

Semana 3

Clase 4 y 5

Tema 0. Javascript: conociendo el lenguaje

Tema 2. JavaScript en el navegador

EXPLORADOR

...

✓ CLASE 4

JS ejemplo_js.js

CONOCIENDO JAVASCRIPT

Comentarios

JS ejemplo_js.js ●

JS ejemplo_js.js

```
1  /*
2  * Comentarios de bloque
3  */
4
5
6  //Esto es un comentario de linea
7
```

Tipos de datos

- Cadenas de texto
- Números → no diferencia de entero o decimal, para el es un Number
- Booleano
- Array → es una mezcla entre Array y ArrayList
- Objetos → aquí podremos sacar los ficheros JSON
- Indefinido (undefined) → tipo único que tiene JavaScript, es un comodín que vale para todo

- Como JavaScript está pensado en el navegador, nosotros tenemos que ver todo en consola.
- JavaScript no se puede ejecutar fuera del navegador... solución:

- Tener instalado **NodeJS**



<https://nodejs.org/es>

- Y es aconsejable la extensión de Code Runner

Run JavaScript Everywhere

Node.js® is a free, open-source, cross-platform JavaScript runtime environment that lets developers create servers, web apps, command line tools and scripts.

Get Node.js®

Get security support
for EOL Node.js versions

```
PROBLEMAS  SALIDA  TERMINAL  ...  [ ] zsh

● damiansualdea@Mac-Studio-de-Damian-2 Clase 4 % node -v
v22.20.0
● damiansualdea@Mac-Studio-de-Damian-2 Clase 4 % npm -v
10.9.3
○ damiansualdea@Mac-Studio-de-Damian-2 Clase 4 % [ ]
```



```
console.log("Hola a tod@s")
```

Run Code
Ir a definición

```
[Running] node "/Users/damiansualdea/Desktop  
Hola a tod@s
```

```
[Done] exited with code=0 in 0.257 seconds
```

- Para los tipos de datos podemos utilizar **typeof**

```
console.log(typeof("Hola a tod@s"))
```

```
[Running] node  
string
```

```
//declarar variables  
let a = 3  
console.log(typeof a)
```



```
[Running] node ",  
number
```

```
//tipado débil, he cambiado el tipo...  
a = "Hola DWEC"  
console.log(typeof a)
```

```
[Running] node  
string
```

```
//por defecto  
let b = 3  
console.log(typeof b)
```

```
[Running] node "/Users/dan  
undefined
```

```
//para variables muy globales (casos puntuales)  
var c = true
```

```
//definición de constantes  
const D = 23
```

```
//ERROR:
```

```
D = 45
```

```
//el error se produce en ejecución (navegador)  
//en compilación no avisa
```

```
[Running] node "/Users/damiansualdea/Deskt  
/Users/damiansualdea/Desktop/DWEC/Clase 4/  
) = 45  
^
```

TypeError: Assignment to constant variable
at Object.<anonymous> (/Users/damiansu
at Module._compile (node:internal/modu
at Object..js (node:internal/modules/c

Operadores

- Operadores aritméticos
- Operadores de comparación (OJO)
- Operadores lógicos
- Operadores de asignación

Operadores aritméticos

Operador	Ejemplo	Resultado	Descripción
+	5 + 2	7	Suma
-	5 - 2	3	Resta
*	5 * 2	10	Multiplicación
/	5 / 2	2.5	División
%	5 % 2	1	Módulo (resto de la división)
**	5 ** 2	25	Potencia

Operadores de comparación

Operador	Ejemplo	Resultado	Descripción
==	5 == "5"	true	Igualdad (compara valores, ignora el tipo)
===	5 === "5"	false	Estrictamente igual (compara valor y tipo)
!=	5 != "5"	false	Diferente (ignora el tipo)
!==	5 !== "5"	true	Estrictamente diferente (valor o tipo distintos)
>	7 > 5	true	Mayor que
<	7 < 5	false	Menor que
>=	7 >= 7	true	Mayor o igual que
<=	7 <= 6	false	Menor o igual que

Compara su representación...

```
let texto = "3"
console.log("Tipo de dato texto es: "+typeof texto)

let numero = 3
console.log("Tipo de dato num es: "+typeof numero)

console.log(texto == numero)
console.log(texto === numero)
```

```
Tipo de dato texto es: string
Tipo de dato num es: number
true
false
```

```
console.log(numero+texto)
console.log(typeof(numero+texto))
```

```
33
string
```

Operadores lógicos

Operador	Ejemplo	Resultado	Descripción
&&	(5 > 2 && 10 > 5)	true	AND: ambas condiciones deben ser verdaderas
	(5 > 10 10 > 5)	true	OR: basta con que una condición sea verdadera
!	!(5 > 2)	false	NOT: invierte el valor lógico

Operadores de asignación

Operador	Ejemplo	Equivale a	Resultado
<code>+=</code>	<code>x += 5</code>	<code>x = x + 5</code>	Suma y asigna
<code>-=</code>	<code>x -= 3</code>	<code>x = x - 3</code>	Resta y asigna
<code>*=</code>	<code>x *= 2</code>	<code>x = x * 2</code>	Multiplica y asigna
<code>/=</code>	<code>x /= 4</code>	<code>x = x / 4</code>	Divide y asigna
<code>%=</code>	<code>x %= 2</code>	<code>x = x % 2</code>	Resto y asigna
<code>**=</code>	<code>x **= 3</code>	<code>x = x ** 3</code>	Potencia y asigna

Operadores de cadena

Operador	Ejemplo	Resultado	Descripción
+	"Hola " + "Mundo"	"Hola Mundo"	Une (concatena) cadenas
+=	let saludo = "Hola"; saludo += " Juan";	"Hola Juan"	Une y asigna al mismo tiempo

Estructuras de control

- **Condicionales**

- **if / else if / else**
- **switch**
- **Operador ternario** → forma corta del if/else

- **Bucles**

- **for** → cuando sabes cuántas veces repetir
- **while** → repite mientras la condición sea verdadera
- **do...while** → ejecuta al menos una vez
- **for...of** → recorrer arrays
- **for...in** → recorrer propiedades de un objeto

Condicionales

```
let dia = 3;

switch (dia) {
  case 1:
    console.log("Lunes");
    break; ←
  case 2:
    console.log("Martes");
    break;
  case 3:
    console.log("Miércoles");
    break;
  default:
    console.log("Otro día");
}
```

```
//Condicional if
let edad = 18;

if (edad >= 18) {
  console.log("Eres mayor de edad");
} else {
  console.log("Eres menor de edad");
}
```

```
//Con más de una condición
let nota = 85;

if (nota >= 90) {
  console.log("Excelente");
} else if (nota >= 70) {
  console.log("Aprobado");
} else {
  console.log("Necesita mejorar");
}
```

```
let edad2 = 20;
let mensaje = (edad2 >= 18) ? "Adulto" : "Menor";
console.log(mensaje); // Adulto
```

Bucles

```
let contador = 1;

while (contador <= 3) {
  console.log("Iteración " + contador);
  contador++;
}
```

Iteración 1
Iteración 2
Iteración 3

```
let num = 5;

do {
  console.log("Número actual: " + num);
  num--;
} while (num > 0);
```

Número actual: 5
Número actual: 4
Número actual: 3
Número actual: 2
Número actual: 1


En bucles cambia...

```
let frutas = ["Manzana", "Banana", "Uva"];

for (let fruta of frutas) {
  console.log(fruta);
}
```

```
let persona = { nombre: "Ana", edad: 25 };

for (let clave in persona) {
  console.log(clave + ": " + persona[clave]);
}
```



Excepciones – paracaídas

Cuando el programa “se cae” con un error, el try...catch evita que se estrelle.

```
try {  
    // Código que puede dar error  
} catch (error) {  
    // Código que se ejecuta si ocurre un error  
} finally {  
    // Este bloque SIEMPRE se ejecuta  
}
```

Funciones

//Sin parámetros

```
function saludar() {  
  console.log("¡Hola, mundo!");  
}
```

➡saludar(); // Llamada a la función

//Con parámetros por defecto

```
function saludar2(nombre="José") {  
  console.log("¡Hola, " + nombre + "!");  
}
```

saludar2(); // ¡Hola, José!

//Con parámetros

```
function saludar(nombre) {  
  console.log("¡Hola, " + nombre + "!");  
}
```

saludar("Ana"); // ¡Hola, Ana!

//Función con retorno (toda la vida)


```
function sumar(a, b) {  
  return a + b;  
}
```

let resultado = sumar(5, 3);
console.log(resultado); // 8

Cosas diferentes con funciones

//Funciones anónimas | Function expressions

```
const sumar = function(a, b)
{
  return a + b;
};
```



```
console.log(sumar(3, 4)); // 7
```

//Funciones flecha (Arrow functions, ES6)

```
const restar = (a, b) => {
  return a - b;
}
```

```
console.log(restar(10, 3)); // 7
```

```
const restar2 = (a, b) => a - b;
```

```
console.log(restar2(10, 3)); // 7
```


Arrays

```
//Declaración de arrays
const frutas2 = ["Manzana", "Banana", "Uva"];
console.log(frutas2); // ["Manzana", "Banana", "Uva"]
```

```
//Acceder y modificar elementos
console.log(frutas2[0]); // Manzana
frutas2[1] = "Pera";      // cambiamos Banana por Pera
console.log(frutas2);     // ["Manzana", "Pera", "Uva"]
```

```
//Cuidado
console.log(frutas2[345])
```



```
//Recorrer un array
for (let fruta of frutas2) {
  console.log(fruta);
}
```

```
frutas2.push(34)
console.log(frutas2)
```

```
[ 'Manzana', 'Pera', 'Uva', 34 ]
```

Arrays: métodos básicos

Método	Descripción	Ejemplo	Resultado
push()	Agrega un elemento al final	frutas.push("Naranja")	["Manzana", "Pera", "Uva", "Naranja"]
pop()	Elimina el último elemento	frutas.pop()	["Manzana", "Pera"]
unshift()	Agrega un elemento al inicio	frutas.unshift("Fresa")	["Fresa", "Manzana", "Pera", "Uva"]
shift()	Elimina el primer elemento	frutas.shift()	["Pera", "Uva"]
indexOf()	Devuelve el índice del elemento (o -1 si no existe)	frutas.indexOf("Uva")	2
includes()	Verifica si existe un elemento	frutas.includes("Pera")	true

Arrays: métodos para recorrer

Método	Descripción	Ejemplo	Resultado
forEach()	Recorre el array ejecutando una función	frutas.forEach(f => console.log(f))	Muestra cada fruta
map()	Crea un nuevo array transformado	[1,2,3].map(n => n*2)	[2, 4, 6]
filter()	Crea un nuevo array con los que cumplen una condición	[1,2,3,4].filter(n => n%2===0)	[2, 4]
reduce()	Reduce el array a un solo valor	[1,2,3].reduce((a,b)=>a+b)	6

Arrays: otros métodos

Método	Descripción	Ejemplo	Resultado
sort()	Ordena el array (alfabéticamente por defecto)	<code>["Banana","Manzana"].sort()</code>	<code>["Banana","Manzana"]</code>
reverse()	Invierte el orden del array	<code>[1,2,3].reverse()</code>	<code>[3,2,1]</code>
join()	Convierte el array en un string separado por comas (o por el separador indicado)	<code>["A","B","C"].join("-")</code>	<code>"A-B-C"</code>
slice()	Devuelve una parte del array sin modificar el original	<code>[1,2,3,4].slice(1,3)</code>	<code>[2,3]</code>
splice()	Añade, quita o reemplaza elementos en el array	<code>frutas.splice(1,1,"Mango")</code>	reemplaza en la posición 1

Objetos en JavaScript

Tema2.

JavaScript en el navegador

Ahora empieza lo chulo...

¿Qué vamos a hacer?

- A través de JavaScript vamos a ser capaces de añadir, modificar y gestionar todo el HTML, cambiar las clases a un elemento... el objetivo es hacerlo dinámico.
- No necesitamos que esté todo el HTML generado previamente, sino que vamos a poder añadir o modificar según lo necesitamos

Ejemplo: Quiero hacer una página que cargue todos los autobuses que van de una ciudad a otra. Proceso:

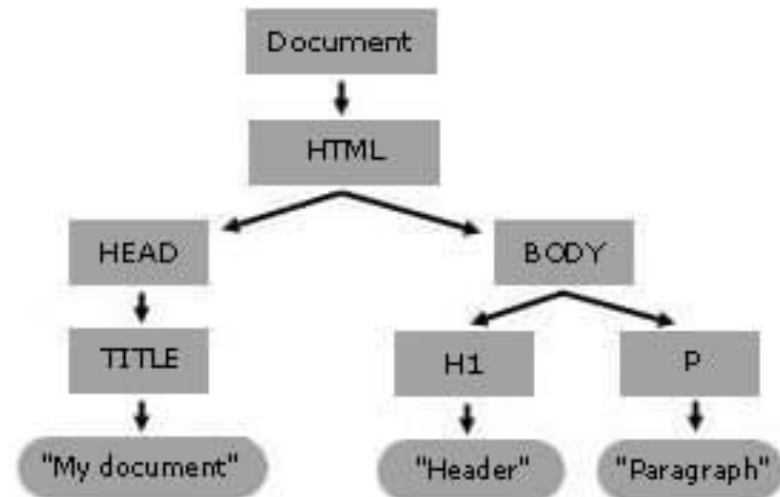
- Lo que hacemos es acceder a un servidor (**API Rest**).
- Me dará un fichero en formato **JSON**
- **Problema:** a lo mejor un día hay muchos autobuses, pocos y ninguno... no lo sabemos.
- **Solución:** Crearlos de forma dinámica... desde JavaScript

Son páginas dinámicas y así podremos trabajar con eventos, datos, animaciones, etc.

2.2. Modelo de objetos del documento (DOM)

- El **DOM** es una representación en memoria del documento HTML que muestra un navegador.
- Cada uno de los elementos HTML que conforman el documento se convierte en objetos en memoria con sus propiedades y métodos a los que es posible acceder mediante una API.
- Todo cambio que se realice en el DOM cambiará la visualización del sitio web en el navegador.


```
<html lang="en">
  <head>
    <title>My Document</title>
  </head>
  <body>
    <h1>Header</h1>
    <p>Paragraph</p>
  </body>
</html>
```



Eventos

- Los **eventos** son sucesos que le ocurren a elementos HTML; JavaScript se puede suscribir a dichos eventos y actuar en consecuencia.
 - Por ejemplo, si el usuario pulsa sobre un botón, dispara el evento **onClick** y JavaScript, al suscribirse a ese evento, puede actuar en consecuencia mostrando una alerta.
- Un **evento**, por tanto, es una manera de comunicación: un **elemento dispara un evento** y hay manejadores que lo tratan.
- Un **manejador** es una **función** que se ejecuta en el **momento en el que se lanza un evento** al que está suscrito

Los más utilizados:

- **Eventos del ratón:** mousedown, mouseenter, mouseleave, mousemove, mouseout, mouseover, mouseup, dblclick, click.
- Eventos **táctiles:** touchstart, touchmove, touchend, touchcancel.
- Eventos del **teclado:** keydown, keypress, keyup.
- Eventos de **formulario:** focus, change, submit.
- Eventos de la ventana: scroll, resize, load.

https://developer.mozilla.org/es/docs/Web/API/Document_Object_Model/Events


Document Ready

- La creación del DOM **no es un proceso inmediato**: el navegador, para mostrar una página, tiene que leer el documento HTML y convertirlo a un objeto en memoria.
- **Antes** de tener el DOM listo, el **navegador empezará a ejecutar JavaScript**; es por ello por lo que el código JavaScript que vaya a interactuar con el DOM tiene que esperar a que el DOM se haya generado, de otra manera no podrá interactuar con él porque literalmente, no hay nada con lo que interactuar.
- Para ello lo más sencillo es añadir un manejador al evento ***DOMContentLoaded*** que dispara el documento cuando el DOM ya está listo.

✓ CLASE 5

→  index.html

→  script.js

→  style.css

 index.html ●

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="style.css">
  <title>Manipulando el DOM</title>
</head>
<body>

  <script src="script.js"></script>

</body>
</html>
```

 script.js ●

//Siempre!!!!

```
document.addEventListener('DOMContentLoaded', function(event) {
  // Código JavaScript que manipula el DOM

  })
```

Selectores y métodos de acceso

- Un documento HTML no es más que un árbol de nodos: hay un nodo padre ***document*** del que cuelgan el resto de los elementos HTML; estos son nodos del árbol y si un elemento HTML tiene texto dentro, es un nodo de tipo texto.
- Conocer esta **estructura de árbol** es importante para **recorrer el DOM** en busca de elementos: cada nodo tendrá **únicamente un padre (excepto *document*)** y un **padre** podrá **tener uno o varios hijos**. Tanto desde ***document***, como desde cualquier nodo, se puede recorrer el DOM.

Métodos de selección del DOM

Método	Descripción	Ejemplo	Resultado
getElementById(id)	Devuelve un único elemento con el id indicado.	document.getElementById("titulo")	<h1 id="titulo">Hola</h1>
getElementsByName(name)	Devuelve una NodeList con los elementos que tengan un atributo name.	document.getElementsByName("usuario")	[<input name="usuario">, ...]
getElementsByTagName(tag)	Devuelve una HTMLCollection con todos los elementos que tengan esa etiqueta.	document.getElementsByTagName("p")	[<p>..</p>, <p>..</p>]
getElementsByClassName(class)	Devuelve una HTMLCollection con todos los elementos de esa clase.	document.getElementsByClassName("parrafo")	[<p class="parrafo">..</p>]
querySelector(selector)	Devuelve el primer elemento que cumpla con el selector CSS.	document.querySelector(".parrafo")	<p class="parrafo">...</p>
querySelectorAll(selector)	Devuelve una NodeList (estática) con todos los elementos que cumplan el selector CSS.	document.querySelectorAll("p")	[<p>..</p>, <p>..</p>]

2.3. Manipulación del DOM

- Accediendo al DOM se puede manipular su contenido, el navegador de primeras mostrará lo que le llegue desde el documento HTML (primera carga del DOM), pero mediante JavaScript se puede modificar ese DOM creado y la visualización en el navegador reflejará esos cambios

Métodos para manipular el documento

Método	Descripción	Ejemplo
<code>createElement(etiqueta)</code>	Crea un nuevo elemento HTML. El argumento es la etiqueta que tendrá.	<pre>const nuevoParrafo = document.createElement("p");</pre>
<code>appendChild(node)</code>	Agrega un nodo hijo dentro de otro elemento.	<pre>document.body.appendChild(nuevoParrafo);</pre>
<code>removeChild(child)</code>	Elimina el nodo hijo indicado.	<pre>contenedor.removeChild(nuevoParrafo);</pre>

Manipulando elementos: Clases

Método

`classList.add('clase')`

`classList.remove('clase')`

`classList.toggle('clase')`

Descripción

Añade una clase al elemento.

Quita una clase del elemento.

Alternar la clase: si la tiene, la quita; si no, la añade.

Ejemplo

`titulo.classList.add("rojo");`

`titulo.classList.remove("rojo");`

`titulo.classList.toggle("activo");`

Manipulando elementos: Contenido

Propiedad

innerHTML

innerText

Descripción

Permite añadir o modificar contenido con **HTML**.

Permite añadir o modificar solo **texto plano**.

Ejemplo

```
caja.innerHTML = "<b>Texto en  
negrita</b>";
```

```
caja.innerText = "Texto simple";
```

Manipulando elementos: Atributos

Método

`getAttribute(nombre)`

`setAttribute(nombre, valor)`

Descripción

Obtiene el valor de un atributo.

Cambia o añade un atributo.

Ejemplo

```
imagen.getAttribute("src");
```

```
imagen.setAttribute("alt", "Foto  
de perfil");
```

index.html ●

```
<body>
```

```
<section id="principal">
```

```
</section>
```

```
<script src="script.js"></script>
```

```
</body>
```

style.css ●

```
body{
```

```
  margin: 20px;
```

```
  background-color: #f5deb3;
```

```
}
```

```
.parrafo{
```

```
  color: blue;
```

```
  size: 1.2em;
```

```
  font-family: 'Franklin Gothic Medium', 'Arial Narrow', Arial, sans-serif;
```

```
}
```

```
//Siempre!!!!
```

JS script.js

```
document.addEventListener('DOMContentLoaded', function(event) {  
    //Código JavaScript que manipula el DOM  
    const sectionPrincipal = document.getElementById('principal');
```

```
//Vamos a crear elementos
```

```
const divP //Añadimos la clase parrafo con elementos div y p (style.css)
```

```
const parrafo1.parrafo1.classList.add('parrafo')
```

```
parrafo1.in
```

```
//append con comas mete muchos nodos, appendchild sería correcto
```

```
divP.append(parrafo1);
```

```
//Añadimos
```

```
parrafo1.cl
```

```
divP.classList.add('contenedor');
```

```
//Lo acabo de meter en el documento HTML
```

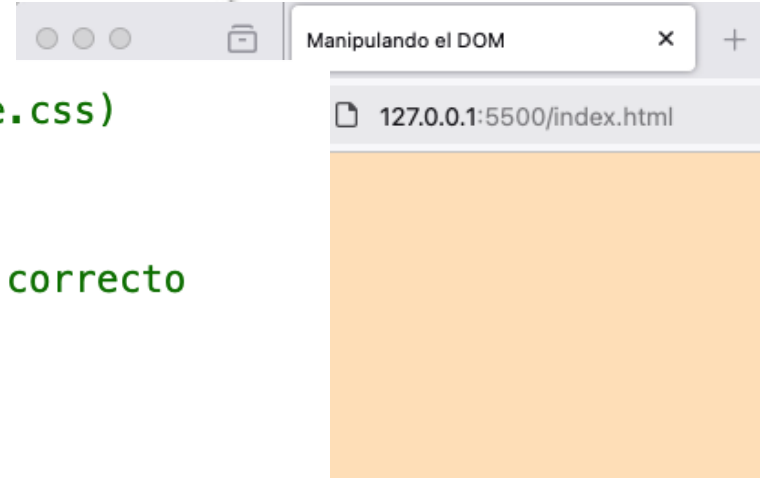
```
//append co sectionPrincipal.append(divP);
```

```
divP.append(parrafo1);
```

```
//Lo acabo de meter en el documento HTML
```

```
sectionPrincipal.append(divP);
```

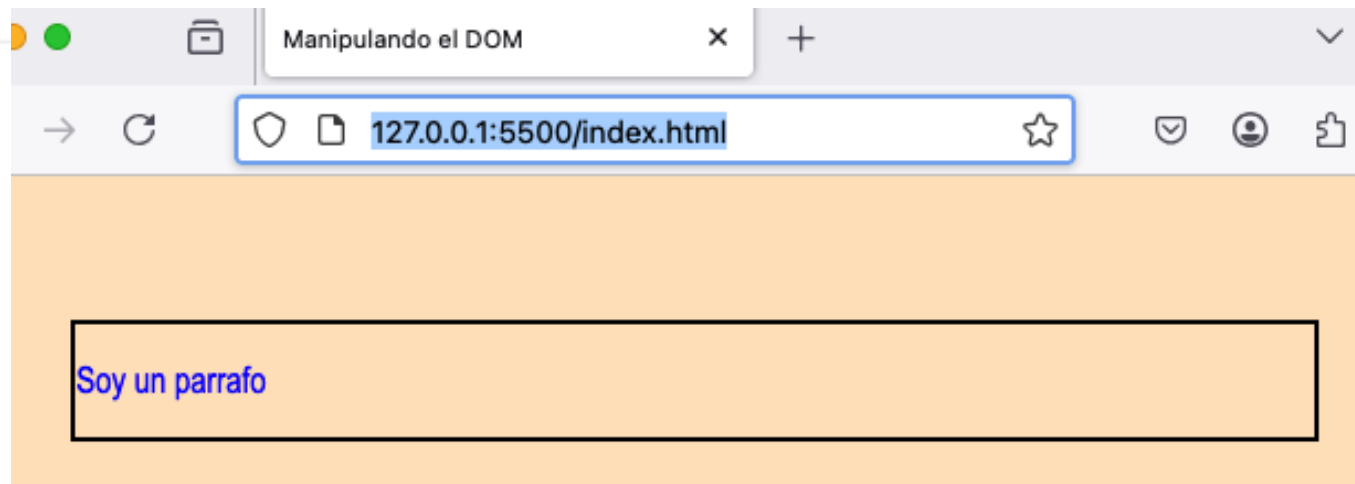
```
})
```



```
<section id="principal">  
  <div>  
    <p class="parrafo">Soy un parrafo</p>  
  </div>  
</section>
```

CSS style.css ●

```
.contenedor{  
  display: flex;  
  flex-direction: column;  
  border: 2px solid black;  
}
```



JS script.js ●

```
//Añadimos la clase parrafo con elementos div y p (style.css)  
parrafo1.classList.add('parrafo')
```

```
//append con comas mete muchos nodos, appendchild sería correcto  
divP.append(parrafo1);
```

```
divP.classList.add('contenedor');
```

```
//Lo acabo de meter en el documento HTML  
sectionPrincipal.append(divP);
```

```
<section id="principal">  
  <div class="contenedor"> flex  
    <p class="parrafo">Soy un parrafo</p>  
  </div>  
</section>  
<script src="script.js"></script>
```

A red arrow points from the right towards the "parrafo" class attribute in the HTML code snippet.

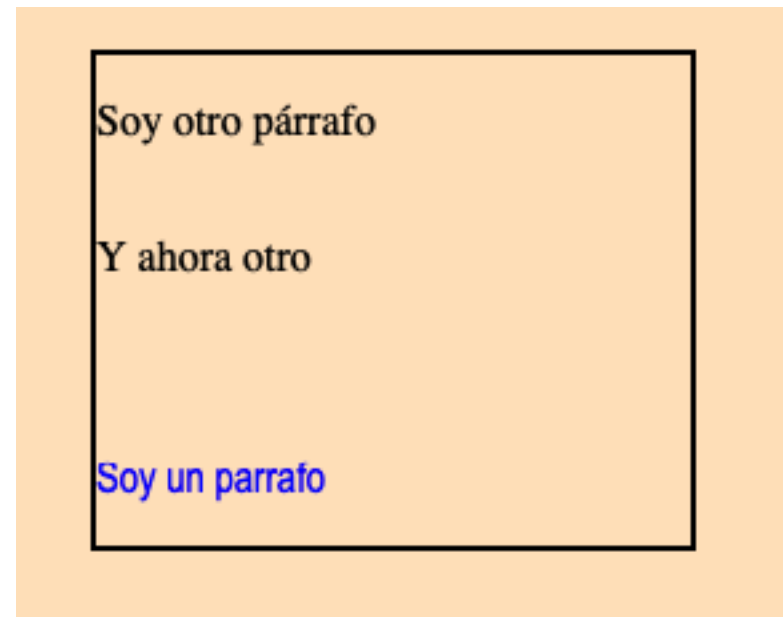
- No se utiliza innerHTML, no tenemos el control de la estructura

CSS style.css •

//Esto no se hace

`divP.innerHTML="<p>Soy otro párrafo</p>
 <p>Y ahora otro<p>"`

```
▼ <section id="principal">
  ▼ <div class="contenedor"> flex desbordamiento
    <p>Soy otro párrafo</p>
    <br>
    <p>Y ahora otro</p>
    <p></p>
    <p class="parrafo">Soy un parrafo</p>
  </div>
</section>
```



JS script.js ● Vamos a limpiar y continuar 😊

//Siempre!!!!

```
document.addEventListener('DOMContentLoaded', function(event) {  
  //Código JavaScript que manipula el DOM  
  const sectionPrincipal = document.getElementById('principal');  
  //1º  
  const divP = document.createElement('div');  
  divP.classList.add('contenedor');  
  //2º  
  const parrafo1 = document.createElement('p');  
  parrafo1.innerHTML = texto()  
  parrafo1.classList.add('parrafo')  
  //3º  
  divP.append(parrafo1);  
  
  //Lo acabo de meter en el documento HTML  
  sectionPrincipal.append(divP);  
  
  })  
  function texto(){  
    return "Soy un parrafo"  
  }  
}
```

CSS style.css ●

```
.btn{  
  padding: 10pt;  
  background-color: blue;  
  color: white;  
  border: 1px solid blueviolet;  
}  
  
.btn:hover{  
  background-color: aquamarine;  
  color: black;  
}
```

//Añadimos 3 botones

```
const btn_1 = document.createElement('button');  
const btn_2 = document.createElement('button');  
const btn_3 = document.createElement('button');
```

//Añadimos el estilo a los botones

```
btn_1.classList.add('btn');  
btn_2.classList.add('btn');  
btn_3.classList.add('btn');
```

//Metemos el texto al botón

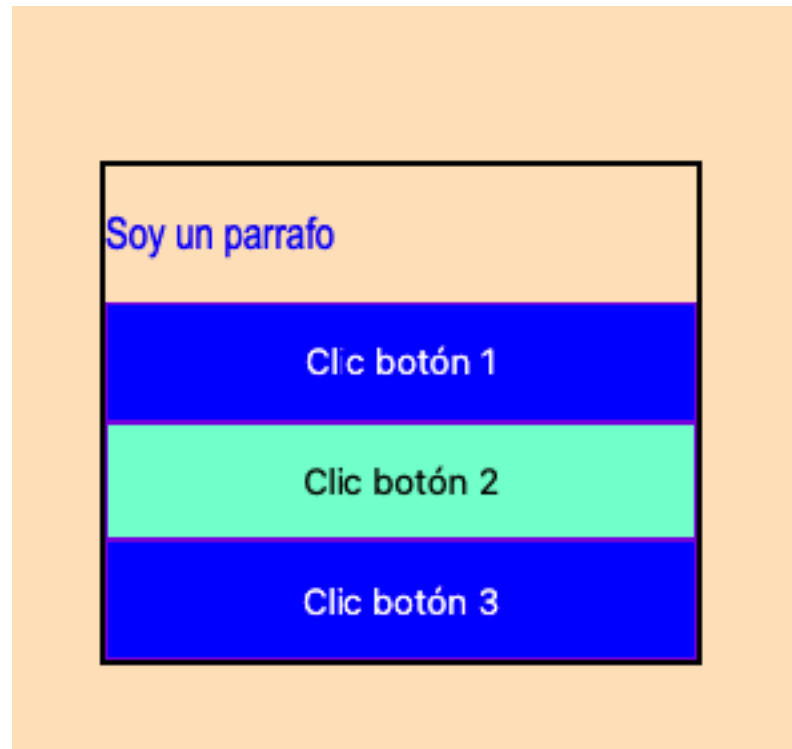
```
btn_1.innerText = "Clic botón 1";  
btn_2.innerText = "Clic botón 2";  
btn_3.innerText = "Clic botón 3";
```

//Lo incluimos

```
divP.append(btn_1, btn_2, btn_3)
```

//Lo acabo de meter en el documento HTML

```
sectionPrincipal.append(divP);
```



```
<body>  
  <section id="principal">  
    <div class="contenedor">  
      <p class="parrafo">Soy un parrafo</p>  
      <button class="btn">Clic botón 1</button>  
      <button class="btn">Clic botón 2</button>  
      <button class="btn">Clic botón 3</button>  
    </div>  
  </section>  
<script src="script.js"></script>
```

JS script.js ●

```
//Capturamos eventos!!!
```

```
btn_1.addEventListener('click', function(event){  
    alert("Botón 1 pulsado :)")  
});
```

```
//Lo acabo de meter en el documento HTML  
sectionPrincipal.append(divP);
```

