For a array like:-

Var myList = [‘apples’,’óranges’,’bananas’];

myList.forEach(function(value,index){

//do something like

Alert(“my shopping list has ” + value + “in it”);

});

//this is absolutely valid but all the browsers don’t support it since forEach is a newer method

In JS you can access the html things due to the DOM model….and methods of the same…  
like getElementByID(‘p’);

//it returns an array of all the paragraphs …

getElementsByTagNames(‘p’);

for more on this:-

[https://www.youtube.com/watch?v=W3EK4MlZW4g](https://www.youtube.com/watch?v=W3EK4MlZW4g&list=PLoYCgNOIyGACTDHuZtn0qoBdpzV9c327V&index=5)

java script listeners:-

adding listeners to elements of html…:-

//first grabbing the element by id to a variable…Ex:-

Var firstTextBox = document.getById(‘first-t-box’);

firstTextBox.addEventListener(“click”,function() {

alert(“the element is clicked!!”);

});

//like this …whenever for the element that event is happened then the function beneath it is called

//list of some listeners:-

//all are in small

Click

Mouseenter

Mouseleave

Mousedown

Mouseup

Mousemove

Keyup

Keydown

Blur

Focus

//etc…

We can also put the function outside and put as a parameter, such as:-

Var firstTextBox = document.getById(‘first-t-box’);

firstTextBox.addEventListener(“click”,go);

Function go() {

alert(“the element is clicked!!”);

}

….

So that you could use the function go for multiple html elements

After getting any element by Id…we can use that variable’s one method called innerHtml = “something”

What is basically does is it puts that thing inside that pair of tags defined by our varbale.

---

no undeclared vars,

almost no == all ===,

.foreach preferred over traditional for loop,

no two variables sharing same declaration i.e. decalred on same line and therefore been provided different declarations on different lines.

strings should be in double quotes

last element in json should not have a seprating or trailing comma.

paranthesis of functions should not have a space between funtion keyword and paranthesis.

expects semicolon after end of every statement.

unused vars warn,

else on the line where if ends,

naming conventions:-

var names, property names, function names should be in camel case as expected.

constants are snakecased but with all alphas in caps.

filename used snake-case too.

convetions:-

prefer .bind(this) instead of declaring some variable as say self = this.

prefered and not compulsory since sometimes you need to context i.e. the original this at that scope and the outer this of the outer scope indeed, so at these type of situations we need to do something like var self= this or we might pass this in the paramter, as in we can technically do it too.

setTimeout does not say that that function would execute after that milli seconds, it says the minimum time in milli seconds after which it would be pushed ot event dispatcher, and after the clearance of call stack it would be pushed to call stack and executed. Generally the delay between the time specified and the actual time taken is very less , just few milli seconds so we hardly ever notice it. But it could be there for big projects

---

zeus notes:-

The constructor property returns the constructor function for all JavaScript variables.

Automatic String Conversion.

Automatic Type Conversion.

We can set timeout easily with javascript for any fucntion if we think that it might take some time so we can stop and make a timeout for it.

Hoisting is a javascript's default behaviour of moving all declarations to the top of the current scope.

Only the declaration is moved to the top of the declared scope and not the initialization.

Similarly all functions are also moved or hoisted to top of the declaration scope, and the body along with the function name is also moved to top which means it's initialization is also moved to the top along with the declaration unlike the variable.

Function expressions are more like variables, only the declaration gets hoisted and not the initialization.

It is not advisable to use function expression since it creates a lot of confusion when the function is created within if else block.

Avoid using delete since its behavior isnt consistent.

All primitive types in javascript are immutable , and all derived ones are mutable i.e. they work on reference and not on the actual copy.

The call() method is a predefined JavaScript function method.

It can be used to invoke (call) a function with an owner object as the first argument (parameter).

The only difference is:

call() takes any function arguments separately.

apply() takes any function arguments as an array.

The apply() method is very handy if you want to use an array instead of an argument list.

Splice method is used to replace a range of values from an array to another one.

localeCompare method can be used to comapre strings.

Sort method can take function as an input so as to sort in a custom way.

Every method of array can come handy to check conditions for a complete array to check whether complete array passes same test.

This refers to in global scope as window object and in function as the object, if using this, otherwise there also its window object.

Namespaces is almost like object only, not much difference.

Javascript contains no keyword class, we need to create class by function only.

They parameters passed would be the parameters for the so called constructor.

And creating methods of the class is by creating 'this.method-name = function(){}'

private members and functions are the ones which have local function scope.

Static members and functions can be made by directly creating and accesing them by class or so called function name itself.

RegEx is used to validatea form using js. First part is the expression and the following part is the mode which is optional.

SessionStorage can be used to store variable even after refresh, and lasts till the browser is closed.

tr:nth-child(even) {background: #CCC}

tr:nth-child(odd) {background: #FFF}

sessionStorage.clear() to free it up.

Input = date can be used but it's format which it displays cannot be changed.

RegEx can be created in two ways

var patt = /w3schools/i

either that or

new RegExp("(x|y)")

we cannot store array in sessionStorage so we can do a trick, we can make the data which we want to store in array we can make it in json, and then use stringify to make it string .

Then we can save that string in sessionStorage.

And at the time of retrival we can retrive that string and use json.parse and then we can access every thing as json object's elements

Event listeners can be directly added in js by statement:-

object.addEventListener(eventListner, myScript);

once unload event is triggered :-

All the resources still exist (img, iframe etc.)

Nothing is visible anymore to the end user

UI interactions are ineffective (window.open, alert, confirm, etc.)

An error won't stop the unloading workflow

we could use position: sticky for footers

for dragging and dropping we have amazingevent handlers like:-

Events fired on the draggable target(the source element):

ondragstart- occurs when the user starts to drag an element

ondrag - occurs when an element is being dragged

ondragend occurs when the user has finished dragging the element

Events fired on the drop target:

ondragover occurs when the dragged element is over the drop target

ondragenter occurs when the dragged element enters the drop target

ondragleave occurs when the dragged element leaves the drop target

ondrop occurs when the dragged element is dropped on the drop target

preventDefault():The preventDefault() method cancels the event if it is cancelable, meaning that the default action that belongs to the event will not occur.

onkeydown

onkeypress

onkeyup

---

promise is a js thing which is like a asyn function, explaining by a example:-

var p = new Promise(function(resolve,reject) {

// do some async task here first

if(saySomeBool) {

setSomethingHEre = 1;

}

else{

setSomethingHere = 0;

}

//basically we can do anything here we want, this would be executed first and then the then part , asyncronously

});

p.then(function() {

//do something with setSomethingHere kinda thing

}).catch(function() {

//err handling

})

best way to loop through objects is to convert them to array using one the three methods and the loopin ginto an array just like we do using "for of" loop. Three methods that we can use are Object.keys, Object.values, or Object.entries. example:-  
  
const fruits = {

apple: 28,

orange: 17,

pear: 54,

}

const entries = Object.entries(fruits)

console.log(entries)

// [

// [apple, 28],

// [orange, 17],

// [pear, 54]

// ]

// now to loop it we can do as :-

for (const [fruit, count] of entries) {

console.log(`There are ${count} ${fruit}s`)

}

// Result

// There are 28 apples

// There are 17 oranges

// There are 54 pears

similarly for other both methods