<codoacodo/>

Agencia de
Aprendizaje
a lo largo
de la vida

Desarrollo Fullstack

BA

Les damos la bienvenida

Vamos a comenzar a grabar la clase

Agencia de Aprendizaje a lo largo de la vida

Buenos Aires Ciudad Educación

Clase 16

Clase 17

Clase 18

Javascript Arrays

- Arrays
- Métodos de los Arrays

Javascript Para la Web

- **Dom**
- **Manejo de eventos**

Asincronismo en Javascript

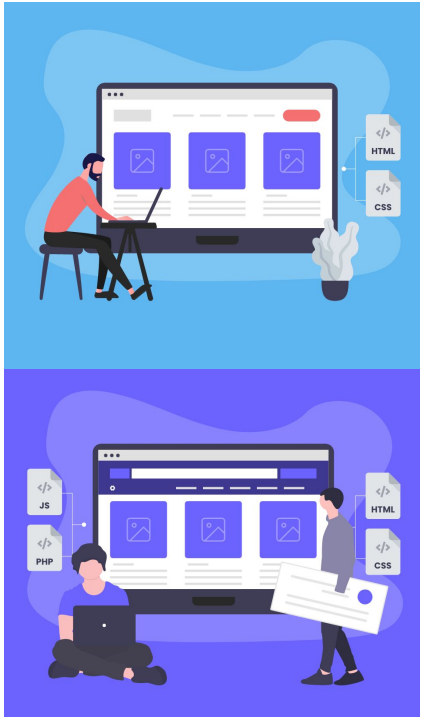
- ¿Qué es? Call
- Stack Event
- Loop
- Promesas
- Async/Await

JAVASCRIPT

DOM y Eventos

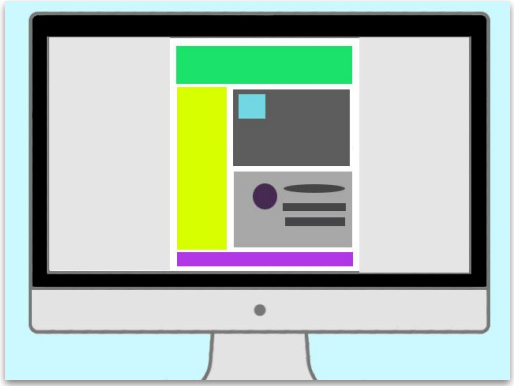
Introducción al DOM

- Imaginemos tener un documento impreso en papel. Este documento es como una página web, **la página en sí es estática.**
- Ahora, el **DOM** sería como una **versión interactiva de ese documento impreso**.
- El **DOM convierte** un documento web **estático** en una **estructura que se puede manipular dinámicamente con JavaScript** u otros lenguajes de programación.
- **Es la interfaz** que permite a los desarrolladores **interactuar y modificar la estructura y contenido** de una página web **de manera dinámica**.



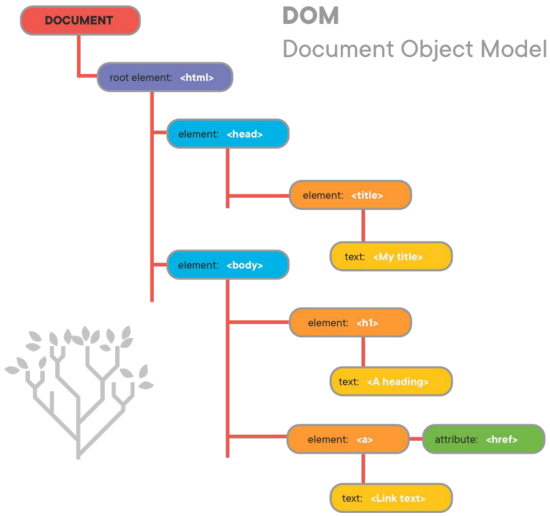
¿Qué es el DOM?

- El DOM, o **Modelo de Objetos del Documento** en español, es una **representación jerárquica *(estructura en forma de árbol)* de la estructura de un documento HTML**.
- Para entender qué es el DOM **debe quedar claro que cada palabra, párrafo, imagen, etc**. en un documento **html tienen etiquetas asignadas.**



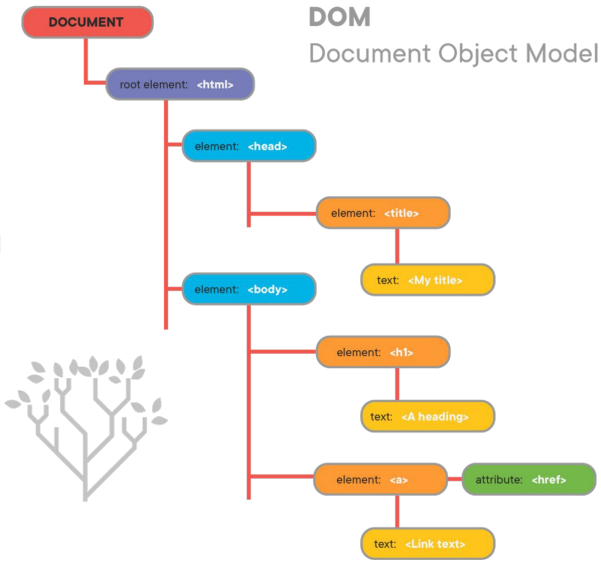
¿Qué es el DOM?

- Dicho lo anterior, **el DOM representa la estructura del documento como un árbol de objetos**, donde **cada nodo** en el árbol **corresponde a un elemento, atributo o** parte del **contenido del documento.**
- Es una **representación estructurada y programática** del contenido del documento que **se puede manipular con código**.



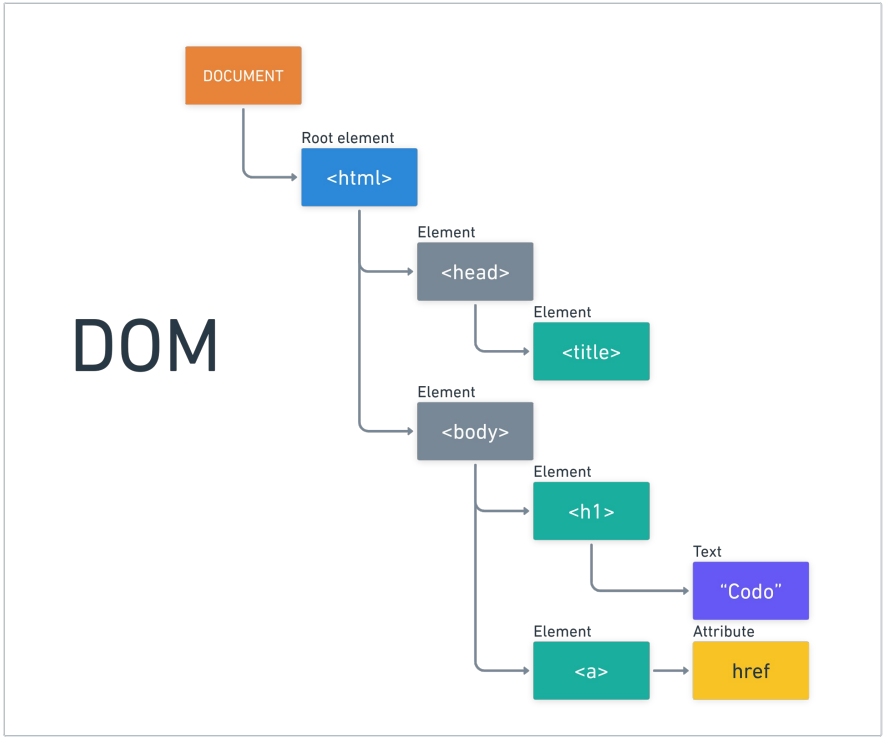
Nodos en el DOM

- Se dijo que **el DOM es un árbol de elementos**, donde **cada elemento es un nodo.**
- **Cada nodo** en el árbol **corresponde a un elemento, atributo o** parte del **contenido del documento.**



Nodos en el DOM

- El **DOM es una interfaz** de plataforma que proporciona un conjunto estándar de objetos para representar documentos HTML.
- Una **interfaz se puede definir como un punto de interacción entre dos sistemas o entidades**, por ejemplo el usuario realiza una acción y recibe información.



Conceptos fundamentales de HTML para entender el DOM

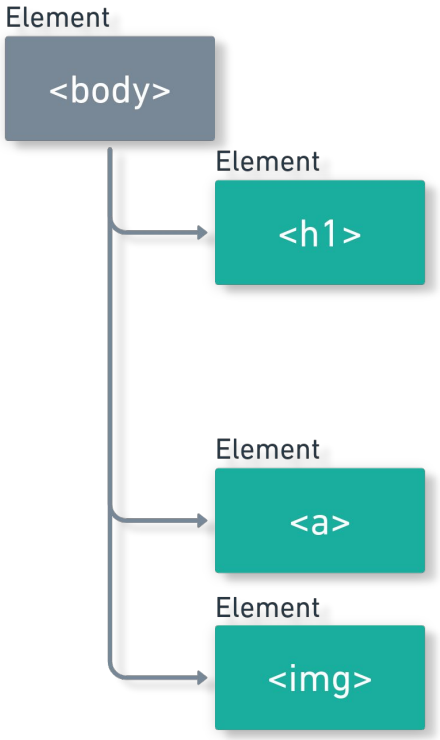
(repaso)

Agencia de Aprendizaje a lo largo de la vida

Buenos Aires Ciudad
Educación

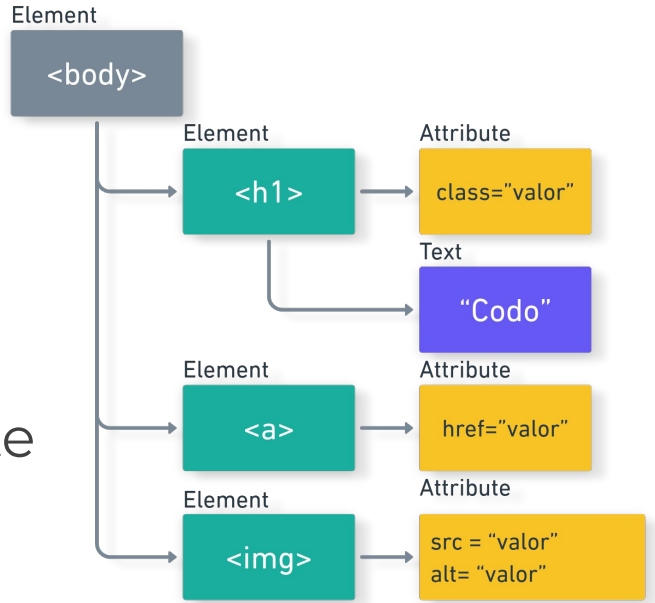
Elementos HTML:

- Los elementos HTML son las **partes fundamentales** de un documento HTML.
- Cada elemento comienza con una **etiqueta de apertura**, contiene el contenido del elemento y termina con una **etiqueta de cierre**.



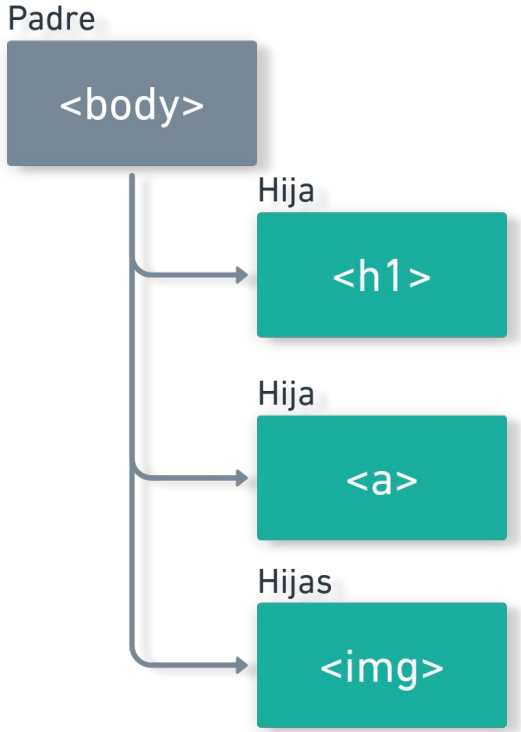
Atributos HTML:

- Atributos HTML: Los **atributos** proporcionan **información adicional** sobre el elemento.
- Los atributos **se especifican en la etiqueta de apertura** y generalmente tienen un **nombre y un valor**.



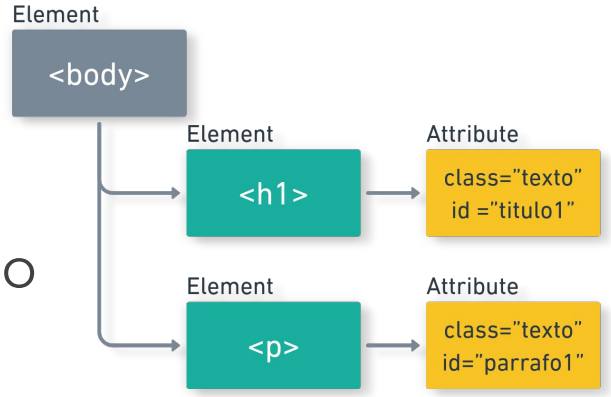
Padre e Hija

- En HTML, los elementos pueden estar anidados, creando una estructura jerárquica.
- El elemento que contiene otro se llama "padre", y el contenido se convierte en "hija" del elemento padre.



Atributos globales id, class

- Estos atributos son **comúnmente utilizados para identificar y seleccionar elementos en el DOM.**
- El atributo **id debe ser único** en todo el documento, mientras que el atributo **class puede ser compartido por varios elementos**.



Tipos de nodos en DOM

Agencia de Aprendizaje a lo largo de la vida

Buenos Aires Ciudad
Educación

Tipos de nodos

- **Nodo de Documento - Document:** Representa todo el documento. Es el **nodo raíz del árbol DOM**.
- **Nodo de Atributo - Attr:** Representa un atributo de un elemento. Cada nodo de atributo está asociado con un nodo de elemento y contiene información sobre ese atributo específico.

Tipos de nodos

- **Nodo de Comentario - Comment:**
  Representa un **comentario en el código HTML.** Los comentarios en HTML están encerrados entre <!-- y -->.
- **Nodo de Texto - Text:**
  Representa el contenido de texto dentro de un elemento. Por ejemplo, si tienes un párrafo Este es un texto , el contenido "Este es un texto" sería un nodo de texto.

Tipos de nodos

- **Nodo de Elemento - Element:**
  Representa un elemento HTML o XML y forma la estructura principal del DOM. Por ejemplo, un nodo de elemento podría ser un <p> (párrafo) o un (división) en HTML.
  Puede tener tanto **nodos hijos** como atributos.

Representación del DOM

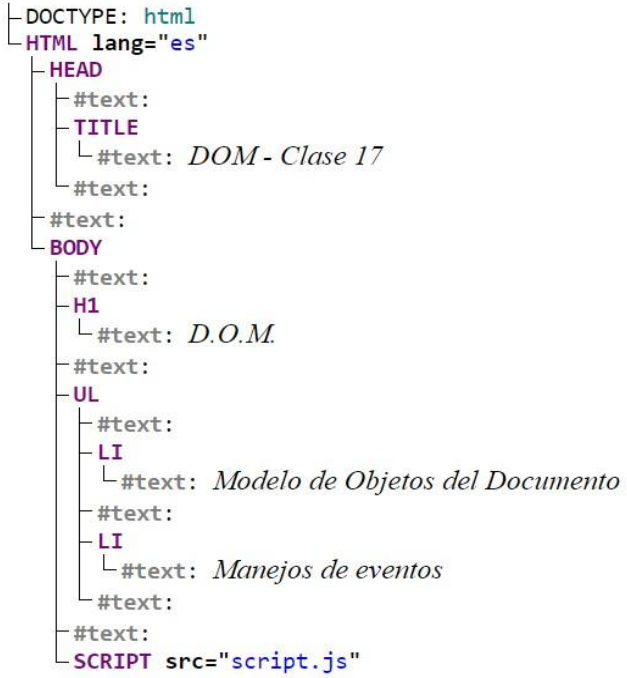
index.html

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

<head>
  <title>DOM - Clase 17</title>
</head>

<body>
  <h1>D.O.M.</h1>
  <ul>
    <li>Modelo de Objetos del Documento</li>
    <li>Manejos de eventos</li>
  </ul>
  <script src="script.js"></script>
</body>

</html>
```



Acceso al DOM

(Interface programática)

Agencia de Aprendizaje a lo largo de la vida

Buenos Aires Ciudad
Educación

Acceso al DOM

- El objeto **document representa la estructura del documento HTML actual**. Cada elemento HTML, atributo, texto y otros componentes de la página son representados como nodos dentro del árbol del DOM (Document Object Model).
- Este objeto **proporciona una interfaz programática** para interactuar con el documento.
- Al **acceder al objeto document, se puede navegar por este árbol**, interactuar con sus nodos y realizar manipulaciones dinámicas en tiempo de ejecución**.**

Acceso al DOM: Interfaz para Manipulación Dinámica:

- El objeto **document** expone una variedad de **métodos y propiedades útiles.**
- Puedes **agregar, eliminar o modificar elementos y atributos HTML**, cambiar estilos, manejar eventos y más.
- Los más usuales se exponen en la siguiente tabla.

DOM: Métodos de acceso a elementos.

Función	Descripción	Sintaxis
`getElementById()`	Este método devuelve un elemento del DOM que tiene el ID especificado.	`const elemento = document.getElementById("miElemento");`
`getElementsByClassName()`	Retorna una colección de elementos que tienen una clase específica.	`const elementos = document.getElementsByClassName("miClase");`
`getElementsByTagName()`	Retorna una colección de elementos con un nombre de etiqueta HTML específico.	`const elementos = document.getElementsByTagName("p");`
`querySelector()`	Retorna el primer elemento que coincide con un selector CSS especificado.	`const elemento = document.querySelector("#miId");`

Desafío I

```
/**
DESAFIO 1
Manipulamos objetos del DOM.
1- Declaramos una variable de tipo const inicializada mediante el método getElementById.
Obtenemos el elemento con el id "title" del documento HTML y luego imprimimos el
contenido de la variable por consola, por ultimo imprimimos solo el contenido de texto
del elemento.
2- Declaramos una variable de tipo const inicializada mediante el método
getElementByClassName. Obtenemos los elementos con la clase "items" del documento HTML y
luego imprimimos el contenido de la variable por consola.
*/
```

replit

https://replit.com/@Ferlucena/DOM#script.js

DOM: Métodos para crear y agregar elementos.

- Estos métodos **permiten dinámicamente crear nuevos elementos y agregarlos al DOM** durante la ejecución del código JavaScript.
- Pueden ser utilizados para **construir o modificar la estructura** de la página web de manera dinámica.

Modificación del DOM: Métodos para Crear Elementos.

Función	Descripción	Sintaxis
`createElement()`	Crea un nuevo elemento HTML con el nombre de la etiqueta especificada.	`var nuevoElemento = document.createElement("div");`
`createTextNode()`	Crea un nuevo nodo de texto con el contenido especificado.	`var nuevoTexto = document.createTextNode("Contenido del nuevo elemento");`

Modificación del DOM: Métodos para Agregar Elementos.

Función	Descripción	Sintaxis
`appendChild()`	Agrega un nodo hijo al final de la lista de hijos de un nodo padre.	`padre.appendChild(nuevoElemento);`
`innerText()`	Es utilizada para obtener o establecer el contenido de texto de un elemento, excluyendo las etiquetas HTML.	`var miElemento = document.getElementById("miElemento");` `miElemento.innerText = "Texto modificado";`
`innerHTML()`	Se utiliza para obtener o establecer el contenido HTML de un elemento, incluyendo todas las etiquetas HTML.	`var miElemento = document.getElementById("miElemento");` `miElemento.innerHTML = "<p>Nuevo contenido HTML</p>";`
`document.write()`	Es un método que se utiliza para escribir directamente en el documento HTML durante la carga inicial de la página. **Nota:** Escribir después de que la página ha cargado **puede causar problemas** y se recomienda su **uso con precaución.**	`document.write("Este texto se escribirá en el documento HTML");`

Desafío II

```
/**
DESAFIO 2
Paso 1: Seleccionar el contenedor
Paso 2: Crear un nuevo párrafo y agregarlo al contenedor
Paso 3: Crear un nodo de texto y asignarlo al nuevo párrafo
Paso 4: Agregar contenido HTML al nuevo párrafo usando innerHTML
Paso 5: Utilizar document.write para agregar texto directamente al documento
*/
```

replit

https://replit.com/@Ferlucena/DOM#js/crearAgregarElementosDOM.js

Eventos en DOM

Agencia de Aprendizaje a lo largo de la vida

Buenos Aires Ciudad
Educación

Eventos

- Los eventos en el DOM (Document Object Model) son **interacciones o sucesos que ocurren en una página web** y que **pueden ser detectados** y **manejados mediante JavaScript**.
- Los eventos permiten que una página web sea interactiva, ya que **responden a las acciones del usuario** o a **cambios en el estado del documento**.

Eventos

(Conceptos básicos)

Agencia de Aprendizaje a lo largo de la vida

Buenos Aires Ciudad
Educación

Eventos

- **Tipo de Eventos**

Los eventos pueden ser de **diversos tipos**, como **clics** de ratón, **pulsaciones de teclas**, cambios en el tamaño de la ventana, envío de formularios, etc.

- **Elementos y Eventos**

Cada **elemento en el DOM** puede estar **asociado con uno o más tipos de eventos**. Por ejemplo, un botón puede responder al evento de clic, mientras que un campo de entrada puede responder a eventos de teclado.

- **Manejadores de Eventos (Event Handlers)**

Los manejadores de eventos son funciones de JavaScript que se ejecutan cuando ocurre un evento específico. Estas funciones están vinculadas a elementos particulares y se activan cuando se produce el evento correspondiente.

Event Handlers: manejadores de eventos

La vinculación de **elementos** del DOM con los **manejadores eventos** puede realizarse de dos maneras:

- **A través de HTML:**

Vinculando eventos directamente en las etiquetas HTML **utilizando atributos de eventos** como **onclick, onmouseover**, etc. Si bien esta manera puede parecer más cómoda, **se desaconseja su uso por considerarse una mala práctica.**

- **A través de JavaScript:**

Es mejor práctica vincular eventos mediante JavaScript utilizando métodos como **addEventListener().**

Manejadores de eventos (Event Handlers)

Sintaxis a través de atributos HTML:

- `<button onclick="miFuncion()">Haz clic</button>`

Sintaxis a través de JS:

- `let miElemento = document.getElementById("miElemento");`
- `miElemento.addEventListener("click", function() {`
- `  // Código a ejecutar cuando se hace clic en el elemento`
- `});`

En este caso, "click" es el tipo de evento al que estamos prestando atención. Cuando el elemento con el id "miElemento" recibe un clic, se ejecutará la función proporcionada como manejador de eventos.

La expresión `miElemento.addEventListener("click", function(){});` se utiliza para decirle al nodo miElemento que esté atento al evento de tipo 'click' y, cuando ocurra, ejecute la función anonima function().

Event Handlers: manejadores de propiedad

Algunos eventos específicos también pueden tener propiedades en el objeto del elemento que actúan Por ejemplo, element.onclick o element.onchange.

Este enfoque tiene algunas limitaciones y generalmente no se recomienda por varias razones.

- **Solo se puede asignar un manejador de eventos para cada tipo de evento.** Si se intentara asignar un segundo manejador, este sobrescribirá al primero.
- **Solo funciona para eventos directos** como click, cambios etc, no es aplicable a eventos mas complejos como propagación, captura etc.

Manejadores de propiedad

Sintaxis a través de JS:

```
var miElemento = document.getElementById("miElemento");

miElemento.onclick = function() {
  // Código a ejecutar cuando se hace clic en el elemento
};

// Este segundo manejador sobrescribe al primero
miElemento.onclick = function() {
  // Nuevo código a ejecutar
};
```

Event types más comunes

Eventos de ratón	
Función	**Descripción**
`click`	Se activa cuando se hace clic con el ratón.
`dblclick`	Se activa cuando se hace doble clic con el ratón.
`mousedown`	Se activa cuando se presiona un botón del ratón.
`mouseup`	Se activa cuando se libera un botón del ratón.
`mousemove`	Se activa cuando se mueve el ratón sobre un elemento.

Event types más comunes

Eventos de teclado	
Función	**Descripción**
`keydown`	Se activa cuando una tecla del teclado es presionada.
`keyup`	Se activa cuando una tecla del teclado es liberada.
`keypress`	Se activa cuando una tecla que produce un carácter imprimible es presionada.

Event types más comunes

Eventos de Formulario	
Función	**Descripción**
`submit`	Se activa cuando se envía un formulario.
`change`	Se activa cuando cambia el valor de un elemento de formulario (input, select, etc.).
`mousefocus`	Se activa cuando un elemento de formulario recibe foco.
`blur`	Se activa cuando un elemento de formulario pierde el foco.

Event types más comunes

Eventos de Ventana	
Función	**Descripción**
`load`	Se activa cuando se completa la carga de la página.
`unload`	Se activa cuando la página está siendo descargada (no es comúnmente utilizado).
`resize`	Se activa cuando se cambia el tamaño de la ventana del navegador.
`scroll`	Se activa cuando se realiza un desplazamiento en la página.

Event types más comunes

Eventos de Documento	
Función	**Descripción**
`DOMContentLoaded`	Se activa cuando el DOM ha sido completamente cargado y analizado, sin esperar a que se carguen imágenes y estilos.

Event types más comunes

Eventos de Animación y Transición	
Función	**Descripción**
`animationstart`, `animationend`, `animationiteration`	Se activan respectivamente al comenzar, finalizar y al repetirse una animación.
`transitionend`	Se activa cuando una transición CSS ha finalizado.

Event types más comunes

Eventos de Arrastrar y Soltar	
Función	**Descripción**
`dragstart`	Este evento se activa cuando un elemento comienza a ser arrastrado.
`dragend`	Se activa cuando el usuario ha terminado de arrastrar un elemento.
`dragenter`	Se activa cuando un elemento arrastrado entra en la zona de destino.
`dragleave`	Se activa cuando un elemento arrastrado sale de la zona de destino.
`dragover`	Este evento se activa continuamente mientras un elemento arrastrado se encuentra sobre la zona de destino.
`drop`	Se activa cuando un elemento arrastrado se suelta en la zona de destino.

Desafío III

```
DESAFIO 3
En el siguiente desafío crearemos un documento html, un archivo css y un archivo
javascript.
1- El html tendrá un boton y dos input en ellos probaremos los eventos click, change y
keydown.
2- Seguidamente tendrá un bloque div con el que se probarán los eventos mouseover y
mouseout. En este punto incluiremos un evento onclick que muestre un alert al hacer click.
3- Por último incluiremos una sección de formulario con usuario y clave, trabajaremos con
el evento submit.
*/
```

replit

https://replit.com/@Ferlucena/DOM-desafio3#index.html

Herramientas que utilizamos en clases

VSCode+plugins

No te olvides de dar el presente

Recordá:

- Revisar la Cartelera de Novedades.
- Hacer tus consultas en el Foro.
- Realizá los ejercicios obligatorios.

Todo en el Aula Virtual.