 A blog about objects and its internal representation in Javascript

In JavaScript, everything is an object. Objects in JavaScript are a fundamental concept, and they play a crucial role in the language. In this blog post, we will explore the internal representation of objects in JavaScript.

JavaScript is an object-oriented language. Everything in JavaScript is an object, including numbers, strings, functions, and even arrays. Objects in JavaScript are a collection of properties, and properties are a key-value pair. The value can be anything, including other objects, functions, or primitives. The keys are always strings.

JavaScript objects can be created using object literals or constructor functions. An object literal is a collection of key-value pairs enclosed in curly braces.

For example:

let person = { name:

'John',

age: 30

}

In the above code, `person` is an object with two properties, `name` and `age`. The

values of these properties are a string and a number, respectively.

Constructor functions are a way to create objects in JavaScript. A constructor function is

a special type of function that is used to create objects. To create a constructor function,

you use the `function` keyword and capitalize the first letter of the function name. For

example:

function Person(name, age) {

this.name = name;

this.age = age;

}

In the above code, we have created a constructor function called **Person**. This function takes two parameters, **name** and **age**. The **this** keyword is used to create properties on the newly created object. The **new** keyword is used to create a new instance of the **Person** object. For example:

let person1 = new Person('John', 30);

let person2 = new Person('Jane', 25);

In the above code, we have created two instances of the **Person** object, **person1** and **person2**. Each instance has its own set of properties.

Internally, objects in JavaScript are implemented as a collection of key-value pairs. Each object has a hidden property called a prototype, which is a reference to another object. The prototype object is used to provide inheritance in JavaScript.

When you try to access a property on an object, JavaScript will first look for that property on the object itself. If the property is not found on the object, JavaScript will look for the property on the object's prototype. This process continues up the prototype chain until the property is found or until the end of the prototype chain is reached.

JavaScript also provides a way to define methods on objects. A method is a function that is a property of an object. When you define a method on an object, the method can be called on the object itself. For example:

let person = {

name: 'John',

age: 30,

greet: function() {

console.log('Hello, my name is ' + this.name + ' and I am ' + this.age + ' years old.');

}

};

person.greet(); // logs "Hello, my name is John and I am 30 years old."

In the above code, we have defined a method called **greet** on the **person** object. The **this** keyword is used to reference the object itself within the method.

In conclusion, objects in JavaScript are a fundamental concept, and they play a crucial role in the language. Objects are a collection of properties, and properties are a key-value pair. Objects in JavaScript are implemented as a collection of key-value pairs, and each object has a hidden property called a prototype, which is a reference to another object. The prototype object is used to provide inheritance in JavaScript. Methods can also be defined on objects, which are functions that are properties of the