

CURSO .JS

JavaScript en la Web



Autor: Jon Vadillo
www.jonvadillo.com

JavaScript en el navegador

- Cuando ejecutamos JavaScript en un navegador, tenemos los siguientes elementos disponibles:
 - **DOM** (Document Object Model) → contenido
 - **BOM** (Browser Object Model) → navegador
 - **JavaScript**

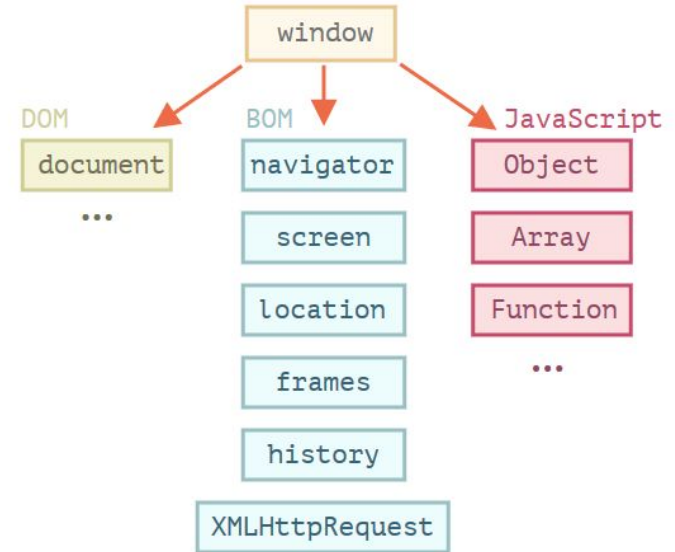


Image source: javascript.info

BOM

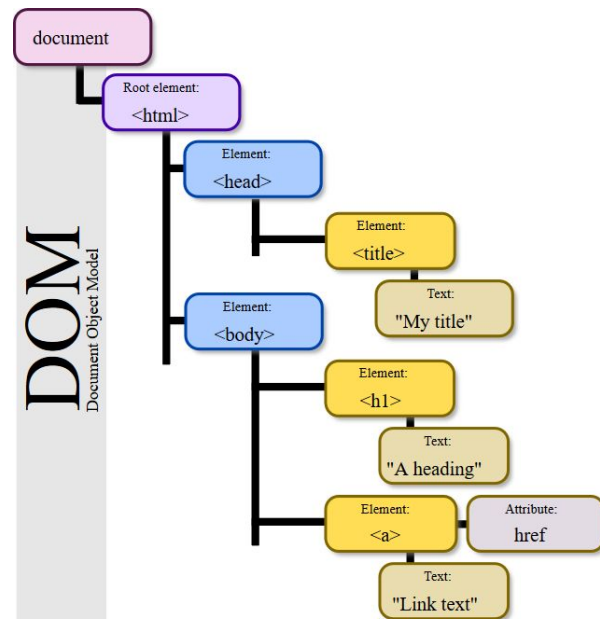
- Permite acceder mediante JS a elementos del navegador que no sean el contenido, como por ejemplo:
 - **location**: permite acceder y manipular la URL de la barra de direcciones
 - **history**: permite manejar el historial de navegación
 - **navigator**: permite acceder a elementos del dispositivo como la batería, la vibración, geolocalización,...
 - **screen**: información sobre la pantalla
 - etc.

“El contenido de la página es almacenado en DOM y el **acceso y la manipulación** se hace vía JavaScript”

- Mozilla Developer Network

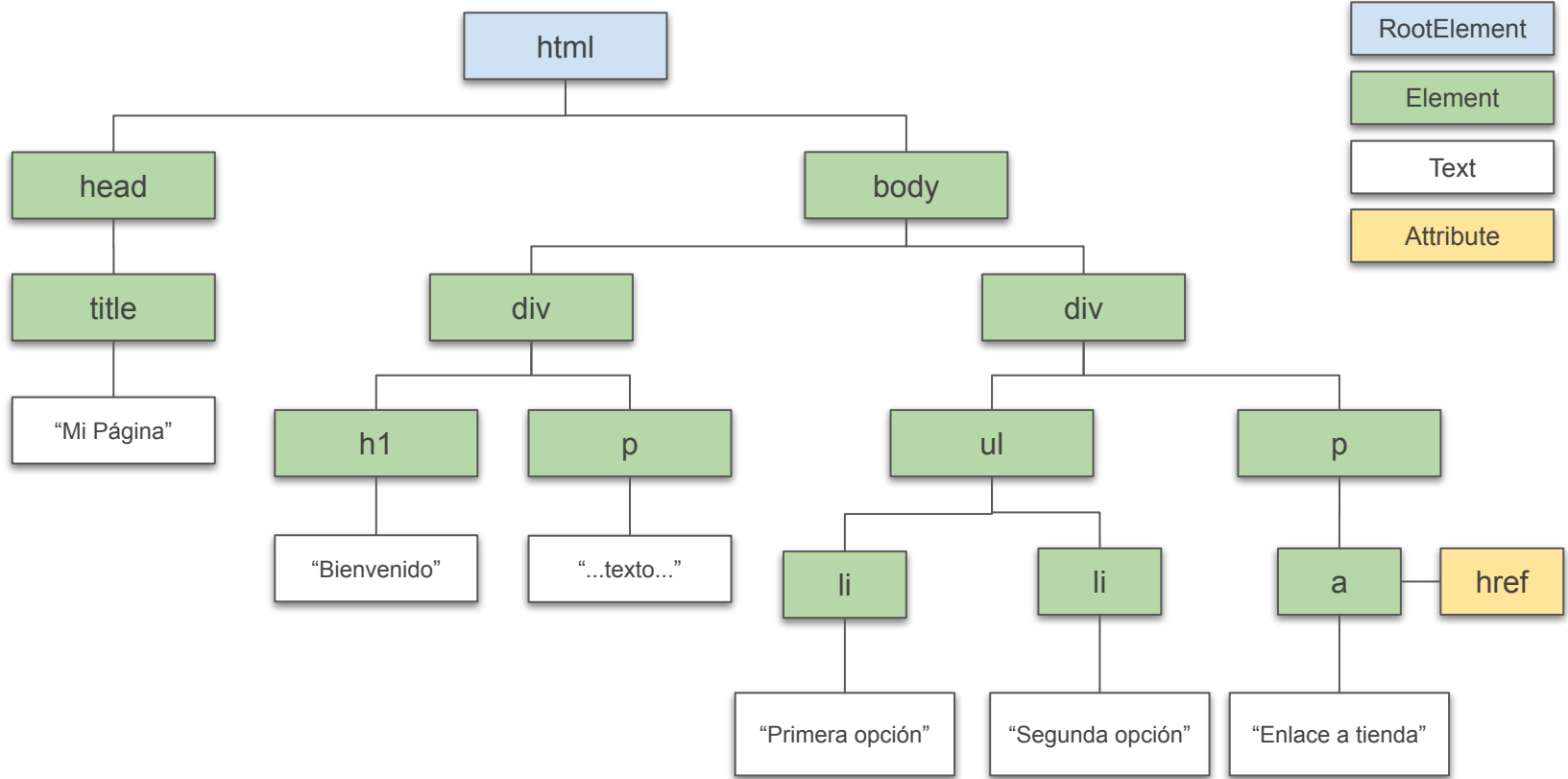
DOM

- Una página web es un documento.
- El DOM es una **representación del documento HTML** como un **grupo de nodos y objetos** que tienen propiedades y métodos.
- Cada etiqueta HTML es un objeto y todos forman una estructura en forma de árbol.
- Estos **objetos son accesibles mediante JS**, pudiendo así modificar la estructura, estilo y contenido de la página.



source: Wikimedia

Tipos de nodos (nodeType)



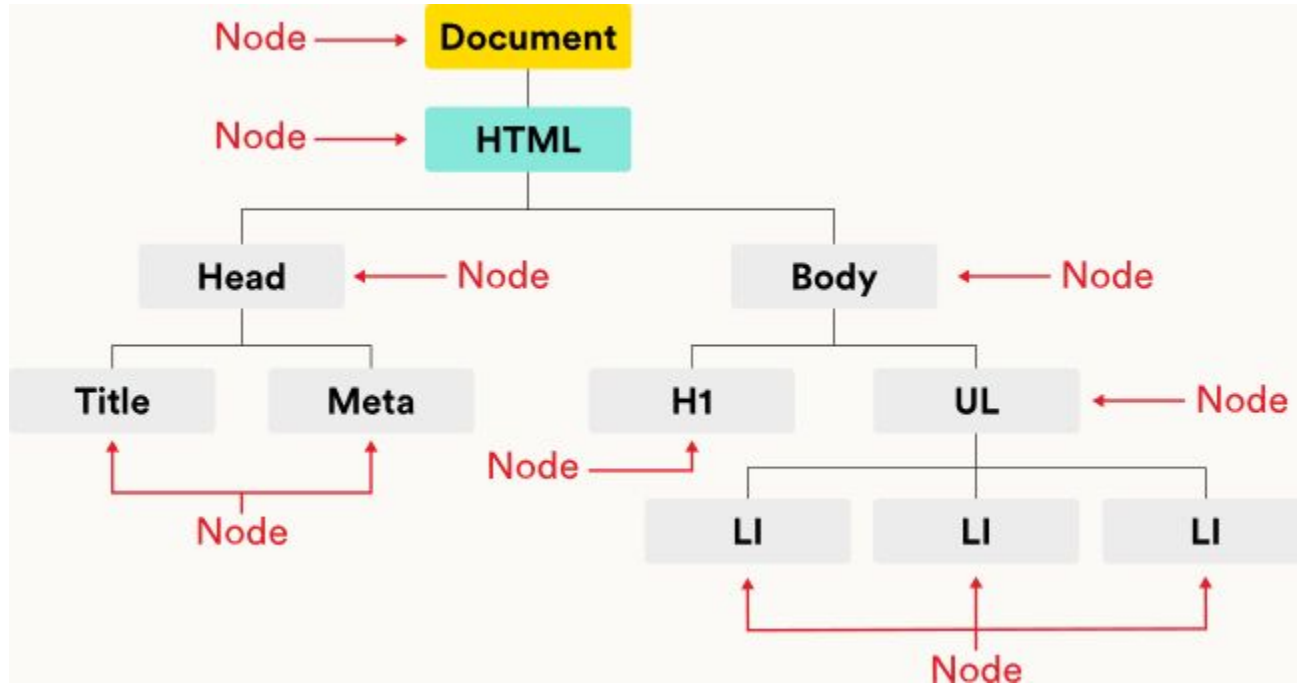


Image source: <https://fundamentals.generalassemb.ly/>

Inspector

Consola

Depurador

Red

Editor de estilos

Rendimiento

Memoria

»

Buscar en HTML

<h2 class="headline | color_gray_ultra_dark font_secondary width_full headline_md ">

...

</h2>

<div class="col desktop_12 tablet_8 mobile_4">

<div class="byline | uppercase color_gray_ultra_dark margin_bottom width_full">

Ana Torres Menárguez

Ignacio Zafra

...

</div>

<p class="description | color_gray_medium block width_full false false">...</p>

<div class="width_full margin_top">...</div>

</div>

</article>

<article class="story_card story_card_default | flex flex_wrap align_content_start

st_column.|desktop_8.tablet_...

>

article.story_card.story_card_default.|...

>

h2.headline.|color_gray_ultra_dark.font...

>

»

+

✎

Filtrar estilos

Dispositivo

Flexbox

Rejilla

Modelo de caja

elemento {

color: inherit;

a, a:hover, a:visited {

text-decoration: none;

a {

color: #333;

Heredado de h2

...een and (min-width: 1001px)

.b_chain_left .first_column

.story_card:first-of-type

.headline {

font-size: 3.8rem;

Selecione un elemento Flex p continuar.

No hay rejillas en esta página

margin

border

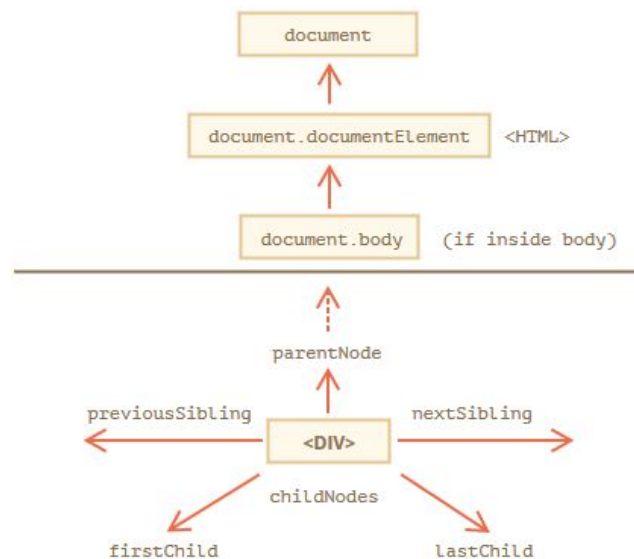
padding

0 0 0 639.!

Recorrer elementos del DOM

- Podemos navegar desde un elemento a otros elementos que estén relacionados:

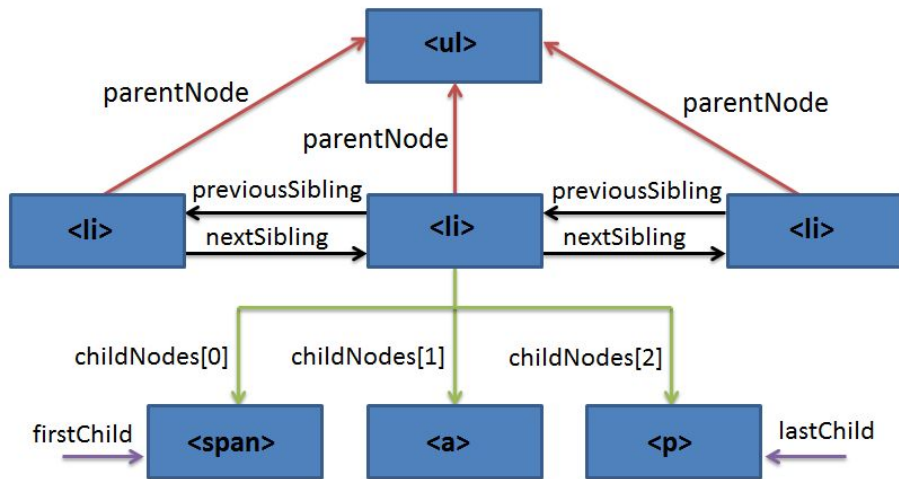
Palabra clave	Significado
<code>parentNode</code>	Nodo padre de un nodo
<code>childNodes</code>	Array conteniendo los hijos de un nodo
<code>firstChild</code>	Primer hijo de un nodo (empezando por la izquierda)
<code>lastChild</code>	Ultimo hijo de un nodo (el más a la derecha)
<code>nextSibling</code>	Próximo nodo hermano (situado a la derecha)
<code>previousSibling</code>	Anterior nodo hermano (situado a la izquierda)



[Image source: javascript.info](http://javascript.info)

Recorrer elementos del DOM

```
<ul>
  <li>node</li>
  <li>
    <span>node</span>
    <a href="#">node</a>
    <p>node</p>
  </li>
  <li>node</li>
</ul>
```



¿Cómo acceder a elementos del DOM?

- El DOM nos deja accesible **el elemento document**, el cual representa la raíz del documento y **proporciona muchos métodos** y propiedades..
- Métodos importantes:

<code>document.getElementById("id-principal");</code>	Devuelve el elemento con el <i>id</i> indicado
<code>document.getElementsByClassName("subtitulo");</code>	Devuelve un array con los elementos que tienen las clases indicadas.
<code>document.getElementsByTagName("LI");</code>	Devuelve un listado de elementos que tienen la etiqueta indicada
<code>document.getElementsByName("username");</code>	Devuelve un listado de elementos que tienen el atributo name indicado

getElementById

```
<div id="elem">  
  <div id="elem-content">...contenido...</div>  
</div>
```

```
// conseguir el elemento  
let elem = document.getElementById('elem');  
  
// mostrar su contenido  
console.log(elem.textContent); // ...contenido...
```

getElementsByName & getElementsByClassName

```
<form name="formulario">  
  <div class="panel">Panel informativo A</div>  
  <div class="panel">Panel informativo B</div>  
</form>
```

```
// conseguir el elemento por nombre  
let form = document.getElementsByName('formulario')[0];  
// conseguir el elemento por clase, dentro del formulario  
let paneles = form.getElementsByClassName('panel');  
alert(paneles.length); // 2, hay dos elementos con clase 'panel'
```

getElementsByTagName

```
<label>
  <input type="radio" name="edad" value="menor"> Menor de edad
</label>
<label>
  <input type="radio" name="edad" value="mayor"> Mayor de edad
</label>
```

```
let inputs = table.getElementsByTagName('input');
for (let input of inputs) {
  alert( input.value + ': ' + input.checked );
}
```

QuerySelector

- Devuelve el primer elemento que **coincide con el selector CSS** indicado.

```
<div id="prueba">  
  <span id="elementoA" class="clase" title="Azul"></span>  
  <span id="elementoB" class="clase" title="Verde"></span>  
  <span id="elementoC" class="clase" title="Rojo"></span>  
</div>
```

```
let elementos = document.querySelector('#elementoC').title; //Rojo  
let elementos = document.querySelector('#prueba .clase').title; //Azul  
let verde = document.querySelector('#prueba .noexiste').title; //ERROR
```

QuerySelectorAll

```
<div id="prueba">
  <span id="elementoA" class="miclase" title="Azul"></span>
  <span id="elementoB" class="miclase" title="Verde"></span>
  <a id="elementoC" href="#"></a>
</div>
```

```
// Obtener todos los elementos de clase 'miclase'
let elementos = document.querySelectorAll('.miclase');
// Obtener todos los elementos 'span' y 'a'
let elementos = document.querySelectorAll('span', 'a');
```


Resumen selectores

Method	Searches by...
<code>querySelector</code>	CSS-selector
<code>querySelectorAll</code>	CSS-selector
<code>getElementById</code>	<code>id</code>
<code>getElementsByName</code>	<code>name</code>
<code>getElementsByTagName</code>	tag or <code>'*'</code>
<code>getElementsByClassName</code>	class

Hands on!

- Selecciona el elemento con id “*titular*” y muestra su contenido en la consola mediante la propiedad *textContent*.
- Selecciona todos los elementos con clase “*importante*” y muestra su contenido por la consola.

```
<h1 id="titular">Lorem ipsum dolor sit amet </h1>
<p>Sed justo <span class="importante">mauris</span>, luctus id lorem at, egestas
condimentum leo. Nam in diam id felis lacinia <span class="titular">posuere</span>
non eu enim. Donec pulvinar neque convallis.</p>
<p>Neque <span class="importante">congue</span> iaculis. Nam vel sem sit amet
ligula </a>mollis semper id eu mauris. Nam gravida ultrices nisi non porttitor.
Vestibulum a <span class="importante">vehicula</span> risus. Nunc ut imperdiet
mauris.</p>
```

Hands on!

- o Crea una página HTML con el siguiente contenido y selecciona todos los enlaces (etiqueta <a>). Muestra en la consola el número de elementos seleccionados.

```
<h1>Lorem ipsum dolor sit amet </h1>
<p>consectetur adipiscing elit. Duis purus tellus, condimentum ut <a
href="https://deusto.es/">euismod</a> eu, tristique at odio.</p>
<p>Sed justo mauris, luctus id lorem at, egestas condimentum leo. Nam in diam
id felis lacinia <a href="https://deusto.es/">posuere non eu</a> enim. Donec
pulvinar neque convallis.</p>
<p>neque congue iaculis. Nam vel sem sit <a href="https://deusto.es/">amet
ligula </a>mollis semper id eu mauris. Nam gravida ultrices nisi non
porttitor. Vestibulum a vehicula risus. Nunc ut imperdiet mauris. </p>
```

Cambiar el estilo

- Es posible acceder a los estilos de un elemento y cambiar sus valores.
- Los elementos tienen una propiedad llamada **style** que contiene sus estilos.
 - Los nombres de propiedades CSS compuestos por varias palabras se escribirán en formato *lower camel case*: `fontSize`, `maxWidth`, `marginTop`, ...

```
// Ocultar el body
document.body.style.display="none";

// Cambiar el color a rojo
document.getElementById("id").style.color="red";
document.getElementById("id").style.fontSize="22px";
```

Hands on!

- Crea una página HTML y cambia el color de fondo del cuerpo de la página (*body*) utilizando JS.
- Realiza las siguientes operaciones en el código HTML dado a continuación:
 - Selecciona el elemento *h2* de la página y cambia su color de texto a *#ff0000*
 - Establece la propiedad *font-weight* a *700* en los elementos de clase *"importante"*.

```
<h1>Lorem ipsum dolor sit amet </h1>
<h2>Nam gravida ultrices nisi non porttitor.</h2>
<p>Sed justo <span class="importante">mauris</span>, luctus id lorem at, egestas
condimentum leo. Nam in diam id felis lacinia non eu enim. Donec pulvinar neque
convallis.</p>
<p>Neque <span class="importante">congue</span> iaculis. Nam vel sem sit amet
ligula </a>mollis semper id eu mauris. Nam gravida ultrices nisi non porttitor.
Vestibulum a <span class="importante">vehicula</span> risus. Nunc ut imperdiet
mauris.</p>
```

Alterar el DOM

- Podemos utilizar distintas funciones para alterar el DOM desde JavaScript.
 - `element.innerHTML()`: devuelve (o cambia) el contenido HTML del elemento

```
<div id="txt">  
  <p>primer parrafo hijo</p>  
  <p>segundo parrafo hijo</p>  
</div>
```

```
let txt = document.getElementById("txt");  
console.log(txt.innerHTML);  
/* RESULTADO:  
<p>primer parrafo hijo</p>  
<p>segundo parrafo hijo</p>  
*/  
  
txt.innerHTML = "<p>Nuevo Párrafo</p>"
```

Alterar el DOM

- **element.textContent:** devuelve (o cambia) el texto del elemento seleccionado.

```
<div id="post">  
  <h1>Aprendiendo JS</h1>  
  <p>Estamos aprendiendo JS</p>  
</div>
```

```
let parrafo = document.getElementsByTagName('p')[0];  
alert(parrafo.textContent); // Estamos aprendiendo JS  
// Actualizar el contenido  
parrafo.textContent = 'Nuevo contenido para el párrafo'
```

Crear elementos

- `document.createElement(tag)`: crea un elemento con la etiqueta indicada.

```
var div = document.createElement("div");  
var p = document.createElement("p");
```

```
let div = document.createElement('div');  
div.className = "panel-principal";  
div.innerHTML = "<p>Bienvenido a <strong>mi web</strong>";
```


Métodos para la inserción

<code>node.append(...nodes or strings)</code>	Inserta nodos/strings al final del nodo
<code>node.prepend(...nodes or strings)</code>	Inserta nodos/strings al principio del nodo
<code>node.before(...nodes or strings)</code>	Inserta nodos/strings antes del nodo
<code>node.after(...nodes or strings)</code>	Inserta nodos/strings después del nodo
<code>node.replaceWith(...nodes or strings)</code>	Reemplaza el nodo por los nodos/strings

Métodos para la inserción

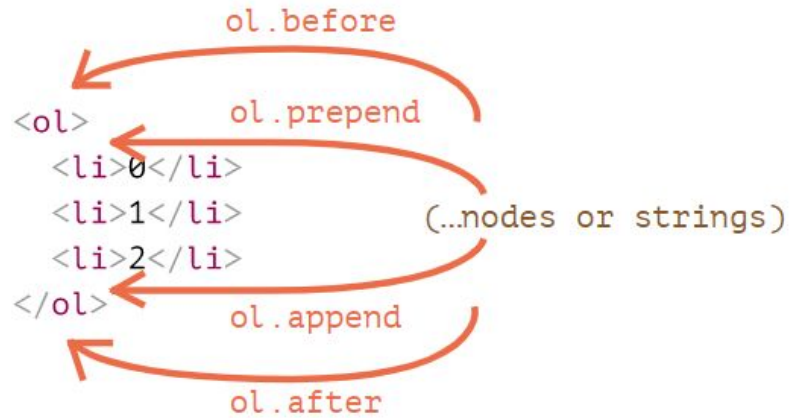
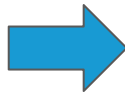


Image source: javascript.info

Hands on!

- Genera una lista HTML con los datos de un objeto como el siguiente:

```
const estudiante = {  
  nombre : 'Amaia',  
  apellidos : 'Jainaga Urrutia',  
  edad : 27,  
  email : 'amaia@email.com'  
}
```



Datos de estudiante

- nombre: Amaia
- apellidos: Jainaga Urrutia
- edad: 27
- email: amaia@email.com

Hands on!

- Genera una tabla HTML como la de la imagen. Los datos de la tabla deberán estar almacenados en un array de objetos con la siguiente estructura:

```
const tareas = [  
  {  
    id : 1,  
    descripcion : '...',  
    responsable : 'Mikel',  
    fecha : '03-10-2020'  
  },  
  ...  
]
```



Lista de tareas

ID	Fecha	Responsable	Descripción
1	03-10-2020	Mikel	Lorem ipsum dolor sit amet, consectetur adipiscing elit.
2	12-08-2020	Unai	Lorem ipsum dolor sit amet, consectetur adipiscing elit.
3	11-12-2020	Ane	Lorem ipsum dolor sit amet, consectetur adipiscing elit.
4	07-06-2020	Nora	Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Alternativa para añadir HTML

- `node.insertAdjacentHTML(where, html)`: dado un código HTML, lo inserta en el lugar indicado:
 - **beforebegin** → lo inserta justo antes del elemento
 - **afterbegin** → lo inserta dentro del elemento, al comienzo.
 - **beforeend** → lo inserta dentro del elemento, al final.
 - **afterend** → lo inserta justo después del elemento

```
document.body.insertAdjacentHTML('beforebegin','<p>Primer párrafo</p>');
```

Alternativa para añadir HTML

```
<!-- beforebegin -->  
<p>  
  <!-- afterbegin -->  
  foo  
  <!-- beforeend -->  
</p>  
<!-- afterend -->
```

Crear HTML con Template Literals

```
let estudiante = {  
  id : 126,  
  nombre : 'Maite',  
  edad : 23,  
  email : 'maite@email.com'  
}  
  
const lista =  
  `


    <li>Nombre: ${estudiante.nombre}</li>  
    <li>Edad: ${estudiante.edad}</li>  
    <li>Email: ${estudiante.email}</li>  
  </ul>`;  
  
document.body.innerHTML = lista;
```

Borrar un nodo

```
<div id="div-01">Este es el div-01</div>  
<div id="div-02">Este es el div-02</div>  
<div id="div-03">Este es el div-03</div>
```

```
var el = document.getElementById('div-01');  
// Elimina el div con el id 'div-03'  
el.remove();  
el = document.getElementById('div-02');  
// Elimina el div con el id 'div-03'  
el.nextElementSibling.remove();
```


Clases CSS

- Podemos acceder a las clases de un elemento HTML mediante la propiedad `className`.

```
<p class="descripcion principal">
```

```
let elemento = document.getElementsByTagName('p')[0];  
// Acceder a las clases  
console.log(elemento.className); // descripcion principal  
// Establecer un nuevo valor para el atributo class del elemento:  
elemento.className = "otraclase segundaclase"
```

Clases CSS

- En ocasiones queremos solo añadir/quitar una clase concreta. Para ello podemos utilizar los métodos `elem.classList.add("class")` y `elem.classList.remove("class")`

```
<p class="descripcion">
```

```
let elemento = document.getElementsByTagName('p')[0];  
// Añadir la clase principal  
elemento.classList.add("principal");  
// Eliminar la clase descripcion  
elemento.classList.remove("descripcion");
```

Clases CSS

- El objeto **classList** también contiene otros dos métodos muy utilizados:
 - `elem.classList.toggle("class")` → añade la clase si no existe o la elimina si ya existe.
 - `elem.classList.contains("class")` → comprueba si la clase existe y devuelve true/false.

Eventos en la Web

- Los eventos son **acciones u ocurrencias que suceden en nuestra página web**. El navegador nos avisará de estos eventos para que podamos **responder o reaccionar** si lo deseamos.
- Ejemplos:
 - El usuario hace click en un botón.
 - El usuario presiona una tecla del teclado.
 - La página termina de cargar su contenido.
 - El usuario mueve el mouse sobre un elemento de la página.
 - Se envía un formulario.

Eventos en la Web

- Podemos definir **controladores de eventos** (*event listeners* o *event handlers*).
- Los controladores serán bloques de código que se ejecutarán cuando un evento concreto ocurra.
- **Tipos de eventos**
 - Mouse → click, mouseover/mouseout, mousedown/mouseup, mousemove
 - Form → submit, focus
 - Keyboard → keydown, keyup
 - DOMContentLoaded

Capturar eventos

- Existen 3 formas de capturar eventos:
 - En atributos HTML (inline event handlers)
 - Como propiedad de un elemento del DOM
 - Funciones `addEventListener` y `removeEventListener`

Atributos HTML

```
<button onclick="alert('Hola!');">Púlsame</button>
```

```
<input type="text" onfocus="alert(Me has seleccionado!');">
```

Atributos HTML

```
<button onclick="saludar();">Púlsame</button>
```

```
function saludar() {  
    alert('Hola!');  
}
```


Eventos en propiedades

```
<button id="enviar">Púlsame</button>
```

```
function saludar() {  
    alert('Hola!');  
}  
  
let elem = document.getElementById('enviar');  
elem.onclick = saludar;
```

addListener

- Permite añadir más de un controlador (handler) a un evento

```
<button id="enviar">Púlsame</button>
```

```
function saludar() {  
    alert('Hola!');  
}  
  
let elem = document.getElementById('enviar');  
elem.addEventListener('click', saludar);
```

addEventListener

```
function saludar() {  
    alert('Hola!');  
}  
  
function hablar() {  
    alert('¿Qué tal?');  
}  
  
let elem = document.getElementById('enviar');  
elem.addEventListener('click', saludar);  
elem.addEventListener('click', hablar);
```

removeEventListener

- Permite **eliminar** los controladores anteriormente definidos.

```
let elem = document.getElementById('enviar');  
elem.addEventListener('click', saludar);  
  
...  
  
elem.removeEventListener('click', saludar);
```

El objeto Event

- Contiene información adicional y se pasa automáticamente a los controladores.

```
<button id="enviar">Púlsame</button>
```

```
function saludar(event) {  
    // Tipo de evento y elemento que lo ha producido  
    alert(event.type + " en " + event.currentTarget);  
}  
  
let elem = document.getElementById('enviar');  
elem.addEventListener('click', saludar);
```

Fases de un evento

- Un evento estándar de JS pasa por 3 fases:
 1. Captura: el evento recorre el DOM hasta el elemento que lo produce
 2. El evento llega al elemento
 3. Bubbling: el evento viaja por los padres hasta la raíz.

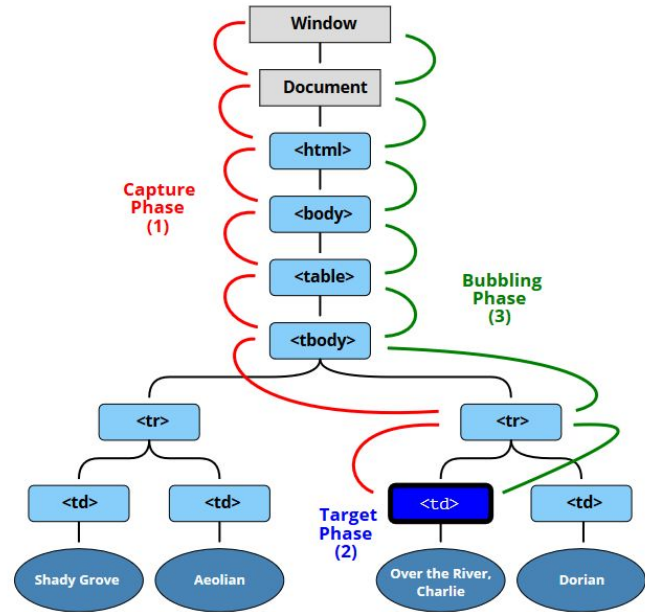


Image source: javascript.info

Event bubbling

- Cuando un evento ocurre en un elemento, primero se ejecuta en el elemento y luego atraviesa todo el DOM, propagándose por sus padres.
- Se van ejecutándose en orden los handlers (controladores) añadidos mediante **addEventListener**.
- **event.target**: elemento donde ocurre el evento
- **event.currentTarget**: hace referencia al elemento al cual el controlador del evento fue asociado
- **stopPropagation()**: detiene la propagación del evento

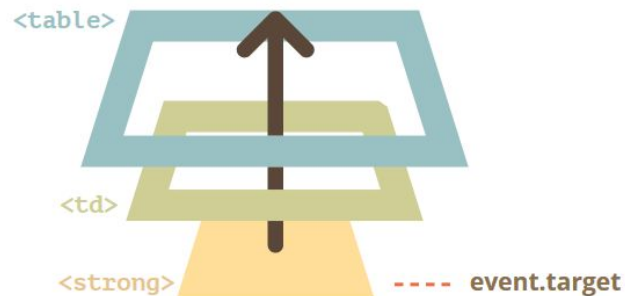


Image source: javascript.info

Acciones por defecto del navegador

- Algunos eventos tienen asociadas **acciones por defecto parte del navegador**:
 - Un click en un enlace → navega a la URL del enlace
 - Un click en un botón de envío de un formulario → envía el formulario
- Al manejar eventos es probable que queramos **evitar que ocurra la acción por defecto** del navegador.
 - Existen distintas formas de hacerlo, la más común es mediante el método `event.preventDefault()`:

```
<a href="/" onclick="event.preventDefault()">Click</a>
```


Acciones por defecto del navegador

- Algunos eventos tienen asociadas **acciones por defecto parte del navegador**:
 - Un click en un enlace → navega a la URL del enlace
 - Un click en un botón de envío de un formulario → envía el formulario
- Al manejar eventos es probable que queramos **evitar que ocurra la acción por defecto** del navegador.

Acciones por defecto del navegador

- Existen distintas formas de hacerlo, la más común es mediante el método `event.preventDefault()`, el cual cancela el evento.

```
<a href="/" onclick="event.preventDefault()">Click</a>
```

```
function stopDefAction(event) {  
    event.preventDefault();  
}
```

Acciones por defecto del navegador

```
<form id="registro" method="post">
  <input type="text" name="nombre" />
  <input type="submit" value="Registrarme"/>
</form>
```

```
var eventHandler = function(event) {
  alert("El plazo de registro ha expirado");
  event.preventDefault();
};

var form = document.getElementById("registro");
form.addEventListener("submit", eventHandler);
```

Hands on!

- Captura los eventos de click en los enlaces, detén la navegación y muestra un *alert* con la dirección de la URL.

```
<fieldset id="contents">
  <legend>Contenido</legend>
  <p>
    Puedes leer la <a href="https://wikipedia.org">Wikipedia</a> o
    visitar <a href="https://w3.org"><i>W3.org</i></a> para aprender más
    sobre el desarrollo de aplicaciones web.
  </p>
</fieldset>
```

Hands on!

- ¿Recuerdas el ejercicio de la lista de tareas? Añádele un formulario para recoger los valores de una nueva tarea y que la añada al final de la tabla al clickar en el botón de envío.
- Mejora la aplicación añadiendo un enlace o botón de borrado a cada una de las tareas.
- Sigue mejorando la aplicación y añade la posibilidad de marcarla como resuelta (es suficiente con añadir un botón y que el texto se muestra ~~tachado~~ cuando esté resuelta.

Lista de tareas

ID	Fecha	Responsable	Descripción
1	03-10-2020	Mikel	Lorem ipsum dolor sit amet, consectetur adipiscing elit.
2	12-08-2020	Unai	Lorem ipsum dolor sit amet, consectetur adipiscing elit.
3	11-12-2020	Ane	Lorem ipsum dolor sit amet, consectetur adipiscing elit.
4	07-06-2020	Nora	Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Hands on!

- o Partiendo del código disponible en <https://codepen.io/jonvadillo/pen/xxwXNRR> implementa el código JS necesario para que la galería de imágenes muestre la imagen seleccionada en cada momento y evite la navegación de los enlaces.



Hands on!

- Mejora el ejercicio anterior para que genera la galería de imágenes partiendo de una lista de objetos como la siguiente:

```
const url = [  
  {  
    id: 1,  
    url : "https://picsum.photos/id/1015/550/400"  
  }, {  
    id: 1,  
    url : "https://picsum.photos/id/1018/550/400"  
  },  
  ...  
];
```



Sources

- [Mozilla MDN](https://developer.mozilla.org/es/): <https://developer.mozilla.org/es/>
- [Modern JavaScript](https://javascript.info/): <https://javascript.info/>