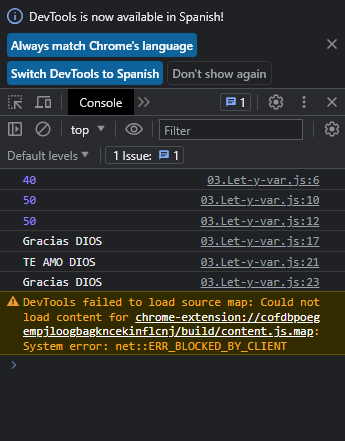
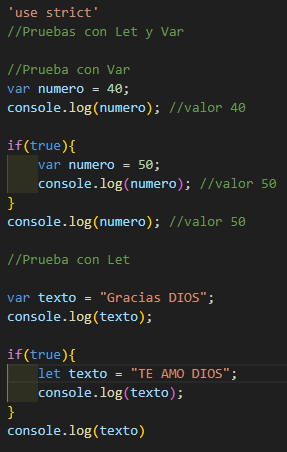
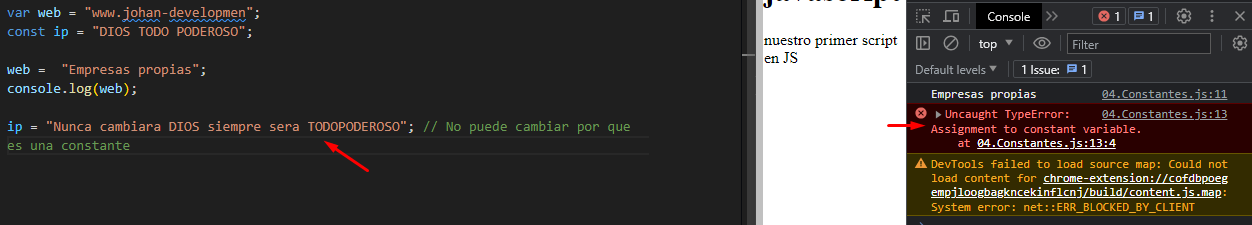
**JAVASCRIPT**

JavaScript (JS) es un lenguaje de programación ligero, interpretado, o compilado justo-a-tiempo (just-in-time) con funciones de primera clase. Si bien es más conocido como un lenguaje de scripting (secuencias de comandos) para páginas web, y es usado en muchos entornos fuera del navegador, tal como Node.js, Apache CouchDB y Adobe Acrobat JavaScript es un lenguaje de programación basada en prototipos, multiparadigma, de un solo hilo, dinámico, con soporte para programación orientada a objetos, imperativa y declarativa (por ejemplo programación funcional).

1. **’use** **strict’** se debe implementar en el código en la primera línea para que se active el modo estricto, este es una buena practica ya que así no se podrá declarar una variable o modificar si no se usa la palabra Var
2. **VAR** se para declarar una variable de forma global y **Let** es para declararlo de manera local en el bloque de código que se ejecuta. En el ejemplo se ve que el valor de texto al salir del condicional if vuelve a ser el anterior mientras en var se mantiene el valor actual.



1. **Constante es una variable que su valor no se puede modificar siempre es la misma.**

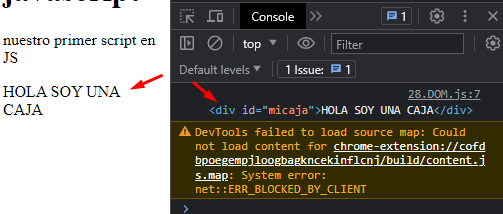
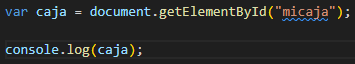


1. **Typeof se usa para mostrar que tipo de dato es la variable**
2. **22.Arrays-multidimensionales.js**
3. //ARRAYS MULTIDIMENSIONALES
4. var categoria = ['Cristiana','Acción', 'Terror', 'Comedia'];
5. var peliculas = ['San Andres','Jackson','¿Qué paso ayer?','TE AMO DIOS'];
6. var cine = [categoria,peliculas];
7. console.log(cine);
8. //el primer [] es la columna y el segundo [] el elemento
9. console.log(cine[0][0]);
10. console.log(cine[1][3]);
11. //Añadir un elemento al array en JS se usa PUSH
12. peliculas.push("Batman");
13. console.log(peliculas);
14. /\*Prompt para introducir elementos a un array
15. var elemento = prompt("");
16. do{
17. elemento = prompt("introduce tu película");
18. peliculas.push(elemento);
19. }while(elemento != "acabar");
20. //.POP elimina el ultimo elemento del Array me servirá para eliminar la palabra acabar al final de la ejecución
21. peliculas.pop();
22. console.log(peliculas);
23. \*/
24. //Eliminar un elemento del Array
25. //Primero debo buscar el indice
26. var indice = peliculas.indexOf('Jackson');
27. if(indice > -1){//sera -1 en caso de que no lo encuentre
28. peliculas.splice(indice, 1);//splice para eliminar
29. }
30. console.log(indice);
31. console.log("Debe faltar Jackson = " + peliculas)
32. // .JOIN convierte el arrays a un string de texto separado por una coma
33. var peliculasString = peliculas.join();
34. console.log("Peliculas string: " + peliculasString);
35. // .SPLIT convertir un string a un array
36. var cadena = "texto1, texto2, texto3";
37. var cadena\_array = cadena.split(", ");
38. console.log(cadena\_array);
40. // .SORT me ordena los elementos del Array en orden alfabético
41. var abcdario = ['e','d','a','f','b','c'];
42. console.log(abcdario.sort());
43. //.reverse invierte el orden del array
44. var abcdario2 = ['reverse','e','d','a','f','b','c'];
45. console.log(abcdario2.reverse());
46. //Recorrer un Arrary con FOR IN
47. for(let indice in abcdario2){
48. document.write("<ul>"+abcdario2[indice]+"</ul>");
49. }
50. //Busquedas
51. var busqueda = abcdario2.find(function(indice){
52. return indice == "e";
53. });
54. console.log("la letra que buscabas si esta y es: " + busqueda);
55. //forma reducida
56. var busqueda = abcdario2.find(indice => indice =="a");
57. console.log("REDUCIDA la letra que buscabas si esta y es: " + busqueda);
58. //Buscar el índice se usa .finIndex
59. var busqueda = abcdario2.findIndex(indice => indice =="a");
60. console.log("El indice de la letra a es: " + busqueda);
61. var precios = [10, 20 , 30, 40, 50, 60, 70 , 80, 90, 100];
62. var buscarvalor = precios.some(precio => precio >= 50);
63. console.log(buscarvalor);

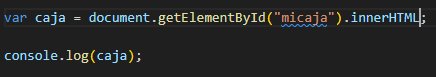
**DOM - Document object model**

El DOM (Document Object Model, en español Modelo de Objetos del Documento) es una API definida para representar e interactuar con cualquier documento HTML o XML. El DOM es un modelo de documento que se carga en el navegador web y que representa el documento como un árbol de nodos, en donde cada nodo representa una parte del documento (puede tratarse de un elemento, una cadena de texto o un comentario).

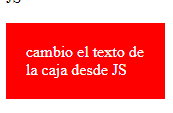
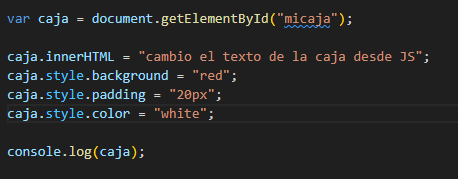
1. Document.getElementById( ) se usa para capturar la información que está dentro de un id y manipularlo.

I

1. .innerHTML saca el contenido del id

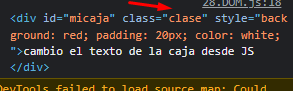


También se puede cambiar el contenido en este ejemplo le cambio el contenido y el estilo.

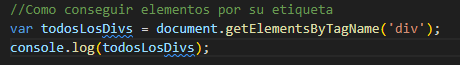


**var caja = document.querySelector("#micaja");** con este código **querySelector(“#id”)** puedo tener el mismo resultado.

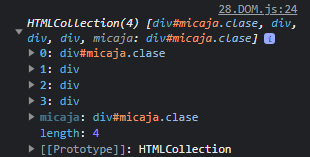
1. caja.className = "clase"; permite ponerle una clase al objeto. También se puede agregar mas clases poniendo mas entre las “ clase1 clase2”



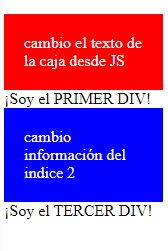
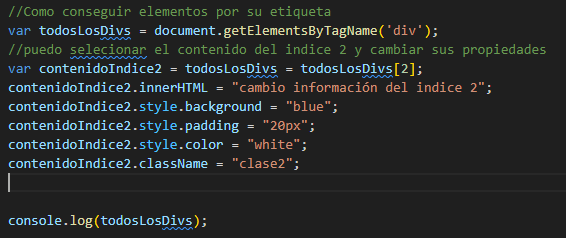
1. **document.getElementsByTagName('div');** Como conseguir elementos por su etiqueta



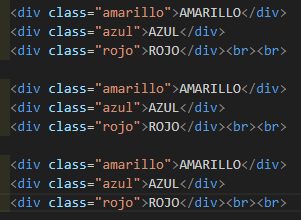
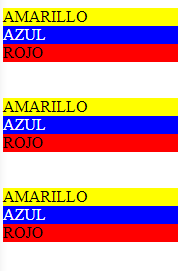
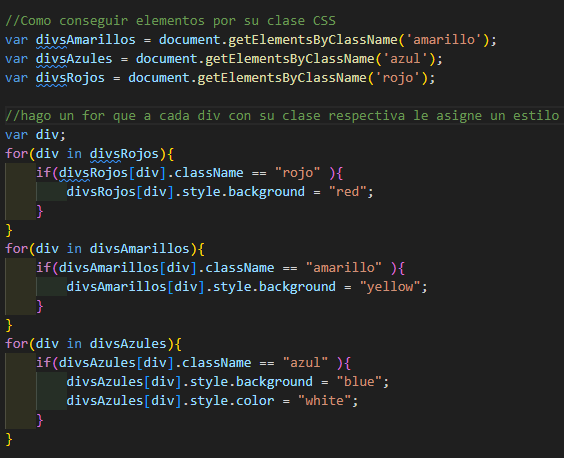
Se ve donde selecciono los 4 div que hay en el html y los mete en un array



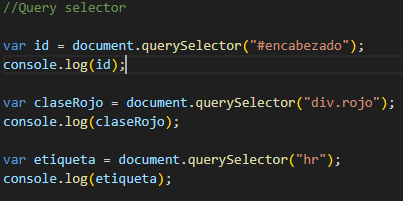
Para acceder a uno en especifico y modificarlos uso el índice del elemento y le aplico los cambios



1. Conseguir elementos por clase

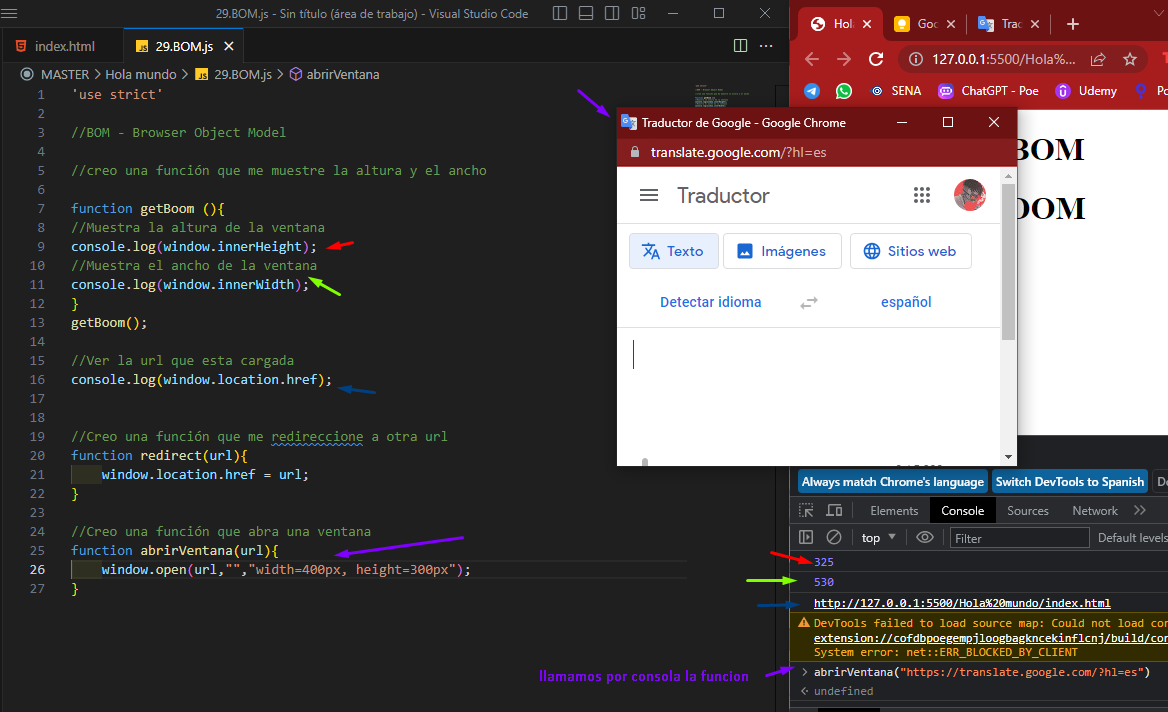


1. querySelector(“#idclaseoetiqueta”) puedo obtener los en un array lo que selecione



**BOM**

Mediante BOM, es posible redimensionar y mover la ventana del navegador, modificar el texto que se muestra en la barra de estado y realizar muchas otras manipulaciones no relacionadas con el contenido de la página HTML.



Eventos

1. //Eventos para el raton
2. var boton = document.querySelector('#boton');//Seleccionamos el botón pór su ID
3. //Función que cambia el color
4. function cambiarColor(){
5. console.log("me has dado click");
6. var bg = boton.style.background;
7. if(bg == "green"){
8. boton.style.background = "red";
9. }else{
10. boton.style.background = "green"
11. }
12. boton.style.padding = "10px";
13. boton.style.boder= "1pc solid #ccc"
14. return true;
15. }
16. var boton = document.querySelector('#boton');
17. //CLICK
18. /\*Con addEventListener capturo el evento dar click en el boton y
19. ejecuto la función cambiar color  \*/
20. boton.addEventListener('click',function(){
21. cambiarColor()
22. });
23. //Mouse over cuando paso el mouse por encima cambia a gris
24. boton.addEventListener('mouseover', function(){
25. boton.style.background = "#ccc";
26. });
27. //Mouseout ejecuta la función después de quitar el mouse de encima
28. boton.addEventListener('mouseout', function(){
29. boton.style.background = "purple";
30. });

mouseovercambiaconclickmouseoutalquitarmouse

1. **Formularios**

//Focus cuando estoy dentro de la casilla de texto

var input = document.querySelector("#campo\_nombre");

input.addEventListener('focus', function(){

    console.log("Estas dentro del input focus");

});

//Blur cuando le doy fuera de la casilla de texto

input.addEventListener('blur', function(){

    console.log("Estas fuera del input blur");

});

//Keydown la tecla que presiono en ese momento

input.addEventListener('keydown', function(event){

    console.log("Estas pulsando una tecla keydown: ", String.fromCharCode(event.keyCode));

});

//Keypress captura el evento cuando presiono la tecla

input.addEventListener('keypress', function(event){

    console.log("Tecla presionada keypress:", String.fromCharCode(event.keyCode));

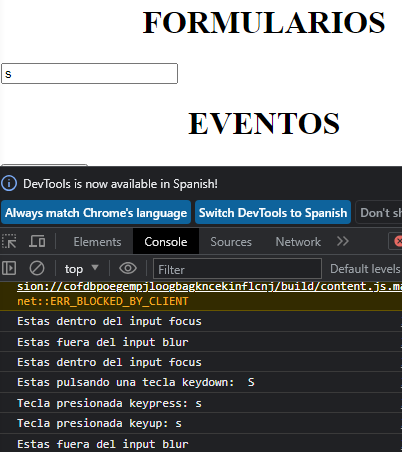
});

//Keyup captura el evento cuando levanto el dedo de la tecla

input.addEventListener('keypress', function(event){

    console.log("Tecla presionada keyup:", String.fromCharCode(event.keyCode));

});



1. **Event load**  cuando cargamos nuestro JS y no se aplica en el documento porque todavía no ha cargado el body o html, se usa este evento para que así cuando se cargue todo se ejecute el código correctamente

//Event load

window.addEventListener('load', () => {

//aquí adentro va el código

});

1. **JSON** (acrónimo de JavaScript Object Notation, 'notación de objeto de JavaScript') es un formato de texto sencillo para el intercambio de datos.

//Creamos un objeto

var metas = {

    titulo: 'Dios gracias',

    year: '2023',

    pais: 'Colombia'

};

console.log(metas);

//Cambiar el valor de un atributo

metas.year = "2024"

metas.pais = "Argentina"

console.log(metas);

Consola 

1. **Local Storage:** El objeto Storage (API de almacenamiento web) nos permite almacenar datos de manera local en el navegador y sin necesidad de realizar alguna conexión a una base de datos. En este artículo te mostraré cómo utilizarlo mediante JavaScript.

//Con esto comprobamos que el LocalStorage sea compatible

if(typeof(Storage)!== 'undefined'){

    console.log("LocalStorage disponible");

}else{

    console.log("LocalStorage no esta disponible");

}

//Guardar datos en el LocalStorage

localStorage.setItem("titulo","DIOS es mi pastor nada me faltara");

//Recuperar elemento y guardarlo

console.log(localStorage.getItem("titulo"));

//Asi lo meto en el documento

document.querySelector("#metas").innerHTML= localStorage.getItem("titulo");

//Guardar Objetos

var usuario = {

    nombre: "Johan",

    email: "johandres.a2v@mail.com",

    web: "johandres-development.com.ar"

};

//debemos convertir los datos de JSON puro a un JSON String

//usamos el método JSON.stringify que realiza la conversión

this.localStorage.setItem("usuario", JSON.stringify(usuario));

//Recuperar un objeto JSON string y volverlo un objeto JSON

//usamos JSON.parse que toma el JSON string y lo convierte en objeto JSON puro

var usuarioJS = JSON.parse(this.localStorage.getItem("usuario"));

console.log(usuarioJS);

//puedo mostrar por pantalla información del objeto que se ha recuperado

document.querySelector("#metas").append(usuarioJS.web);

//borrar los elementos del local storage

//Se usa .removeItem para 1 solo elemento

localStorage.removeItem("usuario");

//Se usa para borrar todos los elementos - esta comentado para ver lo anterior

                //localStorage.clear("usuario");

var formulario = document.querySelector("#formpeliculas");

formulario.addEventListener('submit', function(){

    //Tomo el valor obtenido en el input y lo guardo en titulo

    var titulo = document.querySelector('#addpelicula').value;

    //Condicional para que no se guarde vacío

    if(titulo.length >= 1){

        //en el siguiente código le digo que tome el valor de titulo y lo guarde como key y como valor

        localStorage.setItem(titulo, titulo);

    }

});

//Recorrer y mostrar todos los elementos del local storage

var ul = document.querySelector('#peliculas-list');

for (var i in localStorage){

    //Condicional si el dato que esta es un string se mostrara en el document

    if(typeof localStorage[i] == 'string'){

        var li = document.createElement("li");

        li.append(localStorage[i]);

        //Se añade a ul los li que son string

        ul.append(li);

    }

}

//Borrar película

var formulario = document.querySelector("#formBorrarPeliculas");

formulario.addEventListener('submit', function(){

    //Tomo el valor obtenido en el input y lo guardo en titulo

    var titulo = document.querySelector('#borrarpelicula').value;

    //Condicional para que no se guarde vacío y solo elimine si tiene algo escrito

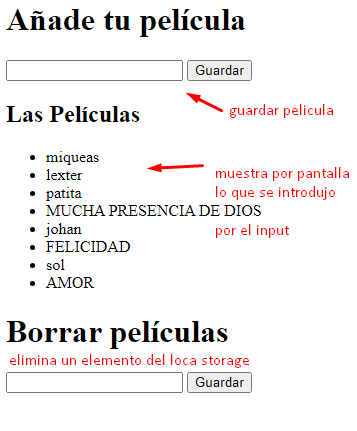
    if(titulo.length >= 1){

        //Elimina el elemento que contiene titulo

        localStorage.removeItem(titulo);

    }

});



1. **Fetch – peticiones asincrónicas**

La API Fetch proporciona una interfaz JavaScript para acceder y manipular partes del canal HTTP, tales como peticiones y respuestas. También provee un método global fetch() (en-US) que proporciona una forma fácil y lógica de obtener recursos de forma asíncrona por la red.

Estructura básica:

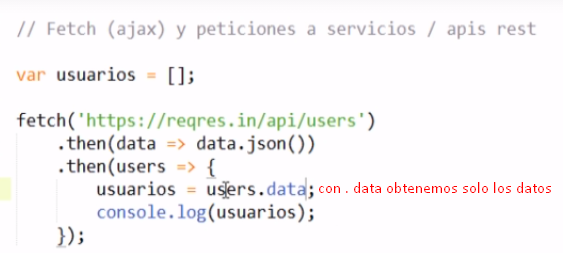
**fetch('http://example.com/movies.json')** //hago la promesa la api rest

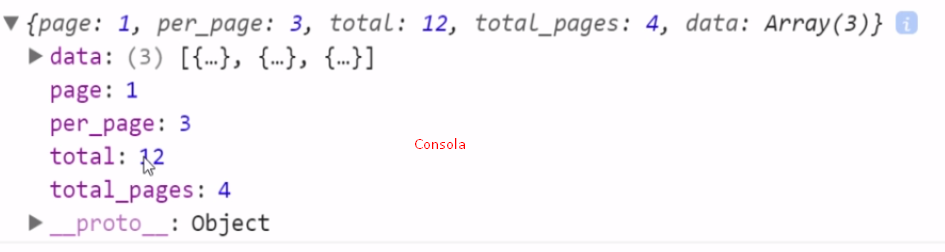
**.then(response => response.json())** // la convierto en format .json usable

**.then(data => console.log(data));** // recoge la data y la imprime por consola

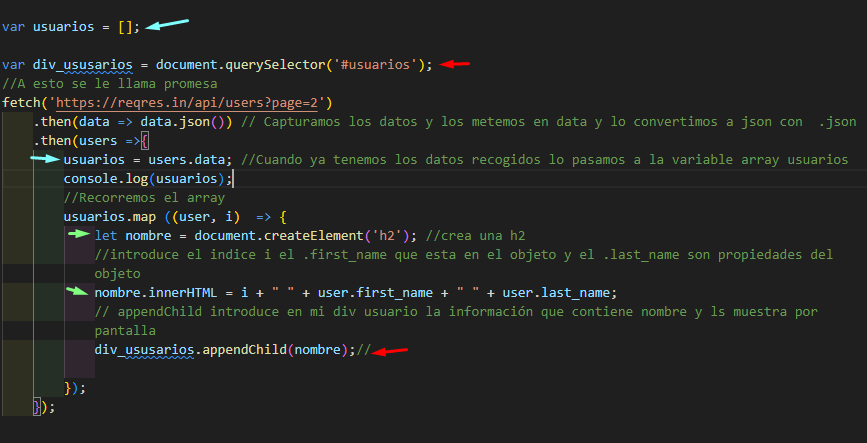
Después de transformar la información a json podemos asignarla a una variable que sea un array y darle mas usos a los datos.

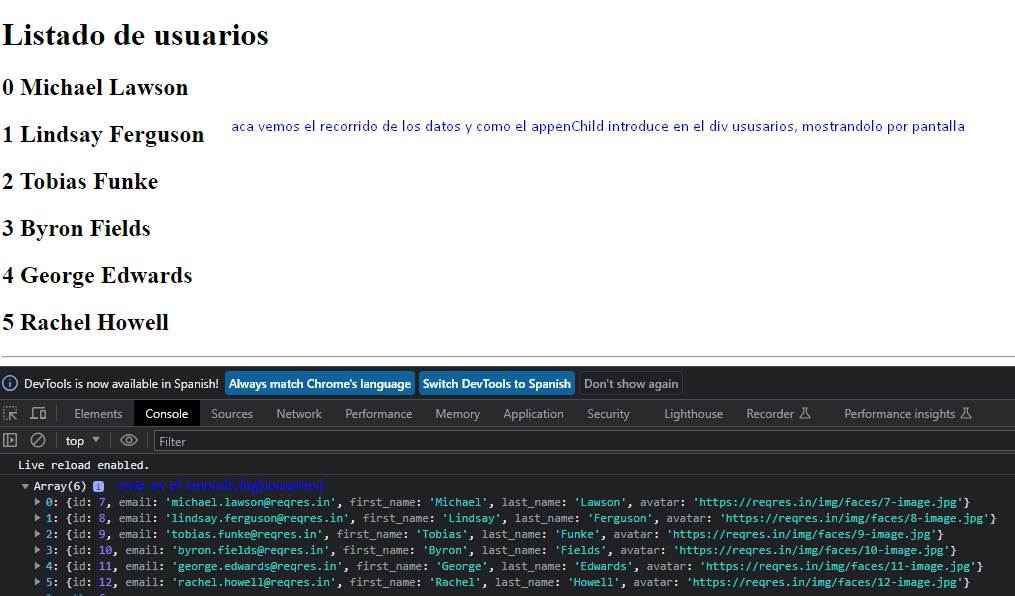
Ejemplo:





1. **.map**  lo podemos usar para recorrer los datos del array json. El método map() crea un nuevo array con los resultados de la llamada a la función indicada aplicados a cada uno de sus elementos.





El método **appendChild()** Agrega un nuevo nodo al final de la lista de un elemento hijo de un elemento padre especificado.

Siguiendo el ejemplo anterior haremos varias promesas fetch entre esas será mostrar los usuarios y un 1 usuario que en este ejemplo el objeto lleva por nombre janet.

var usuarios = [];

var div\_usuarios = document.querySelector('#usuarios');

var div\_janet = document.querySelector('#janet');

//A esto se le llama promesa

    getUsuarios()

    .then(data => data.json())

    .then(users =>{

        listadoUsuarios(users.data);

        return getJanet();// le digo que después de capturar el primer usuario agrega el usuario janet

    })

    .then(data => data.json())//Encadenamos la promesa getJanet

    .then(user =>{

        mostrarJanet(user.data);

    });

//Funciones que devuelven una promesa, usuarios y usuario Janet

    function getUsuarios (){

        return fetch('https://reqres.in/api/users?page=2');

    }

    function getJanet (){

        return fetch('https://reqres.in/api/users/2');

    }

//Recorremos el array con esta función esta función muestra los usuarios

    function listadoUsuarios(usuarios){

        usuarios.map ((users, i)  => {

            let nombre = document.createElement('h2');

            nombre.innerHTML = i + " " + users.first\_name + " " + users.last\_name;

            div\_usuarios.appendChild(nombre);

        });

    }

    function mostrarJanet(user){

            console.log(user);

            let nombre = document.createElement('h4');

            let avatar = document.createElement('img');// creo la variable avatar y creo mi etiqueta img

            nombre.innerHTML = user.first\_name + " " + user.last\_name ;

            avatar.src = user.avatar; //uso la propiedad src para introducir la imagen

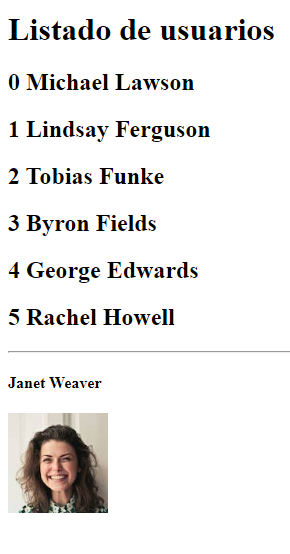
            avatar.width = "100";// le doy un ancho de 100 NOTA no se puede poner px o % no funciona

            div\_janet.appendChild(nombre);

            div\_janet.appendChild(avatar);

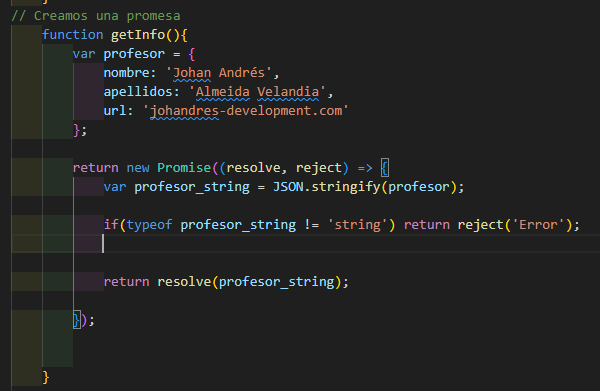
        ;

    }



1. **CREAR UNA PROMESA:**

Una Promise (promesa en castellano) es un objeto que representa la terminación o el fracaso de una operación asíncrona. Dado que la mayoría de las personas consumen promises ya creadas. Estructura básica:



Se estancia como un objeto usando new Promise y está compuesto por 2 elementos la respuesta y el rechazo, eso quiere decir que si hay respuesta se hará lo siguiente y si no lo otro que sería la terminación de la llamada en caso de error.

1. **Catch** Se usa para capturar un error

//catch se utiliza para capturar un error

    .catch(error =>{

        //Aquí se introduce la acción que tomara dependiendo del error

        console.log(error);

    });

1. **Try-catch** La declaración try...catch señala un bloque de instrucciones a intentar (try), y especifica una respuesta si se produce una excepción (catch). Dentro de try va el código que queremos probar y el catch recoge la excepción o error que se puede estar produciendo y se puede ejecutar una línea de código según la necesidad.

try {

    var year = 2019;

    alert(yea);

} catch (error) {

    console.log(error);

    alert(error);

}

**Jquery**

Es una biblioteca multiplataforma de JavaScript, creada inicialmente por John Resig, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web.1​ Fue presentada el 14 de enero de 2006 en el BarCamp NYC. De acuerdo a un análisis de la Web (realizado en 2017) JQuery es la biblioteca de JavaScript más utilizada, por un amplio margen.

La función de **jQuery ready(),** su funcionalidad es la de ejecutar funciones una vez cargada en su totalidad una página web (DOM)

$(document).ready(function(){

});

1. **Selectores**

$(document).ready(function(){

    console.log("Estamos listo");

//Selector de ID: con # puedo seleccionar algo en en html por su id

    var rojo = $("#rojo").css("background","red").css("color","white");

    var amarillo = $("#amarillo").css("background","yellow").css("color","black");

    var verde = $("#verde").css("background","green").css("color","white");

//Selector de clases: se usa . y el nombre de la clase

    var mi\_clase = $('.zebra').css("padding","5px");

    console.log(mi\_clase);

    //Una clase es como un array de los elementos que están dentro de y puedo seleccionarlo [] o con .eq()

    console.log("[ ] " + mi\_clase[0]);

    console.log(".eq " + mi\_clase.eq(1));

    $('.sin\_borde').click(function(){

        console.log("Click dado!!");

        $(this).addClass("zebra");

    });

//Selectores por etiqueta: simplemente entre las comillas ponemos la etiqueta para seleccionarlo

    var parrafos = $('p').css("cursor","pointer");

/\*\* $(this) esto es el elemento  \*/

    parrafos.click(function(){

        var texto = $(this);

        if(!texto.hasClass('texto-grande')){

            $(this).addClass('texto-grande');

        }else{

        $(this).removeClass('texto-grande');

        }

    });

//Selectores de atributo se usa $('[atributo="valor"]') de esta manera lo seleccionamos y realizamos cambios en el

    $('[title="Google"]').css("background","blue").css("color","white");

    $('[title="SENA"]').css("background","red").css("color","white");

//Otros

//FIND y PARENT se ponen entre comillas y separados por coma , lo que nos sirve para agregar cambios a varios elementos a la vez

    $('p,a').addClass('margen-superior');

    //Find busca dentro del div caja los elementos con la clase resaltado

    var busqueda = $("#caja").find('.resaltado');

    console.log("\*\*\*\*FIND\*\*\*\*")

    console.log(busqueda);

    //parent() lo que hace es salir del contenedor actual a una posición anterior se hace 2 parent para que se posicione en el li que tiene la clase .resaltado

    var busqueda2 = $("#elemento2").parent().parent().find('.resaltado');

    console.log("\*\*\*\*PARENT\*\*\*\*")

    console.log(busqueda2);

});

1. **Eventos en jquery**

$(document).ready(function(){

// MouseOver y MouseOut

    var caja = $("#caja");

    var caja2 = $("#caja2");

    //mouseover cuando esta encima del elemento

    caja.mouseover(function(){

        $(this).css("background", "red");

    });

    //mouseout cuando sale del area del elemento

    caja.mouseout(function(){

        $(this).css("background", "green");

    });

    function color\_blue(){

        $(this).css("background", "blue");

    };

    function color\_gris(){

        $(this).css("background", "gray");

    };

//Hover este método realiza la misma función que mouseover y mouseout

    caja2.hover(color\_blue,color\_gris);

//Click, Doble click

    caja.click(function(){

        $(this).css("background","black").css("color","white");

    });

    caja.dblclick(function(){

        $(this).css("background","white").css("color","black").css("border","2px solid black");

    });

//Focus y blur

    var nombre = $("#nombre")

    var datos = $("#datos")

    nombre.focus(function(){

        $(this).css("border", "2px solid green");

    });

    nombre.blur(function(){

        $(this).css("border", "2px solid transparent");

        $("#datos").text($(this).val()).show();//.tex introduce el valor escrito en el input .show hace visible el elemento div oculto

    });

//Mousedown es cuando presiono y mouseup cuando suelto el click

    datos.mousedown(function(){

        $(this).css("border-color", "red");

    });

    datos.mouseup(function(){

        $(this).css("border-color", "blue");

    });

    //Mousemove - nota event esta en desuso

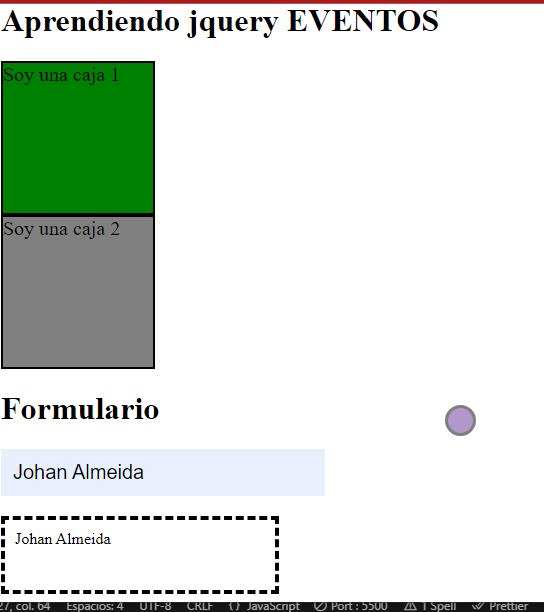
    //con este código hago que el div sígueme siga el puntero

    $(document).mousemove(function(){

        $("#sigueme").css("left", event.clientX).css("top", event.clientY);

    });

});



1. **Animaciones**

$(document).ready(function(){

    //Animaciones

    $("#mostrar").hide();

    $("#mostrar").click(function(){

        $(this).hide();

        $("#ocultar").show();

        $("#caja").fadeIn("slow");//FadeIn oculta y slow muestra la animación lenta

        //$("#caja").fadeTo("slow",1);//FadeTo realiza las misma función pero tiene 2 estados el 1 es activo

        //Si le pongo 0.8 en ves de 1 esto hará que se opaque y se puede graduar

    });

    $("#ocultar").click(function(){

        $(this).hide();

        $("#mostrar").show();

        $("#caja").fadeOut("slow");//FadeOut hace visible y slow muestra la animación lenta

        //$("#caja").fadeTo("slow",0);//FadeTo realiza las misma función pero tiene 2 estados el 0 es inactivo

    });

    //toggle realiza ambos efectos de ocultar y visualizar

    //slideToggle realiza un efecto de arriba hacia abajo

    $("#todoenuno").click(function(){

        $("#caja2").toggle("fast");

    });

    $("#animar").click(function(){

        $("#caja2").animate({

                                marginLeft: "500px",//mueve el contenedor al centro

                                fontSize: "40px",

                                height: "115"

                            }, "slow")

                            .animate({

                                borderRadius: "900px",

                                marginTop: "80px"

                            }, "slow").animate({

                                borderRadius: "100px",

                                marginTop: "0px"

                            }, "slow")

    });

});

**AJAX**

AJAX (de las siglas en Inglés Asynchronous JavaScript And XML ) es una práctica de programación utilizada para construir páginas web más complejas y dinámicas, utilizando una tecnología conocida como XMLHttpRequest (en-US).

Lo que AJAX nos permite hacer, es modificar partes específicas del DOM de una página HTML sin la necesidad de refrescar la página entera. AJAX tambien nos permite trabajar asincrónicamente; esto significa que tu código seguirá corriendo mientras esa parte de la página de tu sitio web intenta recargarse (en comparación, una carga síncrona bloquearía tu código hasta que esa parte de la página web terminara de recargarse).

1. **Ajax GET y POST**

$(document).ready(function(){

    //Load ayuda a incrustar información

    //$("#datos").load("https://reqres.in/");

    //GET

    $.get("https://reqres.in/api/users?page=2", {page: 3}, function(response){

        response.data.forEach((element, index) => {

            $("#datos").append("<p>"+element.first\_name+" "+element.last\_name+"</p>")

        });

    });

    //POST

    var usuario = {

        "name": "johan almeida",

        "empleo": "programador"

    }

    $.post("https://reqres.in/api/users?page=2", usuario, function(response){

        console.log(usuario);

    });

    //Usuario POST

    $("#formulario").submit(function(e){

        e.preventDefault();

    });

    var usuario22 = {

        "name: ": $('input[name="name"]').val(),//.val toma el dato escrito en html

        "empleo: ": $('input[name="empleo"]').val()

    }

    console.log(usuario22)

    //Por POST desde el HTML

    $.post($(this).attr("action"), usuario22, function(response2){

        console.log(response2);

    }).done(function(){

        alert("datos agregados correctamente")

    });

//Con el método $.ajax puedo realizar muchas mas funciones de manera mas simple y limpio

    $.ajax({

        type: 'POST',

        url: $(this).attr("action"),

        data: usuario,

        beforeSend: function(){

            console.log("Enviando usuario...");

        },

        success: function(response){

            console.log(response)

        },

        error: function(){

            console.log("A ocurrido un error");

        },

        timeout:10000

    })

    return false;

});

1. **JQUERY UI :** es una biblioteca de componentes construidos sobre la biblioteca jQuery que permiten desarrollar sobre un conjunto de plug-ins, widgets y efectos visuales preconstruidos para la creación de aplicaciones web. Cada componente o módulo se desarrolla de acuerdo a la filosofía de jQuery.
2. **D**
3. **D**
4. **D**
5. **D**
6. **D**
7. **D**
8. **D**
9. **DIOS GRACIAS**