JavaScript asíncrono

Las funciones que se ejecutan en **paralelo** con otras funciones se denominan **asíncronas.**

Un buen ejemplo es setTimeout() de JavaScript.

JavaScript asincrónico

Los ejemplos utilizados en el capítulo anterior fueron muy simplificados.

El propósito de los ejemplos fue demostrar la sintaxis de las funciones de devolución de llamada:

Ejemplo

```
function myDisplayer(something) {
 document.getElementById("demo").innerHTML = something;
}

function myCalculator(num1, num2, myCallback) {
 let sum = num1 + num2;
 myCallback(sum);
}

myCalculator(5, 5, myDisplayer);
```

En el ejemplo anterior, myDisplayeres el nombre de una función.

Se pasa myCalculator()como argumento.

En el mundo real, las devoluciones de llamadas se utilizan con mayor frecuencia con funciones asincrónicas.

Un ejemplo típico es JavaScript setTimeout().

Esperando un tiempo de espera

Al utilizar la función de JavaScript setTimeout(), puede especificar una función de devolución de llamada que se ejecutará cuando se agote el tiempo de espera:

Ejemplo

```
setTimeout(myFunction, 3000);

function myFunction() {
```

```
 document.getElementById("demo").innerHTML = "I love You !!";
}
```

En el ejemplo anterior, myFunctionse utiliza como devolución de llamada.

myFunctionse pasa setTimeout()como argumento.

3000 es el número de milisegundos antes del tiempo de espera, por lo que myFunction()se llamará después de 3 segundos.

Nota

Cuando pase una función como argumento, recuerde no utilizar paréntesis.

Derecha: setTimeout(myFunction, 3000);

Equivocado:~~setTimeout(miFunción(), 3000)~~;

En lugar de pasar el nombre de una función como argumento a otra función, siempre puedes pasar una función completa:

Ejemplo

```
setTimeout(function() { myFunction("I love You !!!"); }, 3000);

function myFunction(value) {
 document.getElementById("demo").innerHTML = value;
}
```

En el ejemplo anterior, function(){ myFunction("I love You !!!"); } se utiliza como devolución de llamada. Es una función completa. Esta función completa se pasa a setTimeout() como argumento.

3000 es el número de milisegundos antes del tiempo de espera, por lo que myFunction()se llamará después de 3 segundos.

Esperando intervalos:

Al utilizar la función de JavaScript setInterval(), puede especificar una función de devolución de llamada que se ejecutará para cada intervalo:

Ejemplo

```
setInterval(myFunction, 1000);
```

```
function myFunction() {
 let d = new Date();
 document.getElementById("demo").innerHTML=
 d.getHours() + ":" +
 d.getMinutes() + ":" +
 d.getSeconds();
}
```

En el ejemplo anterior, myFunctionse utiliza como devolución de llamada.

myFunctionse pasa setInterval()como argumento.

1000 es el número de milisegundos entre intervalos, por lo que myFunction()se llamará cada segundo.

Alternativas de devolución de llamada

Con la programación asincrónica, los programas JavaScript pueden iniciar tareas de larga ejecución y continuar ejecutando otras tareas en paralelo.

Pero los programas asincrónicos son difíciles de escribir y de depurar.

Por ello, la mayoría de los métodos asincrónicos modernos de JavaScript no utilizan devoluciones de llamada. En cambio, en JavaScript, la programación asincrónica se resuelve mediante **promesas** .