Programación con JavaScript II

Sesión sincrónica 2

Asincronía

Bienvenida y actividad de bienestar

Duración: 10 minutos.

Nombre de la práctica: Un momento para respirar.
Descripción de la práctica: Aprender a respirar por la nariz y a tranquilizar tu mente.

Palabras clave: Fortalezas de carácter y autorregulación.

Instrucciones para el aprendedor:
La autorregulación, también percibida como control, es una fortaleza de carácter muy importante dentro de la psicología positiva. Este concepto implica regular lo que uno siente y hace, ser disciplinado, así como mantener un control sobre los apetitos y, especialmente, sobre las emociones.

En la actualidad vivimos situaciones muy estresantes que provocan que nuestra reacción instintiva y natural ante ellas sea estallar en ira. Sin embargo, las consecuencias de este comportamiento no solo se quedan en nosotros, sino que también pueden llegar a afectar a terceros.
A continuación, se presenta un ejercicio que te ayudará a cultivar la fortaleza de autorregulación:

1. Toma dos minutos de tu tiempo, siéntate en un lugar cómodo, donde no haya mucho ruido que te pueda distraer.
2. Escucha música de relajación (crea tu propio ambiente de meditación).
3.Comienza a respirar y exhalar por la nariz.
Trata de que tu respiración y exhalación duren el mismo tiempo.
4.Fija tu mente en tu respiración, en cómo entra y sale el aire de tu cuerpo.
Así durante dos minutos.

Te recomendamos que si durante este periodo algún pensamiento (olvidé algo en la oficina, más tarde tengo que hacer tal actividad, etc.) llega a tu mente, solo déjalo pasar y regresa a la concentración en tu respiración.
Al finalizar los dos minutos sentirás paz en tu ser. Comienza a hacer este ejercicio de respiración y meditación todos los días y poco a poco ve aumentando los minutos de este.

Fuente: Conferencia Rosalinda Ballesteros.

Actividad guiada Parte 1

Duración: 75 minutos.

¿Cuál es la diferencia entre la programación asíncrona no bloqueante y la programación asíncrona bloqueante?

¿Para qué se utilizan las promesas?

¿Qué entiendes por *callback hell*?

¿Cuáles son las palabras reservadas para implementar funciones asíncronas?

En este tema pondrás en práctica los conceptos de asincronía que has aprendido. Para alcanzar el objetivo, realiza las siguientes actividades:

Ejercicio 1

1. Crea un documento **html** en el directorio que hayas destinado para tus archivos de la sesión guiada 1 del curso Programación con JavaScript II.
2. En ese mismo directorio, crea una carpeta en donde guardarás los archivos **js**.
3. En esta carpeta, crea el archivo **asincronia.js.**
4. Crea una función anónima autoejecutable con código síncrono bloqueante que realice lo siguiente:
    a. Imprime en consola la cadena “Código Síncrono”.
    b. Imprime en consola la cadena **Inicio**.
    c. Crea la función **dos**, que imprimirá en la consola el texto “Dos”.
    d. Crea la función **uno**, que imprimirá en la consola el texto “Uno”. Luego, invoca a la función **dos**() y finalmente imprimirá en la consola el texto “Tres”.
    e. Invoca a la función **uno**().
    f. Imprime en consola la cadena “Fin”.
5. ¿Cuál es el comportamiento de la función y por qué?
6. Ingresa tu código en la página http://latentflip.com/loupe/ y analiza el comportamiento del mismo.

Actividad guiada
Parte 1

7. Crea una segunda función anónima autoejecutable con código asíncrono no bloqueante que realice lo siguiente:
    a. Imprime en consola la cadena “Código Asíncrono”.
    b. Imprime en consola la cadena Inicio.
    c. Crea la función **dos**, que imprimirá en la consola el texto “Dos”.
    d. Crea la función **uno**. En su cuerpo, crea un setTimeout y pásale como primer parámetro una función anónima que imprima en la consola el texto “Uno” y pásale un 0 como segundo parámetro. Luego, invoca a la función **dos**() y finalmente imprimirá en la consola el texto “Tres”.
    e. Invoca a la función **uno**().
    f. Imprime en consola la cadena “Fin”.
8. Ingresa tu código en la página http://latentflip.com/loupe/ y analiza el comportamiento del mismo.

Ejercicio 2

1. Crea una función callback que eleve al cuadrado un número recibido como primer parámetro y como segundo parámetro un callback con las siguientes características:
    a. Declara la función con el nombre **cuadradoCallback** y pásale los parámetros indicados.
    b. En el cuerpo, coloca un setTimeout y pásale como primer parámetro una función anónima, en la que el primer parámetro recibirá una función callback, que, a su vez, recibirá como parámetro un **valor** y una función a ejecutar (**valor*valor**). Como segundo parámetro, indícale que el temporizador sea dinámico, es decir, puede ser 0, pero si no es 0, entonces crea un número random y multiplícalo por 1000.
2. Invoca la función **cuadradoCallback** y pásale 0 como primer parámetro y la siguiente función callback como segundo parámetro:

```
(valor,resultado) =>{
console.log("Inicia Callback");
console.log(`Callback: ${valor},${resultado}`);

});
```

Actividad guiada Parte 1

3. Vuelve a invocar la función callback dentro de la primera invocación, pero con el número 1 como parámetro y solo imprimirá en consola el siguiente resultado (ya no es necesario que imprima “inicia Callback”).
4. Vuelve a realizar el paso 3, pero ahora con los siguientes valores, desde el 2 hasta el número 5.

Consideraciones:

Aprendedor:

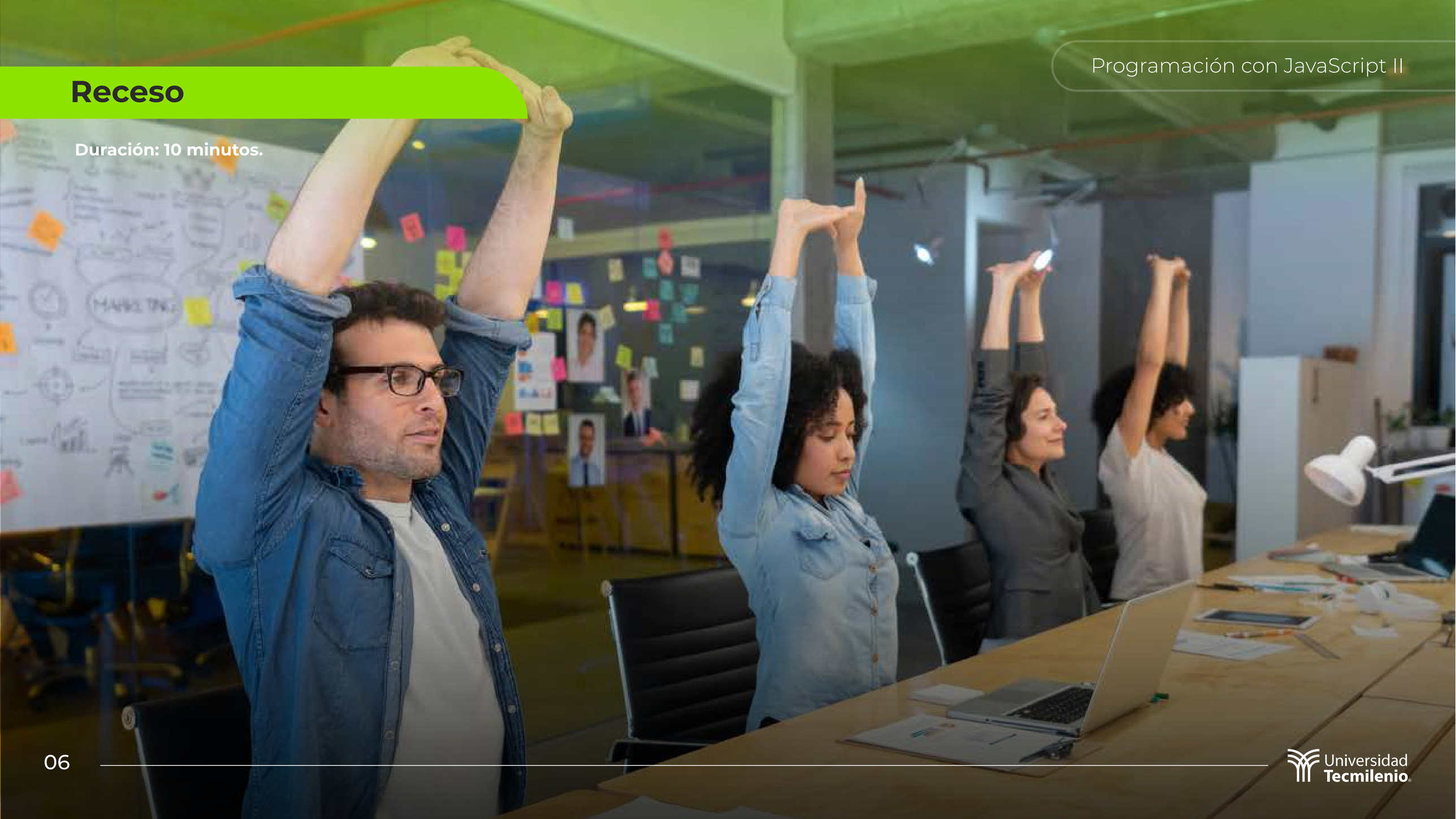
Es posible que este tipo de funciones te resulten confusas, por lo que te sugiero que revises los videos adicionales y que intentes realizar los ejercicios antes de la sesión guiada, así podrás llegar con dudas específicas para tu instructor.

Instructor:

Es importante recalcar a los aprendedores la problemática que puede presentar el callback hell.

Receso

Duración: 10 minutos.



Actividad guiada Parte 2

Duración: 75 minutos.

Ejercicio 3

Crea una función que calcule el cuadrado, utilizando promesas:

1. Declara la función y nómbrala **cuadradoPromesa** y pásale como parámetro **valor**.
2. Dentro del cuerpo, realiza lo siguiente:
    a. Evalúa que el **valor** que se recibe es diferente a un número. Crea un return en el que se va a ejecutar el método reject() del objeto **Promise.**
    b. Pasa como parámetro al **reject** la cadena `**Error, El valor ingresado ${value} no es un número**´.
    c. Crea un return que regresará una instancia del objeto **Promise**. Esta promesa debe recibir una función flecha, la cual recibe como parámetros **resolve y reject.** En el cuerpo de la función, crea un setTimeout que recibirá como primer parámetro una función anónima con el **resolve**. Después, dentro de este resolve, crea el objeto con las llaves **valor y resultado** y con los valores **valor y valor * valor**, respectivamente.
3. Ejecuta la función pasándole como parámetro un 0.
4. Utiliza el método *then* para ir trabajando la sincronía. Este then deberá recibir como parámetro una función que, a su vez, recibirá como parámetro un objeto que imprimirá en la consola el texto “inicia Promesa” y posteriormente imprimirá en la consola `**Promesa: ${obj.value},${obj.result}**` y regresará con un return una segunda llamada a la función **cuadradoPromesa**, pero que recibirá como parámetro el número 1.
5. Crea otro then en el que pases como parámetro la función flecha que solamente imprimirá `**Promesa: ${obj.value},${obj.result}**` y regresará la llamada a la función **cuadradoPromesa** en la que recibirá como parámetro el número 2.
6. Vuelve a realizar el paso cinco para los números 3, 4 y 5.
7. Genera un último then que manipule el resultado del número 5 e imprime en pantalla el texto “Fin Promesa”.
8. Ahora utiliza el método catch para recibir el error en caso de que se haya presentado uno en la validación. Posteriormente, envía el mensaje a la consola con un **console.error**.

Actividad guiada Parte 2

Ejercicio 4

Crea una función que calcule el cuadrado de un número, utilizando las funciones asíncronas *async* y await.

1. Crea una función asíncrona utilizando la palabra reservada async y nómbrala **funcionAsincronaDeclarada.**
2. Con la sentencia *try*:
    a. Imprime en consola el texto “inicio Async Function”.
    b. Declara una variable con nombre **obj** a la que le asignarás la llamada a la función **cuadradoAsincrona** con el número 0 como parámetro y anteponiendo la palabra reservada await antes del nombre de la función.
    a. Imprime en consola el valor y el resultado de obj **`Async Function: ${obj.value},${obj.result}´**.
    b. Asigna a la variable una nueva llamada a la función, pero ahora el parámetro es el número 1.
    c. Vuelve a imprimir el resultado.
    d. Realiza los pasos anteriores, pero ahora con los números del 2 al 5.
    e. Imprime al final el texto “Fin Async Function”.
3. Utiliza la sentencia *catch* para detectar el error.
4. Ejecuta la función **funcionAsincronaDeclarada.**

Consideraciones:

Instructor: Hay algunos pequeños temas de promesa que posiblemente el usuario no haya comprendido muy bien, procura poner énfasis para detectar y explicar esas dudas que pudieran surgir.

Cierre

Duración: 10 minutos.

Ahora has aprendido a usar uno de los pilares fundamentales de JavaScript: la asincronía. Esto te ayudará a evitar que tus aplicaciones se queden en espera de algún resultado y que tu usuario se desespere y termine por no querer utilizar tu aplicación.