Objects and its internal representation

in JavaScript

One of the most significant data types in JavaScript are objects. They are used to represent real-world objects like people, automobiles, and houses. Additionally, they can be used to represent ideas that are more ethereal, such as games, websites, and databases.

In JavaScript, an object is a group of properties. There are names and values for each property. The value can be any JavaScript data type, including a string, number, boolean, object, or function. The name is a string.

Using the object literal syntax or the Object() constructor function, objects can be constructed. A more condensed method of constructing objects is the object literal syntax. It enables you to define the properties and their values without calling the Object() constructor function.

In JavaScript, a hash table serves as an object's internal representation. A data structure that associates keys with values is a hash table. The values are the property values, while the keys are the names of the properties. To easily look for properties by name, utilize the hash table.

An example of an object made using the object literal syntax is shown below:

```
let myObject = {

    name: "praveenkumar",
    age: 24,
    city: "salem"
  };
```

myObject has three properties: name, age, and city. "praveenkumar" is the string value for the name property, 24 is the number value for the age property, and "salem" is the string value for the city property.

The dot notation can be used to retrieve an object's property. For instance, you could use the following code to retrieve the name property's value:

```
let name = myObject.name;
Console.log(name); //output: praveenkumar
```

The name variable will now have the string value "praveenkumar" in it.

The bracket notation can also be used to retrieve an object's properties. When the property name is a string that contains spaces or other special characters, the bracket notation is helpful. For instance, you would enter the following code to determine the value of city property:

```
let city = myObject[“city”];
Console.log(city); //output: salem
```

The city variable will now have the string value "salem" in it.

The Object () constructor function can be used to generate objects in the following ways:

```
let myObject = new Object ();

myObject.name = " praveenkumar ";
myObject.age = 24
myObject.city = "salem";
```

MyObject is now constructed and has the same characteristics as the object created using the object literal syntax. Objects can also have methods. An object's associated function is known as a method. Actions on an object are carried out via methods.

The function keyword is used to specify methods. The method name, the list of parameters, and the method body are placed after the function keyword. The list of variables provided to the method when it is called is known as the parameter list. When a method is called, a block of code called the body of the method is run.

An example of a method defined for the myObject object is as follows:

```
let myObject = {
  name: " praveenkumar ",
  age: 24,
  city: "salem",
  greeting: function () {
    return `Hello, my name is ${this.name}`;
  }
};

const greeting = myObject.greeting();
console.log(greeting); //output: Hello, my name is praveenkumar
```

myObject's greeting () method is defined. There are no parameters required and a string is returned by the greeting () method.

An example of an object made using the new keyword syntax is shown below:

```
let person = new Object();
person.name = ' praveenkumar ';
person.age = 24;
person.characteristic = 'Introvert';
//Output: { name: 'praveenkumar', age: 24, characteristic: 'Introvert' }
```