Programador JavaScript Avanzado

Unidad 7: Trabajo con Clases

Contactos: tecnologia@elearning-total.com
Web: www.elearning-total.com

Indice

Unidad 7: Trabajo con Clases

- Herencia

Objetivos

Que el alumno logre:

- Implementar el paradigma de programación orientada a objetos

Herencia

En JavaScript, la herencia de clases se realiza utilizando la palabra clave extends para establecer una relación de herencia entre dos clases. Esta funcionalidad fue introducida en ECMAScript 6 (ES6) y permite que una clase hija herede propiedades y métodos de una clase padre.

Aquí tienes un ejemplo básico de cómo se realiza la herencia de clases en JavaScript:

```
class Estudiante extends Persona {
 constructor(nombre, edad, curso) {
  super(nombre, edad);
  this.curso = curso;
 }

 estudiar() {
  console.log(`${this.nombre} está estudiando ${this.curso}.`);
 }
}

let estudiante = new Estudiante('Juan', 25, 'Matemáticas');
estudiante.saludar(); // Salida: Hola, soy Juan y tengo 25 años.
estudiante.estudiar(); // Salida: Juan está estudiando Matemáticas.
```

Esta herencia se logra mediante la palabra clave extends y el uso de super dentro del constructor de la clase hija para llamar al constructor de la clase padre. Las clases en JavaScript ofrecen una forma más estructurada y clara de trabajar con la programación orientada a objetos.

Es importante tener en cuenta que JavaScript no admite múltiple herencia de clases (una clase heredando de varias clases). Sin embargo, se pueden simular patrones de herencia múltiple utilizando composición o utilizando características como Mixins o composición de prototipos para lograr un comportamiento similar.

Diferencia entre Clase y Función

Las diferencias principales entre class y function en JavaScript son su sintaxis y su comportamiento en cuanto a la creación de objetos y el manejo de la herencia.

Sintaxis:

Class: La sintaxis de clase es una forma más declarativa y fácil de entender para crear objetos y definir métodos en JavaScript. Se utiliza la palabra clave class seguida del nombre de la clase y el cuerpo de la clase entre llaves {}.

Function: Las funciones en JavaScript pueden ser utilizadas como funciones regulares o como constructores para crear objetos. La sintaxis es más flexible, ya que puede ser una función anónima o nombrada, y puede contener lógica arbitraria.

```
// Sintaxis de clase
class Persona {
  constructor(nombre) {
    this.nombre = nombre;
  }
}

// Sintaxis de función
function Persona(nombre) {
  this.nombre = nombre;
}
```

Creación de objetos:

Class: Se utilizan las clases para instanciar objetos utilizando new. Los métodos definidos dentro de la clase se convierten en métodos en el prototipo del objeto.

Function: Las funciones pueden ser invocadas directamente para realizar tareas o utilizarse como constructores para crear objetos utilizando new. Sin embargo, no tienen la estructura específica de una clase y la declaración de métodos se hace de manera diferente.

```
// Creación de objeto usando una clase
let persona1 = new Persona('Juan');

// Creación de objeto usando una función constructora
let persona2 = new Persona('Ana');
```

Herencia:

Class: La sintaxis de clase permite la herencia mediante la palabra clave extends, lo que facilita la creación de relaciones de herencia entre clases.

Function: Para lograr la herencia utilizando funciones constructoras, se puede establecer la relación de herencia a través de prototipos, pero la sintaxis es menos clara y más verbosa.

```
// Herencia con clase
class Estudiante extends Persona {
  constructor(nombre, curso) {
    super(nombre);
    this.curso = curso;
  }
}

// Herencia con función constructora
function Estudiante(nombre, curso) {
  Persona.call(this, nombre);
  this.curso = curso;
}
```

En resumen, las clases (class) ofrecen una sintaxis más clara y concisa para definir objetos y métodos, así como un manejo más directo de la herencia en comparación con las funciones (function). Sin embargo, ambas formas tienen su lugar en JavaScript y pueden ser utilizadas según las necesidades específicas del proyecto o preferencias del desarrollador.

Resumen

En esta Unidad…

Trabajamos con Paradigmas

En la próxima Unidad…

Trabajaremos con clases