

# UTILIZACIÓN DE DOM EN JAVASCRIPT

**UT6.- EL MODELO DOM (Modelo de  
Objetos del Documento)**



Susana López Luengo

# El modelo de objetos del documento (DOM)

- Es un estándar de W3C que define cómo acceder a los documentos, como por ejemplo HTML y XML.
  - Es una interfaz de programación de aplicaciones (API) de la plataforma de W3C.
  - Permite a los scripts acceder y actualizar dinámicamente su contenido, estructura y estilo de documento.
  - <https://www.w3.org/2005/03/DOM3Core-es/introduccion.html>
-

# El modelo de objetos del documento (DOM)

- Fue utilizado por primera vez con el navegador Netscape Navigator V.2.0.
  - A esta primera versión de DOM se le denomina DOM nivel 0.
  - El primer navegador de Microsoft que utilizó el DOM nivel 0 fue IE 3.0.
-

# El modelo de objetos del documento (DOM)

- Debido a las diferencias entre los navegadores, W3C emitió la especificación DOM nivel 1 a finales de 1998
- En DOM nivel 1 ya se consideraba la manipulación de todos los elementos existentes en los archivos HTML
- A finales del año 2000, W3C emitió DOM nivel 2, en la cual se incluía el manejo de eventos en el navegador y la interacción con hojas de estilo CSS.
- En 2004 se emitió DOM nivel 3, en la cual se utiliza la definición de tipos de documento (DTD) y la validación de documentos.

<https://www.w3.org/2005/03/DOM3Core-es/introduccion.html>

---

# El modelo de objetos del documento (DOM)

- Actualmente DOM se divide en tres partes según la W3C:
    - **Core DOM** - modelo estándar para todos los tipos de documentos.
    - **XML DOM**: modelo estándar para documentos XML.
    - **HTML DOM** - modelo estándar para documentos HTML.
-

# HTML DOM

- El HTML DOM es un modelo de datos estándar y una interfaz de programación para HTML. En este se definen:
    - Los elementos HTML como objetos.
    - Las propiedades de todos los elementos HTML.
    - Los métodos para acceder a todos los elementos HTML.
    - Los eventos de todos los elementos HTML.
  - En definitiva, HTML DOM es un estándar que permite obtener, cambiar, agregar o eliminar elementos HTML
-

## **Elemento.addEventListener("Evento", funcion);**

- Ejemplo:

```
boton.addEventListener("click", function(){  
    alert("Hola mundo"); });
```

- Se puede utilizar notación flecha

```
boton.addEventListener("click", () =>{  
    alert("Hola mundo"); });
```

- También se pueden utilizar una expresión de la función

```
boton.addEventListener("click", saludo);  
function saludo(){ alert ("Hola mundo");}
```

*OJO!! Recuerda que hay que cargar la página antes de empezar a trabajar*

```
window.addEventListener("load", iniciar);
```

# HTML DOM evento.target

Extraer el elemento que ha capturado el evento

Cuando se ejecuta

**Elemento.addEventListener("Evento", prueba);**

- Al definir la función se puede pasar el evento como parámetro (e)
- Para extraer el elemento al que se ha aplicado ese evento utilizamos target

```
function prueba(e){  
    elemento = e.target  
    console.log(elemento.innerHTML); }  
}
```

<https://developer.mozilla.org/es/docs/Web/API/Event/target>

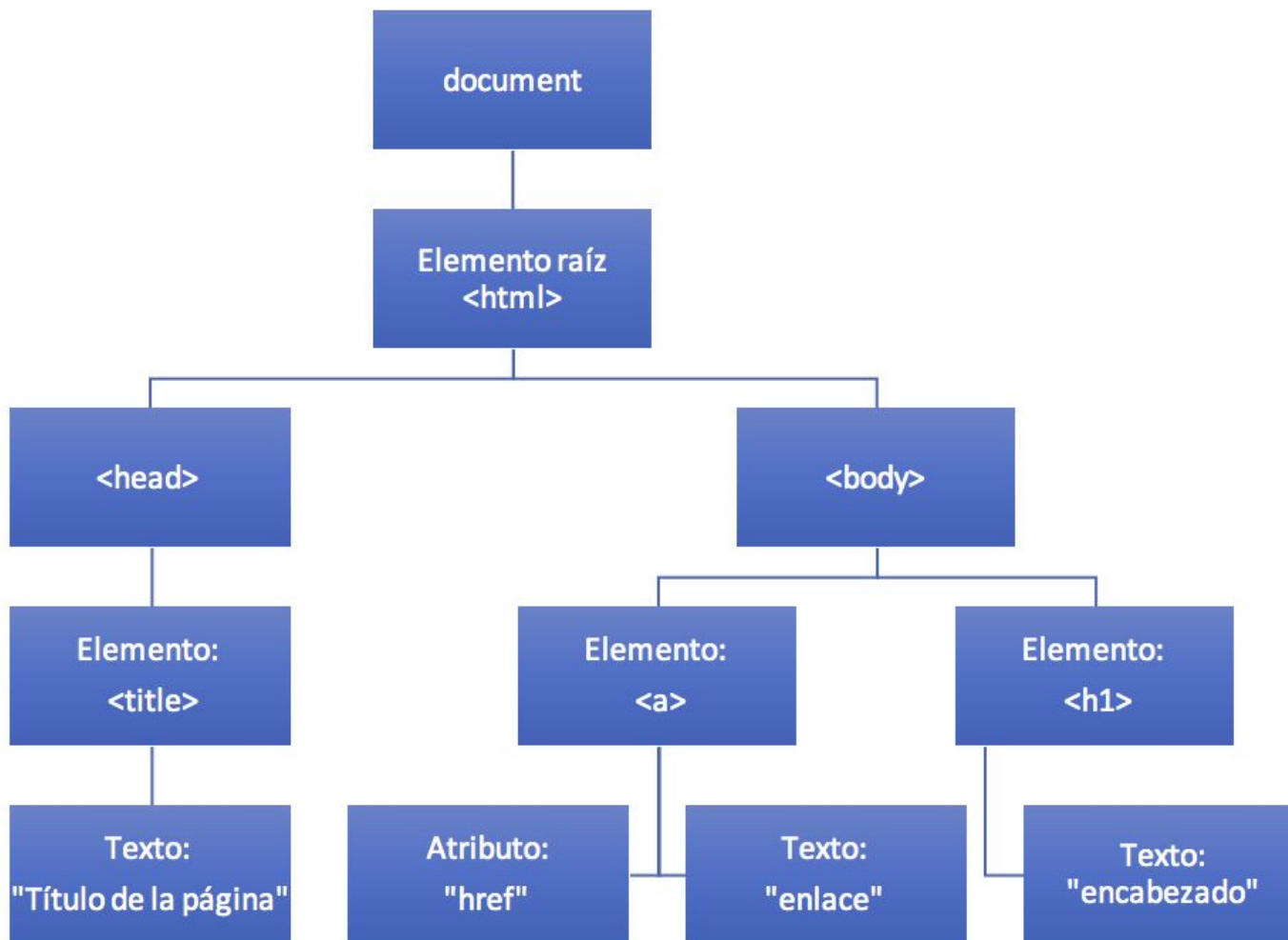
[https://www.w3schools.com/jsref/event\\_target.asp](https://www.w3schools.com/jsref/event_target.asp)

---



# HTML DOM

## Ejemplo de estructura de árbol DOM:



# HTML DOM

## Interfaz de programación DOM

- A HTML DOM se puede acceder con lenguajes de programación como JavaScript
  - En DOM todos los elementos o nodos HTML se definen como objetos, por lo que la interfaz de programación son las propiedades y los métodos de cada objeto.
  - Cuando un documento HTML se carga en un navegador web, se convierte en un objeto de document que representa al nodo raíz del documento HTML y es el "propietario" de todos los demás nodos
  - El objeto document proporciona propiedades y métodos para acceder a todos los objetos de nodo, desde dentro de JavaScript.
-

# HTML DOM

## Ejemplo

```
<!DOCTYPE html>
```

```
<html>
```

```
  <body>
```

```
    <p id="ejemplo">Primer párrafo</p>
```

```
  <script>
```

```
    document.getElementById("ejemplo").innerHTML = "¡Hola mundo!";
```

```
  </script>
```

```
  </body>
```

```
</html>
```

En el ejemplo:    **<p id="ejemplo">Primer párrafo</p>**

**<p>** --- elemento    **id** --- atributo    **Primer párrafo** --- texto

---

# HTML DOM

## Métodos para acceder a elementos

- **document.getElementById(id)** Encuentra un elemento por su id
  - **document.getElementsByTagName(name)**  
Encuentra los elementos por el nombre de su etiqueta
  - **document.getElementsByClassName(name)**  
Encuentra todos los elementos que contengan la clase
  - **document.querySelector(Selector CSS)** Devuelve el primer elemento que coincida con el selector CSS
  - **document.querySelectorAll(Selector CSS)** Devuelve todos los elementos que coincidan con el selector CSS
-

# HTML DOM. Métodos para acceder a elementos. Ejemplos

```
<!DOCTYPE html>
<html>
<head>
  <style>
    .micolor{
      color: pink;
    }
  </style>
</head>
<body>
  <p id="parrafo1">Primer párrafo</p>
  <p id="parrafo2" class="micolor">Segundo párrafo</p>
  <p id="parrafo3">Tercer párrafo</p>
  <p id="parrafo4" class="micolor">Cuarto párrafo</p>
  <script>
    document.getElementById("parrafo1").innerHTML = "¡Hola mundo!";
  </script>
</body>
</html>
```

```
>> document.getElementById("parrafo1").innerHTML
< "¡Hola mundo!"

>> document.getElementsByClassName("micolor").length
< 2

>> document.getElementsByClassName("micolor")[0].innerHTML
< "Segundo párrafo"

>> document.getElementsByTagName("p")
< ▶ HTMLCollection { 0: p#parrafo1, 1: p#parrafo2.mico, length: 4, ... }

>> document.getElementsByTagName("p")[0].innerHTML
< "¡Hola mundo!"

>> document.querySelector("p.micolor").innerHTML
< "Segundo párrafo"

>> document.querySelectorAll("p.micolor")[0].innerHTML
< "Segundo párrafo"

>> document.querySelectorAll("p.micolor")[1].innerHTML
< "Cuarto párrafo"
```

# HTML DOM

## Métodos para modificar elementos

- **elemento.innerHTML** Cambia el código HTML dentro de un elemento
  - **elemento.setAttribute(atributo, valor)** Cambia el valor de un atributo del elemento
  - **elemento.textContent** Cambia el contenido de texto del elemento
  - **element.style.property = nuevo estilo** Cambia el estilo de un elemento HTML
-



```
<!DOCTYPE html>
<html>
<head>
  <style>
    .micolor{
      color:pink;
    }
  </style>
</head>
<body>
  <p id="parrafo1">Primer párrafo</p>
  <p id="parrafo2" class="micolor">Segundo párrafo</p>
  <p id="parrafo3">Tercer párrafo</p>
  <p id="parrafo4" class="micolor">Cuarto párrafo</p>

  <script>
    document.getElementById("parrafo1").innerHTML = "¡Hola mundo!";
  </script>
</body>
</html>
```

¡Hola mundo!

Segundo párrafo

Tercer párrafo

Cuarto párrafo

Cambiar color fondo

Cambiar color

Ver textcontent

[Ver Ejemplo1](#)

```
>> document.querySelectorAll("p.micolor")[1].setAttribute("style", "background-color: red;");
```

```
← undefined
```

```
>> document.querySelectorAll("p.micolor")[1].textContent
```

```
← "Cuarto párrafo"
```

```
>> document.querySelectorAll("p.micolor")[1].style.color = "blue"
```

```
← "blue"
```

# HTML DOM

## Métodos para añadir y borrar elementos

- **document.createElement(Elemento)** Crea un elemento HTML
  - **document.removeChild(Elemento)** Elimina un elemento HTML
  - **document.appendChild(Elemento)** Añade un elemento HTML (al final)
  - **document.insertBefore(Nuevo, EleExistente)** añade un hijo, justo antes de otro hijo
  - **document.replaceChild(NuevoEle, AntiguoEle)** Reemplaza un elemento HTML
  - **document.write(texto)** Escribe dentro de flujo de salida HTML
-



# HTML DOM. Crear un elemento. Ejemplo

```
>> var encabezado = document.createElement("h2");
```

```
← undefined
```


```
>> var texto = document.createTextNode("SALUDOSSS")
```

```
← undefined
```

```
>> encabezado.appendChild(texto)
```

```
← ▶ #text "SALUDOSSS"
```

```
>> document.body.appendChild(encabezado)
```

```
← ▶ <h2> 
```

1º- Crear el elemento h2

2º- Crear el Nodo texto que irá en el encabezado

3º- Añadir el texto al elemento encabezado

4º- Añadir el encabezado al body del documento

**SALUDOSSS!**

---

# HTML DOM. Crear elemento con insertBefore

Ejecutar [Ejemplo2](#)

Primer párrafo

Segundo párrafo

Encabezado

Imagen

Primer párrafo

**hola :-)**

Segundo párrafo

Encabezado

Imagen

*Se quiere añadir un H1 entre Primer párrafo y Segundo párrafo*

```
>> mih1 = document.createElement("h1");
< ▶ <h1>

>> mitexto = document.createTextNode("hola :-)")
< ▶ #text "hola :-)"

>> mih1.appendChild(mitexto);
< ▶ #text "hola :-)"

>> parrafo2 = document.body.children[1]
< ▶ <p id="parrafo2" class="micolor"> 🚫

>> document.body.insertBefore(mih1,parrafo2)
< ▶ <h1> 🚫
```

1º- Crear el elemento h1

**(createElement)**

2º- Crear el Nodo texto que irá en el encabezado **(createTextNode)**

3º- Añadir el texto al elemento encabezado **(appendChild)**

4º Localizar el hermano que le queremos insertar el h1 delante **(parrafo2)**

5º- Añadir el encabezado al body del documento antes de parrafo2 **(insertBefore)**

# HTML DOM. OJO con firstChild !!!

Ejecutar [Ejemplo2](#)

Partiendo de esta situación...

**SALUDOSSS!!! - 1**

**SALUDOSSS!!! - 2**

Encabezado

Imagen

```
<div id=caja1> </div>
```

Observamos que hay un espacio en blanco en el div

```
>> cont= document.getElementById("caja1");
```

```
< ▶ <div id="caja1"> ☒
```

Seleccionamos el elemento padre (caja1)

```
>> cont.children
```

```
< ▶ HTMLCollection [ h2 ☒ , h2 ☒ ]
```

**children** devuelve los elementos de cont (2 h2)

```
>> cont.childNodes
```

```
< ▶ NodeList(3) [ #text ☒ , h2 ☒ , h2 ☒ ]
```

**childNodes** devuelve los nodos hijo (incluyendo el “ “)

```
>> cont.firstChild
```

```
< ▶ #text " " ☒
```

**firstChild** devuelve el primer nodo: “ “

```
>> cont.firstElementChild
```

```
< ▶ <h2> ☒
```

**firstElementChild** devuelve el primer elemento: h2

# HTML DOM. Borrar un elemento. Ejemplo

```
>> cont = document.getElementById("caja1");
```

```
< > <div id="caja1"> ☒
```

```
>> hijo= cont.firstChild
```

```
< > <h2> ☒
```

```
>> cont.removeChild(hijo)
```

```
< > <h2>
```

1º- Seleccionar el elemento padre (caja1)

2º- Seleccionar el elemento hijo (h2) -- *el primer hijo*

3º- Borrar el hijo seleccionado

*Ojo!!: No utilizamos firstChild*

Primer párrafo

Segundo párrafo

**SALUDOSSS!!!---1**

**SALUDOSSS!!!---2**

**SALUDOSSS!!!---3**

Encabezado

Imagen

Primer párrafo

Segundo párrafo

**SALUDOSSS!!!---2**

**SALUDOSSS!!!---3**

Encabezado

Imagen



# HTML DOM. Sustituir un elemento. Ejemplo

```
>> cont=document.getElementById("caja1");
< ▶ <div id="caja1"> 

>> elemento = document.createElement("h2");
< ▶ <h2>

>> texto = document.createTextNode("Mi saludo....");
< ▶ #text "Mi saludo...."

>> elemento.appendChild(texto);
< ▶ #text "Mi saludo...."

>> cont.replaceChild(elemento,cont.firstChild);
< ▶ <h2>
```

**SALUDOSSS!!! - 1**

**SALUDOSSS!!! - 2**

**SALUDOSSS!!! - 3**

1º- Seleccionar el elemento padre (caja1)

2º- Crear el nuevo elemento hijo (h2)

3º- Crear el texto del encabezado

4º Unir textNode al h2

5º Sustituir el elemento que queramos por el nuevo.

**Mi saludo....**

**SALUDOSSS!!! - 2**

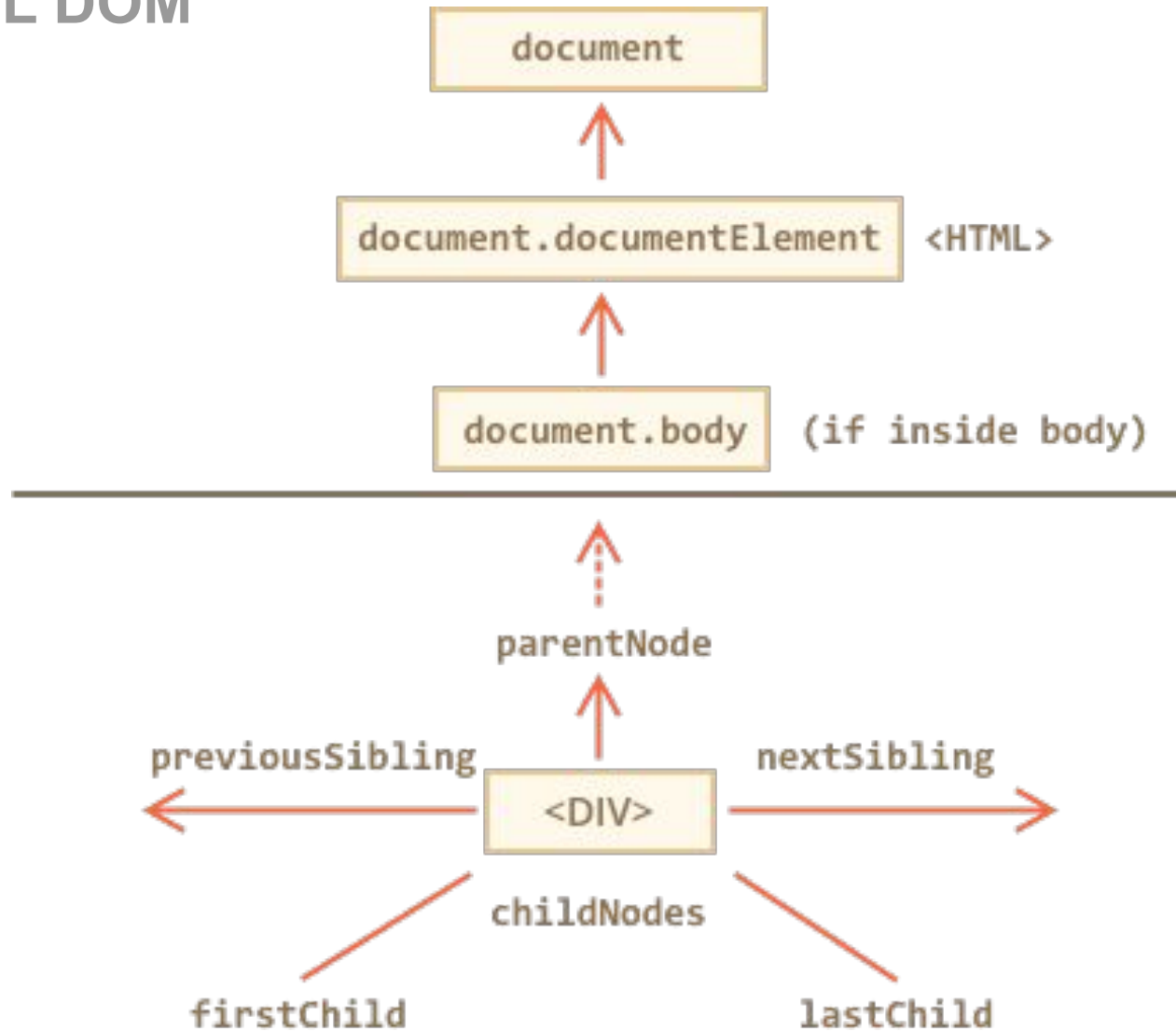
**SALUDOSSS!!! - 3**

# HTML DOM

## Métodos para navegar entre Nodos DOM

- **document.childNodes** Devuelve una lista con todos los elementos hijos de un elemento
  - **document.parentNode** Devuelve el padre del elemento
  - **document.nextSibling** Seleccionar el siguiente elemento hermano
  - **document.previousSibling** Seleccionar el elemento hermano anterior
  - **document.firstChild** Primer hijo del elemento
  - **document.lastChild** Último hijo del elemento
-

# HTML DOM



[https://www.w3schools.com/js/js\\_htmlDOM\\_navigation.asp](https://www.w3schools.com/js/js_htmlDOM_navigation.asp)

# HTML DOM

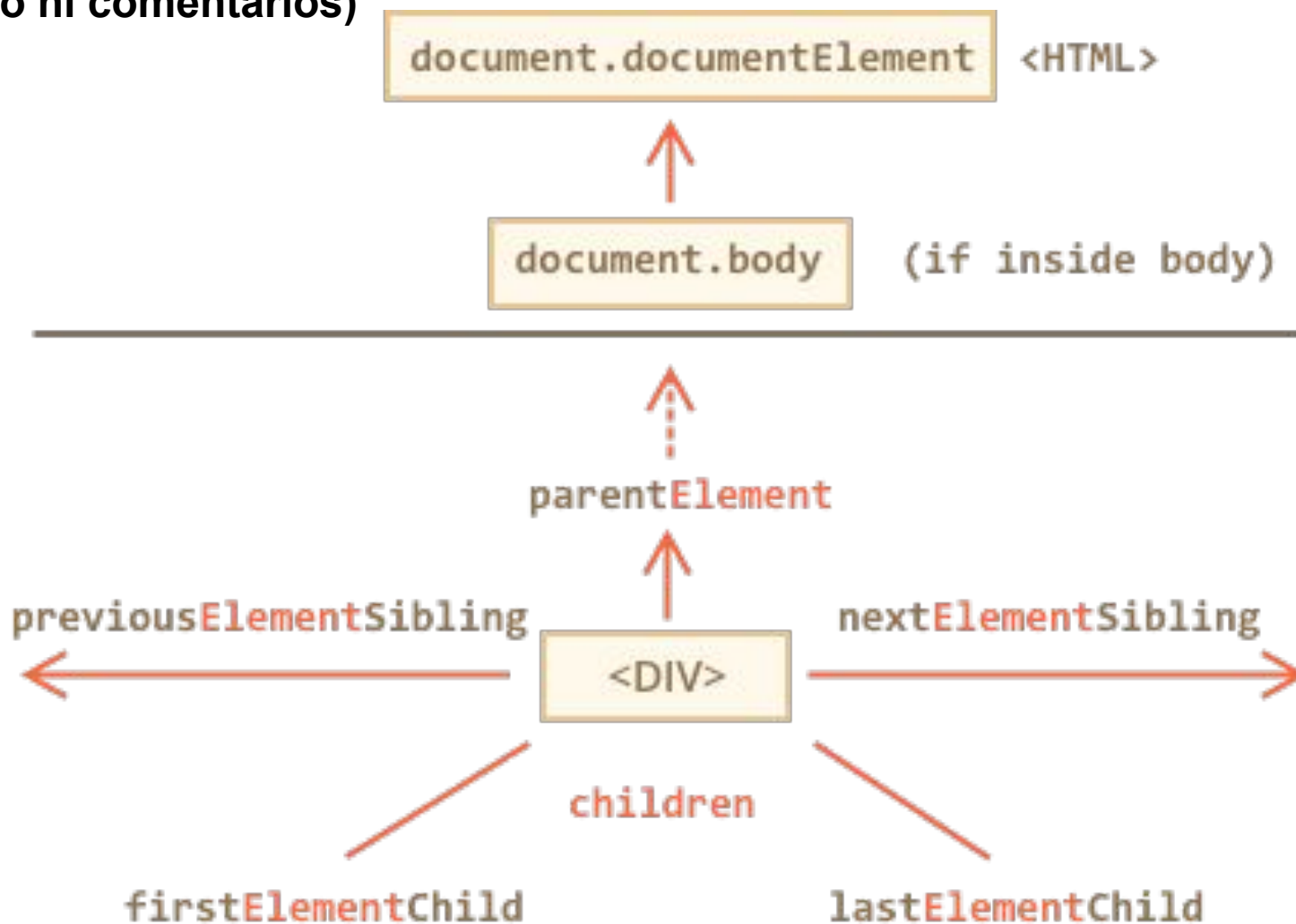
Métodos para navegar entre en elementos HTML (no texto ni comentarios)

- **document.children** Devuelve una lista con todos los elementos hijos de un elemento
  - **document.parentElement** Devuelve el padre del elemento
  - **document.nextElementSibling** Seleccionar el siguiente elemento hermano
  - **document.previousElementSibling** Seleccionar el elemento hermano anterior
  - **document.firstChild** Primer hijo del elemento
  - **document.lastChild** Último hijo del elemento
-







# HTML DOM

Métodos para navegar entre nodos en vez de elementos HTML (no texto ni comentarios)



- <https://javascript.info/dom-navigation>

# HTML DOM. Navegar por elementos. Ejemplo

>> <code>cont=document.getElementById("caja1");</code>	Seleccionamos el elemento padre (caja1)
< ▶ <code>&lt;div id="caja1"&gt;</code> 	
>> <code>hijo1 =cont.firstElementChild;</code>	Accedemos al primer hijo (h2)
< ▶ <code>&lt;h2&gt;</code> 	Accedemos a su texto:SALUDOS-1
>> <code>hijo1.textContent</code>	Accedemos a su hermano. Vemos que su texto es SALUDOS - 2
< "SALUDOSSS!!! - 1"	
>> <code>hijo2 = hijo1.nextElementSibling;</code>	Accedemos al último elemento. su texto es SALUDO -3
< ▶ <code>&lt;h2&gt;</code> 	
>> <code>hijo2.textContent</code>	Accedemos al elemento anterior
< "SALUDOSSS!!! - 2"	
>> <code>final = cont.lastElementChild;</code>	Accedemos al padre <div id="caja1">
< ▶ <code>&lt;h2&gt;</code> 	
>> <code>cont.lastElementChild.textContent</code>	
< "SALUDOSSS!!! - 3"	
>> <code>final.previousElementSibling.textContent;</code>	
< "SALUDOSSS!!! - 2"	
>> <code>hijo2.parentElement.id</code>	
< "caja1"	

**SALUDOSSS!!! - 1**

**SALUDOSSS!!! - 2**

**SALUDOSSS!!! - 3**

## HTML DOM. Propiedades referentes a Clases

- **Element.className:** Obtiene y establece el valor del atributo class del elemento especificado. [Ejemplos](#)
  - **Element.classList** devuelve una colección activa de DOMTokenList de los atributos de clase del elemento.
  - **Métodos de classList:** [Ejemplos](#)
    - **add (clase1 [,clasen])** : Añade clases al elemento
    - **remove (clase1[,clasen])**: Elimina las clases
    - **toggle(clase[,force])**: Si la clase existe, la elimina, si no existe, la añade. force=true:añade, force=false:elimina
    - **replace (claseant,clasenueva)**: Reemplaza clase existente por una nueva
    - **contains (clase)**: Comprueba si la clase existe en el elemento.
    - **item(indice)**: Devuelve el valor de la clase por índice
-

**ESCUELA PÚBLICA:**

**DE TOD@s**

**PARA TOD@s**

Vallecas