# JS en la Web: Manipulación del DOM con JavaScript

## Conclusión 1: Conociendo DOM

* Recorrer el árbol del DOM
* Utilizar querySelector para seleccionar los elementos dentro del árbol de DOM
* Funcionamiento de la estructura del DOM

## Conclusión 2: Comportamiento del formulario

* Utilizar data-attributes
* Utilizar el método addEventListener para escuchar eventos de un elemento
* Prevenir el comportamiento patrón del formulario

## Conclusión 3: Agregar una tarea a la lista

* Utilizar template strings
* Colocar un elemento hijo dentro del elemento padre utilizando el método appendChild
* Crear elementos utilizando el método createElement

|  |
| --- |
|  |

## Conclusión 4: Concluir una tarea

* Agregar la clase CSS utilizando el método toggle
* Utilizar el atributo parentElement para subir un elemento en el árbol del DOM
* Encontrar el objetivo del evento utilizando la propiedad target
* Utilizar IIFE

## Conclusión 5: Eliminar una tarea

* Utilizar import/export
* Remover elementos del DOM con el método remove

## Anexos:

|  |
| --- |
|  |
| |  | | --- | |  | |

* <https://fontawesome.com> librería de iconos
* En la consola se puede acceder a path de un arrow function que capture eventos, para así ver la composición del DOM
* En la consola se puede acceder a target para así ver el tipo de elemto que com´pone el evento

## Código:

* Document
* .querySelector()
* .addEventListener(acción,función(){})---Evento / Elemento que recibirá el evento / Lo que pasará cuando el evento sea realizado
* .value
* .preventDefault()
* .createElement()
* .classList.add("card")
* .innerHTML
* .appendChild(task)
* .remove
* .innerText
* .appendChild(checkComplete());
* Return
* event.target;
* .classList.toggle("fas");
* .target.parentElement;
* .remove();
* export default
* import

## Vocabulario:

* DOM: Es el árbol compuesto por la sintaxis estructural del sistema. Es el Modelo de Objetivos del Documemnto. El DOM se compone por nodos (tags de sintaxis del html)
* Data attribute: son el nombre que te ayudara a llamar al DOM en JS con querySelector sin recurrir a clases, tags o etc. HTML5 (en-US) está diseñado de forma tal que sea fácil extender los datos asociados a un elemento en particular sin necesidad de que tengan un significado definido. Los atributos data-\* permiten almacenar información adicional sobre un elemento HTML cualquiera sin tener que recurrir a artilugios tales como la utilización de atributos no estándar, propiedades adicionales en el DOM o Node.setUserData(). Se llaman “[]”
* Arroiw functions o funciones anónimas
* Inmediately invoke function expresión IIFE = funcione que se declaran apenas se ejecutan
* Encapsular código: es cuando se derivan secciones de código con objetivos específicos, los cuales se separan para evitar el exceso de info. Se conectan las encapsulaciones en js con expor default e input y declarando la llamada de js en HTML con type=”module”. Crear una carpeta para organizar los códigos, darles nombres que tengan sentido para los archivos, funciones y componentes, utilizar import/export para crear comunicación entre los archivos y garantizar el encapsulamiento.