Javascript (DOM #01)

- El modelo de objeto de documento (DOM) es una interfaz de programación para los documentos HTML.
- Facilita una representación estructurada del documento y define de qué manera los programas pueden acceder, al fin de modificar, tanto su estructura, estilo y contenido.
- **Una página web es un documento** . Este documento puede exhibirse en la ventana de un navegador o también como código fuente HTML.
- fuente

> **¿Quieres apoyar a los directores?** 😍
>
> Tienes varias jugosas alternativas:
>
> 1. Suscríbete al canal de Youtube (es gratis) click aquí
> 2. Si estás viendo un video no olvides regalar un 👍 me gusta y comentario 🙏🏽
> 3. También puedes ser miembro del canal de Youtube click aquí
> 4. Puedes adquirir cursos premium en Udemy 👇🏽 👇🏽 👇🏽 ¿Quiéres apoyar a los directores?
>   - Curso de HTML + CSS + Bootstrap 5 + Git y más UDEMY
>   - Curso de React + Firebase UDEMY
>   - Curso Vue.js + Firebase UDEMY

documento

- documento La interfaz Document representa cualquier página web cargada en el navegador y sirve como punto de entrada al contenido de la página (El árbol DOM).

```js
console.log(document);
```

Algunas propiedades:

```js
console.log(document.head);
console.log(document.title);
```

Algunos métodos:

- Documento.getElementsByClassName(Cadena nombreDeClase)
- Documento.getElementsByTagName(Cadena tagName)
- Documento.getElementById(Cadena id)
- Document.querySelector(Selector de cadena)
- Document.querySelectorAll(Selector de cadena)
- Documento.createDocumentFragment()
- Document.createElement(Cadena nombre)

obtenerElementoPorId

- obtenerElementoPorId : Devuelve una referencia al elemento por su ID.

```html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>

<body>

    <h1 id="tituloWeb">Lorem, ipsum dolor.</h1>

    <script src="app.js"></script>
</body>

</html>
```

```js
console.log(document.getElementById("tituloWeb"));
console.log(document.getElementById("tituloWeb").textContent);
console.log(document.getElementById("tituloWeb").innerHTML);
```

¿qué pasa en este caso?

```html
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <script src="app.js"></script> <!-- Qué pasará?? -->
</head>
```

script, DOMContentLoaded, aplazar

- **Contenido DOM cargado** : el navegador HTML está completamente cargado y el árbol DOM está construido, pero es posible que los recursos externos como `<img>` y hojas de estilo aún no se hayan cargado.

```js
document.addEventListener("DOMContentLoaded", () => {
    console.log(document.querySelector("h1"));
});
```

- **aplazar** : El atributo difiere indica al navegador que no espere por el script. En lugar de ello, debe seguir procesando el HTML, construir el DOM. El script se carga “en segundo plano” y se ejecuta cuando el DOM está completo.
- Los scripts con diferir siempre se ejecutan **cuando el DOM está listo** (pero antes del evento DOMContentLoaded).
- defer no funciona igual en todos los navegadores.

```html
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <script src="app.js" defer></script>
</head>
```

Selector de consultas

- Selector de consultas : Devuelve el primer elemento del documento que coincide con el grupo especificado de selectores.

```html
<h1 class="text-primary" id="tituloWeb">Lorem, ipsum dolor.</h1>
```

```js
console.log(document.querySelector("h1"));
console.log(document.querySelector(".text-primary"));
console.log(document.querySelector("#tituloWeb"));
```

Otro ejemplo:

```js
<div class="container">
    <p class="text-danger">Lorem, ipsum dolor.</p>
    <p class="text-danger">Lorem, ipsum dolor.</p>
    <p class="text-danger">Lorem, ipsum dolor.</p>
</div>

<p class="text-danger">parrafo volando</p>
```

```js
// El primer elemento que encuentre
console.log(document.querySelector(".text-danger"));

// Todos
console.log(document.querySelectorAll(".text-danger"));

// Todo lo que esté en "container"
console.log(document.querySelectorAll(".container .text-danger"));
```

querySelector frente a getElementById

- El método querySelector le permite recuperar un elemento mediante una consulta de selector de CSS
- El método getElementById recupera un elemento por su ID DOM.

complejas que se representan fácilmente usando un selector CSS. Si desea seleccionar un elemento por su ID, usar getElementById es una buena opción.

- fuente
- A menudo será necesario realizar selecciones más complejas en su HTML, y ahí es donde querySelector puede ser más útil; Usarlo de manera constante también puede hacer que su código sea más fácil de leer para otros codificadores.
- En otras palabras, el principal beneficio de usar querySelector o querySelectorAll es que podemos seleccionar elementos usando selectores CSS, lo que nos da **una forma uniforme de manejar la selección de elementos** , y eso lo convierte en una forma preferida de seleccionar elementos para muchos desarrolladores.
- Si usa una herramienta como Babel para admitir navegadores más antiguos, entonces puede ser irrelevante, ya que las funciones más nuevas se pueden convertir a código compatible con versiones con anteriores cuando compila su script.
- 800.000 selecciones por segundo, querySelector es aprox. 6% más lento.
- fuente

```html
<div id="container">
    <p class="text-danger">Lorem, ipsum dolor.</p>
    <p class="text-danger">Lorem, ipsum dolor.</p>
    <p class="text-danger">Lorem, ipsum dolor.</p>
</div>

<p>parrafo volando</p>
```

```js
console.log(document.querySelectorAll("div p"));
```

elemento

- elemento : eventos disponibles para los elementos HTML

Algunas propiedades:

```js
const h1 = document.querySelector("#tituloWeb");

console.log(h1.className);
console.log(h1.id);
```

```
console.log(h1.textContent);

h1.textContent = "nuevo texto";
h1.style.backgroundColor = "red";
h1.style.color = "white";
```

Algunos métodos:

- addEventListener: Registre un controlador de evento para un tipo de evento específico en un elemento.
- appendChild: Inserta un nodo así como el último nodo hijo de este elemento.
- hasAttributes: Verifica si el elemento tiene o no algún atributo.

Eventos

En JavaScript, la interacción con el usuario se consigue mediante la captura de los eventos que éste produce. Un evento es una acción del usuario ante la cual puede realizarse algún proceso (por ejemplo, el cambio del valor de un formulario, o la pulsación de un enlace).

agregarListener de eventos

- agregarListener de eventos : Registre un evento a un objeto en específico.
- **El Objeto específico puede ser un elemento simple** en un archivo, el mismo documento, una ventana o un XMLHttpRequest.
- Eventos estándar

```
target.addEventListener(tipo, listener);
```
js

- tipo: tipo de evento a escuchar.
- oyente: El objeto que recibe una notificación cuando ocurre un evento del tipo especificado. Debe ser un objeto implementando la interfaz EventListener o **solo una función en JavaScript** .

hacer clic

```js
const boton = document.querySelector("#boton");
const parrafo = document.querySelector("#parrafo");

boton.addEventListener("click", () => {
    parrafo.textContent = "Nuevo texto desde evento";
});
```

Práctica:

- Intenta hacer esto

```html
<!DOCTYPE html>
<html lang="es">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Seleccionar Color</title>
    <link
      crossorigin="anonymous"
      href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
      integrity="sha384-1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6
      rel="stylesheet"
    >
</head>

<body>

    <div class="container mt-5 text-center">
        <label
          class="form-label"
          for="inputColor"
        >Color picker</label>
        <input
          id="inputColor"
          class="form-control form-control-color mb-3 w-100"
          title="Seleccione un color"
```

```
        <button
          id="boton"
          class="btn btn-primary w-100"
        >Visualizar</button>
    </div>
    <div class="container mt-5">

        <p
          id="textoHexa"
          class="lead text-center"
        >#563d7c</p>
        <div
          id="cardColor"
          class="card text-center p-5"
          style="background-color: #563d7c;"
        ></div>
    </div>

    <script src="app.js"></script>
</body>

</html>
```

```
const inputColor = document.getElementById("inputColor");
const boton = document.getElementById("boton");
const textoHexa = document.getElementById("textoHexa");
const cardColor = document.getElementById("cardColor");

console.log(inputColor.value);

boton.addEventListener("click", () => {
    console.log(inputColor.value);
    textoHexa.textContent = inputColor.value;
    cardColor.style.backgroundColor = inputColor.value;
});
```

bono opcional

- Copiar color en el portapapeles:

```
    textoHexa.textContent = inputColor.value;
    cardColor.style.backgroundColor = inputColor.value;
    navigator.clipboard
        .writeText(inputColor.value)
        .then(() => console.log("texto copiado"))
        .catch((e) => console.log(e));
});
```

crearElemento

- **crearElemento** : El método `document.createElement()` crea un elemento HTML especificado por su tagName.

Crear un `<li>`

```js
const li = document.createElement("li");
li.textContent = "item desde javascript";
console.log(li)
```

añadirNiño

- **añadirNiño** : Agrega un nuevo nodo al final de la lista de un elemento hijo de un elemento padre especificado.

```html
<ul id="listaDinamica">
    <li>Elemento estático</li>
</ul>
```

```js
// elemento donde vamos a incorporar los <li>
const listaDinamica = document.querySelector("#listaDinamica");

// Creamos el <li>
const li = document.createElement("li");

// // Agregamos texto al <li>
li.textContent = "item desde javascript";
```

```
listaDinamica.appendChild(li); // ¿qué pasó aquí?
```

> **ADVERTENCIA**
>
> - Si el **niño hace una referencia a un nodo existente en el documento** , el método appendChild se mueve de su posición actual a su nueva posición.
> - Esto significa que el nudo no puede estar en dos puntos del documento de manera simultánea.
> - Así que si el nudo ya contiene un padre, primero es eliminado, y después se añade a la nueva posición.
> - Se puede usar **Node.cloneNode** para hacer una copia del nudo antes de agregarlo debajo de su nuevo elemento padre.

No recomendado:

```js
const listaDinamica = document.querySelector("#listaDinamica");

const arrayElementos = ["Perú", "Bolivia", "Colombia"];

arrayElementos.forEach((pais) => {
    const li = document.createElement("li");
    li.textContent = pais;
    listaDinamica.appendChild(li);
});
```

No recomendado:

```js
arrayElementos.forEach((pais) => {
    listaDinamica.innerHTML += `
    <li>${pais}</li>
    `;
});
```

> **Reflujo**

actualización en un sitio interactivo.

- reflujo del navegador

Consideración de seguridad

- innerHTML LEER 🙏

Fragmento

- nuevo DocumentFragment()
- crearFragmentoDeDocumento()
- La interfaz DocumentFragment representa un objeto de documento mínimo que no tiene padre.
- Se utiliza como una versión ligera de Documento que almacena un segmento de una estructura de documento compuesta de nodos como un documento estándar.
- La gran diferencia se debe al hecho de que **el fragmento de documento no forma parte de la estructura de árbol del documento activo.**
- Los cambios realizados en el fragmento no afectan el documento (incluso en reflujo) ni inciden en el rendimiento cuando se realizan cambios.

```js
const listaDinamica = document.querySelector("#listaDinamica");

const arrayElementos = ["Perú", "Bolivia", "Colombia"];

const fragment = document.createDocumentFragment(); // new DocumentFragment()

arrayElementos.forEach((pais) => {
    const li = document.createElement("li");
    li.textContent = pais;
    fragment.appendChild(li);
});

listaDinamica.appendChild(fragment);
```

insertar antes

```js
parentNode.insertBefore(newNode, referenceNode);
```

```js
arrayElementos.forEach((pais) => {
    const newNode = document.createElement("li");
    newNode.textContent = pais;

    // Nos devuelve el primer elemento
    const referenceNode = fragment.firstChild;

    // En caso de que no exista un nodo hijo tirará null
    console.log("primer newNode", referenceNode);

    // fragment.insertBefore(newNode, referenceNode);
    // Si "referenceNode" es null, el newNode se insertará al final de la lista.
    fragment.insertBefore(newNode, referenceNode);
});
```

Práctica createElement

Supongamos que necesitamos incorporar de forma dinámica este elemento:

```html
<li class="list">
  <b>País: </b> <span class="text-primary">aquí va el país</span>
</li>
```

```js
const listaDinamica = document.querySelector("#listaDinamica");

const arrayElementos = ["Perú", "Bolivia", "Colombia"];

const fragment = new DocumentFragment();

arrayElementos.forEach((pais) => {
    const li = document.createElement("li");
    li.className = "list";

    const bold = document.createElement("b");
    bold.textContent = "País: ";
```

```
    span.textContent = pais;

    li.appendChild(bold);
    li.appendChild(span);
    fragment.appendChild(li);
});

listaDinamica.appendChild(fragment);
```

HTML interno

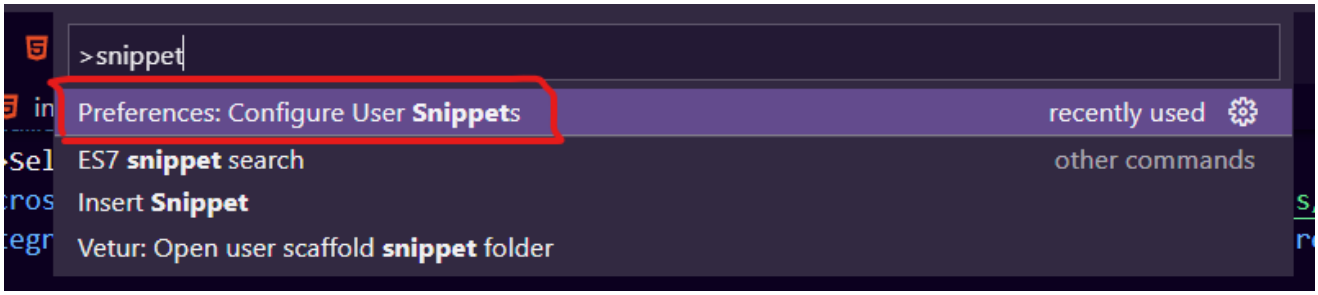
```
let template = "";

arrayElementos.forEach((pais) => {
    template += `
    <li class="list">
        <b>País: </b> <span class="text-primary">${pais}</span>
    </li>
    `;
});

listaDinamica.innerHTML = template;
```

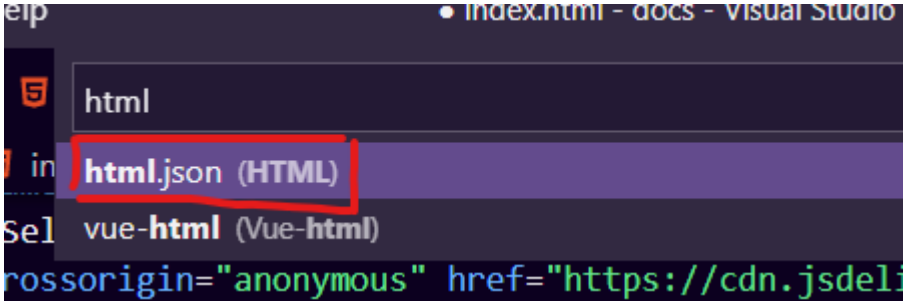
> **innerHTML frente a createElement**
>
> Ojo que aquí estamos reemplazando fragment por let template, por ende hace un efecto parecido y minimizamos el reflow, ya que solo una vez que tenemos nuestro templateString listo, lo incorporamos al HTML.
>
> Aquí un texto completo: **internalHTML vs createElement**
>
> Pero la batalla es brutal jajaja aquí algunas opiniones al respecto:
>
> - 1
> - 2
> - 3
> - 4
>
> ¡Alerta de spoiler! No utilizaremos este método.

Crear fragmento

- retazo

```
ctrl + shift + p
```






```json
"Template in HTML": {
    "prefix": "template",
    "body": ["<template>$1</template>"],
    "description": "Agrega el template en html"
}
```

plantilla

- plantilla : es un mecanismo para mantener el contenido HTML del lado del cliente que no se renderiza cuando se carga una página, pero que posteriormente puede ser instanciado durante el tiempo de ejecución empleando JavaScript.
- Piense en la plantilla como un fragmento de contenido que está siendo almacenado para un uso posterior en el documento.
- El analizador procesa el contenido del elemento `<template>` durante la carga de la página, pero sólo lo hace para asegurar que esos contenidos son válidos; Sin embargo, estos contenidos del elemento no se renderizan.

HTML, CSS y Bootstrap 5 (bluuweb)

```html
<ul id="listaDinamica"></ul>

<template id="liTemplate">
    <li class="list">
        <b>País: </b> <span class="text-primary"></span>
    </li>
</template>

<script src="app.js"></script>
```

```js
const listaDinamica = document.querySelector("#listaDinamica");

const liTemplate = document.querySelector("#liTemplate");
// es aconsejable clonar
const clone = liTemplate.content.cloneNode(true);

clone.querySelector("span").textContent = "Perú";

listaDinamica.appendChild(clone);
```

Fragmento + Plantilla

```js
const listaDinamica = document.querySelector("#listaDinamica");

const arrayElementos = ["Perú", "Bolivia", "Colombia"];

const fragment = document.createDocumentFragment();
const liTemplate = document.querySelector("#liTemplate");

arrayElementos.forEach((pais) => {
    const clone = liTemplate.content.cloneNode(true);
    clone.querySelector("span").textContent = pais;
    fragment.appendChild(clone);
});

listaDinamica.appendChild(fragment);
```

CONSEJO

> **cuenta que el uso directo del valor del contenido** podría provocar un comportamiento inesperado; consulte la sección **Evitar el error de DocumentFragment a continuación** .

```js
const listaDinamica = document.querySelector("#listaDinamica");
const arrayElementos = ["Perú", "Bolivia", "Colombia"];
const fragment = document.createDocumentFragment();
const liTemplate = document.querySelector("#liTemplate");

const clickPais = (e) => console.log("evento", e.target);

arrayElementos.forEach((pais) => {
    const clone = liTemplate.content.firstElementChild.cloneNode(true);
    clone.querySelector("span").textContent = pais;

    clone.addEventListener("click", clickPais);

    fragment.appendChild(clone);
});

listaDinamica.appendChild(fragment);
```

```js
const clickPais = (e) => e.target.append(" click");
```

Plantilla de práctica

- ver ejemplo

```html
<!DOCTYPE html>
<html lang="es">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Carrito Objeto</title>
    <link crossorigin="anonymous" href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/
        integrity="sha384-1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2Qv
```

HTML, CSS y Bootstrap 5 (bluuweb)

```
</body>

    <main class="container mt-5">
        <div class="row text-center">
            <article class="col-sm-4 mb-3">
                <div class="card">
                    <div class="card-body">
                        <h5 class="card-title">Frutilla 🍓</h5>
                        <button class="btn btn-primary" data-fruta="frutilla">Agregar
                    </div>
                </div>
            </article>
            <article class="col-sm-4 mb-3">
                <div class="card">
                    <div class="card-body">
                        <h5 class="card-title">Banana 🍌</h5>
                        <button class="btn btn-primary" data-fruta="banana">Agregar</
                    </div>
                </div>
            </article>
            <article class="col-sm-4 mb-3">
                <div class="card">
                    <div class="card-body">
                        <h5 class="card-title">Manzana 🍏</h5>
                        <button class="btn btn-primary" data-fruta="manzana">Agregar
                    </div>
                </div>
            </article>
        </div>
    </main>

    <section class="container mt-3">
        <ul class="list-group" id="carrito">
            <!-- <li class="list-group-item d-flex justify-content-between align-iter
                <span class="lead">A list item</span>
                <span class="badge bg-primary rounded-pill">14</span>
            </li> -->
        </ul>
    </section>

    <template id="template">
        <li class="list-group-item d-flex justify-content-between align-items-center'
            <span class="lead">A list item</span>
            <span class="badge bg-primary rounded-pill">14</span>
```

HTML, CSS y Bootstrap 5 (bluuweb)

```
    <script src="app.js"></script>
</body>

</html>
```

```
const carrito = document.querySelector("#carrito");
const template = document.querySelector("#template");
const fragment = document.createDocumentFragment();
const agregar = document.querySelectorAll(".card button");

const carritoObjeto = {};

const agregarCarrito = (e) => {
    // console.log(e.target.dataset);
    // console.log(e.target.dataset.fruta);

    const producto = {
        titulo: e.target.dataset.fruta,
        id: e.target.dataset.fruta,
        cantidad: 1,
    };

    if (carritoObjeto.hasOwnProperty(producto.id)) {
        producto.cantidad = carritoObjeto[producto.id].cantidad + 1;
    }

    carritoObjeto[producto.id] = producto;

    pintarCarrito();
};

agregar.forEach((boton) => boton.addEventListener("click", agregarCarrito));

const pintarCarrito = () => {
    carrito.textContent = "";

    Object.values(carritoObjeto).forEach((item) => {
        const clone = template.content.cloneNode(true);
        clone.querySelector(".lead").textContent = item.titulo;
        clone.querySelector(".rounded-pill").textContent = item.cantidad;
        fragment.appendChild(clone);
```

```
});
```

Última actualización: 27/11/2021, 13:07:12

← Javascript (Objetos)

Javascript (métodos de matriz) →