Programación con Java Script I

Sesión sincrónica 7

DOM

Bienvenida y actividad de bienestar

Duración: 10 minutos.

Nombre de la práctica: La mejor versión de ti mismo
Descripción de la práctica: Escribe durante por lo menos 20 minutos acerca de la mejor versión posible de ti mismo.
Palabras clave: Emociones positivas, fortalezas de carácter, autorregulación, esperanza
Instrucciones para el participante: Imagina que dentro de 20 años has crecido en todas las áreas o maneras que te gustaría crecer y las cosas te han salido tan bien como te las imaginaste.

¿Cómo es esa mejor versión de ti mismo?
¿Qué hace él o ella cotidianamente?
¿Qué dicen los demás acerca de él o ella?

No es necesario que compartas este escrito, ya que el objetivo de esta reflexión es enfocarse en la experiencia que viviste mientras reflexionabas en esa mejor versión posible de ti mismo.
Fuente: Ejercicio contribuido por Taylor Kreiss de University of Pennsylvania Positive Psychology Center, y basado en el libro A Primer in Positive Psychology de Christopher Peterson.

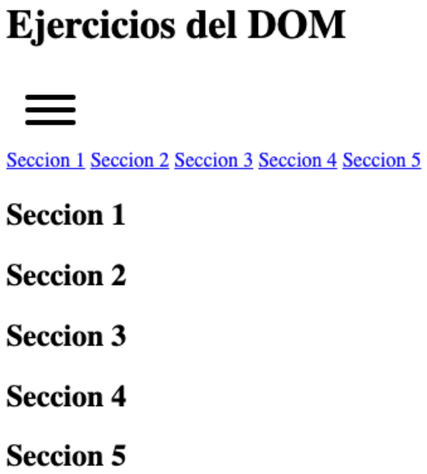
Actividad guiada
Parte 1

Duración: 75 minutos.

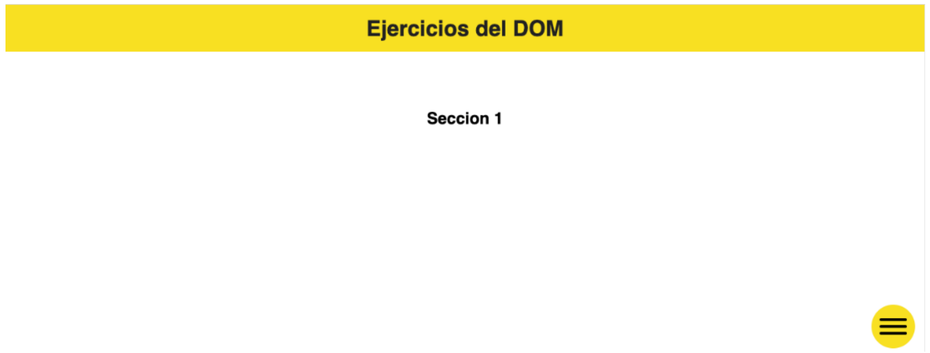
En el siguiente ejercicio vas a practicar los conceptos del DOM aprendidos en esta experiencia educativa, a través de la creación de una aplicación que tendrá un menú de hamburguesa con cinco secciones, que serán accedidas desde el menú.
Para alcanzar nuestro objetivo, realiza las siguientes actividades:

1. Crea un archivo html y crea la estructura básica.
2. Desde la página de **bootstrap** (en la página https://getbootstrap.com/), busca la liga CND del css para incluirlos en tu proyecto.
3. En el cuerpo incluye un header con un **h1** y ponle el título “Ejercicios del DOM”.
4. Crea un botón con la clase **panel-btn, hamburger, hamburger—vortex del tipo button.**
5. Dentro de dicho botón incluye un **span** con la clase burger-box y dentro de este incluye un nuevo span con la clase **hamburger-inner**.
6. Crea un aside con la clase panel.
7. Dentro del **aside**, agrega una barra de navegación con la clase **menu.**
8. Crea cinco links que apunten desde la **#seccion1** hasta la **#seccion5** y coloca el texto desde seccion1 hasta seccion5 respectivamente.
9. Crea las cinco secciones, cada una de ellas tendrá el id correspondiente a su **sección** (seccion1, seccion2, etc.) y la clase section, así como un **h2** con la leyenda sección 1, sección 2, hasta la sección 5
10. Al ejecutar tu código, podrás validar que tenga el siguiente aspecto:

Actividad guiada Parte 1



Liga el archivo dom.css que se encuentra en la zona de descargas y automáticamente tu página deberá lucir así:



Actividad guiada
Parte 1

Sin embargo, aún no funciona el menú. Para activar su funcionamiento, crea una carpeta js y dentro de ella crea el archivo js con el que se va a controlar el menú hamburguesa. También crea un archivo js que va a controlar el flujo de los js, podría llamarse **index_dom.**

1. En el archivo js del menú hamburguesa, exporta por default una función que se llame **hambugerMenu**, la cual va a recibir como parámetros el botón que activa (llamémosle **panelBtn**) y el panel (que es el elemento que se va a mover).
2. Como se va a utilizar la delegación de eventos del DOM, y para poder trabajar más fácilmente, declara una constante y asígnale la clase **document.**
3. Asigna, mediante la delegación de eventos, el evento **click** al document. Indícale que, si el evento que lo origina coincide con el selector que se está enviando en panelBtn, intercambia la clase **is-active** al elemento **panel**, puedes hacerlo con la instrucción

```
d.querySelector(panel).classList.toggle("is-active");
```

4. Para poder probar el funcionamiento, es necesario importar en el archivo que controla el funcionamiento del DOM (index_dom.js) la función hamburgerMenu de **“ruta_del_archivo/nombre_archivo.js”.**
5. Crea una constante y asígnale la clase **document**.
6. Indícale a JavaScript que la carga del documento, con addEventListener, habilite la función **hamburgerMenu**.
7. Pásale como parámetros a hamburgerMenu el nombre del selector del botón que activa el menú (en este caso “.panel-btn”) y el elemento que se va a mover (selector “·panel”).
8. Ejecuta tu código y ahora podrás visualizar esto cuando aprietas en el menú de hamburguesa:

Actividad guiada
Parte 1

Si bien ya está funcionando, el código aún tiene algunas fallas, una de ellas es que cuando das clic en el **span** que forma las líneas, este no funciona.

1. Indícale a JavaScript que, si el objeto que originó el evento es el botón o cualquiera elemento hijo del botón, intercambie la clase is-active al elemento panel.
2. Valida el resultado y deberás poder desplegar el menú oprimiendo sobre cualquier parte del botón.

Ahora, si consultas cualquier página para ver cómo debería funcionar el menú hamburguesa (https://jonsuh.com/hamburgers/, por ejemplo), podrás observar que falta animar el botón.

1. Agrega al elemento hamburger (que en este caso es panelBtn) la clase is-active.
2. Valida que el botón ahora cambia cuando haces clic sobre él.

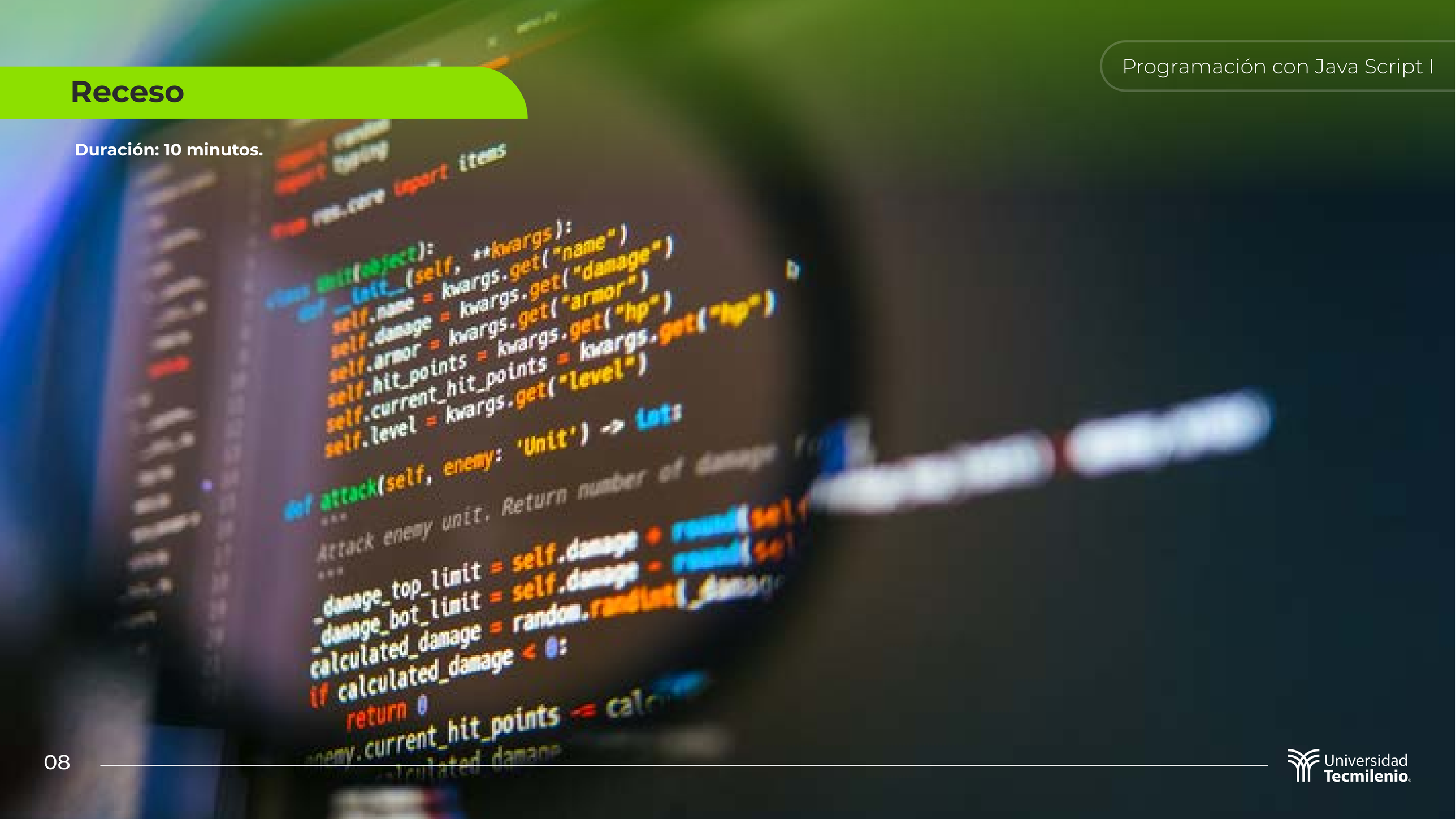
Actividad guiada
Parte 1

Otro error que se presenta es que cuando haces clic sobre alguna de las secciones, se logra visualizar que la página sí se desplaza hasta esa página, sin embargo, el menú sigue presente, es decir, no desaparece como se esperaría que lo hiciera. Para ello, realiza las siguientes modificaciones:

1. Agrega un tercer parámetro a la clase **hamburgerMenu** que haga referencia a un link del menú **(menuLink)**.
2. En el cuerpo de la clase, agrega otro condicional para la delegación de eventos y valida que, si el objeto generó el evento, revisa que su selector coincida con lo que venga en la variable **menuLink**, se le quite la clase **is-active** tanto al **panelBtn** para que regrese del tache a líneas como al panel para que el menú desaparezca.
3. Valida que tu código funcione correctamente.

Receso

Duración: 10 minutos.



Actividad guiada
Parte 2

Duración: 75 minutos.

Ahora comienza a ingresar elementos a una de las secciones. En la sección 1, se va a agregar un reloj y una alarma.

1. En el archivo html:

   a. Modifica en el menú el contenido link con el id #seccion1 y colócale Reloj Digital y Alarma Sonora.
   b. Modifica el título de la sección con id **seccion1** y colócale Reloj Digital y Alarma Sonora.
   c. Crea una div vacía con el id reloj.
   d. Crea un div con la clase fila flex centrada.
   e. Crea cuatro botones:

      i. El primero con el id **activar-reloj** y la cadena Iniciar Reloj en su contenido.
      ii. El segundo con el id **desactivar-reloj** y la cadena Detener Reloj en su contenido.
      iii. El tercero con el id **activar-alarma** y la cadena Iniciar Alarma en su contenido.
      iv. El cuarto con el id **desactivar-alarma** y la cadena Detener Alarma en su contenido.

2. Crea el archivo reloj.js y realiza lo siguiente:

   a. Declara una constante y asígnale la clase document.
   b. Para crear el reloj, exporta la función para la creación de este:

      i. Nómbrala digitalClock.
      ii. Pásale como parámetros el reloj como tal (clock), el botón que lo inicia (btnPlay) y el botón que lo detiene (btnStop).
      iii. Declara la variable clockTempo en donde se guardará el reloj que se va a crear.
      iv. Utiliza la delegación de eventos y agrégale al document el evento click.
      v. Dentro de la delegación, detecta qué objeto originó el evento y valida:

Actividad guiada Parte 2

Duración: 75 minutos.

1. Si el evento lo originó el btnPlay:
    a. Asigna a la variable clockTempo un setInterval que se ejecute cada segundo. Lo que realizará dentro es:
        i. Guardar en una variable que se llame clockHour, que será igual al evento toLocaleTimeString() de un nuevo objeto de tipo Date.
        ii. Asigna el contenido de la variable clockHour (la cual estará interpolando cada segundo) dentro de un h3 al elemento que como selector tenga el contenido de la variable clock en su contenido innerHTML (esta es la instrucción que se espera: d.querySelector(clock).innerHTML = `<h3>${clockHour}</h3>`;).
    b. Para evitar que el usuario presione más de una vez el botón, una vez que esté ejecutándose el reloj, deshabilita el botón, asignando la propiedad disabled igual a true en el target del objeto que origina el evento.
2. Si el evento lo originó el btnStop:
    a. Limpia el interval que se encuentra en la variable clockTempo.
    b. Desaparece el reloj poniendo a nulo la propiedad innerHTML del selector clock.
    c. Vuelve a habilitar el botón, asignando la propiedad disabled en false para el botón que originó el evento.
c. Importa al archivo index_dom.js la función digitalClock del archivo reloj.js.
d. En la delegación de eventos, indica que mande a ejecutar la función digitalClock con los selectores del reloj y los botones como parámetros, es decir, (#reloj, #activar-reloj y desactivar-reloj).
e. Cuando termines, valida que tu página se visualice así:

Actividad guiada
Parte 2

f. Para crear la alarma, exporta la función para la creación de esta:

i. Nómbrala alarm.
ii. Pásale como parámetros el archivo de sonido (sound), el botón que lo inicia (btnPlay) y el botón que lo detiene (btnStop).
iii. Declara la variable **alarmTempo** en donde se guardará la etiqueta audio que se va a crear.
iv. Crea una constante en donde se va a guardar un elemento del DOM, podrías utilizar el nombre **$alarm**, y asígnale la etiqueta audio, utilizando el método **createElement** de la clase **document**.
v. Asígnale al atributo src de la constante $alarm el valor que venga en la variable **sound**, que es la ruta del archivo alarma.mp3 que puedes obtener desde los archivos adjuntos.
vi. Utiliza la delegación de eventos y agrégale al **document** el evento **click.**

Actividad guiada
Parte 2

v. Asígnale al atributo src de la constante $alarm el valor que venga en la variable sound, que es la ruta del archivo alarma.mp3 que puedes obtener desde los archivos adjuntos.
vi. Utiliza la delegación de eventos y agrégale al document el evento clic.
vii. Dentro de la delegación, detecta qué objeto originó el evento y valida:

1. Si el evento lo originó el btnPlay:
    a. Asigna a la variable alarmTempo un setTimeout que se ejecute después de 2 segundos. Lo que se realizará dentro es:
    i. Hacer sonar el sonido de la alarma.
    b. Para evitar que el usuario presione más de una vez el botón, una vez que esté ejecutándose el reloj, deshabilita el botón, asignando la propiedad disabled igual a true en el target del objeto que origina el evento.
2. Si el evento lo originó el btnStop:
    a. Limpia el timeout que se encuentra en la variable alarmTempo.
    b. Detén el sonido con $alarm.pause().
    c. Resetea el sonido con currentTime = 0.
    d. Vuelve a habilitar el botón, asignando la propiedad disabled en false para el botón que originó el evento.

g. Importa al archivo index_dom.js la función alarm, solo necesitarás modificar el import que se realizó del reloj digital, pues las dos funciones provienen del mismo archivo.
h. En la delegación de eventos, indica que mande a ejecutar la función alarm con la ubicación del archivo alarma.mp3 (se sugiere que la guardes en una carpeta llamada assets dentro de tu proyecto) y los botones como parámetros, es decir, (“assets/alarma.mp3”, #activar-alarma y desactivar-alarma).
i. Valida que tu página tenga el comportamiento esperado.

Consideraciones
Instructor, refuerza los conocimientos de aquellos temas que consideres necesarios para que el aprendedor pueda realizar este ejercicio.