UNIDAD 5. Modelo de Objetos del Documento: DOM

Desarrollo Web en Entorno Cliente

2º DAW

Contenidos

Introducción

El **DOM**, *Document Object Model* (Modelo de Objetos del Documento), es una estructura de objetos jerárquica que representa los elementos de una página HTML o de un documento XML.

Esta estructura nos va a permitir acceder a sus nodos, pudiendo modificarlos, añadirlos, eliminarlos, etc. para manipular el contenido del documento, modificar su estructura o su aspecto, dando así dinamismo a la visualización del documento.

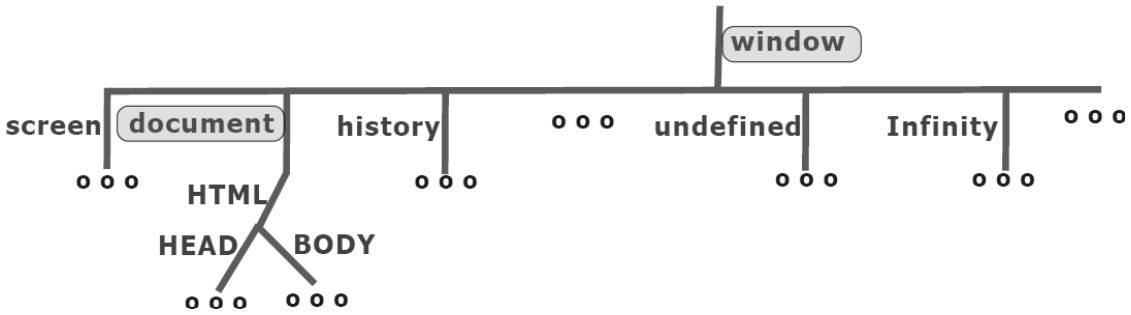
El navegador, en un segundo plano, se encarga de mantener el documento que se visualiza en pantalla y el DOM sincronizados, se modo que cualquier modificación que hagamos en el DOM, se traslade automáticamente al documento que se está visualizando.

Toda esta funcionalidad se encuentra encapsulada en **objetos para cada tipo de nodo** con **propiedades, métodos y eventos**, que pueden ser manipulados por cualquier lenguaje de programación. Uno de los lenguajes más utilizados para manejar el DOM es JavaScript, que es con el que vamos a trabajar nosotros.

De este modo, los objetos DOM en JavaScript, nos van a permitir manipular elementos HTML desde nuestros scripts. Cada objeto DOM tiene una caja visual asociada a cada etiqueta HTML (la misma que manejamos con CSS), por lo que las modificaciones afectarán a la caja visual del elemento HTML asociado.

Realmente hemos estado trabajando con el DOM desde el principio, a través del método `document.getElementById("idX")`, que obtiene el objeto DOM del elemento HTML con atributo id="idX". En esta unidad veremos muchos otros usos posibles del DOM.

Entorno global: `window`, `document` y Web APIs



Veamos cómo ubicar esta estructura de objetos jerárquica dentro de nuestro entorno de ejecución de JavaScript:

- El objeto **`window`** es el entorno global de ejecución de JavaScript en el navegador. Sus propiedades dan acceso a los elementos de página Web, del navegador y de JavaScript.
  - https://javascript.info/browser-environment

- Este entorno global da también acceso a las numerosas **APIs del navegador** actual: DOM, Canvas, Fetch, Storage, Full Screen, Touch Events, Service Workers, WebRTC, WebGL, etc.
    - https://developer.mozilla.org/en-US/docs/Web/API

- **`document`** da acceso a la página HTML con la API DOM (Document Object Model) y se referencia como: `window.document`, `this.document` o `document`. (`this` es una referencia entorno de ejecución y referencia `window` cuando está en el entorno global).
    - https://javascript.info/dom-nodes
    - https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model

Estructura del DOM

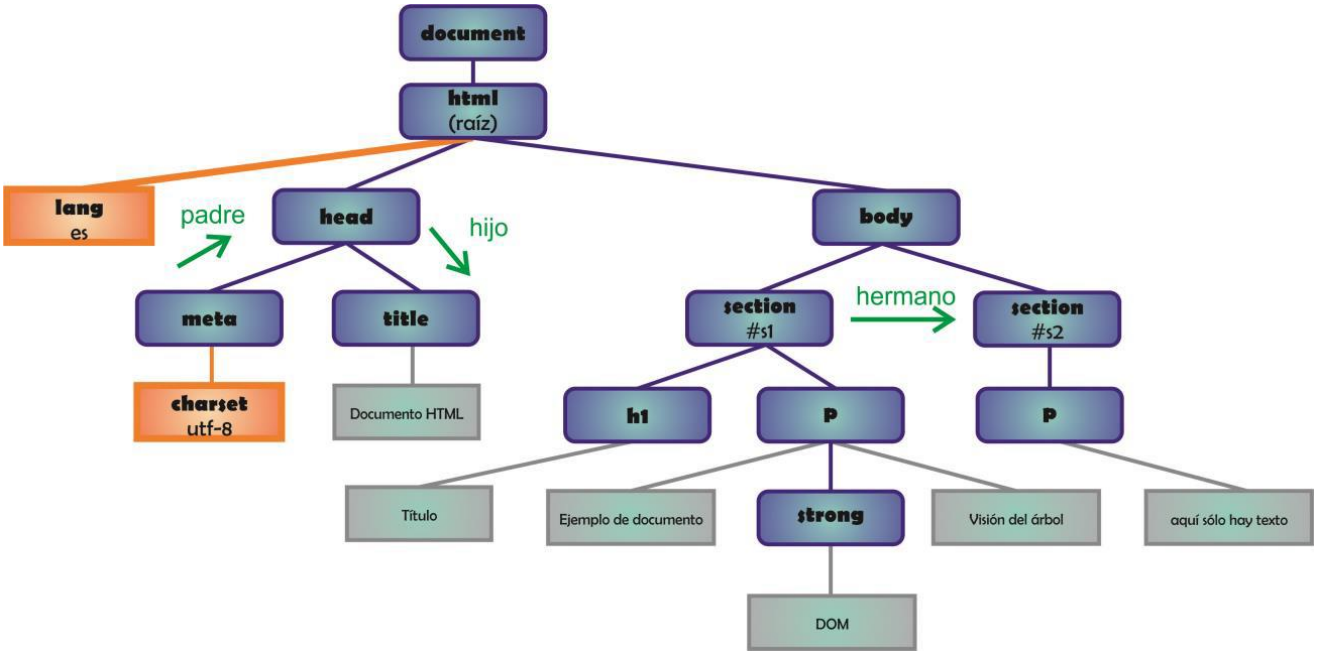
Como hemos dicho anteriormente, el DOM es una estructura de datos jerárquica (un árbol de nodos) que representa una página HTML y cuya raíz es el documento. En el siguiente ejemplo, el documento HTML:

```
<!DOCTYPE html>
<html lang="es">
   <head>
   <meta charset="UTF-8">
   <title>Documento HTML</title>
   </head>
<body>
<section id="s1">
   <h1>Título</h1>
   <p>Ejemplo de documento<strong>DOM</strong>Visión en árbol</p>
</section>
<section id="s2">
   <p>Aquí solo hay texto</p>
</section>
</body>
</html>
```

Es representado por el DOM según el esquema de la siguiente página.

En este esquema del DOM podemos ver que todos los objetos parten de un objeto superior que representa al propio documento y que todo objeto distinto a este tiene un padre (`parent`).

También podemos observar que hay distintos tipos de objetos: elementos HTML, atributos, textos etc. e incluso hay diferentes tipos de objetos elementos HTML: secciones, encabezados, párrafos, etc. Esto nos va a permitir manipular cada parte de una página manejando el objeto que la representa.



Tipos de nodos: introducción

La especificación completa de DOM define 12 tipos de nodos, aunque las páginas HTML habituales se pueden manipular manejando solamente cuatro o cinco tipos de nodos:

- **Document**: nodo raíz del que derivan todos los demás nodos del árbol.
- **Element**: representa cada una de las etiquetas HTML. Se trata del único nodo que puede contener atributos y el único del que pueden derivar otros nodos.
- **Attr**: se define un nodo de este tipo para representar cada uno de los atributos de las etiquetas HTML, es decir, uno por cada par atributo=valor.
- **Text**: nodo que contiene el texto encerrado por una etiqueta HTML.
- **Comment**: representa los comentarios incluidos en la página HTML.

Los otros tipos de nodos existentes que no se van a considerar son DocumentType, CDataSection, DocumentFragment, Entity, EntityReference, ProcessingInstruction y Notation.

De esta forma, del nodo raíz **`document`** 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 nodo correspondiente a su etiqueta padre.

La transformación de las etiquetas HTML habituales generan dos nodos: el primero es el nodo de tipo **`Element`** (correspondiente a la propia etiqueta HTML) y el segundo es un nodo de tipo **`Text`**, que contiene el texto encerrado por esa etiqueta HTML.

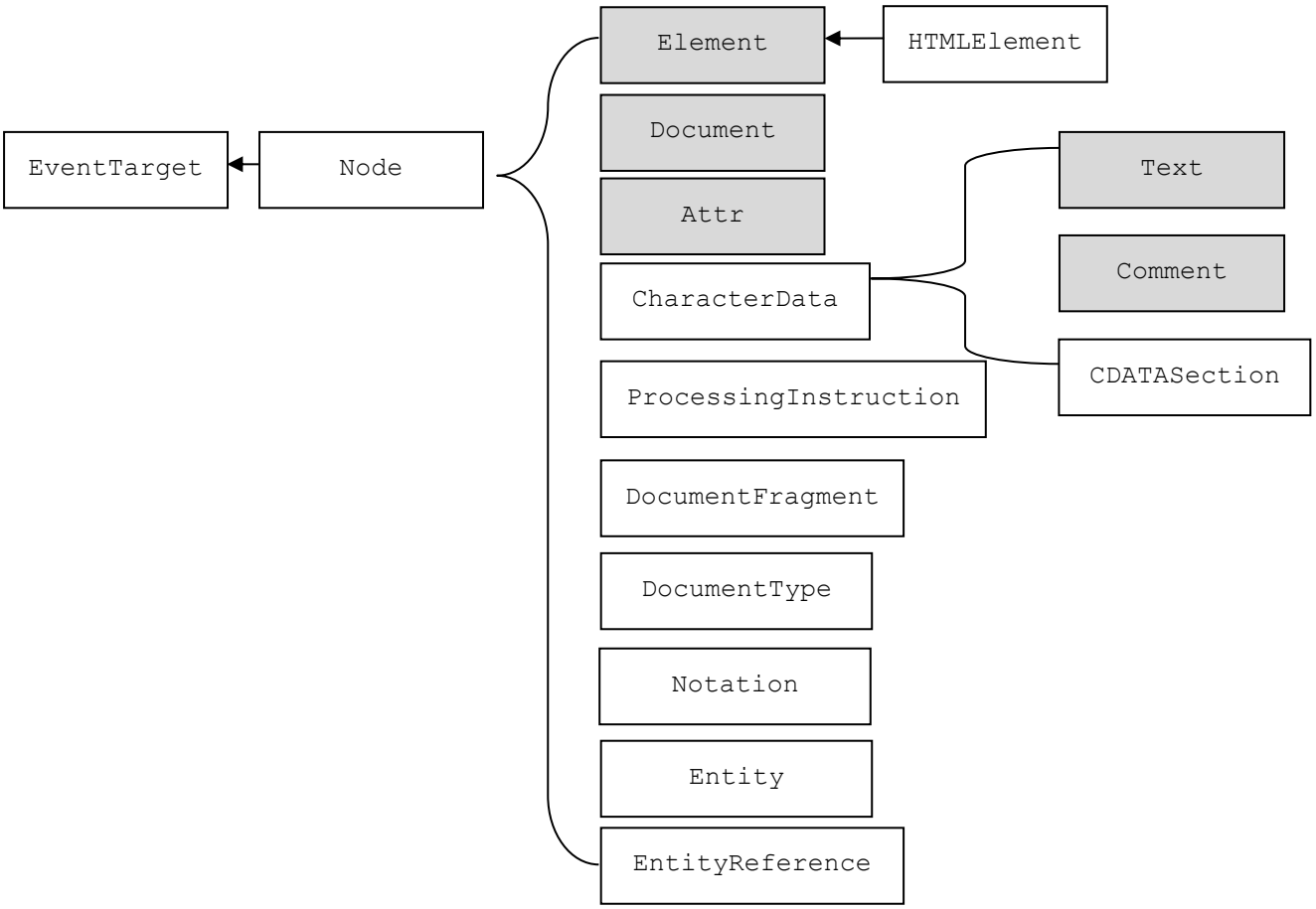
Ejemplo:

```
<title>Página sencilla</title>
```

Genera un nodo **`Element`** `title` y un nodo **`Text`** `Página sencilla`.

Tipos de nodos: `Node`

La interfaz `Node`, nos proporciona propiedades y métodos comunes a todos los nodos, pero, como vemos en el siguiente esquema, **no hay ningún nodo que sea directamente de tipo `Node`**, por lo que, los nodos con los que vamos a trabajar heredan los métodos y propiedades de `Node` (tenemos sombreados en gris los que vamos a manejar para páginas HTML):



Propiedades de `Node`

Algunas propiedades de `Node`:

Información	Descendientes	Hermanos	Ascendientes
baseURI	chidNodes	nextSibling	parentNode
nodeName	firstChild	previousSibling	parentElement
nodeType	lastChild		ownerDocument
nodeValue	textContent		

Todas las propiedades son de **solo lectura** excepto `nodeValue y textContent.`

Propiedad	Contenido
`Node.baseURI`	Devuelve una cadena que representa el URL base: protocolo, nombre de dominio, estructura de directorio, hasta el último /.

<table>
<tr><th>Propiedad</th><th>Contenido</th></tr>
<tr><td>Node.nodeName</td><td>Devuelve una cadena con el nombre del nodo. El contenido de este atributo depende del tipo de nodo.
Por ejemplo: en un <code>Element</code>, contendrá el nombre de la etiqueta (ej. <code>"img"</code>); en un <code>Text</code>, contendrá la cadena <code>"#text";</code> en un nodo <code>Document</code>, contendrá la cadena <code>"#document".</code></td></tr>
<tr><td>Node.nodeType</td><td>Devuelve un número entero sin signo que representa el tipo de nodo. Posibles valores son:
<table>
<tr><th>Nombre</th><th>Valor</th></tr>
<tr><td>ELEMENT_NODE</td><td>1</td></tr>
<tr><td>ATTRIBUTE_NODE</td><td>2</td></tr>
<tr><td>TEXT_NODE</td><td>3</td></tr>
<tr><td>CDATA_SECTION_NODE</td><td>4</td></tr>
<tr><td>ENTITY_REFERENCE_NODE</td><td>5</td></tr>
<tr><td>ENTITY_NODE</td><td>6</td></tr>
<tr><td>PROCESSING_INSTRUCTION_NODE</td><td>7</td></tr>
<tr><td>COMMENT_NODE</td><td>8</td></tr>
<tr><td>DOCUMENT_NODE</td><td>9</td></tr>
<tr><td>DOCUMENT_TYPE_NODE</td><td>10</td></tr>
<tr><td>DOCUMENT_FRAGMENT_NODE</td><td>11</td></tr>
<tr><td>NOTATION_NODE</td><td>12</td></tr>
</table></td></tr>
<tr><td>Node.nodeValue</td><td>Devuelve/establece el valor del nodo actual. El valor del nodo depende del tipo de datos. En el caso de los nodos de tipo <code>Text</code>, contiene el texto del nodo.</td></tr>
<tr><td>Node.childNodes</td><td>Devuelve una estructura “viva” <code>NodeList</code>, que contiene todos los hijos de este nodo. Una estructura “viva” quiere decir que, si cambian los hijos del nodo, la estructura se actualiza automáticamente.</td></tr>
<tr><td>Node.firstChild</td><td>Devuelve un nodo que representa el primer hijo directo del nodo, o <code>null</code> si el nodo no tiene hijos.</td></tr>
<tr><td>Node.lastChild</td><td>Devuelve un nodo que representa el último hijo directo del nodo, o <code>null</code> si el nodo no tiene hijos</td></tr>
<tr><td>Node.textContent</td><td>Devuelve/establece el contenido textual de un elemento y de todos sus descendientes.</td></tr>
<tr><td>Node.nextSibling</td><td>Devuelve un nodo que representa el siguiente hermano del nodo, o <code>null</code> si no tiene.</td></tr>
<tr><td>Node.previousSibling</td><td>Devuelve un nodo que representa el nodo hermano anterior en el árbol, o <code>null</code> si no hay ninguno.</td></tr>
<tr><td>Node.parentNode</td><td>Devuelve el nodo padre del nodo. Si no tiene padre (ej. es el nodo raíz), devuelve <code>null</code>.</td></tr>
</table>

Propiedad	Contenido
Node.parentElement	Devuelve un nodo de tipo `Element` que es el padre del nodo. Si el nodo no tiene padre, o no es de tipo `Element`, devuelve `null`.
Node.ownerDocument	Devuelve el nodo `Document` al que pertenece este nodo. Si fuera el propio nodo `Document`, devuelve `null`.

Métodos de `Node`

Algunos métodos de `Node`:

Información	Añadir hijos	Eliminar hijos	Reemplazar hijos	Clonar
contains	appendChild	removeChild	replaceChild	cloneNode
hasChildNodes	insertBefore			

Método	Devuelve
Node.contains(*otroNodo*)	Devuelve `true` si `otroNodo` es descendiente del nodo dado, `false` en caso contrario.
Node.hasChildNodes()	Devuelve `true` si un nodo tiene nodos hijos, `false` en caso contrario.
Node.appendChild(*otroNodo*)	Añade `otroNodo` como último hijo del nodo actual. Si el argumento es un nodo que forma parte del árbol del DOM, ese nodo se moverá de posición.
Node.insertBefore(*otroNodo, nodoReferencia*)	Inserta `otroNodo` como nodo hijo de `Node`, delante de `nodoReferencia`, hijo que se pasa como parámetro. Si no se pasa este último nodo, se inserta como último hijo.
Node.removeChild(*otroNodo*)	Elimina el nodo hijo del nodo actual `otroNodo`.
Node.replaceChild(*unNodo, otroNodo*)	Reemplaza `unNodo` hijo por `otroNodo` hijo.
Node.cloneNode()	Clona un nodo y opcionalmente, su contenido (solo si pasamos como argumento `true`).

Ejemplo: eliminar un elemento

```
<body>
    <p id="provisional">Párrafo provisional</p>...
</body>
...
let parrafo = document.getElementById("provisional");
parrafo.parentNode.removeChild(parrafo);
//Otra opción con un método de Document:
//parrafo.remove(parrafo);
```

Tipos de nodos: `ParentNode`

La interfaz `ParentNode` contiene unas propiedades que son específicas de los objetos `Node` que pueden tener hijos. No podemos crear ningún objeto de este tipo, accederemos a sus propiedades a través de los tipos de nodos que las heredan `Element` y `Document`.

Propiedades de `ParentNode`:

Total hijos	Nodos hijos	Nodos que representan una colección HTMLCollection
childElementCount	firstElementChild	children
	lastElementChild	

Todas las **propiedades** son de **solo lectura**.

Propiedad	Contenido
ParentElement.childElementCount	Devuelve el número de hijos de este nodo que son elementos.
ParentElement.children	Devuelve una `HTMLCollection` (similar a un array) "viva" de los objetos de tipo `Element` que son hijos de este nodo.
ParentElement.firstElementChild	Devuelve el primer nodo de tipo `Element` que es hijo de este nodo. Devuelve `null` si no hay ninguno.
ParentElement.lastElementChild	Devuelve el último nodo de tipo `Element` que es hijo de este nodo. Devuelve `null` si no hay ninguno.

Tipos de nodos: `Document`

Es el nodo raíz del árbol del DOM, representa el documento que se está visualizando en el navegador y de él "cuelgan" todos los nodos asociados a cada elemento del documento.

Propiedades de `Document`

Algunas propiedades de `Document`:

Nodos hijos	Propiedades	Nodos que representan una colección HTMLCollection	
body	characterSet	forms	embeds
head		images	scripts
documentElement		links	

Todas las propiedades son de **solo lectura** excepto `body` y `head`.

Propiedad	Contenido
Document.body	Devuelve el nodo `<body>` o `<frameset>` del documento actual.

`Document.characterSet`	Devuelve una cadena que representa el juego de caracteres utilizado por el documento. Ej. `"UTF-8"`
`Document.documentElement`	Devuelve un nodo de tipo `Element`, que es hijo directo del documento. En los documentos HTML, normalmente es el objeto que representa al elemento `<html>`.
`Document.embeds`	Devuelve una `HTMLCollection` (similar a un array) de los elementos `<embed>` que contiene el documento.
`Document.forms`	Devuelve una `HTMLCollection` (similar a un array) de los elementos `<form>` que contiene el documento.
`Document.head`	Devuelve el nodo `<head>`.
`Document.images`	Devuelve una `HTMLCollection` (similar a un array) de los elementos `<img>` que contiene el documento.
`Document.links`	Devuelve una `HTMLCollection` (similar a un array) de los elementos `<a>` y `<area>` que contiene el documento (solo los que tengan un valor para la propiedad `href`).
`Document.scripts`	Devuelve una `HTMLCollection` (similar a un array) de los elementos `<script>` que contiene el documento.

HTMLCollection

Algunos las propiedades de los nodos son del tipo HTMLCollection (ej. forms).
Una HTMLColection representa una colección genérica (similar a un array) de elementos (en el orden en que aparecen en el documento). Además, ofrece métodos y propiedades para acceder a ellos.

Propiedad	Contenido
`HTMLCollection.lenght`	Devuelve el número de ítems de la colección (solo lectura).

Método	Devuelve
`HTMLCollection.item(i)`	Devuelve el nodo que tiene ese índice en la lista, `null` si está vacía.

Ejemplo: acceder a los elementos de HTMLCollection
Se puede acceder a los elementos a través de un índice, como si fuera un array, con el nombre del elemento (propiedad `name`), si lo tiene, o a través del método `item()`.

```
<form name="formulario">
...
let miFormulario = document.forms[0];
let miFormulario = document.forms["formulario"]
let miFormulario = document.forms.item(0);
```

Métodos de `Document`: acceder a elementos hijos

Método	Devuelve
`Document.getElementById(id)`	Devuelve una referencia al objeto que tiene el `id` pasado como parámetro.

Document.getElementsByName(*name*)	Devuelve todos los nodos de tipo `Element` del documento que tienen una propiedad `name` con el valor que se pasa como parámetro.
Document.getElementsByTagName (*etiqueta*)	Devuelve todos los nodos de tipo `Element` del documento con la etiqueta que se pasa como parámetro.
Document.querySelector(*selectorCSS*)	Devuelve el primer nodo de tipo `Element` del documento que es seleccionado por el selector CSS que se pasa como parámetro.
Document.querySelectorAll(*selectorCSS*)	Devuelve todos los nodos de tipo `Element` del documento que son seleccionados por el selector CSS que se pasa como parámetro.

Ejemplo: acceder a todos los párrafos de un documento y mostrar su texto

```
<body>
   <p>Párrafo DOM primero</p>
   <p class="rojo">Párrafo DOM segundo</p>
...
</body>
...
let párrafos = document.getElementsByTagName("p");
for (i = 0; i < parrafos.length; i++) {
   console.log(parrafos[i].innerHTML);
}
```

Métodos de `Document`: crear elementos

Método	Devuelve
Document.createElement(*element*)	Crea un elemento del tipo especificado por el parámetro. Devuelve el nuevo elemento.
Document.createTextNode(*cadena*)	Crea un nodo de tipo texto. Devuelve un puntero a ese elemento.

Ejemplo: crear un nuevo párrafo e insertarlo después de los otros párrafos

```
<body>
   <p>Párrafo DOM primero</p>
   <p class="rojo">Párrafo DOM segundo</p>
   <p class="rojo">Párrafo DOM tercero</p>
...
</body>
...
let elementoP = document.createElement("p");
let texto = "Párrafo DOM cuarto";
let nodoTexto = document.createTextNode(texto);
elementoP.appendChild(nodoTexto);
document.body.insertBefore(elementoP, document.scripts[0]);
```

Tipos de nodos: Element

Es el tipo de nodo que manejamos más habitualmente del DOM.

Propiedades de `Element`

Propiedad	Contenido
`Element.className`	Devuelve/establece la/s clase/s de `Element`. En caso de ser varias clases, se escriben separadas por comas.
`Element.id`	Cadena que contiene el `id` de `Element`.
`Element.innerHTML`	Cadena que contiene el contenido HTML de `Element`.
`Element.outerHTML`	Cadena que contiene el elemento HTML y su contenido.
`Element.name`	Devuelve/establece el valor de la propiedad `name` de un elemento. Solo se puede aplicar a algunos elementos (`<a>`, `<applet>`, `<button>`, `<form>`, `<frame>`, `<iframe>`, `<img>`, `<input>`, `<map>`, `<meta>`, `<object>`, `<param>`, `<select>`, y `<textarea>`)
`Element.tagName`	Cadena que contiene la etiqueta HTML del elemento dado (solo lectura).
`Element.style`	Devuelve un objeto que tiene como propiedades los atributos de estilo del elemento transformadas a camelCase.

Ejemplo: acceder a todos los elementos de una clase y cambiarles el color del fondo

```
...
.rojo {
    color: red;
}
</head>
<body>
    <p>Párrafo DOM primero</p>
    <p class="rojo">Párrafo DOM segundo</p>
    <p class="rojo">Párrafo DOM tercero</p>
...
</body>
...
let elementosRojos = document.getElementsByClassName("rojo");
for (i = 0; i < elementosRojos.length; i++) {
    elementosRojos[i].style.backgroundColor = "green";
}
```

Métodos de `Element`

Eventos	Buscar nodos en el árbol	Atributos	Eliminar nodo
addEventListener	closest	getAttribute	remove
removeEventListener	getElementsByClassName	getAttributeNames	
	getElementsByTagName	setAttribute	
	querySelector	removeAttribute	
	querySelectorAll	hasAttribute	
		hasAttributes	

Método	Devuelve
`Element.addEventListener()`	Asocia un manejador de eventos a un tipo de evento sobre el elemento.
`Element.removeEventListener()`	Elimina un manejador de eventos del nodo actual.
`Element.closest(`*`selectorCSS`*`)`	Devuelve el elemento ascendiente más cercano del elemento actual (o el propio elemento) que coincida con el selector CSS que se pasa como parámetro.
`Element.getElementsByClassName(`*`clases`*`)`	Devuelve un `HTMLCollection` "vivo" que contiene todos los descendientes del elemento actual que poseen la lista de clases que se pasa como parámetro.
`Element.getElementsByTagName(`*`etiqueta`*`)`	Similar a `Document.getElementsByTagName()`, pero busca solo entre los descendientes del elemento actual.
`Element.querySelector(`*`selectorCSS`*`)`	Similar a `Document.querySelector()`.
`Element.querySelectorAll(`*`selectorCSS`*`)`	Similar a `Document.querySelectorAll()`.
`Element.getAttribute(`*`atrib`*`)`	Devuelve el valor del atributo cuyo nombre se pasa como parámetro.
`Element.getAttributeNames()`	Devuelve un array de cadenas con los nombres de los atributos del elemento actual, si no tuviera atributos, devuelve el array vacío.
`Element.setAttribute(atrib,valor)`	Establece el valor de un atributo del nodo actual.
`Element.removeAttribute(`*`atrib`*`)`	Elimina un atributo del elemento actual.
`Element.hasAttribute(`*`atrib`*`)`	Devuelve `true` si `Element` tiene el atributo especificado, `false` en caso contrario.
`Element.hasAttributes()`	Devuelve `true` si `Element` tiene atributos, `false` en caso contrario.
`Element.remove()`	Elimina `Element` del árbol DOM.

Ejemplo: acceder al primer párrafo y asignarle la clase rojo

```
<head>
...
    <style>
```

```
        .rojo {
            color: red;
        }
    </style>
</head>
<body>
    <p>Párrafo DOM primero</p>
    <p class="rojo">Párrafo DOM segundo</p>
    <p class="rojo">Párrafo DOM tercero</p>
...
</body>
...
let primerParrafo = document.firstElementChild;
primerParrafo.setAttribute("class", "rojo")
```

Tipos de nodos: `HTMLElement`

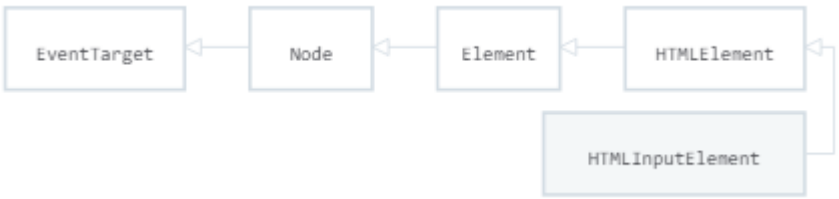
Representan cualquier elemento HTML.

Métodos de `HTMLElelement`

Método	Devuelve
`HTMLElement.blur()`	Quita el foco del teclado del elemento.
`HTMLElement.click()`	Envía un evento `click` al elemento.
`HTMLElement.focus()`	Envía el foco del teclado al elemento.

Tipos de nodos: `HTMLInputElement`

Representa cualquier elemento `input`.



Propiedades de `HTMLInputElelement`

Propiedades que se aplican cualquier input que no esté oculto

Propiedad	Contiene
HTMLInputElelement.name	Devuelve/establece la cadena valor del atributo `name`.
HTMLInputElelement.type	Devuelve/establece la cadena valor del atributo `type` del elemento.
HTMLInputElelement.disabled	Devuelve/establece el atributo booleano `disabled`.
HTMLInputElelement.autofocus	Devuelve/establece el atributo booleano `autofocus`, que especifica si un campo tiene el foco cuando la página se carga.
HTMLInputElelement.required	Devuelve/establece el atributo booleano `required`.
HTMLInputElelement.value	Devuelve/establece la cadena valor del atributo `value` del input.

Propiedades que se aplican solo a inputs con `type= "checkbox"` o `"radio"`

Propiedad	Contiene
HTMLInputElelement.checked	Devuelve/establece el atributo booleano que indica el estado actual del elemento cuando `type` es `checkbox` o `radio`.
HTMLInputElelement.defaultChecked	Devuelve/establece el atributo booleano que indica el estado por defecto de un `checkbox` o `radio`.

Propiedades que se aplican solo a inputs con `type= "image"`

Propiedad	Contiene
HTMLInputElelement.alt	Devuelve/establece la cadena valor del atributo `alt`.
HTMLInputElelement.height	Devuelve/establece la cadena valor del atributo `height`.
HTMLInputElelement.src	Devuelve/establece la cadena valor del atributo `src`.
HTMLInputElelement.width	Devuelve/establece la cadena valor del atributo `width`.

Propiedades que se aplican a inputs con con `type= "text"` o `"number"`

Propiedad	Contiene
HTMLInputElelement.autocomplete	Devuelve/establece la cadena valor del atributo `autocomplete`. Posibles valores son `"on"` y `"off"`. Este atributo se ignora si el atributo `type` es `hidden`, `checkbox`, `radio`, `file`, `button`, `submit`, `reset`, `image`.
HTMLInputElelement.maxLength	Devuelve/establece el número valor del atributo maxlength.
HTMLInputElelement.size	Devuelve/establece el número valor del atributo `size` del elemento. Se aplica solo cuando `type` es `text`, `search`, `tel`, `url`, `email` o `password`.
HTMLInputElelement.pattern	Devuelve/establece la cadena valor del atributo `pattern`. Se aplica solo cuando `type` es `text`, `search`, `tel`, `url`, `email` o `password`.

HTMLInputElelement.placeholder	Devuelve/establece la cadena valor del atributo `placeholder`. Se aplica solo cuando `type` es `text`, `search`, `tel`, `url`, `email` o `password`.
HTMLInputElelement.readOnly	Devuelve/establece el atributo booleano `readonly`. Este atributo se ignora si el atributo `type` es `hidden`, `range`, `color`, `checkbox`, `radio`, `file`, `button`, `submit`, `reset`, `image`.
HTMLInputElelement.min	Devuelve/establece la cadena valor del atributo `min`.
HTMLInputElelement.max	Devuelve/establece la cadena valor del atributo `max`.