**Objects and its internal representation in JavaScript**

In JavaScript, all values except primitive values are objects. A primitive value is a value that has no properties or methods.

Example:

* String (“Rishi”, ”Apple”, ”Chennai”)
* Number (10, 3.14)
* Boolean (true, false)
* Null
* Undefined
* Symbol (Symbol(‘description’);)
* BigInt (BigInt(9999), BigInt(“9999”))

Primitive values are immutable (they cannot be changed). If a = 9.81, you can change the value of a, but you cannot change the value of 9.81. Objects are important data types in JavaScript and are different from primitive data types. Primitive data types can contain only one value but objects can contain many values and are written as **key : value** pairs. Every object has some property associated with some value. These values can be accessed using these properties associated with them. Thus a JavaScript object is a collection of named values.

Example:

let person = {firstName:"Rishi", lastName:"Rahul", age:25, gender:"Male"};

const keyword is commonly used to declare objects.

Example:

const person = {firstName:"Rishi", lastName:"Rahul", age:25, gender:"Male"};

**Object Properties:**

The named values in JavaScript objects are called properties.

|  |  |
| --- | --- |
| **Property** | **Value** |
| firstName | Rishi |
| lastName | Rahul |
| age | 25 |
| gender | Male |

Methods are actions that can be performed on objects. Object properties can be both primitive values, other objects, and functions. An object method is an object property containing a function definition.

|  |  |
| --- | --- |
| **Property** | **Value** |
| firstName | Rishi |
| lastName | Rahul |
| age | 25 |
| gender | Male |
| fullName | function() {return this.firstName + " " + this.lastName;} |

Thus JavaScript objects are containers for named values, called properties and methods.

**Ways of creating a JavaScript Object:**

There are different ways to create new objects:

* Create a single object, using an object literal.
* Create a single object, with the keyword new.
* Define an object constructor, and then create objects of the constructed type.
* Create an object using Object.create().

**Using an Object Literal:**

* This is the easiest way to create a JavaScript Object.
* Using an object literal, we can both define and create an object in one statement.
* An object literal is a list of **name:value** pairs (like age:25) inside curly braces {}.

Example:

const person = {firstName:"Rishi", lastName:"Rahul", age:25, gender:"Male"};

Spaces and line breaks are not important. An object definition can span multiple lines.

Example:

const person = {

firstName: "Rishi",

lastName: "Rahul",

age: 25,

gender: "Male"

};

This example creates an empty JavaScript object, and then adds properties to it.

Example:

const person = {};

person.firstName = "Rishi";

person.lastName = "Rahul";

person.age = 25;

person.gender = "Male";

**Using the JavaScript Keyword new:**

In this method, we create a new JavaScript object using new Object(), and then add the properties.

Example:

const person = new Object();

person.firstName = "Rishi";

person.lastName = "Rahul";

person.age = 25;

person.gender = "Male";

The examples above do exactly the same. But the use of new Object() should be avoided. For readability, simplicity and execution speed, use the object literal method.

**JavaScript Objects are Mutable:**

Objects are mutable: They are addressed by reference, not by value. If person is an object, the following statement will not create a copy of person:

const x = person;

The object x is not a copy of person. The object x itself is a person. Both x and person are the same object. So any changes to x will also change person, because x and person are the same object.

Example:

const person = {

firstName:"Rishi",

lastName:"Rahul",

age:25,

gender:"Male"

}

const x = person;

x.age = 10;

In this case, the value of age gets changed to 10.