1. Html APIS: Geolocation, Audio, Video, Fetch, WAAPI, websockets, localstorage

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

element.insertAdjacentHTML('beforeend', '<div>two</div>');

insertADjacentText

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

<script id="some-template" type="text/html">

<p>

Some text

</p>

</script>

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

element.getComputedStyle(); // Gets the css style

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

element.style.cssText = "margin:0 0 0 20px" // Sets the css style

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

array.find();

array.map();

array.filter();

array.reduce();

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

var newObject = {...oldObject, new\_prop: "This the is the adde prop, to the copied object"}

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

Object.is(uno, dos); // I returns true if passed variables reference the same object

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

html <progress></progress>

-----MONGOOSE

Object Document Mapper https://app.pluralsight.com/player?course=mongoose-for-nodejs-mongodb&author=mark-scott&name=mongoose-for-nodejs-mongodb-m1&clip=2&mode=live

-----------NPM

-Add npm commands

-calc css vw

unitsvw: hundredths of the viewport width.

vh: hundredths of the viewport height.

vmin: hundredths of whichever is smaller, the viewport width or height.

vmax: hundredths of whichever is larger, the viewport width or height.

-calc function

-Tilde : ASCII 126 &tilde; ~

-Caret: CODIGO ASCII 94 &Hat;

-document write

-documetn create

-delete

-add string.replace method

-sockets

-setintervals

-object details: https://www.udemy.com/the-complete-javascript-course/learn/v4/t/lecture/5869112?start=0

-object details2: https://www.udemy.com/the-complete-javascript-course/learn/v4/t/lecture/10788440?start=420

-node express api code

-add while examples

-if continue https://www.udemy.com/the-complete-javascript-course/learn/v4/t/lecture/5869114?start=900

-for break https://www.udemy.com/the-complete-javascript-course/learn/v4/t/lecture/5869114?start=1020

-for loop backwards

-DIV contentEditable=true

-css scroll customizing

-function statement vs function expression returns value (uno does anything and the other) https://www.udemy.com/the-complete-javascript-course/learn/v4/t/lecture/5869106?start=0

-make excercise of hoisting

-SCOPE CHAIN [When using a variable it is important to think in its scope chan rather than the execution stack] : https://www.udemy.com/the-complete-javascript-course/learn/v4/t/lecture/5869134?start=525

-this. variable: https://www.udemy.com/the-complete-javascript-course/learn/v4/t/lecture/5869136?start=0

https://www.udemy.com/the-complete-javascript-course/learn/v4/t/lecture/5869138?start=112

-Functions with no parens (as a callback) https://www.udemy.com/the-complete-javascript-course/learn/v4/t/lecture/5869158?start=345

-Objects and inheritance: https://www.udemy.com/the-complete-javascript-course/learn/v4/t/lecture/5869180

-WEB WORKERS...

-WEB COMPONENTS

investigate on:

-formControlName="someControlElementName"

-[formGroup]="someForm"

-someForm.get('someControlElementName')

-someForm.setValue({...});

-someForm.patchValue({...});

-this.formBuilder.group({...}); //easier to replace new FormGroup creation

-Examples of validators: [Validators.required, Validators.minLengt(3)]

-myControl.setValidators([Validators.required, Validators.maxLength(30)]) //To update validators, after we set the new rules, we need to refresh validation

-myControl.clearValidators();//To update validators, after we clear the rules, we need to refresh validation

-myControl.updateValueAndValidity();

-modulo6-video2: valueChanges.subscribe

-Changing the error mesaages from the template to the component

-...valueChanges.debounceTime(1000).subscribe //A delay in showing the error

-Creating a formsArray : this.myArray= this.fb.array([...]);

express:

app.listen(3000); to run our application on the especified port

app.use(cors()); // to connect our backend with the frontend

nodemon

bodyparser

mogoose

<router-outlet> to contain any component difned by the router

async

await

localStorage.setItem()

var users = await User.find({}, `-password -\_\_v`);

style="cursor: pointer"

endpoint definition

arrow Function " => " changes the scope of this

relationship 1 to many: author: { type: mongoose.Schema.Types.ObjectId, ref: 'User' }

-Javascript storing things by value, looking at was was stored in memory

-storing things by reference, looking to what was stored in objects and arrays. if the original changes, the copy changes too

-All arrays are objects

-myArray.forEach(); //Investigate how that function works

-Statememts can have labels like loop: for (){}; ... break loop;

-throw

-try and catch

-class

-function with no function keyword

-black box testing vs white box testing

-how to makea public property : https://app.pluralsight.com/player?course=organizing-javascript-functionality&author=kyle-simpson&name=b35d2756-c552-419d-9917-e4223d0b7ab9&clip=3 minute1:50

naming convention, same above 3:35, camel case functions and snake case properties

function as first class value

Working with "rel" instead of ID's or attaching "data" attributes https://app.pluralsight.com/player?course=organizing-javascript-functionality&author=kyle-simpson&name=5d59fbd0-5adb-4efb-babb-3108f2578d71&clip=1

-preventDEfault

-work with modern ajax

-try avoid using specific dom elemnts, by tag, by id, by class

-adding a css anmation triggered by JS

-jasvascript event.target

-debugger; statement to create debugging things

-how to create a module: https://app.pluralsight.com/player?course=organizing-javascript-functionality&author=kyle-simpson&name=148b1388-10f5-4b8a-b01e-b572cd3ba2fc&clip=0

-passing arguments to an IIFE and using global scope: https://app.pluralsight.com/player?course=organizing-javascript-functionality&author=kyle-simpson&name=148b1388-10f5-4b8a-b01e-b572cd3ba2fc&clip=2

-DOMContentLoaded: https://developer.mozilla.org/es/docs/Web/Reference/Events/DOMContentLoaded

-Organizing modules :https://app.pluralsight.com/player?course=organizing-javascript-functionality&author=kyle-simpson&name=148b1388-10f5-4b8a-b01e-b572cd3ba2fc&clip=4

We must have all of our js modules, al of them will have their public api, and then we will have an app.js that will load our modules.

Code of the module loader app.js:

<script>

document.addEventListener("DOMContentLoaded", function (event) {

carousel.init();

profiles.init();

details.init();

});

</script>

-Investigate PubSub: https://app.pluralsight.com/player?course=organizing-javascript-functionality&author=kyle-simpson&name=148b1388-10f5-4b8a-b01e-b572cd3ba2fc&clip=5

USING EVENTS FOR ABSTRACTION

always using strict mode

gulp or grunt

HTTP2 not optimizing with old school meythods

//---------For Operators & Types

i+=5; //Increments 5 after used in an operation.

i++; //Increments 1 after used in an operation.

++i; //Increments 1 before used in an operation.

and && or || not ! ;

typeof myVar; //Regresa el tipo de una variable.

isNaN('berry')//Nos dara un true;

//--------String processing

str1.concat(" string 2"); //Concat strings, just like "+".

str1.charAt(3); //Returns the character specified as argument.

str1.indexOf("ab"); //Returns the position of the first occurrence of a specified value in a string, returns -1 if does not find it.

str1.lastIndexOf("ab"); //Returns the position of the last occurrence of a specified value in a string, returns -1 if does not find it.

str1.split("");//Is used to split a string into an array of substrings, and returns the new array. If an empty string ("") is used as the separator, the string is split between each character.

str1.length;

str1.substring(0,9);

var xx = parseInt("40 years"); //Results 40, it parses a string and returns an integer.

str1.toLowerCase();

str1.toUpperCase();

str1.(searchvalue,newvalue); //Searches a string for a specified value, or a regular expression, and returns a new string where the first specified value is replaced. To replace all occurrences of a specified value, use the global (g) modifier.

str1.replace(/allInstancesa/g, "New instances"); //Performs a global search and replacement.

str1.replace(/allInstancesa/gi, "New instances"); //Performs a global search and replacement, it is CaSe iNSensiTive.

//--------Number processing

numberVar = Math.random(); //Return a random number between 0 (inclusive) and 1 (exclusive).

numberVar = Math.floor(1.6); //Round a number downward to its nearest integer.

num.toFixed(2); // Converts a number into a string, keeping a specified number of decimals.

//--------DOM

.nodeName; // Returns in uppercase the name of the node (document, element, attribute, comment or text).

.tagName; //Returns the tag name of the element. In HTML, the returned value of the tagName property is always in UPPERCASE. This property is read-only.

.nodeType; // Returns the number of the type of the node (document-9, element-1, attribute-2, comment-8 or text-3).

.nodeValue; // The node value, not defined for some node types

.ownerDocument;// Document Referencia del documento al que pertenece el nodo

.firstChild;// Returns the first child node of the specified node (element, comment or text).

.firstElementChild; //Returns the first child element of the specified element.

.lastChild; // Returns the last child node of the specified node (element, comment or text).

,lastElementChild; //Returns the last child element of the specified element.

.childNodes; //Returns a collection of a node's child nodes, as a NodeList object.

.children; //Returns a collection of an element's child elements, as an HTMLCollection object.

.previousSibling; //Returns the previous node of the specified node, in the same tree level.

.previousElementSibling; // Returns the previous element of the specified element, in the same tree level.

.nextSibling; //Returns the node immediately following the specified node, in the same tree level.

.nextElementSibling; //Returns the element immediately following the specified element, in the same tree level.

.attributes; //Returns a collection of attributes of the specified element node, as a NamedNodeMap object.

.hasChildNodes(); // No Parameters, Returns true if the specified node has any child nodes, otherwise false.

.appendChild(nodo); // Appends a node as the last child of a node.

borraMe.parentNode.removeChild(borraMe); // Removes a specified child node of the specified element, returns the removed node as a Node object.

.replaceChild(nuevoNodo, anteriorNodo); // Replaces a child node with a new node. The new node could be an existing node in the document, or you can create a new node.

.insertBefore(nuevoNodo,anteriorNodo); // Inserts a node as a child, right before an existing child, which you specify.

.createElement("BUTTON"); //Creates an Element Node with the specified name.

.createComment("text")// Creates a Comment node with the specified text.

.createTextNode("text");// Creates a Text Node with the specified text.

.cloneNode();// Creates a copy of a node, and returns the clone, clones all attributes and their values.

.cloneNode(true);// Clone the node, its attributes, and its descendants.

.getAttribute("href"); // Returns the value of the attribute with the specified name, of an element.

.createAttribute("href"); //Creates an attribute with the specified name, and returns the attribute as an Attr object.

.setAttribute("href", "www.google.com"); //adds the specified attribute to an element, and gives it the specified value. If the specified attribute already exists, only the value is set/changed.

.setAttributeNode(att); //Adds the specified attribute node to an element. If the specified attribute already exists, this method replaces it.

.innerHTML; // Sets or returns the HTML content (inner HTML) of an element.

.textContent; // Sets or returns the textual content of the specified node, and all its descendants.

.style.property= "new style";

document.documentElement; //Root element of the page, object HTMLElement.

document.body - //The body of the document

.querySelector('#ejemplo1 > #dos.dos'); //Gets the first element that matches selector specified.

.querySelectorAll(".example"); //Gets all elements in the document that match selector specified.

.getElementById("identiFy"); // Returns the element that has the ID attribute with the specified value

.getElementsByTagName("div"); // Returns a node list (collection/array of nodes) containing all elements with a specified tag name.

.getElementsByClassName("container"); // Returns a node list (collection/array of nodes) containing all elements with a specified class.

href="javascript:window.print()" //Make a link that print the screen

onclick="window.print(); return false;"//Make any element that pront the screen when clicked.

//--------Tables

.rows; //Returns a collection of all <tr> elements in a table/tbody.

.tBodies; //Returns a collection of all <tbody> elements in a table.

.createCaption(); //Creates an empty <caption> element and adds it to the table.

.createTFoot(); //Creates an empty <tfoot> element and adds it to the table.

.createTHead(); //Creates an empty <thead> element and adds it to the table.

.deleteCaption(); //Removes the first <caption> element from the table.

.deleteRow(0); //Removes a row (<tr>) from the table.

.deleteTFoot(); //Removes the <tfoot> element from the table.

.deleteTHead(); //Removes the <thead> element from the table.

.insertRow(0); //Creates an empty <tr> element and adds it to the table.

tableObject.cells; //Returns a collection of all <td> or <th> elements in a table row. The elements in the collection are sorted as they appear in the source code.

tablerowObject.insertCell(0); //Inserts a cell into the current row at the specified position.

tablerowObject.deleteCell(0);//Deletes a cell in the current table row in the specified index.

//--------Loops & Comparisons

while(){};

do{}while();

switch(variable a comparar){ case 5: console.log('Este merito fue'); break; default: console.log('Este es el default'); break;}

for(var i in objeto o array)//El for in no sirve en arrays u objetos con propiedades numericas, solo para objetos con propiedades nombradas como strings, mejor usar un loop "for" normal

//--------Functions

return variAble; //This is the sintax.

(function(){alert('Hola');})(); //Self Invoking function

alert(arguments.callee); //(Should be execcuet inside of a Function) Property "arguments" creates an array of the arguments of the function, and "callee" in this case prints the function itself.

//--------Messaging

console.log('Prints anything on the console of the browser');

confirm('Aceptar');// PopUp that requests user to accept by clicking on "Accept" button, will return true, or false in case the user clicks on "Cancel".

prompt('Introducir valores');// PopUp that requests user to enter text in a field, entered text will eb stored in a variable.

//--------BOM

document.URL; // Gets the url string of the page.

document.referrer; // Returns the URL of the document that loaded the current document.

document.lastModified; //Returns the date and time the current document was last modified. This property is read-only.

window.open("www.google.com","\_blank","toolbar=yes, titlebar=no,location=yes, directories=no, status=no, menubar=yes, scrollbars=yes, resizable=no, copyhistory=yes, width=650, height=900, left=660, top=0");

var timeID = setTimeout(functionNoParentesis, 3000,param1,param2,...); //Calls a function or evaluates an expression after a specified number of milliseconds.

clearTimeout(timeID); //clears a timer set with the setTimeout() method. The ID value returned by setTimeout() is used as the parameter for the clearTimeout() method.

var timeID = setInterval(functionNoParentesis, 3000,param1,param2,...); //Calls a function or evaluates an expression at specified intervals (in milliseconds).

clearInterval(timeID); //Clears a timer set with the setInterval() method. The ID value returned by setInterval() is used as the parameter for the clearInterval() method.

//--------Document

document.embeds; //Returns a collection of all <embeds> elements in the document. The elements in the collection are sorted as they appear in the source code.

document.forms; //Returns a collection of all <form> elements in the document. The elements in the collection are sorted as they appear in the source code.

document.images; //Returns a collection of all <img> elements in the document. The elements in the collection are sorted as they appear in the source code.

document.links; //The links collection returns a collection of all links in the document. The links in the collection represents <a> elements and/or <area> elements with a href attribute. If the element is missing the href attribute, nothing is returned. The elements in the collection are sorted as they appear in the source code.

document.title; //Sets or returns the title of the current document (the text inside the HTML title element).

//--------Location

location.hash = "part5"; // Sets or returns the anchor part of a URL, including the hash sign (#). Note: When this property is used to set the anchor part, do not include the hash sign (#).

location.host; or location.host="www.ejemplo.com"; //Sets or returns the hostname and port of a URL. If the port number is not specified in the URL (or if it is the scheme's default port - like 80, or 443), some browsers will not display the port number.

location.hostname; or location.hostname="ejemplo.com"; //Sets or returns the hostname of a URL.

location.href; location.href="http://www.example.com/default.htm";// Sets or returns the entire URL of the current page.

location.pathname; or location.pathname=path; // Sets or returns the pathname of a URL.

location.port; or location.port=2095;// Sets or returns the port number of a URL. If the port number is not specified in the URL (or if it is the scheme's default port - like 80, or 443), some browsers will display 0 or nothing.

location.protocol; or location.protocol="https:"; //Sets or returns the protocol of the current URL, including the colon (:). The protocol is a standard that specifies how data are transmitted between computers.

locatin.search; or location.search="?email=someone@example.com"; //Sets or returns the querystring part of a URL, including the question mark (?) The querystring part is the part of the URL after the question mark (?). This is often used for parameter passing.

location.assign("http://www.w3schools.com"); //Loads a new document.

location.reload(true); //Does the same as the reload button in your browser. By default, the reload() method reloads the page from the cache, but you can force it to reload the page from the server by setting the forceGet parameter to true: location.reload(true).

location.replace("http://www.w3schools.com"); //Replaces the current document with a new one. The difference between this method and assign(), is that replace() removes the URL of the current document from the document history, meaning that it is not possible to use the "back" button to navigate back to the original document.

//--------Navigator

navigator.appCodeName; //Returns the code name of the browser. Note: All modern browsers returns "Mozilla", for compatibility reasons.

navigator.appName; //Returns the name of the browser.

navigator.appVersion; //Returns the version information of the browser.

navigator.cookieEnabled; //Returns a Boolean value that specifies whether cookies are enabled in the browser.

document.cookie; = "username=John Doe"; // Creating a cookie.

navigator.geolocation; //Returns a Geolocation object that can be used to locate the user's position. Since this can compromise user privacy, the position is not available unless the user approves it.

navigator.language; //Returns the language of the browser.

navigator.onLine; //Determines whether the browser is online.

navigator.platform; //Returns for which platform the browser is compiled.

navigator.product; //This property may not work as expected: All browsers returns "Gecko".

navigator.userAgent; //Returns the user-agent header sent by the browser to the server.

navigator.javaEnabled(); //Returns a Boolean value that specifies whether the browser has Java enabled.

taintEnabled() //Removed in JavaScript version 1.2. Specifies whether the browser has data tainting enabled.

//--------Screen

screen.availHeight; //Returns the height of the screen (excluding the Windows Taskbar).

screen.availWidth; //Returns the width of the screen (excluding the Windows Taskbar).

screen.colorDepth; //Returns the bit depth of the color palette for displaying images.

screen.height; //Returns the total height of the screen.

screen.pixelDepth; //Returns the color resolution (in bits per pixel) of the screen.

screen.width; //Returns the total width of the screen.

//--------Events

element.addEventListener("click", displayDate, false);/\*Attaches an event handler to an element without overwriting existing event handlers.

You can add many event handlers to one element.

You can add many event handlers of the same type to one element, i.e two "click" events.

You can add event listeners to any DOM object not only HTML elements. i.e the window object.

The addEventListener() method makes it easier to control how the event reacts to bubbling.\*/

<a href="http://www.google.com" onclick="event.preventDefault()"// Cancels the event if it is cancelable, meaning that the default action that belongs to the event will not occur.

//--------Forms

sistema = document.form1.sistema; //Primero Seleccionamos la etiqueta Select dentro de la forma

siSel = sistema.selectedIndex; //Luego obtenemos cual es el Option seleccionado

vaSis = sistema[siSel].value; // Ahora obtenemos el valor del Option

document.getElementsByTagName("input")[0].focus();//Give focus to an <input> element

.setSelectionRange(0,500); // Selects from the focused input a portion of the text inside.

//--------OBJECTS

var myObj = {}; // Notacion Literal de Objeto

var myObj = new Object(); // Constructor de objeto

myObj['name'] = 'Fabian';

myObj.age = 28;

var myArray = [21, true, 'Fabian', [myObj.name] ];//Un objeto dentro de un array

var myObject = { //Un objeto con un array dentro

name: 'Eduardo',

type: 'Most excellent',

interests: [fabian[1]]

};

//Objetos en Objetos

var friends = {

bill:{

firstName: "Bill",

lastName: "Gates",

number: "(206) 555-5555",

address: ['One Microsoft Way','Redmond','WA','98052']}

};

//------------------------------------------------------

function Persona (name, lastname, age, address){

this.nombre = name;

this.apellido = lastname;

this.edad = age;

this.direccion = address;

};

var fabian = new Persona ("Fabian", "Alvarado", "29", "Genova 15");

var diReccion = 'direccion';

console.log(fabian[diReccion]);

console.log( anyObj.hasOwnProperty('name') ); // True o False dependiendi si tiene una propiedad en especifico

//-------------------

anyObj.prototype.calcular = function() { //Esto es para a la clase anyObj agregarle el metodo calcular

statements;

};

//---------------

anyObj.prototype = new Animal(); // Esto es para que la clase anyObj herede todo de Animal.

myFunction.call(myObject, "My name", 33); //Used to invoke a function, must have the owner object as first parameter, takes the function arguments separately by comas.

myFunction.apply(myObject, ["My name", 33]); //Used to invoke a function, must have the owner object as first parameter, takes the function arguments in an array.

//--------ARRAYS

var aRray = [5, 'hola', '56'];

array.length;

var yo = [[5,6,7,8] , ['Tu','Yo', 'Nosotros'] , [3]];

console.log(yo[1][2]);

array1.concat(array2, array3, array4);//This method is used to join two or more arrays.

array1.join();//This method joins the elements of an array into a string, and returns the string.The elements will be separated by a specified separator. The default separator is comma (,).

array1.pop();//Removes the last element of an array, and returns that element. This method changes the length of the original array.

array1.shift();//Removes the first element of an array, and returns that element. This method changes the length of the original array.

array1.push("something1", "something2");//Adds new items to the end of an array, and returns the new length. The new item(s) will be added at the end of the array.

array1.unshift("something1", "something2");//Adds new items to the beginning of an array, and returns the new length. The new item(s) will be added at the end of the array.

array1.reverse(); //Reverses the order of the elements in an array.

array1.filter(function(currentValue, index, arr), thisValue); //Creates an array filled with all array elements that pass a test (provided as a function).It does not execute the function for array elements without values. It does not change the original array.

//--------SNIPETS

//ASI SE INCLUYE UN JS DENTRO DEL DOCUMENTO CON JS

function addJavascript(jsname,pos) {

var th = document.getElementsByTagName(pos)[0];

var s = document.createElement('script');

s.setAttribute('type','text/javascript');

s.setAttribute('src',jsname);

th.appendChild(s);

}

addJavascript('newExternal.js','body');

addJavascript('newExternal2.js','head');

//Segunda Opcion \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

var th = document.getElementsByTagName('head')[0];

var s = document.createElement('script');

s.setAttribute('type','text/javascript');

s.setAttribute('src','/Content/global/scripts/js/video/video-colorbox.js');

th.appendChild(s);

////////////////Invertir cadena////////////////////

function invertir(cadena) {

var x = cadena.length;

var cadenaInvertida = "";

while (x>=0) {

cadenaInvertida = cadenaInvertida + cadena.charAt(x);

x--;

}

return cadenaInvertida;

}

invertir('fabian');

//////////////////

//COMO OBTENER LA LISTA DE ELEMENTOS DE UN TIPO DE UNA PAGINA

var rp = document.getElementsByTagName('script');

var num = 'lista';

for(var xx = 0; xx < rp.length; xx++){

num = num + '<br>' + rp[xx].src;

}

$('body').html(num);

//document.write(num);

//////////

//COMO OBTENER LIGAS DE UNA PAGINA Y SUS HREFS ////////////////////

var rp = document.getElementsByTagName('a');

var num = 'lista';

for(var xx = 0; xx < rp.length; xx++){

num = num + '<br/><br/><br/>' + rp[xx].innerHTML + ' <br/> ' + rp[xx].href;

}

$('body').html(num);

//document.write(num);

////////////////

//Reemplazar links de una pagina ////////////////////////////////////////////////////////////////////////

var links = document.getElementsByTagName("a");

var linkUrl = "";

var urlAdfly = "";

var urlFinal = "";

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

linkUrl = links[i].href;

urlAdfly = linkUrl.substring(0,14);

if(urlAdfly == 'http://adf.ly/'){

links[i].href = linkUrl.substring(22,(linkUrl.length));

console.log(links[i].href);

};

};

///PLAN FOR THE MASTER PIECE GAME

-Multiplayers using sockets

-Use webworkers

-Use WAAPI

-Use Web Components

-Use event abstraction logic – PUBSUB

//---------For Operators & Types

          i+=5; //Increments 5 after used in an operation.

          i++; //Increments 1 after used in an operation.

          ++i; //Increments 1 before used in an operation.

          and && or || not ! ;

          typeof myVar; //Regresa el tipo de una variable.

          isNaN('berry')//Nos dara un true;

//--------String processing

          str1.concat(" string 2"); //Concat strings, just like "+".

          str1.charAt(3); //Returns the character specified as argument.

          str1.indexOf("ab"); //Returns the position of the first occurrence of a specified value in a string, returns -1 if does not find it.

          str1.lastIndexOf("ab"); //Returns the position of the last occurrence of a specified value in a string, returns -1 if does not find it.

          str1.split("");//Is used to split a string into an array of substrings, and returns the new array. If an empty string ("") is used as the separator, the string is split between each character.

          str1.length;

          str1.substring(0,9);

          var xx = parseInt("40 years"); //Results 40, it parses a string and returns an integer.

          str1.toLowerCase();

          str1.toUpperCase();

          str1.(searchvalue,newvalue); //Searches a string for a specified value, or a regular expression, and returns a new string where the first specified value is replaced. To replace all occurrences of a specified value, use the global (g) modifier.

          str1.replace(/allInstancesa/g, "New instances"); //Performs a global search and replacement.

          str1.replace(/allInstancesa/gi, "New instances"); //Performs a global search and replacement, it is CaSe iNSensiTive.

//--------Number processing

          numberVar = Math.random(); //Return a random number between 0 (inclusive) and 1 (exclusive).

          numberVar = Math.floor(1.6); //Round a number downward to its nearest integer.

          num.toFixed(2); // Converts a number into a string, keeping a specified number of decimals.

//--------DOM

          .nodeName; // Returns in uppercase the name of the node (document, element, attribute, comment or text).

          .tagName; //Returns the tag name of the element. In HTML, the returned value of the tagName property is always in UPPERCASE. This property is read-only.

          .nodeType; // Returns the number of the type of the node (document-9, element-1, attribute-2, comment-8 or text-3).

          .nodeValue; // The node value, not defined for some node types

          .ownerDocument;// Document Referencia del documento al que pertenece el nodo

          .firstChild;// Returns the first child node of the specified node (element, comment or text).

          .firstElementChild; //Returns the first child element of the specified element.

          .lastChild; // Returns the last child node of the specified node (element, comment or text).

          ,lastElementChild; //Returns the last child element of the specified element.

          .childNodes; //Returns a collection of a node's child nodes, as a NodeList object.

          .children; //Returns a collection of an element's child elements, as an HTMLCollection object.

          .previousSibling; //Returns the previous node of the specified node, in the same tree level.

          .previousElementSibling; // Returns the previous element of the specified element, in the same tree level.

          .nextSibling; //Returns the node immediately following the specified node, in the same tree level.

          .nextElementSibling; //Returns the element immediately following the specified element, in the same tree level.

          .attributes; //Returns a collection of attributes of the specified element node, as a NamedNodeMap object.

          .hasChildNodes(); // No Parameters, Returns true if the specified node has any child nodes, otherwise false.

          .appendChild(nodo); // Appends a node as the last child of a node.

           borraMe.parentNode.removeChild(borraMe); // Removes a specified child node of the specified element, returns the removed node as a Node object.

          .replaceChild(nuevoNodo, anteriorNodo); // Replaces a child node with a new node. The new node could be an existing node in the document, or you can create a new node.

          .insertBefore(nuevoNodo,anteriorNodo); // Inserts a node as a child, right before an existing child, which you specify.

          .createElement("BUTTON"); //Creates an Element Node with the specified name.

          .createComment("text")// Creates a Comment node with the specified text.

          .createTextNode("text");//  Creates a Text Node with the specified text.

          .cloneNode();// Creates a copy of a node, and returns the clone, clones all attributes and their values.

          .cloneNode(true);// Clone the node, its attributes, and its descendants.

          .getAttribute("href"); // Returns the value of the attribute with the specified name, of an element.

          .createAttribute("href"); //Creates an attribute with the specified name, and returns the attribute as an Attr object.

          .setAttribute("href", "www.google.com"); //adds the specified attribute to an element, and gives it the specified value. If the specified attribute already exists, only the value is set/changed.

          .setAttributeNode(att); //Adds the specified attribute node to an element. If the specified attribute already exists, this method replaces it.

          .innerHTML; // Sets or returns the HTML content (inner HTML) of an element.

          .textContent; // Sets or returns the textual content of the specified node, and all its descendants.

          .style.property= "new style";

          document.documentElement; //Root element of the page, object HTMLElement.

          document.body - //The body of the document

          .querySelector('#ejemplo1 > #dos.dos'); //Gets the first element that matches selector specified.

          .querySelectorAll(".example"); //Gets all elements in the document that match selector specified.

          .getElementById("identiFy"); // Returns the element that has the ID attribute with the specified value

          .getElementsByTagName("div"); // Returns a node list (collection/array of nodes) containing all elements with a specified tag name.

          .getElementsByClassName("container"); // Returns a node list (collection/array of nodes) containing all elements with a specified class.

          href="javascript:window.print()" //Make a link that print the screen

          onclick="window.print(); return false;"//Make any element that pront the screen when clicked.

          //--------Tables

          .rows; //Returns a collection of all <tr> elements in a table/tbody.

          .tBodies; //Returns a collection of all <tbody> elements in a table.

          .createCaption(); //Creates an empty <caption> element and adds it to the table.

          .createTFoot(); //Creates an empty <tfoot> element and adds it to the table.

          .createTHead(); //Creates an empty <thead> element and adds it to the table.

          .deleteCaption(); //Removes the first <caption> element from the table.

          .deleteRow(0);  //Removes a row (<tr>) from the table.

          .deleteTFoot(); //Removes the <tfoot> element from the table.

          .deleteTHead(); //Removes the <thead> element from the table.

          .insertRow(0);  //Creates an empty <tr> element and adds it to the table.

          tableObject.cells; //Returns a collection of all <td> or <th> elements in a table row. The elements in the collection are sorted as they appear in the source code.

          tablerowObject.insertCell(0); //Inserts a cell into the current row at the specified position.

          tablerowObject.deleteCell(0);//Deletes a cell in the current table row in the specified index.

//--------Loops & Comparisons

          while(){};

          do{}while();

          switch(variable a comparar){ case 5: console.log('Este merito fue'); break; default: console.log('Este es el default'); break;}

          for(var i in objeto o array)//El for in no sirve en arrays u objetos con propiedades numericas, solo para objetos con propiedades nombradas como strings, mejor usar un loop "for" normal

//--------Functions

          return variAble; //This is the sintax.

          (function(){alert('Hola');})(); //Self Invoking function

          alert(arguments.callee); //(Should be execcuet inside of a Function) Property "arguments" creates an array of the arguments of the function, and "callee" in this case prints the function itself.

//--------Messaging

          console.log('Prints anything on the console of the browser');

          confirm('Aceptar');// PopUp that requests user to accept by clicking on "Accept" button, will return true, or false in case the user clicks on "Cancel".

          prompt('Introducir valores');// PopUp that requests user to enter text in a field, entered text will eb stored in a variable.

//--------BOM

          document.URL; // Gets the url string of the page.

          document.referrer; // Returns the URL of the document that loaded the current document.

          document.lastModified; //Returns the date and time the current document was last modified. This property is read-only.

          window.open("www.google.com","\_blank","toolbar=yes, titlebar=no,location=yes, directories=no, status=no, menubar=yes, scrollbars=yes, resizable=no, copyhistory=yes, width=650, height=900, left=660, top=0");

          var timeID = setTimeout(functionNoParentesis, 3000,param1,param2,...); //Calls a function or evaluates an expression after a specified number of milliseconds.

          clearTimeout(timeID); //clears a timer set with the setTimeout() method. The ID value returned by setTimeout() is used as the parameter for the clearTimeout() method.

          var timeID = setInterval(functionNoParentesis, 3000,param1,param2,...); //Calls a function or evaluates an expression at specified intervals (in milliseconds).

          clearInterval(timeID); //Clears a timer set with the setInterval() method. The ID value returned by setInterval() is used as the parameter for the clearInterval() method.

          //--------Document

          document.embeds; //Returns a collection of all <embeds> elements in the document. The elements in the collection are sorted as they appear in the source code.

          document.forms; //Returns a collection of all <form> elements in the document. The elements in the collection are sorted as they appear in the source code.

          document.images; //Returns a collection of all <img> elements in the document. The elements in the collection are sorted as they appear in the source code.

          document.links; //The links collection returns a collection of all links in the document. The links in the collection represents <a> elements and/or <area> elements with a href attribute. If the element is missing the href attribute, nothing is returned. The elements in the collection are sorted as they appear in the source code.

          document.title; //Sets or returns the title of the current document (the text inside the HTML title element).

          //--------Location

          location.hash = "part5"; // Sets or returns the anchor part of a URL, including the hash sign (#). Note: When this property is used to set the anchor part, do not include the hash sign (#).

          location.host; or location.host="www.ejemplo.com"; //Sets or returns the hostname and port of a URL. If the port number is not specified in the URL (or if it is the scheme's default port - like 80, or 443), some browsers will not display the port number.

          location.hostname; or location.hostname="ejemplo.com"; //Sets or returns the hostname of a URL.

          location.href; location.href="http://www.example.com/default.htm";// Sets or returns the entire URL of the current page.

          location.pathname; or location.pathname=path; // Sets or returns the pathname of a URL.

          location.port; or location.port=2095;// Sets or returns the port number of a URL. If the port number is not specified in the URL (or if it is the scheme's default port - like 80, or 443), some browsers will display 0 or nothing.

          location.protocol; or location.protocol="https:"; //Sets or returns the protocol of the current URL, including the colon (:). The protocol is a standard that specifies how data are transmitted between computers.

          locatin.search; or location.search="?email=someone@example.com";  //Sets or returns the querystring part of a URL, including the question mark (?) The querystring part is the part of the URL after the question mark (?). This is often used for parameter passing.

          location.assign("http://www.w3schools.com");  //Loads a new document.

          location.reload(true);  //Does the same as the reload button in your browser. By default, the reload() method reloads the page from the cache, but you can force it to reload the page from the server by setting the forceGet parameter to true: location.reload(true).

          location.replace("http://www.w3schools.com"); //Replaces the current document with a new one. The difference between this method and assign(), is that replace() removes the URL of the current document from the document history, meaning that it is not possible to use the "back" button to navigate back to the original document.

          //--------Navigator

          navigator.appCodeName; //Returns the code name of the browser. Note: All modern browsers returns "Mozilla", for compatibility reasons.

          navigator.appName; //Returns the name of the browser.

          navigator.appVersion; //Returns the version information of the browser.

          navigator.cookieEnabled;  //Returns a Boolean value that specifies whether cookies are enabled in the browser.

          document.cookie; = "username=John Doe"; // Creating a cookie.

          navigator.geolocation;  //Returns a Geolocation object that can be used to locate the user's position. Since this can compromise user privacy, the position is not available unless the user approves it.

          navigator.language; //Returns the language of the browser.

          navigator.onLine; //Determines whether the browser is online.

          navigator.platform; //Returns for which platform the browser is compiled.

          navigator.product;  //This property may not work as expected: All browsers returns "Gecko".

          navigator.userAgent;  //Returns the user-agent header sent by the browser to the server.

          navigator.javaEnabled();  //Returns a Boolean value that specifies whether the browser has Java enabled.

          taintEnabled()  //Removed in JavaScript version 1.2. Specifies whether the browser has data tainting enabled.

          //--------Screen

          screen.availHeight; //Returns the height of the screen (excluding the Windows Taskbar).

          screen.availWidth;  //Returns the width of the screen (excluding the Windows Taskbar).

          screen.colorDepth;  //Returns the bit depth of the color palette for displaying images.

          screen.height;  //Returns the total height of the screen.

          screen.pixelDepth;  //Returns the color resolution (in bits per pixel) of the screen.

          screen.width; //Returns the total width of the screen.

      //--------Events

      element.addEventListener("click", displayDate, false);/\*Attaches an event handler to an element without overwriting existing event handlers.

                              You can add many event handlers to one element.

                              You can add many event handlers of the same type to one element, i.e two "click" events.

                              You can add event listeners to any DOM object not only HTML elements. i.e the window object.

                              The addEventListener() method makes it easier to control how the event reacts to bubbling.\*/

      <a href="http://www.google.com" onclick="event.preventDefault()"// Cancels the event if it is cancelable, meaning that the default action that belongs to the event will not occur.

//--------Forms

          sistema = document.form1.sistema;   //Primero Seleccionamos la etiqueta Select dentro de la forma

          siSel = sistema.selectedIndex;      //Luego obtenemos cual es el Option seleccionado

          vaSis = sistema[siSel].value;       // Ahora obtenemos el valor del Option

          document.getElementsByTagName("input")[0].focus();//Give focus to an <input> element

          .setSelectionRange(0,500); // Selects from the focused input a portion of the text inside.

//--------OBJECTS

          var myObj = {}; // Notacion Literal de Objeto

          var myObj = new Object(); // Constructor de objeto

          myObj['name'] = 'Fabian';

          myObj.age = 28;

          var myArray = [21, true, 'Fabian', [myObj.name] ];//Un objeto dentro de un array

          var myObject = { //Un objeto con un array dentro

            name: 'Eduardo',

            type: 'Most excellent',

            interests: [fabian[1]]

          };

          //Objetos en Objetos

          var friends = {

            bill:{

            firstName: "Bill",

            lastName: "Gates",

            number: "(206) 555-5555",

            address: ['One Microsoft Way','Redmond','WA','98052']}

          };

          //------------------------------------------------------

          function Persona (name, lastname, age, address){

            this.nombre = name;

            this.apellido = lastname;

            this.edad = age;

            this.direccion = address;

          };

          var fabian = new Persona ("Fabian", "Alvarado", "29", "Genova 15");

          var diReccion = 'direccion';

          console.log(fabian[diReccion]);

          console.log( anyObj.hasOwnProperty('name') ); // True o False dependiendi si tiene una propiedad en especifico

          //-------------------

          anyObj.prototype.calcular = function() { //Esto es para a la clase anyObj agregarle el metodo calcular

            statements;

          };

          //---------------

          anyObj.prototype = new Animal(); // Esto es para que la clase anyObj herede todo de Animal.

          myFunction.call(myObject, "My name", 33); //Used to invoke a function, must have the owner object as first parameter, takes the function arguments separately by comas.

          myFunction.apply(myObject, ["My name", 33]); //Used to invoke a function, must have the owner object as first parameter, takes the function arguments in an array.

//--------ARRAYS

          var aRray = [5, 'hola', '56'];

          array.length;

          var yo = [[5,6,7,8] , ['Tu','Yo', 'Nosotros'] , [3]];

          console.log(yo[1][2]);

          array1.concat(array2, array3, array4);//This method is used to join two or more arrays.

          array1.join();//This method joins the elements of an array into a string, and returns the string.The elements will be separated by a specified separator. The default separator is comma (,).

          array1.pop();//Removes the last element of an array, and returns that element. This method changes the length of the original array.

          array1.shift();//Removes the first element of an array, and returns that element. This method changes the length of the original array.

          array1.push("something1", "something2");//Adds new items to the end of an array, and returns the new length. The new item(s) will be added at the end of the array.

          array1.unshift("something1", "something2");//Adds new items to the beginning of an array, and returns the new length. The new item(s) will be added at the end of the array.

          array1.reverse(); //Reverses the order of the elements in an array.

          array1.filter(function(currentValue, index, arr), thisValue); //Creates an array filled with all array elements that pass a test (provided as a function).It does not execute the function for array elements without values. It does not change the original array.

|  |  |  |
| --- | --- | --- |
| **Topic** | **Question** | **Answer** |
| Essential | Javascript | Procedural |
| Essential | Naming a variable | .-Cannot start with a number -No spaces -No reserved words -No characters except $ \_ |
| Essential | Print a console a warning | console.warn(); |
| Essential | Print a console an error | console.error(); |
| Essential | Primitive data types | A data type that represents a single value, meaning it is not an object. |
| Essential | Undefined - Primitive | Undefined represents lack of existence. Never set a variable to this type, so when you see a variable as "undefined" you will know that something went wrong and it has the default value assigned by the Engine. |
| Essential | Null - Primitive | Represents lack of existence. It can be manually set to null so you know when you defined a varible to be null, unlike the varibles set to undefined by the Engine. |
| Essential | Boolean - Primitive |  |
| Essential | Number - Primitive | The only numeric type in JS, all numbers are floating. |
| Essential | String - Primitive |  |
| Essential | Symbol - Primitive |  |
| Essential | Operator: | A special function syntactically differently that generally takes two parameters and returns one result. It uses the "infix notation" which means that the operator is placed between the 2 values to process. |
| Math | Random method | Math.random(); |
| Math | Round method with n as argument | Math.round(n); |
|  | Switch | var sign = prompt("elige uno, dos, tres o cuatro").toLowerCase(); switch (sign){  case "uno":  alert("Elegiste 1");  break;  case "dos":  alert("Elegiste dos");  break;  case "tres":  alert("Elegiste tres");  break;  default:  alert("Elegoste algo fuera del rango");  break; } alert("La seleccion fue concluida"); |
|  | Cast into number | Number(myVar); |
|  | Object Literal | "use strict"; var myO = {  nombre: "Fabian",  apellido: "Alvarado" }; myO.edad = 33; // Added edad alert(myO.edad); |
|  | Object Constructor | Creating the object as a template or prototype, then creating as many instances as needeed. |
|  | ES6 Class Constructor | class Cat {  constructor(name, color) {  this.color = color;  this.name = name;  }  speak() {  return "meaw";  } } var Simon = new Cat("Simon", "Blanco"); console.log(Simon.name); console.log(Simon.speak()); |
|  | Types of objects in javascript | Host objects, Core objects and objects defined by the coder. |
|  | Host objects | Document, Window, Element, Event, Node, Comment, Console. |
|  | Core Objects | Math, Object, String, Boolean, Array, Date, Number. |
|  | What is the result of "alert(this);" in the global scope. | [Object window] |
|  | All the coding resulting of the developer is by default child of the Window Object |  |
|  | What would happen if we redeclare one of the methods that are by default methods of the window? Like redefining the alert() so it logs a console message instead of alerting? | They forget what they were and now execute our code. |
|  | "use strict"; |  |
|  | Why would be useful to create a prioperty like this: cat["eye color"] = "Green"; | Because we can use "invalid identifiers" not allowed in a different way. In addition we could use variables. |
|  | Every property has a property descriptor to show the attributes of the property. | console.log(Object.getOwnPropertyDescriptor(Simon, 'name')); |
|  | How to set a property as read-only | Object.defineProperty(Simon, 'name',{writable: false}); |
|  | What is the code you encapsulate a function in to make it a IIFE? | ( function(){}(); // Anonymus function calling itself wrapped up in parenthesys. ); |
|  | Lexical Scoping | Inner functions are not available outside but external variables are accesible inside the other functions. |
|  | What is closures? | Functions with preserved data. X x x |
|  | What is variable shadowing? | To name the same 2 different variables in 2 different scopes so making them to keep their value and not to overwrite themselves. |
|  | In a for loop, after it executes, does it still sum a last time to the counter? | In a for loop when it ends, after latest validation, the counter still sums ++ one last time. |
|  | Syntax parser | The program that converts the code we write into a set of computer instructions that the computer understands and verifies our grammar. |
|  | Lexical Environment | Where it is written and what sorrunds it. |
|  | Execution context | A wrapper to help manage the code that is running. |
|  | Object | A collection of name-value pairs. |
|  | Whats does the Javascript Engine creates first when loads the page? | The global object and and the variable "this" |
|  | Javascript phases (Where "hoisting" happens). | There are two phases in Javascript when the Engine executes the code: 1.- In the first phase, "Creation Phase" the Javascript Engine creates the global object and the variable "this". Then it takes all Declared Variables "var x..." (not the ones only initilized x = ...) and reserves a space in memory for them assigning the value of "undefined". Next, it reserves space in memory for all Declared Functions , not just the name but the entire declared functions (notice that the Engine is not reserving space for Function Expressions, they are taken like declared variables because of the word "var" hence they were stored in memory and assigned "undefined").  2. The second phase is the code execution, in the order it was written, where the value asignment happens and the varibles that were assigned "undefined" value receive their real values.  m(); function m (){  console.log(x); }; var x = 50; // Experiment removing var to see that it changes" |
|  | Single threaded: | One command at a time |
|  | Synchronous execution: | One line of code at a time and in the order they were created. |
|  | Invocation: | Running a function (by using parenthesis) |
|  | When a function is invoked… | A new Execution Context is created and put at the top of the Execution Stack. In every Execution Context the 2 phases take place, it first creates the space in memory for the variables and declared functions, and then excecutes the code line by line. Once the code within the current execution context finishes, the execution context pops off and continues executing the previous execution context. |
|  | Variable environment | Where the variable lives, its scope. |
|  | Scope Chain | The outer environment of a function is where the function is created, not where the function is invoked from. So the scope chain means that if we want to use a variable and it is not available in the scope of the function, it will go one level up and try to find the variable but following the scope chain just up. |
|  | Block variables / Block scoping | Let variables, that will be allowed inside only a block, defined by the curly braces signs. The space in memory will be still reserved for it but they cannot be accessed outside the block. In addition the variables cannot be used in the code in lines before the lines they are created. |
|  | Asynchronous: | More than one at a time. |
|  | Event Queue | To start accepting events, the Javascript Engine first finishes with the Execution stack created by code, only then it starts handling the Events queue, this means that it starts creating the Execution Context for every function coming from an event. |
|  | Dynamic Typing: | A variable can change its type during the execution, from boolean to string, to number etc. |
|  | Operator Precedence: | Is the order that the JS Engine calls function operators based on the precedence value, for instance \* is higher than +. |
|  | Operator Associativity: | Is the order that the JS Engine calls same function operators in the same expression, it could be right to left or left to right, based on the established associativity, for instance a=b=c; |
|  | Coercion: | Converting a value from one type to another. |
|  | == | It before comparing tries to coerce the values. |
|  | === | It compares strictly the values with no coercion. Use these by default. |
|  | JS with 15 floating points is precize, but more than that it is not. | 1.000000000000001 - Still precise. |
|  | Default values: | window.myVar = window.myVar || "The content of my var in case there is not"; |
|  | Object literals can be created on the fly and be passed to a function like any other value. | executeThis({name:"Fabian", lastname: "Alvarado"}); |
|  | Eval("5 + 5"); Do not use it unnecessarily, better to find alternatives. | If the argument is an expression, eval() evaluates the expression. If the argument is one or more JavaScript statements, eval() executes the statements. |

|  |  |  |
| --- | --- | --- |
| **Topic** | **Question** | **Answer** |
| Forms | What Attribute tells the browser that form-data (input) should not be validated when submitted? | novalidate |
| Forms | What is the syntaxis of "novalidate" attribute? (Use this format: <element attribute="value">) | <form novalidate> |
| Forms | What is the syntaxis of "required" attribute? (Use this format: <element attribute="value">) | <input required> |
| Forms | What Attribute validates that the form-data (input) should not be les than x number of characteres? <element attribute="value">) | <input minlength="20"> |
| Forms | In the element label, what Attribute assigns the label to an escpecific input? Please write the code to assign a label to an input element with id "name" <element attribute="value">) | <label for="name"> |
| Forms | What HTML Tag do you use to group related elements inside a form? | fieldset |
|  | What Attribute disables the input tag so it cannot be clicked or written on it? <element attribute="value">) | <input disabled> |
|  | A button element can be of 3 types, what are they? | Button, submit, reset |

1. **Performance**
   1. It is better to have CSS styles in the head so the browsers knows how to render the html before loading it.
   2. It is better to have all script tags at the bottom of the html so the scripts run until the html is processed, in addition the render of the page will not be paused due to scripts running.
   3. It is a good idea to point frameworks to CDN locations so if the user had visited some page using certain library, it will not download it again when visiting your page because it will be in the cache already, in adidtion it will be routed to the nearest server.
   4. Minification, tools to minify our code, like Grunt.
   5. Concatenation, tools to merge several styles or scripts in a sigle file, like Grunt.

 display: grid;

      place-items: center;

 user-select-none;

box-sizing: border-box;

cursor: pointer;

unicode-bidi: bidi-override;

direction: rtl;

csv variable

 html{

            --colorcito: #f00;

        }

document.importNode(template.content, true);

var code = document.getElementById('code').contentWindow.document;

this.ctrlEl.scrollTop = this.ctrlEl.scrollHeight;

Schema.find().cursor();

Try/catch/finally

@font-face {

  font-family: Jura;

  src: url("./assets/fonts/Jura-Bold.ttf");

  font-weight: bold;

}

text-rendering: optimizeLegibility;

-webkit-user-drag: none;

 -webkit-app-region: drag;

 user-select: none;

text-overflow: ellipsis;

tr:nth-child(1) th

 pointer-events: all;

 tr:nth-child(even) {

border-collapse:

 cursor: pointer;

::-webkit-calendar-picker-indicator

textarea::placeholder

::before {

  content: "\26A0";

display:flex;

  flex-direction: row;

  justify-content: space-between;

  align-items: stretch;

  stroke: white;

  stroke-opacity: 0.05;

  shape-rendering: crispEdges;

::selection {

-webkit-fill-available

\*::-webkit-scrollbar {

/\* Track \*/

\*::-webkit-scrollbar-track {

/\* Handle \*/

\*::-webkit-scrollbar-thumb {

/\* Handle on hover \*/

\*::-webkit-scrollbar-thumb:hover {

\*::-webkit-scrollbar-button:vertical:decrement {

\*::-webkit-scrollbar-button:vertical:increment {

\*::-webkit-scrollbar-button:horizontal:decrement {

\*::-webkit-scrollbar-button:horizontal:increment {

 flex;

  flex-direction: row;

  justify-content: space-between;

  align-items: stretch;

@keyframes spin {

  to {

    transform: rotate(360deg);

  }

}

animation: spin 0.6s linear infinite;

letter-spacing: -2px;

performance.now

file\_system.statSync

 touch-action: none;

white-space: nowrap; /\* To make the text to hide the excess of charaters\*/

  text-overflow: ellipsis; /\* To make the text to hide the excess of charaters\*/

border-spacing: 0px;

 will-change: opacity;

 -webkit-app-region: no-drag;

  /\*-webkit-app-region: none; /\* We explicitly make the button class not dragging are so the regular events work\*/

For of

For in

selectionStart

selectionEnd