JAVASCRIPT

JavaScript is a high-level programming language that is one of the core technologies of the World Wide Web. It is used as a client-side programming language by over almost 95% of the websites. The programming language was originally used only to develop web browsers but is now used for server-side website deployments and non-web browser applications as well.

<script></>

or <script src="index.js" > </>

document.getElementById(" id" ).innerText= text --------- could change text

let c = 0 ----- declaration of variable

console.log(c) ------- is used for debugging , will give value stored in c

Date() - -display date and time

//////////////////The console. log() is a function in JavaScript which is used to print any kind of variables defined before in it or to just print any message that needs to be displayed to the user./////////

let a= 4 let b= 5 let c= a+b so console.log(c) --- will give 9

clt+ k+ c ----- to comment out multiple lines

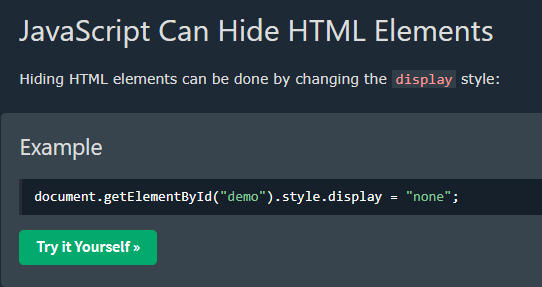
function name() {

} Variables - -global scope and defined scope

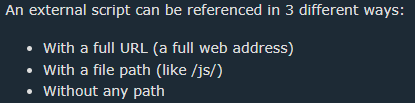
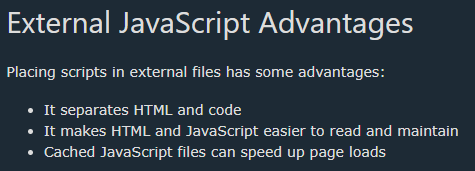
onclick=’ function’ ------------- in html

also var.innerText = text ------------------ to change data through id

functionname() ----------------- to run a function



External reference of js





Example –

Let username = “aadarsh “ ----------- string stored in variable . U can add strings also.

Let message = “ you have three notifications “

Console.log( message + “ , “ +username )



If let a = “10” -------- here 10 will be stored as a string not integer

And let a =10 --------here as integer

Textcontent and innerText and innerHTML

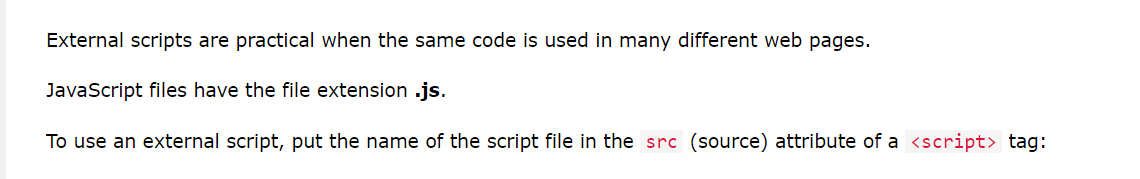
Javascript can change Html content and Images also



It can also change CSS

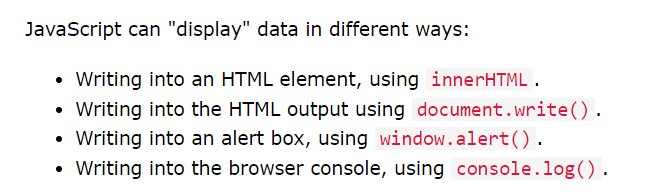


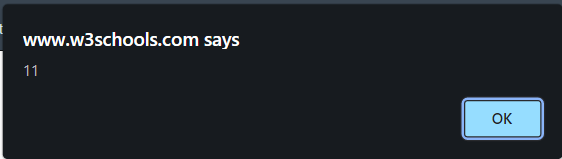
Scripts can be placed in the <body>, or in the <head> section of an HTML page, or in both.





Cached JavaScript files can speed up page loads – so having external link to file increases speed





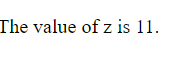
Document.write(“hello”)--------- prints hello

This is with window.alert() ; or simply alert();

Window.print() -------------- provides printing displayed page option

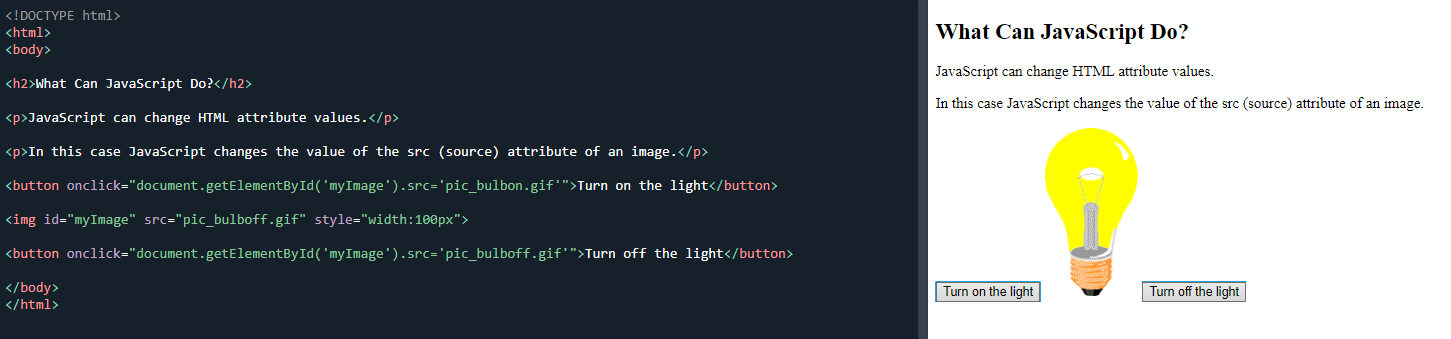
document.getElementById("demo").innerHTML =

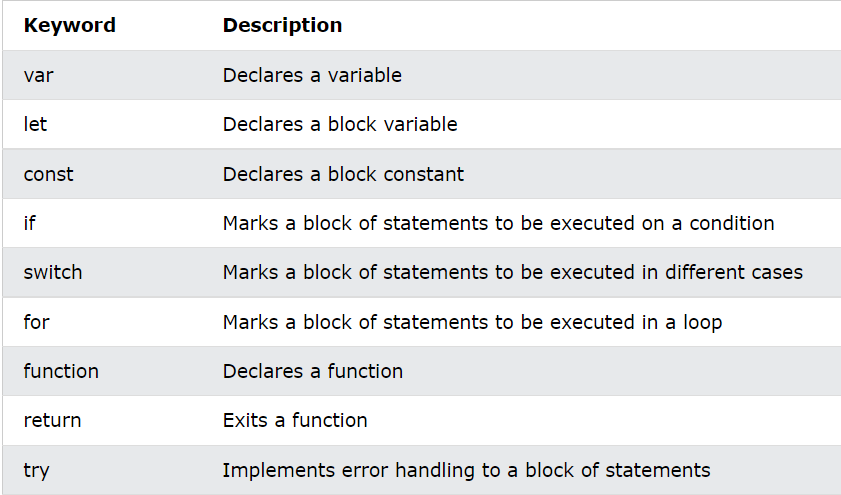
"The value of z is " + z + ".";

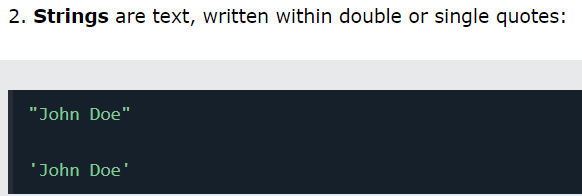


 ------------ onclick= ‘’ single inverted commas for javascript actions

JavaScript ignores multiple spaces. You can add white space to your script to make it more readable.



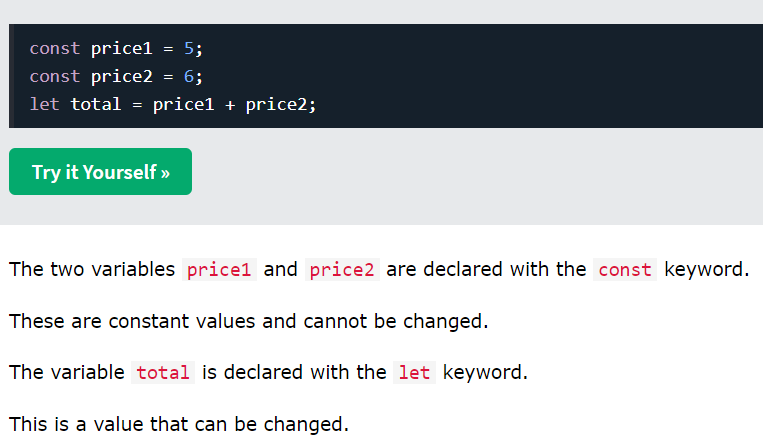






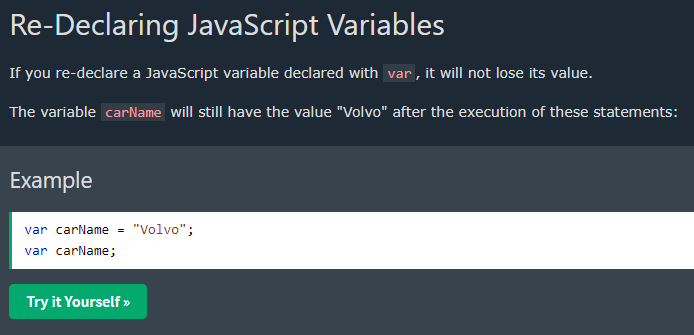




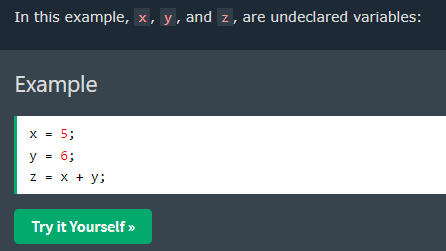
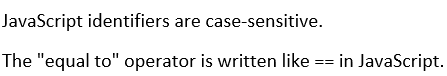


You can not redeclare a variable with let but with var u can.

let allows you to declare variables that are limited to the scope of a block statement, or expression on which it is used, unlike the var keyword, which declares a variable globally, or locally to an entire function regardless of block scope.

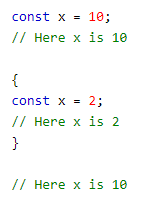
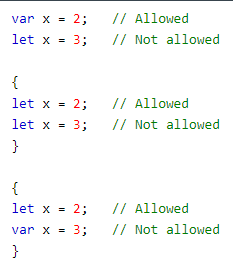


Can also happen with undeclared variables

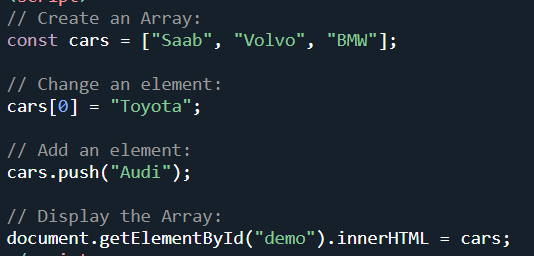




You cannot re-declare a variable declared with let or const.



Variables defined with var are hoisted to the top and can be initialized at any time. Meaning: You can use the variable before it is declared:



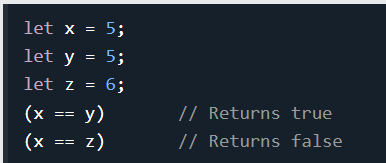
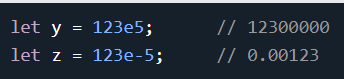
To add element array.push(“”)



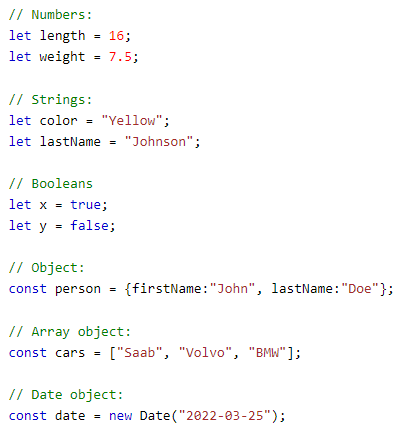
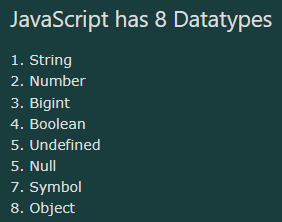
All operators <https://www.w3schools.com/js/js_arithmetic.asp>



since the first operand is a string, all operands are treated as strings



Boolean returns values in T and F



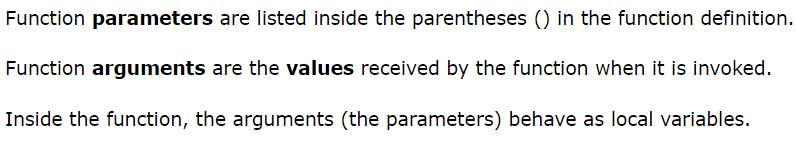
[ “” , “” ,] -------arrays and { property:” “ } ----------- objects

“<br>” ------ for net line in JS

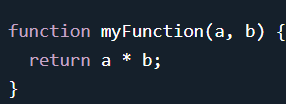
Typeof variable ------- to know datatype

A JavaScript function is defined with the function keyword, followed by a name, followed by parentheses ().

Function names can contain letters, digits, underscores, and dollar signs (same rules as variables).The parentheses may include parameter names separated by commas: (parameter1, parameter2, ...)

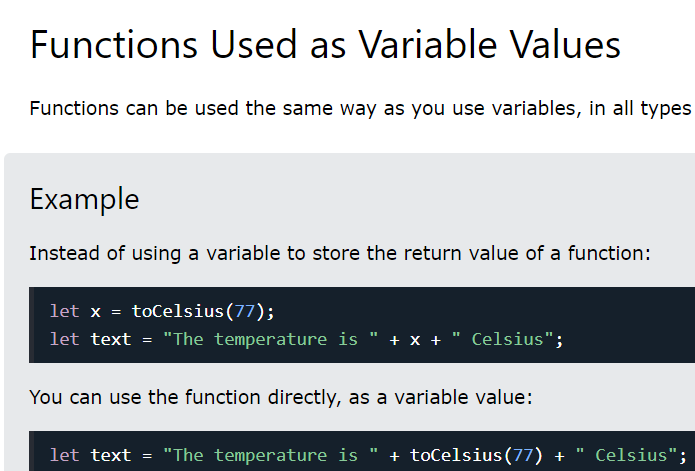


When JavaScript reaches a return statement, the function will stop executing.

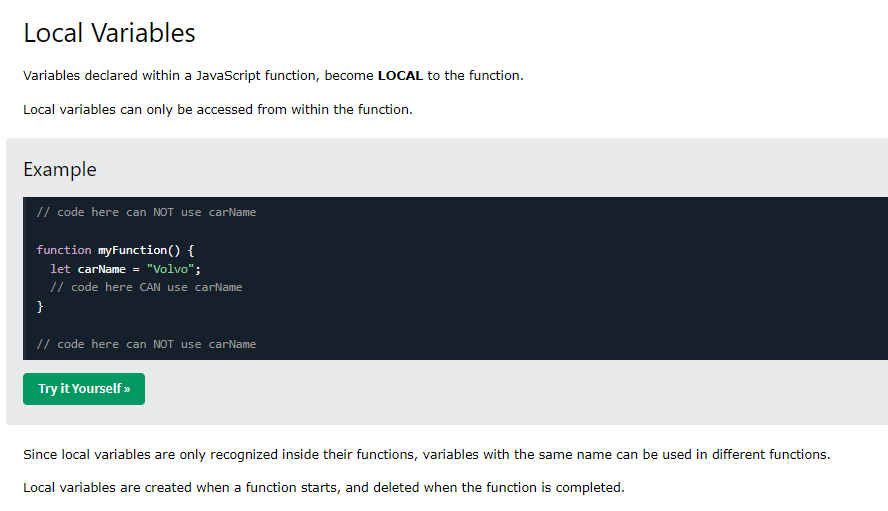


Example - <https://www.w3schools.com/js/tryit.asp?filename=tryjs_farenheit_to_celsius>

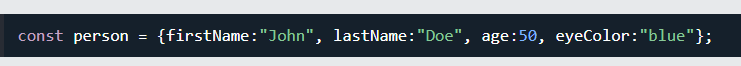




This is cool and NOTE Below also very important





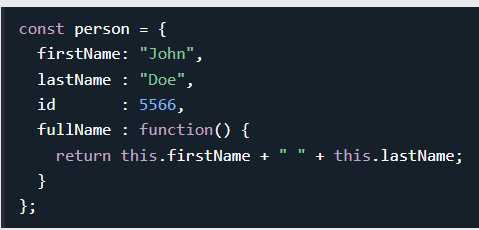


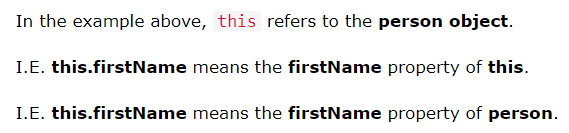
Creating object and giving them properties. It is a common practice to declare objects with the const keyword.

document.getElementById("demo").innerHTML = person["firstName"] + " " + person["lastName"];

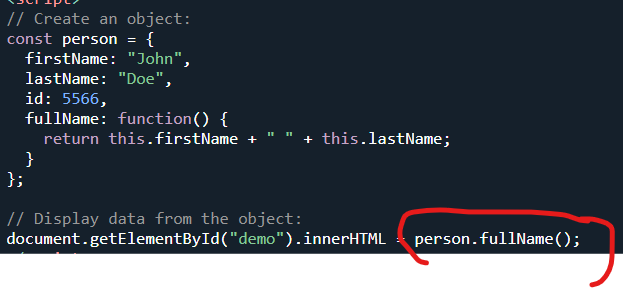
document.getElementById("demo").innerHTML = person.firstName + " " + person.lastName;

A method is a function stored as a property.





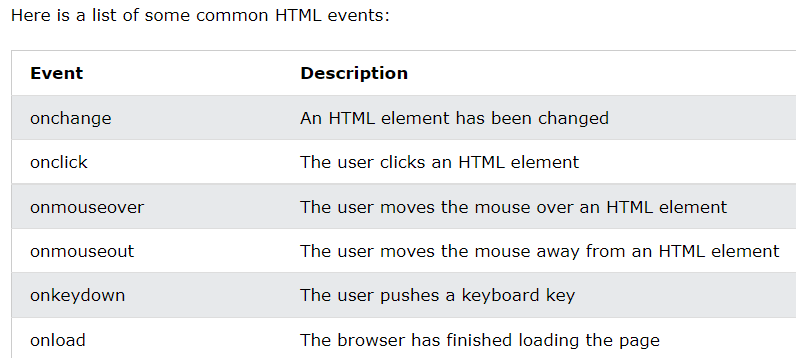
Do Not Declare Strings, Numbers, and Booleans as Objects! this is not a variable. It is a keyword. You cannot change the value of this.



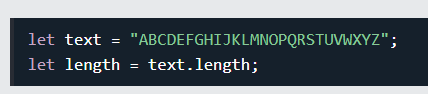
the code changes the content of its own element (using this.innerHTML):



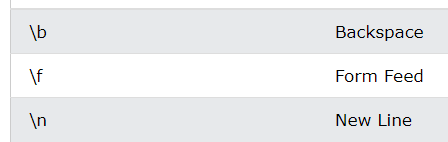
Avoid String, Number, and Boolean objects. They complicate your code and slow down execution speed.



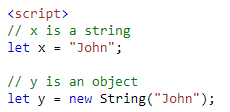
To find the length of a string, use the built-in length property:



The backslash (\) escape character turns special characters into string characters: 



JavaScript objects cannot be compared. Comparing two JavaScript objects always returns false Boolean between objects return false



Here y is a object and that how to declare string as object – let y = new String();

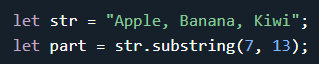
JavaScript String Methods



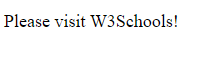
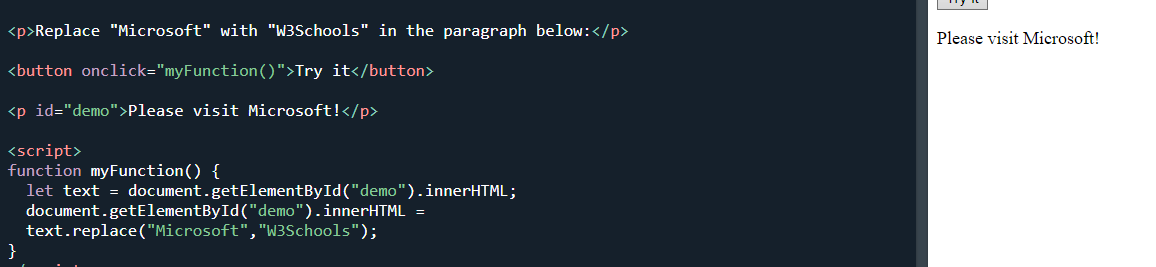
Extracting String Parts





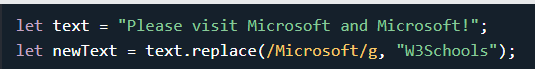
The difference is that start and end values less than 0 are treated as 0 in substring(). And substr() is similar to slice().

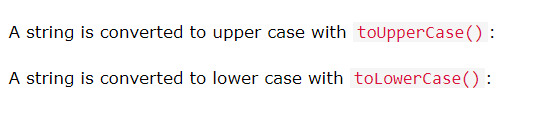
REPLACE() method on string



By default, the replace() method is case sensitive. Writing MICROSOFT (with upper-case) will not work:

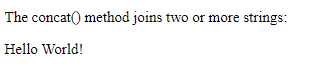
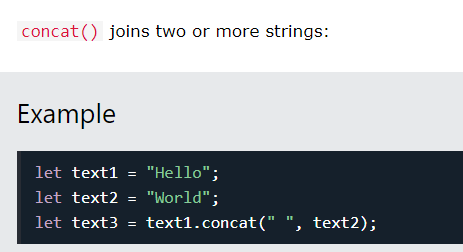
To replace all matches, use a regular expression with a /g flag (global match):





Trim() , trimstart /end padStart(no.of times , padding element) and padEnd(no.of times , padding element)

(https://www.w3schools.com/js/js\_string\_methods.asp)



Extracting String Characters

JavaScript String charAt()

JavaScript String charCodeAt()

Converting a String to an Array using JavaScript String split()

<https://www.w3schools.com/js/js_string_methods.asp>

I will be back

--------------------------------------------------------------------------------------------------------------------------

JavaScript String Search

String indexOf(), String lastIndexOf(), String search(), String match(), String matchAll()

String includes(), String startsWith(), String endsWith()







Both indexOf(), and lastIndexOf() return -1 if the text is not found:

let text = "Please locate where 'locate' occurs!";

text.indexOf("locate", 15); ------ 15 as a starting point



<https://www.w3schools.com/js/js_string_search.asp>

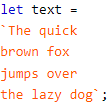
<https://www.w3schools.com/js/js_string_search.asp>

--------------------------------------------------------------------------------------------------------------------------

Back-Tics Syntax

Template Literals use back-ticks (``) rather than the quotes ("") to define a string:

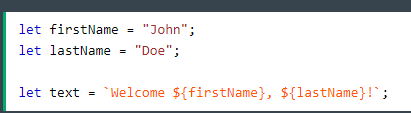
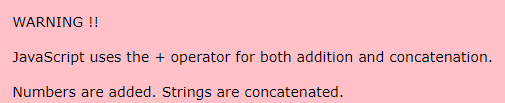


 Template Literals is not supported in Internet Explorer.

Template literals provide an easy way to interpolate variables and expressions into strings.

The method is called string interpolation. Using - ${...}

Variable Substitutions for example

If both the variable store the string numeric value then also all operation on it are possible giving output as integer value except the + one as it will concat if used. Except that all operators like - , \* , / will work normally despite numeric value being string. x/y = 10 , x-y =90, etc. But x+y = 10010

On the other hand let x = 100 / "Apple"; such a stupid calculation will give NaN (NaN - Not a Number as output)

Infinity (or -Infinity) is the value JavaScript will return if you calculate a number outside the largest possible number.Comparing two JavaScript objects always returns false.

--------- number object (Do not create Number objects.)

JavaScript interprets numeric constants as hexadecimal if they are preceded by 0x.

JavaScript BigInt variables are used to store big integer values that are too big to be represented by a normal JavaScript Number. JavaScript integers are only accurate up to 15 digits:

To create a BigInt, append n to the end of an integer or call BigInt():



Two values

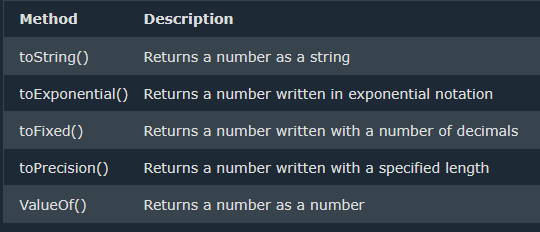
MAX\_SAFE\_INTEGER -🡪 9007199254740991

MIN\_SAFE\_INTEGER-🡪 - 9007199254740991

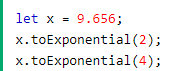
The Number.isInteger() method returns true if the argument is an integer.

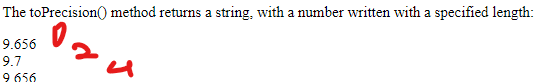
JavaScript Number Methods : These number methods can be used on all JavaScript numbers.







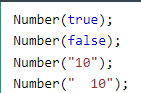


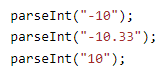
toFixed(2) is perfect for working with money. 



Converting Variables to Numbers



answer is 1 , 0 , 10 , 10  

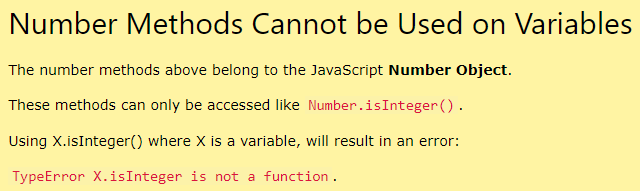

answer is 10, 10 , 10



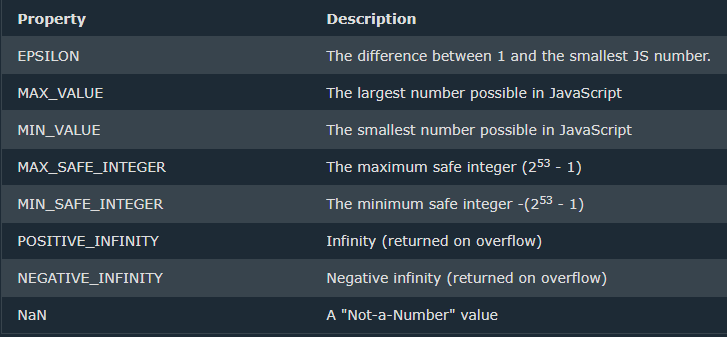
Number Object Methods

These object methods belong to the Number object:



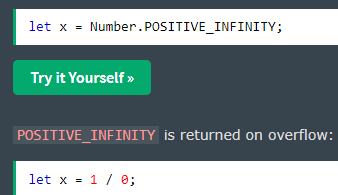
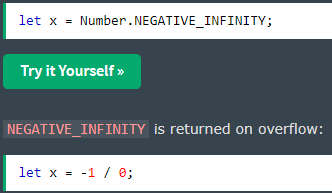
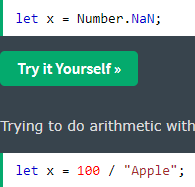


JavaScript Number Properties

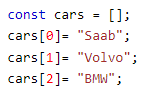
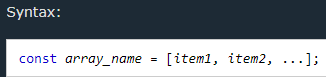


 will give value of e

will give max number possible in js similarly with MIN\_VALUEsimilarly with min safe.

 Infi -InfiNaN

Creating an Array



const cars = ["Saab", "Volvo", "BMW"];

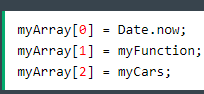
 using new also creates arrays

This statement changes the value of the first element in cars: cars[0] = "Opel";

Arrays are Objects

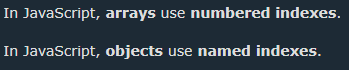
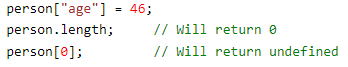
Arrays are a special type of objects. The typeof operator in JavaScript returns "object" for arrays.

Array Elements Can Be Objects

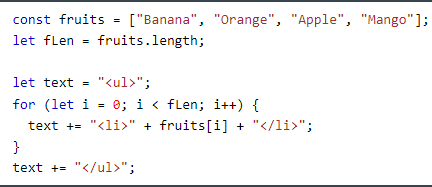
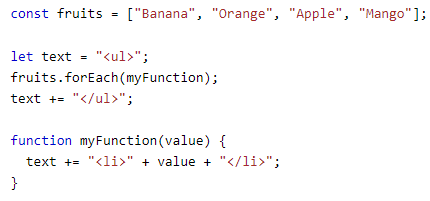
you can have variables of different types in the same Array. You can have objects in an Array. You can have functions in an Array. You can have arrays in an Array: 

Accessing the Last Array Element

If you use named indexes, JavaScript will produce incorrect results.



Looping Array Elements

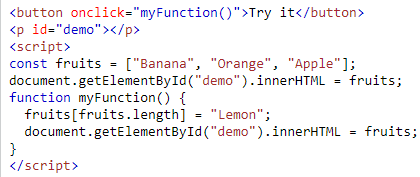
Using for loop and for each Array methods and making a unordered list.



Adding Array Elements



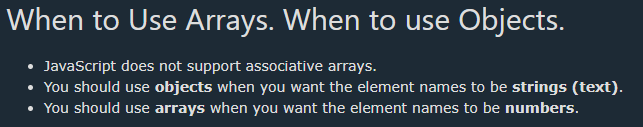
New element can also be added to an array using the length property:

cool keeps adding lemon with button click.

const fruits = ["Banana", "Orange", "Apple"];

fruits[6] = "Lemon";-----------this create element 3,4,5 as undefined and total 6 elements will be there . 3, 4,5 is stated as undefined.

U can store any damn thing in this array number , string , methods , objects,etc.

How to Recognize an Array----------- To solve this problem defined a new method Array.isArray(): --------will give true and false

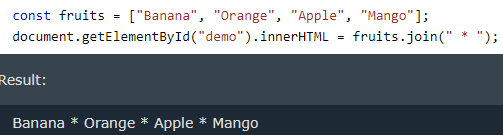
JavaScript Array Methods and Properties

<https://www.w3schools.com/jsref/jsref_obj_array.asp>

The JavaScript method toString() converts an array to a string of (comma separated) array values. Arrayname.toString();

The join() method also joins all array elements into a string.

It behaves just like toString(), but in addition you can specify the separator:



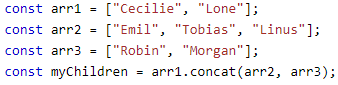
The pop() method removes the last element from an array: -------fruits.pop();

The push() method adds a new element to an array (at the end):------- fruits.push("Kiwi");

The shift() method removes the first array element and "shifts" all other elements to a lower index. -------- fruits.shift();

The unshift() method adds a new element to an array (at the beginning), and "unshifts" older elements:------ fruits.unshift("Lemon");

Array elements can be deleted using the JavaScript operator delete. Using delete leaves undefined holes in the array. Use pop() or shift() instead.-------- delete fruits[0];

The concat() method can take any number of array arguments: 

The splice() method can be used to add new items to an array: The first parameter (2) defines the position where new elements should be added (spliced in).The second parameter (0) defines how many elements should be removed.



Similarly splice() to remove elements without leaving "holes" in the array:  so O,A,M

The slice() method creates a new array and slice() method does not remove any elements from the source array.

JavaScript Sorting Arrays

The sort() method sorts an array alphabetically: & The reverse() method reverses the elements in an array.

Sort() will not work to sort out numeric value stored as it appears to stored as string so one can add a function to sorting for it to work

 Similarly for decending {return b-a}

To understand -

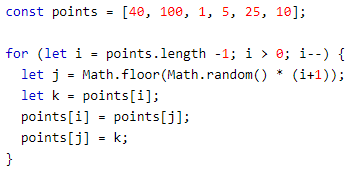
If the result is negative, a is sorted before b. If the result is positive, b is sorted before a.

If the result is 0, no changes are done with the sort order of the two values

Sorting an Array in Random Order

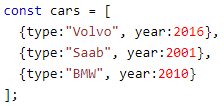


The Fisher Yates Method – for sorting array randomly (most effective)



You can use Math.max.apply(null, array); to find the highest number in an array , Similarly Math.min.apply(null, arr); for lowest

JavaScript arrays often contain objects:



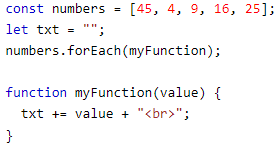
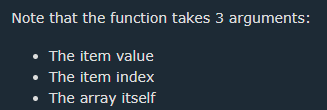
Example of sorting object array – here cars are sorted In ascending order( <https://www.w3schools.com/js/tryit.asp?filename=tryjs_array_sort_object1>)

Another solid example of sorting type section of objects alphabetically in array

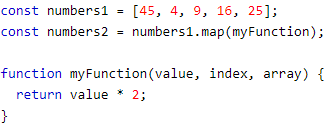
(<https://www.w3schools.com/js/tryit.asp?filename=tryjs_array_sort_object2>)

JavaScript Array forEach() - The forEach() method calls a function (a callback function) once for each array element.

To display all elements –

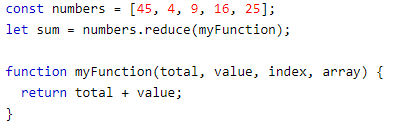


The map() method creates a new array by performing a function on each array element. The map() method does not execute the function for array elements without values.



The filter() method creates a new array with array elements that pass a test. ( eg - return value > 18) so numbers.filter( function\_name);

The reduce() method runs a function on each array element to produce (reduce it to) a single value. – basically sums up all elements

 Output – 99

here another parameter total is used. The reduce() method can also accept an initial value: let sum = numbers.reduce(myFunction, 100); so --------- sum will be 199

The every() method checks if all array values pass a test. ------ similar like filter but checks all array and gives true or false for array collectively.

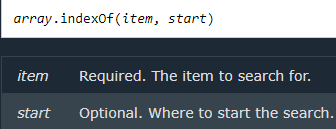
The some() method checks if some array values pass a test. ----- it is like every but for some. Checks if some pass the test. If yes then true.

IMP - The indexOf() method searches an array for an element value and returns its position.



Array.indexOf() returns -1 if the item is not found.

If the item is present more than once, it returns the position of the first occurrence.



The find() method returns the value of the first array element that passes a test function.

The findIndex() method returns the index of the first array element that passes a test function.

The Array.from() method returns an Array object from any object with a length property or any iterable object. ----------------- basically can create array of a string. const myArr = Array.from("ABCDEFG");--------- A,B,C,D,E,F,G

JavaScript Array Keys() & Array entries()------SIR KE UPER SE GAYA

JavaScript Array includes()- This allows us to check if an element is present in an array (including NaN, unlike indexOf).



An array declared with const cannot be reassigned, and it doesn’t mean Arrays are Constants, although Elements in it Can be Reassigned.

Also note JavaScript const variables must be assigned a value when they are declared.Meaning: An array declared with const must be initialized when it is declared.

An array declared with var does not have block scope

JavaScript Date Objects

const d = new Date(); ----------tells now date and time

const d = new Date("2022-03-25"); -------- mention specified date ( string) and now time



JavaScript counts months from 0 to 11: January = 0. December = 11.

Specifying a month higher than 11, will not result in an error but add the overflow to the next year.

The toDateString() method converts a date to a more readable format:

const d = new Date(); d.toDateString(); ///Mon Jan 23 2023

To study date in more detail – { https://www.w3schools.com/js/

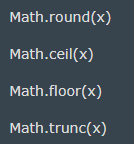
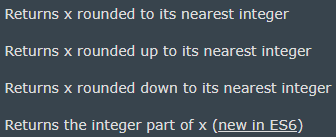
* js\_date\_formats.asp
* js\_date\_methods.asp
* js\_date\_methods\_set.asp

}

JavaScript Math Object

Unlike other objects, the Math object has no constructor. The Math object is static. All methods and properties can be used without creating a Math object first. The syntax for any Math property is : Math.property

 ----Output: 5 if its 4.4 then it becomes 4

 ------ Output: 5 only greater side

 --- Output: 4

Math.trunc(x) returns the integer part of x.

Math.sign(x) returns if x is negative, null or positive: 1 for +ve -1 for -ve

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

Math.sqrt(x) returns the square root of x. Also Math.sin(x), Math.cos().

Math.abs(x) returns the absolute (positive) value of x.

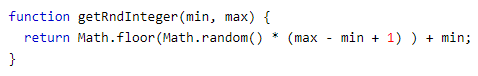
Math.min(0, 150, 30, 20, -8, -200); Math.min / max

Math.random() returns a random number between 0 (inclusive), and 1 (exclusive).

Reference list of maths object properties – (<https://www.w3schools.com/jsref/jsref_obj_math.asp>)

NOTE- Math.random() used with Math.floor() can be used to return random integers.

Math.floor(Math.random() \* 10); ----- returns a random number btw 0 - 9



Above function generates random number with min and max range inclusive of both( as +1 includes max also) .

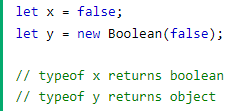
The Boolean() Function

You can use the Boolean() function to find out if an expression (or a variable) is true.

/// document.getElementById("demo").innerHTML = 10 > 9; this will also give true directly

/// Boolean(1 + 7 + 3.14) ------ true //Any expression (except zero) will be T

/// The Boolean value of "" (empty string) is false, The Boolean value of undefined is false, The Boolean value of NaN is false.

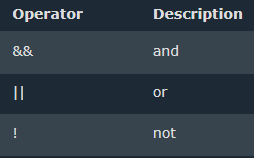
Difference between (x==y)- T and (x===y) – F.

Note - Comparing two JavaScript objects always return false.

Conditional (Ternary) Operator ---- variablename = (condition) ? value1:value2



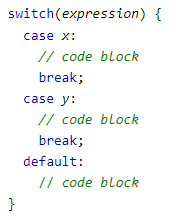
== equal to & === equal value and equal type

** LOGICAL Operators**

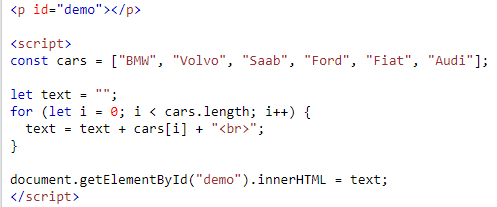
/// "2" > "12" ---------When comparing two strings, "2" will be greater than "12", because (alphabetically) 1 is less than 2.

If , If else, If else if else

The switch statement is used to perform different actions based on different conditions.

If multiple cases matches a case value, the first case is selected. If no matching cases are found, the program continues to the default label. If no default label is found, the program continues to the statement(s) after the switch.

The standard way of printing on html surface using text method at javascript is-

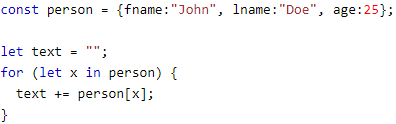


The for statement creates a loop with 3 optional expressions:

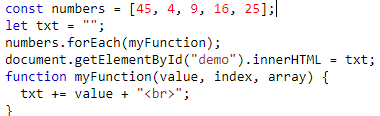
 you can use var or let to declare i within loop – it will have scope differences.

When let is used to declare the i variable in a loop, the i variable will only be visible within the loop.

The JavaScript for in statement loops through the properties of an Object/Array object, etc.

 John Doe 25

The forEach() method calls a function (a callback function) once for each array element.

 did before

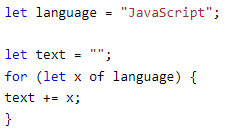
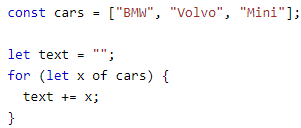
Operator precedence describes the order in which operations are performed in an arithmetic expression.

* Multiplication (\*) and division (/) have higher precedence than addition (+) and subtraction (-).
* Expressions in parentheses are computed before the rest of the expression
* Function are executed before the result is used in the rest of the expression

(*Source: js\_precedence.asp*)

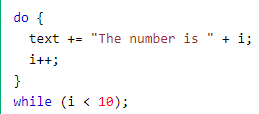
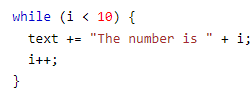
The JavaScript for of statement loops through the values of an iterable object.

It lets you loop over iterable data structures such as Arrays, Strings, Maps, NodeLists, and more. Similar like for in but it does it for all iteratable objects.



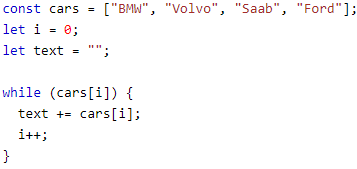
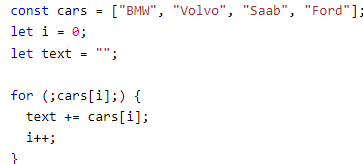
Loop over an array and a string

The while loop loops through a block of code as long as a specified condition is true.



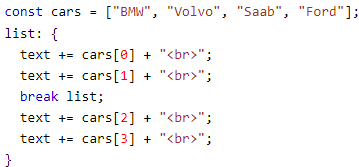
The do while loop is a variant of the while loop. This loop will execute the code block once, before checking if the condition is true, then it will repeat the loop as long as the condition is true.

To collect the car names from the cars array using for and while



* The break statement "jumps out" of a loop.
* The continue statement "jumps over" one iteration in the loop.

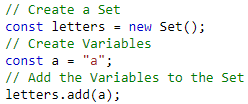
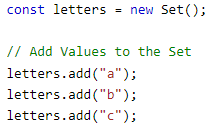
 The break and the continue statements are the only JavaScript statements that can "jump out of" a code block. With a label reference, the break statement can be used to jump out of any code block:

 only above 2 r printed: BMW & Volco

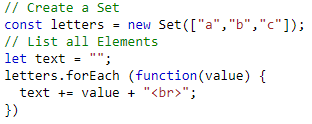
A JavaScript Set is a collection of unique values. Each value can only occur once in a Set. A Set has no keys. keys() returns the same as values()

Passing an Array to new Set(), Create a new Set and use add() to add values, Create a new Set and use add() to add variables

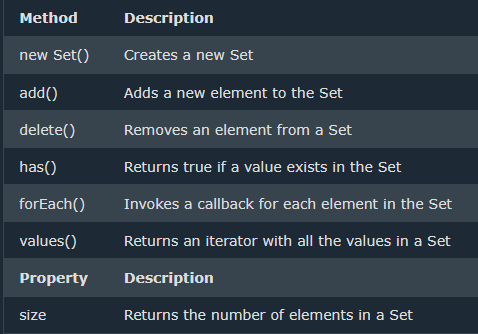
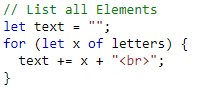


The add() Method – to add

If you add equal elements, only the first will be saved.

 forEach() in Set letters

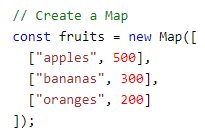
 -----value()

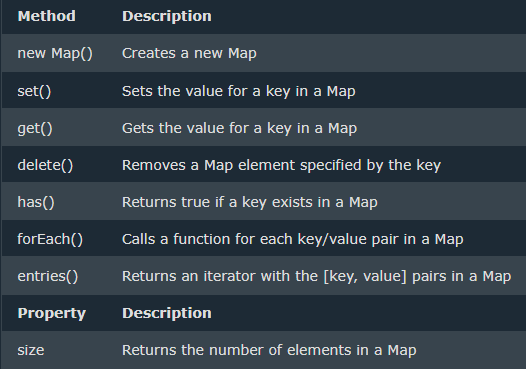


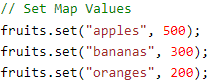
MAP

A Map holds key-value pairs where the keys can be any datatype.

A Map remembers the original insertion order of the keys.

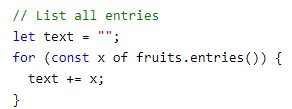
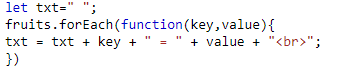
 Delete----



You can add elements to a Map with the set() method:  

The get() method gets the value of a key in a Map:

forEach method



The entries() method returns an iterator object with the [key, values] in a Map.

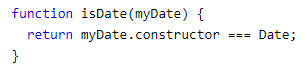
In JavaScript there are 5 different data types that can contain values:

String, number, Boolean – simple DT object, function --- complex DT

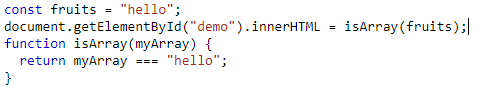
And 2 data types that cannot contain values: null & undefined

The constructor property returns the constructor function for all JavaScript variables. --------- tells about the DType.

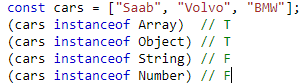


 like this u can check| T if its Date

A REALLY good example to understand === is –



In JavaScript null is "nothing". It is supposed to be something that doesn't exist.

Unfortunately, in JavaScript, the data type of null is an object. Undefined and null are equal in value but different in type. 

The instanceof operator returns true if an object is an instance of the specified object.

Converting Strings to Numbers, Numbers to Strings, Dates to Numbers, Numbers to Dates, Booleans to Numbers and Numbers to Booleans using Number() , toString(), String() and Unary + operator like this –

 changing DTypes

JAVASCRIPT BITWISE

<https://www.w3schools.com/js/js_bitwise.asp>

leaving it as it is irritating

RegExp

A regular expression can be a single character, or a more complicated pattern.

Regular expressions can be used to perform all types of text search and text replace operations. 

Example explained:

/w3schools/i - is a regular expression.

w3schools is a pattern (to be used in a search).

i is a modifier (modifies the search to be case-insensitive).

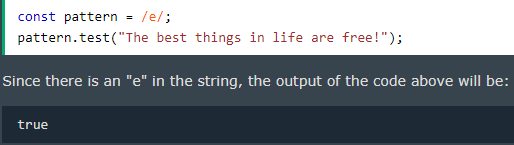
The search() method uses an expression to search for a match, and returns the position of the match.

The replace() method returns a modified string where the pattern is replaced.



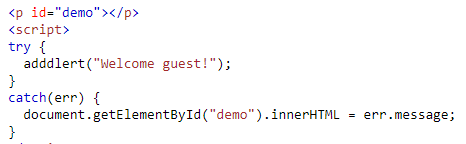
The test() method is a RegExp expression method. It searches a string for a pattern, and returns true or false, depending on the result.

The following example searches a string for the character "e":



JavaScript Errors

* The try statement defines a code block to run (to try).
* The catch statement defines a code block to handle any error.
* The finally statement defines a code block to run regardless of the result.
* The throw statement defines a custom error.

JavaScript catches adddlert as an error, and executes the catch code to handle it.

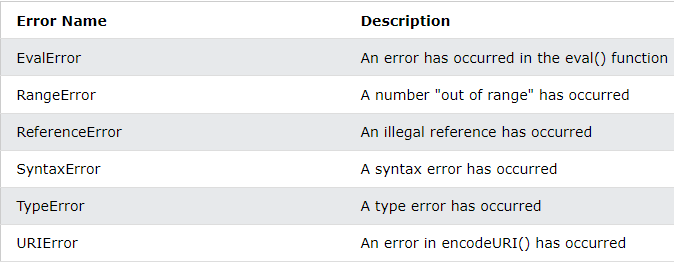
The throw statement allows you to create a custom error. Technically you can throw an exception (throw an error). The exception can be a JavaScript String, a Number, a Boolean or an Object. JavaScript will actually create an Error object with two properties: name and message.

A really good example

(https://www.w3schools.com/js/tryit.asp?filename=tryjs\_throw\_error)

The finally statement lets you execute code, after try and catch, regardless of the result.

Six different values can be returned by the error name property.

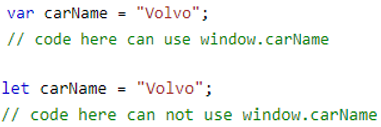
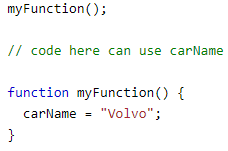


A ReferenceError is thrown if you use (reference) a variable that has not been declared. A TypeError is thrown if you use a value that is outside the range of expected types. 

URI (Uniform Resource Identifier) Error - A URIError is thrown if you use illegal characters in a URI function.

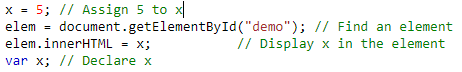
JavaScript Scope

* Variables declared within a JavaScript function, become LOCAL to the function.
* Variables defined inside a function are not accessible (visible) from outside the function.
* Variables declared Globally (outside any function) have Global Scope. Global variables can be accessed from anywhere in a JavaScript program.
* In JavaScript, objects and functions are also variables.
* If you assign a value to a variable that has not been declared, it will automatically become a GLOBAL variable.

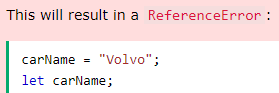


* With JavaScript, the global scope is the JavaScript environment. In HTML, the global scope is the window object.
* Global variables defined with the var keyword belong to the window object and with let they doesn’t.
* WARNING - Do NOT create global variables unless you intend to.

Hoisting is JavaScript's default behavior of moving all declarations to the top of the current scope (to the top of the current script or the current function).



Variables defined with let and const are hoisted to the top of the block, but not initialized. Meaning: The block of code is aware of the variable, but it cannot be used until it has been declared. JavaScript only hoists declarations, not initializations.



"use strict"; Defines that JavaScript code should be executed in "strict mode". The purpose of "use strict" is to indicate that the code should be executed in "strict mode".

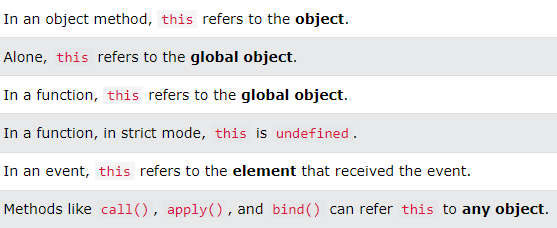
With strict mode, you can not, for example, use undeclared variables. Strict mode is declared by adding "use strict"; to the beginning of a script or a function.

So "use strict"; only matters to new compilers that "understand" the meaning of it. So, Using a variable or object, without declaring it, is not allowed, Deleting a variable (or object) or function is not allowed.

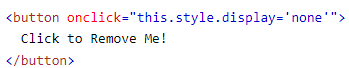
To know more – (<https://www.w3schools.com/js/js_strict.asp>)

What is this?

In JavaScript, the this keyword refers to an object. Which object depends on how this is being invoked (used or called). The this keyword refers to different objects depending on how it is used.



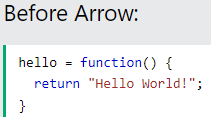
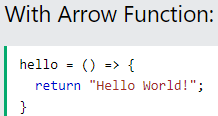
this in Event Handlers - In HTML event handlers, this refers to the HTML element that received the event.



This can be used in variety of way , but we will not go in much detail as I find it unnecessary. Two most used ways would be in html event handler and inside an object. To know more – (<https://www.w3schools.com/js/js_this.asp>)

JavaScript Arrow Function

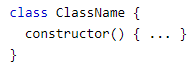
 /// 20

// Hello Universe!

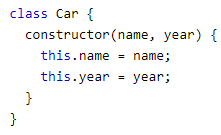
JavaScript Classes

Use the keyword class to create a class. Always add a method named constructor().



A JavaScript class is not an object. It is a template for JavaScript objects.

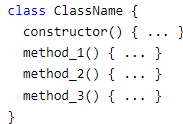
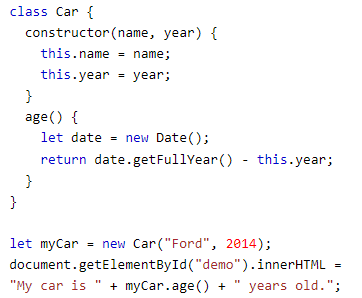
EXAMPLE-

When you have a class, you can use the class to create objects.

The constructor is used to initialize object properties. If you do not define a constructor method, JavaScript will add an empty constructor method.

Class methods are created with the same syntax as constructor

 You can also add parameters to method – (<https://www.w3schools.com/js/tryit.asp?filename=tryjs_class_method2>)

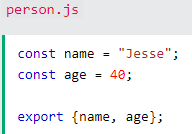
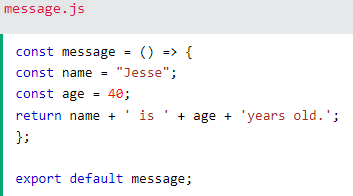
**JavaScript modules** allow you to break up your code into separate files.This makes it easier to maintain a code-base.Modules are imported from external files with the import statement.Modules also rely on type="module" in the <script> tag.

Modules with functions or variables can be stored in any external file.

Exports

There are two types of exports: **Named Exports and Default Exports.**

Let us create a file named person.js, and fill it with the things we want to export. exports all at once at the bottom

 using export {var, function}; export default function,…..;

Let us create another file, named message.js, and use it for demonstrating default export. You can only have one default export in a file.

Import - You can import modules into a file in two ways, based on if they are named exports or default exports.

Named exports are constructed using curly braces. Default exports are not.

 -----named export

 ------ default export

JavaScript JSON

JSON is a format for storing and transporting data.

JSON is often used when data is sent from a server to a web page.

JSON stands for JavaScript Object Notation, JSON is a lightweight data interchange format, JSON is language independent \*, JSON is "self-describing" and easy to understand.

Code for reading and generating JSON data can be written in any programming language.

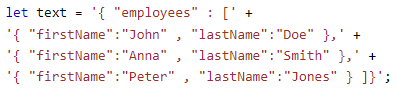
This JSON syntax defines an employees object: an array of 3 employee objects-



* Data is in name/value pairs
* Data is separated by commas
* Curly braces hold objects
* Square brackets hold arrays

Just like in JavaScript, an array can contain objects. A common use of JSON is to read data from a web server, and display the data in a web page.

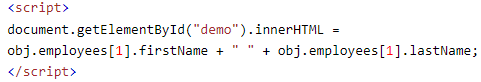
For simplicity, this can be demonstrated using a string as input. First, create a JavaScript string containing JSON syntax –

 Then, use the JavaScript built-in function JSON.parse() to convert the string into a JavaScript object.

 ////JSON.parse(string\_var)

Use the JavaScript function JSON.stringify() to convert it into a string. 

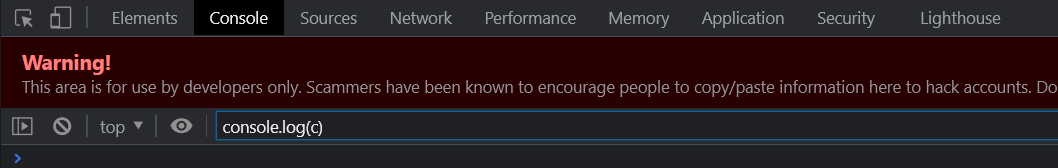
Finally, use the new JavaScript object in your page –



Searching for (and fixing) errors in programming code is called code debugging.

Debugging is a challenging task, however, most modern browsers have a built-in JavaScript debugger that can be activated by pressing the F12 key. This allows you to set breakpoints, examine variables, and turn on and off error reporting while the code is executing.

If your browser supports debugging, you can use console.log() to display JavaScript values in the debugger window:



The log() method writes (logs) a message to the console. The log() method is useful for testing purposes.

**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.

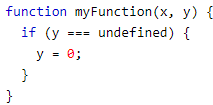
If no debugging is available, the debugger statement has no effect. With the debugger turned on, this code will stop executing before it executes the third line. 

Always use the same naming convention for all your code. For example:

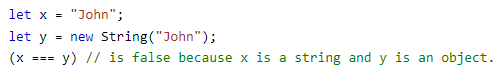
* Variable and function names written as camelCase
* Global variables written in UPPERCASE
* Constants (like PI) written in UPPERCASE
* Do not start names with a $ sign.
* Minimize the use of global variables. This includes all data types, objects, and functions. Global variables and functions can be overwritten by other scripts. Use local variables instead.
* It is a good coding practice to put all declarations at the top of each script or function
* Declaring objects and arrays with const will prevent any accidental change of type.
* Beware that numbers can accidentally be converted to strings or NaN.  Subtracting a string from a string, does not generate an error but returns NaN.
* Accessing the HTML DOM is very slow, compared to other JavaScript statements. If you expect to access a DOM element several times, access it once, and use it as a local variable: 
* Don't create new variables if you don't plan to save values.
* Putting your scripts at the bottom of the page body lets the browser load the page first.

Use Parameter Defaults

If a function is called with a missing argument, the value of the missing argument is set to undefined. Undefined values can break your code. It is a good habit to assign default values to arguments.



Always treat numbers, strings, or booleans as primitive values. Not as objects. Declaring these types as objects, slows down execution speed, and produces nasty side effects



COMMON MISTAKES IN JS -(<https://www.w3schools.com/js/js_mistakes.asp>)

An alternative is to use defer="true" in the script tag. The defer attribute specifies that the script should be executed after the page has finished parsing, but it only works for external scripts. Or -

<script src="path/to/script.js" defer></script>

JS reserved words – (<https://www.w3schools.com/js/js_reserved.asp>)

JavaScript defines 7 types of primitive data types – string, number, Boolean, Null, undefined, symbol, bigint

**Javascript Object**

An object method is an object property containing a function definition.  There are different ways to create new objects:

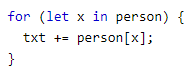
* Create a single object, using an object literal. -- const person = {};
* Create a single object, with the keyword new.
* Create an object using Object.create().

Objects are mutable: They are addressed by reference, not by value. If person is an object, the following statement will not create a copy of person.  The object x is not a copy of person. It is person. Both x and person are the same object. Any changes to x will also change person, because x and person are the same object. If x=10 then person.age also equal 10.

The syntax for accessing the property of an object is:

objectName.property or objectName[“property”]

Looping through the properties of an object -

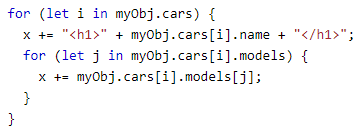
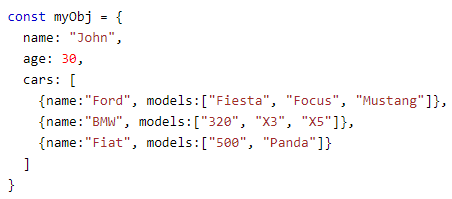


The delete keyword deletes a property from an object--- 

The delete operator is designed to be used on object properties. It has no effect on variables or functions. The delete operator should not be used on predefined JavaScript object properties. It can crash your application.

Nested Arrays and Objects

Values in objects can be arrays, and values in arrays can be objects



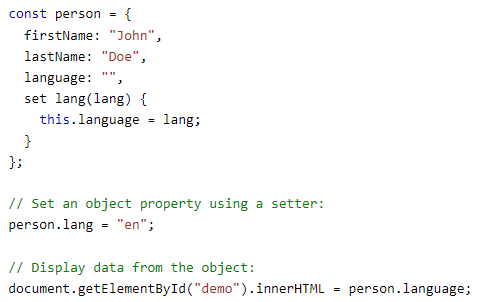
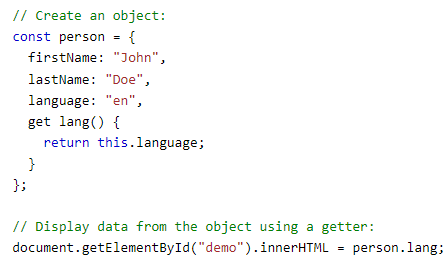
To access arrays inside arrays, use a for-in loop for each array: Output – (<https://www.w3schools.com/js/tryit.asp?filename=tryjs_array_nested>)

Some common solutions to display JavaScript objects are:

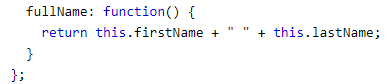
* Displaying the Object Properties by name
* Displaying the Object Properties in a Loop
* Displaying the Object using Object.values() (Object.values() converts an object to an array.)
* Displaying the Object using JSON.stringify()

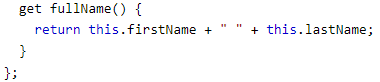
Note - JSON.stringify will not stringify functions within objects. This can be "fixed" if you convert the functions into strings before stringifying using toString().

JavaScript Getter - example uses a lang property to get the value of the language property



This example uses a lang property to set the value of the language property



Here person.fullName access fullName as a function.

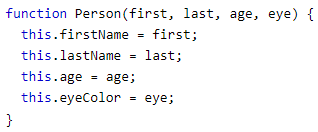
Here person.fullName without () access fullName as a property. So for ease of syntax 2nd one could be used in regular code writing.

Why Using Getters and Setters?

* It gives simpler syntax
* It allows equal syntax for properties and methods
* It can secure better data quality ( u can set set to uppercase and then set value later.)

The Object.defineProperty() method can also be used to add Getters and Setters - checkout example of this while playing with counter. -(<https://www.w3schools.com/js/tryit.asp?filename=tryjs_object_accessors_set4>)

JavaScript Object Constructors





Like this two objects can be created using constructor function. Sometimes we need a "blueprint" for creating many objects of the same "type". The way to create an "object type", is to use an object constructor function.

Adding a new property to an existing object is easy --- example ----

myFather.nationality = "English";

To add a new property to a constructor, you must add it to the constructor function. You cannot add a new method to an object constructor the same way you add a new method to an existing object. Adding methods to an object constructor must be done inside the constructor function.

All JavaScript objects inherit properties and methods from a prototype:

* Date objects inherit from Date.prototype
* Array objects inherit from Array.prototype
* Person objects inherit from Person.prototype

The Object.prototype is on the top of the prototype inheritance chain. Date objects, Array objects, and Person objects inherit from Object.prototype.

We also learned that you can not add a new property to an existing object constructor, // But sometimes u want to do so Using the prototype Property, so in that case -  The JavaScript prototype property also allows you to add new methods to objects constructors.



Note - ( about maps )

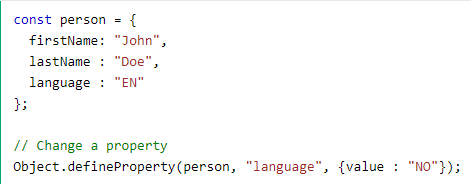
clear() - Removes all the elements from a Map.

keys() - Returns an iterator object with the keys in a Map.

values() - Returns an iterator object of the values in a Map.

Changing a Property Value in object

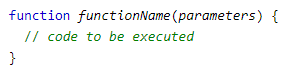


Eg - 

// Display Properties

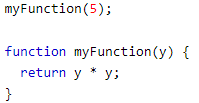
document.getElementById("demo").innerHTML = Object.getOwnPropertyNames(person);---------- This example list all properties of an object

Function

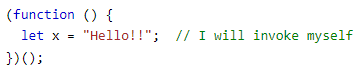
A function expression can be stored in a variable 

After a function expression has been stored in a variable, the variable can be used as a function: 

Hoisting is JavaScript's default behavior of moving declarations to the top of the current scope. Because of this, JavaScript functions can be called before they are declared



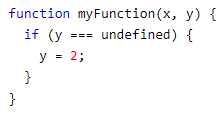
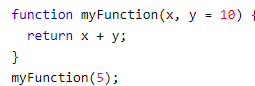
NOTE - Function expressions can be made "self-invoking". A self-invoking expression is invoked (started) automatically, without being called. Function expressions will execute automatically if the expression is followed by (). You cannot self-invoke a function declaration. You have to add parentheses around the function to indicate that it is a function expression.



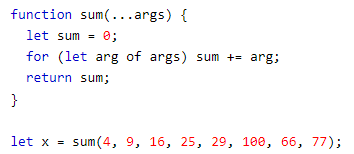
The arguments.length property returns the number of arguments received when the function was invoked.

If a function is called with missing arguments (less than declared), the missing values are set to undefined.

Sometimes this is acceptable, but sometimes it is better to assign a default value to the parameter.

 or 

The rest parameter (...) allows a function to treat an indefinite number of arguments as an array. For eg

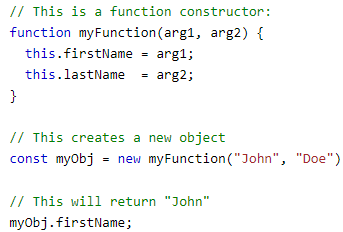
// output – 326

A really nice written code to find largest number – (<https://www.w3schools.com/js/tryit.asp?filename=tryjs_function_arguments>)

Similarly creating to find sum of all – for loop, argument.length and return result.

Note - Changes to arguments are not visible (reflected) outside the function.

Invoking a Function with a Function Constructor –



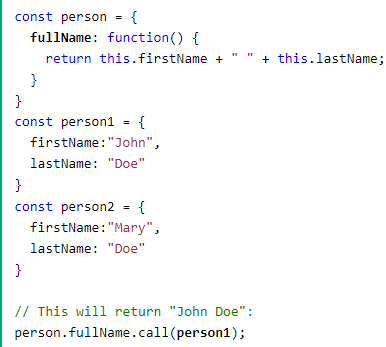
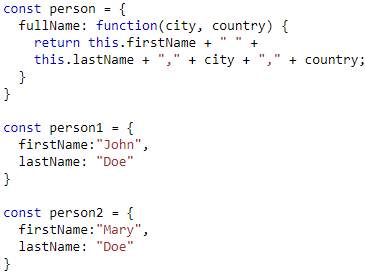
The JavaScript call() Method

This example calls the fullName method of person, using it on person1

Similarly for person2 –

Person.fullName.call(person2);

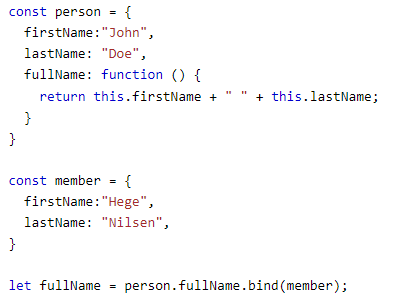
Also a example of The call() Method with Arguments so - person.fullName.call(person1, "Oslo", "Norway");

The JavaScript apply() Method - The apply() method is similar to the call(). The difference is: The call() method takes arguments separately.The apply() method takes arguments as an array. 

Interesting –

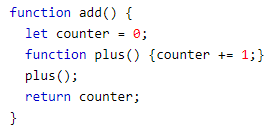
With the bind() method, an object can borrow a method from another object. The example below creates 2 objects (person and member). The member object borrows the fullname method from the person object.

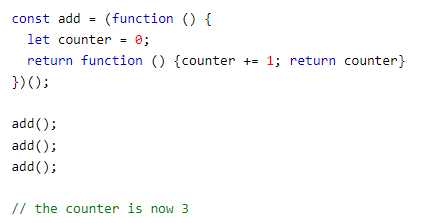


Variables created without a declaration keyword (var, let, or const) are always global, even if they are created inside a function.

Nested functions have access to the scope "above" them.

In this example, the inner function plus() has access to the counter variable in the parent function:

This could have solved the counter dilemma, if we could reach the plus() function from the outside.We also need to find a way to execute counter = 0 only once. We need a closure.



The variable add is assigned to the return value of a self-invoking function. The self-invoking function only runs once. It sets the counter to zero (0), and returns a function expression. This way add becomes a function. The "wonderful" part is that it can access the counter in the parent scope. This is called a JavaScript closure. It makes it possible for a function to have "private" variables. The counter is protected by the scope of the anonymous function, and can only be changed using the add function.

Class Inheritance

To create a class inheritance, use the extends keyword.

A class created with a class inheritance inherits all the methods from another class.

The super() method refers to the parent class.

By calling the super() method in the constructor method, we call the parent's constructor method and gets access to the parent's properties and methods.

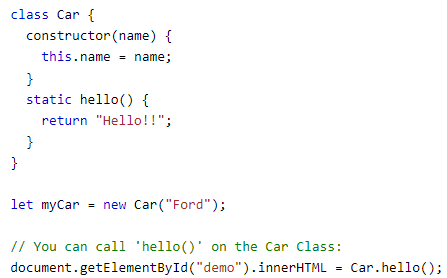
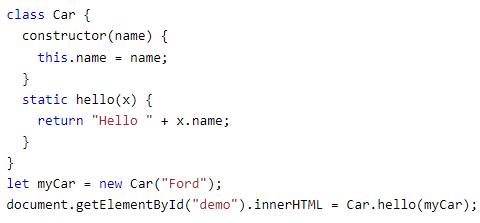


Classes also allows you to use getters and setters.

It can be smart to use getters and setters for your properties, especially if you want to do something special with the value before returning them, or before you set them. To add getters and setters in the class, use the get and set keywords.

Unlike functions, and other JavaScript declarations, class declarations are not hoisted. That means that you must declare a class before you can use it:

Static class methods are defined on the class itself. You cannot call a static method on an object, only on an object class.

Output – Hello!!

If you want to use the myCar object inside the static method, you can send it as a parameter.

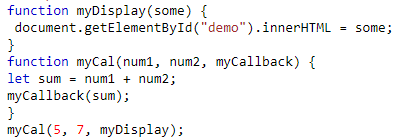
JS Async

A callback is a function passed as an argument to another function. This technique allows a function to call another function. A callback function can run after another function has finished

JavaScript functions are executed in the sequence they are called. Not in the sequence they are defined.

Sometimes you would like to have better control over when to execute a function. Suppose you want to do a calculation, and then display the result. You could call a calculator function (myCalculator), save the result, and then call another function (myDisplayer) to display the result. Or, you could call a calculator function (myCalculator), and let the calculator function call the display function (myDisplayer).

Using a callback, you could call the calculator function (myCalculator) with a callback (myCallback), and let the calculator function run the callback after the calculation is finished:



In the example above, myDisplayer is a called a callback function. It is passed to myCalculator() as an argument.



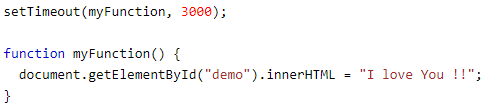
Example to remove negative numbers from an array – (<https://www.w3schools.com/js/tryit.asp?filename=tryjs_callback10>)

In the example above, (x) => x >= 0 is a callback function. It is passed to removeNeg() as an argument.

**Asynchronous JavaScript**

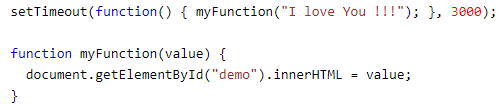
Functions running in parallel with other functions are called asynchronous. A good example is JavaScript setTimeout().

When using the JavaScript function setTimeout(), you can specify a callback function to be executed on time-out.



In the example above, myFunction is used as a callback. myFunction is passed to setTimeout() as an argument. 3000 is the number of milliseconds before time-out, so myFunction() will be called after 3 seconds. So myFunction and setTimeout both are running parallely.

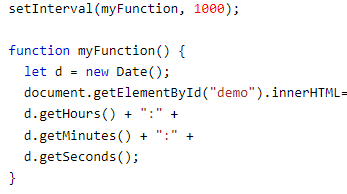
Instead of passing the name of a function as an argument to another function, you can always pass a whole function instead –



In the example above, function(){ myFunction("I love You !!!"); } is used as a callback. It is a complete function. The complete function is passed to setTimeout() as an argument.

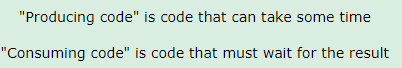
Something cool –

When using the JavaScript function setInterval(), you can specify a callback function to be executed for each interval.

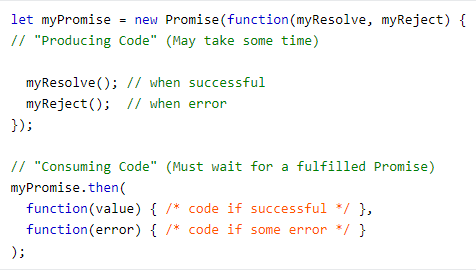
 H:M:S

In the example above, myFunction is used as a callback. myFunction is passed to setInterval() as an argument. 1000 is the number of milliseconds between intervals, so myFunction() will be called every second. Output It creates a working real time clock – basically refreshing the time after each sec and displaying.

With asynchronous programming, JavaScript programs can start long-running tasks, and continue running other tasks in parallel. But, asynchronus programmes are difficult to write and difficult to debug. Because of this, most modern asynchronous JavaScript methods don't use callbacks. Instead, in JavaScript, asynchronous programming is solved using Promises instead.



A JavaScript Promise object contains both the producing code and calls to the consuming code:

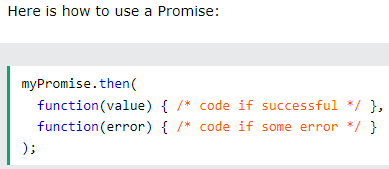


A JavaScript Promise object can be:

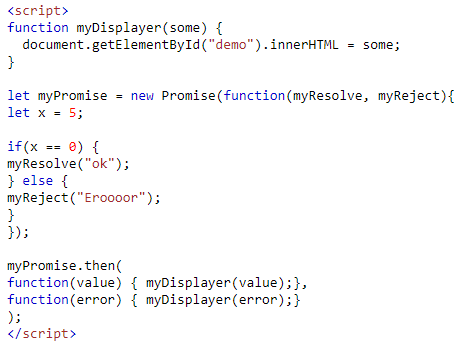
Pending, Fulfilled, Rejected

The Promise object supports two properties: state and result.

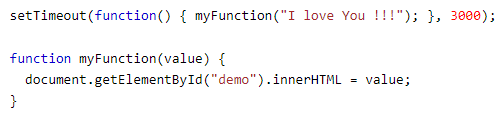
* While a Promise object is "pending" (working), the result is undefined.
* When a Promise object is "fulfilled", the result is a value.
* When a Promise object is "rejected", the result is an error object.

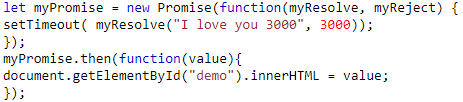


Using promises –



Lets take an example with promise and without it –

 now with promise –



Using call back for creating program to wait for a file -

(<https://www.w3schools.com/js/tryit.asp?filename=tryjs_callback7>)

Promises in js in 100 sec – (<https://www.youtube.com/watch?v=RvYYCGs45L4&ab_channel=Fireship>)

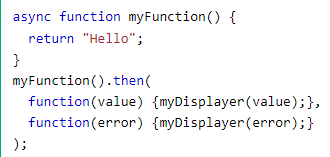
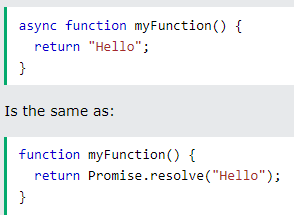
(<https://www.youtube.com/watch?v=DHvZLI7Db8E&ab_channel=WebDevSimplified>)

Text

Description automatically generated

Async

The keyword async before a function makes the function return a promise:



It makes the function return value act as a promise. U can further use then with it .

**Await** Syntax

The await keyword can only be used inside an async function. The await keyword makes the function pause the execution and wait for a resolved promise before it continues.



Text

Description automatically generated

HTML DOM

When a web page is loaded, the browser creates a Document Object Model of the page. The HTML DOM model is constructed as a tree of Objects:

JavaScript can manipulate HTML elements, attributes, styles, events, and dynamically add or remove them on a web page.

"*The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."*

The HTML DOM is a standard for how to get, change, add, or delete HTML elements.



In the example above, getElementById is a method, while innerHTML is a property.

Finding HTML Elements

By Id - 

By class name - 

by Tag Name –

Text, letter

Description automatically generatedGraphical user interface, text

Description automatically generated

by HTML Object Collections –

(<https://www.w3schools.com/js/tryit.asp?filename=tryjs_dom_form_elements>

The following HTML objects (and object collections) are also accessible:

document.anchors, document.body, document.documentElement, document.embeds, document.forms, document.head, document.images, document.links, document.scripts, document.title

by CSS Selectors - 

This example returns a list of all <p> elements with class="intro".

Some changes in HTML –



A picture containing text

Description automatically generated - changing attributes



document.write() can be used to write directly to the HTML output stream: 

JavaScript Form Validation

If a form field (fname) is empty, this function alerts a message, and returns false, to prevent the form from being submitted:

Text

Description automatically generated(https://www.w3schools.com/js/tryit.asp?filename=tryjs\_validation\_js)

JavaScript Can Validate Numeric Input - (<https://www.w3schools.com/js/tryit.asp?filename=tryjs_validation_number>)

HTML form validation can be performed automatically by the browser: If a form field (fname) is empty, the required attribute prevents this form from being submitted:

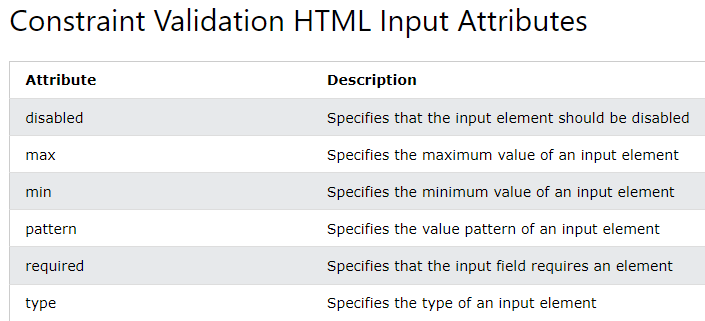
Graphical user interface, text

Description automatically generated

Data validation is the process of ensuring that user input is clean, correct, and useful. Typical validation tasks are:

* has the user filled in all required fields?
* has the user entered a valid date?
* has the user entered text in a numeric field?

Most often, the purpose of data validation is to ensure correct user input. Validation can be defined by many different methods, and deployed in many different ways. Server side validation is performed by a web server, after input has been sent to the server. Client side validation is performed by a web browser, before input is sent to a web server.



Changing HTML Style - 

Javascript Animations

Reacting to Events

A JavaScript can be executed when an event occurs, like when a user clicks on an HTML element. To execute code when a user clicks on an element, add JavaScript code to an HTML event attribute: onclick=JavaScript

Examples of HTML events:

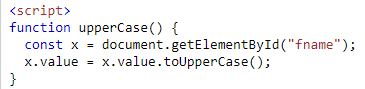
When a user clicks the mouse, When a web page has loaded, When an image has been loaded, When the mouse moves over an element, When an input field is changed, When an HTML form is submitted, When a user strokes a key





The onload and onunload events are triggered when the user enters or leaves the page. The onload event can be used to check the visitor's browser type and browser version, and load the proper version of the web page based on the information. - 

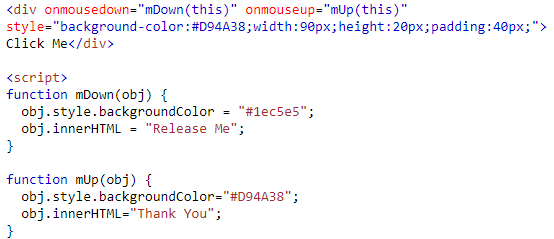
The onchange event is often used in combination with validation of input fields. Below is an example of how to use the onchange. The upperCase() function will be called when a user changes the content of an input field.

 - automatically makes input text to uppercase

The onmouseover and onmouseout events can be used to trigger a function when the user mouses over, or out of, an HTML element:



You can also note here use of (this) is done, and obj is used in parallel to function corresponding to that element. Similarly –onmousedown & onmouseup



HTML DOM Events – (<https://www.w3schools.com/jsref/dom_obj_event.asp>)

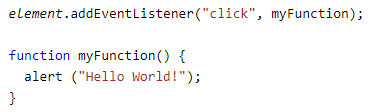
The addEventListener() method attaches an event handler to the specified element.The addEventListener() method attaches an event handler to an element without overwriting existing event handlers.

You can add many event handlers to one element. You can easily remove an event listener by using the removeEventListener() method.



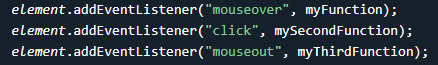
The first parameter is the type of the event (like "click" or "mousedown" or any other HTML DOM Event.) The second parameter is the function we want to call when the event occurs. Third is optional.



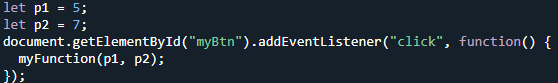


method allows you to add many events to the same element & different types to the same element like – mouseover, mouseout, etc





When passing parameter values, use an "anonymous function" that calls the specified function with the parameters:

 /// 35

Event Bubbling or Event Capturing?

**there are two ways of event propagation in the HTML DOM, bubbling and capturing.**

*Event propagation is a way of defining the element order when an event occurs. If you have a <p> element inside a <div> element, and the user clicks on the <p> element, which element's "click" event should be handled first?*

* In bubbling the inner most element's event is handled first and then the outer: the <p> element's click event is handled first, then the <div> element's click event.
* In capturing the outer most element's event is handled first and then the inner: the <div> element's click event will be handled first, then the <p> element's click event.

With the addEventListener() method you can specify the propagation type by using the "useCapture" parameter: 

The default value is false, which will use the bubbling propagation. True – capture

Really good example - <https://www.w3schools.com/js/tryit.asp?filename=tryjs_addeventlistener_usecapture>

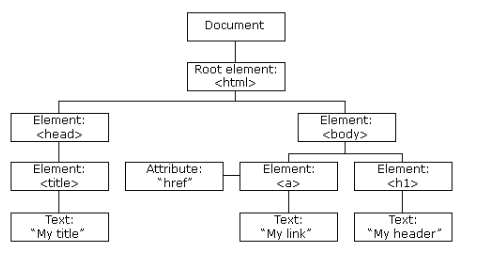
The removeEventListener() method removes event handlers that have been attached with the addEventListener() method: example –

<https://www.w3schools.com/js/tryit.asp?filename=tryjs_addeventlistener_remove>

--------------------------------------------------------------------------------------------------------------------------------------

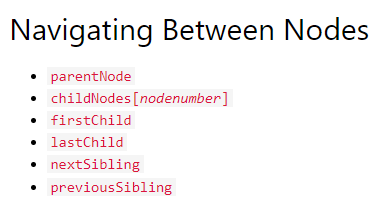
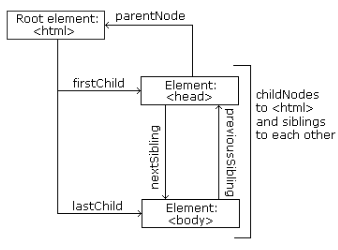
DOM Nodes

According to the HTML DOM standard, everything in an HTML document is a node:



With the HTML DOM, all nodes in the node tree can be accessed by JavaScript. New nodes can be created, and all nodes can be modified or deleted. The nodes in the node tree have a hierarchical relationship to each other.

The terms parent, child, and sibling are used to describe the relationships. In a node tree, the top node is called the root (or root node). Every node has exactly one parent, except the root (which has no parent). A node can have a number of children .Siblings (brothers or sisters) are nodes with the same parent.



You can use the following node properties to navigate between nodes with JavaScript.

<title id="demo">DOM Tutorial</title>

The element node <title> (in the example above) does not contain text. It contains a text node with the value "DOM Tutorial". The value of the text node can be accessed by the node's innerHTML property:







Accessing the innerHTML property is the same as accessing the nodeValue of the first child.

Example if –



 &

 will give same result this also - 

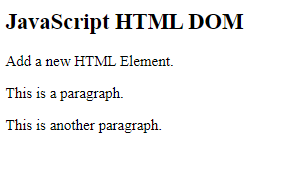
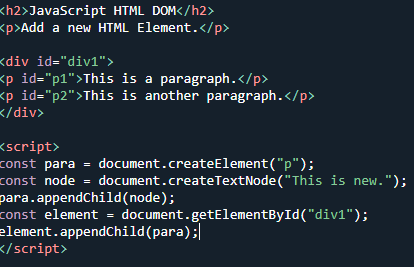
There are two special properties that allow access to the full document:

* document.body - The body of the document
* document.documentElement - The full document

Nodemane - document.getElementById("id01").nodeName like this will give name of the tag of id mentioned in uppercaps, in this case – H1

Creating New HTML Elements (Nodes)

To add a new element to the HTML DOM, you must create the element (element node) first, and then append it to an existing element.



* This code creates a new <p> element:
* const para = document.createElement("p");
* To add text to the <p> element, you must create a text node first. This code creates a text node:
* const node = document.createTextNode("This is a new paragraph.");
* Then you must append the text node to the <p> element:
* para.appendChild(node);
* Finally you must append the new element to an existing element.
* This code finds an existing element:
* const element = document.getElementById("div1");
* This code appends the new element to the existing element:
* element.appendChild(para);

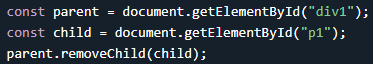
## Removing Existing HTML Elements

To remove an HTML element, use the remove() method:

const elmnt = document.getElementById("p1"); elmnt.remove(); ------- this removes element with id p1. Find the element you want to remove: const elmnt = document.getElementById("p1");

Then execute the remove() method on that element: elmnt.remove();

Also U can do –



Replacing HTML Elements - To replace an element to the HTML DOM, use the replaceChild() method:



The HTMLCollection Object

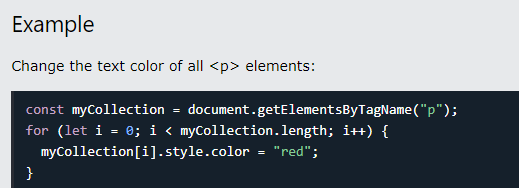
The getElementsByTagName() method returns an HTMLCollection object.

An HTMLCollection object is an array-like list (collection) of HTML elements.

Example - <https://www.w3schools.com/js/tryit.asp?filename=tryjs_dom_htmlcollection>

The length property defines the number of elements in an HTMLCollection -  - The length property is useful when you want to loop through the elements in a collection. In above example – it would have given 3 paragraphs.

Check out a really good example for its utilization –



NOTE –

A NodeList object is a list (collection) of nodes extracted from a document. A NodeList object is almost the same as an HTMLCollection object. A NodeList and an HTMLcollection is very much the same thing.







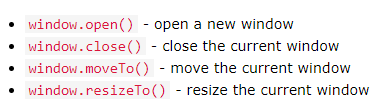
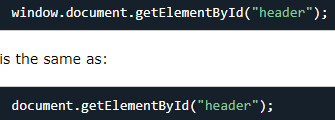
* An HTMLCollection is a collection of document elements.
* A NodeList is a collection of document nodes (element nodes, attribute nodes, and text nodes).
* HTMLCollection items can be accessed by their name, id, or index number.
* NodeList items can only be accessed by their index number.

# **JavaScript Window - The Browser Object Model**

The Browser Object Model (BOM) allows JavaScript to "talk to" the browser.

The Window Object

The window object is supported by all browsers. It represents the browser's window. All global JavaScript objects, functions, and variables automatically become members of the window object. Global variables are properties of the window object. Global functions are methods of the window object. Even the document object (of the HTML DOM) is a property of the window object:

also

Two properties can be used to determine the size of the browser window.

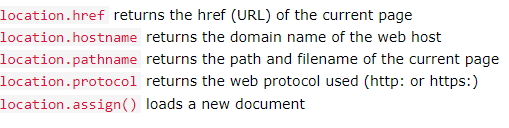
Both properties return the sizes in pixels:

* window.innerHeight - the inner height of the browser window (in pixels)
* window.innerWidth - the inner width of the browser window (in pixels)

The window.screen object contains information about the user's screen. The window.screen object can be written without the window prefix.

To get screen properties use can use them

The window.location object can be used to get the current page address (URL) and to redirect the browser to a new page.



The window.location.href property returns the URL of the current page.

Among these I find location.assign() – most useful, it open a new page, document. Eg - location.assign(“www.hello.com”) – this opens the site hello.com

The window.history object contains the browsers history.

COOL STUFF NOW

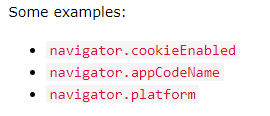
To protect the privacy of the users, there are limitations to how JavaScript can access this object.

Some methods:

* history.back() - same as clicking back in the browser. loads the previous URL in the history list.
* history.forward() - same as clicking forward in the browser

Check out example if need one - <https://www.w3schools.com/js/js_window_history.asp>

The window.navigator object contains information about the visitor's browser.



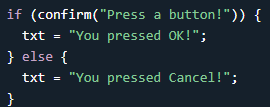
* The cookieEnabled property returns true if cookies are enabled, otherwise false
* The appName property returns the application name of the browser
* The appCodeName property returns the application code name of the browser
* The product property returns the product name of the browser engine
* The appVersion property returns version information about the browser
* The platform property returns the browser platform (operating system). Eg – Win32
* The language property returns the browser's language
* The onLine property returns true if the browser is online
* Navigator.geolocation is also a attribute

# **JavaScript Popup Boxes**

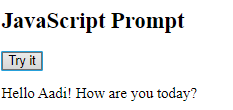
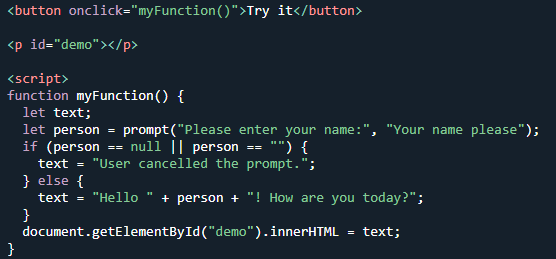
JavaScript has three kind of popup boxes: Alert box, Confirm box, and Prompt box.

 - When an alert box pops up, the user will have to click "OK" to proceed.

A confirm box is often used if you want the user to verify or accept something. When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed. If the user clicks "OK", the box returns true. If the user clicks "Cancel", the box returns false.  eg –



A prompt box is often used if you want the user to input a value before entering a page. When a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed after entering an input value. If the user clicks "OK" the box returns the input value. If the user clicks "Cancel" the box returns null.



To display line breaks inside a popup box, use a back-slash followed by the character n. 

JavaScript Timing Events - JavaScript can be executed in time-intervals. This is called timing events. The setTimeout() and setInterval() are both methods of the HTML DOM Window object.

The two key methods to use with JavaScript are:

* setTimeout(function, milliseconds)

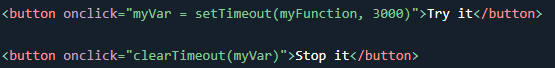
Executes a function, after waiting a specified number of milliseconds.

* setInterval(function, milliseconds)

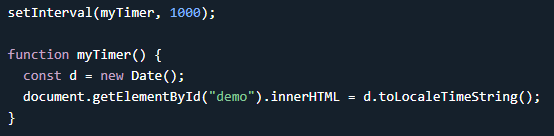
Same as setTimeout(), but repeats the execution of the function continuously.



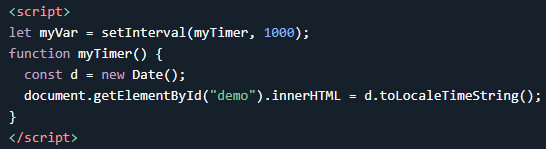
The clearTimeout() method stops the execution of the function specified in setTimeout().The clearTimeout() method uses the variable returned from setTimeout().If the function has not already been executed, you can stop the execution by calling the clearTimeout() method

Examples - also <https://www.w3schools.com/js/tryit.asp?filename=tryjs_settimeout2>

setInterval() - The first parameter is the function to be executed. The second parameter indicates the length of the time-interval between each execution. This example executes a function called "myTimer" once every second (like a digital watch).

 repeat display timer after a specific time (1 sec)

Stopping using clearInterval- IMP we need to create variable ( myvar in this case) 



So clearTimeout and clearInterval

Javascript Cookies - Cookies let you store user information in web pages.

Cookies are data, stored in small text files, on your computer.

When a web server has sent a web page to a browser, the connection is shut down, and the server forgets everything about the user.

Cookies were invented to solve the problem "how to remember information about the user":

* When a user visits a web page, his/her name can be stored in a cookie.
* Next time the user visits the page, the cookie "remembers" his/her name.

Cookies are saved in name-value pairs like:



When a browser requests a web page from a server, cookies belonging to the page are added to the request. This way the server gets the necessary data to "remember" information about users.

JavaScript can create, read, and delete cookies with the document.cookie property. With JavaScript, a cookie can be created like this: 

You can also add an expiry date (in UTC time). By default, the cookie is deleted when the browser is closedWith a path parameter, you can tell the browser what path the cookie belongs to. By default, the cookie belongs to the current page. 

With JavaScript, cookies can be read like this:

let x = document.cookie;

document.cookie will return all cookies in one string much like: cookie1=value; cookie2=value; cookie3=value; etc……….

With JavaScript, you can change a cookie the same way as you create it, after =

Deleting a cookie is very simple. You don't have to specify a cookie value when you delete a cookie. Just set the expires parameter to a past date: 

JavaScript Cookie Example

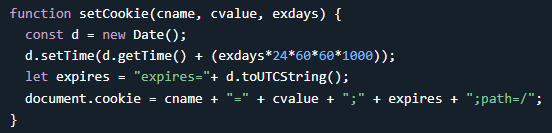
In the example to follow, we will create a cookie that stores the name of a visitor.

The first time a visitor arrives to the web page, he/she will be asked to fill in his/her name. The name is then stored in a cookie.

The next time the visitor arrives at the same page, he/she will get a welcome message.

For the example we will create 3 JavaScript functions:

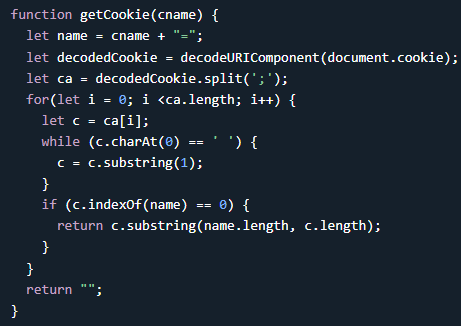
* A function to set a cookie value



The parameters of the function above are the name of the cookie (cname), the value of the cookie (cvalue), and the number of days until the cookie should expire (exdays).

The function sets a cookie by adding together the cookiename, the cookie value, and the expires string.

* A function to get a cookie value



Take the cookiename as parameter (cname).

Create a variable (name) with the text to search for (cname + "="). Decode the cookie string, to handle cookies with special characters, e.g. '$'

Split document.cookie on semicolons into an array called ca (ca = decodedCookie.split(';')).

Loop through the ca array (i = 0; i < ca.length; i++), and read out each value c = ca[i]).

If the cookie is found (c.indexOf(name) == 0), return the value of the cookie (c.substring(name.length, c.length).

If the cookie is not found, return "".

* A function to check a cookie value



Last, we create the function that checks if a cookie is set. If the cookie is set it will display a greeting. If the cookie is not set, it will display a prompt box, asking for the name of the user, and stores the username cookie for 365 days, by calling the setCookie function

Together all three makes a example – (It stores values in cookies for next visit to same page) <https://www.w3schools.com/js/tryit.asp?filename=tryjs_cookie_username>

Web APIs

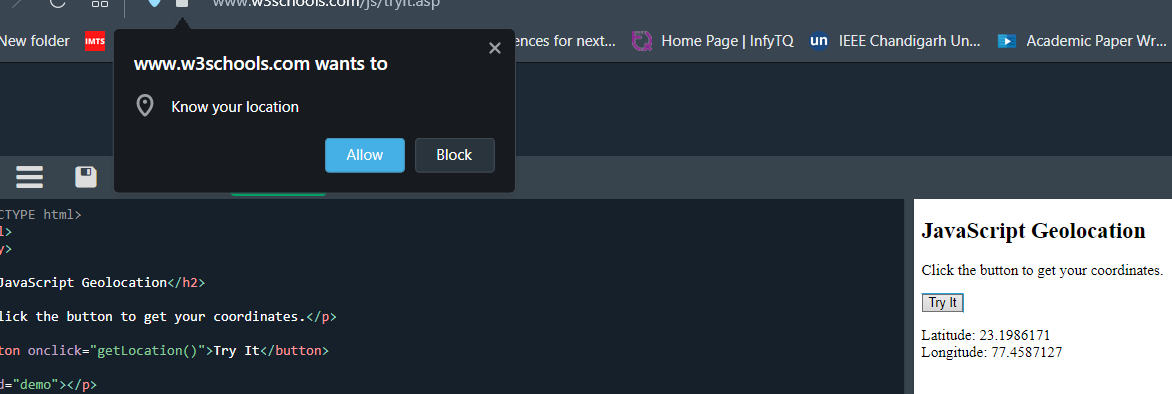
API stands for Application Programming Interface.

A Web API is an application programming interface for the Web. A Browser API can extend the functionality of a web browser. A Server API can extend the functionality of a web server.

Browser API

All browsers have a set of built-in Web APIs to support complex operations, and to help accessing data.For example, the Geolocation API can return the coordinates of where the browser is located.

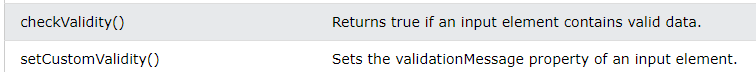




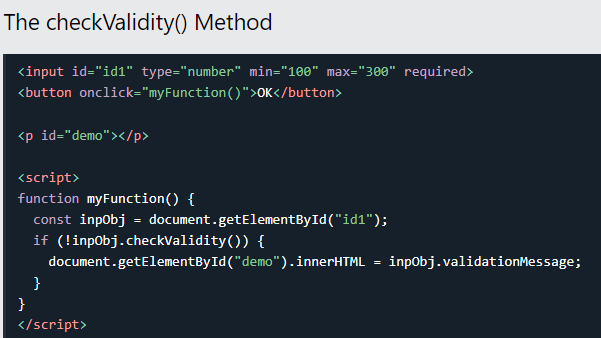
Third Party APIs - Third party APIs are not built into your browser. To use these APIs, you will have to download the code from the Web.

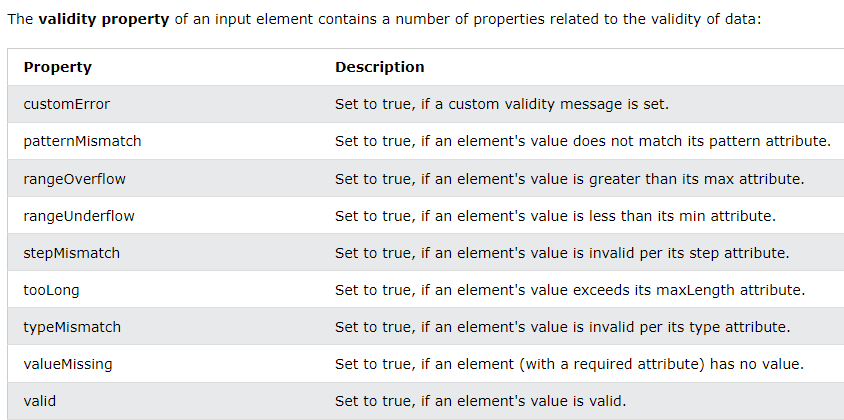
Examples:

* YouTube API - Allows you to display videos on a web site.
* Twitter API - Allows you to display Tweets on a web site.
* Facebook API - Allows you to display Facebook info on a web site.



If an input field contains invalid data, display a message:

This is something to note 



We saw examples of validate.rangeOverflow and validate.rangeUnderflow

The Web History API provides easy methods to access the windows.history object. The window.history object contains the URLs (Web Sites) visited by the user. We saw earlier history.forward() & history.back() – history of previous and next url from browser

Also The go() method loads a specific URL from the history list:

 history.go(-2) or (2) 🡪next

Web Storage API

The Web Storage API is a simple syntax for storing and retrieving data in the browser. It is very easy to use:

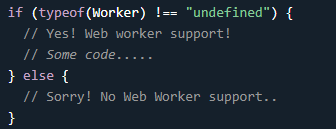
The localStorage object provides access to a local storage for a particular Web Site. It allows you to store, read, add, modify, and delete data items for that domain. The data is stored with no expiration date, and will not be deleted when the browser is closed. The data will be available for days, weeks, and years.

It takes a name and a value as parameters. It takes a name as parameter.

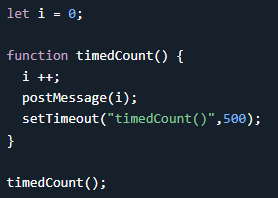
The sessionStorage object is identical to the localStorage object. The difference is that the sessionStorage object stores data for one session. The data is deleted when the browser is closed. You can also do setItem() and getItem() with sessionStorage.

Web Workers API - A web worker is a JavaScript running in the background, without affecting the performance of the page. When executing scripts in an HTML page, the page becomes unresponsive until the script is finished.

A web worker is a JavaScript that runs in the background, independently of other scripts, without affecting the performance of the page. You can continue to do whatever you want: clicking, selecting things, etc., while the web worker runs in the background.

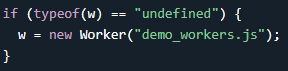
Before creating a web worker, check whether the user's browser supports it:  if(typeof(Worker) ! == undefined 

Now, let's create our web worker in an external JavaScript. Here, we create a script that counts. The script is stored in the "demo\_workers.js" file

The important part of the code above is the postMessage() method - which is used to post a message back to the HTML page. Also setTimeout is used as a recursion if u see – it calls itself after delay of 500 milisec

Now that we have the web worker file, we need to call it from an HTML page.

The following lines checks if the worker already exists, if not - it creates a new web worker object and runs the code in "demo\_workers.js"



Then we can send and receive messages from the web worker. Add an "onmessage" event listener to the web worker.

The data from the web worker is stored in event.data.

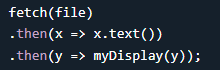
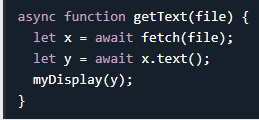
When a web worker object is created, it will continue to listen for messages (even after the external script is finished) until it is terminated. To terminate a web worker, and free browser/computer resources, use the terminate() method. 

Eg - <https://www.w3schools.com/js/tryit.asp?filename=tryjs_webworker>

Since web workers are in external files, they do not have access to the following JavaScript objects:

The window object, The document object and The parent object

JavaScript Fetch API - The example below fetches a file and displays the content

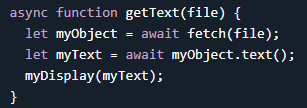
 or  since fetch is async function

In JavaScript, async and await are keywords used to handle asynchronous operations and make asynchronous code easier to read and write.

The async keyword is used to define an asynchronous function. When a function is declared as async, it automatically returns a Promise. This allows you to use await inside the function to wait for the resolution of another Promise without blocking the execution of the code.

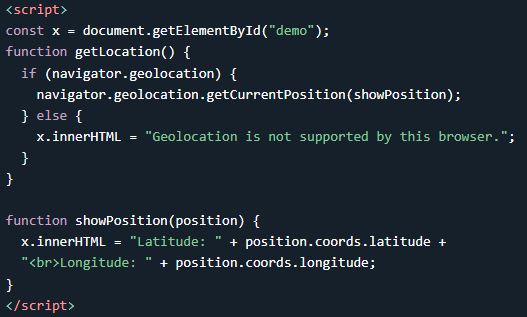
The await keyword can only be used inside an async function. It is used to pause the execution of the function until a Promise is resolved or rejected. It allows you to write asynchronous code in a more synchronous-like manner, making it easier to handle and reason about asynchronous operations.

Or even better: Use understandable names instead of x and y:



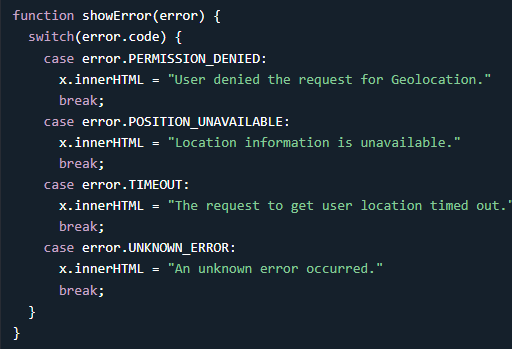
Web Geolocation API - The HTML Geolocation API is used to get the geographical position of a user. Since this can compromise privacy, the position is not available unless the user approves it. The Geolocation API will only work on secure contexts such as HTTPS.

The getCurrentPosition() method is used to return the user's position.

 previously also done with navigator.geolocation()

Handling Errors and Rejections

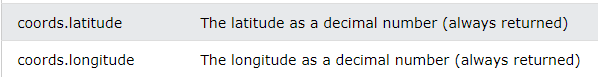
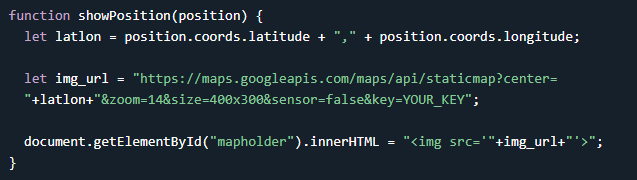
The second parameter of the getCurrentPosition() method is used to handle errors. It specifies a function to run if it fails to get the user's location



navigator.geolocation.getCurrentPosition(showPosition, showError);

To display the result in a map, you need access to a map service, like Google Maps.

In the example below, the returned latitude and longitude is used to show the location in a Google Map



The Geolocation object also has other interesting methods:

* watchPosition() - Returns the current position of the user and continues to return updated position as the user moves (like the GPS in a car).
* clearWatch() - Stops the watchPosition() method.

Navigator.geolocation().watchPosition(showPosition, showError);

AJAX - AJAX is a developer's dream, because you can:

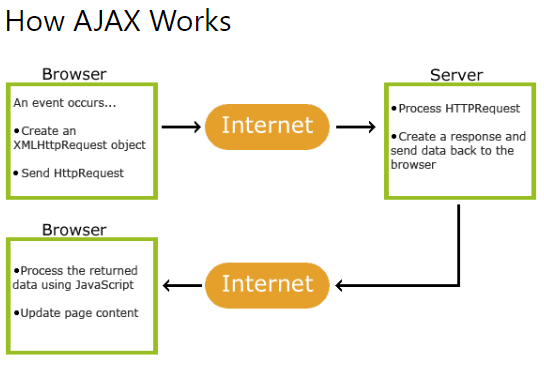
* Read data from a web server - after the page has loaded
* Update a web page without reloading the page
* Send data to a web server - in the background

AJAX = Asynchronous JavaScript And XML. AJAX is not a programming language.

AJAX just uses a combination of:

* A browser built-in XMLHttpRequest object (to request data from a web server)
* JavaScript and HTML DOM (to display or use the data)

AJAX applications might use XML to transport data, but it is equally common to transport data as plain text or JSON text. AJAX allows web pages to be updated asynchronously by exchanging data with a web server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.



Modern Browsers can use Fetch API instead of the XMLHttpRequest Object.

The Fetch API interface allows web browser to make HTTP requests to web servers.

Both fetch and XMLHttpRequest (XHR) are used in JavaScript for making HTTP requests to retrieve data from a server, but they have some differences in terms of syntax and functionality.

* fetch: Uses a modern Promise-based syntax and returns a Promise that resolves to the response object.
* XMLHttpRequest: Uses an older callback-based syntax and involves multiple steps to handle the response.

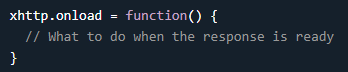
The keystone of AJAX is the XMLHttpRequest object.

* Create an XMLHttpRequest object
* Define a callback function
* Open the XMLHttpRequest object
* Send a Request to a server

The XMLHttpRequest object can be used to exchange data with a web server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.

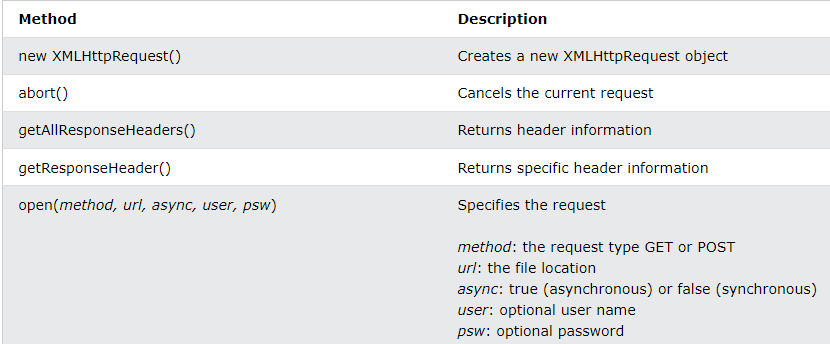
To create request - 

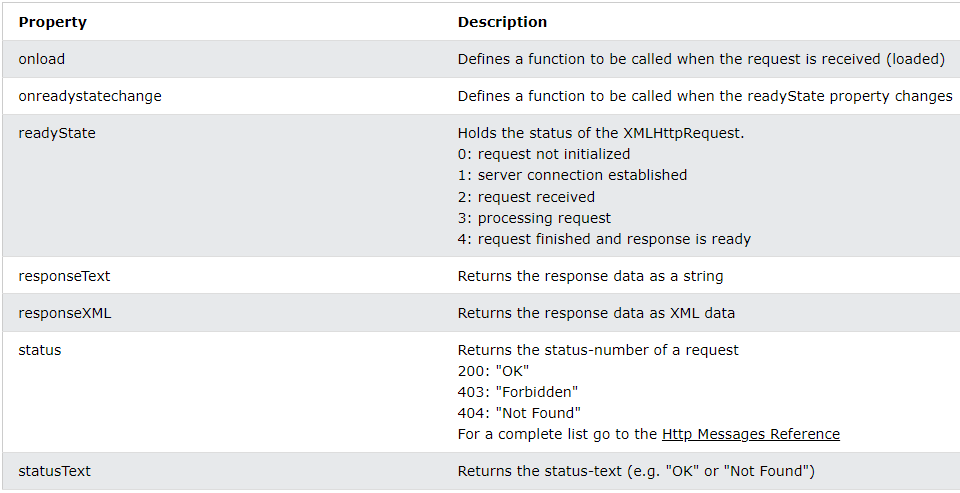
Define a Callback Function - A callback function is a function passed as a parameter to another function. In this case, the callback function should contain the code to execute when the response is ready.

To send a request to a server, you can use the open() and send() methods of the XMLHttpRequest object



eg - <https://www.w3schools.com/js/tryit.asp?filename=tryjs_ajax_xmlhttp>

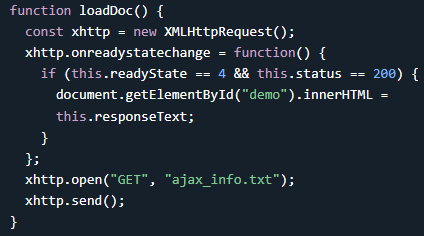
**NOTE - For security reasons, modern browsers do not allow access across domains. This means that both the web page and the XML file it tries to load, must be located on the same server. If you want to use the example above on one of your own web pages, the XML files you load must be located on your own server.** ****

****

If you have more than one AJAX task in a website, you should create one function for executing the XMLHttpRequest object, and one callback function for each AJAX task.

The function call should contain the URL and what function to call when the response is ready. (A)

Note- The readyState property holds the status of the XMLHttpRequest. The onreadystatechange property defines a callback function to be executed when the readyState changes. The status property and the statusText properties hold the status of the XMLHttpRequest object. (B)



1. (B)

The onreadystatechange function is called every time the readyState changes. When readyState is 4 and status is 200, the response is ready

Within the onreadystatechange event handler, the function checks if the readyState is equal to 4 (which means the operation has been completed) and the status is equal to 200 (which indicates a successful response from the server).

If the above conditions are met, the content of the HTML element with the id "demo" is updated with the received response text using the innerHTML property. In this case, this.responseText represents the response data obtained from the server.

To send a request to a server, we use the open() and send() methods of the XMLHttpRequest object

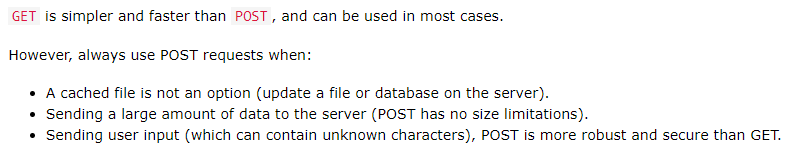
async: true (asynchronous) or false (synchronous). The file can be any kind of file, like .txt and .xml, or server scripting files like .asp and .php (which can perform actions on the server before sending the response back).

Server requests should be sent asynchronously. The async parameter of the open() method should be set to true.

NOTE – **By sending asynchronously, the JavaScript does not have to wait for the server response, but can instead:**

* **execute other scripts while waiting for server response**
* **deal with the response after the response is ready**

The default value for the async parameter is async = true. You can safely remove the third parameter from your code.

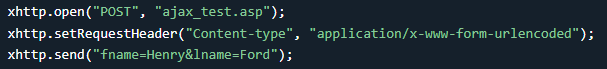
****

****you may get a cached result. To avoid this, add a unique ID to the URL.  or It is then appended as a query parameter named "id" to the URL. By adding this unique ID, it ensures that each request is treated as unique, and it helps prevent caching of the response.

Note - If you want to send information with the GET method, add the information to the URL



POST

To POST data like an HTML form, add an HTTP header with setRequestHeader(). Specify the data you want to send in the send() method

To execute a synchronous request, change the third parameter in the open() method to false. Synchronous XMLHttpRequest (async = false) is not recommended because the JavaScript will stop executing until the server response is ready. If the server is busy or slow, the application will hang or stop.

Server Response Properties-

The responseText property returns the server response as a JavaScript string, and you can use it accordingly: 

The responseXML property returns the server response as an XML DOM object. Using this property you can parse the response as an XML DOM object

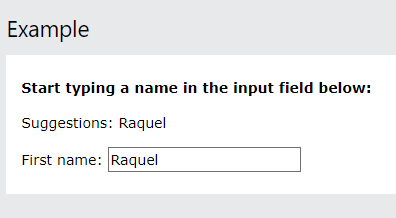
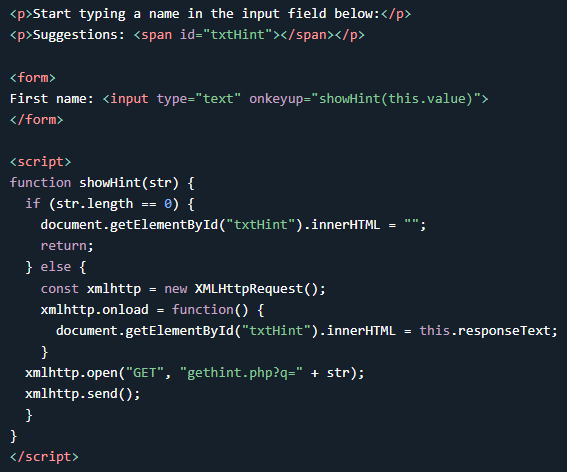
Also 2 more –

* The getAllResponseHeaders() function returns all the header information of a resource, like length, server-type, content-type, last-modified, etc
* The getResponseHeader() method returns specific header information from the server response.

An example of AJAX – XML - <https://www.w3schools.com/js/js_ajax_xmlfile.asp>

In this example data is retrieved from xml file and is used through js to display in a table.

AJAX PHP Example - <https://www.w3schools.com/js/js_ajax_php.asp>



In the example above, when a user types a character in the input field, a function called showHint() is executed. The function is triggered by the onkeyup event. First, check if the input field is empty (str.length == 0). If it is, clear the content of the txtHint placeholder and exit the function.

AJAX ASP Example – it is similar to php one just gethint.asp is used instead rest all same.

AJAX Database Example - <https://www.w3schools.com/js/tryit.asp?filename=tryjs_ajax_database>

The following example will demonstrate how a web page can fetch information from a database with AJAX. When a user selects a customer in the dropdown list above, a function called showCustomer() is executed. The function is triggered by the onchange event. he page on the server called by the JavaScript above is a PHP file called "getcustomer.php".

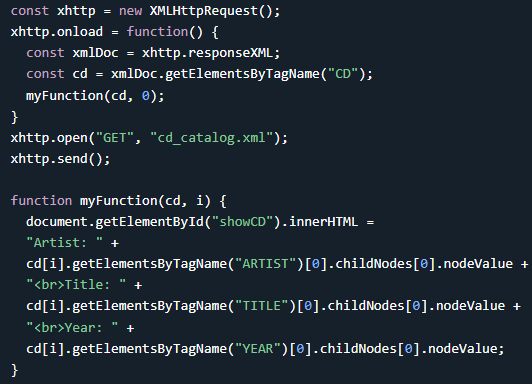
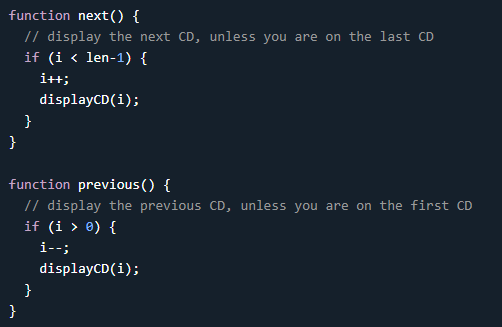
The source code in "getcustomer.php" runs a query against a database, and returns the result in an HTML table

XML Application - CD collection Displayer

Display the First CD in an HTML div Element (A)

And Navigate Between the CDs (B) - <https://www.w3schools.com/js/tryit.asp?filename=tryjs_ajax_app_navigate>

Show Album Information When Clicking On a CD - <https://www.w3schools.com/js/tryit.asp?filename=tryjs_ajax_app>

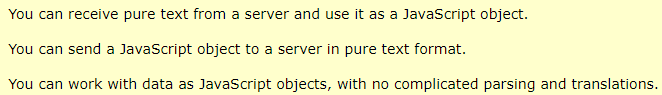
JSON - JSON stands for JavaScript Object Notation. JSON is a lightweight data-interchange format. JSON is plain text written in JavaScript object notation. JSON is used to send data between computers. JSON is language independent \*.

If you parse the JSON string with a JavaScript program, you can access the data as an object.

Why Use JSON? The JSON format is syntactically similar to the code for creating JavaScript objects. Because of this, a JavaScript program can easily convert JSON data into JavaScript objects.

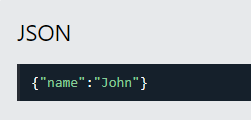
Since the format is text only, JSON data can easily be sent between computers, and used by any programming language. JavaScript has a built in function for converting JSON strings into JavaScript objects: JSON.parse()

JavaScript also has a built in function for converting an object into a JSON string: JSON.stringify()



Data is in name/value pairs. Data is separated by commas In JSON, keys must be strings

Curly braces hold objects. Square brackets hold arrays

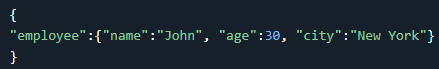
 The following JSON and XML examples both define an employees object, with an array of 3 employees:



In JSON, values must be one of the following data types: a string, a number, an object (JSON object), an array, a Boolean, null

JSON values cannot be one of the following data types: a function, a date, undefined

Object in json:

 can be T & F also  or ,

Values in JSON can be arrays. 

A common use of JSON is to exchange data to/from a web server. When receiving data from a web server, the data is always a string. Parse the data with JSON.parse(), and the data becomes a JavaScript object.

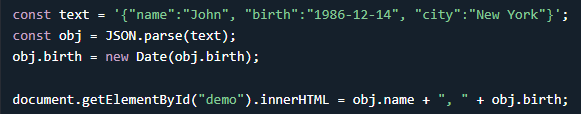
Imagine we received this text from a web server: '{"name":"John", "age":30, "city":"New York"}'

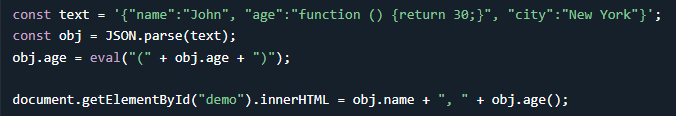
Use the JavaScript function JSON.parse() to convert text into a JavaScript object: 

Use the JavaScript object in your page:

 or 

Date objects are not allowed in JSON. If you need to include a date, write it as a string. You can convert it back into a date object later:

 Same case with the functions -



You should avoid using functions in JSON, the functions will lose their scope, and you would have to use eval() to convert them back into functions.

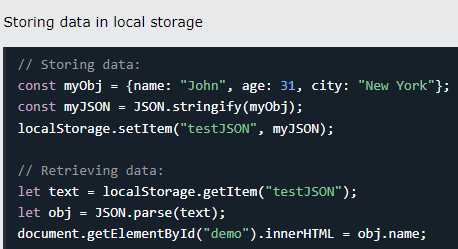
Convert a JavaScript object into a string with JSON.stringify().

Use the JavaScript function JSON.stringify() to convert it into a string. 

myJSON is now a string, and ready to be sent to a server. Same with arrays



When storing data, the data has to be a certain format, and regardless of where you choose to store it, text is always one of the legal formats.



In JSON, date objects are not allowed. The JSON.stringify() function will convert any dates into strings.



In JSON, functions are not allowed as object values. The JSON.stringify() function will remove any functions from a JavaScript object, both the key and the value:



**Note - This can be omitted if you convert your functions into strings before running the JSON.stringify() function.**

This is a JSON string -🡪 

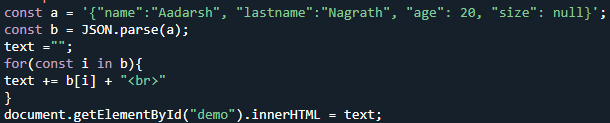
This is a JSON object literal 🡪JSON object literals are surrounded by curly braces {}. JSON object literals contains key/value pairs.

Keys must be strings, and values must be a valid JSON data type

It is a common mistake to call a JSON object literal "a JSON object". JSON cannot be an object. JSON is a string format. Accessing an object –

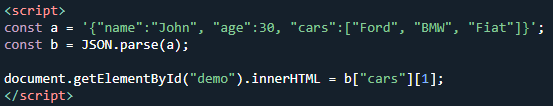


Looping through Properties –

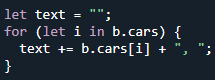


JSON array literal – ‘array literal’ (JSON string)| Literal 🡪

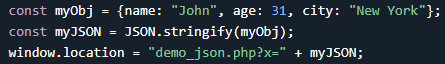
You can create a JavaScript array by parsing a JSON string that contains JSON array literal.



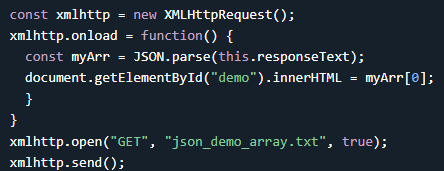
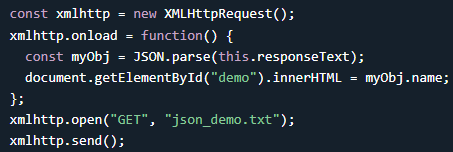
Objects can contain arrays, like above – “cars” : [ ] and their elements can also be accessed

 U can also loop through array within object literal of JSON

If you have data stored in a JavaScript object, you can convert the object into JSON, and send it to a server:

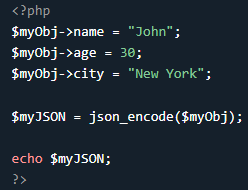
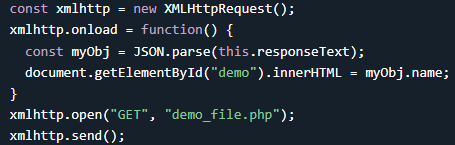


You can request JSON from the server by using an AJAX request. As long as the response from the server is written in JSON format, you can parse the string into a JavaScript object.



When using the JSON.parse() on JSON derived from an array, the method will return a JavaScript array, instead of a JavaScript object. (B above)

The PHP File 🡪PHP has some built-in functions to handle JSON. Objects in PHP can be converted into JSON by using the PHP function json\_encode():

Here is a JavaScript on the client, using an AJAX call to request the PHP file from the (B).

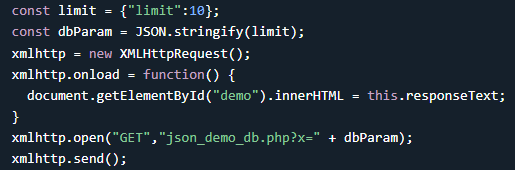
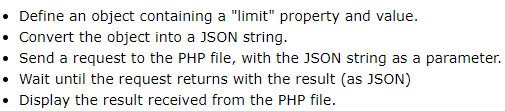
Same with the array in PHP - $myArr =array(“John”,”Aadi”,….)

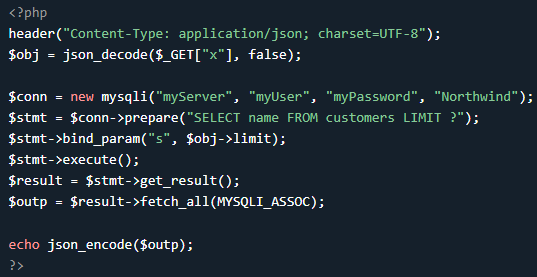
PHP and database – AVOID ( SIR KE UPER SE JAA RAHA YE)

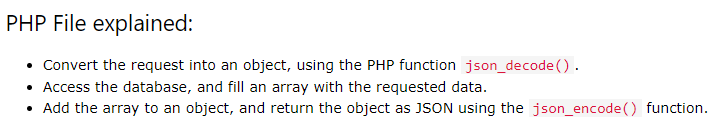
PHP is a server side programming language, and can be used to access a database.

Imagine you have a database on your server, and you want to send a request to it from the client where you ask for the 10 first rows in a table called "customers".

On the client, make a JSON object that describes the numbers of rows you want to return. Before you send the request to the server, convert the JSON object into a string and send it as a parameter to the url of the PHP page.



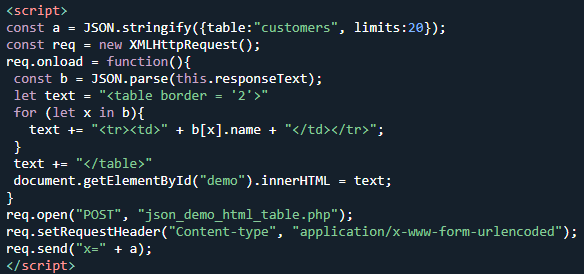
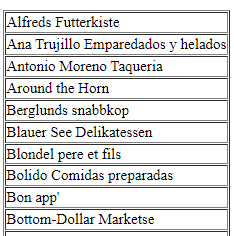


PHP Method = POST When sending data to the server, it is often best to use the HTTP POST method. To send AJAX requests using the POST method, specify the method, and the correct header. The data sent to the server must now be an argument to the send() method.

Eg - <https://www.w3schools.com/js/tryit.asp?filename=tryjson_php_db_post>

The only difference in the PHP file is the method for getting the transferred data WHICH IS POST METHOD, REST IS SAME.

Make an HTML table with data received as JSON

  Dynamic HTML Table- Make the HTML table based on the value of a drop down menu 

Example - <https://www.w3schools.com/js/tryit.asp?filename=tryjson_html_table_dynamic>

JSONP is a method for sending JSON data without worrying about cross-domain issues. JSONP does not use the XMLHttpRequest object. JSONP uses the <script> tag instead.



Requesting a file from another domain can cause problems, due to cross-domain policy. Requesting an external script from another domain does not have this problem. JSONP uses this advantage, and request files using the script tag instead of the XMLHttpRequest object.



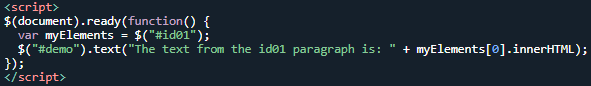
WE ARE NOT GOING INTO DETAIL OF THIS

JavaScript / jQuery DOM Selectors

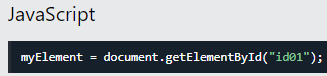
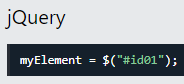
jQuery was created in 2006 by John Resig. It was designed to handle Browser Incompatibilities and to simplify HTML DOM Manipulation, Event Handling, Animations, and Ajax.

Finding HTML Element by Id

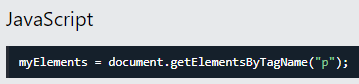
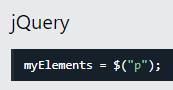
 id=”id01”> Hello World! </…







Finding HTML Elements by Tag Name



Finding HTML Elements by Class Name

