Clases y Objetos

Tema 08

Trabajando con objetos

JavaScript está diseñado en un paradigma simple basado en objetos. Un objeto es una colección de propiedades, y una propiedad es una asociación entre un nombre (o *clave*) y un valor.

Visión general sobre Objetos

Los objetos en JavaScript, como en tantos otros lenguajes de programación, se pueden comparar con objetos de la vida real. El concepto de Objetos en JavaScript se puede entender con objetos tangibles de la vida real.

En JavaScript, un objeto es un entidad independiente con propiedades y tipos. Compáralo con una taza, por ejemplo. Una taza es un objeto con propiedades. Una taza tiene un color, un diseño, un peso, un material del que está hecha, etc. Del mismo modo, los objetos de JavaScript pueden tener propiedades que definan sus características.

Objetos y propiedades

Un objeto de JavaScript tiene propiedades asociadas a él. Una propiedad de un objeto se puede explicar como una variable adjunta al objeto. Las propiedades de un objeto básicamente son lo mismo que las variables comunes de JavaScript, excepto por el nexo con el objeto. Las propiedades de un objeto definen las características del objeto. Accedes a las propiedades de un objeto con una simple notación de puntos:

JS

```
objectName.propertyName;
```

Objeto Literal

Primero, necesitamos ver la diferencia entre estructuras de datos (data structures) y objetos orientados a objetos. Las estructuras de datos tienen datos públicos y ningún comportamiento. Esto significa que no tienen métodos.

Podemos fácilmente crear tales objetos usando la sintaxis de objeto literal. Se ve de esta manera:

```
const producto = {
  nombre: 'manzana',
  categoria: 'frutas',
  precio: 1.99
}

console.log(producto);
```

JS

```
var myCar = {
  make: "Ford",
  model: "Mustang",
  year: 1969,
};
```

```
const producto = {
  nombre: 'manzana',
  categoria: 'frutas',
  precio: 1.99
}

console.log(producto);
```

Los objetos en JavaScript son colecciones dinámicas de pares clave-valor. La clave es siempre una cadena y debe ser única en la colección. El valor puede ser una primitiva, un objeto o incluso una función.

Podemos acceder a una propiedad usando el punto o la notación cuadrada.

```
console.log(producto.nombre);
//"manzana"

console.log(producto["nombre"]);
//"manzana"
```

Aquí hay un ejemplo de donde valor es el otro objeto.

```
const producto = {
  nombre: 'manzana',
  categoria: 'frutas',
  precio: 1.99,
  nutrientes : {
   carbs: 0.95,
   grasas: 0.3,
   proteina: 0.2
 }
}
```

El valor de la propiedad `carbs` es un nuevo objeto. Aquí es como nosotros podemos acceder a la propiedad `carbs` .

```
console.log(producto.nutrientes.carbs);
//0.95
```

Usar una función constructora

Como alternativa, puedes crear un objeto con estos dos pasos:

1. Definir el tipo de objeto escribiendo una función constructora. Existe una fuerte convención, con buena razón, para utilizar en mayúscula la letra inicial.
2. Crear una instancia del objeto con el operador `new`.

Para definir un tipo de objeto, crea una función para el objeto que especifique su nombre, propiedades y métodos. Por ejemplo, supongamos que deseas crear un tipo de objeto para coches. Quieres llamar `Car` a este tipo de objeto, y deseas que tenga las siguientes propiedades: `make`, `model` y `year`. Para ello, podrías escribir la siguiente función:

```
function Car(make, model, year) {
  this.make = make;
  this.model = model;
  this.year = year;
}
```

JS

```
function Car(make, model, year) {
  this.make = make;
  this.model = model;
  this.year = year;
}
```

Observa el uso de `this` para asignar valores a las propiedades del objeto en función de los valores pasados a la función.

JS

```
var kenscar = new Car("Nissan", "300ZX", 1992);
var vpgscar = new Car("Mazda", "Miata", 1990);
```

<s0>Un objeto puede tener una propiedad que en sí misma es otro objeto. Por ejemplo, supongamos que defines un objeto llamado `person` de la siguiente manera:

JS

```
function Person(name, age, sex) {
  this.name = name;
  this.age = age;
  this.sex = sex;
}
```

y luego instancias dos nuevos objetos `person` de la siguiente manera:

JS

```
var rand = new Person("Rand McKinnon", 33, "M");
var ken = new Person("Ken Jones", 39, "M");
```

Declaración de clases

Una manera de definir una clase es mediante una **declaración de clase**. Para declarar una clase, se utiliza la palabra reservada `class` y un nombre para la clase "Rectangulo".

JS

```
class Rectangulo {
  constructor(alto, ancho) {
    this.alto = alto;
    this.ancho = ancho;
  }
}

const cuadrado = new Rectangulo(10, 10);

console.log(cuadrado.area); // 100
```

```
class Punto {
  constructor(x, y) {
    this.x = x;
    this.y = y;
  }
```

```
const p1 = new Punto(5, 5);
const p2 = new Punto(10, 10);
```

Caso1

Declarar un clase Tienda que permita registrar:

- Nombre de la tienda.
- Dirección de la tienda.
- Propietario de la tienda.
- Rubro de la tienda.

Luego invocar al menos tres (3) objetos usando esta clase.

Caso 2

Declarar una clase Jugador que permita registrar nombre, número de camiseta, edad, y si está lesionado. Luego instanciar al menos cinco (5) objetos usando esta clase, y asociarlos a un array.

El método push puede recibir un objeto por parámetro, el cual se asociará al array empleado. Es posible agregar más propiedades a la clase jugador.