

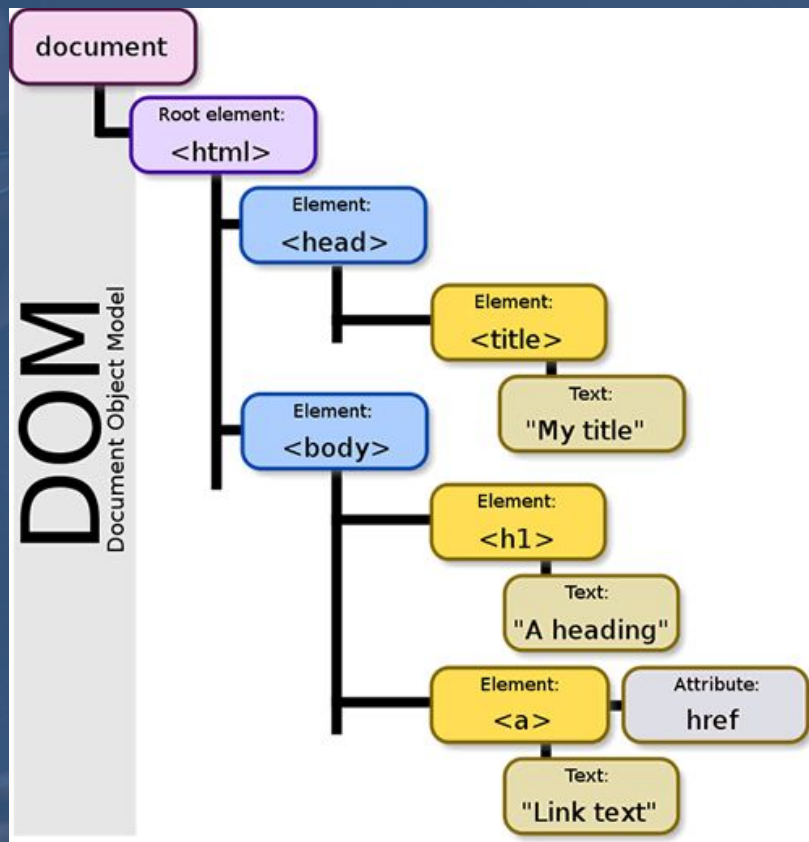


JavaScript DOM

¿Qué es el DOM?

- El Modelo de Objetos de Documento es una plataforma y una interfaz de lenguaje neutral que permite que los programas y scripts accedan y actualicen dinámicamente el contenido, la estructura y el estilo de un documento.
- El HTML DOM es un modelo de objeto estándar y una interfaz de programación para HTML.
- Es un estándar sobre cómo obtener, cambiar, agregar o eliminar elementos HTML.





Árbol DOM

Tipo	Descripción
Elemento	Representa un elemento (div, span)
Atributo	Representa un atributo (id, class, src)
Texto	Representa el contenido textual de un elemento
Comentario	Representa un comentario (<!-- ... !>)
Documento	Representa el documento entero (el nodo raíz del árbol del DOM)
DocumentType	Representa el tipo del documento (<!DOCTYPE html>)

Tipos de Nodos

La interfaz de programación DOM

- En el DOM, todos los elementos HTML se definen como objetos.
- La interfaz de programación son las propiedades y métodos de cada objeto.

```
...  
<p id="demo"></p>  
...  
<script type="text/javascript">  
  let parrafo = document.getElementById("demo");  
  parrafo.innerHTML = "¡Hola Mundo!";  
</script>
```

Ejemplo

Propiedad	Descripción
document.anchors	Returns all <a> elements that have a name attribute
document.body	Returns the <body> element
document.cookie	Returns the document's cookie
document.forms	Returns all <form> elements
document.head	Returns the <head> element
document.images	Returns all elements
document.scripts	Returns all <script> elements
document.title	Returns the <title> element
document.URL	Returns the complete URL of the document

Algunas propiedades de document

Recuperando Elementos

- Document posee funciones para acceder a sus elementos por id, clase, etiqueta y selector.

```
...
<p id="demo" class="cont"></p>
<div id="division" class="cont"></div>
...
<script type="text/javascript">
  let unParrafo = document.getElementById("demo");
  let misConts = document.getElementsByClassName("cont");
  let misDivs = document.getElementsByTagName("div");
  let misRad = document.getElementsByName("opcion");
  let demoEl = document.querySelector("#demo");
  let divsCont = document.querySelectorAll("div.cont");
</script>
```

Objeto Documento

Lista de Nodos

***Salvo getElementById
y querySelector, el
resto devuelven
colecciones**

Modificando Elementos

- Con **innerHTML** se puede modificar el contenido de un elemento HTML.

```
...  
<p id="demo" class="cont"></p>  
...  
<script type="text/javascript">  
  let unParrafo = document.getElementById("demo");  
  unParrafo.innerHTML = "Hola Mundo";  
</script>
```

innerHTML



Modificando Atributos

- Similar al ejemplo anterior, se puede cambiar cualquier atributo de un elemento.

```
...  
<img id="demo" class="cont"/>  
...  
<script type="text/javascript">  
  let unaFoto = document.getElementById("demo");  
  unaFoto.src = "foto.jpg";  
</script>
```

*atributos cuyo nombre tienen “-” (*kebab-case*), pasa a camel-case. Por ejemplo “z-index” a `zIndex`.

Modificando Estilo

- El HTML DOM permite que JavaScript cambie el estilo de los elementos HTML.

```
...  
<p id="demo" class="cont"></p>  
...  
<script type="text/javascript">  
  let p = document.getElementById("demo");  
  p.style.color = "red";  
  p.style.border = "1px solid black";  
</script>
```

Cambiando CSS

classList

className

Cambiar display

Ejemplo con visibility

Validación de Formularios

- La validación de formularios HTML se puede realizar mediante JavaScript.
- Se puede acceder a los elementos a través del formulario o directamente a los elementos.

```
...  
<form id="demo" name="demo">  
  <label>Nº</label> <input id="nro" name="nro" />  
</form>
```

```
...  
<script type="text/javascript">  
  let frm = document.demo;  
  let frmBis = document.forms["demo"];  
  let elemNro = frm["nro"]; //también frm.nro  
</script>
```

Validación de Forms



Valor de los Inputs

- En los inputs comunes, con **value** accedemos a su valor, como cualquier otro atributo.
- En inputs tipo button, reset o submit establece el texto a mostrar.
- No se puede usar en inputs tipo file.
- Sirve también para textarea y select.
- Para "checkbox" y "radio", sirve para el valor asociado, pero **checked** dice si está chequeado o no.

input value

select value

textarea value

checkbox checked

radio checked

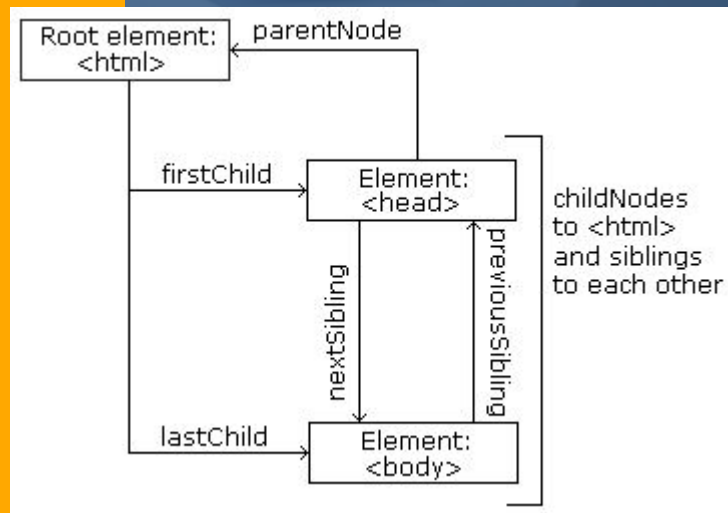
Propiedad	Descripción	Doc
document.createElement(element)	Crea un elemento HTML	<u>ver</u>
elemento.removeChild(hijo)	Quita un elemento HTML hijo	<u>ver</u>
elemento.remove()	Quita ese elemento	<u>ver</u>
elemento.appendChild(hijo)	Añade un elemento como hijo al final	<u>ver</u>
elemento.replaceChild(nvo, vjo)	Reemplaza un elemento hijo	<u>ver</u>
padre.insertAdjacentElement("pos",elem)	Inserta un elemento en una posición relativa a otro	<u>ver</u>
padre.insertBefore(elem, nvo)	Inserta un nuevo elemento antes que otro de referencia	<u>ver</u>

Manipulando el DOM

Navegación entre Elementos

- Todo el árbol puede ser recorrido usando las relaciones de los elementos.
- **parentNode** representa el nodo padre.
- **childNodes** es la colección de hijos.
- **firstChild** y **lastChild** son el 1º y último hijo respectivamente.
- **nextSibling** y **previousSibling** son los hermanos siguiente y anterior.

Navegación



Manejadores de Eventos

- JavaScript permite reaccionar a eventos.
- Puede hacerse directamente en el HTML, con el atributo on*event* donde “event” sería el evento deseado.

```
...  
<input id="demo" onclick="saludar()" />  
...  
<script type="text/javascript">  
  function saludar() {  
    let txt = document.querySelector("#demo").value;  
    alert("¡Hola " + txt + "!");  
  }  
</script>
```

Eventos

Eventos HTML

addEventListener

- Esta función adjunta un controlador de eventos al elemento especificado sin pisar otros.

```
...  
<input id="demo" />  
...  
<script type="text/javascript">  
  let elem = document.querySelector("#demo");  
  elem.addEventListener("click", saludar);  
  
  function saludar() {  
    let txt = document.querySelector("#demo").value;  
    alert(";Hola " + txt + "!");  
  }  
</script>
```


[Event Listener](#)

[removeEventListener](#)

Temporizadores

- Existen dos funciones clave para eventos temporales.
- `setInterval` se usa para intervalos que se repiten.
- `setTimeout` se usa para eventos de una sola vez.
- Ambas funciones devuelven ID que permite detener el evento.
- `clearInterval` y `clearTimeout` detienen el evento asociado al ID especificado.

Temporizadores



A man with glasses is looking down at a smartphone in his hands. He is wearing a dark t-shirt and a watch. The background is a brick wall. The entire image is overlaid with a semi-transparent blue filter.

¡Gracias!

¿Preguntas?

N
ACTI