JavaScript Orientado a Objetos

Manejo de Estructuras de Datos: las Clases

Introducción al Manejo de Estructuras de Datos Tipo Clase en JavaScript

JavaScript fue inicialmente diseñado como un lenguaje de scripting para mejorar la interactividad en navegadores web. Desde su concepción, ha evolucionado significativamente para convertirse en un lenguaje de programación **versátil y poderoso**.

Con la introducción del estándar ECMAScript 6 (también conocido como ES6 o ECMAScript 2015), se incorporaron mejoras significativas en la forma en que los desarrolladores pueden estructurar y organizar su código, incluido el manejo de estructuras de datos.

Introducción al Manejo de Estructuras de Datos Tipo Clase en JavaScript

Una de las características clave de ES6 es la introducción de las **clases**, que proporcionan una sintaxis más clara y orientada a objetos para la programación en JavaScript. Las clases permiten a los desarrolladores modelar **objetos y estructuras de datos de una manera más formal**, facilitando la creación y gestión de instancias de objetos.

En JavaScript, antes de la introducción de las clases con ECMAScript 6 (ES6), los objetos se creaban comúnmente utilizando **funciones constructoras y prototipos** en lugar de la sintaxis de clases. La diferencia principal radica en la sintaxis y en cómo se definen y heredan propiedades y métodos. Aquí tienes un ejemplo de cómo se instancia un objeto **persona.**

```
// Definición de una función constructora
function Persona(nombre, edad) {
  this.nombre = nombre;
  this.edad = edad;
}

// Agregar un método al prototipo de la función constructora
Persona.prototype.saludar = function() {
  console.log(`Hola, soy ${this.nombre} y tengo ${this.edad} años.`);
};

// Crear objetos utilizando la función constructora
const persona1 = new Persona("Juan", 25);
const persona2 = new Persona("Ana", 30);

// Llamada a métodos de los objetos
persona1.saludar(); // Salida: Hola, soy Juan y tengo 25 años.
persona2.saludar(); // Salida: Hola, soy Ana y tengo 30 años.
```

En el ejemplo anterior, Persona es una función constructora que se utiliza para crear objetos persona1 y persona2. Las propiedades (nombre y edad) se asignan utilizando el operador this dentro de la función constructora, y el método saludar se agrega directamente en la función constructora (tal y como ya has visto anteriormente) o al prototipo de la función para que todas las instancias hereden ese método. Ahora bien, aquí al lado tienes un ejemplo de cómo se crearía una clase de Persona.

```
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
const persona1 = new Persona("Juan", 25);

// Llamar al método saludar
persona1.saludar();
```

Introducción al Manejo de Estructuras de Datos Tipo Clase en JavaScript

Diferencias clave entre la sintaxis de clase y la de función constructora:

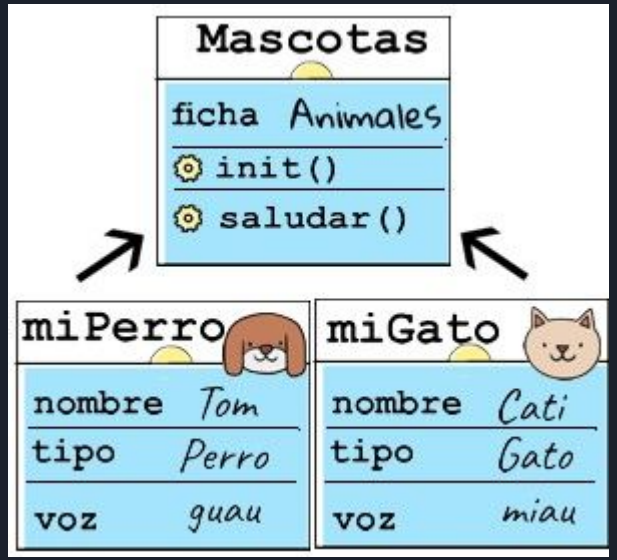
- **Sintaxis:** La sintaxis de clases proporciona una manera **más declarativa y estructurada** de definir objetos, mientras que las funciones constructoras utilizan una sintaxis más basada en funciones.
- **Herencia:** Las clases admiten una sintaxis más clara para la herencia, utilizando las palabras clave **extends y super**. En el caso de funciones constructoras, la herencia se logra a través de la cadena de prototipos.
- **Legibilidad:** La sintaxis de clases tiende a ser más legible y está más alineada con otros lenguajes de programación orientada a objetos, facilitando la comprensión del código para aquellos familiarizados con esos lenguajes.

En general, las clases en JavaScript introducidas en ES6 simplifican la creación de objetos y proporcionan una sintaxis más amigable y consistente para la programación orientada a objetos en comparación con las funciones constructoras y prototipos.

¿Qué es una clase?

Podemos definir una clase como una forma de organización de propiedades y métodos que van a ser aplicados a objetos concretos de modo que se tenga una estructura de datos más organizada.

Por ejemplo, una clase podría ser “mascotas”. Una vez definida dicha clase, objetos concretos de la misma podrían ser “perros” y “gatos”. En ambos casos, estos animales poseen propiedades como pueden ser nombre, tipo y voz. Podríamos añadir métodos a la clase como, por ejemplo, saludar().



¿Cómo instanciar una clase?

En primer lugar empleamos la palabra reservada **class** y le damos un nombre. A continuación, utilizamos un **constructor**, el cual permite además pasar parámetros de entrada. A continuación, podemos pasar esos parámetros como propiedades mediante la palabra reservada **this.** **Finalmente, para añadir métodos únicamente es necesario definir el nombre del método seguido de paréntesis así como la función a realizar entre llaves {}.**

```
class Mascotas {
  constructor(nombre, tipo, voz) {
    this.nombre = nombre;
    this.tipo = tipo;
    this.voz = voz;
  }
  // Método para que cada animal salude
  saludar() {
    alert(`${this.nombre}, es un ${this.tipo} y hace ¡${this.voz}!`);
  }
}

// Crear objetos perro y gato
const perro = new Mascotas("Tom", "perro", "guau");
const gato = new Mascotas("Whiskers", "gato", "miau");

// Imprimir información sobre el perro y el gato
perro.saludar();
gato.saludar();
```

Herencia de clases

Se entiende como herencia de clases a la característica donde una clase hija obtiene las propiedades y métodos de una clase padre mediante una relación establecida entre ambas por medio de la palabra clave **extends** y el método **super()**. Esta es una función especial que llama al constructor de la clase padre, por lo que antes de continuar, pasa a ejecutarse el constructor de la clase padre.

```
class Animal{
    constructor(){
        alert('Soy un animal');
    }
}
class Mascotas extends Animal {
  constructor(nombre, tipo, voz) {
      super();
      this.nombre = nombre;
      this.tipo = tipo;
      this.voz = voz;
  }
  saludar() {
    alert(`${this.nombre}, es un ${this.tipo} y hace ¡${this.voz}!`);
  }
}

// Crear objetos perro y gato
const pato = new Mascotas("Lucas", "pato", "cuac");
pato.saludar();
```

Reforzando conocimientos

https://www.youtube.com/watch?v=Rh1sHvUTyqg

Webgrafía

[1] “Lenguaje Javascript”, Lenguajejs.com. [En línea]. Disponible en: https://lenguajejs.com/javascript/ [Consultado: 12-nov-2023].

[2] “Clases y objetos en Javascript”, Espaciolatino.com. [En línea]. Disponible en: https://javascript.espaciolatino.com/lenguaje/clases-objetos.htm [Consultado: 13-nov-2023].

[3] P. el Destino, “Curso Javascript - CLASES (Class, Extends, Super) [ES6]”, 03-may-2020. [En línea]. Disponible en: https://www.youtube.com/watch?v=Rh1sHvUTyqg [Consultado: 13-nov-2023].