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

JSON.parse(text) returns the object/array that exists in the JSON string

JSON.stringify(object/array) returns a JSON string version of the object. Can be written or interpreted later

localStorage.setItem("name of item being stored",JSONstring) this will store the JSON string. In this case it is stored in the local storage

localStorage.getItem("name of item being stored") this will return a JSON format string by retrieving the JSON string from storage, in this case local storage

forms

HTML forms can have execute functions.

onsubmit="return validateForm()" this will return either true or false, allowing or disallowing the form to be submitted

required adding required to an input field will ensure that the user cannot submit unless there is something in that field

disabled this input field will be disabled

max sets a maximum value allowed

min sets min value allowed

objects

properties can be deleted

delete object.property this property and its value no longer exist

property attributes

value the value of the property (ex: age value is 45)

enumerable

configurable

writable is the property writable?

var person = {

firstName: "John",

lastName : "Doe",

id : 5566,

fullName : function() {

return this.firstName + " " + this.lastName;

}

};

this used to reference the object's self

accessing a property that stores a function without () will return the function definition. This can be used to redefine the function or to add a new function

constructing an object

// Constructor function for Person objects

function Person(first, last, age, eye) {

this.firstName = first;

this.lastName = last;

this.age = age;

this.eyeColor = eye;

}

// Create a Person object

var myFather = new Person("John", "Doe", 50, "blue");

var x1 = {};            // new object  
var x2 = "";            // new primitive string  
var x3 = 0;             // new primitive number  
var x4 = false;         // new primitive boolean  
var x5 = [];            // new array object  
var x6 = /()/           // new regexp object  
var x7 = function(){};  // new function object

**object prototypes**

object.prototype can be used to reference the constructor to add functions or properties to a constructor

function myFunction(a, b) {  
    return a \* b;  
}

this is the same as this

var x = function (a, b) {return a \* b};

a function surrounded by parentheses and ending in closed parentheses will invoke itself

(function () {})();

arguments.length returns the number of arguments in the current function

more arguments can be passed into a function than are allowed, parameter names will just not exist, the arguments can be accessed through the arguments keyword

function findMax() {

var i;

var max = -Infinity;

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

if (arguments[i] > max) {

max = arguments[i];

}

}

return max;

}

document.getElementById("demo").innerHTML = findMax(4, 5, 6);

object1.function.call(object2) the function from object 1 is called on object 2

object1.function.apply(object2, array) this takes arrays in when being called. if object2 is null or a non object, this still works

**Closures**

-when a function that calls itself returns something, it is replaced with that thing

var example = (function () {

var counter = 0;

return 4;

})();

-after running once, **example == 4** instead of the entire function. it is no longer a function and is instead a variable

var add = (function () {

var counter = 0;

return function () {return 4;}

})();

-after running once, **example == function () {return 4;}.** calling the function will return 4. calling the variable will return the function

var add = (function () {

var counter = 0;

return function () {return counter += 1;}

})();

-after running once, **example == function () {return counter += 1;}**. each subsequent time the example function is run the counter will increment and return the new number. This way, the original counter can be defined as 0, keep a local scope, and no longer be changed again.

HTML DOM (document object model)

find HTML elements

document.getElementById(id) Find an element by element id

document.getElementsByTagName(name) Find elements by tag name

document.getElementsByClassName(name) Find elements by class name

changing elements

element.innerHTML = new html content Change the inner HTML of an element

element.attribute = new value Change the attribute value of an HTML element

element.setAttribute(attribute, value) Change the attribute value of an HTML element

element.style.property = new style Change the style of an HTML element

adding and deleting elements

document.createElement(element) Create an HTML element

document.removeChild(element) Remove an HTML element

document.appendChild(element) Add an HTML element

document.replaceChild(element) Replace an HTML element

document.write(text) Write into the HTML output stream

add event handlers

document.getElementById(id).onclick = function(){code} Adding event handler code to an onclick event

find objects

document.anchors Returns all <a> elements that have a name attribute 1

document.applets Returns all <applet> elements (Deprecated in HTML5) 1

document.baseURI Returns the absolute base URI of the document 3

document.body Returns the <body> element 1

document.cookie Returns the document's cookie 1

document.doctype Returns the document's doctype 3

document.documentElement Returns the <html> element 3

document.documentMode Returns the mode used by the browser 3

document.documentURI Returns the URI of the document 3

document.domain Returns the domain name of the document server 1

document.domConfig Obsolete. Returns the DOM configuration 3

document.embeds Returns all <embed> elements 3

document.forms Returns all <form> elements 1

document.head Returns the <head> element 3

document.images Returns all <img> elements 1

document.implementation Returns the DOM implementation 3

document.inputEncoding Returns the document's encoding (character set) 3

document.lastModified Returns the date and time the document was updated 3

document.links Returns all <area> and <a> elements that have a href attribute 1

document.readyState Returns the (loading) status of the document 3

document.referrer Returns the URI of the referrer (the linking document) 1

document.scripts Returns all <script> elements 3

document.strictErrorChecking Returns if error checking is enforced 3

document.title Returns the <title> element 1

document.URL Returns the complete URL of the document

animations using javascript

var id = setInterval(frame, 5);

function frame() {

if (/\* test for finished \*/) {

clearInterval(id);

} else {

increment here

/\* code to change the element style \*/

}

}

here, the var id is created. an interval is set to 5 frames. upon each interval, frame runs. if the conditions for completion are not yet met, increment and run whatever code is needed. when conditions are met, clearInterval

this will replace any onclick events with the provided event

document.getElementById("myBtn").onclick = displayDate;

this will add the event, rather than replace

document.getElementById("myBtn").addEventListener("click", displayDate);

to stop bubbling…

onclick="event.stopPropagation()

event.stopImmediatePropagation()

remove event listeners

element.removeEventListener("mousemove", myFunction);

passing parameters

element.addEventListener("click", function(){ myFunction(p1, p2); });

when passing parameters, call an anonymous function that calls the desired function. why?

document.body - The body of the document

document.documentElement - The full document

this script creates an element <p>, creates text, appends the text into the element, finds an existing element with id = "div1", and appends the new <p> into it

var para = document.createElement("p");

var node = document.createTextNode("This is new.");

para.appendChild(node);

var element = document.getElementById("div1");

element.appendChild(para);

this inserts element para into element element, before element child

element.insertBefore(para, child);

parent.removeChild(child);

parent.replaceChild(para, child);

**HTMLCollections**

this will return an HTMLCollection of all elements with the <p> tag

var x = document.getElementsByTagName("p");

this is not an array, though is seems like it is

**NodeList**

list of nodes extracted from the document

this returns a list of all <p> nodes in the document

var myNodeList = document.querySelectorAll("p");

the difference is that html collections items can be accessed by their names, ids, and index numbers while NodeLists can only be accessed through index numbers

however, NodeLists can contain attribute and text nodes as well

**JS Browser BOM (Browser Object Model)**

used for interacting with the browser itself

window.open() - open a new window

window.close() - close the current window

window.moveTo() -move the current window

window.resizeTo() -resize the current window

var w = window.innerWidth  
|| document.documentElement.clientWidth  
|| document.body.clientWidth;  
  
var h = window.innerHeight  
|| document.documentElement.clientHeight  
|| document.body.clientHeight;

screen.width -returns width of the screen in pixels

screen.height -returns height of the screen in pixels

screen.availWidth -returns available screen width (subtracts interface features like the taskbar)

screen.availHeight -returns available screen height

screen.colorDepth -returns number of bits used by one color

screen.pixelDepth -returns pixel depth, same as color depth

window.location.href returns the href (URL) of the current page

window.location.hostname returns the domain name of the web host

window.location.pathname returns the path and filename of the current page

window.location.protocol returns the web protocol used (http: or https:)

window.location.assign(new page) loads a new document

history.back() - same as clicking back in the browser

history.forward() - same as clicking forward in the browser

navigator this object contains information about the users browser

navigator.cookieEnabled returns true if cookies are enabled

**the following results are usually misleading**

navigator.appName returns application name of the browser

navigator.appCodeName returns application codename for the browser

navigator.product returns product name of the browser engine

navigator.appVersion returns version information about the browser

navigator.userAgent returns user-agent header sent by the browser to the server

navigator.platform returns operating system

navigator.language returns browsers language

navigator.online returns true if the browser is online

navigator.javaEnabled() returns true if java is enabled

popup alerts

alert("I am an alert box!");

if (confirm("Press a button!")) {  
    txt = "You pressed OK!";  
} else {  
    txt = "You pressed Cancel!";  
}

var person = prompt("Please enter your name", "default");  
  
if (person == null || person == "") {  
    txt = "User cancelled the prompt.";  
} else {  
    txt = "Hello " + person + "! How are you today?";  
}

timers

myVar = setTimeout(function, milliseconds); the function is called after the time counts to 0

clearTimeout(myVar); the timer is stopped

myVar = setInterval(function, milliseconds); function is called every interval of time

clearInterval(myVar); interval timer is stopped

**Cookies**

document.cookie = "username=John Doe; expires=Thu, 18 Dec 2013 12:00:00 UTC";

when the time entered is reached, the cookie expires. default expiration is when the browser is closed

document.cookie = "username=John Doe; expires=Thu, 18 Dec 2013 12:00:00 UTC; path=/";

path determines where the cookie belongs. default is the current page

var x = document.cookie read the cookies

overwite a cookie by creating a new cookie over an old one

document.cookie = "username=John Smith; expires=Thu, 18 Dec 2013 12:00:00 UTC; path=/";

delete a cookie by creating an empty cookie over an old one and an already expired date

document.cookie = "username=; expires=Thu, 01 Jan 1970 00:00:00 UTC; path=/;";

this returns the decoded cookie. difference is that this can handle special characters

decodeURIComponent(document.cookie)

in order to parse a cookie, decode the cookie, split is into an array, search each array item for the cookie being searched for, pull out the substring that contains the cookie value and return it

function getCookie(cname) {  
    var name = cname + "=";  
    var decodedCookie = decodeURIComponent(document.cookie);  
    var ca = decodedCookie.split(';');  
    for(var i = 0; i <ca.length; i++) {  
        var c = ca[i];  
        while (c.charAt(0) == ' ') {  
            c = c.substring(1);  
        }  
        if (c.indexOf(name) == 0) {  
            return c.substring(name.length, c.length);  
        }  
    }  
    return "";  
}

**AJAX**

exchange data behind the scenes with the web server. can be used to update parts of the page without reloading

XMLHttpRequest this object is the most important part. used to exchange data with the server

*variable*= new XMLHttpRequest();

this following function will load a txt file and populate an element with its contents

first, the XMLHttpRequest is created. then the request is specified (open). then it sends the request to the server (send). each time the ready state changes, a function runs. When the request is finished and the response is ready (4) and the status is returned as ok (200), response data as a string is inserted into the HTML element.

function loadDoc() {

var xhttp = new XMLHttpRequest();

xhttp.onreadystatechange = function() {

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

document.getElementById("demo").innerHTML = this.responseText;

}

};

xhttp.open("GET", "ajax\_info.txt", true);

xhttp.send();

}

|  |  |
| --- | --- |
| new XMLHttpRequest() | Creates a new XMLHttpRequest object |
| abort() | Cancels the current request |
| getAllResponseHeaders() | Returns header information |
| getResponseHeader() | Returns specific header information |
| open(*method, url, async, user, psw*) | Specifies the request  *method*: the request type GET or POST *url*: the file location *async*: true (asynchronous) or false (synchronous) *user*: optional user name *psw*: optional password |
| send() | Sends the request to the server Used for GET requests |
| send(*string*) | Sends the request to the server. Used for POST requests |
| setRequestHeader() | Adds a label/value pair to the header to be sent |

|  |  |
| --- | --- |
| onreadystatechange | Defines a function to be called when the readyState property changes |
| readyState | Holds the status of the XMLHttpRequest. 0: request not initialized  1: server connection established 2: request received  3: processing request  4: request finished and response is ready |
| responseText | Returns the response data as a string |
| responseXML | Returns the response data as XML data |
| status | Returns the status-number of a request 200: "OK" 403: "Forbidden" 404: "Not Found" For a complete list go to the [Http Messages Reference](https://www.w3schools.com/tags/ref_httpmessages.asp) |
| statusText | Returns the status-text (e.g. "OK" or "Not Found") |