Módulo Javascript

Funciones Flecha y Callback

Objetivos: Conocer las funciones flecha y funciones callback en javascript, para qué sirven y cómo utilizarlas.

Funciones flecha (Arrow Functions)

Las funciones flecha (también conocidas como funciones de flecha gorda o arrow functions en inglés) son una forma más concisa de definir funciones. En lugar de la palabra clave **function**, se utilizan una sintaxis más breve utilizando una flecha **=>**.

La sintaxis básica de una función flecha es la siguiente:

```javascript
(param1, param2, ...) => {
  // cuerpo de la función
  return resultado;
}
```

En este ejemplo, **(param1, param2, ...)** son los parámetros de entrada de la función, y el cuerpo de la función se encuentra dentro de las llaves **{}**. Al igual que con las funciones normales, la palabra clave **return** se utiliza para devolver un resultado.

Por ejemplo, la siguiente función normal que suma dos números:

```javascript
function sumar(num1, num2) {
```

```
  return num1 + num2;
}
```

Se puede reescribir utilizando una función flecha de la siguiente manera:

JavaScript
```
let sumar = (num1, num2) => num1 + num2;
```

En este ejemplo, la función flecha se asigna a la variable **sumar**. La flecha => reemplaza la palabra clave **function** y el cuerpo de la función se encuentra después de la flecha sin necesidad de usar llaves **{}**. Como la función solo tiene una línea de código, la flecha también se puede usar para omitir la palabra clave **return**.

Las funciones flechas también son útiles para crear funciones anónimas. Por ejemplo, la siguiente función anónima que multiplica un número por dos:

JavaScript
```
let doble = function(num) {
  return num * 2;
}
```

Se puede reescribir utilizando una función flecha de la siguiente manera:

JavaScript
```
let doble = num => num * 2;
```

En este ejemplo, la función flecha no tiene paréntesis alrededor del parámetro de entrada, ya que solo hay un parámetro. Cuando se utiliza una sola línea de código, la función flecha también puede omitir las llaves **{}** y la palabra clave **return**.

En resumen, las funciones flecha son una forma más concisa de definir funciones en JavaScript. Son útiles para crear funciones anónimas y funciones de una sola línea.

Retrollamadas (Callback Functions)

Una función de retrollamada (callback) es una función que se pasa como argumento a otra función y que se ejecuta después de que la función principal haya terminado de hacer su trabajo.

Las funciones de retrollamada son comunes en eventos y llamadas asíncronas, como las solicitudes de Ajax o las operaciones de lectura/escritura de archivos. Son útiles porque permiten que el código se ejecute de manera asíncrona, lo que significa que otras partes del programa pueden continuar ejecutándose mientras se espera la respuesta de una operación asíncrona.

Las funciones de retrollamada pueden ser declaradas como funciones separadas o pueden ser definidas en línea como funciones anónimas. También pueden tomar argumentos adicionales que se pasan como parámetros en la función principal.

En general, las funciones de retrollamada son una forma poderosa de controlar el flujo de ejecución en JavaScript y se utilizan ampliamente en el desarrollo web.

Ejemplo de uso:

JavaScript

```
function sayHello(name, callback) {
  console.log("Hello, " + name + "!");
  callback();
}

function goodbye() {
  console.log("Goodbye!");
}

sayHello("John", goodbye);
```

En este ejemplo, la función **sayHello** toma dos argumentos: **name** (un string) y **callback** (una función). La función **sayHello** imprime "Hello, {nombre}!" en la consola y luego ejecuta la función de retrollamada **callback**.

La función **goodbye** simplemente imprime "Goodbye!" en la consola.

Cuando llamamos a **sayHello** con los argumentos **"John"** y **goodbye**, primero se imprimirá "Hello, John!" en la consola y luego se ejecutará la función de retrollamada **goodbye**, que imprimirá "Goodbye!" en la consola.

Uso de las callback functions en JavaScript

Las callback functions se utilizan comúnmente en JavaScript para manejar eventos y realizar tareas asincrónicas. Por ejemplo, podemos usar una callback function para cargar una imagen desde un servidor y mostrarla en una página web después de que se haya cargado.

JavaScript

```
function loadImage(url, callback) {
  const image = new Image();
  image.src = url;

  image.onload = function() {
    callback(image);
  };
}

loadImage("image.jpg", function(image) {
  document.body.appendChild(image);
});
```

En este ejemplo, la función loadImage recibe una URL y una callback function como argumentos. La función crea una nueva instancia de la clase Image y establece la URL como su fuente. Después, la función establece una callback function para el evento onload de la imagen. Cuando la imagen se carga, la función llama a la callback function con la imagen como argumento.

La callback function es una función anónima que recibe la imagen como argumento y la añade al cuerpo del documento. La callback function se llama solo después de que se haya cargado la imagen, lo que garantiza que la imagen se mostrará correctamente.