JavaScript *callbacks*

Una devolución de llamada es una función que se pasa como argumento a otra función.

Esta técnica permite que una función llame a otra función.

Una función de devolución de llamada puede ejecutarse después de que otra función haya finalizado.

Secuencia de funciones

Las funciones de JavaScript se ejecutan en el orden en que se llaman, no en el orden en que se definen.

Este ejemplo terminará mostrando "Adiós":

Ejemplo

```
function myFirst() {
 myDisplayer("Hello");
}

function mySecond() {
 myDisplayer("Goodbye");
}

myFirst();
mySecond();
```

Este ejemplo terminará mostrando "Hola":

Ejemplo

```
function myFirst() {
 myDisplayer("Hello");
}

function mySecond() {
 myDisplayer("Goodbye");
}

mySecond();
myFirst();
```

Control de secuencia

A veces desearía tener un mejor control sobre cuándo ejecutar una función.

Supongamos que desea realizar un cálculo y luego mostrar el resultado.

Podrías llamar a una función de calculadora (myCalculator), guardar el resultado y luego llamar a otra función (myDisplayer) para mostrar el resultado:

Ejemplo

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

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

let result = myCalculator(5, 5);
myDisplayer(result);
```

O bien, puede llamar a una función de calculadora (myCalculator) y dejar que la función de calculadora llame a la función de visualización (myDisplayer):

Ejemplo

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

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

myCalculator(5, 5);
```

El problema con el primer ejemplo anterior es que hay que llamar a dos funciones para mostrar el resultado.

El problema con el segundo ejemplo es que no puedes evitar que la función de calculadora muestre el resultado.

Ahora es el momento de realizar una devolución de llamada.

Devoluciones de llamadas de JavaScript

Una devolución de llamada es una función que se pasa como argumento a otra función.

Usando una devolución de llamada, puede llamar a la función calculadora (myCalculator) con una devolución de llamada (myCallback), y dejar que la función calculadora ejecute la devolución de llamada una vez finalizado el cálculo:

Ejemplo

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

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

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

En el ejemplo anterior, myDisplayerse llama **función de devolución de llamada** .

Se pasa myCalculator()como **argumento** .

Nota

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

Derecha: myCalculator(5, 5, myDisplayer);

Equivocado:~~miCalculadora(5, 5, miDisplayer());~~

Ejemplo

```
// Create an Array
const myNumbers = [4, 1, -20, -7, 5, 9, -6];

// Call removeNeg with a callback
const posNumbers = removeNeg(myNumbers, (x) => x >= 0);

// Display Result
document.getElementById("demo").innerHTML = posNumbers;
```

```
// Keep only positive numbers
function removeNeg(numbers, callback) {
 const myArray = [];
 for (const x of numbers) {
 if (callback(x)) {
 myArray.push(x);
 }
 }
 return myArray;
}
```

En el ejemplo anterior, `(x) => x >= 0` es una **función de devolución de llamada** .

Se pasa removeNeg()como **argumento** .

¿Cuándo utilizar una devolución de llamada?

Los ejemplos anteriores no son muy interesantes.

Se simplifican para enseñarle la sintaxis de devolución de llamada.

Donde las devoluciones de llamadas realmente brillan es en las funciones asincrónicas, donde una función tiene que esperar a otra función (como esperar a que se cargue un archivo).