[Escuela de código]

Apuntes de clase:

Manipulación del DOM

Tabla de contenido

Manipulación con JS

JavaScript y Web-Assembly son los únicos lenguajes de programación que pueden correr de manera nativa en el navegador, por lo tanto estos necesitan poder comunicarse con el contenido que este presenta. El contenido podría ser el HTML o el CSS de una página. Para hacer esto, JavaScript, dispone de muchas herramientas, entre las más comunes e importantes, podemos encontrar las APIs del Navegador y el DOM.

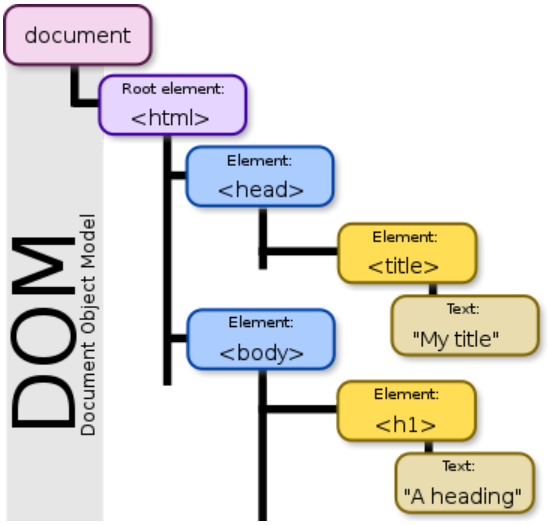
DOM

"DOM" viene de las siglas en inglés Document Object Model, algo así como el Objeto del Modelo del Documento. No es nada más ni nada menos que una descripción del Árbol que rige el HTML y CSS de una página WEB.

¿Qué podemos hacer con el DOM?

Seguramente te estés preguntando qué podemos hacer con el DOM y JS, probablemente llegados a este punto, estés familiarizado con HTML y CSS. Seguramente alguna vez te ha ocurrido que has estado haciendo algo con HTML y CSS, y has pensado: "Cómo me gustaría hacer que esto se comportara de esta manera cuando aprieto este botón" o "Como me gustaría que este menú haga 'esto' ", o muchas otras cosas que implican cierta lógica y posibilidades, que simplemente no existen o son muy difíciles de implementar únicamente con HTML y CSS.

Ahí es donde entra JavaScript, para brindarte una mano y extender tus posibilidades como desarrollador web, permitiendo: "Validar usuarios y registrarlos", "Generar una paleta de colores dinámica para tu web", "Armando un Chat para comunicarte con tus usuarios"' y muchas otras cosas más.



¿Cómo usar el DOM?

Para todas las explicaciones que se darán a continuación, se utilizara el siguiente código de ejemplo:

HTML

```
<body>
  <div>
    <h1 class="main-title">Publica tu
aprendizaje</h1>
    <section>
      <article id="post1">
        <div class="post-header">
          <img
         src="..."
         class="icon-avatar"
         alt="avatar icon"
         width="75"
          />
          <h2>Einstein y su intelecto</h2>
        </div>
        <div>
          <p>
            ...
          </p>
        </div>
      </article>
      <article id="post2">
        <div class="post-header">
          <img
        src="..."
        class="icon-avatar"
        alt="avatar icon"
        width="75"
          />
          <h2>React y su seguridad</h2>
        </div>
        <div>
          <p>
            ...
          </p>
        </div>
      </article>
      <article id="post3">
        <div class="post-header">
          <img
       src="..."
       class="icon-avatar"
       alt="avatar icon"
       width="75"
          />
          <h2>Algoritmos y su velocidad</h2>
        </div>
        <div>
          <p>
            ...
          </p>
```

CSS

```
ul {
  list-style-type: none;
}

header {
  background-color: #2a2a2a;
  padding-top: 0.8em;
}

header > nav > ul {
  padding: 10px;
  display: flex;
  justify-content: space-around;
  align-items: center;
}

header > nav > div {
  display: flex;
  width: 100%;
  justify-content: space-around;
  padding: 10px;
}

header > nav > div > button {
  background: none;
  border: none;
  font-size: 1rem;
  font-weight: 300;
  text-transform: uppercase;
  padding-bottom: 10px;
  width: 150px;
}

header > nav > div > button:hover {
  color: white;
}

.active {
  cursor: pointer;
  display: flex;
  justify-content: center;
}

.active::after {
  content: ".";
  color: red;
  font-size: 40px;
  position: absolute;
}

.active-button {
```

```
        </div>
      </article>
    </section>
  </div>
  <script src="./index.js"></script>
</body>
```

```
  color: white;
  flex-direction: column;
  align-items: center;
  border-bottom: 3px solid #c23469;
}
```

Resultado

Seleccionar el documento:

A la hora de seleccionar algo en el DOM, apuntaremos primero al documento en el que estamos trabajando, para hacer esto utilizamos la palabra reservada `document`; te invito a que hagas un experimento, copia todo el codigo que te he dejado arriba, abrelo en una pestaña y oprime ALT + F12, esto te desplegará las herramientas de desarrollo, ve a "console" y escribe `console.log(document)`, como podrás ver, esto te desplegará un enorme objeto, el cual te invito indagues. Dentro de él, podrás encontrar cosas como: los elementos HTML (divs, sections, etc...), el header, los metatags, el titulo de la página y mucho mas. Bueno, todo eso que ves, lo puedes manipular con tu JS, veamos cómo podemos acceder a algunas de esas propiedades.

Selectores

Cuando necesitamos seleccionar un elemento dentro del DOM, disponemos de muchas formas, a estas formas las conocemos como **selectores**, veamos algunos de los más importantes y cómo utilizarlos:

Selectores	Definición
`document.getElementById("miId")`	Este selector nos retorna el primer elemento que coincida con ese id
`document.getElementsByClassName("miClase")`	Este selector nos retorna una lista de elementos que coinciden con la clase brindada
`document.querySelector(".unaClase")`	Este selector es probablemente el que más utilizas, este nos permite buscar un elemento utilizando la sintaxis de CSS. Esto quiere decir que podemos buscar una clase utilizando ".clase", un id "#id" o un elemento en sí "div"
`document.getElementsByTagName("div")`	Nos devuelve todas las etiquetas que coincidan con el tipo dado.

Haciendo usos de estos, podemos acceder a las propiedades de los elementos que seleccionamos.

Texto

A la hora de manipular el DOM, nos encontraremos con que la mayoría del contenido que podemos encontrar de nuestros elementos, es texto, por ello es muy necesario el saber acceder a este, la forma es bastante sencilla:

1. Seleccionamos nuestro elemento y lo guardamos en una variable:

```
const tituloPrincipal = document.querySelector(".main-title");
```

2. Accedemos a su texto

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

3. El resultado:

```
"Publica tu aprendizaje"
```

Clases

Una de las propiedades, a la que más probablemente nos interesa ingresar, será a las clases de un elemento; el hacer esto, no dista tanto de lo que ya hemos visto, ya que en efecto, toda la manipulación del DOM, resulta bastante similar, sin embargo no está demás aclarar, veamos cómo hacerlo:

1. Seleccionamos nuestro elemento:

```
const primerPost = document.querySelector("#post1");
```

2. Accedemos a la propiedad deseada:

```
console.log(primerPost)
```

3. Resultado:

```
▶ DOMTokenList [ "post" ]
```

Childs

Finalmente, pero no menos importante cómo acceder a los "Child Nodes", como ya sabemos, los "Childs", son lo hijos de los elementos, osea aquellos elementos que se encuentran "dentro" de otro (cabe aclarar que realmente no se encuentran dentro sino directamente debajo, ya que la estructura del DOM, es la de un árbol y no la de un conjunto de cajas, como se podría llegar a pensar) y los "Nodes", son el nombre que se le da a cualquier elemento dentro del DOM. Es importante saber acceder a ellos, ya que esto nos permitirá generar nuevos elementos, pero por el momento, veamos cómo acceder a estos:

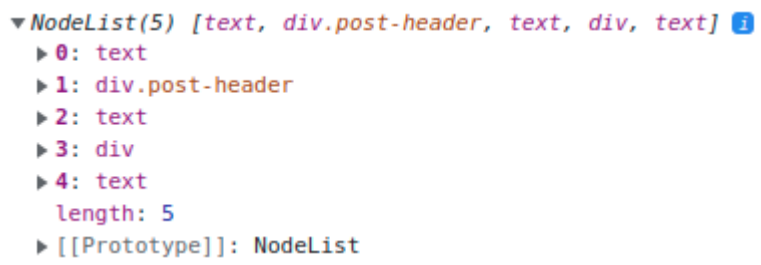
1. Seleccionamos nuestro elemento y lo guardamos en una variable:

```
const tercerPost = document.querySelector("#post3");
```

2. Accedemos a sus "Child Nodes":

```
console.log(tercerPost.childNodes);
```

3. Resultado:



Selector múltiple

Cuando deseamos acceder a un elemento que se encuentra dentro de otro, querySelector admite colocar más de una condición. Funciona igual que los selectores de CSS por descendencia.

Veamos un ejemplo de cómo obtener el elemento párrafo (<p>) del primero post:

```
const parrafo = document.querySelector("#post1 p");
```

En este caso el queryselector busca de izquierda a derecha:

- busca primero un elemento cuyo id sea "post1"
- Luego busca dentro de ese elemento algún hijo o nodo cuya etiqueta sea "p"

Manipulaciones basicas

A continuación, veremos algunos ejemplos para aprender a alterar las propiedades que más comúnmente modificaremos a la hora de trabajar en el DOM.

Alterando Texto

Como podemos ver, tenemos un título principal, al cual quizás nos gustaría acceder y modificarlo con nuestro JS y por ejemplo, agregar la fecha actual, pues bien, hagámoslo.

1. Primero seleccionamos nuestro elemento y lo guardamos en una constante

```
const tituloPrincipal = document.querySelector(".main-title");
```

2. Obtenemos la fecha actual con “Date”:

```
const hoy = new Date();
```

3. Luego podemos acceder a su infinidad de propiedades, entre ellas, `textContent` y redefinirla

```
tituloPrincipal.textContent = `Publica tu aprendizaje:
${hoy.getDate()}/${
 hoy.getMonth() + 1
}/${hoy.getFullYear()}`;
```

Como resultado nos queda algo como:

```
const tituloPrincipal = document.querySelector(".main-title");
const hoy = new Date();

tituloPrincipal.textContent = `Publica tu aprendizaje:
${hoy.getDate()}/${
 hoy.getMonth() + 1
}/${hoy.getFullYear()}`;
```

Como podemos apreciar, logramos modificar el contenido de nuestro título, generando un template string que contiene la fecha actual, generado de manera dinámica.

Modificando atributos

En el ejemplo anterior modificamos el texto del título afectando la propiedad "textContent". Este concepto es genérico, podemos modificar cualquier propiedad que exista dentro del elemento HTML objetivo.

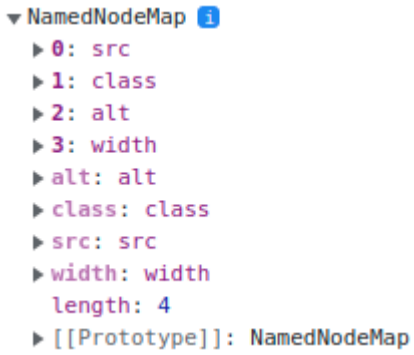
Por ejemplo, veamos como modificar el ícono de un post, modificando la fuente (<src>) de una la imagen (<img>):

1- Apuntamos al icono del primer post:

```
const imagen = document.querySelector("#post1 img");
```

Este elemento "HTML" posee los siguientes atributos modificables:

```
console.log(imagen.attributes)
```



Los ejemplos a continuación se realizarán sobre el atributo "src"

2- Leer la propiedad "src", utilizando getAttribute

```
console.log(imagen.getAttribute("src"))
```

2- Leer la propiedad "src", utilizando directamente la sintaxis "punto" especificando el nombre del atributo:

```
console.log(imagen.src)
```

La única diferencia es que ".src" retorna la ruta absoluta (completa) al archivo de imagen. Utilizando "getAttribute" retorna la ruta relativa (a este proyecto) al archivo de imagen.

3- Modificar la propiedad “src”, utilizando setAttribute

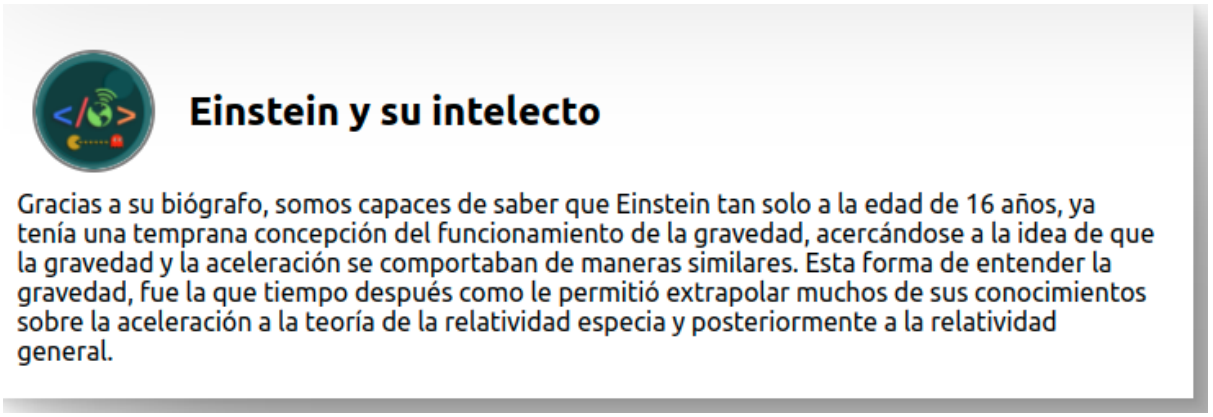
```
imagen.setAttribute("src", "./images/avatar_inove.png")
```

3- Modificar la propiedad “src”, utilizando directamente la sintaxis "punto" especificando el nombre del atributo:

```
imagen.src = "./images/avatar_inove.png"
```

En conclusión, puede utilizar “getAttribute / setAttribute” o la sintaxis “punto” para leer o modificar atributos de los elementos HTML. Dependerá de cada uno qué opción utilizar, la sintaxis “punto” es más “orientada a objetos” pero no permite cambiar un atributo especificado por variable.

Como resultado vemos modificado el ícono de la imagen del primer post:



Agregando elementos al HTML

Desde Javascript podemos crear nuevos elementos HTML utilizando el método "createElement", y podemos agregar estos elementos como hijos (child) a un elemento HTML con "appendChild". Por ejemplo, veamos cómo crear nuevos botones (<buttons>) y agregarlos a un HTML existente:

1- Apuntamos al elemento que será el "padre" de los nuevos botones a generar:

```
const father = document.querySelector("#post1");
```

2- Creamos un nuevo botón totalmente "genérico":

```
const btn1 = document.createElement("button");
```

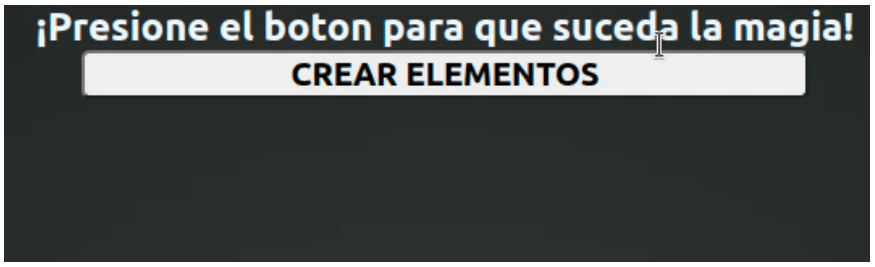
3- Editamos algunas propiedades del botón:

```
btn1.innerText = "BOTON 1";
btn1.classList.add("btn");
```

4- Agregamos este elemento boton (hijo) al padre:

```
father.appendChild(btn1);
```

Podemos repetir esta operación cuantas veces deseemos y obtener un resultado como el siguiente:



Modificando hijos

Quizás nos gustaría saber agregar algún distintivo para nuestro post más clickeado. En este caso agregaremos al header del post3 una etiqueta de span:

1. Seleccionamos nuestro elemento y lo guardamos en una variable:

```
const postHeader = document.querySelector("#post3 .post-header");
```

2. Creamos su hijo, un elemento "span" de HTML con "createElement":

```
const masVisitado = document.createElement("span");
```

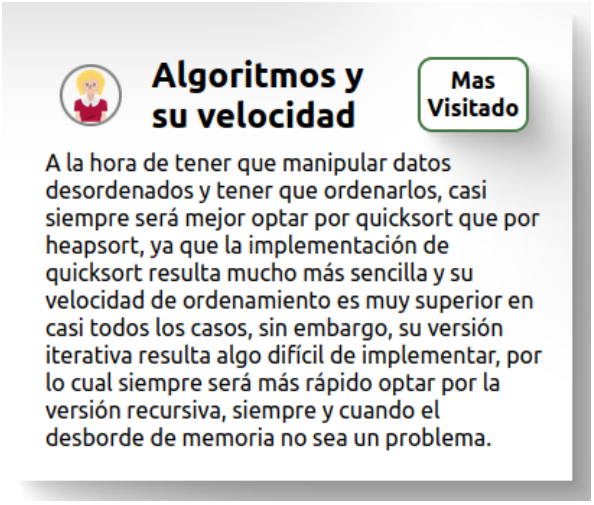
3. Le generamos un contenido al hijo

```
masVisitado.innerText = "Mas Visitado";
```

4. Lo insertamos en el padre correspondiente

```
postHeader.appendChild(masVisitado);
```

5. Resultado



Funciones y eventos

📕 Más información sobre eventos

Antes de continuar, es necesario que demos un viaje rápido por encima de los "eventos", la madre de las interacciones modernas en JS. Los eventos, son señales, que se disparan cuando ocurre algo en particular en nuestro DOM, nosotros con JS podemos apuntar/escuchar estos eventos, utilizando "listeners", los cuales depositamos en aquellos elementos que nos interesa saber qué es lo que está ocurriendo. Entre los eventos que más utilizaremos (cabe aclarar que hay una lista gigantesca), están:

Eventos	Definición
`element.addEventListener("click")`	Este evento, apunta a cuando el elemento es oprimido.
`element.addEventListener("dblclick")`	Este evento se dispara cuando el elemento recibe dos clicks seguidos.
`element.addEventListener("mouseover")`	Este evento se dispara cuando el mouse entra en su área.
`element.addEventListener("mouseout")`	Este evento se dispara cuando el mouse sale de su área.

Cabe aclarar nuevamente que eventos existen un montón y que los iremos viendo a lo largo del tiempo, sin embargo, es probable que estos sean los más utilizados en tu día a día.

Callbacks

Para seguir hablando de clases, eventos y muchas otras cosas más, es necesario que veamos muy por encima un concepto, un poco más extenso, **callbacks**. Este es un término que te cruzaras bastante y la verdad es que es más sencillo de lo que puedes llegar a pensar, aunque es bueno tener claro que por mas que sea sencillo, este tiene muchas implicaciones, a la cuales les prestaremos su debido tiempo mas adelante, mientras tanto, veamos su definición: **se puede entender un "Callback", como a cualquier función que llame a otra función dentro de su cuerpo.** Como puedes ver, el concepto es sumamente sencillo, pero ya verás que útil puede resultar.

Ahora veamos cómo asignar un listener y utilizarlo de manera práctica:

Apuntamos a nuestro elemento que asociaremos con el evento:

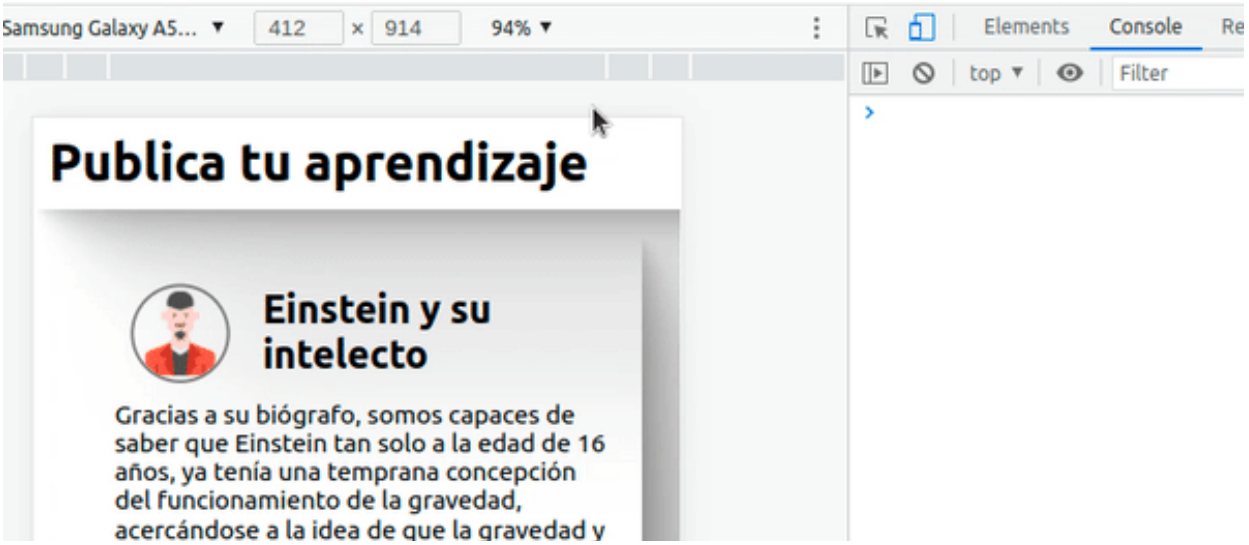
```
const primerPost = document.querySelector("#post1");
```

Agregamos un listener:

```
primerPost.addEventListener("click", function () {
 console.log("Primer post");
});
```

Esta función que estamos llamando dentro de nuestro listener, **es un "callback", ya que es una función dentro de una función.**

Resultado:



Podemos apreciar el `console.log()` correspondiente a nuestro click.

Alterando Clases

Finalmente cómo agregar clases a nuestros elementos utilizando JS, esto es algo muy útil, así que veamos algo de su potencial.

1. Apuntamos a nuestro elemento:

```
const segundoPost = document.querySelector("#post2");
```

2. Agregamos un listener para cuando entramos en la tarjeta:

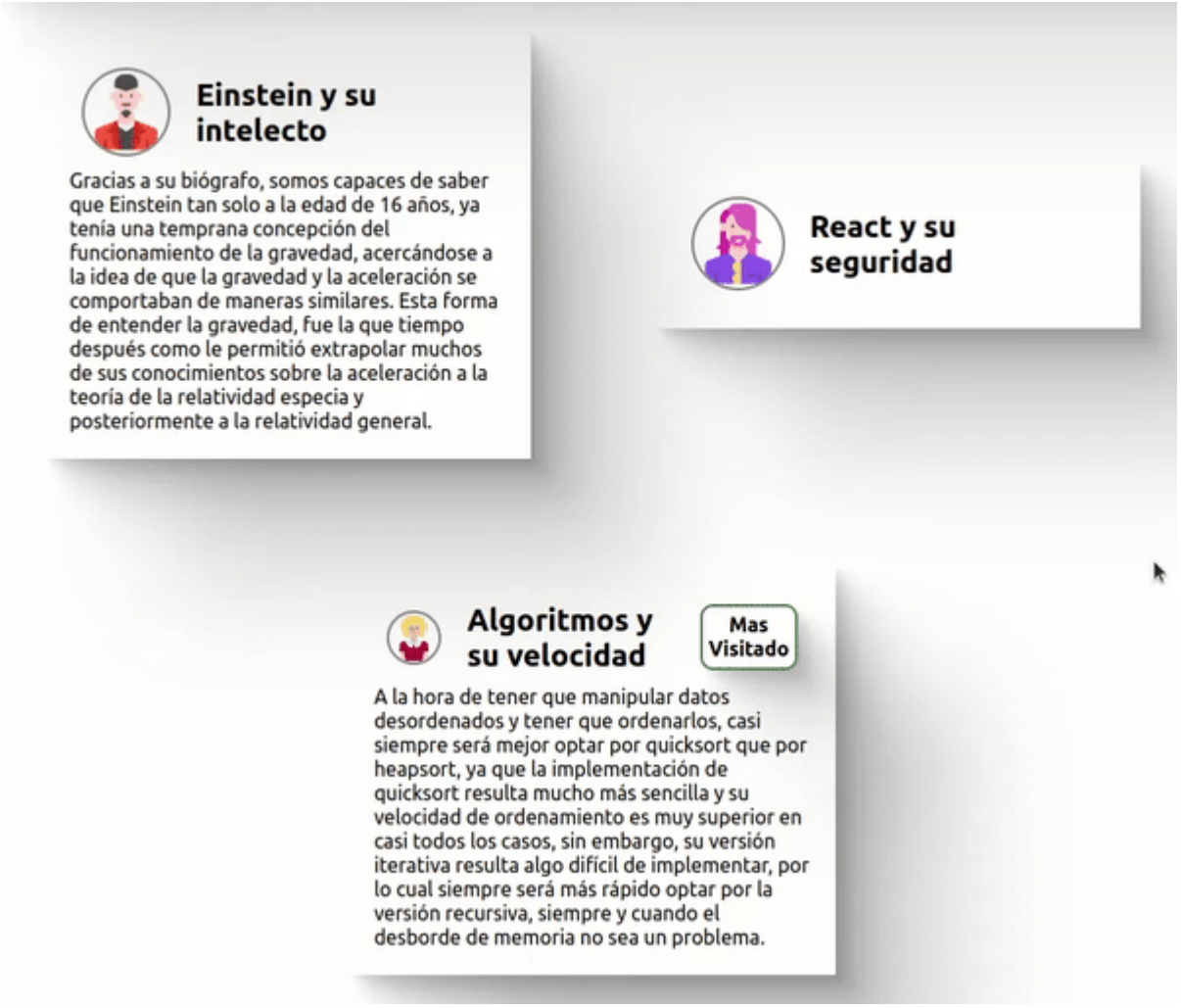
```
segundoPost.addEventListener("mouseover", function () {
// Removemos la clase de cerrado
segundoPost.classList.remove("closed");
// Agregamos la clase de abierto
segundoPost.classList.add("open");
});
```

Dentro del evento, cuando el mouse pase por encima de la tarjeta (mouseover) se quitará la clase "closed" y se agregará la clase "open" para que se abra el post. Las clases "closed" y "open" están definidas en el archivo de estilos CSS.

3. Agregamos otro listener para cuando salimos de la tarjeta:

```
segundoPost.addEventListener("mouseout", function () {
// Agregamos la clase de cerrado
segundoPost.classList.add("closed");
// Quitamos la clase de abierto
segundoPost.classList.remove("open");
});
```

Resultado:


Einstein y su intelecto
Gracias a su biógrafo, somos capaces de saber que Einstein tan solo a la edad de 16 años, ya tenía una temprana concepción del funcionamiento de la gravedad, acercándose a la idea de que la gravedad y la aceleración se comportaban de maneras similares. Esta forma de entender la gravedad, fue la que tiempo después como le permitió extrapolar muchos de sus conocimientos sobre la aceleración a la teoría de la relatividad especia y posteriormente a la relatividad general.
React y su seguridad
Algoritmos y su velocidad
Mas Visitado
A la hora de tener que manipular datos desordenados y tener que ordenarlos, casi siempre será mejor optar por quicksort que por heapsort, ya que la implementación de quicksort resulta mucho más sencilla y su velocidad de ordenamiento es muy superior en casi todos los casos, sin embargo, su versión iterativa resulta algo difícil de implementar, por lo cual siempre será más rápido optar por la versión recursiva, siempre y cuando el desborde de memoria no sea un problema.

Descargar un repositorio de GitHub

Para poder realizar las actividades de aquí en adelante debemos tener instalado y configurado nuestro GitHub. Todos los ejemplos prácticos estarán subidos al repositorio GitHub de **InoveAlumnos**, para aprender como descargar estos ejemplos desde el repositorio referirse al "Instructivo de GitHub: Descargar un repositorio" disponible entre los archivos del campus. De no encontrarse allí, por favor, tenga a bien comunicarse con alumnos@inove.com.ar para su solicitud.

Debemos descargar el repositorio que contiene los ejemplos de clase de ésta unidad:
https://github.com/InoveAlumnos/intro_dom_js

¡Hasta la próxima!

Con esto finaliza esta clase, a partir de ahora tienen todo preparado para empezar a realizar programas y continuar con la ejercitación de la clase.

Si desean conocer más detalles sobre el contenido pueden iniciar un tema de discusión en el lugar correspondiente, o visitar los "Links de interés" que se encuentra al final de este apunte.

Links de interés

- Manipulación del DOM
- Eventos