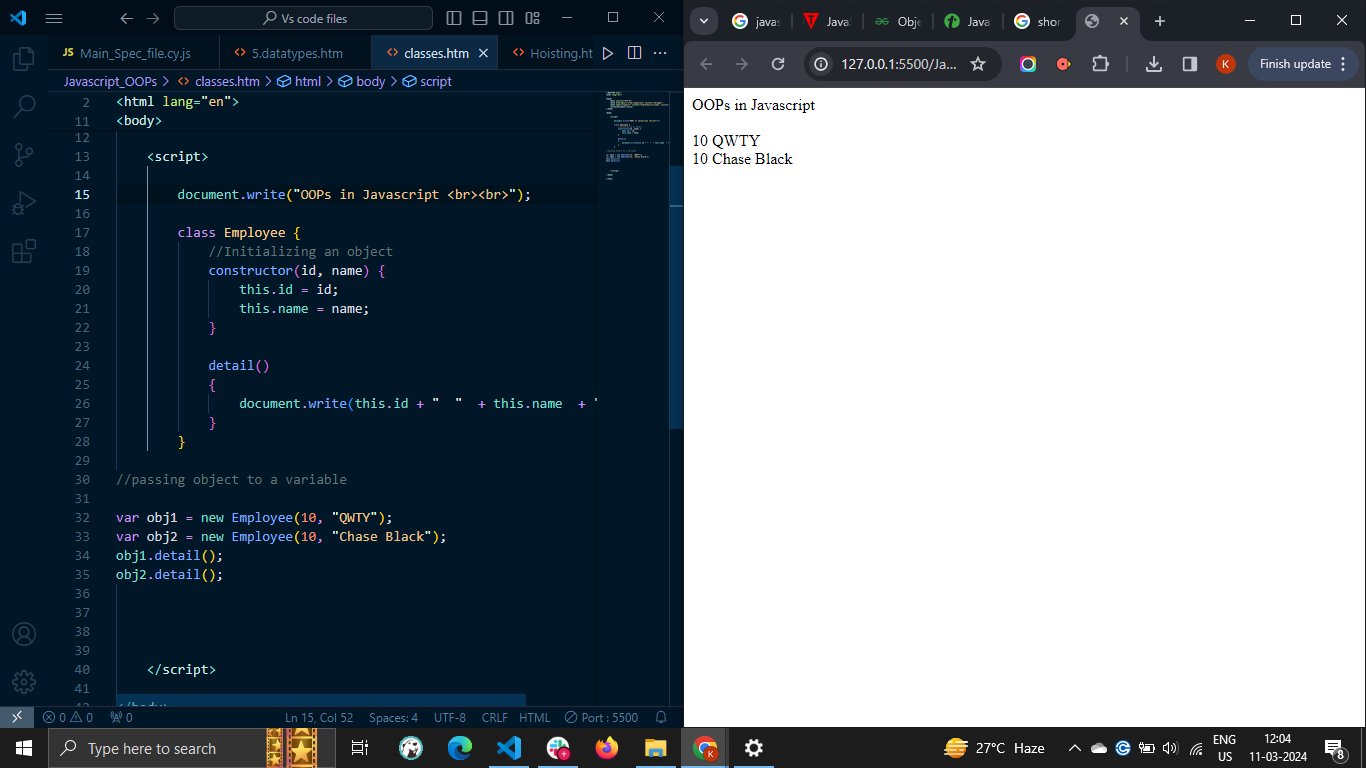
# JavaScript Classes

* In JavaScript, classes are a special type of functions. We can define the class just like function declarations and function expressions.
* The JavaScript class contains various class members within a body including methods or constructor. The class is executed in strict mode. So, the code containing the silent error or mistake throws an error.
* The JavaScript classes are a blueprint or template for object creation. They encapsulate the data and functions to manipulate that data.
* We can create the object using classes. Creating an object from a class is referred to as instantiating the class.
* In JavaScript, the classes are built on prototypes. The classes are introduced to JavaScript in ECMAScript 6 (ES6) in 2009.
* For example, you can think about writing code to represent the car entity. A code can contain the class having car properties. For different cars, you can create an instance of the class and initialize the car properties according to the car model.
* Before ES6, the constructor function was used to define a blueprint of the object. You can define the constructor function as shown below.
* The class syntax contains two components:
  + Class declarations
  + Class expressions

## Class Declarations

* A class can be defined by using a class declaration. A class keyword is used to declare a class with any particular name.
* According to JavaScript naming conventions, the name of the class always starts with an uppercase letter.



## The constructor() method

# When you use the function as an object blueprint, you can initialize the object properties inside the function body. Similarly, you need to use the constructor() method with the class to initialize the object properties.

# Whenever you create an instance of the class, it automatically invokes the constructor() method of the class.

# In below example, we use the constructor() method to create a Car class −

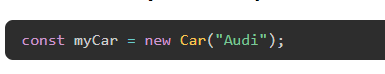
# 

The constructor() method has no specific name but can be created using the 'constructor' keyword. You can initialize the class properties using the 'this' keyword inside the constructor function.

## Creating JavaScript Objects

To create an object of a JavaScript class, we use **new** operator followed by the class name and a pair of parentheses. We can pass thee arguments to it also.

Let's create an object called **myCar** as follows −

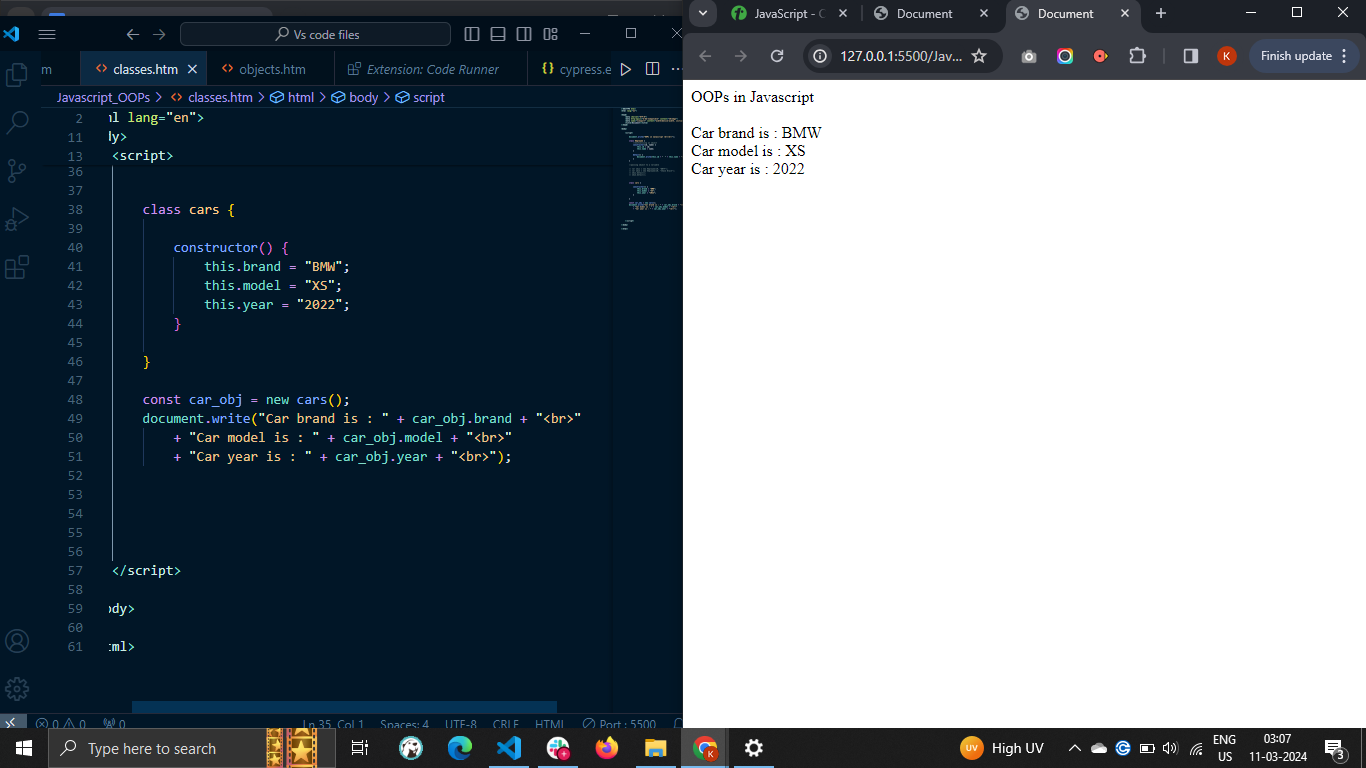


# The this keyword inside the constructor function refers to an object that is executing the current function.

### Example: Creating class objects without arguments

In the example below, we have defined the 'Car' class. The class contains the constructor and initializes the properties with default values.

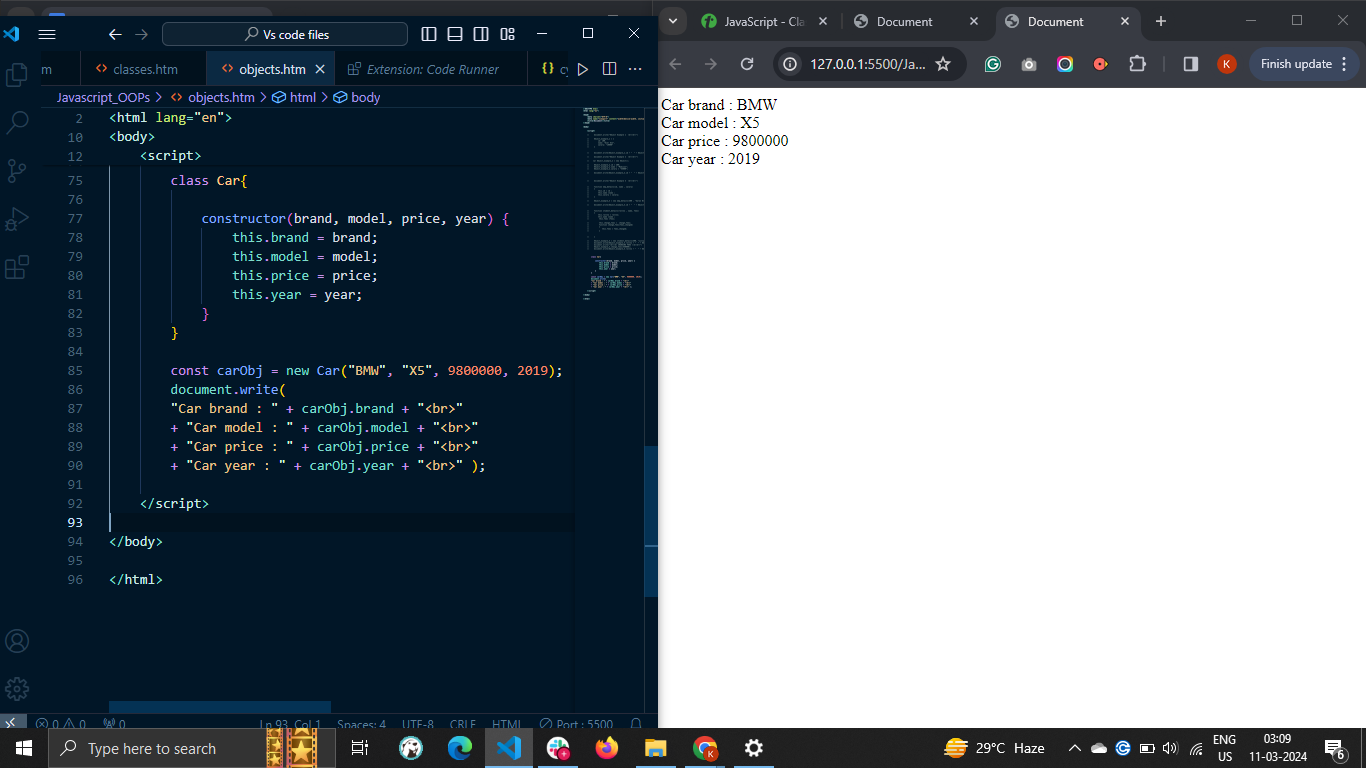
After that, we have created the instance of the class, and you can observe it in the output.



### Example: Creating class objects with arguments

In the example below, we have defined the 'Car' class. The constructor() method of the class takes 4 parameters and initializes the class properties with parametric values.

While creating the 'Car' class instance, we passed 4 arguments. In this way, you can initialize the class properties dynamically.



## JavaScript Class Methods

You can also define the methods inside the class, which can be accessed using the class instance.

### Syntax

Follow the syntax below to define methods inside the class.



In the above syntax, 'methodName' is a dynamic name of the method. To define a class method, you don't need to write any keyword like 'function' before the method name.

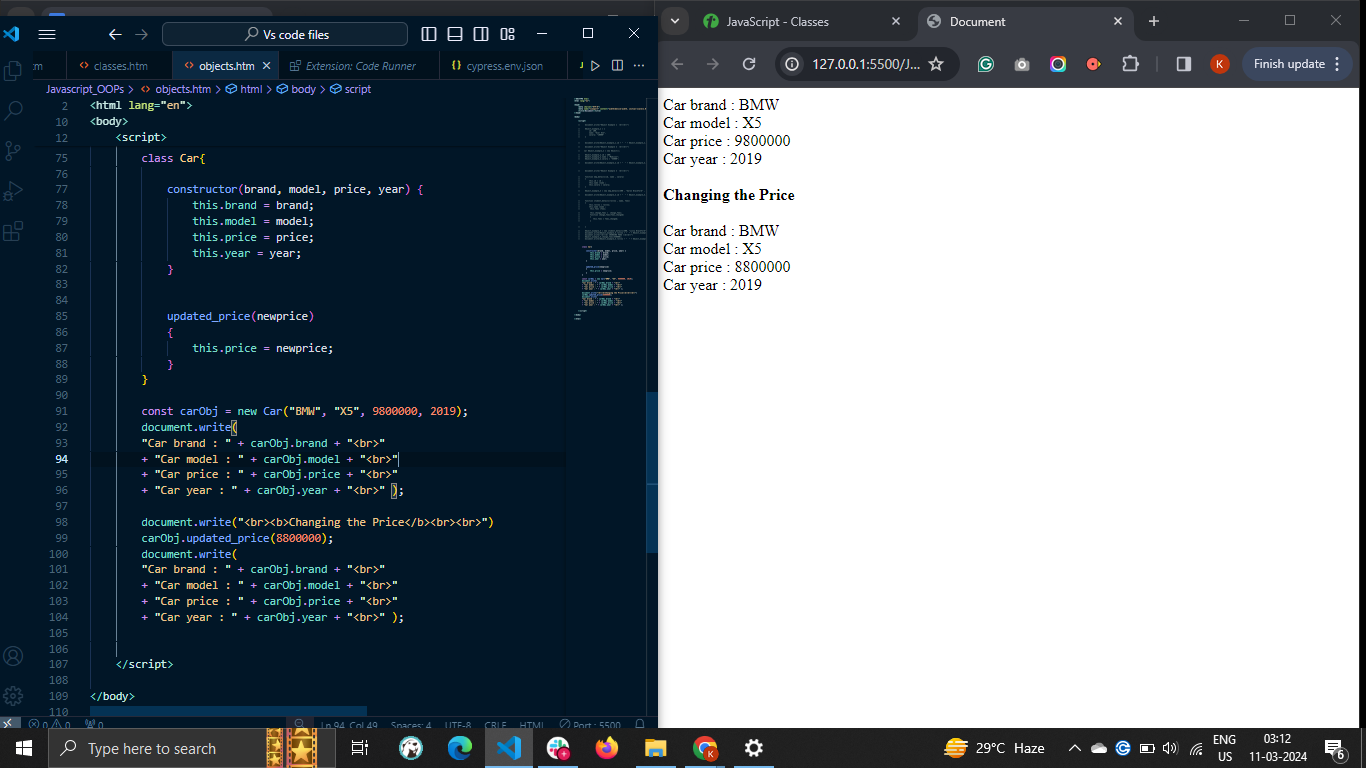
To invoke the class method, you need to use the instance of the class. Here, 'obj' is an instance of the class. You can also pass the parameters to the method.

### Example

The example below demonstrates how to pass parameters to the class methods.

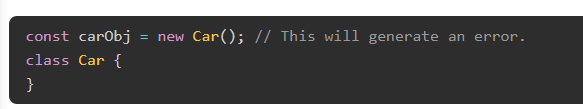
Here, we have defined the updateprice() method to update the price of the car. So, while invoking the updateprice() method, we pass the new price as an argument and use it inside the method body to update the price.

You can see the original and updated price of the car in the output.



## JavaScript Class Hoisting

In JavaScript, the declaration of the class is not hoisted at the top of the code. So, you always need to define the class before you use it.

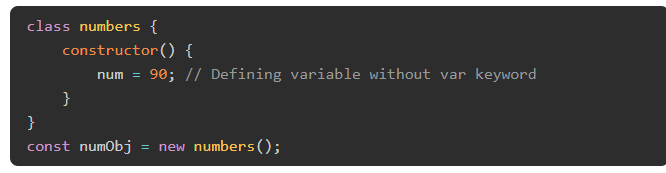


You can try to run the above code. It will generate a reference error as the car class is used before its initialization.

## Strict Mode with Classes

The strict mode is used to avoid unusual errors. The class code is always in the strict mode by default.

Let's understand it via the example below.

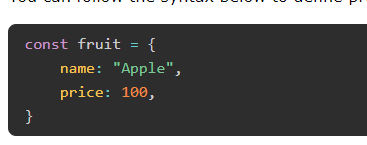


In the above code, we define the 'num' global variable in the constructor() method. In the strict mode of JavaScript, it is not allowed to define the variables without using the var, let, or const keywords. So, the above code will throw an error.

# JavaScript Objects

* A javaScript object is an entity having state and behavior (properties and method). For example: car, pen, bike, chair, glass, keyboard, monitor etc.
* JavaScript is an object-based language. Everything is an object in JavaScript.
* JavaScript is template based not class based. Here, we don't create class to get the object. But, we direct create objects.
* An **object property** in JavaScript is a key: value pair, where key is a string and value can be anything. The key in key: value pair is also called property name. So the **properties** are association between key (or name) and value.
* An object is in other terms a collection of properties (key: value pairs). However, key: value pairs are not stored in the specific order in the object. To write an object syntax, the curly braces are used. Each key: value pair is written within curly braces separated by a comma.
* You can manipulate the object properties in JavaScript. For example, you can add, delete, or update the object's properties.

### Syntax

* You can follow the syntax below to define properties in the object.
* 
* In the above syntax, fruit is an object. The fruit object contains the name and price properties. The value of the name property is 'Apple’, and the price is 100.
* In an object, the key can either be a string or a symbol only. If you use another data type as a key, the object implicitly converts it into the string.
* The property value can be anything like an object, set, array, string, set, function, etc.

## Creating Objects in JavaScript

There are 3 ways to create objects.

1. By object literal
2. By creating instance of Object directly (using new keyword)
3. By using an object constructor (using new keyword)

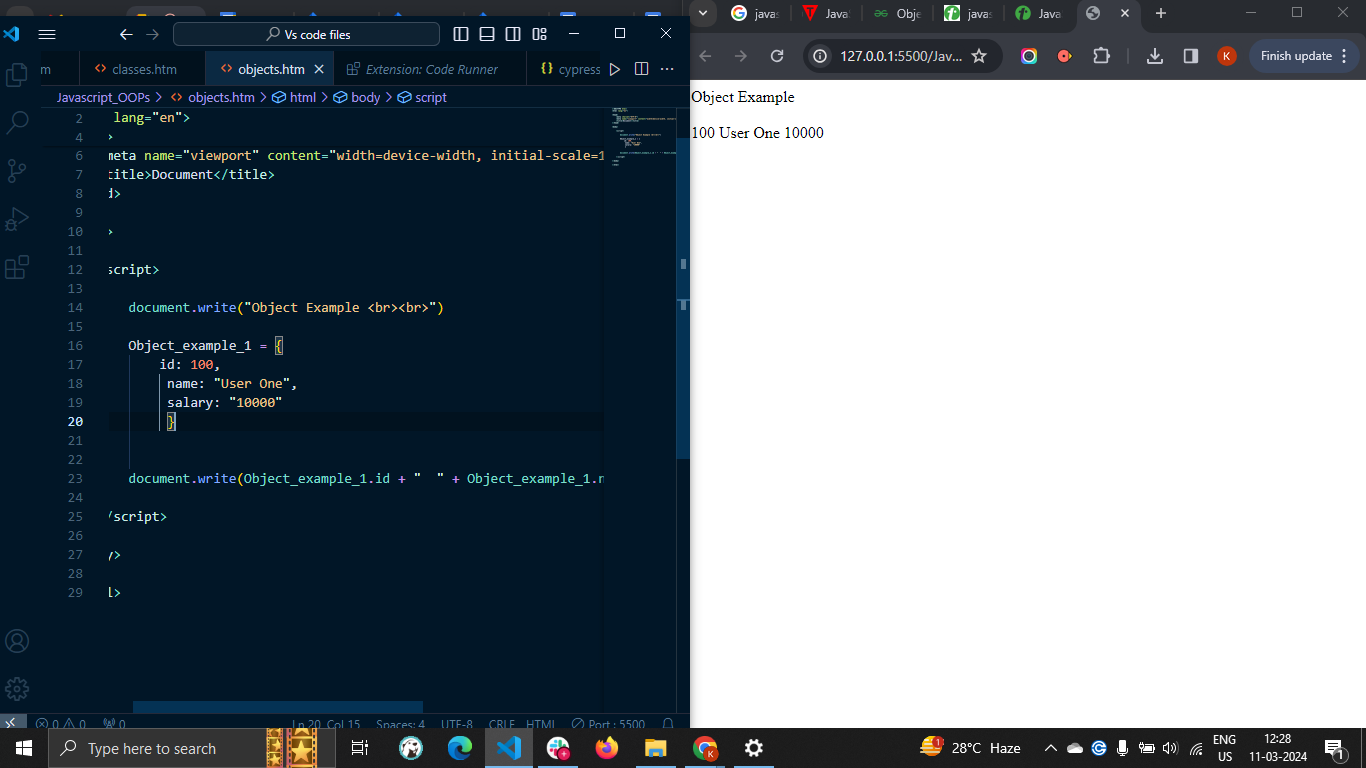
## 1) JavaScript Object by object literal

The syntax of creating object using object literal is given below:

1. object={property1:value1,property2:value2.....property:valueN}

As you can see, property and value is separated by : (colon).

Let’s see a simple example of creating object in JavaScript.

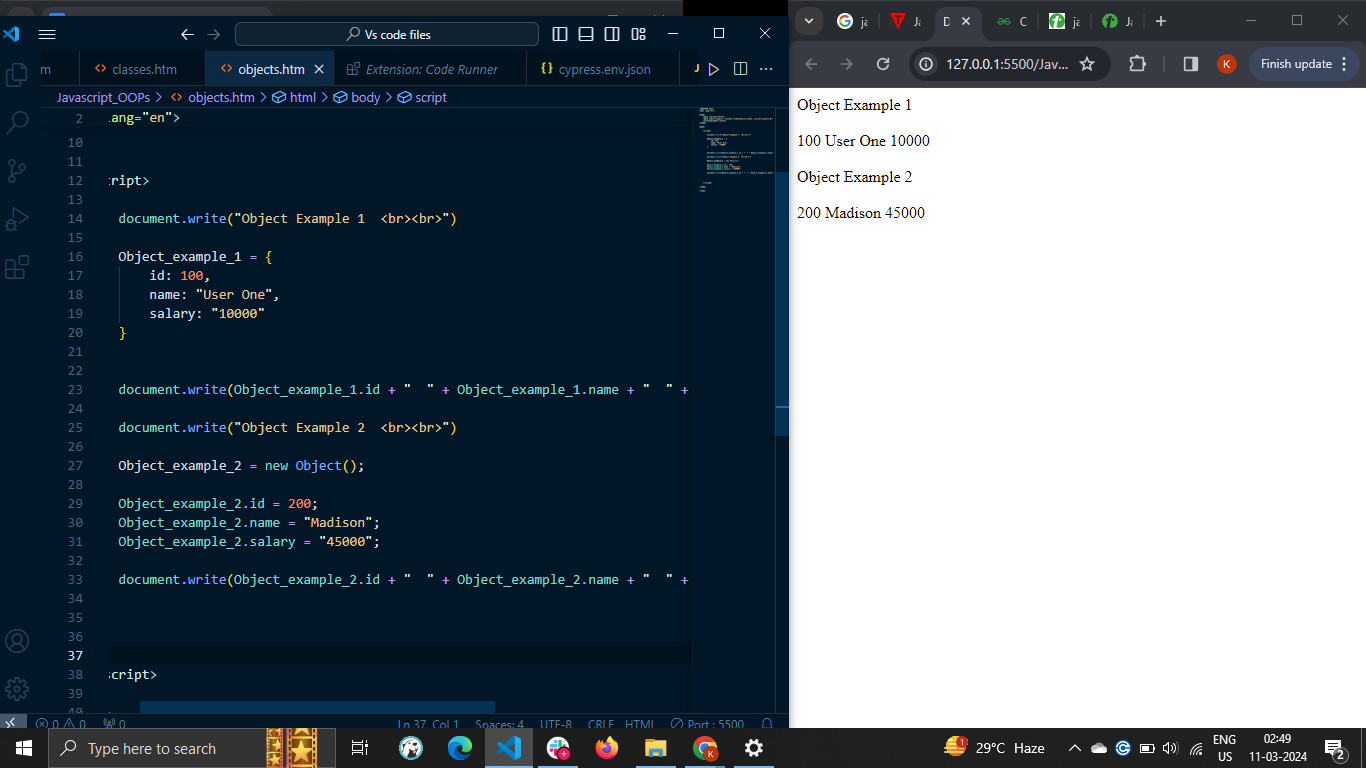
****

## 2) By creating instance of Object

**The syntax of creating object directly is given below:**

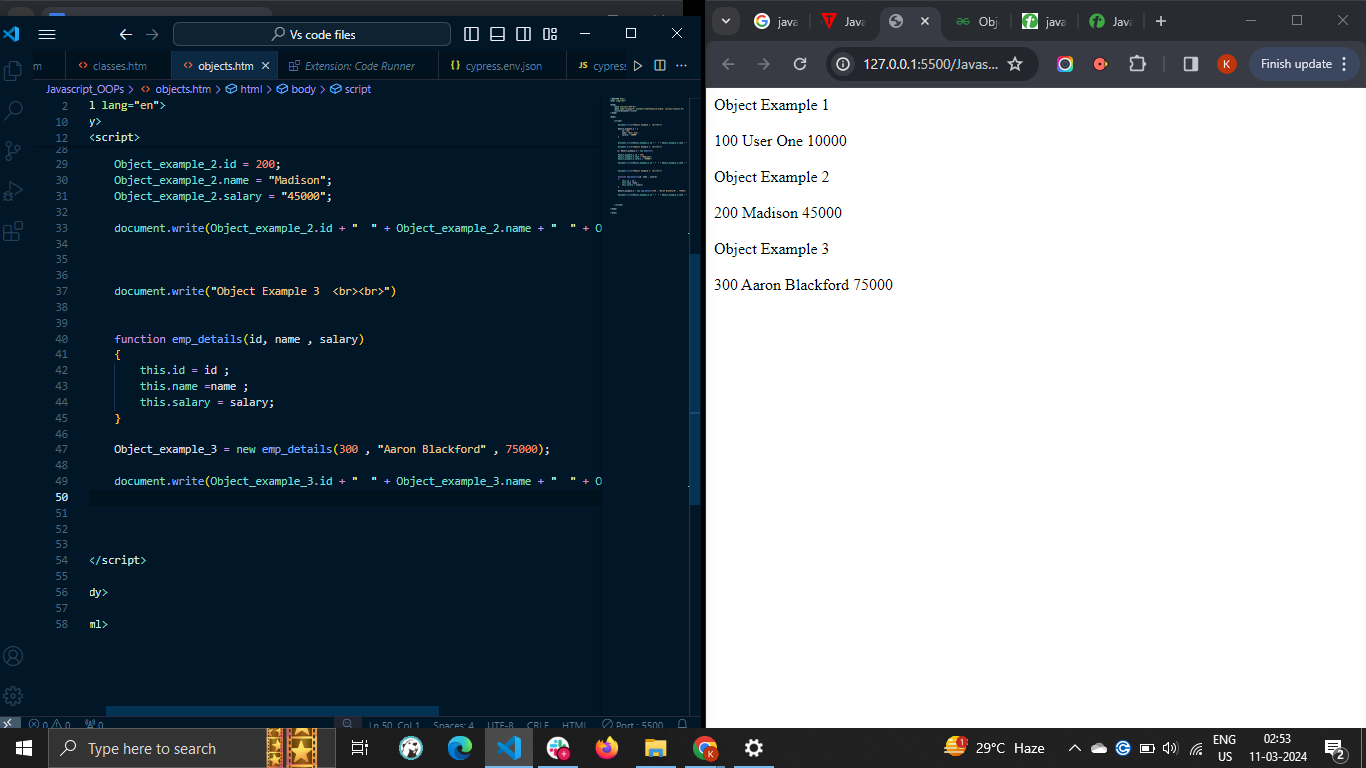
1. **var objectname=new Object();**

**Here, new keyword is used to create object.**

****

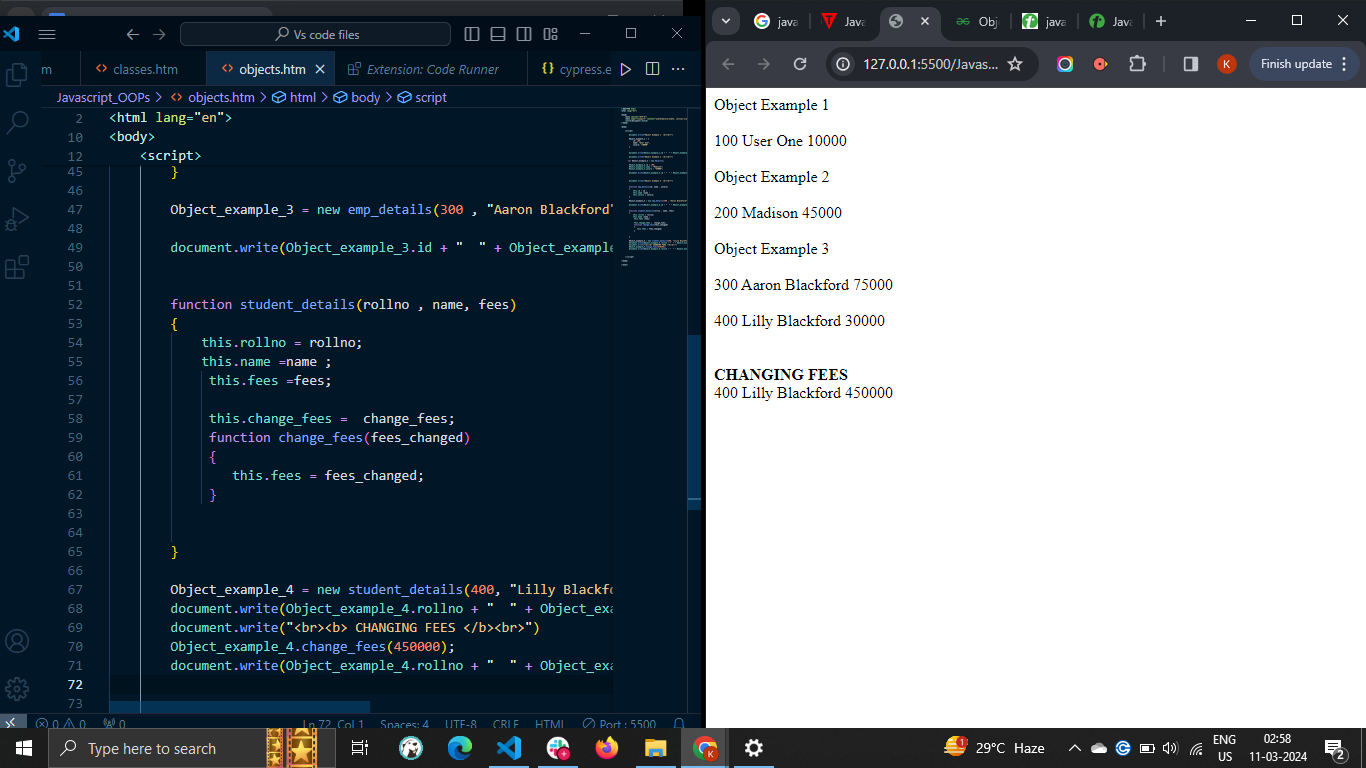
## 3) By using an Object constructor

* Here, you need to create function with arguments. Each argument value can be assigned in the current object by using this keyword.
* The this keyword refers to the current object.



## Defining method in JavaScript Object

We can define method in JavaScript object. But before defining method, we need to add property in the function with same name as method.



## Accessing Object Properties

There are 3 ways to access object properties in JavaScript.

* Using the dot notation
* Using the square bracket notation
* Using the expression

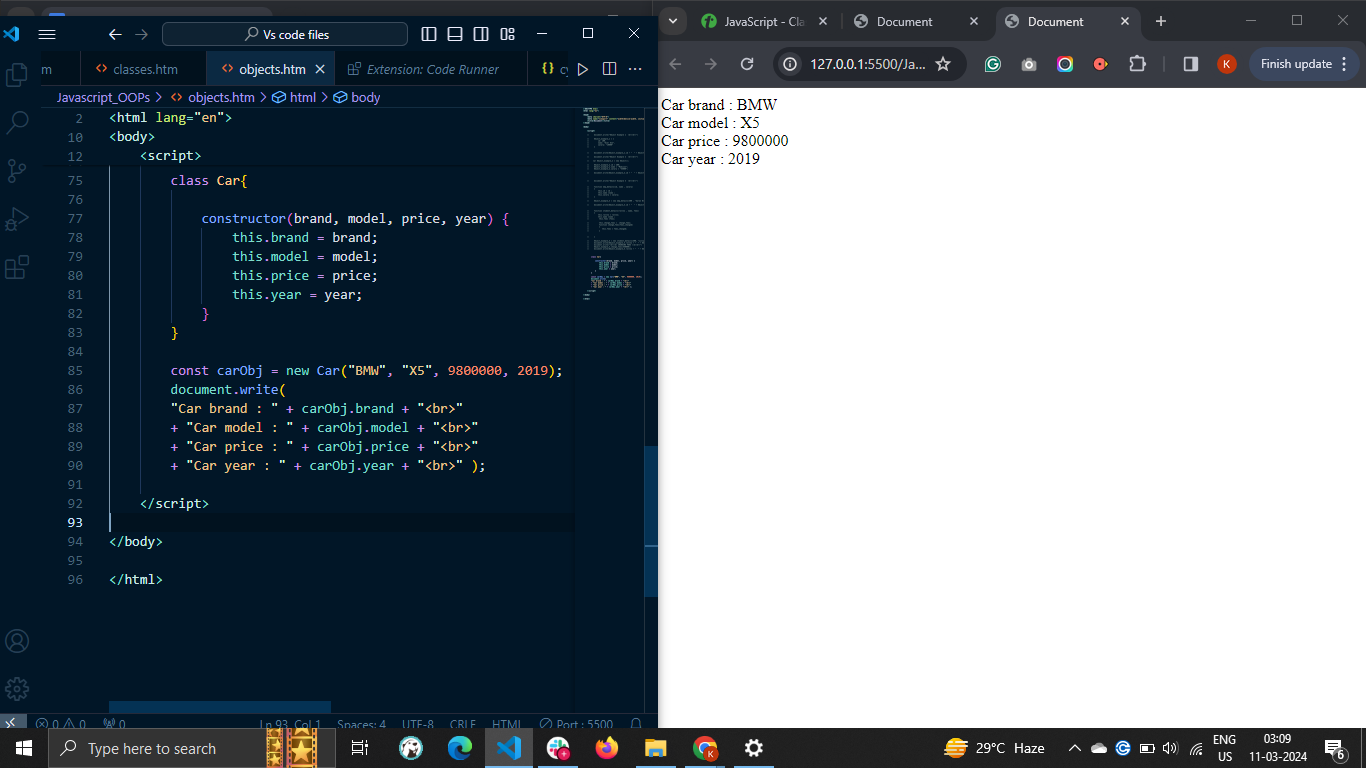
### The Dot Notation

You can access the object property using the dot notation/ syntax.

In the above syntax, 'obj' is an object, and 'prop' is its property whose value you need to access.

#### Example

The 'fruit' object in the example below contains the name and price property. We access the object properties using the dot notation, and you can see property values in the output.

****

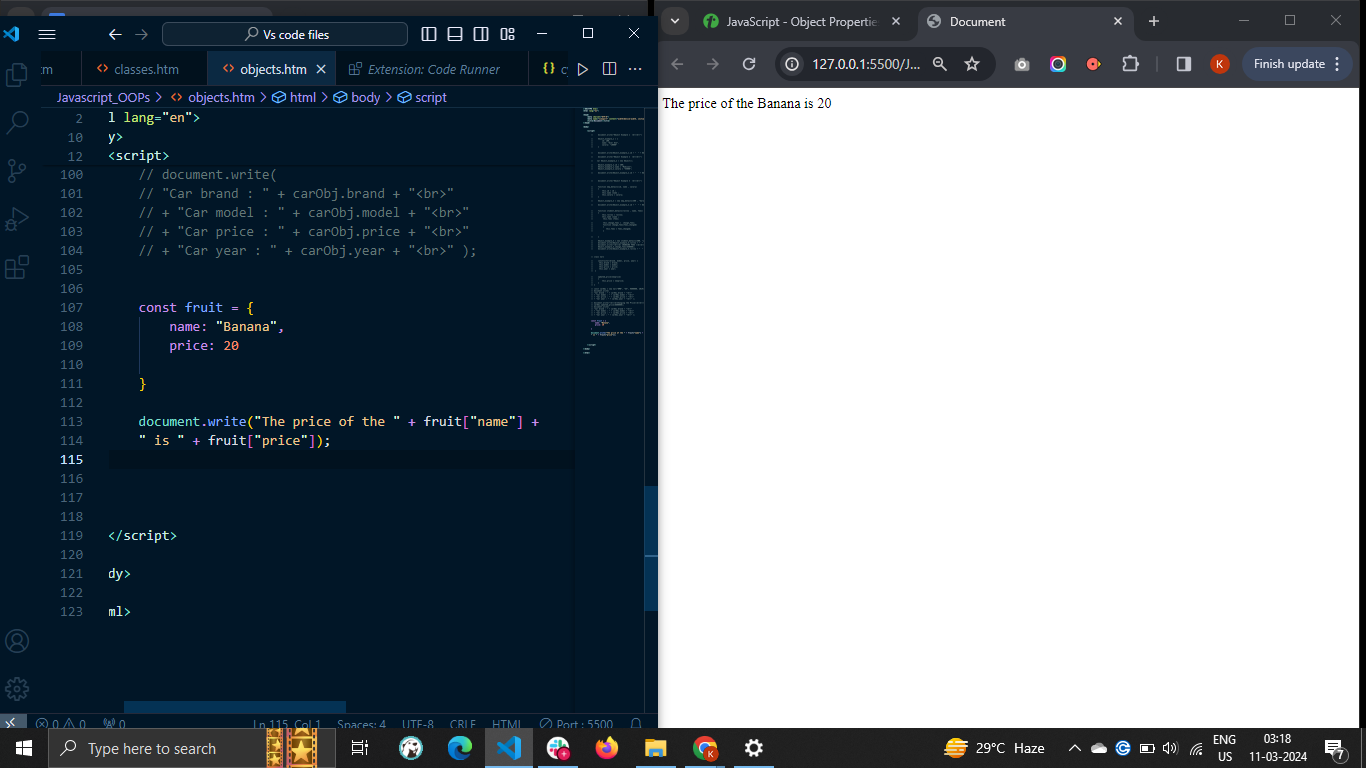
### The Square Bracket Notation

You can use the square bracket pair containing the property as a string followed by the object name to access a particular property.

****

In the above syntax, we access the 'prop' property from the object.

**NOTE :** You can't access the property using the dot notation when you use invalid identifiers as an object key. So, you need to use the square bracket notation. The identifier is invalid if it starts from a number, contains a space, or a hyphen.

****

### Using the expression inside the bracket

Sometimes, you require to access the object properties dynamically using the variable or expression. So, you can write the expression inside the square bracket notation. The expression can be a variable, a mathematical expression, etc.

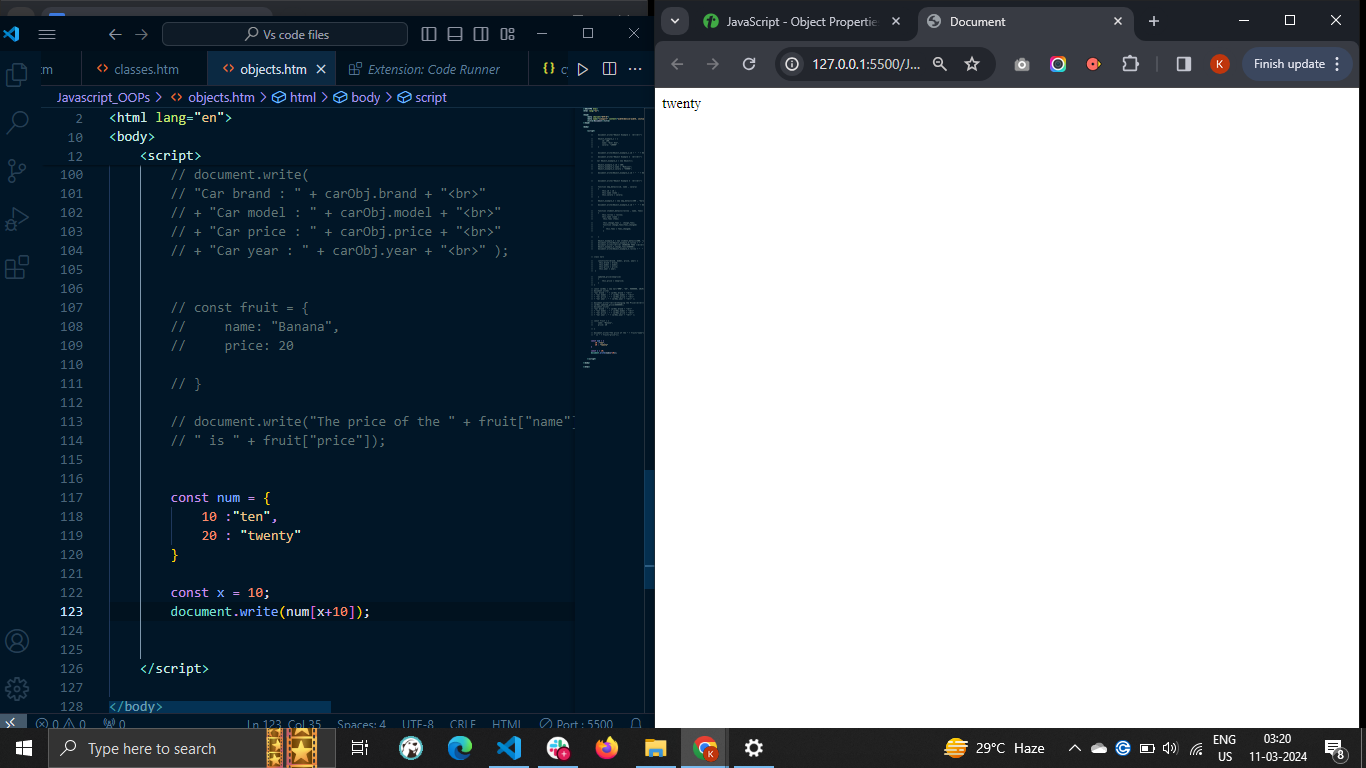
****

The above syntax evaluates the expression first and accesses the property same as a resultant value from the object. You don't need to write the expression in quotes.

#### Example

In the example below, the num object contains the number as a key in the string format and its word representation as a value.

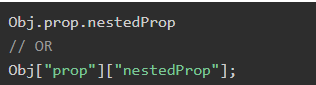
We use the variable x to access the property value from the object. Also, we used the "x + 10" mathematical expression to access the object property dynamically.

****

## Accessing the Nested Object Properties

Accessing the nested object properties is very similar to accessing the object properties. You can either use the dot or square bracket notation.

### Syntax

****

In the above syntax, the prop is a property of the obj object, and nestedProp is a property of the 'prop' object.

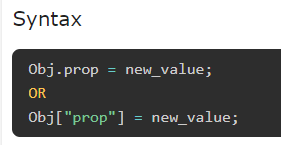
### Example

In the below code, the 'cars' object contains the nested objects named OD and BMW. We access the nested object properties using the dot and square bracket notation**.**

****

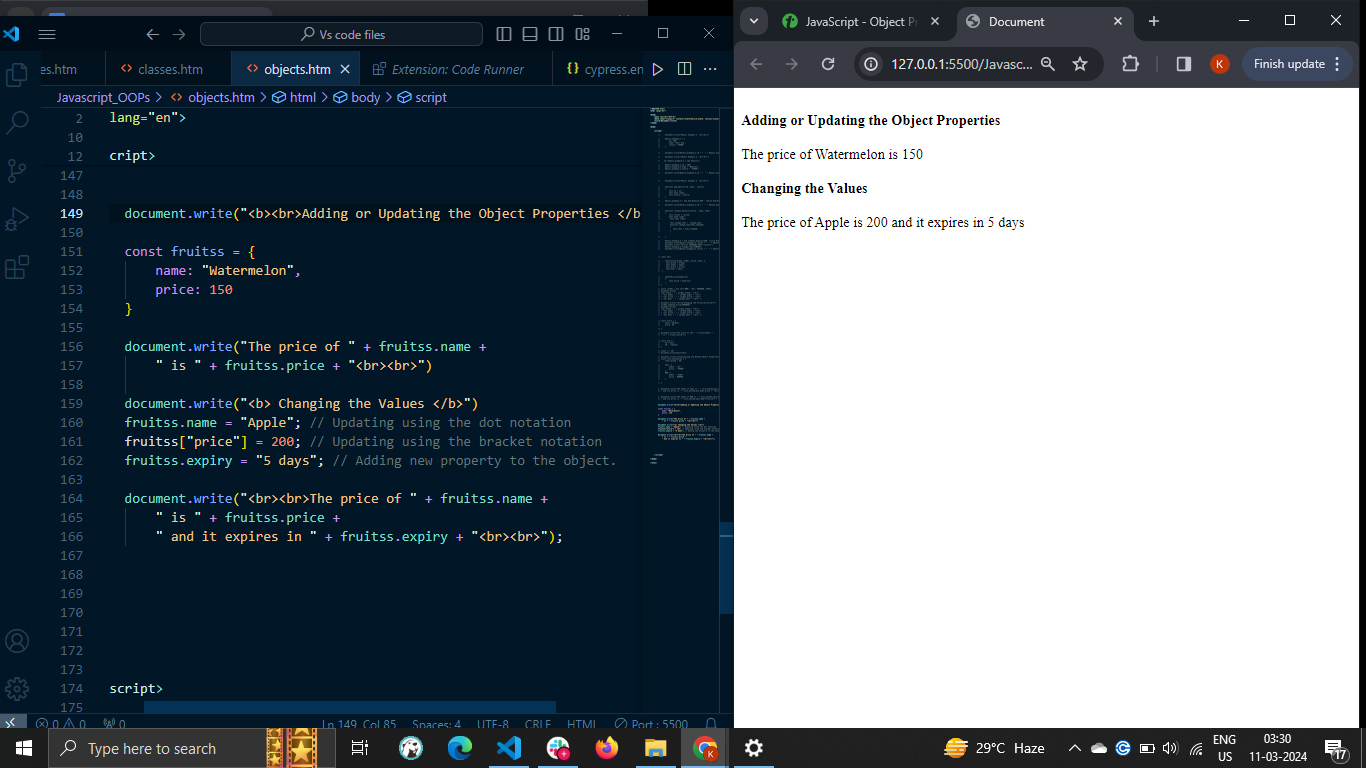
## Adding or Updating the Object Properties

You can update or add new properties to the object using the dot or square bracket notation. You can access the object property and assign a new value to it. If the property already exists, it updates the property value. Otherwise, it adds the property to the object.



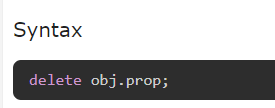
### Example

In the example below, we update the name and price property of the fruit object. Also, we add the expiry property to the fruit object.



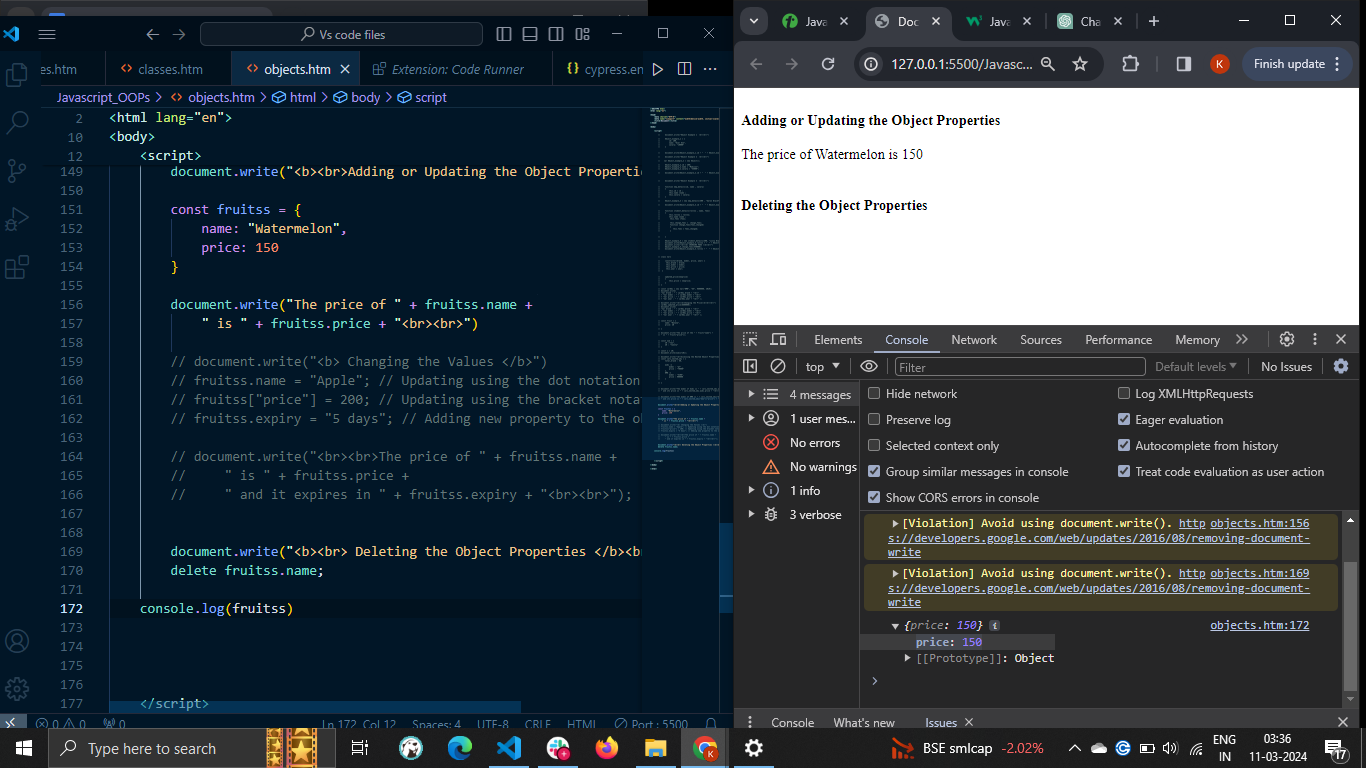
## Deleting the Object Properties

You can use the 'delete' operator to delete the specific object properties.



Example

In the example below, we delete the name property from the fruit object using the delete operator. The output shows that the fruit object contains only the price property after deleting the name property.

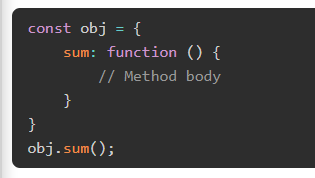


# **JavaScript - Object Methods**

* JavaScript object methods are object properties that contains function definitions. 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 that case the property is known as a method.
* You can either directly add a method to the object or add it as a property value. The method can also take the parameters and return the value. Object methods are a powerful way to add functionality to objects. They allow you to encapsulate code and make it reusable.

### Syntax

Follow the syntax below to add a method to the object.



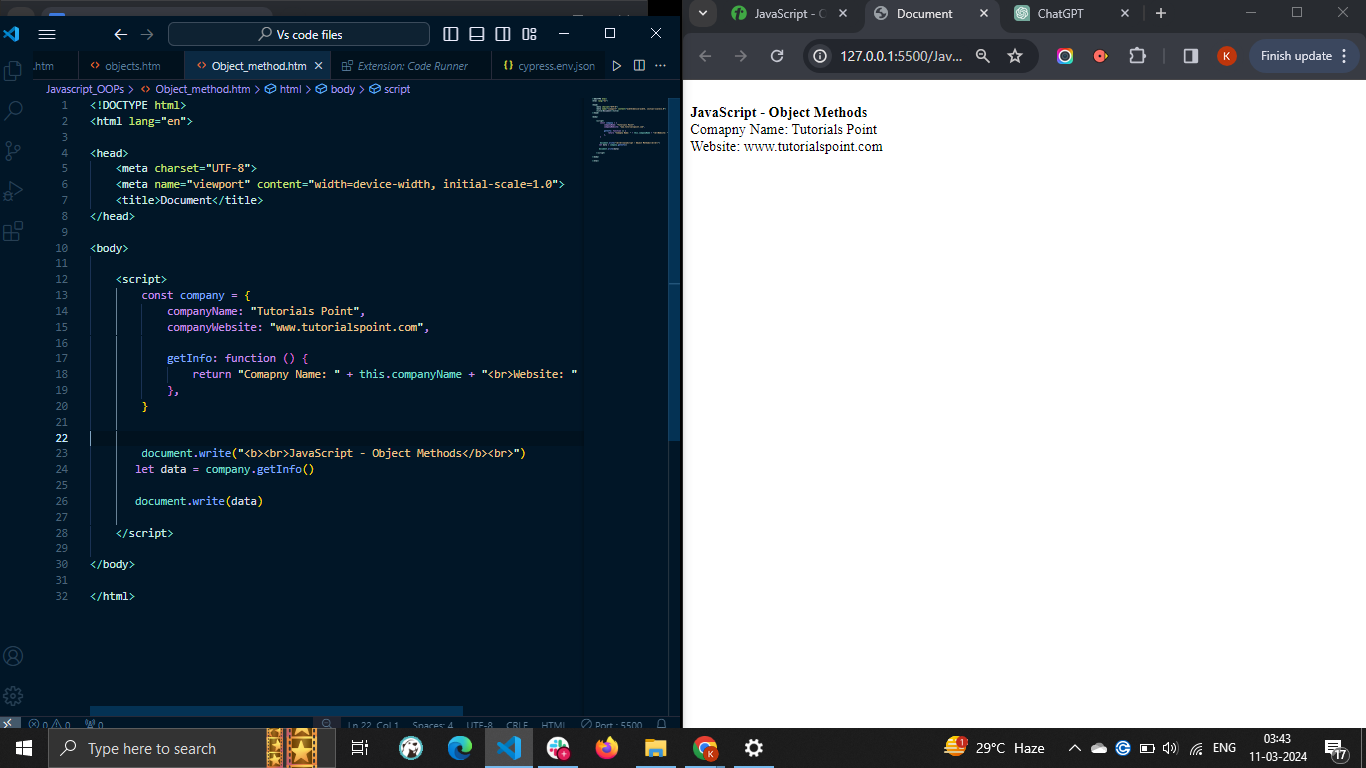
In the above syntax, 'sum' is a method defined inside the 'obj' object. You can access the method as you access the object property and add the pair of parenthesis to invoke the method.

### Example

We added the getInfo() method in the 'company' object in the example below. The getInfo() method returns the string containing the object properties.

Here, we used the 'this' keyword to access the object properties inside the object. The 'this' keyword represents the object itself.

After that, we used the object as a reference to invoke the method.

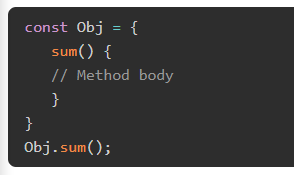


## Object Method Shorthand

The ES6 provides the shortest way to define a method into the object.

### Syntax

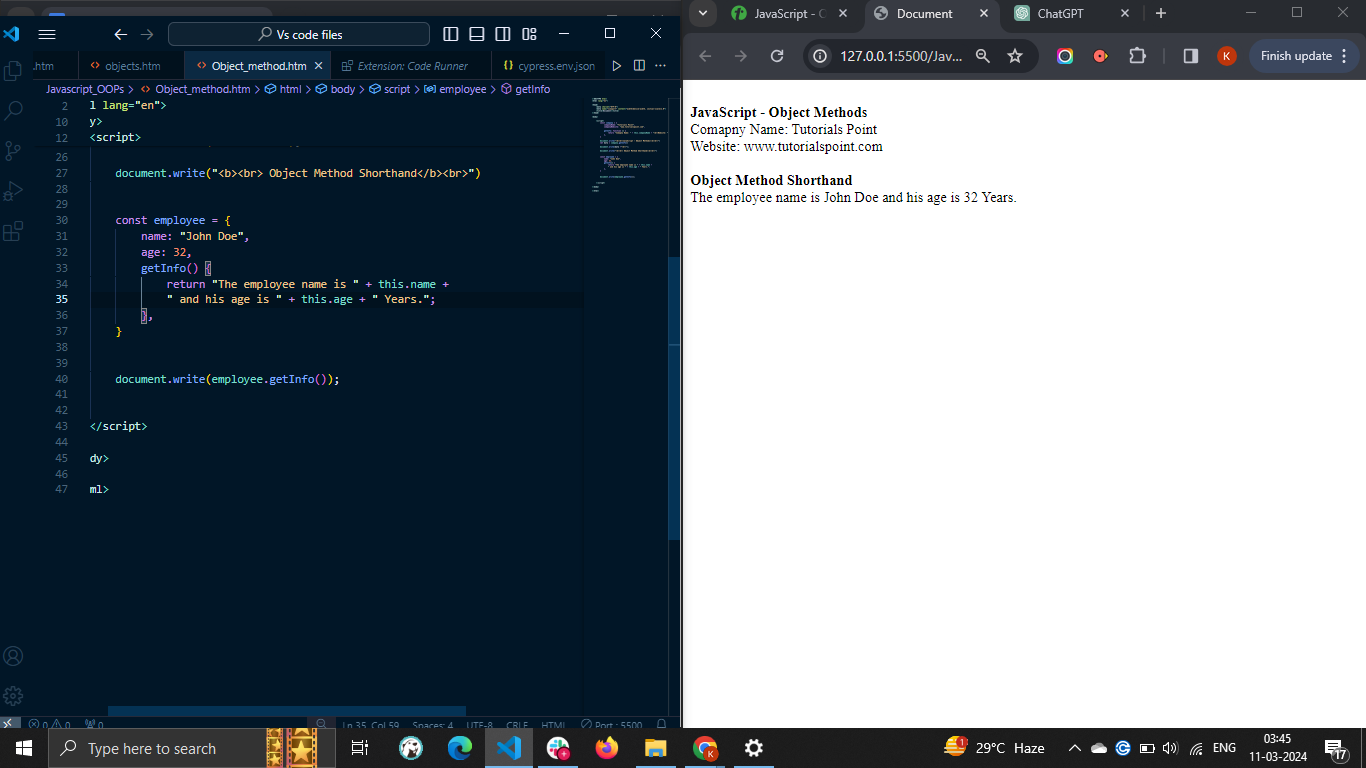
Follow the syntax below to add a method to the object.



Like the previous one, you can access and invoke the method in the above syntax.

### Example

In the example below, we defined the getInfo() method as the previous example.



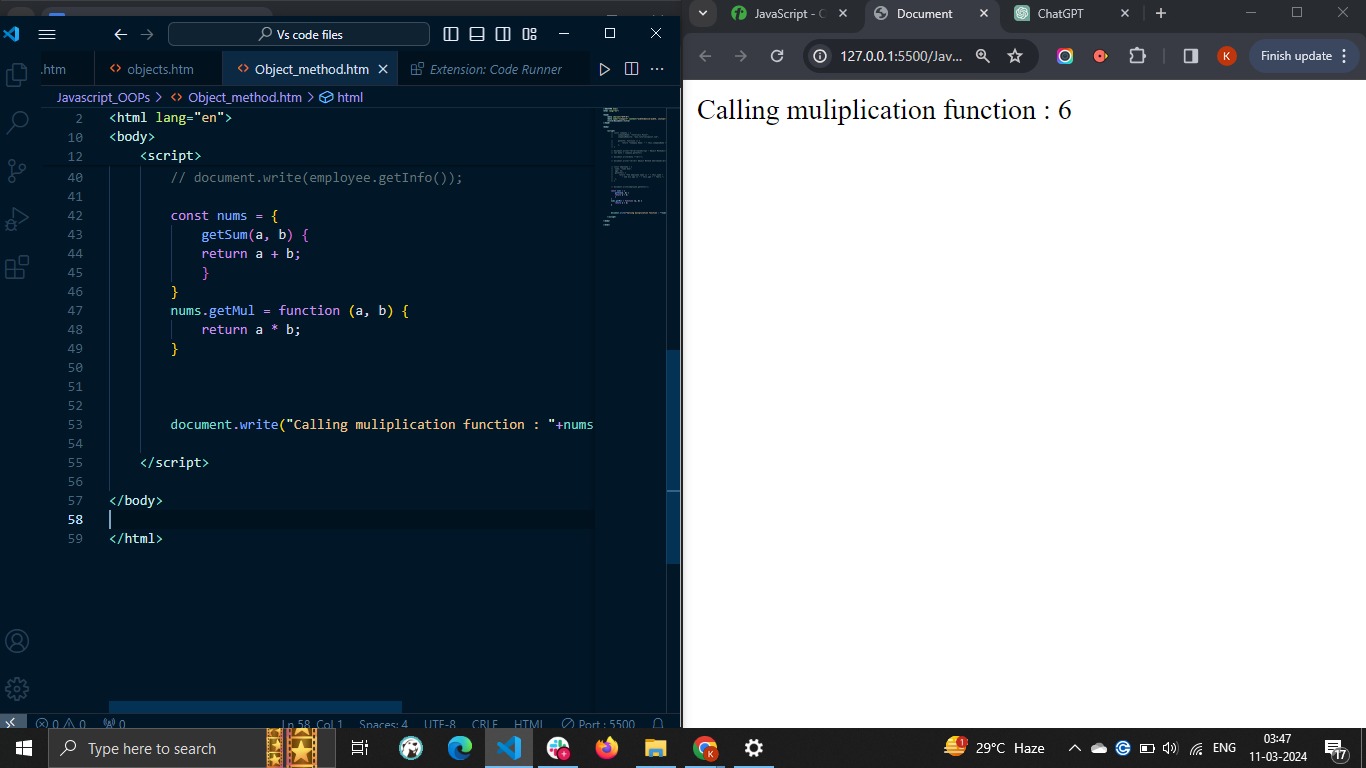
## Updating or Adding a Method to the Object

In JavaScript, updating or adding a new method to the object is same as updating or adding new proeprties to the object. You can either use the dot or square bracket notation to update or add method to the object.

### Example

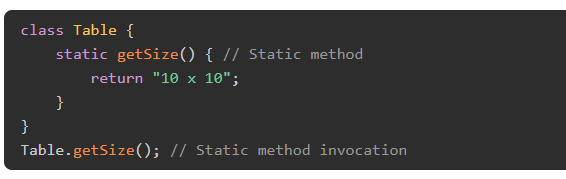
The example below defines the getSum() method inside the 'nums' object.

After that, we add the getMul() method inside the nums object. We invoke the getMul() method by passing two arguments to get the multiplication of them.



# **JavaScript - Static Methods**

* A static method in JavaScript is defined using the static keyword followed by the method name. You can execute the static method by taking the class name as a reference rather than an instance of the class.
* The main benefit of the static method is that it can be used to create a utility function that doesn't require the instance of the class for the execution. For example, a Math object contains various static methods, which we can invoke through Math class directly.
* Also, you can use static methods to add all related methods under a single namespace. Furthermore, static methods give better performance than the regular class methods due to memory optimization.

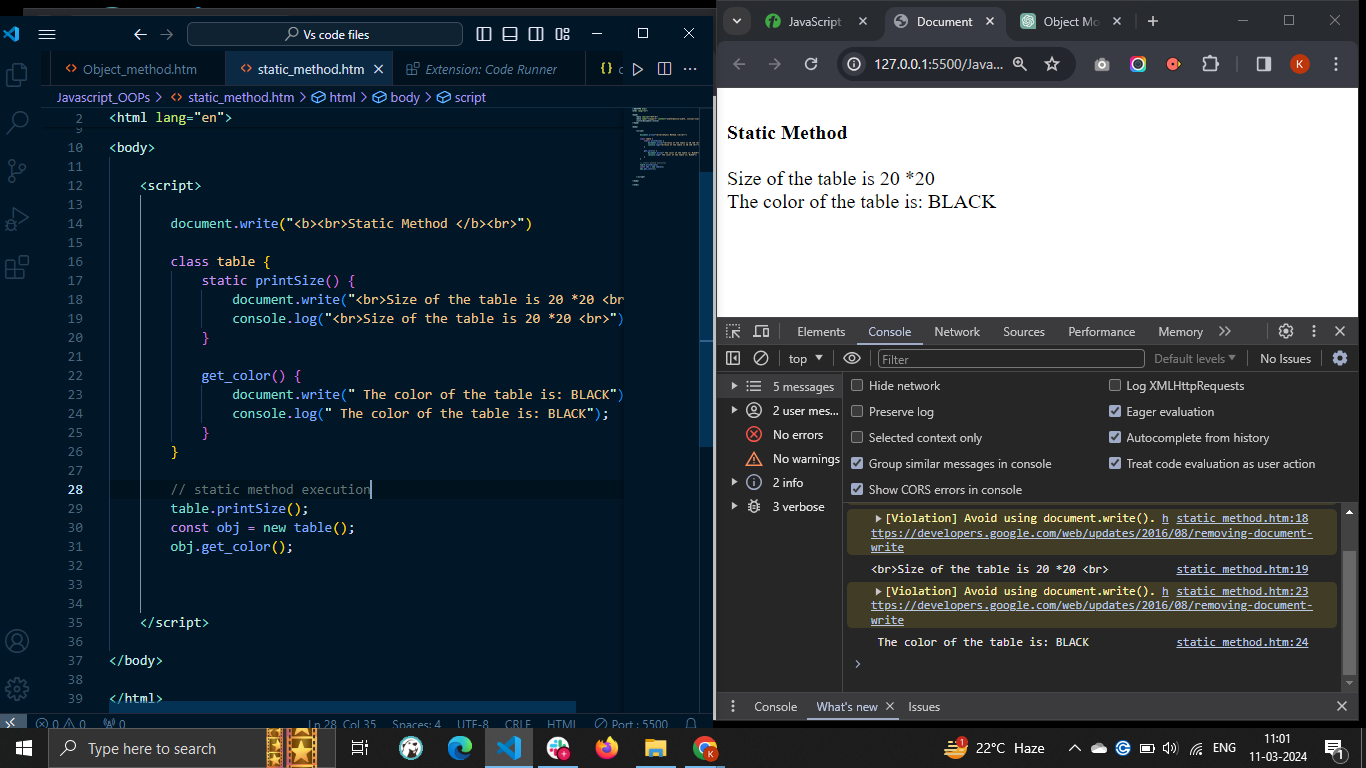


In the above syntax, getSize() is a static method. We used the class name to execute the getSize() method.

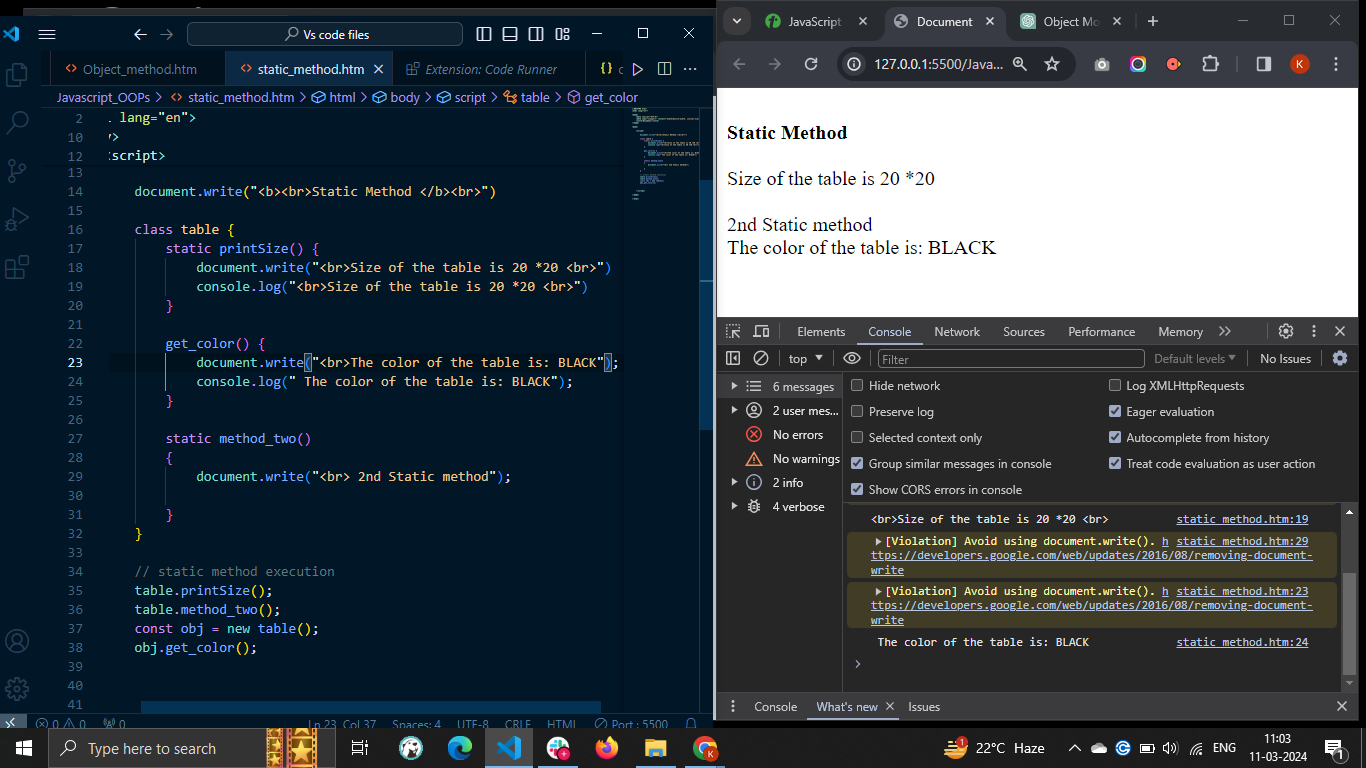
### Example: Static Method

In the example below, printSize() is a static method, and getSize() is a regular method in the table class. You can see that printSize() method is invoked using the table class name, and getSize() method is executed using the class instance.

So, the class can contain static and non-static methods.

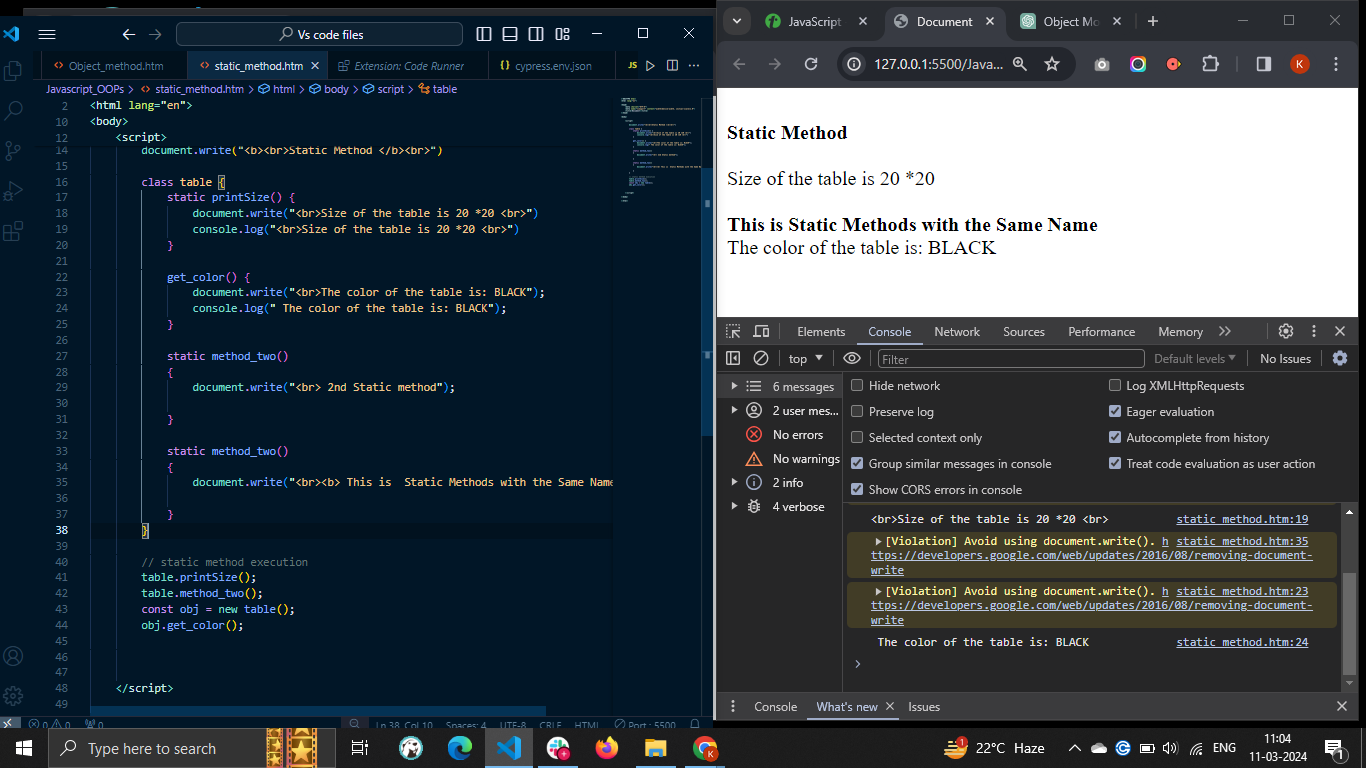


The single class can also contain multiple static methods.



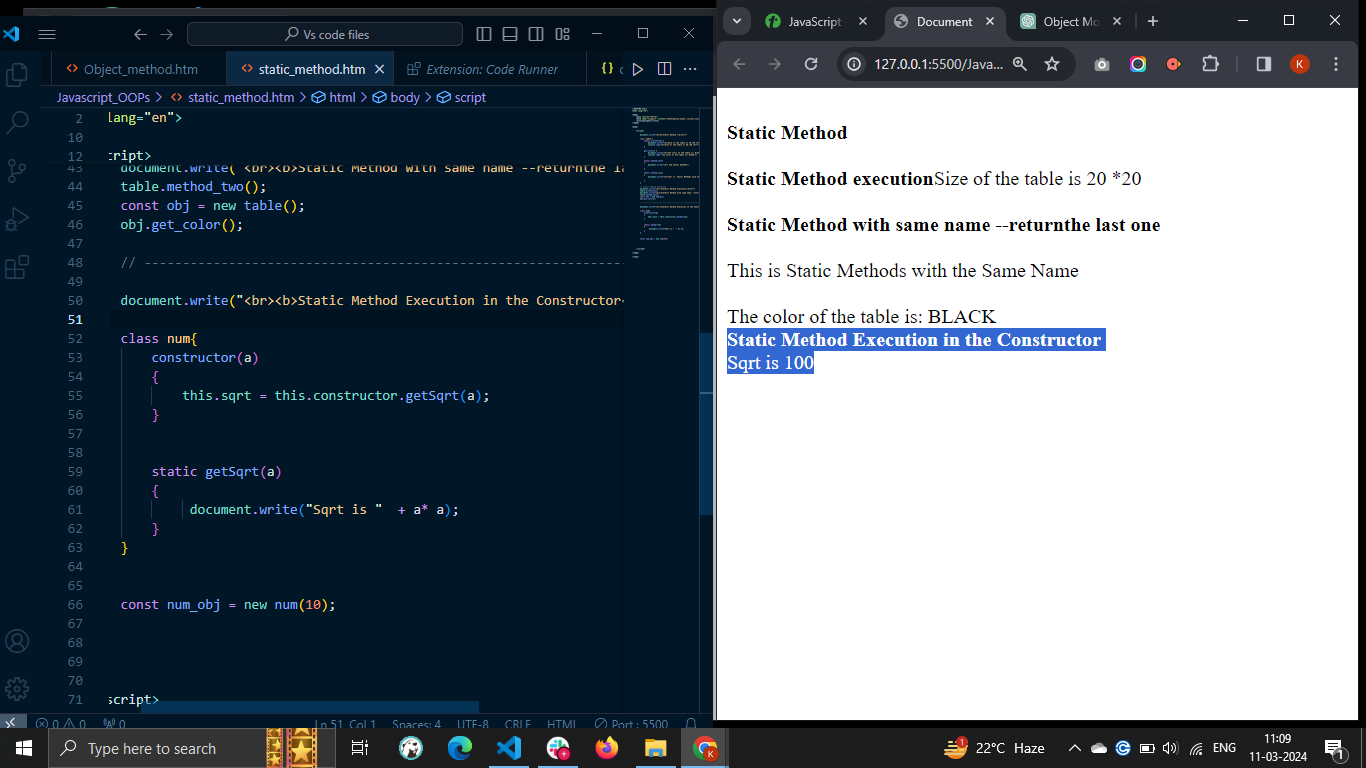
A single class can contain multiple static methods with the same name. When you execute the static method with the same name, it executes the last method.

### Example: Static Methods with the Same Name



You can also execute the static method of the class in the constructor. You can use this keyword followed by the constructor keyword followed by the static method name to execute the static method in the constructor.

### Static Method Execution in the Constructor

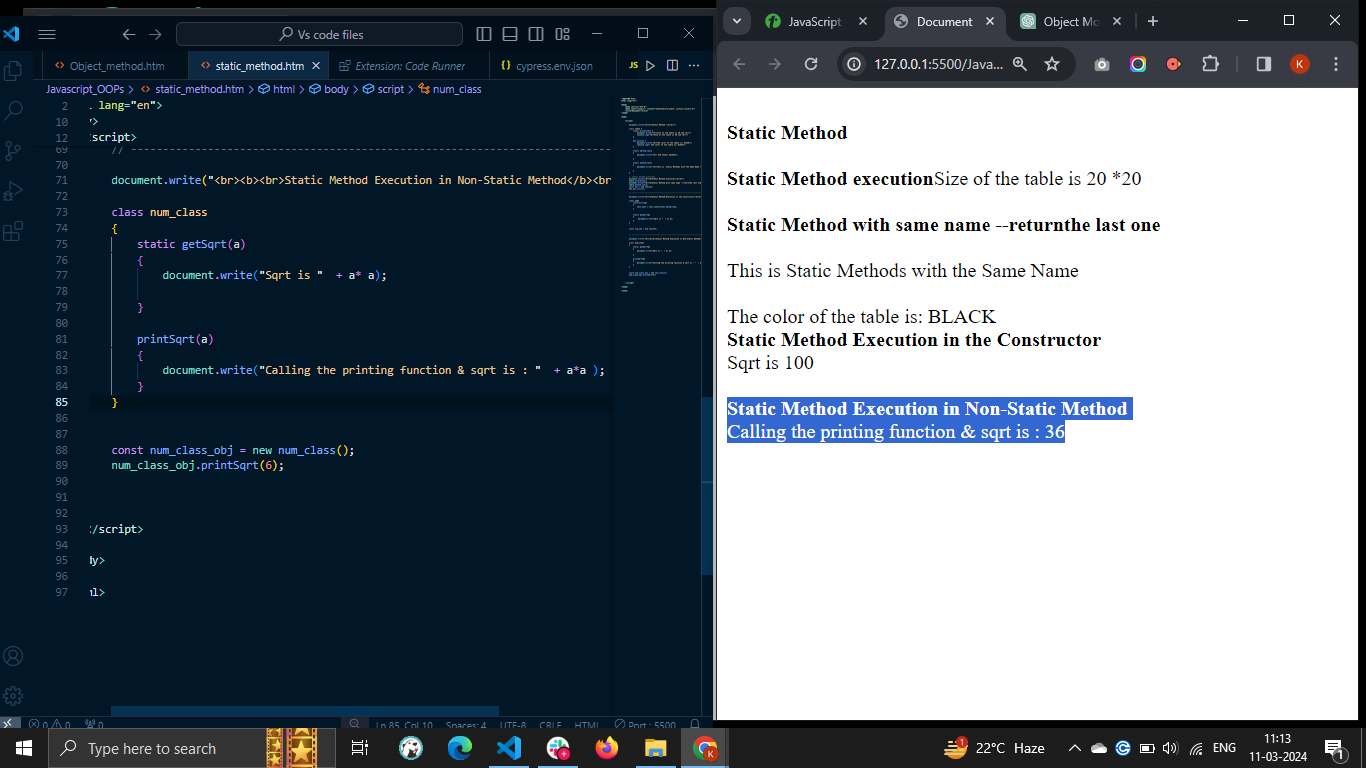


You can also execute the static method in the non-static method. You need to use the class name as a reference to execute the static method in the non-static method.

### Static Method Execution in Non-Static Method

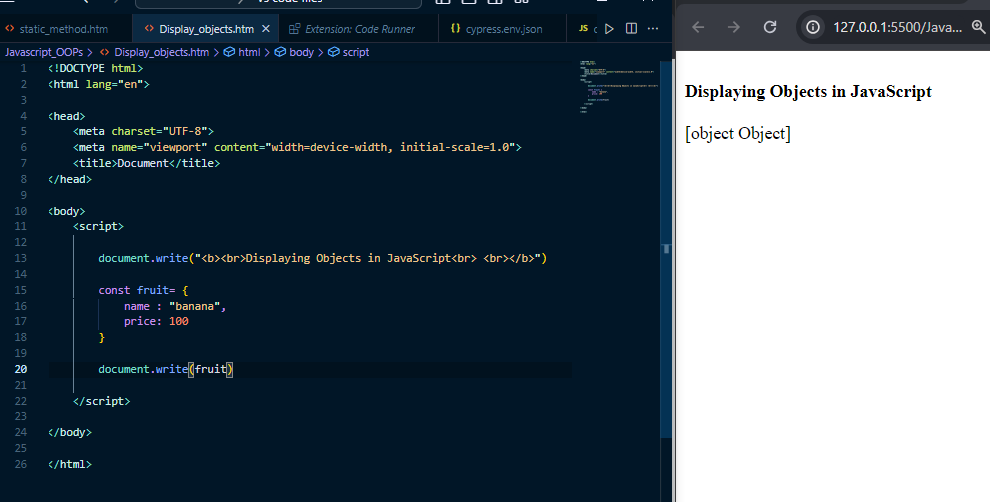
In the example below, getSqrt() is a static method, and printSqrt() is a regular class method. In the printSqrt() method, we execute the getSqrt() method.

We used the instance of the Num class to execute the printSqrt() method.



# **JavaScript - Display Objects**

* There are different ways to display objects in JavaScript. Using the console.log() method, we can display the object in the web console. Sometimes developers require to display the object properties and their value in the HTML or for debugging the code.
* For displaying an object, we can access the different properties and display them. We can also convert the object to a JSON string and display it as a string.
* When you print the object like other variables in the output, it prints the '[object Object]'



To overcome the above problem, you need to use specific approaches to display the object.

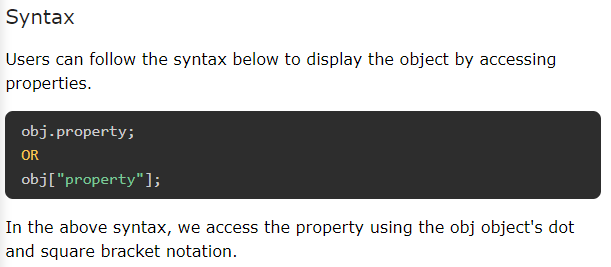
Some approaches to display the JavaScript objects are as follows −

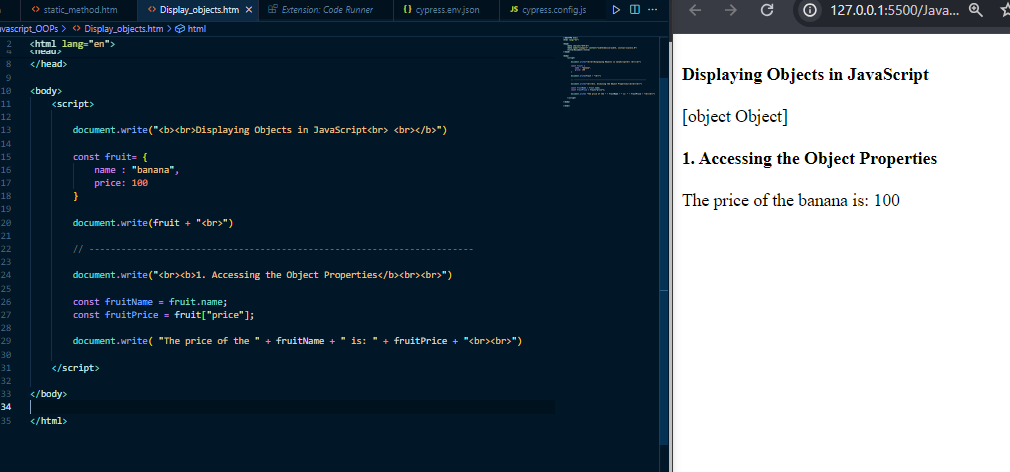
* Accessing the object properties
* Using the JSON.stringify() method
* Using the Object.entries() method
* Using the for...in loop

## 1. Accessing the Object Properties

You can use the **dot notation** or **square bracket** notation to display the property values.

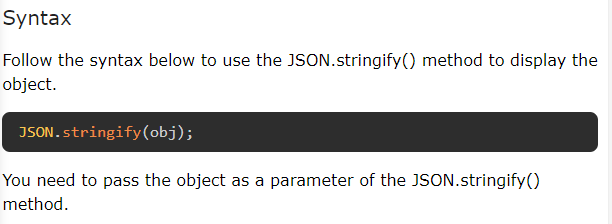
This way, you may get all property values and display them in the output.

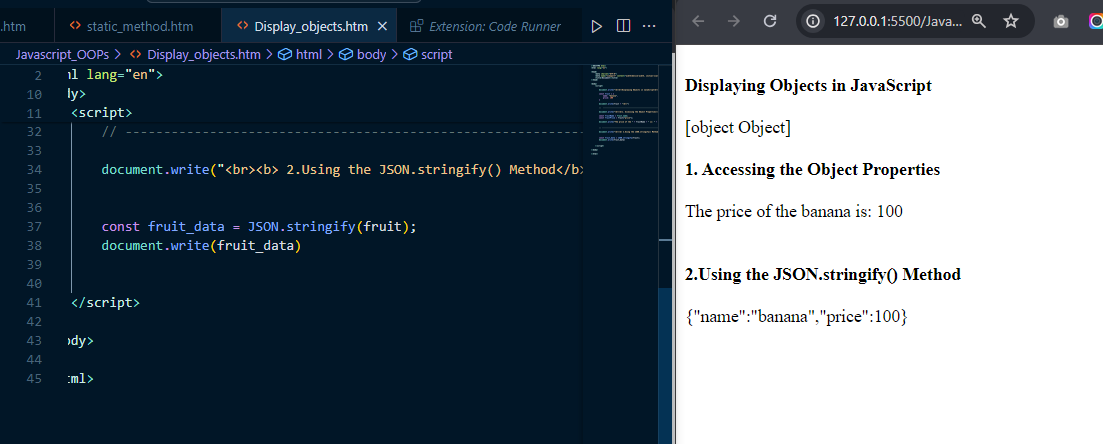




2. Using the JSON.stringify() Method

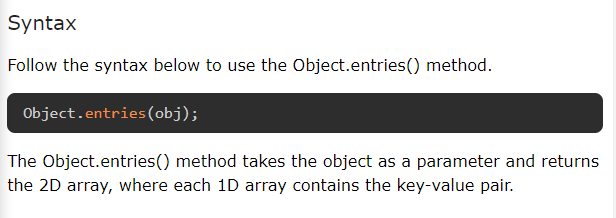
When object contains the dynamic properties or you don't know object properties, you can't print properties and values using the first approach. So, you need to use the JSON.stringify() method. It converts the object into a string.

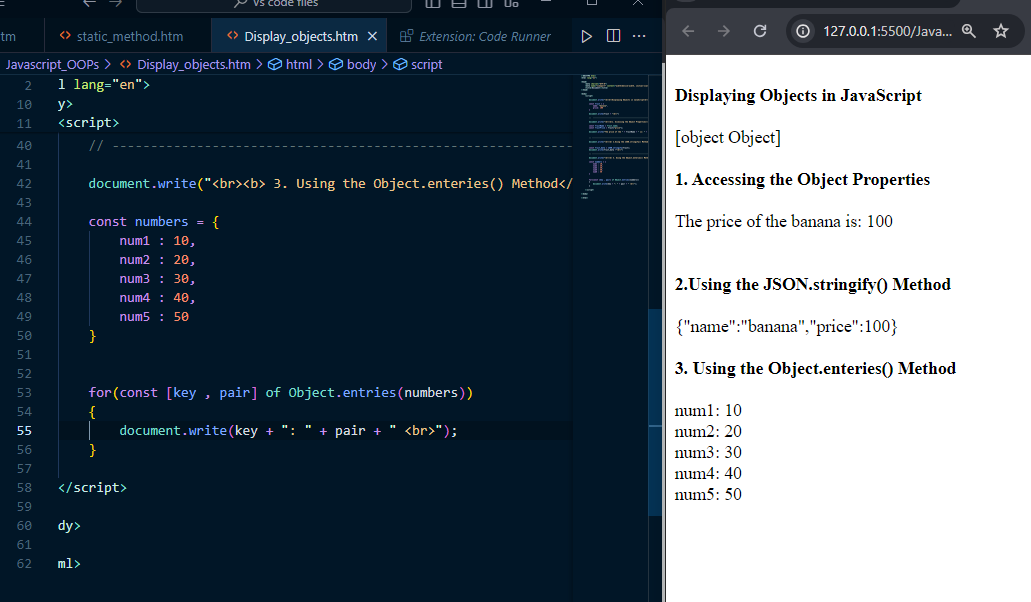




## 3. Using the Object.entries() Method

The Object.entries() is a static method of the Object class, allowing you to extract the properties and values in the 2D array. After that, you can use the loop to traverse the array and display each property and value pair individually.



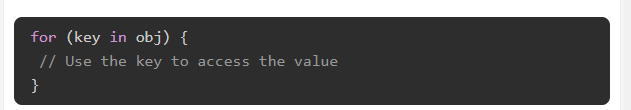


## 4. Using the for...in Loop

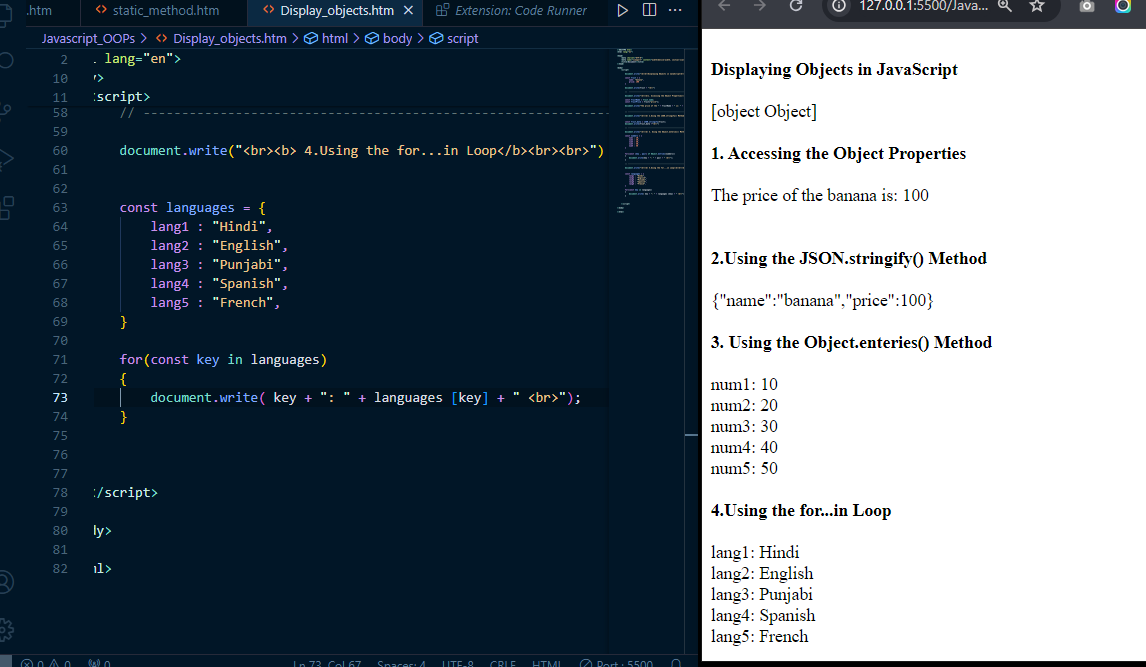
The for...in loop is used to traverse the iterable, and the object is one of them.

### Syntax

Users can follow the syntax below to use the for...in loop to traverse the object and display it in the output.



In the above syntax, obj is an object to display. In the loop body, you can access the value related to the key and print the key-value pair.



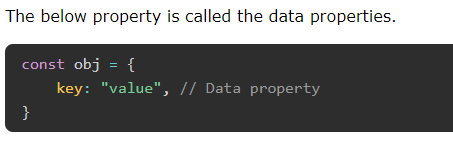
**NOTE : The best way to display the object is using the JSON.stringify() method. It converts the object into a flat string. Other approaches can't be used to display the nested objects, but JSON.stringify() method can be used.**

# **JavaScript - Object Accessors**

The **object accessor** properties in JavaScript are methods that get or set the value of an object. They are defined using the get and set keywords. Accessor properties are a powerful way to control how your objects are accessed and modified.

The JavaScript object can have two kinds of properties.

* Data properties
* Accessor properties



## Object Accessor Properties

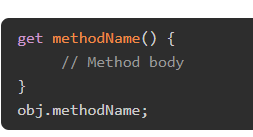
In JavaScript, you can use the getters to get the object properties and setters to set the object properties.

There are two keywords to define accessor properties.

* get − The get keyword is used to define a method to get the object property value.
* set − The set keyword is used to define a method to update the object property value.

## JavaScript Getters

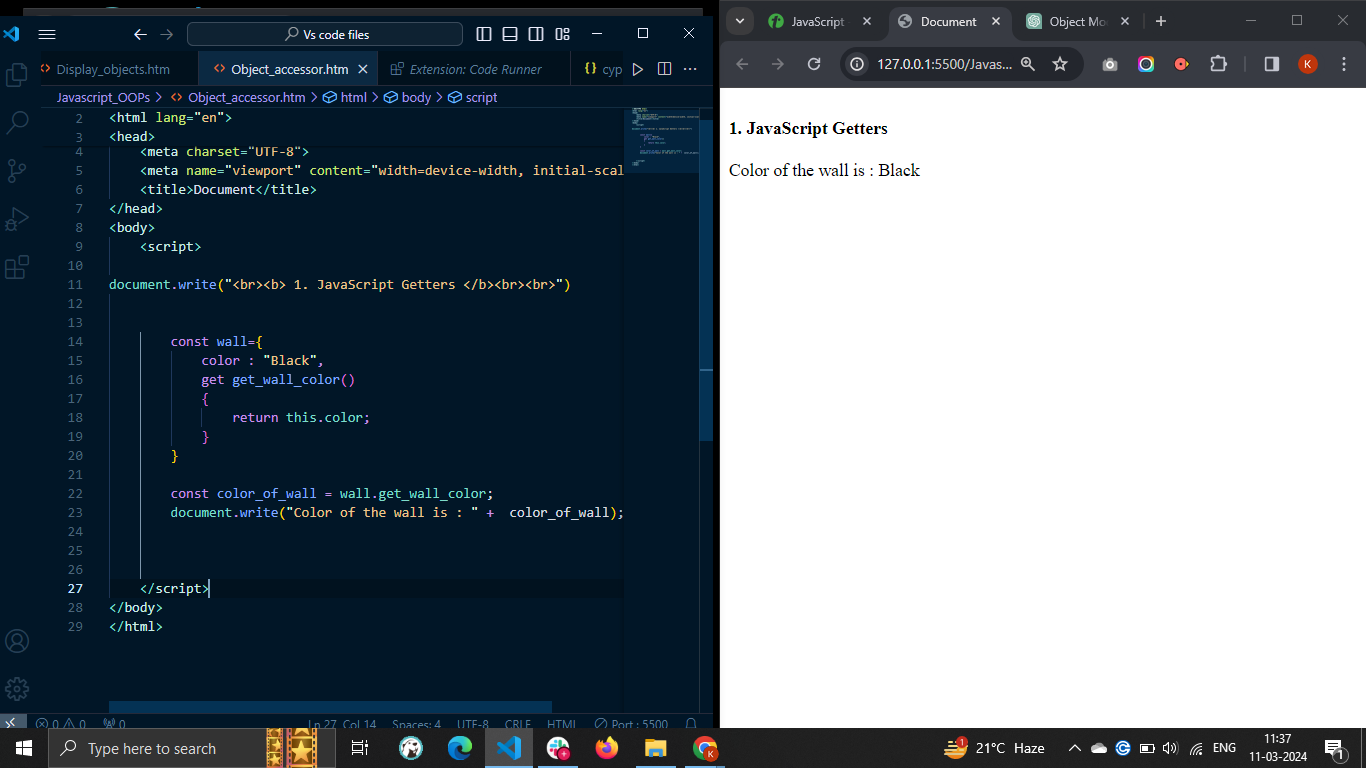
The getters are used to access the object properties. To define a method as getter, we use get keyword followed by method name. Follow the syntax below to define the getter.



In the above syntax, we have defined the getters using the 'get' keyword followed by the method name.

You can use the method name as an object property to get its returned value.

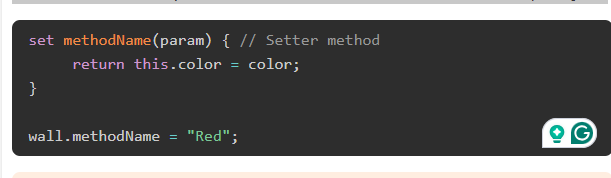
*You don't need to write the pair of parenthesis followed by the method name to execute the getters. You can access it like the object property.*



## JavaScript Setters

The **setters** are used to update the JavaScript object properties. To define a method as setter, we use **set** keyword followed by method name You can follow the syntax below to define setters in the JavaScript object.

Syntax :



* In the above syntax, the 'set' keyword is used to define the setter method.
* The method\_name can be any valid identifier.
* The setter method always takes a single parameter. If you don't pass a parameter or multiple parameters, it will give you an error.
* You can assign value to the setter method as you assign value to the property.

