**El Document Object Model (DOM)**

Vamos a explicar los elementos de estructura de HTML con el siguiente ejemplo simple:

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8"/>

<title>Título de la Página</title>

</head>

<body>

<h1 id="principal">Encabezado nivel 1</h1>

<p>Er zijn vele variaties van passages van Lorem Ipsum beschikbaar maar

het merendeel heeft te lijden gehad van wijzigingen in een of andere vorm,

door ingevoegde humor of willekeurig gekozen woorden die nog niet half

geloofwaardig ogen.

</p>

</body>

</html>

Todos los documentos HTML deben empezar con el tipo de documento:

<!DOCTYPE html>

La anterior estructura tiene la etiqueta raíz **html** y está dividida en dos secciones:

* Head: contenedor con todas las etiquetas de configuración de estilos (css), archivos javascript y metadata como el título de la página y la codificación de caracteres utilizada.
* Body: contenedor de todas las etiquetas visuales de la página: títulos, hipervínculos, imágenes, tablas, formularios, etc.

Por cada uno de los elementos del archivo existe una marca html para definirlo, como, por ejemplo: <title>, <header>, <article>, <section>, <p>, <div>, <span>, <img>, <table> y otros más. Cada marca abre y debe cerrar. Para indicar cierre se escribe la marca antecedida de /. En el listado anterior, las dos últimas líneas son la marca de cierre para </body> y luego, una marca de cierre para </html>. El documento estará bien construido si su estructura es arborescente. La representación en árbol del listado anterior es:

| **Figura 1** |
| --- |
| Figura 1 |

Esta estructura arborescente se llama el DOM (Document Object Model) y puede ser manipulado programáticamente, es decir, existe un API para manipular el DOM y cada uno de sus elementos. El DOM es independiente de la plataforma y del lenguaje de programación(hay un API por cada lenguaje en particular hay uno para JavaScript/TypeScript) para manipular el DOM

Los dos conceptos principales del DOM son Document y Element. EL árbol de la figura 1 es un Document y cada nodo es un HTMLElement.

El API DOM nos provee servicios para:

* encontrar un o unos elemento(s) en el documento. Por ejemplo, encontrar el primer elemento que tiene etiqueta h1.
* modificar un elemento. Por ejemplo, cambiar el contenido del primer párrafo (etiqueta p).
* crear nuevos elementos. Por ejemplo, crear un nuevo párrafo (etiqueta p).

La documentación oficial del API [aquí](https://developer.mozilla.org/en-US/docs/Web/API/HTML_DOM_API). En este tutorial vamos a ver algunos servicios básicos.

**Encontrar un elemento en el documento HTML**

Dado un nodo del árbol DOM, a partir de ese punto se puede buscar un/unos elemento/s utilizando las siguientes funciones:

| **Función** | **Descripción** |
| --- | --- |
| getElementById() | Devuelve el elemento que tiene el atributo ID con el valor especificado. |
| getElementsByClassName() | Devuelve una colección HTMLCollection que contiene todos los elementos con el nombre de clase especificado. |
| getElementsByName() | Devuelve una colección HTMLCollection que contiene todos los elementos con el nombre especificado. |
| getElementsByTagName() | Devuelve una colección HTMLCollection que contiene todos los elementos con el nombre de la etiqueta especificada. |

Veamos un ejemplo utilizando nuestro HTML inicial representado en el árbol de la figura 1.

Queremos obtener el elemento que está identificado con el id cuyo valor es "principal". Utilizamos el método getElementById. La variable document, es una variable global que representa todo el documento HTML actual.

El nodo que recuperamos tiene una propiedad llamada innerText que contiene el valor del elemento (el texto que aparece en la página).

Inner text me da el texto que aparece en la paguina de nviel h1

let nodoH1 = document.getElementById("principal");

console.log(nodoH1.innerText); // Encabezado nivel 1

Ahora, podemos usar la propiedad innerText para cambiar el valor y ver en consola el resultado.

nodoH1.innerText = "Hola mundo contaminado";

console.log(nodoH1.innerText); // Hola mundo contaminado

Al desplegar en el navegador tenemos:

| **Figura 2** |
| --- |
| Figura 2 |

En el código anterior la variable nodoH1 no tenía ningún tipo definido. La función getElementById retorna un Element o null si no lo encuentra. Entonces la declaración correcta de la variable sería:

let nodoH1 :HTMLElement | null = document.getElementById("principal");

Sin embargo, en el modo estricto de tipos en TypeScript, si una variable puede ser null, está no puede accederse directamente, a menos que vaya en un condicional sobre null . Es decir, el código nodoH1.innerText va a producir un error. Una forma de resolverlo es agregando un condicional:

var nodoH1 : HTMLElement | null = document.getElementById("principal");

console.log(nodoH1 == null? '': nodoH1.innerText); // Encabezado nivel 1

La solución anterior, además, tiene otro problema y es el tipo de nodeH1 no es el esperado para otras funciones que necesitan HTMLElement. Necesitamos que el tipo de nodeH1 sea HTMLElement pero que pueda tomar el valor de null . Esto se logra con el operado (sufijo) ! al final de la función que retorna el elemento:

var nodoH1 : HTMLElement | null = document.getElementById("principal")!;

console.log(nodoH1.innerText); // Encabezado nivel 1

**Crear un nuevo elemento en el documento HTML**

Existen varios métodos para crear nuevos elementos en un DOM. La idea básica es crear el nuevo elemento con la función createElement o con el texto HTML '

hola

', tener localizado el lugar (el elemento) dónde se va a insertar el nuevo. Este puede ser agregado ya sea de forma adyacente al elemento actual o como hijo del elemento actual. Vemos un ejemplo:

var nuevoParrafo = document.createElement("p"); // creo el nuevo elemento

nuevoParrafo.innerText = "Párrafo nuevo";

// Se va a crear el elemento con respecto al `h1 `del HTML, la posición se define con respecto a ese h1.

nodoH1.insertAdjacentElement('afterend', nuevoParrafo); // agrego un elemento creado

Puedo usar nodoH1 por que ya lo cree anteriormente

nodoH1.insertAdjacentHTML('afterend', '<p id="two">Otro Párrafo</p>') // agrego un elemento con el texto html

El resultado al desplegar el HTML es:

| **Figura 3** |
| --- |
| Figura 3 |

-InnerText

-Operador ¡