**ECMAScript**

Estrategias

* Conservador (escribo todo en ECMAScript5 y se que va andar en todo)
* Delegador (delegas según la librería)
* Evergree (me centro solo en Chrome y Firefox que ahí andan todos)
* Transpilador (agarra todo el código y lo transforma en ECMAScript5 para que funcione en todos lados)

Polyfill (poner parche a librerías para que, si un navegador no tiene la actualización, muestre algo)

En la consola de JavaScript se puede debugear sin problema.

Lo mejor es poner el link de JS abajo. Justo antes de la etiqueta de cierre de body.

**VARIABLES**

**En ES5**

var alumno = “Juan”;

yerba = prompt(“Cual es la mejor yerba?”);

*método para capturar una variable escrita por el usuario en una alerta.*

Document.write(contenido)

*Para escribir en el DOM contenido JS.*

**TIPOS DE DATOS**

Number (solo tiene number, no tiene int ni nada de eso)  
String  
booleanos  
Arrays [elementos] *se empieza a contar desde 0*array.lenght; *cuantos elementos tiene el array*  
array.push(elemento) *para agregar elementos*  
array.pop() *para sacar elemento*

Array.concat(array) *junto dos array*

document.write("<br>" + array.join("-")); *muestro el texto con las separación que yo elijo, en este caso un -*  
array.sort(); *ordeno de menor a mayor*

**ELEMENTOS**

   var bicicleta = {  
            marca : "Vairo",  
            rodado : 29,  
            pesos : 8.150  
        }

bicicleta.marca

accedo a la propiedad marca

**undefined**  
var nombre;

**null**var nombre = null;

**Nan**var nombre = “String” \* 3;  
*Básicamente aparece cuando se combinan cosas que no se pueden o se hace cualquier cosa*

**ES6**

Se implementa el let y const;

let nombre = “pepe”;

let nombre = “juan”;

*Esto rompe porque no se pueden declarar dos variables con el mismo nombre;*

let nombre = “pepe”;

Nombre = “juan”

*Esto si funciona, porque no declaro una nueva variable, sino que le cambio el valor.*

const CIUDADANIA = “Belga”;

*Los const no se pueden modificar.*

typeof variable;

*devuelve el tipo dato de la variable (string, number, bla)*

**OBJETO MATH**

        let numeroVariable = Math.abs(-4);

        console.log(numeroVariable);

        const numeroPi = Math.PI;

        console.log(numeroPi);

Math.floor redondea para abajo.

## **Clase 14**

Ambos son para el paso de información y la comunicación, la diferencia es como:

Microservicios es por **JSON**

Web services es por **XML**

**Prompt ingresa STRINGS.**

Para obtener un number hay que parsear lo ingresado.

//Prompt por defecto ingresa strings

        let unaVaribale = prompt("Ingresa un numero");

        let otraVariable = prompt("Ingresa otro numero");

        let suma = unaVaribale + otraVariable;

        document.write("El resultado es: " + suma);

        //Parseo el string a number

        unaVaribale = Number(unaVaribale);

        otraVariable = Number(otraVariable);

        suma = unaVaribale + otraVariable;

        document.write("El resultado es: " + suma);

Otra opción es parsear al momento del ingreso.

let unaVaribale = Number(prompt("Ingresa un numero"));

      let otraVariable = Number(prompt("Ingresa otro numero"));

      let suma = unaVaribale + otraVariable;

Otra forma es con parseInt.

parseInt(unaVaribale);

parseInt(otraVariable);

ParseInt permite parsear en distintos modos, como hexadecimal, etc.

Operar con Strings

 let palabra = "palabra";

        //repetirla

        let repetir = palabra.repeat(7);

        document.write(repetir);

        //reemplazar un string o una parte de un string por algo

        console.log(palabra.replace("bra" , " "))

Ejercicios con matrices

//linea de abajo llena

        for ( i = 0 ; i < 5 ; i++){

            document.write("<br>");

            for ( j = 0 ; j < 5 ; j++) {

                if( i >= j ) {

                    document.write("1");

                } else {

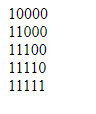
                    document.write("0");

                }

            }

        }

Resultado:



//diagonal principal

        for ( i = 0 ; i < 5 ; i++){

            document.write("<br>");

            for ( j = 0 ; j < 5 ; j++) {

                if( i == j ) {

                    document.write("1");

                } else {

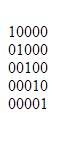
                    document.write("0");

                }

            }

        }

Resultado:



//preguntar por numeros pares o impares

        for ( i = 0; i < 10; i++){

            if(i%2 == 0){

                document.write(i + " es par")

            }

        }

# DOM

COMO INGRESAR AL DOM:

//GET ELEMENTS BY TAG NAME

Accedo, desde JS, al elemento mediando el tipo de elemento (p, h1, a, etc)

        let parrafos = document.getElementsByTagName("p");

 //GET ELEMENTS BY TAG NAME

        let parrafos = document.getElementsByTagName("p");

        console.log(parrafos);

        //ACCEDO AL SEGUNDO PARRAFO EN ESPECIFICO- PRIMERO TENGO QUE

TENER TODOS LOS PARRAFOS EN UN ARRAY CON LA FUNCION ESCRITA ARRIBA

        let segundoParrafo = parrafos[1].innerHTML;

        document.write(segundoParrafo);

OuterHTML devuelve en consola con las etiquetas. Inner no devuelve con etiquetas, solo devuelve contenido.

        parrafos[1].innerHTML = "Parrafo modificado";

modifico el segundo párrafo desde JS.

//GET ELEMENT BY ID

Para que esto funcione, la etiqueta debe tener un id

//GET ELEMENT BY ID

        let parrafoNuevo = document.getElementById("parrafo1");

        document.getElementById("parrafo1").innerHTML = " Este es mi nuevo parrafo modificado desde GET ELEMENT BY ID";

Así modifico los estilos:

        document.getElementById("parrafo1").style.color = "red";

//QUERY SELECTOR

//acceder con query selector

        const TITULO = document.querySelector("h2");

//query selector trae el primer elemento que encuentra con esa etiqueta

        console.log(TITULO);

//acceder con query selector por ID

        const TITULOPORID = document.querySelector("#title");

        console.log(TITULOPORID);

//QUERY SELECTOR POR CLASE

        //acceder con query selector por CLASS

<div id="container1">

        <div class="caja">Caja1</div>

    </div>

    <div id="container2">

        <div class="caja">Caja2</div>

    </div>

  const CAJAPRIMERA = document.querySelector(".caja");

        console.log(CAJAPRIMERA.innerHTML);

        //ACCEDER AL ELEMENTO DEL CONTAINER2 CON QUERY SELECTOR

        const CONTAINERSEGUNDO = document.querySelector("#container2");

        const CAJADELSEGUNDO = CONTAINERSEGUNDO.querySelector(".caja");

        console.log(CAJADELSEGUNDO.innerHTML);

//QUERY SELECTOR ALL

<div id="container1">

        <div class="caja">Caja1</div>

    </div>

    <div id="container1">

        <div class="caja">Caja2</div>

    </div>

    <div id="container1">

        <div class="caja">Caja3</div>

    </div>

const CAJAS = document.querySelectorAll(".caja");

        //cambio el color del segundo nodo

        CAJAS[1].style.color = 'blue';

//ITERAR LOS NODOS CON UN FOR

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

            CAJAS[i].style.color = 'red';

        }

//ITERAR LOS NODOS CON UN FOREACH

 //ITERAR CON FOREACH

        CAJAS.forEach((e) => {

            //cada una de las cajas se va a llamar e

            e.style.color = 'green';

        })

<ul id="lista1">

        <li>item1</li>

        <li>item2</li>

        <li>item3</li>

    </ul>

    <ul id="lista2">

        <li>item1</li>

        <li>item2</li>

        <li>item3</li>

    </ul>

    <script>

        //ME TRAIGO TODOS LOS li DE AMBAS LISTAS

        const ITEMS = document.querySelectorAll("li");

        console.log(ITEMS);

        //ACCEDER A ELEMENTOS DE LA SEGUNDA LISTA

        const SEGUNDALISTA = document.querySelector("#lista2");

        console.log(SEGUNDALISTA);

        const ITEMSLISTADOS = SEGUNDALISTA.querySelectorAll("li");

        console.log(ITEMSLISTADOS[1]);

//CREAR UN ELEMENTO DESDE JS Y METERLO EN EL DOM

 <div id="cajaParaTitulo">

<p id="parrafo1">Lorem ipsum dolor sit amet consectetur adipisicing elit.</p>

<p id="parrafo2">Lorem ipsum dolor sit amet consectetur.</p>

<p id="parrafo3">Lorem ipsum dolor sit.</p>

</div>

<script>

 //crear elemento

let nuevoElemento = document.createElement("h3");

//contenido

let nuevoContenido = document.createTextNode("Nuevo h3 creado desde JS");

//metemos el contenido en la etiqueta

nuevoElemento.appendChild(nuevoContenido);

//agregarle un estilo o formato

nuevoElemento.setAttribute("color" , "blue");

//introducir el elemento en el div correspondiente

document.getElementById("cajaParaTitulo").appendChild(nuevoElemento);

//meterle un estilo desde css

nuevoElemento.setAttribute("class" , "nombreClase");

//como posicionarlo entre el p2 y p3

        let parrafos = document.getElementsByTagName("p")[1];

        parrafos.appendChild(nuevoElemento);

        //COMO ELIMINAR

        parrafos.removeChild(nuevoElemento);

 //COMO ELIMINAR AL PRIMER ELEMENTO

        let cajaMayor = document.getElementById("cajaParaTitulo");

        cajaMayor.removeChild(document.getElementsByTagName("p")[0]);

//FILTRAR UN ARRAY Y DEJAR SOLO NUMEROS PARES.

const numeros = [10,2,3,5,9,72,8];

        //quiero un arreglo de numeros pares

        let numerosPares = [];  
// LO HACEMOS DE DE DISTINTA

        // mediante un FOR

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

            if(numeros[i]%2 == 0){

                numerosPares.push(numeros[i]);

            }

}

Filter siempre devuelve un arreglo igual de grande o menor. NUNCA mayor.

El método filter recibe una función que debe devolver un TRUE o FALSE.

A FILTER no se le puede mandar un string.

        //METODO FILTER

        let numeros\_pares = numeros.filter( (numero){

            return numero%2==0;

        })

        console.log(numeros\_pares);

//ES6 con ARROWFUNCTION

        let lospares = numeros.filter(item => {

            return item%2==0;

        })

        console.log(lospares);

//CON FILTER Y SPLIT ARMO UN ARRAY SOLO DE VOCALES

        const frase = 'esta frase es muy larga';

        const VOCALES = 'aeiou';

        let vocalesDeLaFrase = frase.split('').filter(function(letra){

            if(VOCALES.includes(letra)){

                return letra;

            }

        });

        console.log(vocalesDeLaFrase);

# ARRAYS

Los array pueden ser de elementos distintos.

Let array = [“hola”, 3, true, object];

push: añadir objetos al final del array

pop: sacas el final

shift: agrega al principio

unshift: saca el primer elemento

concat: no modifica al array original

let a = [1,2,3];

let b = [4,5,6];

let concatenado = [].concat(a,b);

console.log(concatenado);

Crea un nuevo array con los elementos de los dos arrays.

//slice no modifica el original

let array1 = [1,2,3,5,6,7,8,9,10];

console.log(array1.slice(5,9));

devuelve, en forma de array, los elementos de la posición 5 (inclusive) a la 9 (no inclusive).

Si le paso solo un elemento, me devuelve desde esa posicion, hasta el final.

console.log(array1.slice(5));

SPLICE

//splice SI modifica el array original

let array2 = [10,20,30,40,50,60,70,80,90,100];

let array3 = array2.splice(2,7);

el primer parámetro implica la posición, el segundo cuantos elementos quiero, por ende, en este caso, borra desde la segunda posición y borra 7 elementos.

**ASI QUEDARÍAN LOS DOS ARRAYS:**

Array2 = [10, 20, 100]

Array3 = [30, 40, 50, 60, 70, 80, 90]

**SPLICE PARA AGREGAR ELEMENTOS**

let array4 = [10,20,30,40,50,60,70,80,90,100];

let array5 = array4.splice(2,0,"A");

console.log(array4);

primer argumento: posición donde se agrega

segundo argumento: cuantos elementos borra antes de agregar. Si le pongo un 1, puede servir como para reemplazar el objeto.

tercer argumento: elemento/s que va a agregar.

let array6 = array4.splice(5,1,"c");

*array 6 va a devolver el elemento que se borra del array 4.*

let array7 = [10,20,30,40,50,60,70,80,90,100];

console.log(array7.join(" - "));

Devuelve el array, en forma de string, y separa los elementos con lo indicado en el join, si no se indica nada los separa con comas.   
*No modifica el original*.

**SPLIT**

let frase = "hola como andas";

console.log(frase.split(""));

transforma un string en un array. Las separación de cada elemento, se decide en lo enviado en el Split, si yo ahí pongo Split.(“o”), la o será la división de cada elemento. Por defecto (como en este ejemplo), todo es un elemento distinto.

**FILL**

let array8 = [1,2,3,4,5];

array8.fill("a",1,3);

reemplaza los elemento de la posición 1 a 3 con “a”.

[1, "a", "a", 4, 5]

**VER DE NUEVO COPYWITHIN**

**SORT**

//sort

let arrayNombres = ["juan", "fede", "emi", "zane"];

console.log(arrayNombres.sort());

ordena alfabéticamente.

let arrayNumeros = [5,6,32,10];

console.log(arrayNumeros.sort());

agarra lo primero de cada elemento (en el caso de 32 agarra el 3 solo.)

[10, 32, 5, 6]

**PARA ORDENAR NUMEROS DE MAYOR A MENOR**

const fc = function(a,b){

    return a-b;

}

console.log(arrayNumeros.sort(fc));

Función que devuelve de mayor a menor. Hay que hacer la función, pero ya por defecto, está comprobado que esto funciona, así que se usa eso.

[5, 6, 10, 32]

Esto es lo que devolvería. A diferencia del otro, este si te devuelve de mayor a menor.

# OBJETOS

let empanadas = [

    {

        id: 1,

        nombre: "carne",

        precio: 100

    },

    {

        id: 2,

        nombre: "jyq",

        precio: 105

    },

    {

        id: 3,

        nombre: "verdura",

        precio: 110

    },

    {

        id: 4,

        nombre: "humita",

        precio: 120

    },

]

// e = elemento (puede llevar el nombre que yo quiera,

//pero se acostumbra a poner eso)

const idPrecio = empanadas.map(e => [

    e.id , e.precio

]);

console.log(idPrecio);

/\*

const precios = empanadas.map(e =>e.precio);

let sumaPrecios = 0;

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

    sumaPrecios += precios[i];

}

console.log("El precio promedio es " + (sumaPrecios/precios.length));\*/

//reduce ---> devuelve solo un elemento o numero

let numeros = [1,2,3];

let reducido = numeros.reduce((acumulador , elemento) => acumulador + elemento ,50); //50 🡪 desde donde empieza a sumar, por defecto es 0

console.log(reducido);

sacar promedio de una manera más sencilla con reduce

# ASYNC AWAIT