M1 - 8 - JS Introducción al DOM

Hasta ahora hemos visto como estructurar nuestra web con HTML y como darle estilo con CSS pero si únicamente utilizamos HTML/CSS, sólo podremos crear páginas “estáticas” , sin embargo, si añadimos Javascript, ya podremos crear páginas “dinámicas”.

Cuando hablamos de páginas dinámicas, nos referimos a que podemos dotar de la potencia y flexibilidad que nos da un lenguaje de programación para crear documentos y páginas mucho más ricas, que brinden una experiencia más completa y con el que se puedan automatizar un gran abanico de tareas y acciones.

¿Qué es el DOM?

Las siglas DOM significan **Document Object Model**, o lo que es lo mismo, la estructura del documento HTML. Una página HTML está formada por múltiples etiquetas HTML, anidadas una dentro de otra, formando un árbol de etiquetas relacionadas entre sí, que se denomina árbol DOM (o simplemente DOM).

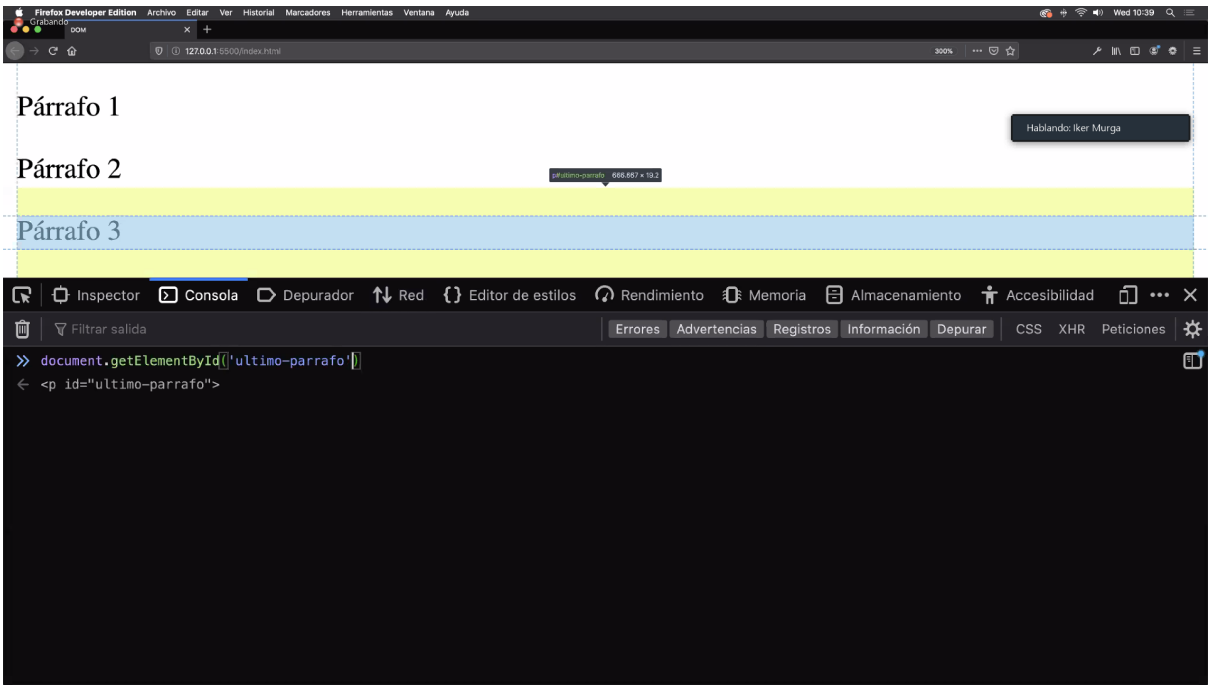
En Javascript, cuando nos referimos al DOM nos referimos a esta estructura, que podemos modificar de forma dinámica desde Javascript, añadiendo nuevas etiquetas, modificando o eliminando otras, cambiando sus atributos HTML, añadiendo clases, cambiando el contenido de texto, etc...

Al estar "amparado" por un lenguaje de programación, todas estas tareas se pueden automatizar, incluso indicando que se realicen cuando el usuario haga acciones determinadas, como por ejemplo: pulsar un botón, mover el ratón, hacer click en una parte del documento, escribir un texto, etc…

Acceder a elementos HTML

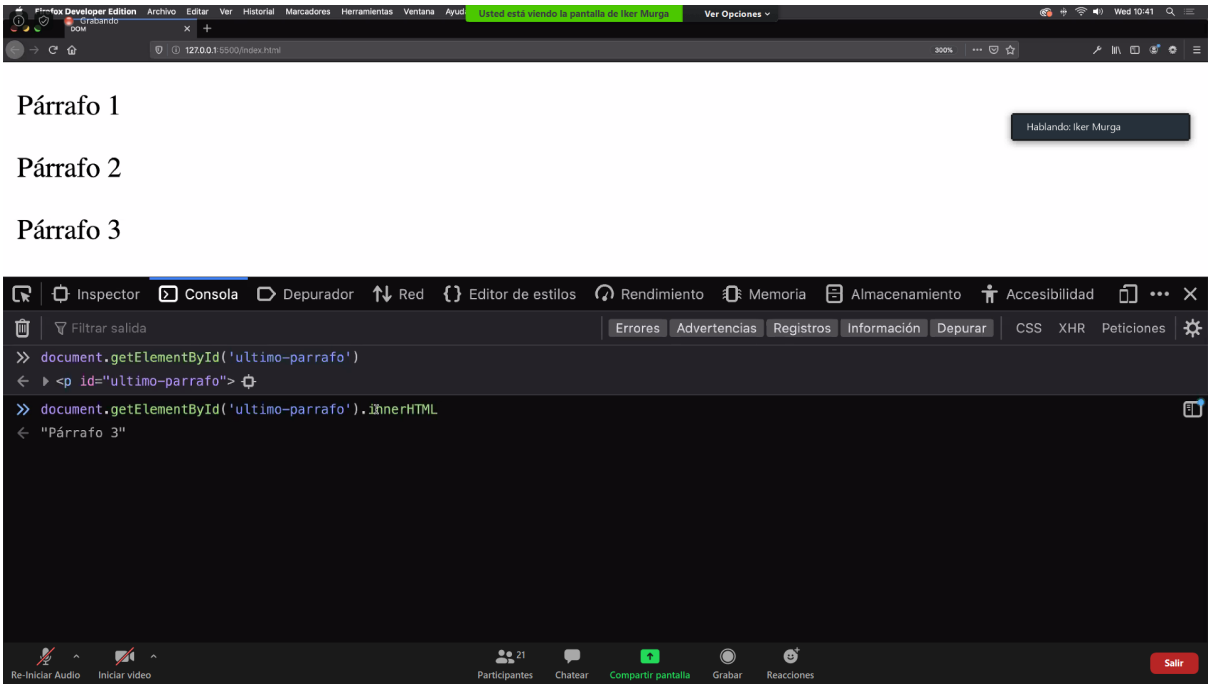
La forma de acceder al DOM es a través de un objeto llamado **document**, que representa el árbol DOM de la página o pestaña del navegador donde nos encontramos.

Para poder acceder al contenido de un elemento de nuestro “document” vamos a usar la función `.getElementById("id del elemento al que queremos acceder")` que nos permite seleccionar un elemento del DOM por su atributo `id=""`. Además de este método tenemos muchos otros como `.getElementsByClassName("clase que queremos seleccionar")` , que nos devuelve un array con todos los elementos que tengan la clase que pasemos como argumento, y muchos otros:



Dentro del elemento seleccionado podemos acceder a todas sus propiedades. Vamos a pedirle el contenido en HTML del mismo poniendo `.innerHTML` Luego este contenido podremos almacenarlo en una variable:

```
let contenido = document.getElementById("ID del elemento").innerHTML
```



Pero al igual que a una variable, nosotros a este elemento podemos modificarle el valor. para ello a esta línea la igualamos a un string con el código html que nosotros queramos introducir en el elemento que tenga ese ID

👉 *Aquí utilizaremos comillas “francesas” para poder poner strings “inteligentes”.*

```
document.getElementById("div1").innerHTML = `
<h1 id="mundo">Hola Mundo</h1>
<p id=>Estamos manipulando el DOM</p>
`;
```

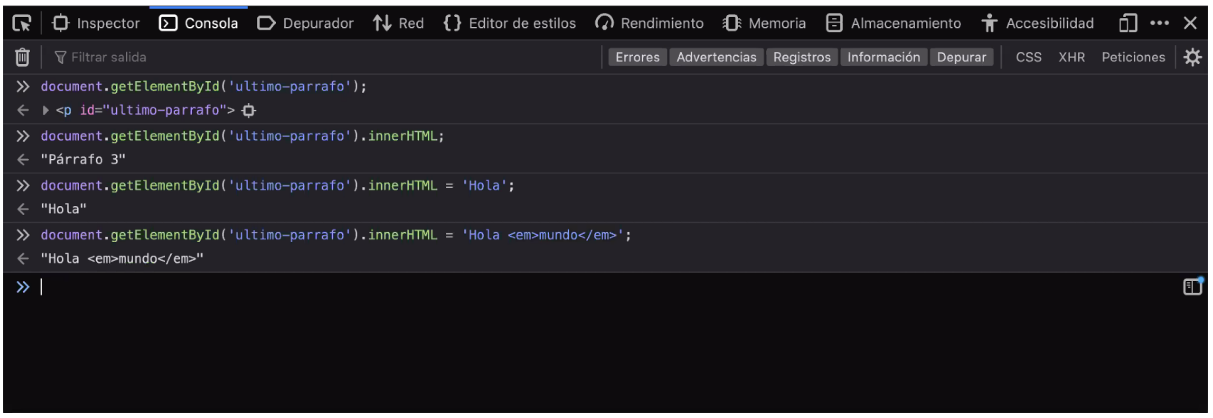
Hola Mundo

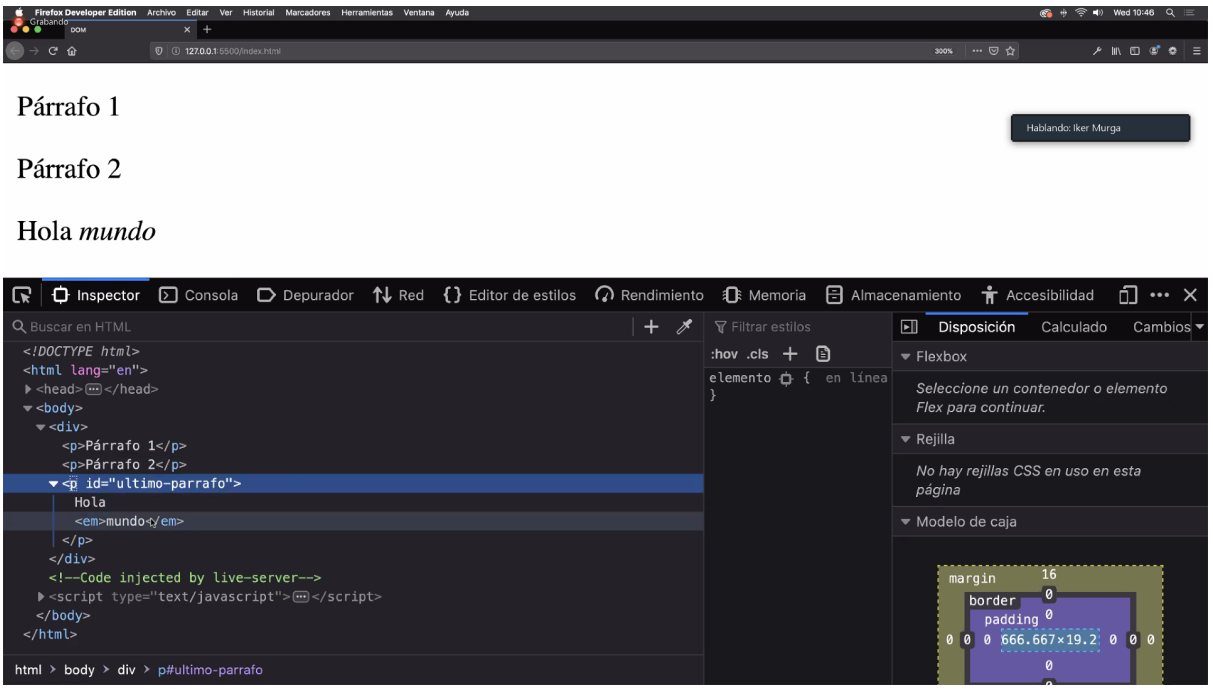
Estamos manipulando el DOM

Párrafo 1

Párrafo 2

Hola *mundo*





Modificar CSS

Al igual que podemos acceder al contenido HTML a través de `.innerHTML` podemos acceder al estilo css que afecta a un elemento HTML usando `.style` :

Con esto accederemos al estilo que afecta directamente a ese elemento y podremos modificar cualquiera de sus etiquetas:

```
document.getElementById("div1").innerHTML = "Adiós"
// Ahora cambiamos su color a "red"
document.getElementById("div1").style.color = "red"
// Luego ponemos su tamaño de letra en 2em
document.getElementById("div1").style.font-size = "2em"
// Por último ponemos el fondo en "white"
document.getElementById("div1").style.background-color = "white"
```

Adiós

Crear elementos en el DOM

Al igual que podemos modificar elementos existentes en nuestro DOM podemos crearlos. Para ello podemos utilizar el método `.createElement("elemento")` y acceder a su HTML interno con la propiedad `.innerHTML` ya vista anteriormente

```
//creamos el nuevo div
let newDiv = document.createelement('div')
//le insertamos nuestro HTML dentro
newDiv.innerHTML = '<p>Hola mundo!</p>'
```

Una vez creado el contenido, podemos utilizar el método .appendChild para “colgarlo” del elemento que queramos, en este caso, vamos a usar el body del document:

```
// "colgamos" el contenido del div
document.body.appendChild(newDiv)
```

Y con esto habriamos creado un elemento en nuestro DOM que no existía en el comento de la carga.

Eliminar elementos del DOM

Por último, podemos ver la forma de eliminar elementos de nuestro DOM, tanto preexistentes, como aquellos que acabemos de crear.

Para ello vamos a utilizar el mótodo `.remove()`

Imaginemos que ahora que ya hemos creado el elemento newDiv en el apartado anterior, queremos eliminarlo. Para ello simplemente podemos aplicar el método a la variable que contiene la selección:

```
newDiv.remove()
```

o podriamos seleccionar un elemento de nuestro dom con un id concreto y mediante el método `.getElementById("id")` seleccionarlo para aplicarle el método:

```
document.getElementById("div1").remove()
```