Curso HTML5, CSS y JavaScript

Módulo 6: Programación asincrónica con JS

- emilianoagustingallo@gmail.com
- https://www.linkedin.com/in/emiliano-gallo/

Capacitarte

Sincrónico vs asincrónico

Un código **síncrono** es aquel donde cada instrucción espera a la anterior para ejecutarse.

Un código **asíncrono** no espera a las instrucciones diferidas y continúa con su ejecución.

Por lo general la **asincronía** permite tener una mejor respuesta en las aplicaciones y reduce el tiempo de espera del cliente.

Ventajas y desventajas

Lenguajes sincrónicos

- Fáciles de leer y entender
- Fáciles de depurar.
- No hay que preocuparse por la disponibilidad de los datos.

Lenguajes asincrónicos

- Más difíciles de entender
- Mas dificiles de depurar. Un código puede compilar y funcionar correctamente. Pero si tenemos una función que depende de otra puede ejecutarse cuando no lo teníamos pensado.
- Excelentes para el manejo de interfaces y mejores tiempos de respuestas.

Programación sincrónica

En el siguiente código, cada instrucción se ejecutará en secuencia hasta terminar.

En consola se mostrará la información en el mismo orden en el que se escribió el código.

```
console.log('Primero');
setTimeout(_ => { console.log('Segundo');},10);
console.log('Tercero');
```

Programación asincrónica

En el siguiente código, cada instrucción se ejecutará en diferentes momentos.

Si ejecutamos este ejemplo veremos imprimirse **‘Primero’**, **‘Tercero’**, **‘Segundo’**. Esto se debe a que estamos usando la instrucción ***setTimeout()*** que difiere la ejecución x milisegundos

```
console.log('Primero');
setTimeout(_ => { console.log('Segundo');},10);
console.log('Tercero');
```

Funciones callbacks

Una función **callback** es una función de primer nivel que se pasa a otra función como variable y ésta es ejecutada en algún punto de la ejecución de la función que la recibe.

En el siguiente ejemplo el **alert()** se va a mostrar luego de que se termina de ocultar el elemento **p**.

```
$("button").click(function(){
  $("p").hide(3000, function(){
      alert("The paragraph is now hidden");
  });
});
```

Funciones callbacks

Las declaraciones de **JS** se ejecutan línea por línea. Sin embargo, cuando realizamos una llamada que puede demorar algunos **ms**,puede que se ejecute la siguiente línea de código aunque la línea anterior no haya finalizado.

Para evitar esto, puede crear una función **callback**. Una función **callback** se ejecuta una vez finalizado el efecto actual.

Promises

Es una **referencia** para un valor que no necesariamente se conoce al momento de creada la promesa.

Permite asociar **callbacks** que se ejecutarán dependiendo del éxito o fracaso de la acción prometida.

Las promesas pueden tener 3 estados definidos:

- pendiente (pending): estado inicial, no cumplida o rechazada.
- cumplida (fulfilled): significa que la operación se completó satisfactoriamente.
- rechazada (rejected): significa que la operación falló.

Algunos ejemplos