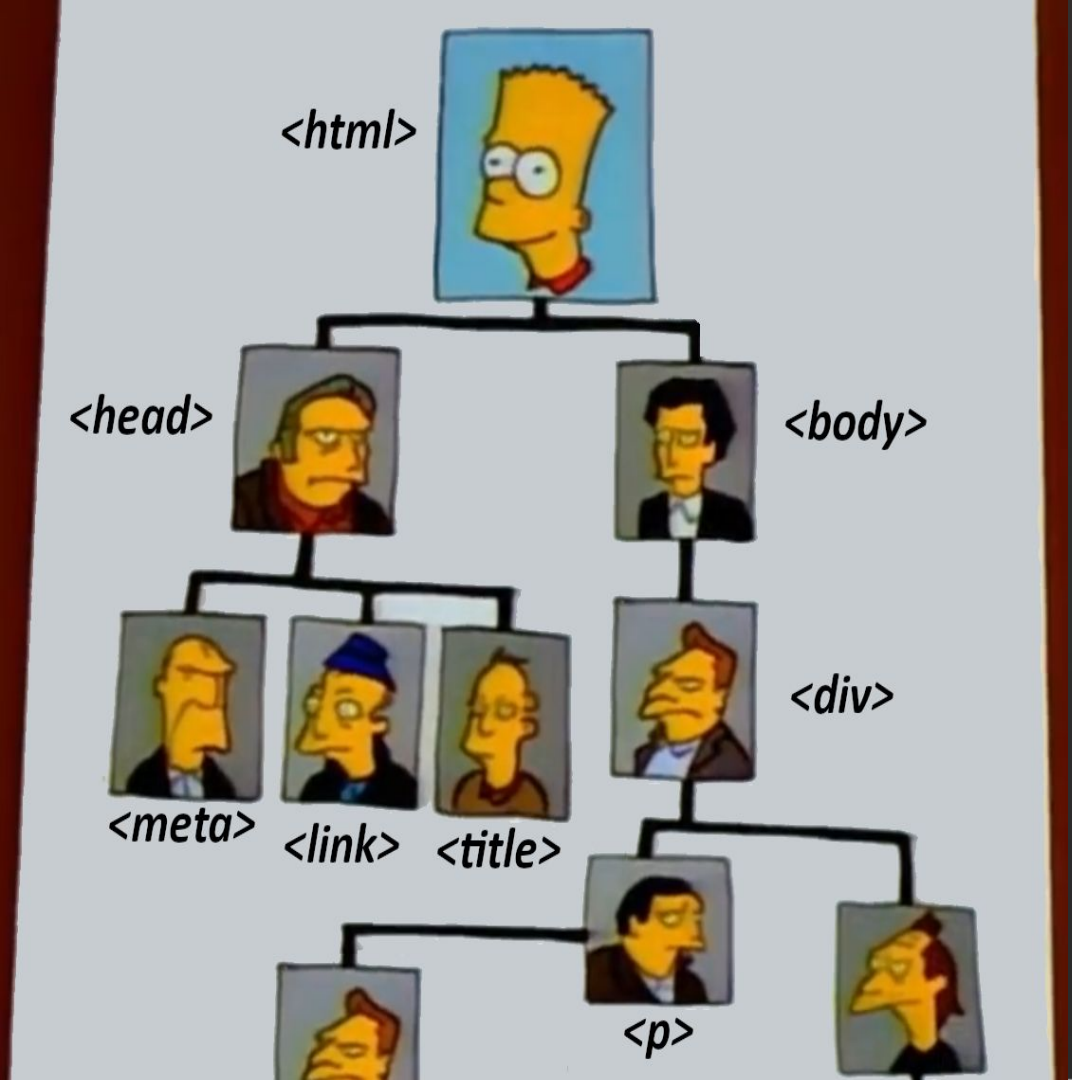
GM2

DOM



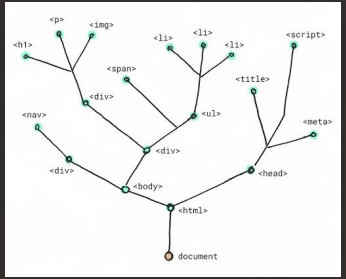
DOM
Document Object Model

El Modelo de Objetos del Documento (DOM) es una estructura de objetos generada por el navegador, la cual representa la página HTML actual. Con JavaScript la empleamos para acceder y modificar de forma dinámica elementos de la interfaz.

Es decir que, por ejemplo, desde JS podemos modificar el texto contenido de una etiqueta <h1>.

GM2

ESTRUCTURA DOM

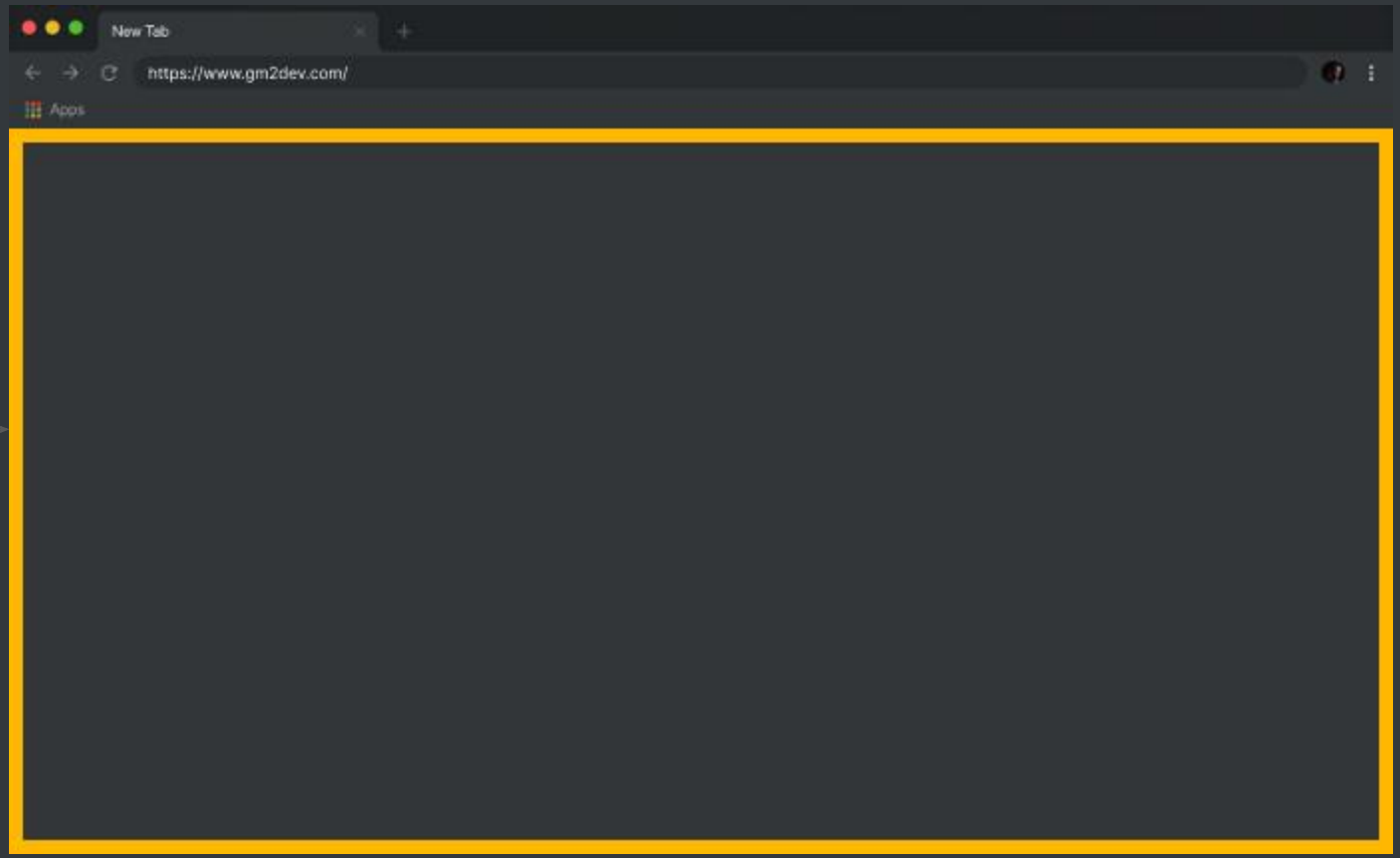


- Cada etiqueta HTML se transforma en un nodo de tipo "Elemento". La conversión de etiquetas en nodos se realiza de forma jerárquica.
- De esta forma, del nodo raíz solamente pueden derivar los nodos HEAD y BODY.
- A partir de esta derivación inicial, cada etiqueta HTML se transforma en un nodo que deriva del correspondiente a su "etiqueta padre".

Para acceder a los elementos de una página, usamos **selectores**. Cada selector puede retornar **un solo elemento o una lista de elementos.** Para poder hacer uso de los selectores debemos hacer uso del objeto **document,** ya que los selectores son métodos del mismo.

GM2

OBJETO DOCUMENT



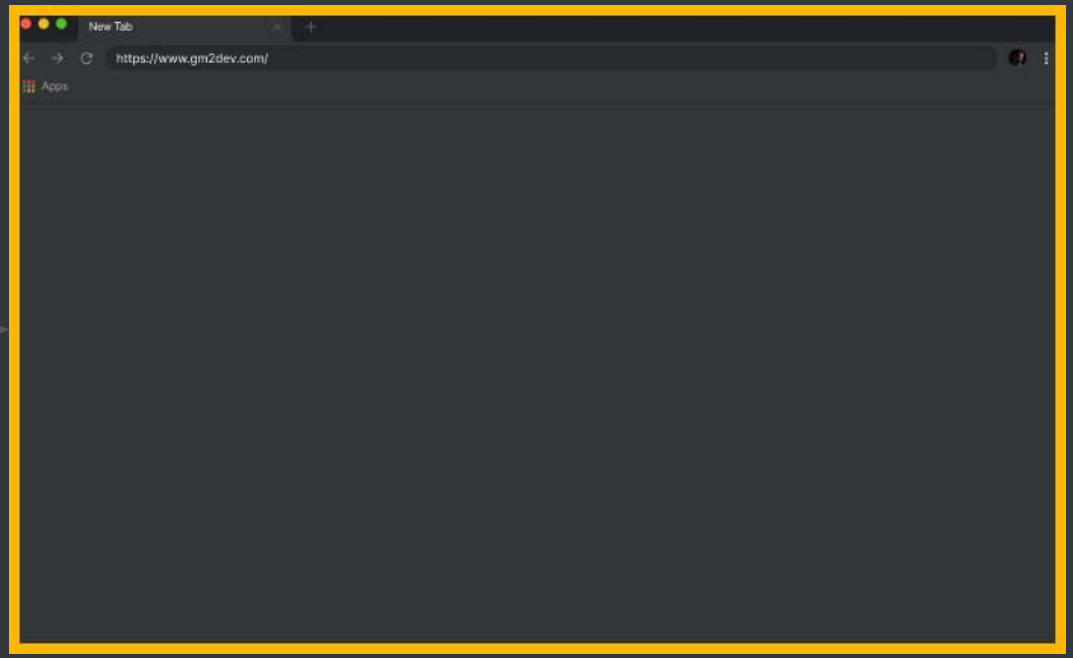
OBJETO **DOCUMENT**

OBJETO DOCUMENT

Representa al HTML y nos va a dar una interfaz, un conjunto de atributos y de métodos para poder efectivamente leer lo que tenemos en el HTML y si quisiéramos modificarlo. **El document es cargado dentro del objeto window** y tiene propiedades como title, url, etc..

GM2

OBJETO WINDOW



OBJETO WINDOW

OBJETO WINDOW

El objeto window es lo primero que se carga en el navegador.
Representa la ventana donde estamos navegando y nos da la interfaz para operar con el navegador. El objeto window tiene propiedades como **lenght**, **innerWidth**, etc

“

Para poder hacer modificaciones al **DOM** siempre tenemos que tener seleccionado el objeto que queremos modificar. Esto lo podemos hacer usando selectores.

”

.querySelector()

GM2

querySelector()

Este selector recibe un string que indica el selector CSS del elemento del DOM que estamos buscando. Por ejemplo:

```
{}
let titulo = document.querySelector('.title');
```

Nos va a retornar el **primer** elemento del HTML que contenga la clase “title”.

//<h1 class="title">Hello, World!</h1>

.querySelectorAll()

GM2

querySelectorAll()

Este selector recibe un string que indica el selector CSS del elemento del DOM que estamos buscando. Por ejemplo:

```
{}  let nombres=document.querySelectorAll('.titles');
```

Nos va a retornar **un listado** de elementos que coincidan con la búsqueda especificada.

```
NodeList(3) [h1.title, h2.title, p.title]
0: <h1 class="title">Hello, World!</h1>
1: <h2 class="title">Subheading</h2>
2: <p class="title">This is a paragraph.</p>
length: 3
```

.getElementById()

GM2

getElementById()

Este selector recibe un string con únicamente el nombre del **id** del elemento del DOM que estamos buscando.

Por ejemplo:

```
{}  let marca=document.getElementById('marca');
```

Nos va a retornar el **elemento cuyo id coincida con el deseado**.

Consola: // This is the main content.

GM2

Comparando selectores

querySelector()	querySelectorAll()	getElementById()
Retorna el **primer elemento** del DOM que cumple con la condición que buscamos.	Retorna **todos los elementos** del DOM que cumplen con la condición que buscamos.	Retorna el elemento del DOM **que cumpla con el id** que buscamos.

“

JavaScript nos da una propiedad y varios métodos que nos permiten hacer diversas acciones con el atributo class de un elemento.

”

classList.add()

GM2

classList.add()

Permite agregar una clase nueva al elemento que tengamos seleccionado.

```
let cita = document.querySelector('.cita');
cita.classList.add('italicas');
```

Antes

```
<p class="cita">El veloz murciélago comía feliz cardillo y kiwi</p>
```

Después

```
<p class="cita italicas">El veloz murcielago comía feliz cardillo y kiwi</p>
```

classList.remove()

GM2

classList.remove()

Permite quitarle una clase existente al elemento que tenemos seleccionado.

```
let cita = document.querySelector('.cita');
cita.classList.remove('cita');
```

Antes

```
<p class="cita">El veloz
murciélago comía feliz
cardillo y kiwi</p>
```

Después

```
<p class="">El veloz
murcielago comía feliz
cardillo y kiwi</p>
```

classList.toggle()

GM2

classList.toggle()

Revisa si existe una clase en el elemento seleccionado. De ser así, la remueve, de lo contrario, si la clase no existe, la agrega.

```
let cita = document.querySelector('p');
cita.classList.toggle('cita');
```

Antes

```
<p class="italicas">El
veloz murciélago comía
feliz cardillo y kiwi</p>
```

Después

```
<p class="italicas cita">El
veloz murcielago comía
feliz cardillo y kiwi</p>
```

classList.contains()

GM2

classList.contains()

Permite preguntar si un elemento tiene una clase determinada. Devuelve un **valor booleano.**

{}

```
let cita = document.querySelector('.italicas');
cita.classList.contains('cita'); // false
```

{}

```
let cita = document.querySelector('.italicas');
cita.classList.contains('italicas'); // true
```

GM2

En resumen

.add()	.remove()	.toggle()
Agrega la clase al elemento.	Elimina la clase del elemento.	Agrega la clase, si es que no la tiene. En caso de tenerla, la remueve.

.innerHTML

GM2

.innerHTML

Si queremos **leer o modificar** el contenido de una etiqueta HTML, vamos a utilizar esta propiedad.

```
{}  document.querySelector('div.nombre').innerHTML ;
```

En este caso, la utilizamos para leer el contenido, pero ¿qué pasa si queremos modificarlo?

GM2

.innerHTML

Para **modificar** el contenido de una etiqueta HTML, vamos a utilizar esta propiedad de la siguiente manera.

```
{}  document.querySelector('div.nombre').innerHTML = '<p>Darío</p>';
```

Si utilizamos la propiedad de esta forma, todo el contenido que teníamos en el **div** con clase **nombre** se va a cambiar por el string “Darío”.

GM2

.innerHTML

Para **modificar** el contenido de una etiqueta HTML, sin perder el contenido que ya estaba, vamos a utilizar esta propiedad de la siguiente manera:

```
{}  document.querySelector('div.compras).innerHTML += '<p>Papitas</p>';
```

De esta forma, estamos agregando al **div** con clase **compras** la palabra "Papitas". De tal manera que si lo leyéramos, diría:

```
{}  <div class="compras"> "Jamón, Queso, Pan" Papitas</div>
```

Contenido que ya estaba.

Contenido que agregamos.

.innerText

GM2

.innerText

Si queremos **leer o modificar** el texto de una etiqueta HTML, vamos a utilizar esta propiedad.

```
{}  document.querySelector('div.nombre').innerText ;
```

En este caso, si en mi **div** con clase **nombre** estuviera escrito “Maria”, la propiedad me retornaría el **string** “Maria”.

Si queremos **guardar** el valor, debemos asignar esa línea de código a una variable. De otra manera, cuando la ejecución continúe, se perderá el valor que hayamos buscado.

GM2

.innerText

Si queremos **modificar** el texto de una etiqueta HTML, vamos a utilizar esta propiedad de la siguiente manera:

```
{}  document.querySelector('div.nombre').innerText = 'Leo';
```

Si utilizamos la propiedad de esta forma, todo el texto que teníamos en el **div** con clase **nombre** se va a cambiar por el string “Leo”.

GM2

.innerText

Si queremos **agregar** contenido al texto de una etiqueta HTML, vamos a utilizar esta propiedad de la siguiente manera:

```
{}  document.querySelector('div.nombre').innerText += 'Messi';
```

En este caso, lo que sucedería es similar a lo que sucede con el otro selector, pero el texto se incluiría dentro de la etiqueta div, quedando:

```
{}  <div class="nombre">Leo Messi</div>
```

Contenido que ya estaba.

Contenido que agregamos.

Propiedad style

GM2

Propiedad Style

Nos permite leer y sobreescribir las reglas CSS que se aplican sobre un elemento que hayamos seleccionado. Nótese que las reglas CSS que llevaban guiones (como **font-size**), en JavaScript se escriben en camelCase (es decir: **fontSize**).

Lista de estilos

{}

```
let titulo = document.querySelector('.title');
titulo.style.color = 'cyan';
titulo.style.textAlign = 'center';
titulo.style.fontSize = '12px';
titulo.style.backgroundColor = '#dddddd';
```