

# BrainBuzz

## Cramsession

---

Last updated November, 2000.  
Click [here](#) for updates.

Click [here](#) to see additional documents related to this study guide.

### Contents

Contents .....	1
What is Javascript? And why are we using Javascript? .....	2
Basic Fundamentals .....	2
For "Non Javascriptable" Browser .....	4
Event-Based Model .....	4
Script Tags in detail .....	6
Events in detail .....	6
Data Types, Variables and Operators in detail .....	7
String in detail .....	11
Conditionals Statements in detail .....	12
Loop Structure in detail .....	13
Functions in detail .....	14
Object in detail .....	15
Javascript Applications .....	16

## Cramsession™ for CIW JavaScript Fundamentals

### Abstract:

This Cramsession will help you to prepare for the CIW exam 1D0-435, JavaScript Fundamentals. Topics include JavaScript, Event-based model, Script Tags, Data Types, Variables, Operators, Strings, Conditional Statements, Functions, Objects, Cookies, Security Issues and Bugs.



Notice: While every precaution has been taken in the preparation of this material, neither the author nor BrainBuzz.com assumes any liability in the event of loss or damage directly or indirectly caused by any inaccuracies or incompleteness of the material contained in this document. The information in this document is provided and distributed "as-is", without any expressed or implied warranty. Your use of the information in this document is solely at your own risk, and Brainbuzz.com cannot be held liable for any damages incurred through the use of this material. The use of product names in this work is for information purposes only, and does not constitute an endorsement by, or affiliation with BrainBuzz.com. Product names used in this work may be registered trademarks of their manufacturers. This document is protected under US and international copyright laws and is intended for individual, personal use only. For more details, [visit our legal page](#).

---

## What is Javascript? And why are we using Javascript?

- Script – similar to macro or batch file, a script is a list of commands that can be executed without user interaction.
- Script language - a relatively simple programming language with which you can write scripts.
- JavaScript is a scripting language developed by Netscape to add interactivity to web pages.
- On many web pages you can find examples such as live clocks, rollover effects, scrollers, form validations, etc.
- With JavaScript, you can break the limitation of static web content and allow your creativity to dictate what you put on your webpage.
- Keep in mind that Java is completely different from JavaScript. Java is a lot more powerful, more complex, and a lot harder to learn. You need to compile a Java program before you can run it, while with JavaScript there is no need for compilation.
- To create Javascript, you simply open up a text editor, type it, save it, and your browser is ready to run it! Unfortunately, some browser compatibility problems exist as JavaScript was solely created by Netscape. It is 100% compatible with Netscape Navigator. However, Internet Explorer's support of what JavaScript has to offer is not. That is why we think that a good rule to follow is to test code using both browsers before uploading it onto the Internet.

## Basic Fundamentals

- JavaScript is case sensitive!!!
- JavaScript gives you access to conditional looping (such as for loops and while loops), decision-making statements (such as if and switch statements), and objects (such as arrays).
- In the world of programming, a loop is a series of instructions that is repeated until a certain condition is met. Each pass through the loop is called an iteration. Loops constitute one of the most basic and powerful programming concepts.
- In programming, an array is a series of objects all of which are the same size and type. Each object in an array is called an array element. For example, you can have an array of integers or an array of characters or an array of anything that has a defined data type. Each element in the array has the same data type and can have the same or different values, and the entire array is stored contiguously in memory with no gaps in between.
- A variable is a symbol or name that stands for a value in your program. For example, in the expression  $x+y$ ,  $x$  and  $y$  are variables. We can use Variables to represent numeric values, characters, character strings, or memory addresses.

- JavaScript variables are loosely typed, meaning that any variable can represent both a string of characters and a number, depending solely upon what was assigned.
- The opposite of a variable is a constant – a value that never changes. We seldom use constants in web programming. You may treat a variable as a constant by not making subsequent changes to its value.
- To use JavaScript, you enclose JavaScript commands inside of `<script> ... </script>` tags.
- Script tags are usually located at the top of HTML code between the `<head> ... </head>` tags.

Example :

```
<html>
<title>Web page title</title>
<head>
  <script>
    JavaScript code
  </script>
</head>

<body>
  The body of your web page
</body>
</html>
```

- We organize JavaScript commands into functions located within the `<script> ... </script>` tags.
- Functions are a sequence of commands that are executed when the function is called.
- We execute the Javascript functions within the `<body> .... </body>` tags of the HTML document.
- `//` is used for JavaScript comments - text after `//` is ignored by browsers for the entire line.
- `;` ends every line of JavaScript code and tells the browser it has reached the end of a JavaScript statement.



---

## For “Non Javascriptable” Browser

- For people who browse the Internet without the ability to view documents that contain JavaScript commands, place JavaScript commands in HTML comment tags as follows :

```
<script>
    <!-- Hides script from old browsers.
    JavaScript code...
    // End the hiding -->
</script>
```

- Browsers that cannot interpret JavaScript will be instructed to skip the HTML commented JavaScript tags to avoid problems with JavaScript commands.
- JavaScript-enabled browsers ignore the HTML comments in the <script> .... </script> and interpret the JavaScript commands as usual.
- To provide additional information for non-JavaScript compatible clients, you may add extra HTML code using the <noscript> .... </noscript> tag in the <body> .... </body> of the HTML document as follow:

```
<noscript>
    HTML for non-JavaScript client

    Go to download.com to get a new browser and come back!
</noscript>
```



## Event-Based Model

- An event is an action or occurrence detected by a program. It can be user actions, such as clicking a mouse button or pressing a key, or system occurrences, such as running out of memory.
- Most modern GUI applications in Windows environments are considered event-driven, as they are designed to respond to events.
- JavaScript works by responding to events caused by the user.
- For example, if the user moves their mouse over a link on your web document, JavaScript responds to this onMouseOver event and runs the corresponding command.
- You use Event Handlers to call a JavaScript function as is defined in the <script> .... </script> tags within the <head> .... </head> tags at the top of the HTML file to respond to an event. To use the handler, equate the event

handler within the appropriate tag to the JavaScript function that was predefined in the <script> .... </script> tags.

Example:

```
<html>
<head>
<title>Web page title</title>
  <script>
    function MyFunction( ) {
      }
  </script>
</head>

<body>
<form>
  <input type="button" onClick="MyFunction( )">
</form>
</body>
</html>
```

- Commonly used events:

Event	Description
onChange	User changes the value of a form field
onClick	User clicks the specified button
onLoad	Web page is loaded into the browser
onMouseOver	User moves the mouse over top of the specified link
onMouseOut	User moves the mouse off of the specified link
onSubmit	User clicks a ' type="Submit" ' form button
onUnload	User exits the page entirely

---

## Script Tags in detail

- Javascript works only if you insert it into the right place.
- The easiest way to insert JavaScript into HTML documents is to use the `<SCRIPT>` `</SCRIPT>` tags.
- JavaScript statements are placed between `<SCRIPT>` `</SCRIPT>` tags, and options can be included to further direct the flow of the script. As an example, the language attribute defines the scripting language being used: `<SCRIPT LANGUAGE="JavaScript">`
- The `<SCRIPT>` tags in theory can be used anywhere valid HTML tags can be inserted. However, it's generally better to place scripts within the `<HEAD>` `</HEAD>` section because information placed in the header is not usually displayed by the browser, and JavaScript statements in the header are processed before the document is displayed; i.e., you will want to make sure they are processed before users call them for execution.
- Scripts are executed in the order they are read by the browser, allowing you to place JavaScript-generated information at specific locations in the document.
- You can include separate JavaScript files using the `<SCRIPT>` attribute SRC.
- Storing JavaScript code in a separate file allows multiple HTML documents to share access to the same set of JavaScript code. An external JavaScript file is loaded into a browser and processed while the rest of the document is displayed. Take a look at this example: `<SCRIPT LANGUAGE="JavaScript" SRC="myscript.js">`
- An external Javascript file usually has a .js extension.
- An external Javascript file must be in ASCII text format.
- If you use SRC, the code in between the script tags will be ignored entirely.

## Events in detail

- Events are monitored by event handlers.
- Event handlers perform pre-defined tasks when the corresponding event is triggered.
- The name of an event handler often starts with on, such as onClick.
- Event handlers are included in HTML documents as HTML attributes.

Example:

```
<A HREF="#">This is a link</A>
```

- An HTML element can have multiple event handlers - any combination of available event handlers can be included in a single tag, with each waiting for its own specific event to take action on.

- Example of event handler for user interaction via the mouse:
  - Event Click handles by **onClick**
  - Event MouseOut handles by **onMouseOut**
- Events corresponding to user's interaction are different from Window Event. In fact, event handlers are available for many HTML elements, from links to the browser window itself. Handlers associated with each HTML tag may differ depending on the HTML tag being used.

Examples of common non-user-interaction events and handlers:

Event	Description	Handler
Load	HTML document loads into browser window	onLoad
Unload	HTML document in browser window is replaced by another document	onUnload
Focus	HTML document is chosen as the window's focus	onFocus
Blur	Window's focus is moved to another HTML document	onBlur
Error	HTML document causes an error while loading into window	onError

- Window-related event handlers are processed as attributes of the <body> tag rather than the <A HREF> tag. This is an example: <body onLoad="window.alert('This is an alert.');"0>
- The concept of FOCUS: Focused window is the window being viewed. If another window is selected, the focus is shifted to that window. The original window subsequently lost the focus.

## Data Types, Variables and Operators in detail

### String

- Strings are set of characters grouped together by quotes and are the most commonly used data type.
- Quotes used are either double (") or single (') quotes.
- You must use the same type of quote at the beginning and end of each string.
- Strings can be made up of any combination of characters in general.
- Special Characters used in Strings

Character	Description
\a	Alert character for producing a bell sound
\b	Backspace character that moves cursor back one space
\f	Form-feed character to send new page command to a printer
\n	New line character to continue output on a new line of text
\r	Carriage return character that moves cursor to beginning of line
\t	Tab character that moves cursor to next tab stop
\"	Escape double-quote that allows " to be used in a string
\'	Escape single-quote that allows ' to be used in a string
\\	Escape backslash that allows \ to be used in a string

## Integer

- Integers are whole numbers that can be expressed in decimal, octal or hexadecimal formats
- Decimal is used most often to represent numbers
- Octal notation is written with a leading zero. For example, the octal number of 33 would be written as 033

Note: Any integer beginning with a zero is automatically treated as an octal number.

- Hexadecimal numbers begin with 0x

---

## Floating Point

- Floating point number = Integer numbers plus a fractional decimal value.
- You can refer to very large or very small floating-point numbers using scientific notation. For example: 4,900,000,000 is written as 4.9E9

## Booleans

- Boolean values can have only two possible values - true or false.
- Booleans values are mostly used to test conditions in JavaScript.
- Check box is a common form of Boolean expression – the box can either be checked or not checked.

## Null

- Null is equivalent to an empty value represented by the keyword null.
- Mostly used in determining the state of a variable.

## Variable

- Variable names consist of letters of the alphabet in either uppercase or lowercase format, numerical digits, or the underscore character.
- Variable names cannot have spaces or any other punctuation.
- The first character of the variable name must be a letter or the underscore character.
- Variable can be declared without an initial value using the var keyword:  
var Mynumber;
- Variables declared without value are undefined, as they have no value. This is a good practice to declare variables for the clarity of your code.
- Another way to declare a variable is to declare and assign it a value at the same time:  
Mynumber = 10;

## Array

- An array is used to group groups of related data, allowing them to be associated with a single variable.
- An empty array is created using this syntax:  
Myarray = new Array();
- New is a keyword used to initialize the array, similar to the var keyword in variable.
- Array element starts with 0.
- Once the array is declared, you assign multiple values to the elements of the array:

```
Myname[0] = "Tom";
Myname[1] = "John";
```

- To declare and assign a value at the same time, try the following:  

```
mybook = new Array("book1", "book2", "book3");
```

## Assignment Operators

= Assign the value of the right-hand operand to the variable on the left.

+=, -=, \*=, /=) Add/Minus/Multiply/Divide the value of the right-hand operand to the left-hand variable and stores the result in the left-hand variable.

&=, |= Assigns result of (left-hand operand && / || right-hand operand) to left-hand operand.

## Arithmetic Operators

Operator	Description
+	Adds two strings or numbers
-	Subtracts two numbers. Negates a numerical value
*	Multiplies two numbers
/	Divides two numbers

Shorthand form - for example, the expression `x = x + 1`; can be written as: `x += 1`;  
 Other shorthand notations:

`x++` is equivalent to `x = x + 1`

`x--` is equivalent to `x = x - 1`

## Comparison Operators

Comparisons result in true or false to direct the flow of a program when used with conditional statements.

Operator	Description
==	Returns true if the values compared are equal
!=	Returns true if the values compared are not equal
>	Returns true if the first value compared is greater than the second one
>=	Returns true if the first value compared is greater than or equal to the second one
<	Returns true if the first value compared is less than the second one.
<=	Returns true if the first value compared is less than or equal to the second one.

## Logical Operators

Compare boolean values and return boolean result of true or false.

Operator	Description
&&	AND operator. Returns true if the values compared are both true. Example: x && y
	OR operator. Returns true if either of the values compared are true. Example: x    y
!	NOT operator. Returns the opposite of a boolean value. Example: !x

Example: To find out if both values are true, a logical operator can be used this way:  
(x > 1) && (x != 3)

## String in detail

Information about each specific string is stored in properties. For example, you may retrieve the size of a string by accessing the length property using dot

notation:

```
size = mystrength.length;
```

String methods are called using dot notation as well. For example:  
mystring.mymethod();

Commonly used string methods include:

Method	Description
toLowerCase( )	Converts all characters in string to lowercase
toUpperCase( )	Converts all characters in string to uppercase
indexOf( string )	Searches forward through a string for the given string value
lastIndexOf( string )	Searches backward through a string for the given string value
substring( A, B )	Returns a section of a string, defined by a beginning ( A ) and ending ( B ) index number
charAt( # )	Returns a single character from a string at the index # specified

## Conditionals Statements in detail

Conditional statements direct program flow in specified directions.

### if...else

- **if** condition evaluates to true then the block of statements1 is executed. At the same time, an else clause specifies a block of statements2 which are executed otherwise.
- **else** clause is completely optional.

Example:

```
if (condition)
  { statements1; }
```

```
else
  { statements2; }
```

### **switch**

- **switch** matches an expression with a specified case and executes the corresponding statements.

```
switch (expression){
  case label1 :
    statement;
    break;
  case label2 :
    statement;
    break;
  ...
  default : statement;
}
```



## **Loop Structure in detail**

### **for**

- **for** loop repeatedly cycles until a test condition is false.
- A loop is repeated however many times depending on the counter.
- Initial-statement is executed once to initialize a counter variable. Then the test is applied and the statements are executed. The increment is applied to the counter variable, and then the loop goes through the next cycle.

```
for (initial-statement; test; increment)
  { statements; }
```

- **for...in** statement is used to cycle through each property of an object or an array element.

```
for (variable in object)
{ statements; }
```

### do...while

- **do...while** statement executes a block of statements repeatedly until a condition becomes false.
- Loop executes the statement at least once.

```
do
{ statements; }
while (condition)
```

### while

- **while** statement executes its statement block as long as the condition is true.
- while loop may not execute the statements even once if the condition is initially false.

```
while (condition)
{ statements; }
```

### break and continue

- **break** and **continue** allow you to jump out of iterating loop.
- **break** allows you to abort execution of the loop, then drops out of the loop to the next statement following the loop.
- **continue** allows you to abort only this single iteration of the loop.

## Functions in detail

- Functions group together a chunk of code under a named subroutine, allowing you to call the function whenever its action is required.
- Function definition describes the function name and arguments which it accepts and the underlying statements.
- Arguments are optional. If you specify arguments, the arguments will become the variables within the function body.
- We prefer to define the functions in the HEAD section to guarantee that functions are loaded before the user has a chance to call them.

```
function funcName(argument1,argument2,etc)
{ statements; }
```

- You call a function by specifying its name followed by a parenthetical list of optional arguments: MyPageFunction();
- In any case, there must be () right after the function name, even if no argument exists.
- Some functions can return a value to the calling statements if you specify the return keyword in the code:

```
function myfunc(x)
{
myvar=2*x
return myvar
}
```

## Object in detail

- JavaScript is a language of objects.
- An **object** is a package of data, a collection of properties and methods classed under a single name.
- In JavaScript, you may create your own objects or use the many built-in objects supplied.
- **Properties** are like the variables of object. You access the properties of an object with a simple notation: objectName.propertyName. For example: myObj.name="Mike";
- **Methods** are the functions of an object. You call a method using the basic syntax: objectName.methodName(). For example: window.close();
- An example of an object structure:

```
function myObj(name, tel, address)
{ this.name = name;
  this.tel = tel;
  this.address = address;
}
```

- To create an instance of the object:

```
Mike =new myObj("Mikey","221144","Taiwan");
```

- Document Object Model **DOM** is a hierarchical naming system that makes all of the objects in the page accessible to JavaScript.
- Netscape's DOM structure is different from Microsoft's DOM.
- The root level of the JavaScript document object model is the window object.
- HTML pages are loaded into a window as document objects, the next level up in the object model.
- Each HTML element is a property of the document object.
- For a view of the Javascript Object Tree, please visit:  
<http://wsabstract.com/javatutors/form3.shtml>
- For in depth discuss on Object Model, please visit the following links:  
<http://rummelplatz.uni-mannheim.de/~skoch/js/part2/part2.htm#html>  
<http://www.webreference.com/js/column40/>

## Javascript Applications

You are encouraged to visit the following web site and study the different Javascript example codes:

<http://developer.irt.org/script/script.htm>

<http://www.javascript-page.com/>

<http://www2.southwind.net/~timlitw/ncuc/>

<http://javascript.internet.com>



Note: Only extensive Practice will help you pass!

## Frame

- To ensure that users load your page in the browser's main window, instead of loading in another site's frame, you add the following script into the HEAD portion of your document:

```
<SCRIPT LANGUAGE="JavaScript">
<!--

if (window != top) top.location.href = location.href;

// -->
</SCRIPT>
```

- top is the highest window object in the hierarchy.
- If the topmost page doesn't use a frame, the object model has only one level of window objects, meaning top = window.
- If the browser is currently displaying a frame-based document, each child frame has a corresponding window object.
- If you want to enforce framing, create a frame-setting document:

```
<HTML>
<HEAD>
<TITLE>My Frame Page</TITLE>
</HEAD>
<FRAMESET COLS="200, *">
  <FRAME SRC="frameA.html" NAME="leftcol">
  <FRAMESET ROWS="100, *">
    <FRAME SRC="frameB.html" NAME="toprw">
    <FRAME SRC="frameC.html" NAME="bottomrw">
  </FRAMESET>
</FRAMESET>
</HTML>
```



- You may control navigation between frames using the following HTML codes to create different navigational links:

```
<A HREF="javascript:history.reload()">Reload (this frame)</A><BR>
<A HREF="javascript:history.back()">Back (this frame)</A><BR>
<A HREF="javascript:history.forward()">Forward (this frame)</A><BR>
<A HREF="javascript:parent.frames[2].history.reload()">Reload (bottomrow
frame)</A><BR>
```

```
<A HREF="javascript:parent.frames[2].history.back()">Back (bottomrow  
frame)</A><BR>
```

```
<A HREF="javascript:parent.frames[2].history.forward()">Forward (bottomrow  
frame)</A>
```

## Window

Take a look at the following HTML code fragment:

```
<A HREF="table.html" TARGET="_self">Table of Contents</A>
```

- As you can see, the target of the first link is "\_self". This name loads the linked document into the same window the link was clicked in, meaning the active window. You can instruct Javascript to load the document into other windows:

Target	Description
_blank	Loads the linked document into a new blank window that is not named
_parent	Loads the linked document into the immediate parent of the document the link is in
_search	Loads the linked document into the browser's search pane. Available in Internet Explorer 5 or later
_self	Loads the linked document into the same window the link was clicked in
_top	Loads the linked document into the topmost window

- We can discover the name of a window through its name property:  
window.name
- The name property belongs to the window object, and can be used with any window object, including a frame.
- To open an URL into another window and then close it:  
Mywin = window.open("http://www.hello.com/", "wind");  
Mywin.close();
- To set the URL of the current window:  
window.location.href = "http://www.hello.com/";

- To move, scroll, and resize a given window:  
`window.moveTo(X, Y)`  
`window.moveBy(X, Y)`  
`window.scrollTo(X, Y)`  
`window.scrollBy(X, Y)`  
`window.resizeTo(Width, Height)`  
`window.resizeBy(X, Y)`
- To write contents into a window:  
`mywin.document.open("text/html", "replace");`  
`mywin.document.write("<HTML><HEAD><TITLE>New Document</TITLE></HEAD><BODY>My Cramsession is here!</BODY></HTML>");`  
`mywin.document.close();`

### Form Validation Event Handlers

- Form validation with Javascript is efficient, as validation takes place on the client machine (so there's no delay for contacting a remote server).
- Form validation is accomplished by integrating form event handlers and other program flow structures, such as if else statements, for loop...etc.

Event handlers for forms	
<b>Onblur:</b>	Executes JavaScript whenever a user, using the mouse, moves the focus away from a form element. You use it to check each element individually and ask the user to fix the invalid input before moving on.
<b>Onsubmit:</b>	Executes JavaScript whenever someone clicks the "submit" button. No validation takes place until the user submits the form.

- Sample code fragment for password field validation:

```
<script Language="JavaScript">
<!--
function Form1_Valid(theForm)
{

    // check to see if the field is blank
    if (theForm.Alias.value == "")
    {
        alert("You must enter an alias.");
        theForm.Alias.focus();
        return (false);
    }

    // require at least 5 characters be entered
    if (theForm.Alias.value.length < 5)
    {
        alert("Please enter a minimum of 5 characters in the \"Alias\" field.");
        theForm.Alias.focus();
        return (false);
    }

    // check if both password fields are identical
    if (theForm.Password.value != theForm.Password2.value)
    {
        alert("The two passwords are different.");
        theForm.Password2.focus();
        return (false);
    }

}

//--></script>
```

- You will need to specify the form action as the JavaScript:

```
<form action="javascript.asp" method="POST" onsubmit="return
Form1_Valid(this)" name="Form1">
```

Please visit the following link for more form validation samples:

<http://developer.netscape.com/docs/examples/javascript/formval/overview.html>

## Cookies

- "Cookie" is a mechanism developed by Netscape Corporation to make up for the stateless nature of the HTTP protocol. It is a small piece of information that the HTTP server sends to the browser when the browser connects for the first time. Thereafter, the browser returns a copy of the cookie to the server each time it connects. The server uses the cookie to remember the status of the user.
- Cookies are text files saved in clients' local machines.
- Cookies cannot retrieve information about your computer system. They are only used to store information that you have provided at some point during a web session.

For a code fragment used to create cookies, please visit:

<http://ourworld.compuserve.com/homepages/schaper/cookie.htm>

- Some people see cookies as the security threat: a user can instruct the browser not to accept cookies at all.
- In theory, the same cookie can be read and set only by the same server.
- The difference between Netscape and MS implementation of cookies: MSIE limits you to one cookie per domain, whereas Netscape allows up to 20 pairs per domain.

## Security Issues and Bugs

- To a limited extent, JavaScript contains security holes.
- Many of the holes have been caught and fixed, but new ones come up all the time.
- Visit [Netscape's](http://www.netscape.com) web site to look for fixes or patches.
- Those items perceived to be security risks require "signed JavaScript". Originally, Javascript is not designed to read or write random text files on the local disk or on the server, invoke automatic printing of the current document, control browser e-mail, news reader, or bookmark windows and menus, access or modify browser preferences settings, capture a visitor's e-mail address or IP address, quietly send e-mail when a visitor loads a page, launch client processes, capture individual keystrokes, change the current browser window size, location, or options ...etc. However, many of these items are now possible in Communicator 4.X.
- One long-standing bug with JavaScript and tables is worth noting. It is recommended that you do not place <SCRIPT> tags inside <TD> tags. Instead, start the <SCRIPT> tag before the <TD> tag, and document.write() the <TD> tag through the </TD> tag.

Visit the official Bug List page at:

<http://developer.netscape.com/tech/javascript/index.html?content=/support/bugs/known/javascript.html>

Special Thanks to [Michael Yu](#) for contributing material for this Cramsession. Make

sure to visit his site at: <http://michaelyu.freesevers.com>

brainbuzz.com

