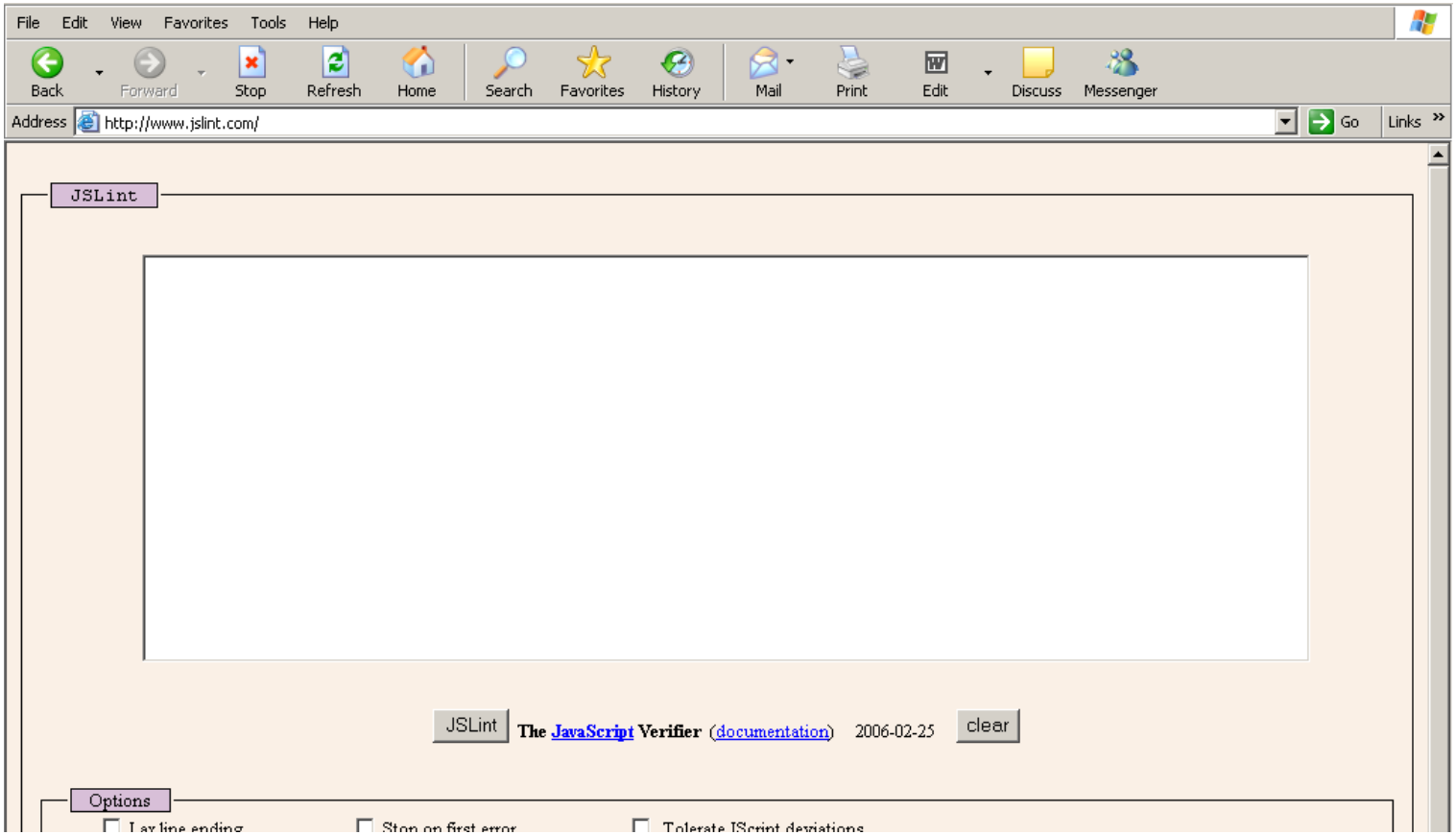
Add On Debuggers



Mozilla JavaScript Debugger



Stand Alone JS Debugger

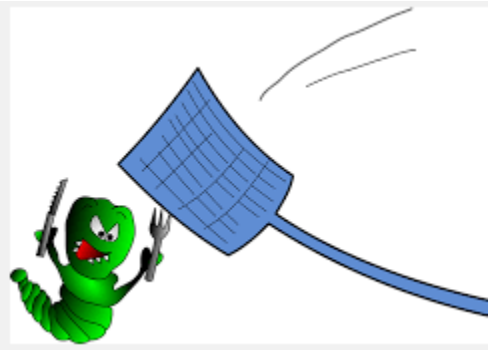


Debugging !

- The instructor should not help you debug your programs !
  - Students should also not help other students with specific bug identification and correction
- A major part of the learning experience is for you to find and correct your own errors !
- Go thru online lesson about algorithms and programming

Debugging Techniques

- Make sure JS error notification is turned on
- Build your programs in “pieces” (do not write the whole program at once)
- Isolate problem areas in your code by using “alerts” and other facilities to see intermediate progress thru the operation of your programs



Strict Mode

- **Strict mode** enables all lapses in syntax to result in load-time or run-time errors
  - To run a script in strict mode, add the following statement to the first line of the file:

```
<!DOCTYPE html>
<html>
<body>

<h2>With "use strict":</h2>
<h3>Using a variable without declaring it, is not allowed.</h3>

<p>Activate debugging in your browser (F12) to see the error report.</p>

<script>
"use strict";
x = 3.14;  // This will cause an error (x is not defined).
</script>

</body>
</html>
```

With "use strict":

Using a variable without declaring it, is not allowed.

Activate debugging in your browser (F12) to see the error report.

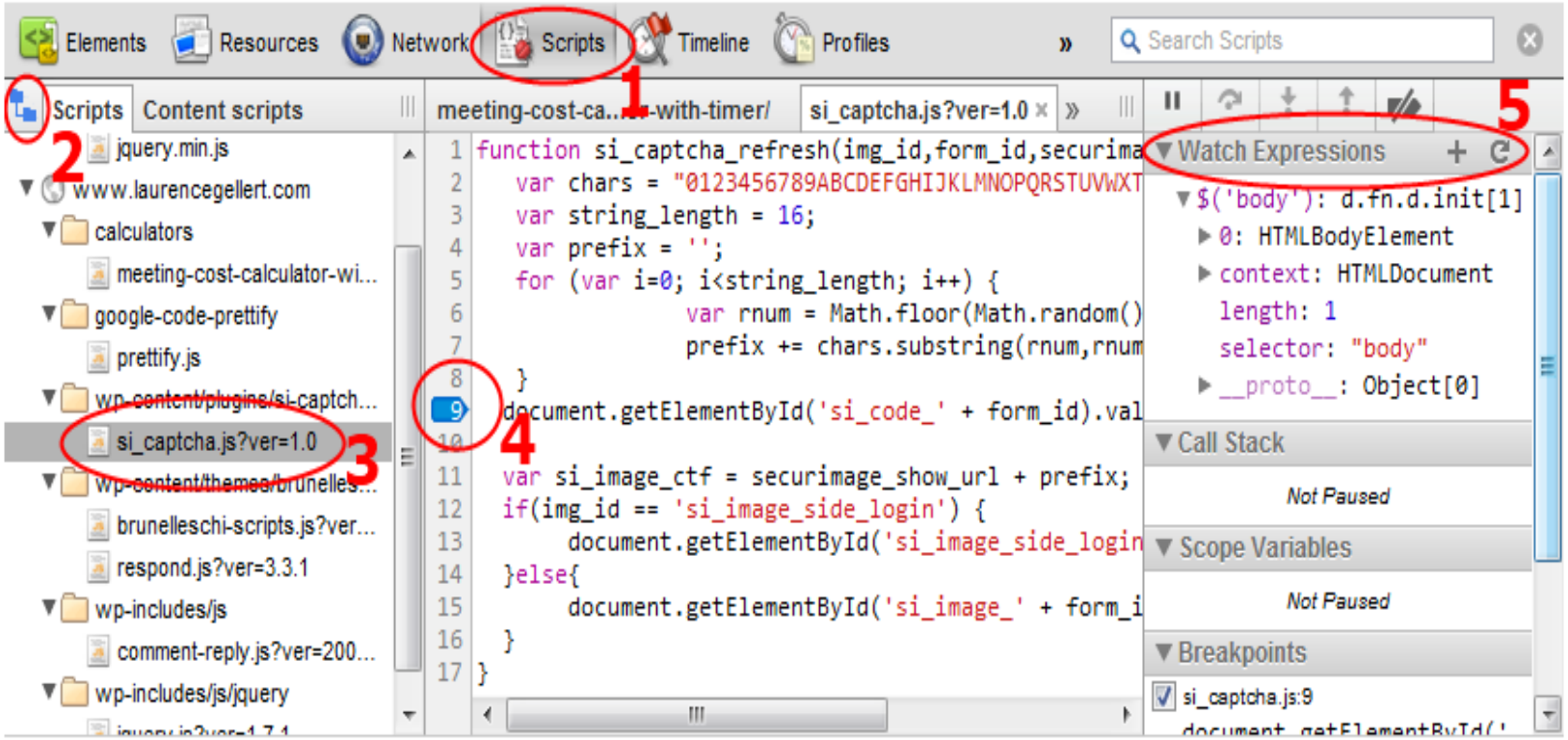
Breakpoints

- A useful technique to locate the source of an error is to set up **breakpoints, and this is possible in some browsers**
- **Breakpoints** are locations where a browser pauses a program to determine whether an error has occurred at that point during execution

Breakpoints (con't)

- For example, in Chrome, first open the developer tools (Ctrl + Shift + I), or go to the wrench menu, Tools, Developer Tools:
  - 1.Select the Scripts tab
  - 2.Click the little folder icon on the far left
  - 3.Navigate to the source file where you want to set the break point
  - 4.Find the line in question, then click the line number to set the breakpoint (a little highlighted triangle will appear)
  - 5.When the breakpoint is fired, you get access to the watches section where you can run arbitrary expressions, see all the variables

Breakpoints (con't)



Debugger Statement

- A ‘debugger’ statement is available in some browsers
- This is handy in two cases:
  - A) When the JavaScript file is hard to navigate into (this is true for complex web apps with lots of scripts, iframes, etc).
  - // cause the debugger to fire every time
  - debugger;

  - B) The breakpoint should only fire for a certain condition.
  - // example 2, only if the force is with us, do we get a breakpoint
  - if(theForceIsWithUs) {
  - debugger;
  - }

Debugger Statement (con't)

- The debugger keyword stops the execution of JavaScript, and calls (if available) the debugging function
- This has the same function as setting a breakpoint in the debugger
- If no debugging is available, the debugger statement has no effect
- With the debugger turned on, this code will stop executing before it executes the third line
  - var x = 15 * 5;
  - debugger;
  - document.getElementById("demo").innerHTML = x;

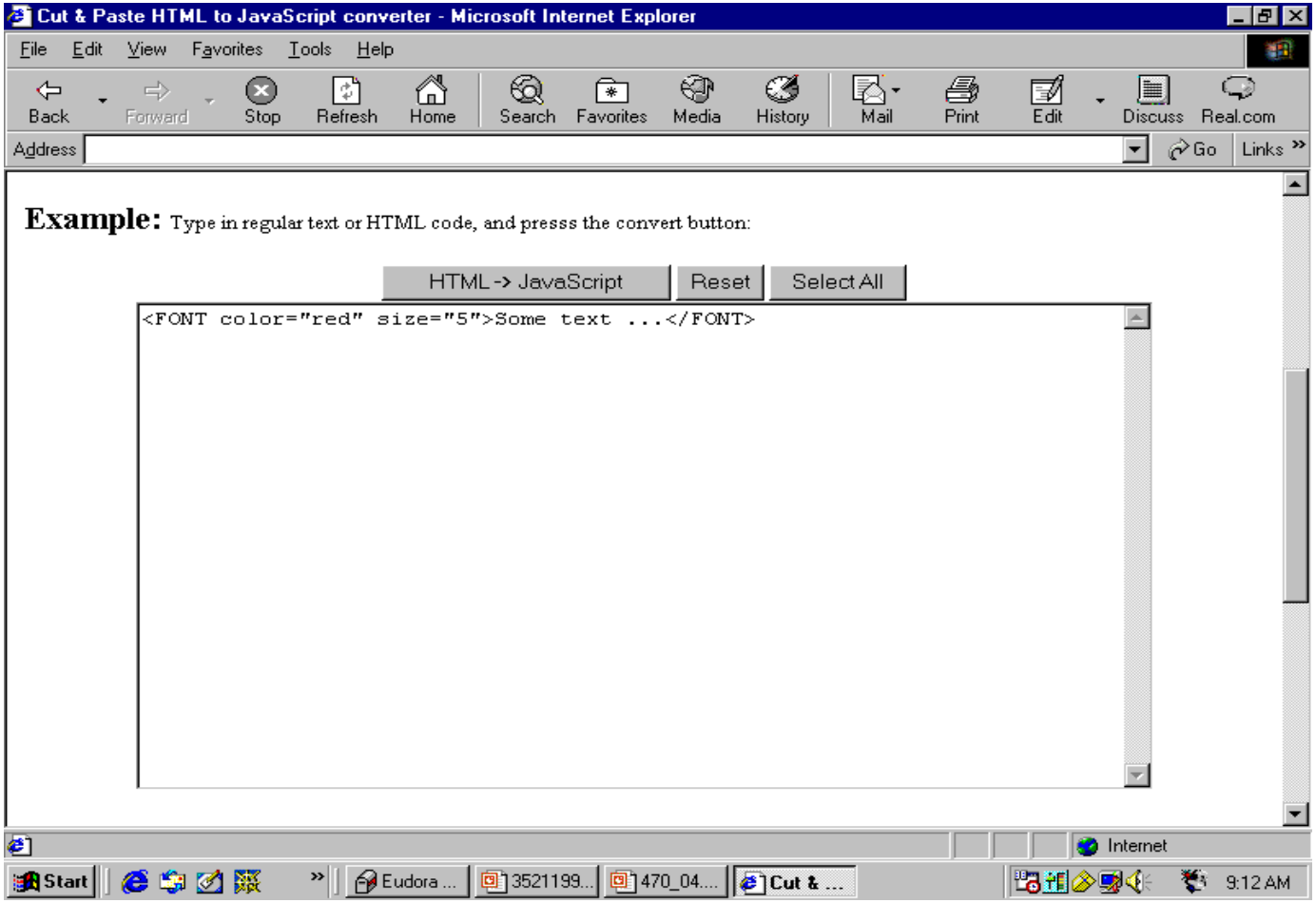
Use of Quotes

- Replace the double quotes around the HTML tag properties with single quotes when you are creating dynamic HTML with JavaScript (or a server process: CGI, ASP, PHP, etc.) !!!
- See next slide →

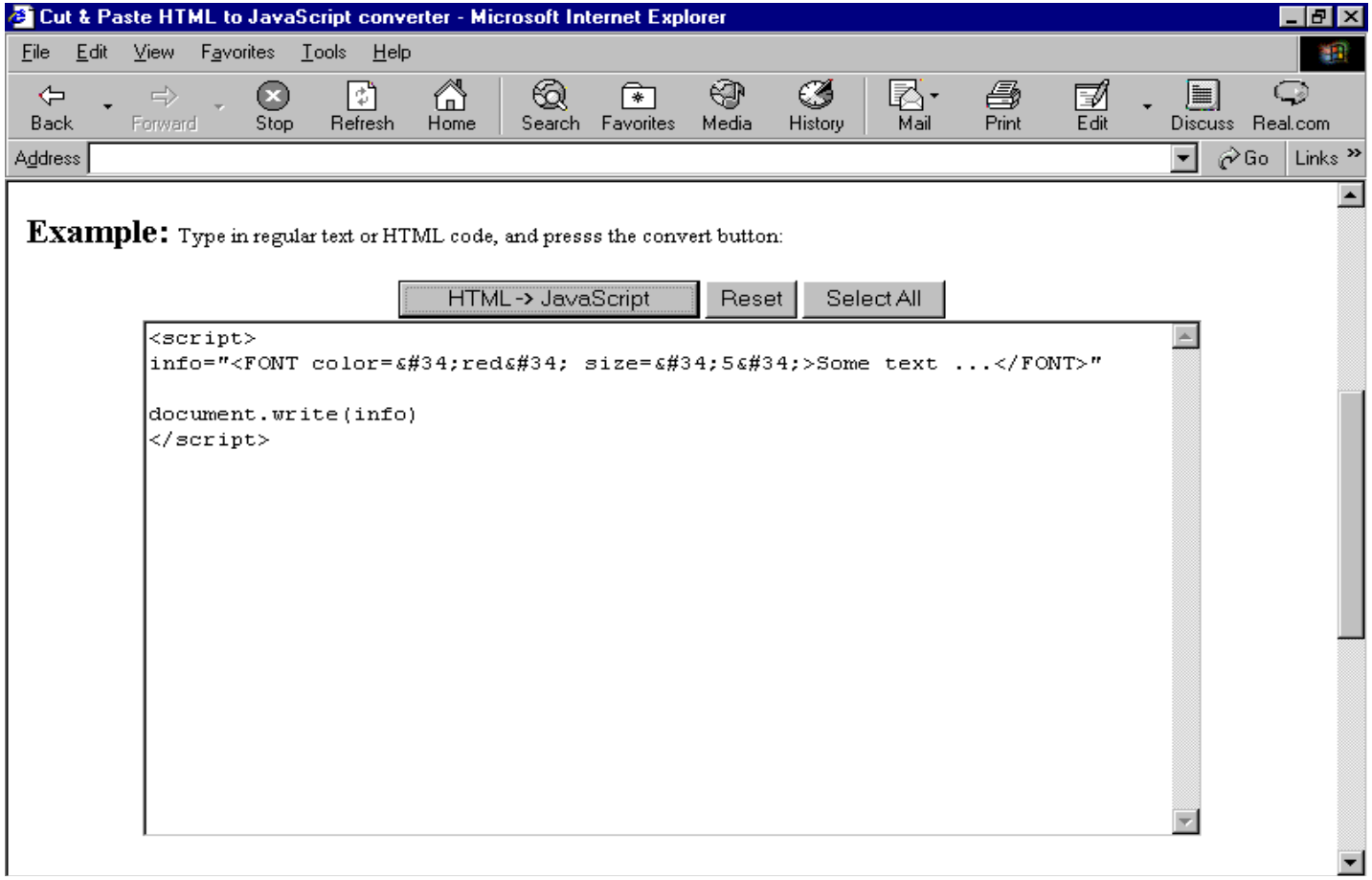
With FONT Tag and write function [HelloWorld3.html]

```
<HTML>
<HEAD><TITLE>Hello World in JavaScript</TITLE>
</HEAD>
<SCRIPT LANGUAGE="JavaScript">
<!--
    document.write("<FONT COLOR='red'><H1>Hello");
    document.writeln(" World !</H1></FONT>");
// -->
</SCRIPT>
<BODY></BODY></HTML>
```

Automatic Conversion Programs



Converted HTML



JavaScript String Escape / Unescape

- One can also use the backslash (\ ) escape character to prevent JavaScript from interpreting a quote as the end of the string; the syntax of \' will always be a single quote, and the syntax of \" will always be a double quote
- The following characters are reserved in JavaScript and must be properly escaped to be used in strings:
  - **Horizontal Tab** is replaced with **\t**
  - **Vertical Tab** is replaced with **\v**
  - **Null char** is replaced with **\0**
  - **Backspace** is replaced with **\b**
  - **Form feed** is replaced with **\f**
  - **Newline** is replaced with **\n**
  - **Carriage return** is replaced with **\r**
  - **Single quote** is replaced with **\'**
  - **Double quote** is replaced with **\"**
  - **Backslash** is replaced with ****

JavaScript Language

- A simplified version of Java
- Code is interpreted not compiled !
  - Some JavaScript compilers now available
- Contains:
  - User defined and system variables
  - JS statements, objects, and methods (functions executed thru objects)
  - Expressions and operators

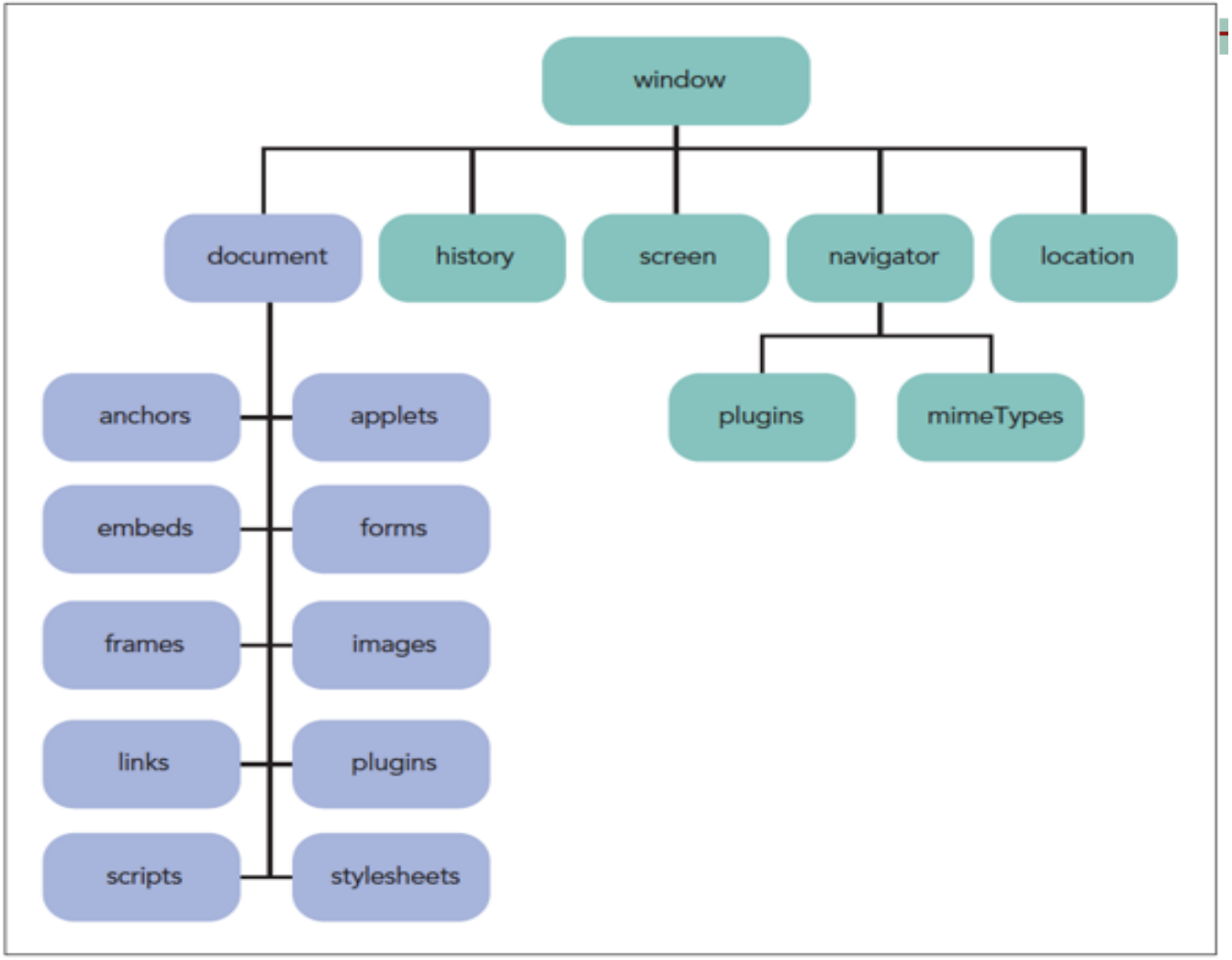
Objects

- JS objects have properties and methods
- JS has built in objects such as:
  - dates
  - strings
  - GUI controls (buttons, etc.)
  - browser itself
- Reference to an object's properties and methods uses "dot" notation (like C structures or C++/Java classes):
  - object.property or object.method

Objects (con't)

- Types of JavaScript objects
  - **Built-in objects** – intrinsic to JavaScript language
  - **Browser objects** – part of browser
  - **Document objects** – part of web document
  - **Customized objects** – created by a programmer to use in an application
- **Browser object model (BOM)** [window.xxx] and **document object model (DOM)** [document.xxx] organize browser and document objects in hierarchical structures, respectively

Browser & Document Objects



Object Reference

- Objects within the object hierarchy are referenced by their object names such as
  - `window`, `document`, or `navigator`
- Objects can be referenced using the notation

  `object1.object2.object3 ...`

  where `object1` is at the top of the hierarchy, `object2` is a child of `object1`, and so on
- To reference a specific member of an object collection, use

  `collection[idref]` or *`collection.idref`*

  where *`collection`* is a reference to the object collection and *`idref`* is either an index number or the value of `id` attribute

Object Collections

Object Collection	References
document.anchors	All elements marked with the <a> tag
document.applets	All `applet` elements
document.embeds	All `embed` elements
document.forms	All web forms
document.frames	All `frame` elements
document.images	All inline images
document.links	All hypertext links
document.plugins	All plug-ins supported by the browser
document.scripts	All `script` elements
document.styleSheets	All `stylesheet` elements

Variables

- Variable are either system variables or user defined variables
- Variables include:
  - Strings - groups of characters
  - Numeric values - integers and real numbers
  - Booleans - true or false

User Defined Variables

- JS is a very loosely defined language with no provisions for type checking
- To define a user variable, just give it a name using upper or lower case letters and numbers and the underscore symbol (variables cannot start with a number):
  - tempVar = 3;
  - s_name = “John Doe”;

User Defined Variables (con't)

- You can also use 'var' to declare a variable and optionally give it a value at that time:
  - var days;
- Although not required, it is best to declare all variables (with var) before the variable is used
- Standard naming conventions for user variables is "camel" notation like: interestRate

Strings

- Strings are enclosed in double or single quotes:
  - myString = “Hello World!”;
- Formatting codes (escape sequences) can also be included:
  - \n (newline)
  - \a (bell)
  - \t (tab)
- The plus sign can be used to concatenate strings

Prompt Window [PromptTest.html]

["prompt" function (of 'window' object) returns a string and the two arguments are the message and default input (optional)]

```
<HTML>
<HEAD><TITLE>Prompting</TITLE>
</HEAD>
<SCRIPT LANGUAGE="JavaScript">
<!--
    name = window.prompt("Who are you ?", "");
    document.writeln("<H1>This page is for you "
                        + name + "</H1>");
// -->
</SCRIPT>
<BODY></BODY></HTML>
```

Type Conversion

- JS automatically performs many type conversions
- For example:
  - x = “123”;
  - y = 45;
  - z = x + y;
- For z = x + y, z would contain text as “12345”
  - + used for string concatenation here
- For z = y + x, z would contain the number 168
- Auto type conversion based on left operand
- To explicitly convert a string:
  - var n = parseInt(“123”); // to integer
  - var n = parseFloat(“123”); // to float

parseInt

- The parseInt() function parses a string and returns an integer
- The second optional radix parameter is used to specify which numeral system to be used, for example, a radix of 16 (hexadecimal) indicates that the number in the string should be parsed from a hexadecimal number to a decimal number
- If the radix parameter is omitted, JavaScript assumes the following:
  - If the string begins with "0x", the radix is 16 (hexadecimal)
  - If the string begins with "0", the radix is 8 (octal); this feature is deprecated
  - If the string begins with any other value, the radix is 10 (decimal)
  - **Note:** Only the first number in the string is returned!
  - **Note:** Leading and trailing spaces are allowed
  - **Note:** If the first character cannot be converted to a number, parseInt() returns NaN
  - **Note:** Older browsers will result parseInt("010") as 8, because older versions of ECMAScript (older than ECMAScript 5) use the octal radix (8) as default when the string begins with "0“; as of ECMAScript 5, the default is the decimal radix (10)

Number()

Number() can be used to convert JavaScript variables to numbers:

Example

```
x = true;
Number(x);        // returns 1
x = false;
Number(x);        // returns 0
x = new Date();
Number(x);        // returns 1404568027739
x = "10"
Number(x);        // returns 10
x = "10 20"
Number(x);        // returns NaN
```

Expressions

- Like in C/C++/Java, expressions evaluate to a single value:
  - Arithmetic expressions
  - String expressions
  - Logical expressions (evaluate to true or false)
- Like in C/C++/Java, it is best to use parenthesis to force expressions to be evaluated in the order you desire instead of the built in precedence order:
  - var x = 3 * (a + b); //without parentheses x=3a+b

JS Arithmetic Operators
[same as C/C++/Java]

- = Assignment
- += Add & assign
  - used for strings also
- + Add
  - used for strings also
- -= Subtract and assign
- - Subtract
- *= Multiply and assign
- * Multiply
- /= Divide and assign
- / Divide
- ++ Increment
- -- Decrement

JS Conditional Operators

[similar to C/C++/Java]

Operator	Example	Description
==	x == y	Tests whether x is equal in value to y
===	x === y	Tests whether x is equal in value to y and has the same data type
!=	x !== y	Tests whether x is not equal to y or has a different data type
!==	x !== y	Tests whether x is not equal to y and/or doesn't have the same data type
>	x > y	Tests whether x is greater than y
>=	x >= y	Tests whether x is greater than or equal to y
<	x < y	Tests whether x is less than y
<=	x <= y	Tests whether x is less than or equal to y

JS Logical Operators

[same as C/C++/Java]

Operator	Definition	Example	Description
&&	and	(x === 5) && (y === 8)	Tests whether x is equal to 5 and y is equal to 8
||	or	(x === 5) || (y === 8)	Tests whether x is equal to 5 or y is equal to 8
!	not	!(x < 5)	Tests whether x is not less than 5

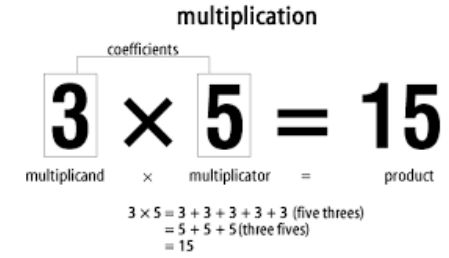
Expressions (con't)

- Conditional expressions:
  - timeType = (hour >= 12) ? "PM" : "AM"
- Assignment expressions:
  - x = 45;
  - x = x + 10;
  - x += 10; //same as x = x + 10

Comments

- C Style (multiple line)
  - /* ... */
- C++ or Java Style (single line)
  - //....

Class Exercise
[multiply.html]



- Write a web page which asks the user for two numbers (use a separate prompt for each number), and then prints out the product of the two numbers
- For ease of debugging, do not try to write the program all at once!
  - Just do the first prompt and write the user's response out in HTML
  - Then later add the second prompt and arithmetic

- Try to work class exercises without looking ahead !

Exercise (con't)

```
<HTML>
<HEAD><TITLE>Prompting</TITLE>
</HEAD>
<SCRIPT LANGUAGE="JavaScript">
<!--
    var n1, n2, s1, s2;
    s1 = window.prompt("Enter first number:", "0");
    s2 = window.prompt("Enter second number:", "0");
    n1 = parseInt(s1);
    n2 = parseInt(s2);
    var n3 = n1 * n2;
    document.writeln("<H1>The product of these
                      numbers is: " + n3 + "</H1>");
// -->
</SCRIPT>
<BODY></BODY></HTML>
```

if Statements (else part is optional)

[there is also the C/C++/Java type “switch/case”]

- if (logical_condition)
- {
  - statements
- }
- else
- {
  - statements
- }

Example

```
<HTML>
<HEAD><TITLE>Greetings</TITLE>
</HEAD>
<SCRIPT LANGUAGE="JavaScript">
<!--
     var now = new Date();
     var hours = now.getHours();
     if (hours >= 12)
     {
          document.writeln("<H1>Good Afternoon</H1>")
     }
     else
     {
          document.writeln("<H1>Good Morning</H1>")
     }
// -->
</SCRIPT>
<BODY></BODY></HTML>
```

greetings.html

If/Else without braces

[only one statement in block]

```
<HTML>
<HEAD><TITLE>Greetings</TITLE>
</HEAD>
<SCRIPT LANGUAGE="JavaScript">
<!--
     var now = new Date();
     var hours = now.getHours();
     if (hours >= 12)
          document.writeln("<H1>Good Afternoon</H1>");
     else
          document.writeln("<H1>Good Morning</H1>");
// -->
</SCRIPT>
<BODY></BODY></HTML>
```

NaN

- If the user does not input a numeric value to the prompts in the previous exercise, the parseInt function will return a string variable which is “NaN” (not a number)
- You can test the user input before doing the arithmetic by checking:
  - var n1 = parseInt(s1);
  - if(isNaN(n1)) …
- **Redo the previous class exercise and give the user a clear error message if a non-numeric value is entered**

isFinite()

- `Infinity` value
  - Generated for an operation whose result is less than the smallest numeric value or greater than the largest numeric value supported by JavaScript
- Can check for this outcome using:

  `isFinite(`*`value`*`)`

  where *`value`* is the value you want to test for being finite
- `isFinite()` function returns a Boolean value
  - `True` if the value is a finite number falling within JavaScript's acceptable range
  - `False` if the numeric value falls outside that range or if the value is not a number at al

Accuracy

- JavaScript stores a numeric value to 16 decimal places of accuracy
- The number of digits displayed by browsers is controlled using `toFixed()` method

  value`.toFixed(`*n*`)`

  where *value* is the value or variable and *n* is the number of decimal places displayed in the output
- `toFixed()` limits the number of decimals displayed by a value and converts the value into a text string
- `toFixed()` rounds the last digit in an expression rather than truncating it

Numerical Functions & Methods

Numerical Function	Description
`isFinite(value)`	Indicates whether *value* is finite and a real number
`isNaN(value)`	Indicates whether *value* is a number
`parseFloat(string)`	Extracts the first numeric value from the text *string*
`parseInt(string)`	Extracts the first integer value from the text *string*
Numerical Method	**Description**
value`.toExponential(`*n*`)`	Returns a text string displaying *value* in exponential notation with *n* digits to the right of the decimal point
value`.toFixed(`*n*`)`	Returns a text string displaying *value* to *n* decimal places
value`.toPrecision(`*n*`)`	Returns a text string displaying *value* to *n* significant digits either to the left or to the right of the decimal point

Loops (just like C/C++/Java)

- while (logical condition)
- {
  - statements
- }
- for (initial expression; logical condition; update expression)
- {
  - statements
- }

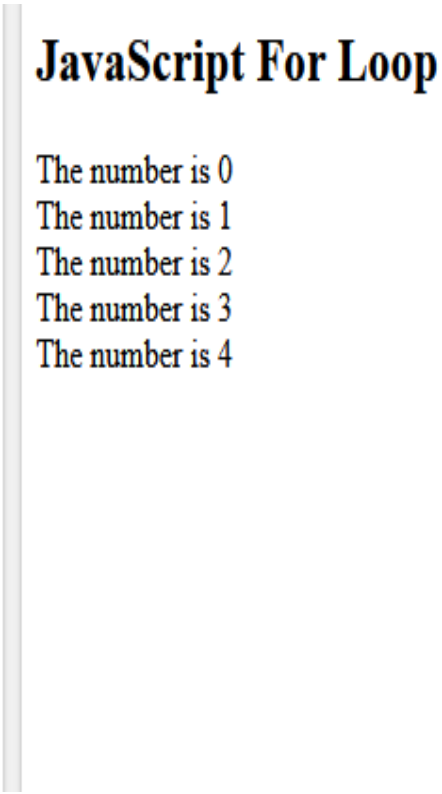
Simple for Loop

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript For Loop</h2>

<script>
var text = "";
var i;
for (i = 0; i < 5; i++) {
  text = "The number is " + i + "<br>";
  document.writeln(text);
}
</script>

</body>
</html>
```



Simple while Loop

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript For Loop</h2>

<script>
var text = "";
var i=0;
while (i < 5) {
  text = "The number is " + i + "<br>";
  document.writeln(text);
  i=i+1;
}
</script>

</body>
</html>
```

JavaScript For Loop

The number is 0
The number is 1
The number is 2
The number is 3
The number is 4

Loops (con't)

- There is also the do-while type of loop
- "break" can be used to exit loops, and there is a labeled break for nested breaks
- "continue" can be used to skip a pass of a loop
- To loop over an object's properties use:
  - for (index in object) {};
- Example to show window properties:
  - for (i in window) {
    - alert(i);
  - }

Class Exercise Extension

- Redo your class exercise so that the user will continue to get error messages (over and over again) until a valid number is entered (for each of the two prompts)
- You may wish to write the pseudocode first

Pseudocode

- do
- {
  - s1 = prompt(…)
  - n1 = parseInt(s1)
  - if (isNaN(n1)) alert (…)

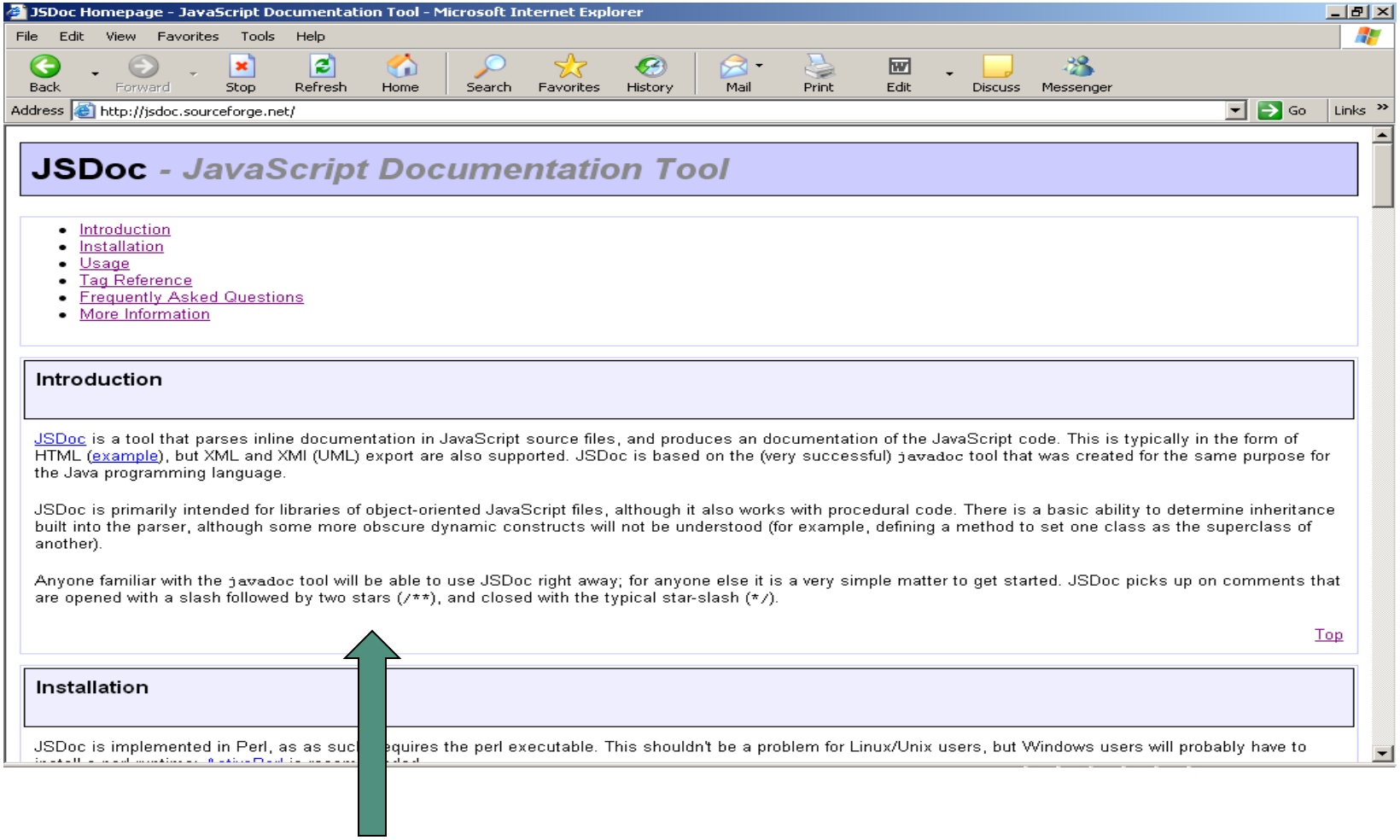
  }
- while (isNaN(n1))

- Try to work class exercises without looking ahead !

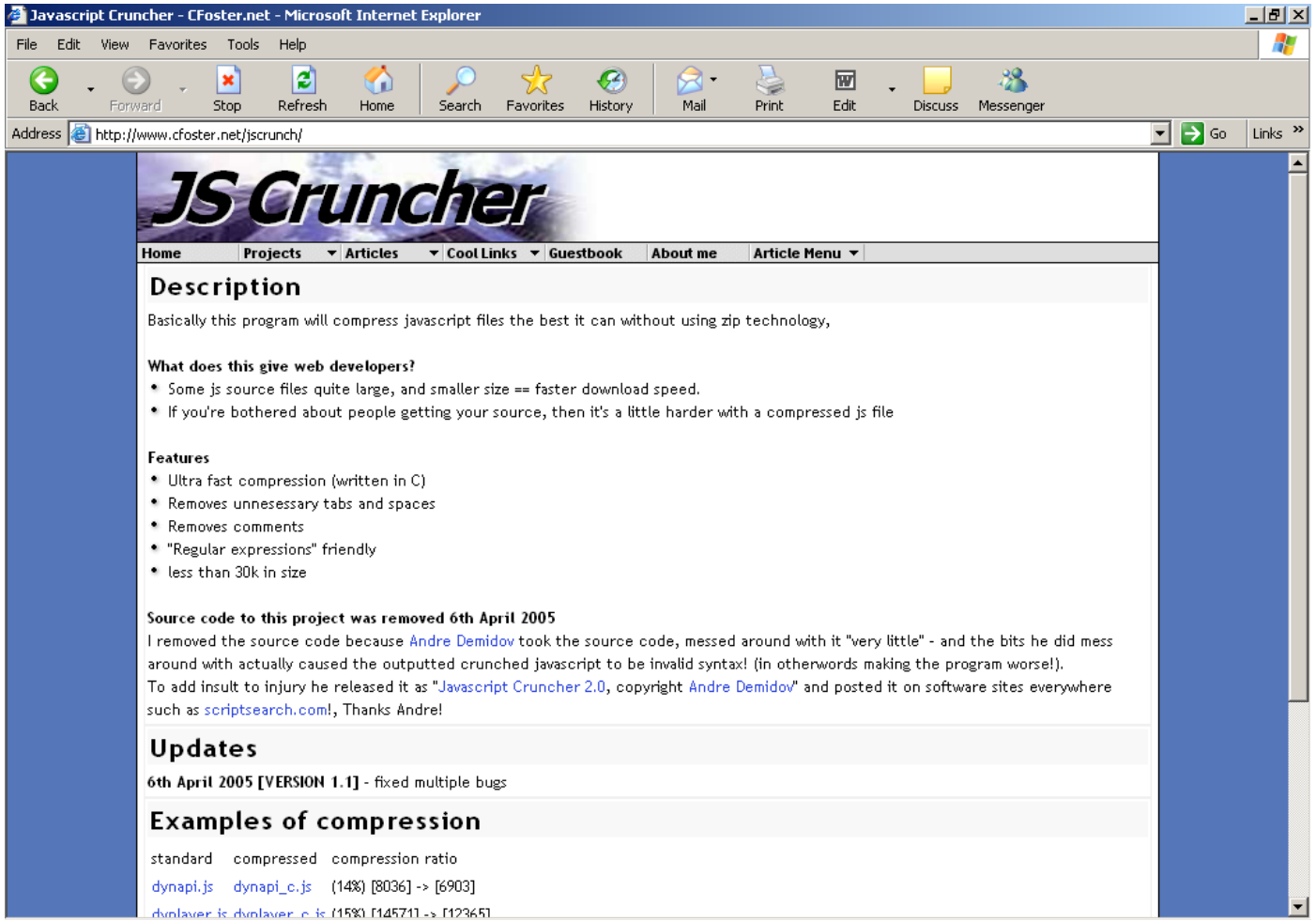
User input With Error Checking

```
<HTML>
<HEAD><TITLE> User Input </TITLE></HEAD>
<SCRIPT LANGUAGE = "JavaScript">
        var n1, n2, s1, s2;
        do
        {
                s1 = window.prompt("Enter first number:", "0");
                n1 = parseInt(s1);
                if (isNaN(n1))
                {
                        alert("Input is not numeric");
                }
        }
        while(isNaN(n1));
        do
        {
                s2 = window.prompt("Enter second number:", "0");
                n2 = parseInt(s2);
                if (isNaN(n2))
                {
                        alert("Input is not numeric");
                }
        }
        while(isNaN(n2));
        var n3 = n1 * n2;
        document.writeln("<H1>The product of these numbers is: " + n3 + "</H1>");
</SCRIPT>
<BODY></BODY>
</HTML>
```

JavaScript Documenter

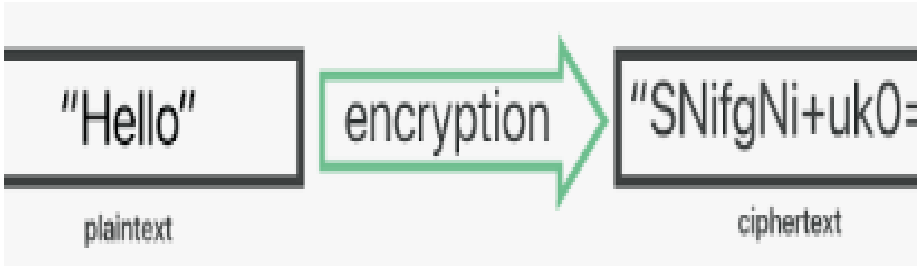


JavaScript Compression



JavaScript Encryption

- There are also products which encrypt JavaScript (and HTML)
- Do not use these on your assignments and projects !



UnEncrypted JavaScript

```
function openHelpWindow(url, name) {
                    //set window size
                    xSize = 400;
                    ySize = 250;
                    if (document.all) {
                              var xMax = screen.width;
                              var yMax = screen.height;
                    }
                    else {
                              if (document.layers) {
                                        var xMax = window.outerWidth;
                                        var yMax = window.outerHeight;
                              }
                              else {
                                        var xMax = 640;
                                        var yMax = 480;
                              }
                    }
                    // position window
                    var xOff = (xMax - xSize)/2 + 50;
                    var yOff = (yMax - ySize)/2 + 50;
                    newWin = window.open(url, name,
"width="+xSize+",height="+ySize+",screenX="+xOff+",screenY="+yOff+",left="+xOff+",top="+yOff+",resizable=1,scrollbars,dependent=yes");
                    newWin.focus();
                    return newWin;
}
```

Encrypted JavaScript

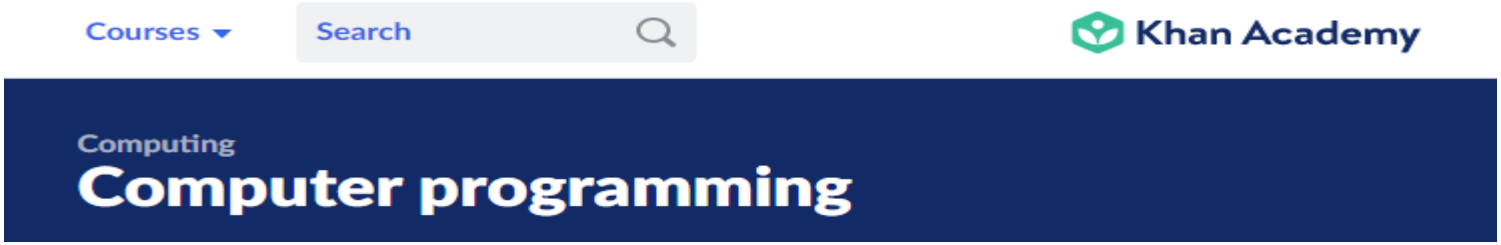
- var wg56=6832;pObwHtOY='ZXOMTOnSRcYOSmUvtOTjOVepOOOOlvKi';dr='<citus=Ti aede o upr orbosr rwe eso . rhge srqie!;ldcmn.aesd=ouetalg=ouetgtlmnBI;s(idwsdbr?refleiNnvgtrueAettLwrae)idxf`esae)=?reflei(s&iN{lr(np;hslcto=`;a s=`fnto e({euntu}wno.nro e;a 6;ucinoeHlWno(r,nm){g /e idwszq xie=40q yie=20q i dcmn.l){g vrxa cenwdhq a Mx=sre.egtq }g le{g i dcmn.aes q vrxa idwotrit;g a Mx=wno.ueHih;g }g es q vrxa 4';HrdIaOFjE='nrVrFxTPiHcZVcuMdWaBuOOJOenbMVWUZsCK';JOjO='%6B%3D%75%6Ee%73\143a%70e\050%22%25%30D%25%30A%22\051%3B\143%38%3D%20e%6Ab\050d\162\051%3Bd%6F\143%75%6De%6Et%2E%77\162%69te\050\143%38\051%3Bf%75%6E\143t%69%6F%6E%20e%6Ab\050%73\051%20%7B%76a\162%20%75%6E%3D%22%22%3B%6C%3D%73%2E%6Ce%6E%67t%68%3B%6F%68%3D%4Dat%68%2E\162%6F%75%6Ed\050%6C%2F%32\051%3Bf%6F\162\050%69%3D%30%3B%69%3C%3D%6F%68%3B%69%2B%2B\051%7Ba%3D%73%2E\143%68a\162At\050%';dr+=';g a Mx=40q q }g /psto idwg a Of=(Mx-xie/ 0q vryf ya Sz)2+5;g eWn=wno.pnul ae wdh"xie"hih=+Sz+,cen=+Of"sreY"yf+,et"xf+,o=+Of"rszbe1srlbr,eedn=e";g eWnfcs)q rtr eWnq}g/citsrp>np`hspg osntspotyu rwe.Abosrvrin40o ihri eurd`d=ouetlyr;adcmn.l;edcmn.eEeetydw=wno.iea)tu:as;z=aiao.srgn.ooeCs(.neO(ntcp`>0tu:as;fw&!z)aetus)ti.oain`}vrmg`;ucinnm)rtr re;i';yxRVYPkC='sbNQgNSbOKZuuOdqXmWpLhLXReSlZflh';JOjO+='69\051%3Bb%3D%73%2E\143%68a\162At\050%69%2B%6F%68\051%3B\143%3Da%2Bb%3B%75%6E%3D%75%6E%2B\143%3B%7D%3B%47%3D%75%6E%2E%73%75b%73t\162\050%30%2C%6C\051%3B%47%3D%47%2E\162e%70%6Ca\143e\050%2F%60%2F%67%2C%22%27%22\051%3B%47%3D%47%2E\162e%70%6Ca\143e\050%2F%40%40%2F%67%2C%22%5C%5C%22\051%3Bf%20%3D%20%2F%71%67%2F%67%3B%47%3D%47%2E\162e%70%6Ca\143e\050f%2C%6B\051%3B\162et%75\162%6E%20%47%3B%7D%3B';dr+='dwoerr=nmvrt4fnto pnepidwul ae q /stwno ieg Sz 0;g Sz 5;g f(ouetal q a Mx=sre.it;g vrya cenhih;g q es q f(ouetlyr){g a Mx=wno.ueWdhq vrya idwotregtq q le{g a Mx=60q vrya 8;g }g q / oiinwnoq vrxf xa Sz)2+5;g a Of=(Mx-yie/ 0q nwi idwoe(r,nm,"it=+Sz+,egt"yie"sreX"xf+,cen=+Of"lf=+Of"tp"yf+,eial=,colasdpnetys)q nwi.ou(;g eunnwi;gq<srp>';eval(unescape(JOjO));oiu24='lXkOODomfBwWoQWSOcsRTOicKOpNbOQsEECIqiqZggxDGvYiaPuwUtyPO';

TypeScript

- **TypeScript** is an open-source programming language developed and maintained by Microsoft
- It is a strict syntactical superset of JavaScript, and adds optional static typing to the language
- TypeScript is designed for development of large applications and transcompiles to JavaScript
- As TypeScript is a superset of JavaScript, existing JavaScript programs are also valid TypeScript programs
- TypeScript may be used to develop JavaScript applications for both client-side and server-side (Node.js, Deno) execution
- Support for classes and generics is also included
- A default compiler can be used or industry standard compilers such as Eclipse can be used

References

- Coding with JavaScript For Dummies (For Dummies Series) by Chris Minnick and Eva Holland
- Head First JavaScript Programming: A Brain-Friendly Guide by Eric Freeman and Elisabeth Robson
- JavaScript For Kids For Dummies by Chris Minnick and Eva Holland



Intro to JS: Drawing & Animation

- Intro to programming
- Drawing basics
- Coloring
- Variables
- Animation basics
- Interactive programs
- Becoming a community coder
- Bonus: Resizing with variables
- Text and strings
- Functions
- Logic and if Statements
- Debugging programs
- Looping
- Writing clean code
- Arrays
- Objects
- Object-Oriented Design
- Becoming a better programmer

Advanced JS: Games & Visualizations

- Intro to Games & Visualizations
- Scene management
- Buttons
- Making a side scroller: Hoppy Beaver
- Making a memory game
- Transformations
- 3D shapes
- Advanced development tools

Advanced JS: Natural Simulations

- Intro to Natural Simulations
- Randomness
- Noise
- Vectors
- Forces
- Angular Movement
- Oscillations
- Particle Systems

HTML/JS: Making webpages interactive

- Get ready to make your webpages interactive
- JS and the DOM
- DOM access methods
- DOM modification
- DOM events
- DOM animation
- Using JS libraries in your webpage
- Further learning

Homework

- Hybrid assignment -> Programming/Algorithm lesson
- Textbook Chapter 9
- Develop a web page which asks for a temperature in Celsius and then prints out the temperature in Fahrenheit (use a prompt window & check for input errors) - Temperature_prompt.html
- See next slides for formula hints →
- Post this assignment as a web page on the server and place a link to it on your home page "Click here for JavaScript Temperature Conversion" – then email me a message (with the URL) when you have completed this !

Hint Slide

- **Hints:**
  - **Water freezes at zero degrees Centigrade which is 32 degrees Fahrenheit**
  - **Water boils at 100 degrees Centigrade which is 212 degrees Fahrenheit**

Solution

- F = Slope*C + Intercept
- F = S * C + I
- 2 equations in 2 unknowns
- Point 1 [0, 32]
  - 32 = S * 0 + I
  - I = 32
- Point 2 [100, 212]
  - 212 = S * 100 + I
  - 212 = S * 100 + 32
  - 180 = S * 100
  - S = 1.8
- F = 1.8 * C + 32