<codoacodo/>

Agencia de
Aprendizaje
a lo largo
de la vida

Desarrollo Fullstack

BA

Les damos la bienvenida

Vamos a comenzar a grabar la clase

Clase 16

Javascript Objetos

- Literales
- Funcionales
- Clases
- For In

Clase 17

Javascript para la Web

- DOM
- Manejo de eventos

Clase 18

Javascript para la Web

- DOM Parte II

JAVASCRIPT

DOM y Eventos

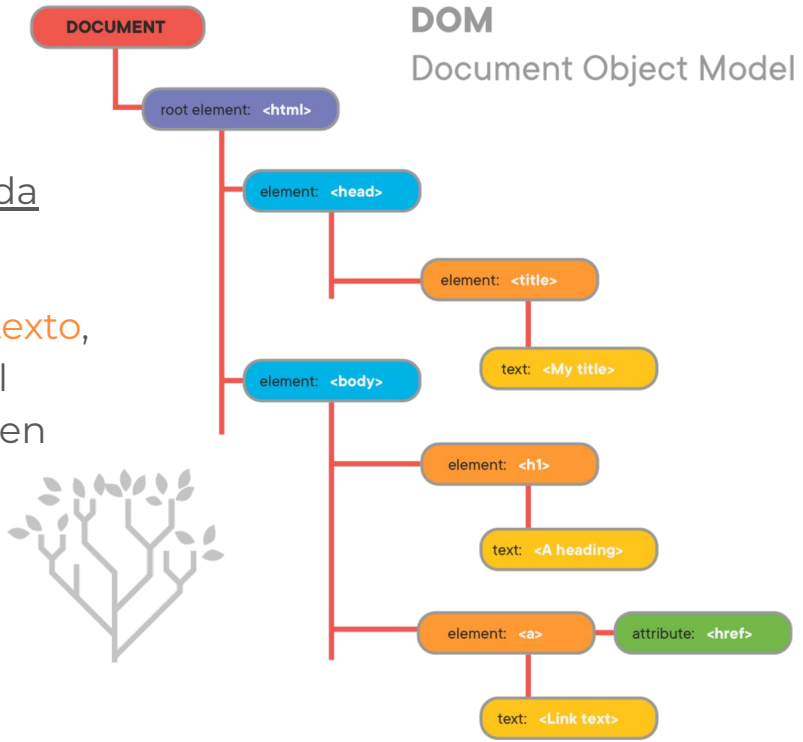
DOM

El Document Object Model es la representación que **hace el navegador** de los elementos en un documento **HTML**. Esto nos permite navegar con Javascript a través de esta estructura en forma de **árbol**.

Nodos

El DOM es un **árbol de elementos**, donde cada elemento es un ***NODO***.

Estos pueden ser etiquetas HTML, atributos, texto, **comentarios** o el mismo Document que es el nodo principal del DOM del cual se desprenden todos los siguientes.



Tipos de Nodos

document

Representa el nodo raíz.

attr

Representa el atributo de un elemento.

comment

Nodos de comentario dentro del documento.

element

Representa una etiqueta HTML y puede tener tanto nodos hijos como atributos.

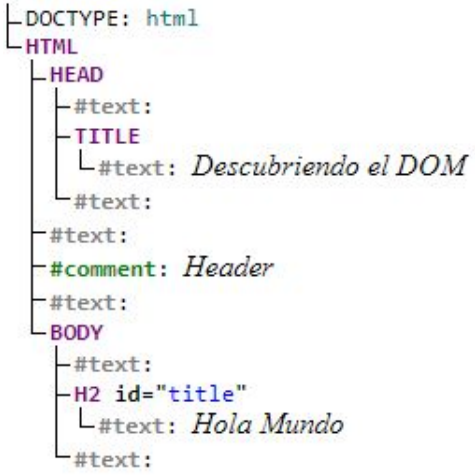
text

Almacena el contenido del texto que se encuentra entre una etiqueta de apertura y una de cierre.

Representación del DOM

En el siguiente ejemplo **podemos observar** como es la representación del DOM para este código HTML.

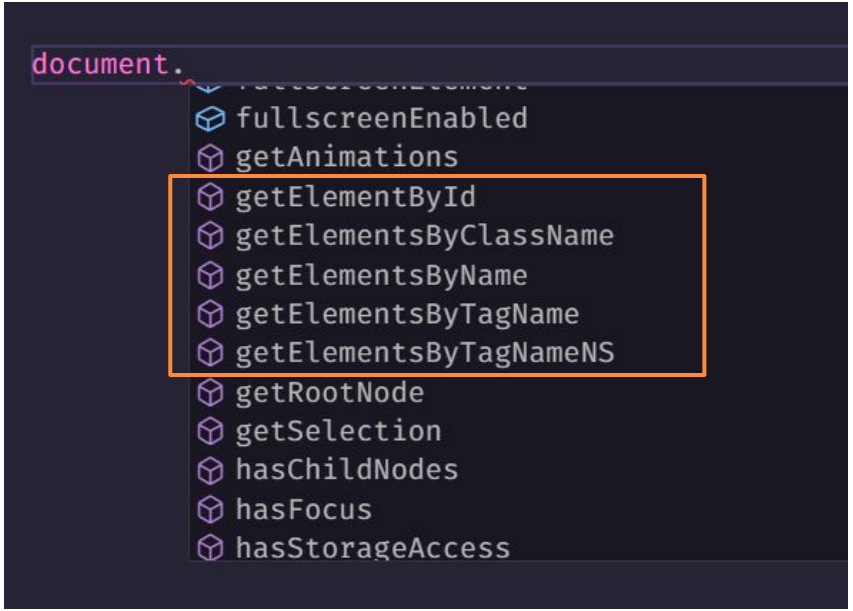
```html
<!DOCTYPE html>
<html>
    <head>
        <title>Descubriendo el DOM</title>
    </head>
    <!-- Header -->
    <body>
        <h2 id="title">Hola Mundo</h2>
    </body>
</html>
```



Acceso al DOM

Javascript nos provee de métodos nativos para **acceder** a los distintos Nodos y sus propiedades.

Lo primero es **invocar** al objeto document para tener acceso a todo nuestro documento.



Acceso al DOM

getElementById('blog')

Trae el nodo del elemento con id #blog.

querySelector('#header nav .link')

Accedemos un nodo como si usaramos un selector CSS.

getElementsByClassName('title')

Selecciona todos los nodos con la clase .title y los devuelve en un array de nodos.

getElementsByTagName('p')

Toma todas las etiquetas 'p' y las devuelve en un array de nodos.

Acceso al DOM

En este caso tomamos el elemento HTML con id='title' y **mostramos** el resultado por la consola del navegador.

```html
<!DOCTYPE html>
<html>
    <head>
        <title>Descubriendo el DOM</title>
    </head>
    <!-- Header -->
    <body>
        <h2 id="title">Hola Mundo</h2>
    </body>

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

```js
const title = document.getElementById('title');

console.log(title);
```

Modificar el DOM

Crear elementos

createElement('div')

Crea una etiqueta div, podemos alojarla en una variable.

createTextNode('Hola mundo')

Crea un nodo de texto con Hola Mundo como contenido.

Agregar elementos

padre.appendChild(hijo)

Agregamos un nodo hijo a un nodo padre.

nodo.innerText = 'Hola';

Inyectamos texto directamente dentro de un nodo o reemplazamos el texto existente.

nodo.innerHTML = ' Hola ';

Inyectamos HTML dentro de un nodo, reemplazando el contenido actual.

document.write = 'texto'

Agrega contenido como hijo directo de document, reemplazando todos los nodos anteriores.

Eventos

Son acciones que suceden en el navegador, que afectan directamente a los **elementos del DOM** y que podemos “escuchar” a través de distintos métodos de **Javascript**.

Events Handlers

Son los **métodos** que existen para reaccionar frente a distintos **eventos** que se produzcan en el navegador.

```
node.addEventListener('eventType', callback);
```

Utilizamos el método **addEventListener** sobre el nodo y pasamos el tipo de evento y una función de callback como parámetros.

```
node.eventType = () => { callback };
```

Accedemos al **método del tipo de evento** del nodo seleccionado y le asignamos una función a ejecutar cuando se realice el evento.

```
<button onclick="action()"></button>
```

Desde el HTML **agregamos un atributo** con el nombre del evento y entre comillas pasamos el **nombre de la función** javascript que debe ejecutarse al producirse el evento.

Si bien está manera puede parecer más cómoda, se desaconseja su uso por considerarse una mala práctica.

Event Types

Si bien la lista de eventos del navegador es enorme, en este caso veremos los más comunes.

`mouse.events`

- click
- mouseover/out
- mousedown/up
- mousemove

`keyboard.events`

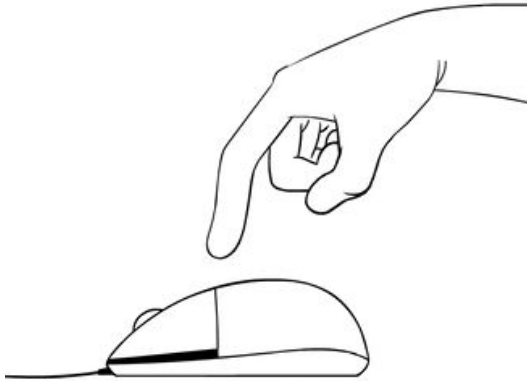
- keydown
- keyup
- keypress

`form.events`

- input
- submit
- blur
- focusout/in
- change

`other.events`

- play
- pause
- load
- scroll



Veamos algunos ejemplos en vivo...

No te olvides de dar el presente

Recordá:

- Revisar la Cartelera de Novedades.
- Hacer tus consultas en el Foro.

Todo en el Aula Virtual.

Gracias