Modulo 2 JavaScript

JavaScript es el unico lenguaje de programacion que entienden los ordenadores. Nos va a permitir realizar paginas dinamicas

Antes de cerrar el body meteremos :
<script type="text/javascript" src="./main.js"></script>
para enlazar nuestro main.js. Y al empezar nuestro main.js: 'use strict'

// comentar
/* xxx */ comentar más de una linea

Variables

Puedes declarar la variable y luego asignar el valor o directamente hacer las 2 juntas. siempre intentaremos usar camel case. Puede ser const o let en función de si varían o no.
let number = 5
const number = 5
A const no se le puede cambiar el valor

Document.querySelector()

Lo bueno es que document.querySelector() nos permite acceder a los elementos de HTML utilizando los selectores de CSS.

- etiqueta : const mainTitle = document.querySelector('h1');
- id: const mainTitle = document.querySelector('#mainTitle');
- clase: const mainTitle = document.querySelector('.mainTitle');

La primera idea es traerse la clase que tengamos en HTML y asignar u na const, y luego trabajar con ella por ejemplo añadiendo un innerHTML. Con este ultimo puedes añadir contenido directamente, quita lo que hay y pone nuevo, o hacer un sumatorio poniendo " +=" .

```
const listElement = document.querySelector('.list');

const content = '<li><a href="#">Home</a></li><li><a href="#">Contact</a></li>';

listElement.innerHTML = content;
```

```
// Cambiamos el contenido del elemento, indicando que sea igual al actual, más una nueva palabra
titleElement.innerHTML = titleElement.innerHTML + ' adalaber';
```

classList()

Con classList puedes añadir o quitar clases.:
Puedes añadir o quitar una o varias a la vez
sectionB.classList.add('hidden', 'otraClase', 'otraMas');

```
const sectionA = document.querySelector('.section-a');
const sectionB = document.querySelector('.section-b');

sectionA.classList.add('hidden');
sectionB.classList.remove('hidden');
```

Nota: antiguamente se declaraban variables con " var = 5" y se usaba document.getElementById('title') en vez que querySelector

Tipos de datos

- numericos (integers o float)
- String ("cadena de caracteres")
- Booleanos (false/true) (siempre true por defecto menos: null, undefined, 0, NaN "")
- Undefined (variable asignada sin valor)
- Nan (valor numerico que devuelve JS cuando no puede operar con ellos) " isNaN"

Convertir string a números:

Cuando recogemos valores de HTML, aunque sean números, (el contenido entre dos etiquetas, el valor de un input...) su tipo será string. Recuérda esto, siempre será un string Con parseInt() podemos convertir una cadena en un número entero.

```
const userAge = document.querySelector('.user__age');
console.log(userAge.innerHTML); // esto es un string
const yearsToRetirement = 67 - parseInt(userAge.innerHTML);
console.log(`Te quedan ${yearsToRetirement} años para jubilarte`);
```

Type of : conocer de que tipo es la variable.

concatenar:

```
'H' + 'o' + 'l' + 'a'; // Devuelve "Hola"
'Faltan ' + '3' + ' días'; // Devuelve "Faltan 3 días"
```

Longitud de string

```
const browserName = 'Mozilla';

console.log('Mozilla is ' + browserName.length + ' code units lon
```

Interpolar::

```
const name = 'Ada';
const surname = 'Lovelace';

console.log(`My name is ${name} ${surname}.`);
// 'My name is Ada Lovelace.'
```

Ejemplo en codepen

JavaScript .2

Operadores lógicos de comparación

AND --> && (y)

devuelve la primera expresión si esta es falsy, de lo contrario devuelve la segunda. Cuando trabajamos con booleanos devuelve verdadero SOLO si ambas condiciones son verdaderas.

OR --> || (o)

Devuelve la primera expresión si esta es truthy, de lo contrario devuelve la segunda expresión. Cuando trabajamos con booleanos devuelve verdadero si AL MENOS una condición se cumple.

NOT --> !

Lo contrario, siempre.

```
//OPERADOR AND / Y (&&)
var x = 4, y = 2;

var resultado = (x < 5 && y > 1); //Resultado true
resultado = (x > 5 && y > 1); //Resultado false

//OPERADOR OR / O (||)
var resultado2 = (x > 5 || y > 1); //Resultado true

//OPERADOR NOT / NO (!)
var resultado3 = !(x==y); //Resultado true

alert (resultado3);
```

```
var nota3 = 3;
if (nota3 < 5) {
    alert ("Has suspendido");
} else if (nota3 == 5){
    alert ("Has sacado un suficiente");
} else if (nota3 == 6){
    alert ("Has sacado un bien");
} else if (nota3 == 7 || nota3 == 8){
    alert ("Has sacado un notable");
} else {
    alert ("Has sacado un sobresaliente");
}
```

Condicinales:

La estructura viene ha decir " si esto es verdad, haz lo siguiente" y si no " lo otro". Pero podemos encadenar varios "if" que vendrían a decir, si esto es verdad haces esto, o si esto es verdad haces esto, o si esto es verdad haces esto, Y SI NO.... lo otro.

```
const age = 35;

if (age > 30) {
  console.log('Tienes más de 30 años'); /
} else {
  console.log('Como mucho tienes 30 años'
}
```

```
const age = 35;

if (age > 30) {
  console.log('Tienes más de 30 años'); //Esta
} else if (age >= 20) {
  console.log('Tienes entre 20 y 30 años'); //
} else if (age >= 10) {
  console.log('Tienes entre 10 y 19 años'); //
} else {
  console.log('Eres un niño entre 0 y 9 años')
}
```

ClassList.contains:

```
const activableSection = document.querySelector('.activable-section');

// Si contiene la clase hidden
if (activableSection.classList.contains('hidden')) {
  // Elimina la clase
  activableSection.classList.remove('hidden');
} else {
  // Si no, en caso contrario
  // Añade la clase hidden
  activableSection.classList.add('hidden');
}
```

Operadores ternario

Como un atajo del if...else.

- Escribimos ? y el código que se va a ejecutar si se cumple la condición
- Escribimos : y el código que se va a ejecutar si NO se cumple la condición

```
var numeroDeDia = 3;
var weekDays = ['lunes', 'martes', 'miércoles', 'jueves', 'viernes', 'sábado', 'domingo']
var dia = (numeroDeDia < weekDays.length) ? weekDays[numeroDeDia - 1] : 'dia incorrecto';
console.log(dia);
```

Aqui dice: dia es igual a " si 3 es menor que 7" dame la posicion del array 2. Si no , devuelve " es incorrecto".

Switch es un tipo de condicionar que nos simplifica el codigo cuando tenemos muchos else if.

```
const todayDate = 'Viernes';

if (todayDate === 'Lunes') {
  console.log('Hoy me tomo un café con Maricarmen')
} else if (todayDate === 'Martes') {
  console.log('Bajar al perro');
} else if (todayDate === 'Miércoles') {
  console.log('Vamos al cine a ver una peli');
} else if (todayDate === 'Jueves') {
  console.log('Juernesss');
} else if (todayDate === 'Viernes') {
  console.log('Cumpleaños de Paco');
} else if (todayDate === 'Sábado') {
  console.log('Comida con los suegros');
} else {
  console.log('Dormir hasta las 12');
}
```

--->

```
const todayDate = 'Viernes';

switch (todayDate) {
  case 'Lunes':
    console.log('Hoy me tomo un café con Maricarmen')
    break;
  case 'Martes':
    console.log('Bajar al perro');
    break;
  case 'Miércoles':
    console.log('Vamos al cine a ver una peli');
    break;
  case 'Jueves':
    console.log('Juernesss');
    break;
  case 'Viernes':
    console.log('Cumpleaños de Paco');
    break;
  case 'Sábado':
    console.log('Comida con los suegros');
    break;
  default:
    console.log('Dormir hasta las 12');
}
```

```
function esPar(valor) {
    return valor % 2 == 0;
}
```

JavaScript 3

Eventos: addEventLister()
Un evento representa una interacción, que normalmente es de la usuaria, tras la cual podemos realizar una acción. Lo que podemos hacer desde JavaScript es escuchar y responder a estos eventos. ¿Cómo? Creando el código que se va a ejecutar cuando el evento sucede. Añadiendo un evento a un click de un boton quedaria asi:
const button = document.querySelector('.alert');
button.addEventListener('click', (handleClickButton)

Eventos de ratón:
click
mouseover: raton sobre el elemento
mouseout: quitar el raton del elemento

Eventos de teclado:
- keydown: pulsar tecla
- keyup: soltar tecla

Eventos de elementos, formularios
- change: cambias contenido input
- submit: boton submit de formulario
- reset: reset de formulario
- blur (cuando quites el foco)

Funciones

Las funciones se declaran, normalmente con parámetros o argumentos (x, y, j...) que usaremos dentro. Y luego las funciones se ejecutan con los parámetros o argumentos reales que queramos, y así pueden ser reutilizadas.

DECLARACIÓN:

```
function sum(a, b) {
  const result = a + b;

  return result;
}
```

EJECUCICIÓN:

```
const sumResult = sum(3, 4);
```

Devuelve 7. pero si le paso por parámetro (5, 5) pues devolverá 10.

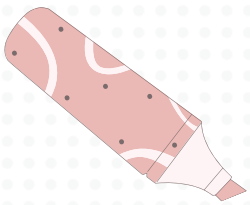
Funciones Arrow

Es una forma diferente de declarar funciones. Si solo tiene un parametro podemos no poner los parentesis, y si solo hace una funcion tambien nos podemos ajhorrar el return. ejemplos

```
const sum = (a, b) => {
  return a + b;
};
```

```
const getWaitingTime = minutes => {
  return `Please, wait ${minutes} minutes`;
};
```

```
const getWaitingTime = minutes => `Please, wait ${minutes} minutes`;
```



¡Esa plaza es mía!