onclick="document.getElementById('demo').innerHTML = 'test' "

this will change the text within the demo ID to 'test'

onclick="document.getElementById('demo').style.fontSize='35px'"

this will change the font size of the text in the demo ID to 35px

document.getElementById("demo").style.display = "none";

document.getElementById("demo").style.display = "block";

document.write

window.alert(alertcontent)

console.log(logcontent)

you can declare variables in script tags

|  |  |
| --- | --- |
| **Keyword** | **Description** |
| break | Terminates a switch or a loop |
| continue | Jumps out of a loop and starts at the top |
| debugger | Stops the execution of JavaScript, and calls (if available) the debugging function |
| do ... while | Executes a block of statements, and repeats the block, while a condition is true |
| for | Marks a block of statements to be executed, as long as a condition is true |
| function | Declares a function |
| if ... else | Marks a block of statements to be executed, depending on a condition |
| return | Exits a function |
| switch | Marks a block of statements to be executed, depending on different cases |
| try ... catch | Implements error handling to a block of statements |
| var | Declares a variable |

**String methods**

Str.length returns length of string

Str.indexOf("index") returns the start index of the first occurrence of the specified string

Str.indexOf("index",4) starts search after specified index number

Str.lastIndexOf("index") returns the start index of the last occurrence of the specified string

Str.search("index") same as indexOf, but cannot take a start position. Can take regular expressions

Str.slice(x,y) returns string >= x and <y. negative values count from the end instead

Str.slice(x) returns all of string after x

Str.substring(x,y) same as slice, cannot accept negative values

Str.substr(x,y) slice, but returns a string of length y starting at index x

Str.replace("replace with","this") returns a string, replacing a first instance of a string with another

Str.toUpperCase() returns the string in all uppercase

Str.toLowerCase() same, lowercase

Str1.concat(str2, str3) join 2 or more strings. This equals str1 + str2 + str3

Str.charAt(x) returns character present at position x

Str.charCodeAt(x) returns Unicode character present at position x

Str.split(",") converts string to array. Array items are separated at the given string

**Number methods**

num.toString() converts to a string

num.toExponential(x) returns exponential number with x number of characters after the decimal point

num.toFixed(x) returns number with x number of decimal places

num.toPrecision(x) returns string with x number of digits

number(str) returns a number pulled from the string

parseInt(str) same, but whole numbers only

parseFloat(str) similar to number()

**array methods**

arr.join(" \* ") returns a string of each array item concatenated with the given string

arr.pop() removes the last item in an array. Also returns the value that was removed

arr.push(str) adds that string to the end of the array. Returns new length of the array

arr.shift() same as pop, but removes the first list item. Returns the removed item

arr.unshift(str) adds the string to the beginning of the array. Returns new length of array

arr[x] = str updates the item at array index x to equal str

delete arr[x] changes the element at position x to be undefined

arr.splice(x,y,str1,str2) add new strings into an array at index x while removing y elements

arr.slice(x,y) same as slice, returns an array, not a string

arr1.concat(arr2) returns an array of 2 or more merged arrays

**sorting arrays**

arr.sort() sorts elements by alphabetical order

arr.reverse() reverse order sort

**sorting numerically** – because a sort is alphabetical and not numeric, a workaround is necessary. By comparing two values to see which one is numerically higher, a numerical sort can be obtained

arr.sort(function(a, b){return a – b}) returns a numerically sorted array

arr.sort(function(a, b){return a + b}) returns a numerically descending sorted array

**date**

var d = new Date(year, month, day, house, minute, second, millisecond) creates a variable as a date

no input creates a date/time form the current date/time

date.getTime() returns ms between date and jan 1, 1970

date.getFullYear() returns year

date.getMonth() returns month (0-11)

date.getDate() returns day of month (1-31)

date.getHours() returns hour (0-23)

date.getMinutes() returns minutes (0-59)

date.getSeconds() returns seconds (0-59)

date.getMilliseconds() returns milliseconds (0-999)

date.getDay() returns day of week (0-6)

alternate versions of these methods exist. Set instead of get

date.setFullYear(1999) sets the year of the date variable

…

Date.setDate() adds days of the week to the date string

Math.abs(x) Returns the absolute value of x

Math.acos(x) Returns the arccosine of x, in radians

Math.asin(x) Returns the arcsine of x, in radians

Math.atan(x) Returns the arctangent of x as a numeric value between -PI/2 and PI/2 radians

Math.atan2(y, x) Returns the arctangent of the quotient of its arguments

Math.ceil(x) Returns the value of x rounded up to its nearest integer

Math.cos(x) Returns the cosine of x (x is in radians)

Math.exp(x) Returns the value of Ex

Math.floor(x) Returns the value of x rounded down to its nearest integer

Math.log(x) Returns the natural logarithm (base E) of x

Math.max(x, y, z, ..., n) Returns the number with the highest value

Math.min(x, y, z, ..., n) Returns the number with the lowest value

Math.pow(x, y) Returns the value of x to the power of y

Math.random() Returns a random number between 0 and 1

Math.round(x) Returns the value of x rounded to its nearest integer

Math.sin(x) Returns the sine of x (x is in radians)

Math.sqrt(x) Returns the square root of x

Math.tan(x) Returns the tangent of an angle

Math.E // returns Euler's number

Math.PI // returns PI

Math.SQRT2 // returns the square root of 2

Math.SQRT1\_2 // returns the square root of 1/2

Math.LN2 // returns the natural logarithm of 2

Math.LN10 // returns the natural logarithm of 10

Math.LOG2E // returns base 2 logarithm of E

Math.LOG10E // returns base 10 logarithm of E

Math.random() random number between 0 and 1

Math.floor(Math.random() \* 10) random number between 0 and 9

Var variable1 = (conditional statement) ? "if true" : "if false"

If (true) {

} else if {

} else {

}

Switch(x){

Case 1:

Break;

Case 2:

Break;

Default:

Break;

}

For(variable;conditional;incremental){}

For (x in y){ }

While(conditional){ }

Do{ } while (condition)

Break exit current loop of switch statement

Continue move onto the next iteration of loop

Codeblock{

Break codeblock; this will break out of this block of code and continue after it ends

}

Typeof x returns the type of x

x.constructor returns the constructor function

string(x) converts x to a string

x.toString() same

var y = "5"  
var x = + y x is a string, x is y converted into a number

Number(true) returns 1

Number(false) returns 0

**Bitwise** – there are some operators that effect variables on a bit-level

|  |  |  |
| --- | --- | --- |
| & | AND | Sets each bit to 1 if both bits are 1 |
| | | OR | Sets each bit to 1 if one of two bits is 1 |
| ^ | XOR | Sets each bit to 1 if only one of two bits is 1 |
| ~ | NOT | Inverts all the bits |
| << | Zero fill left shift | Shifts left by pushing zeros in from the right and let the leftmost bits fall off |
| >> | Signed right shift | Shifts right by pushing copies of the leftmost bit in from the left, and let the rightmost bits fall off |
| >>> | Zero fill right shift | Shifts right by pushing zeros in from the left, and let the rightmost bits fall off |

Regular expressions – an expression that forms a search pattern. instead of quoting a string, surround it by slashes (/) and follow it by a letter

/demo/i means case insensitive

/demo/g is a global match. Can be used to reference all instances of a string in another string, rather than the first instance (good for replace)

/demo/m multiline match

/[abc]/ find any characters between the brackets

/[0-9]/ find any digits between the brackets

/(example1|example2|alsothis)/ find any of the strings separated by |

\s whitespace character. This would be formatted as /\s/

\d digit character

\b find match at beginning or end of word

\uxxxx find Unicode character specified by hex number xxxx

Define quantities

n+ match any string that has at least one n

n\* match any string that has zero or more occurrences of n

n? match any string that has zero or one occurrences of n

/regular expression/.test("string") searches string for pattern and returns true or false

/regexp/.exec("string") returns the found text if regexp is found. Returns null otherwise  
example: /e/.exec("test") will return "e"

Try{  
 Try to run some code here  
}  
Catch(err){  
 If the code errors, this runs  
}  
finally{  
 This will run after try and catch, no matter the result  
}

In this case, err is an error. It has two properties: name and message

Throw error  
you can trigger an error manually with this, jumping ahead to any Catch  
the thrown error can be a string, number, bool, or object

Six different error values

|  |  |
| --- | --- |
| EvalError | An error has occurred in the eval() function |
| RangeError | A number "out of range" has occurred |
| ReferenceError | An illegal reference has occurred |
| SyntaxError | A syntax error has occurred |
| TypeError | A type error has occurred |
| URIError | An error in encodeURI() has occurred |

Range error – using a value outside of range of legal values

Reference error – referencing a variable that has not been declared

Syntax error – syntax incorrect

Type error – using a value outside the range of expected types

URI error – using illegal characters in a URI function

Debugging

Console.log(x) x gets logged into the browser console. Can be views in the console (F12)

Debugger; if the console is open, this will function as a break point

"use strict"; code will run in strict mode. Some mistakes will no longer be auto corrected. Bad syntax is no longer automatically caught. In normal JavaScript, a developer will not receive any error feedback assigning values to non-writable properties. In strict mode, any assignment to a non-writable property, a getter-only property, a non-existing property, a non-existing variable, or a non-existing object, will throw an error.

Example of an object

var person = {  
 firstName: "John",  
 lastName: "Doe",  
 age: 50,  
 eyeColor: "blue"  
};

loading scripts

<script src="myscript.js"></script>

Scripts run as they appear on the html file. It would be best that scripts run after the page has fully loaded. Put scripts at the end  
or, put **defer="true"** in in the script tags  
or, have a function occur on load **window.onload = function() { };**

**JSON**

{

"object":[

{"name1":"value1","name2":"value2","name3":"value3"}

{"name1":"value1","name2":"value2","name3":"value3"}

{"name1":"value1","name2":"value2","name3":"value3"}

]

}

This array contains three objects

{"name1":"value1","name2":"value2","name3":"value3"}

This is just an object and is also valid