OBJECTS IN JAVASCRIPT:

* JavaScript, being a versatile and object-oriented programming language, places a special emphasis on objects. Objects play a fundamental role in structuring and organizing data.
* An object is a collection of properties, and a property is an association between a name (or key) and a value. A property's value can be a function, in which case the property is known as a method.
* In JavaScript, an object is a standalone entity, with properties and type. Compare it with a cup, for example. A cup is an object, with properties. A cup has a colour, a design, weight, a material it is made of, etc. The same way, JavaScript objects can have properties, which define their characteristics.
* **Objects in JavaScript may be defined as an unordered collection of related data, of primitive or reference types, in the form of “key: value” pairs.**
* **Syntax:**

new Object(value)  
Object(value)  
let object\_name = {  
 key\_name : value,  
 ...  
}

* An object can be created with figure brackets {…} with an optional list of properties. A **property** is a “key: value” pair, where a key is a string (also called a “property name”), and the value can be anything.
* For Eg. If your object is a student, it will have properties like name, age, address, id, etc and methods like updateAddress, updateNam, etc.

**OBEJECTS INTERNAL REPRESENTATION IN JAVASCRIPT:**

* **PROPERITIES (using an object literal):**

**Objects in JavaScript are essentially containers for named values called properties. These properties can hold various data types, including numbers, strings, functions, or even other objects.**

* + - **FOR EXAMPLE:**

**let Person = {**

**name: 'John',**

**age: 30,**

**sayHello: function() {**

**console.log('Hello, ' + this.name + '!');**

**}**

**};**

**Properties used here:**

**name: A property with the key 'name' and the value 'John'.**

**age: A property with the key 'age' and the value 30.**

**Output:**

**// Accessing properties**

**console.log(Person.name); // Output: 'John'**

**console.log(Person.age); // Output: 30**

* **METHODS:**

**An object method is a function definition that is stored within an object property.**

**i.e.,**

**Take the example program mentioned in properties:**

* + - **FOR EXAMPLE:**

**let Person = {**

**name: 'John',**

**age: 30,**

**sayHello: function() {**

**console.log('Hello, ' + this.name + '!');**

**}**

**};**

**sayHello: A method (function) with the key 'sayHello'. When invoked, it logs a greeting message using the console.log statement. The message includes the value of the name property.**

**Subsequent output of the statement**

**// Invoking the method**

**Person.sayHello(); // Output: 'Hello, John!'**

* **PROTOTYPES (using the object.create method):**
  + - * + **Objects in JavaScript are linked to a prototype object. If a property or method is not found on the object itself, JavaScript looks up the prototype chain until it finds the property or reaches the end of the chain.**
        + **Object.create method to establish a prototype relationship.**
        + **Also, can be called as using the object.create method**

**FOR EXAMPLE:**

**let animal = {**

**eat: function() {**

**console.log('Eating...');**

**}**

**};**

**let dog = Object.create(animal);**

**dog.bark = function() {**

**console.log('Woof!');**

**};**

**Breakdown of the code:**

**PROTOTYPE OBJECT: animal object:**

**‘animal’ is an object with a method named eat.**

**The ‘eat’ method logs a message to the console, indicating that the animal is eating.**

**dog Object:**

**‘dog’ is created using ‘Object.create(animal)’, establishing a prototype relationship. This means that ‘dog’ inherits properties and methods from the animal object.**

**Additionally, ‘dog’ has its own method named ‘bark’.**

**The ‘bark’ method logs a message to the console, indicating that the dog is barking.**

**OUTPUT: (Example usage)**

**// Invoking methods on the objects**

**animal.eat(); // Output: 'Eating...'**

**// The 'dog' object inherits the 'eat' method from the 'animal' object**

**dog.eat(); // Output: 'Eating...'**

**// Invoking the 'bark' method specific to the 'dog' object**

**dog.bark(); // Output: 'Woof!'**

* **OBJECT CONSTRUCTION:**
  + - * **Objects can be created using object literals, constructor functions, or the class syntax introduced in ECMAScript 2015.**
      * **A constructor, which is essentially a function, can be used with the new keyword to create instances of the same type, as demonstrated below.**

**FOR EXAMPLE:**

**// Object literal**

**let car = {**

**make: 'Toyota',**

**model: 'Camry',**

**year: 2022**

**};**

**// Constructor function**

**function Person(name, age) {**

**this.name = name;**

**this.age = age;**

**}**

**let Person = new Person('Alice', 25);**

**BREAKDOWN OF THE CODE:**

**IN OBJECT LITERAL SECTION:**

**Have created an object named ‘car’ using an object literal. It has three properties: ‘make’, ’model’, and ‘year’, each with a corresponding value. Object literals are a concise way to create simple objects with properties and values.**

**IN CONSTUCTOR FUNCTION:**

**Have defined a constructor function named ‘Person’. Constructor functions are used to create objects with similar structures. In this case, the ‘Person’ constructor takes two parameters (‘name’ and ‘age’) and sets them as properties of the newly created object using this.**

**you've created an instance of the ‘Person’ object named ‘Person’ using the new keyword. The ‘Person’ object has properties ‘name’ and ‘age’ based on the values passed to the constructor.**

**CONCLUSION:**

**Understanding the internal representation of objects in JavaScript is crucial for writing efficient and performant code. Whether you're dealing with simple object literals or complex class-based structures, a solid grasp of how objects work under the hood empowers you to make informed decisions when designing and optimizing your applications. Objects in JavaScript are not just data structures; they are the building blocks of expressive and dynamic web development.**