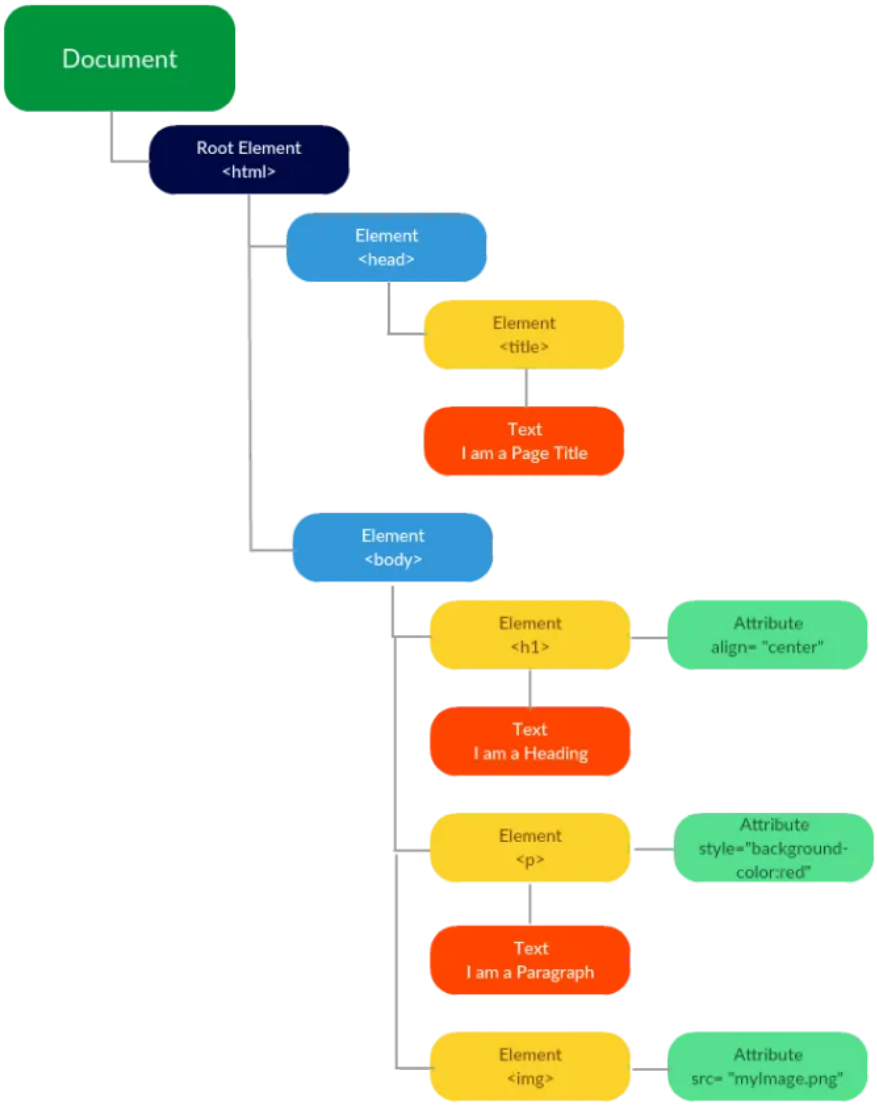
DOM

El **DOM** es un conjunto de utilidades específicamente diseñadas para **manipular documentos** XML, y por tanto documentos **HTML**. El DOM **transforma** el archivo **HTML** en un **árbol de nodos jerárquico**, fácilmente manipulable. Los nodos más importantes son:

- **Document**: Representa el **nodo raíz**. Todo el documento HTML.
- **Element**: Representa el **contenido** definido por un par de **etiquetas** de apertura y cierre, lo que se conoce como un tag HTML. **Puede tener como hijos más nodos de tipo Element y también atributos**.
- **Attr**: Representa el **atributo** de un elemento.
- **Text**: Almacena el **contenido del texto** que se encuentra **entre** una **etiqueta** HTML de **apertura** y **cierre**.



Buscando en el DOM

Para poder **recorrer** el **DOM**, lo podemos hacer a través de un API JavaScript con métodos para acceder y manipular los nodos. Algunas de estas funciones son:

- `getElementById(id)`**:** Devuelve el **elemento** con un **id** especifico.

- **getElementsByName(name)**: Devuelve los **elementos** que un **name** específico.
- **getElementsByTagName(tagname)**: Devuelve los **elementos** con un nombre de **tag** específico.
- **getElementsByClassName(classname)**: Devuelve los **elementos** con un nombre de **clase** específico.
- **getAttribute(attributeName)**: Devuelve el valor del **atributo** con nombre **attributeName**
- `querySelector(selector)` **:** Devuelve un único **elemento** que corresponda con el **selector** , ya sea por **tag, id, o clase**.
- **querySelectorAll(selector):** Devuelve un **array** con los **elementos** que correspondan con la query **introducida en selector**.

```
document.querySelector('.sidebar');
```

Manipulando el DOM

De igual manera podemos **manipular** el **DOM** con las siguientes funciones:

- `createElement(name)` **:** Crea un **elemento** HTML con el **nombre** que le pasemos en el **parámetro** name.
- **createTextNode(text)**: Crea un **nodo** de **texto** que puede ser **añadido** a un **elemento HTML**.
- **createTextAttribute(attribute)**: Crea un **atributo** que puede ser **añadido posteriormente** a un **elemento** HTML.
- `appendChild(node)` **:** Nos permite **hacer hijo un elemento a otro**.
- **insertBefore(new, target)**: Permite **insertar** un **elemento** o nodo new **antes del indicado en target .**
- **removeAttribute(attribute)**: **Elimina** el **atributo** de nombre attribute del nodo desde el que se le llama.
- **removeChild(child): Elimina** el **nodo hijo** que se indica con child
- **replaceChild(new, old)**: **Reemplaza** el nodo old por el que se indica en el parámetro new.

Manejando el DOM: Ejemplo

```
<div id="div1">El texto superior se ha creado dinámicamente.</div>
```

```
window.onload = function() {
  addContent();
}

function addContent () {
  // crea un nuevo div y añade contenido
  var newDiv = document.createElement("div");
  var newContent = document.createTextNode("Hola! ¿Qué tal?");
  // añade texto al div creado.
  newDiv.appendChild(newContent);
  // añade el elemento creado y su contenido al DOM
  var currentDiv = document.getElementById("div1");
  document.body.insertBefore(newDiv, currentDiv);
}
```

Propiedades de los nodos

Todos los nodos tienen algunas propiedades que pueden ser muy útiles para las necesidades de nuestros desarrollos:

- **attributes**: Nos **devuelve** un **objeto** con todos los **atributos** que posee un nodo.
- **className**: Permite **setear o devolver** el nombre de la **clase** (para CSS) que tenga el nodo si la tiene.
- **classList**: Listado de clases del nodo.
- **id**: Igual que className pero para el atributo id.
- **innerHTML**: Devuelve o **permite insertar código HTML** (incluyendo tags y texto) dentro de un nodo.
- **nodeName**: **Devuelve** o nombre del **nodo**, si es un devolverá DIV .
- **nodeValue**: Devuelve el **valor** del **nodo**. Si es de tipo element devolverá null . Pero por ejemplo **si es un nodo de tipo texto, devolverá ese valor**.
- **style**: Permite **insertar** código **CSS** para editar el estilo.
- **tagName**: Devuelve el nombre de la etiqueta HTML correspondiente al nodo. Similar a nodeName, pero solo en nodos de tipo tag HTML.
- **title**: Devuelve o permite **modificar** el valor del **atributo title** de un nodo.

- **childNodes** : Devuelve un array con los nodos hijos del nodo desde el que se llama.
- **firstChild** : Devuelve el **primer hijo**.
- **lastChild** : Devuelve el **último hijo**.
- **previousSibling** : Devuelve el anterior "hermano" o nodo al mismo nivel.
- **nextSibling** : Devuelve el siguiente "hermano" o nodo al mismo nivel.
- **ownerDocument** : Devuelve el nodo raíz donde se encuentra el nodo desde el que se llama.
- **parentNode** : Devuelve el nodo padre del nodo que se llama.

Manejando el DOM: Ejemplo

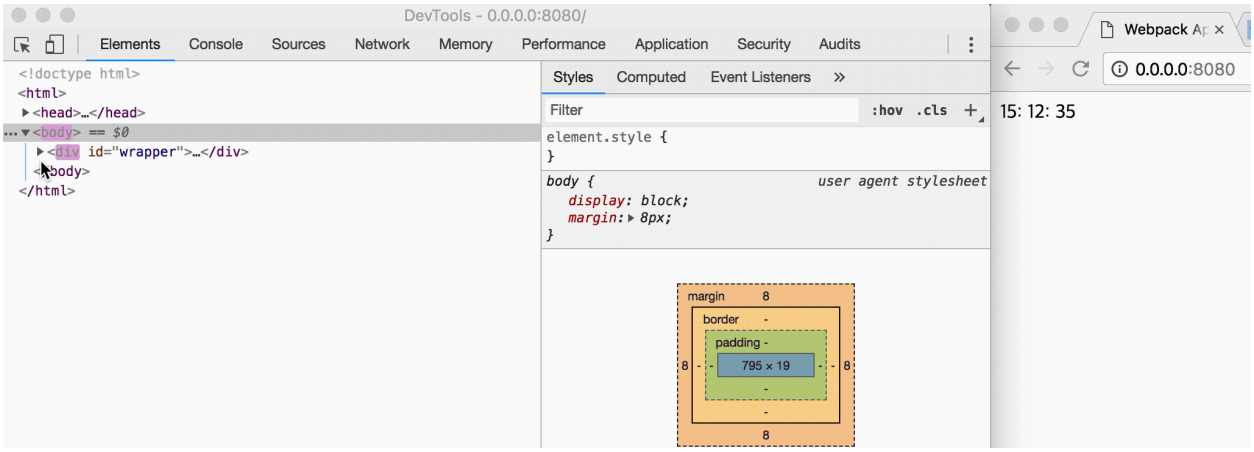
```
function sayHello() {
  console.log('Hello!');
}

function addListeners() {
  document.getElementById("btn").addEventListener("click", sayHello);
}

window.onload = function() {
  // init listeners
  addListeners();

  // init content
  addElement();
}

function addElement() {
  var newElem = document.createElement('div');
  newElem.id = 'nuevoElemento';
  newElem.className = 'bloque';
  newElem.style = 'background:red; width:200px; height:200px';
  var body = document.querySelector('body');
  body.appendChild(newElem);
}
```

Veamos el resultado sobre nuestro HTML:

```
<body>
   <button id="btn">Say hello!</button>
   <div id="nuevoElemento" class="bloque" style="background: red; width: 200px; height: 200px;"></div> </body>
</body>
```

Lo que hemos hecho es **crear** un **elemento** div con un id de nombre **nuevoElemento**, una **clase** bloque y un estilo **CSS** que define un color de fondo red (rojo) y un ancho y alto de 200px.

Todo el API del DOM nos permite cualquier modificación y edición de elementos HTML, de forma que dinámicamente, por ejemplo por medio de eventos, podemos modificar el aspecto y funcionalidad del documento HTML que estamos visualizando.



El DOM es un elemento "vivo" que nos va a permitir interactuar con el navegador (después de que este último haya interpretado nuestros estáticos de HTML/CSS).