Programador JavaScript Avanzado

Unidad 7: Trabajo con Clases

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

Indice

Unidad 7: Trabajo con Clases

Objetivos

Que el alumno logre:

- Implementar el paradigma de programación orientada a objetos

Clases

Una función constructora en JavaScript es una función que se utiliza específicamente para crear objetos. Estas funciones se llaman "constructoras" porque se usan con el operador new para instanciar nuevas instancias u objetos basados en su estructura.

La convención en JavaScript es que las funciones constructoras se escriban con la primera letra en mayúscula, lo que ayuda a distinguirlas de otras funciones. Aquí tienes un ejemplo de una función constructora:

```
function Persona(nombre, edad) {
  this.nombre = nombre;
  this.edad = edad;
  this.saludar = function() {
    console.log(`Hola, soy ${this.nombre} y tengo ${this.edad} años.`);
  };
}

// Uso de la función constructora para crear un objeto
let persona1 = new Persona('Elena', 28);
let persona2 = new Persona('Carlos', 35);

persona1.saludar(); // Salida: Hola, soy Elena y tengo 28 años.
persona2.saludar(); // Salida: Hola, soy Carlos y tengo 35 años.
```

En este ejemplo, Persona es la función constructora. Cuando se usa con new, crea nuevos objetos basados en su estructura. Dentro de la función constructora, se asignan propiedades (nombre y edad) y métodos (saludar) al nuevo objeto usando this.

Es importante tener en cuenta que si defines métodos dentro de la función constructora como se muestra en este ejemplo, cada vez que se crea una nueva instancia, se crea una copia de esa función. En situaciones donde se crean muchas instancias, esto puede llevar a un uso ineficiente de la memoria, ya que cada objeto tendrá su propia copia de los métodos.

Para evitar esto y optimizar el uso de memoria, es común definir métodos compartidos entre todas las instancias utilizando el prototipo, en lugar de definirlos directamente dentro de la función constructora. Este enfoque permite que todas las instancias compartan el mismo método desde el prototipo, lo que ahorra memoria.

Operador this

En JavaScript, this es una palabra clave especial que hace referencia al contexto en el que se ejecuta el código en un determinado momento. El valor de this depende de cómo se llama una función y dónde se utiliza. Puede tomar diferentes valores en diferentes contextos:

En el contexto global:

- En el navegador, this hace referencia al objeto window.
- En Node.js, this en el ámbito global hace referencia al objeto global.

```
console.log(this === window); // En un navegador, esto devolverá 'true'
```

En una función:

Si una función se llama en el contexto de un objeto (método), this hace referencia al objeto al que pertenece el método.

```
const objeto = {
  mensaje: 'Hola',
  saludar() {
    console.log(this.mensaje);
  }
};

objeto.saludar(); // Salida: Hola
```

Cuando se usa this con new en una función constructora:

this hace referencia al nuevo objeto creado por el constructor.

```
function Persona(nombre) {
  this.nombre = nombre;
}

let persona = new Persona('Ana');
console.log(persona.nombre); // Salida: Ana
```

En funciones de flecha (=>):

En las funciones de flecha, this se mantiene léxicamente, lo que significa que toma el valor de this del contexto que lo rodea en el momento de su definición, no de su ejecución.

```
const objeto = {
  mensaje: 'Hola',
  saludar: function() {
    setTimeout(() => {
```

```
    console.log(this.mensaje);
  }, 1000);
 }
};

objeto.saludar(); // Salida: Hola (debido a la función de flecha)
```

Es esencial entender el contexto en el que se utiliza this en JavaScript, ya que puede cambiar dinámicamente dependiendo de cómo se invoca una función, y su comprensión es clave para escribir código eficiente y sin errores.

Clases

En JavaScript, una clase es un tipo de función especial introducida en ECMAScript 6 (ES6) para permitir la creación de objetos y su comportamiento, siguiendo un paradigma de programación orientada a objetos. Las clases proporcionan una sintaxis más clara y una forma más estructurada de definir objetos y trabajar con herencia.

A pesar de que JavaScript no tiene una implementación tradicional de clases como en lenguajes orientados a objetos clásicos (como Java o C++), la introducción de la sintaxis de clases en ES6 ofrece una forma más familiar de trabajar con la programación orientada a objetos.

Aquí hay un ejemplo básico de cómo se define una clase en JavaScript:

```
class Persona {
  constructor(nombre, edad) {
    this.nombre = nombre;
    this.edad = edad;
  }

  saludar() {
    console.log(`Hola, soy ${this.nombre} y tengo ${this.edad} años.`);
  }
}

// Crear una instancia de la clase Persona
let persona = new Persona('Elena', 30);
persona.saludar(); // Salida: Hola, soy Elena y tengo 30 años.
```

En este ejemplo, Persona es una clase que tiene un constructor para inicializar propiedades (nombre y edad) y un método saludar para mostrar un mensaje. Al crear una instancia de la clase (let persona = new Persona('Elena', 30);), se crea un nuevo objeto basado en la estructura definida en la clase.

Resumen

En esta Unidad…

Trabajamos con Paradigmas

En la próxima Unidad…

Trabajaremos con clases