Módulo Javascript

Objetos y Clases

Objetivos: Conocer las características de los objetos y clases en JavaScript y cómo utilizarlos.

Objetos

En JavaScript, los objetos son un elemento clave, prácticamente todo es un objeto.
Los objetos son una estructura de datos que se utiliza para almacenar y organizar información en forma de pares clave-valor. Son una de las características fundamentales del lenguaje, y se utilizan ampliamente en la programación moderna.

Para crear un objeto en JavaScript, se puede utilizar la sintaxis literal de objetos, que consiste en un conjunto de llaves y un conjunto de pares clave-valor separados por comas. Por ejemplo:

```javascript
const persona = {
  nombre: "Juan",
  edad: 30,
  ciudad: "Madrid"
};
```

En este ejemplo, hemos creado un objeto llamado "persona" con tres propiedades: "nombre", "edad" y "ciudad". Cada propiedad está representada por un par clave-valor, donde la clave es el nombre de la propiedad y el valor es el valor asignado a la propiedad.

Las propiedades de un objeto pueden ser de diferentes tipos de datos, incluyendo números, cadenas, arreglos e incluso otros objetos. Por ejemplo:

```javascript
const persona = {
  nombre: "Juan",
  edad: 30,
  direccion: {
    calle: "Calle Mayor",
    numero: 10,
    ciudad: "Madrid"
  },
  amigos: ["Pedro", "Maria", "Luis"]
};
```

En este ejemplo, hemos añadido dos nuevas propiedades al objeto "persona". La propiedad "direccion" es a su vez otro objeto que contiene tres propiedades más: "calle", "numero" y "ciudad". La propiedad "amigos" es un arreglo que contiene los nombres de tres amigos de la persona.

Formas de crear objetos

Hay varias formas de crear objetos en JavaScript, por ejemplo:

1. Objeto literal: se define un objeto directamente dentro del código, utilizando llaves para delimitar las propiedades y valores separados por dos puntos. Ejemplo:

```javascript
const persona = {
  nombre: 'Juan',
  edad: 30,
  ciudad: 'Buenos Aires',
  saludar: function() {
    console.log(`Hola, mi nombre es ${this.nombre} y tengo
${this.edad} años.`);
  }
};
persona.saludar(); // Hola, mi nombre es Juan y tengo 30 años.
```

2. Función constructora: se define una función que sirve como molde para crear múltiples objetos con las mismas propiedades y métodos. Dentro de la función se utilizan las palabras clave this y new para asignar valores y crear nuevas instancias. Ejemplo:

JavaScript
```
function Persona(nombre, edad, ciudad) {
  this.nombre = nombre;
  this.edad = edad;
  this.ciudad = ciudad;

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

const persona1 = new Persona('Juan', 30, 'Buenos Aires');
const persona2 = new Persona('María', 25, 'Córdoba');

persona1.saludar(); // Hola, mi nombre es Juan y tengo 30 años.
persona2.saludar(); // Hola, mi nombre es María y tengo 25 años.
```

3. Clases: a partir de ES6 se introdujo una sintaxis más clara y orientada a objetos para crear objetos con funciones constructoras. Se utiliza la palabra clave class para definir la estructura de la clase y constructor para asignar valores iniciales a las propiedades. Ejemplo:

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

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

const persona1 = new Persona('Juan', 30, 'Buenos Aires');
const persona2 = new Persona('María', 25, 'Córdoba');

persona1.saludar(); // Hola, mi nombre es Juan y tengo 30 años.
persona2.saludar(); // Hola, mi nombre es María y tengo 25 años.
```

En todos los casos, los objetos creados pueden acceder a sus propiedades y métodos utilizando la sintaxis de punto o corchetes. Ejemplo:

JavaScript

```
console.log(persona.nombre); // Juan
console.log(persona['edad']); // 30
```

Además, los objetos en JavaScript son dinámicos, lo que significa que se pueden agregar, modificar o eliminar propiedades y métodos en tiempo de ejecución. Ejemplo:

JavaScript

```
persona.email = 'juan@gmail.com';
console.log(persona.email); // juan@gmail.com

delete persona.ciudad;
console.log(persona.ciudad); // undefined
```

Property value shorthand

El shorthand de valores de propiedad es una característica de JavaScript que permite crear objetos de manera más concisa y legible. Esta característica se introdujo en la versión de ECMAScript 6 y permite definir propiedades de objetos de forma abreviada, lo que reduce la cantidad de código necesario para crear objetos.

La sintaxis para utilizar el shorthand de valores de propiedad es la siguiente:

```JavaScript
let x = 10;
let y = 20;

let objeto = {x, y};
```

En este ejemplo, se ha creado un objeto utilizando el shorthand de valores de propiedad. Las variables x e y se utilizan como nombres de propiedad en el objeto, y sus valores se toman de las variables x e y. Esto es equivalente a escribir lo siguiente:

```JavaScript
let objeto = {x: x, y: y};
```

Otra forma de utilizar el shorthand de valores de propiedad es cuando el nombre de la propiedad y el nombre de la variable tienen el mismo nombre. En este caso, podemos omitir el nombre de la propiedad y simplemente incluir el nombre de la variable:

```JavaScript
let nombre = 'Juan';
let edad = 25;

let objeto = {nombre, edad};
```

Este ejemplo es equivalente a escribir lo siguiente:

```javascript
let objeto = {nombre: nombre, edad: edad};
```

El shorthand de valores de propiedad también se puede utilizar con métodos de objeto. Por ejemplo:

```javascript
let objeto = {
  nombre: 'Juan',
  edad: 25,
  saludar() {
    console.log(`Hola, mi nombre es ${this.nombre} y tengo ${this.edad} años`);
  }
};

objeto.saludar(); // salida: Hola, mi nombre es Juan y tengo 25 años
```

En este ejemplo, se ha creado un método de objeto utilizando la sintaxis de función abreviada. Esto es equivalente a escribir lo siguiente:

```javascript
let objeto = {
  nombre: 'Juan',
  edad: 25,
  saludar: function() {
    console.log(`Hola, mi nombre es ${this.nombre} y tengo ${this.edad} años`);
  }
};
```

```
objeto.saludar(); // salida: Hola, mi nombre es Juan y tengo 25
años
```

En resumen, el shorthand de valores de propiedad es una característica de JavaScript que nos permite crear objetos de forma más concisa y legible. Se utiliza ampliamente en aplicaciones web y en el desarrollo de software en general para reducir la cantidad de código necesario para crear objetos y métodos.

Clases

Las clases son una característica de JavaScript que se introdujo en la versión de ECMAScript 6. Las clases proporcionan una forma más sencilla y legible de crear objetos y definir su comportamiento en lugar de utilizar la sintaxis de prototipos heredados de versiones anteriores de JavaScript.

La sintaxis para crear una clase en JavaScript es la siguiente:

JavaScript

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

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

En este ejemplo, se ha creado una clase Persona con un constructor que inicializa las propiedades nombre y edad. También se ha definido un método saludar que imprime un mensaje en la consola.

Para crear un objeto de la clase Persona, podemos utilizar la sintaxis de new seguida del nombre de la clase y los argumentos del constructor:

```JavaScript
let persona1 = new Persona('Juan', 25);
let persona2 = new Persona('María', 30);

persona1.saludar(); // salida: Hola, mi nombre es Juan y tengo 25 años
persona2.saludar(); // salida: Hola, mi nombre es María y tengo 30 años
```

También podemos extender una clase para crear una subclase con funcionalidad adicional. Por ejemplo:

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

  estudiar() {
    console.log(`Estoy estudiando ${this.carrera}`);
  }
}

let estudiante1 = new Estudiante('Juan', 20, 'Ingeniería');
estudiante1.saludar(); // salida: Hola, mi nombre es Juan y tengo 20 años
estudiante1.estudiar(); // salida: Estoy estudiando Ingeniería
```

En este ejemplo, se ha creado una clase Estudiante que extiende la clase Persona. La subclase Estudiante tiene un constructor que toma un argumento adicional carrera y un método estudiar que imprime un mensaje en la consola.

Al crear un objeto de la clase Estudiante, se ejecuta tanto el constructor de la clase Estudiante como el constructor de la clase Persona a través de la llamada a super(nombre, edad). Esto permite que la subclase tenga acceso a las propiedades y métodos de la clase padre.

En resumen, las clases son una característica de JavaScript que nos permite crear objetos y definir su comportamiento de manera más sencilla y legible que utilizando la sintaxis de prototipos heredados. Las clases se utilizan ampliamente en aplicaciones web y en el desarrollo de software en general para crear estructuras de objetos complejas y fáciles de mantener.