Java Scripts

**https://www.udemy.com/course/javascript-beginnerscourse/**

JavaScript is a programming language used to make websites interactive. If you think about the basic makeup of a website, you have HTML, which describes and defines the basic content and structure of the website, then you have CSS, which tells the browser how this HTML content should be displayed—determining things like color and font. With just HTML and CSS, you have a website that looks good but doesn’t actually do much. JavaScript brings the website to life by adding functionality. JavaScript is responsible for elements that the user can interact with, such as drop-down menus, modal windows, and contact forms. It is also used to create things like animations, video players, and interactive maps.

JavaScript isn’t only used to create websites. It can also be used to build browser-based games and, with the help of certain frameworks, mobile apps for different operating systems. The creation of new libraries and frameworks is also making it possible to build backend programs with JavaScript, such as web apps and server apps.

# first program in JS

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>JavaScript Demo</title>

</head>

<body>

<script language="javascript" type="text/javascript">

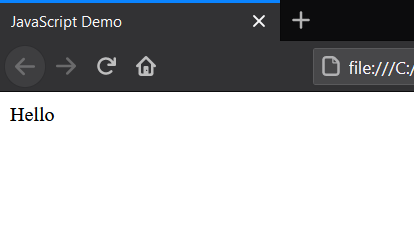
document.write('Hello');

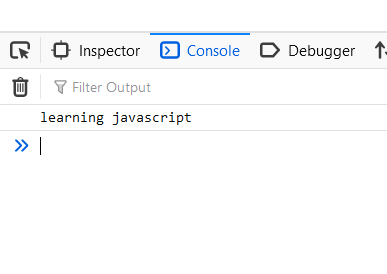
console.log("learning javascript");

</script>

</body>

</html>

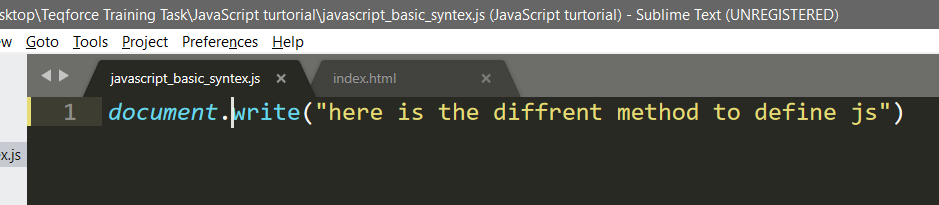


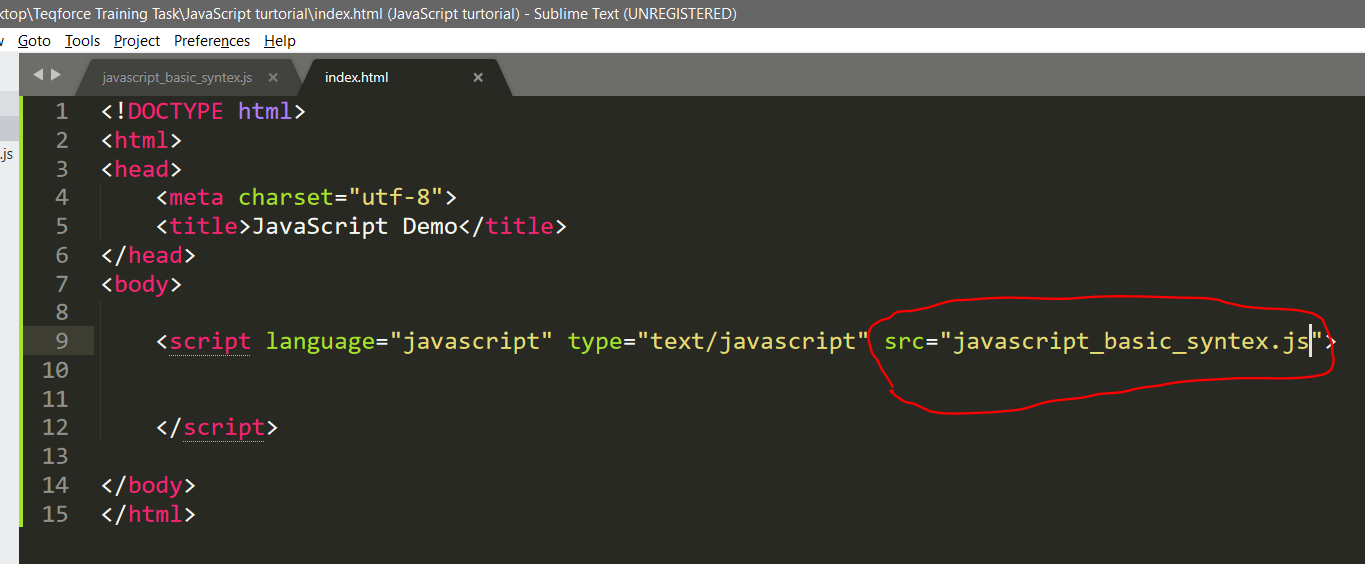


>>>>> **document.write** method is used for getting output**.**

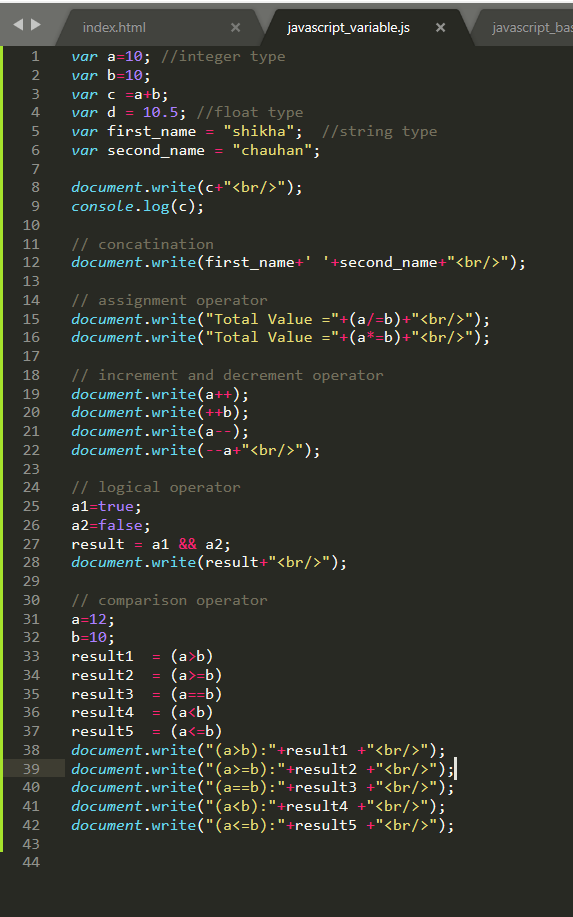
>>>>>> **console.log** method is also for output in java script consol

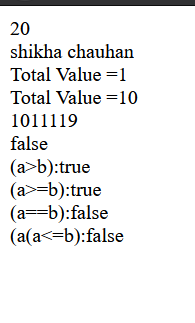
# create js file seprate and link with html file





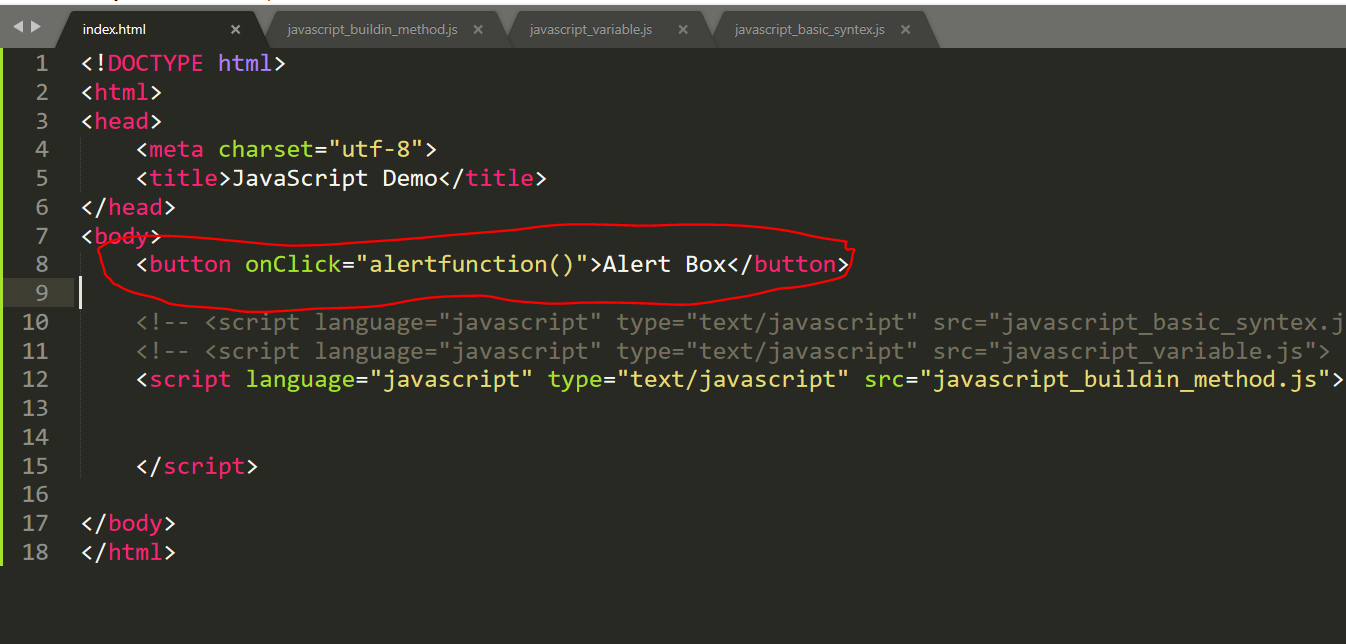
**Variables**

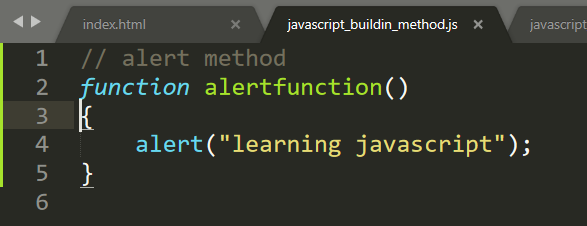




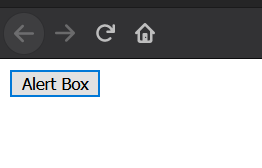
**Build-In Methods in Javascript**

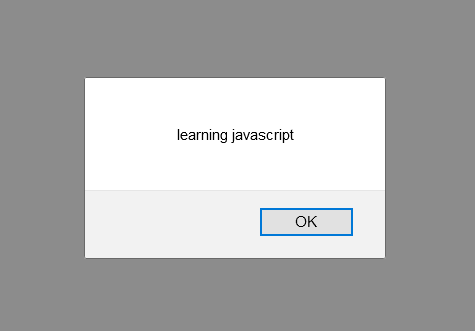
Alert function:



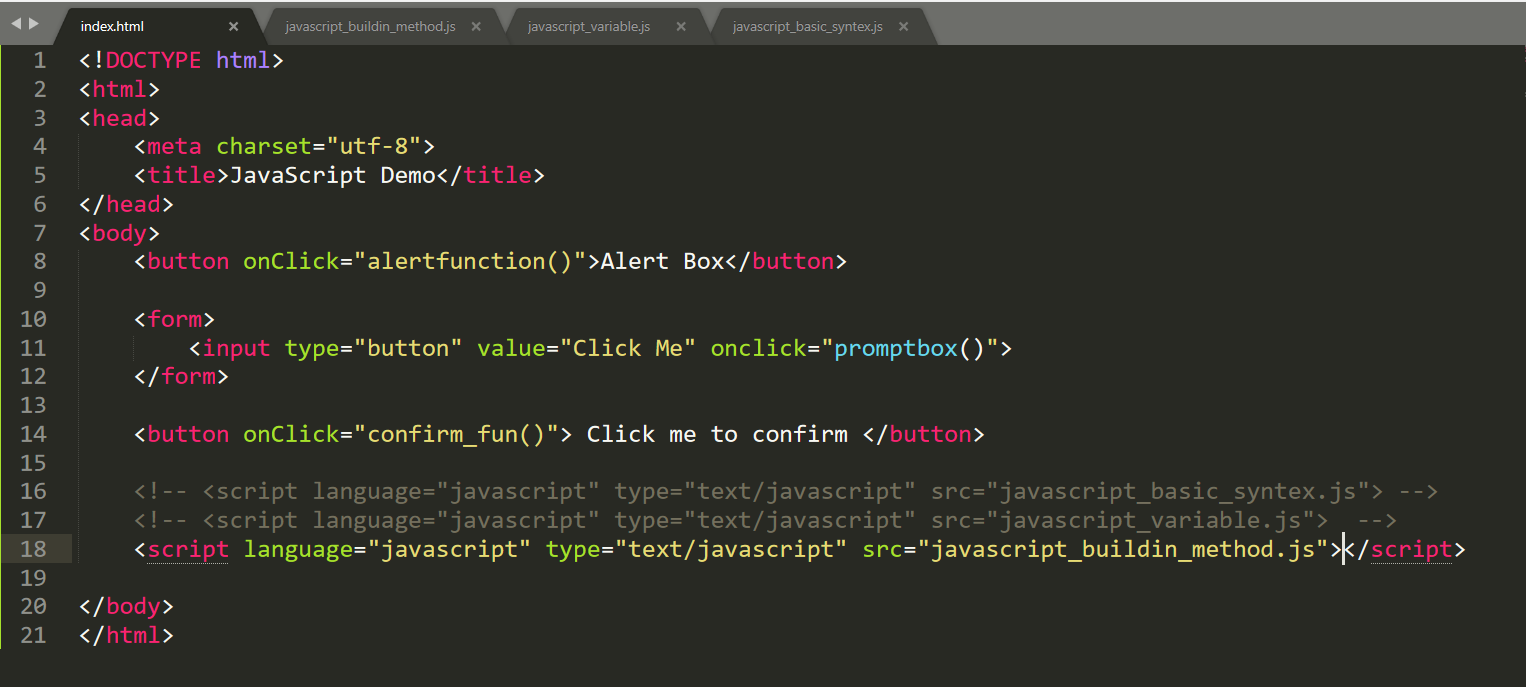


Output:



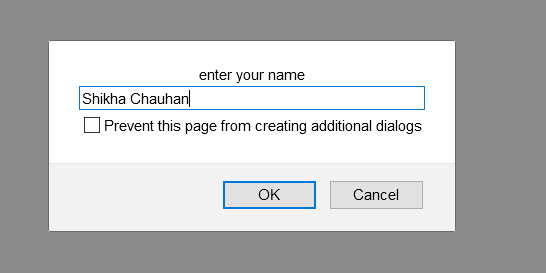


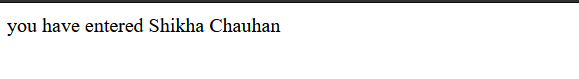
Inbuild function:

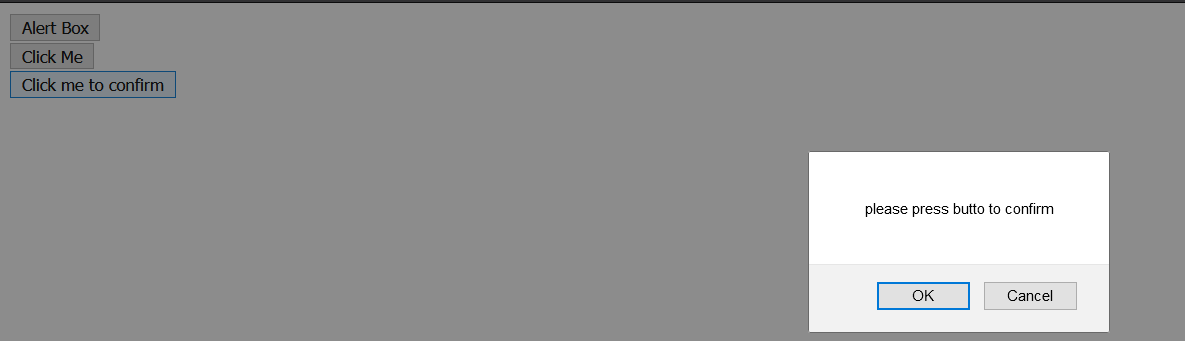




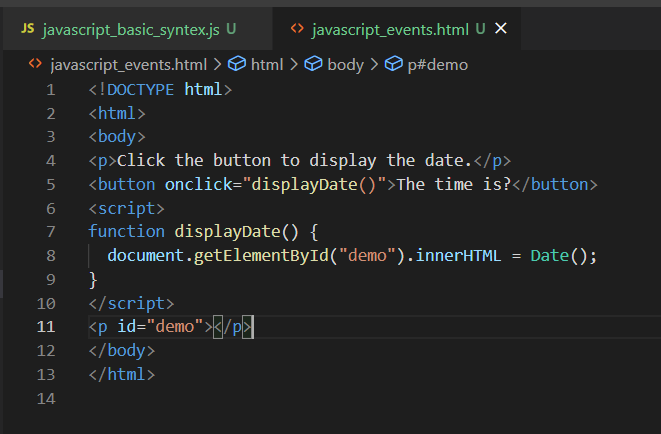
Output:

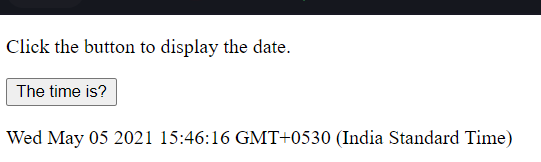




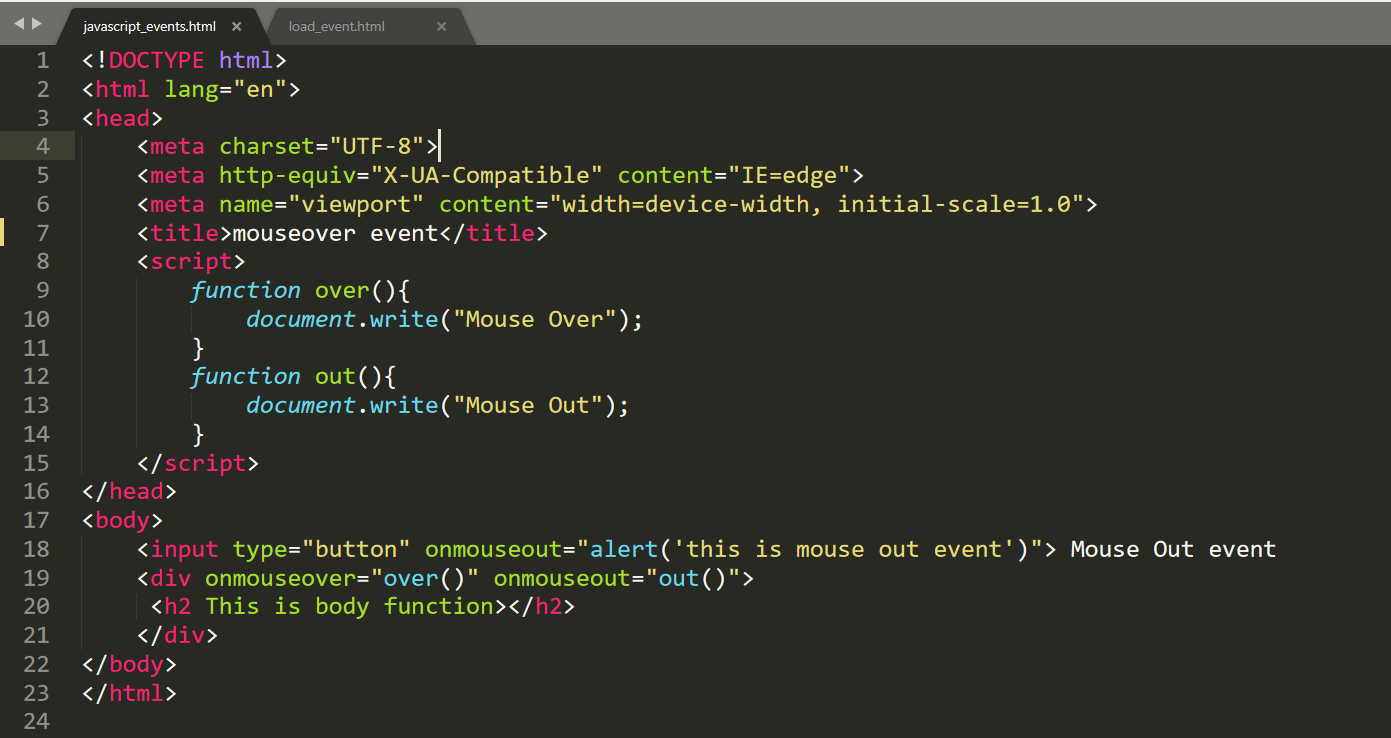


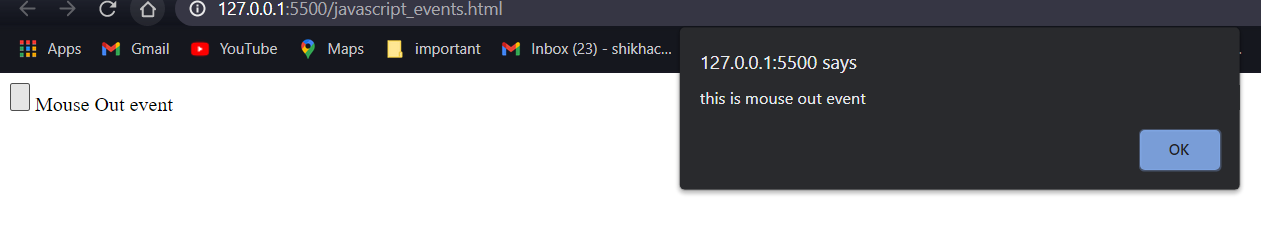
Events: onclick event



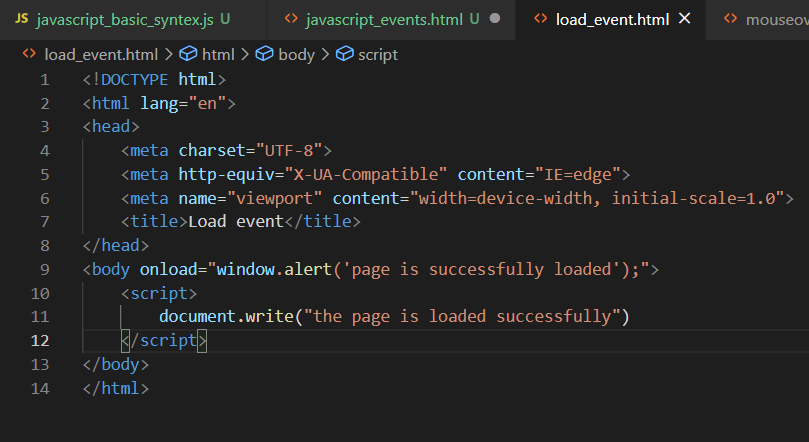


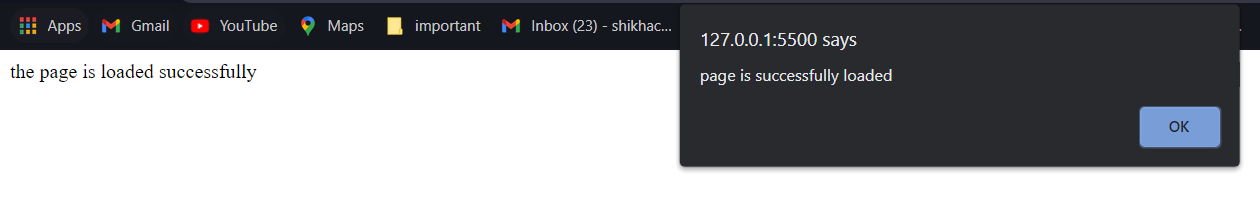
Mouseover event:



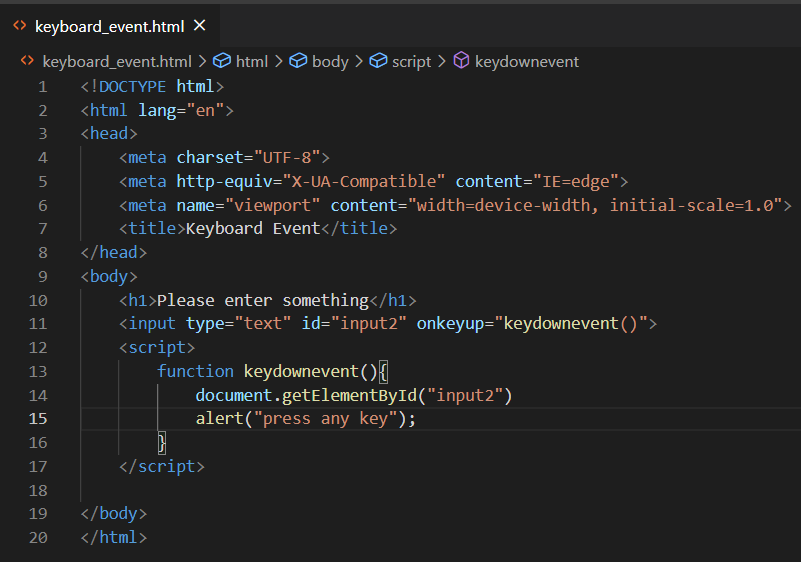


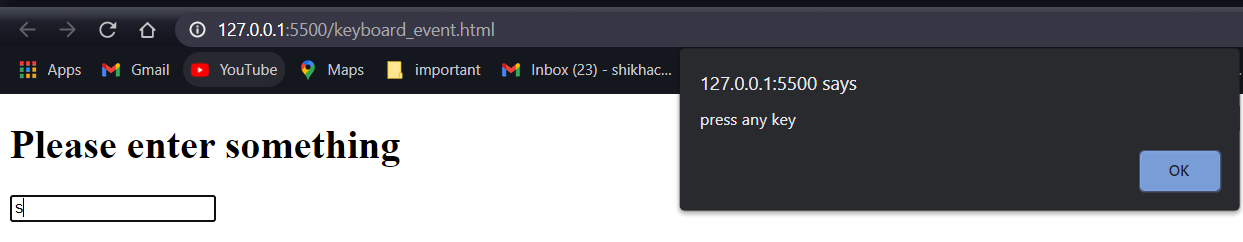
Load Event:



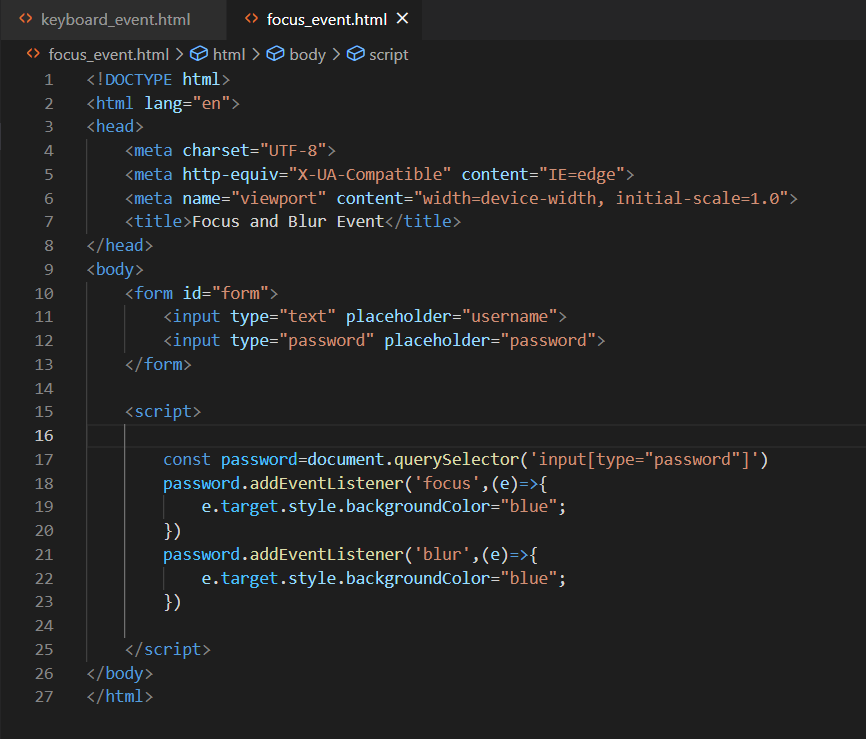


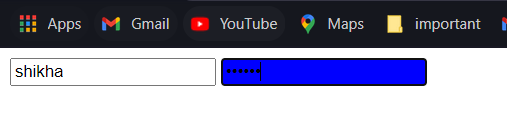
Keyboard Event:



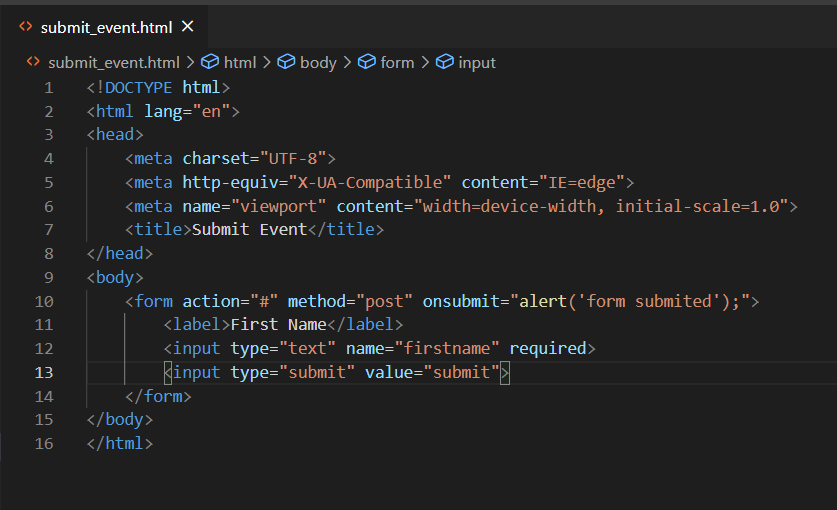


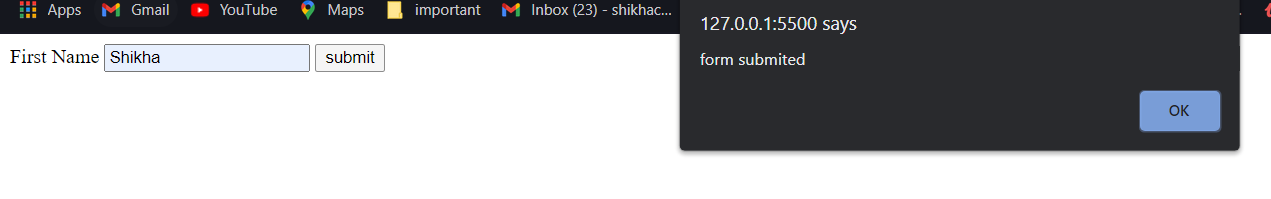
Focus or blur Event:





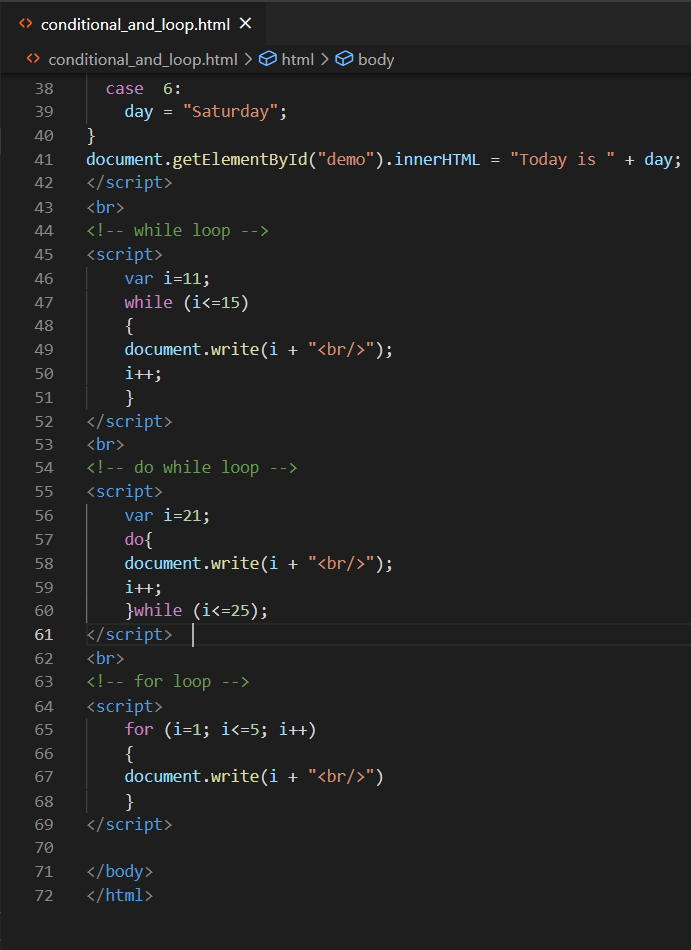
Submit Event:



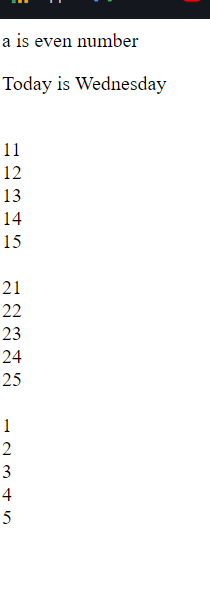


Conditional and loop statement:

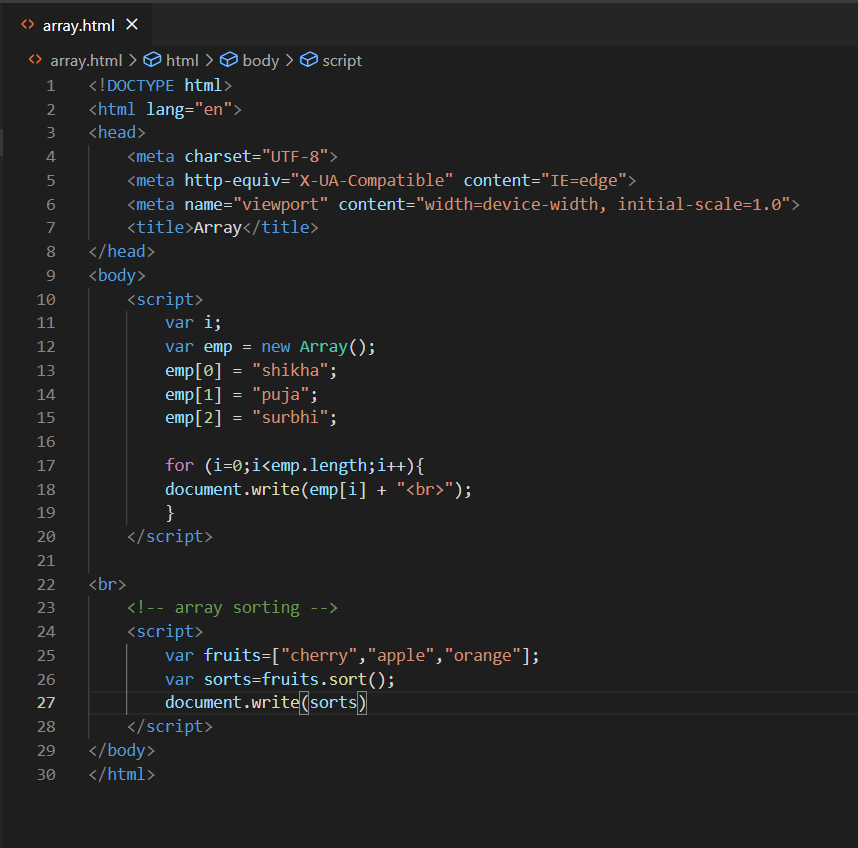


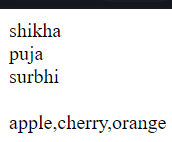


Output:



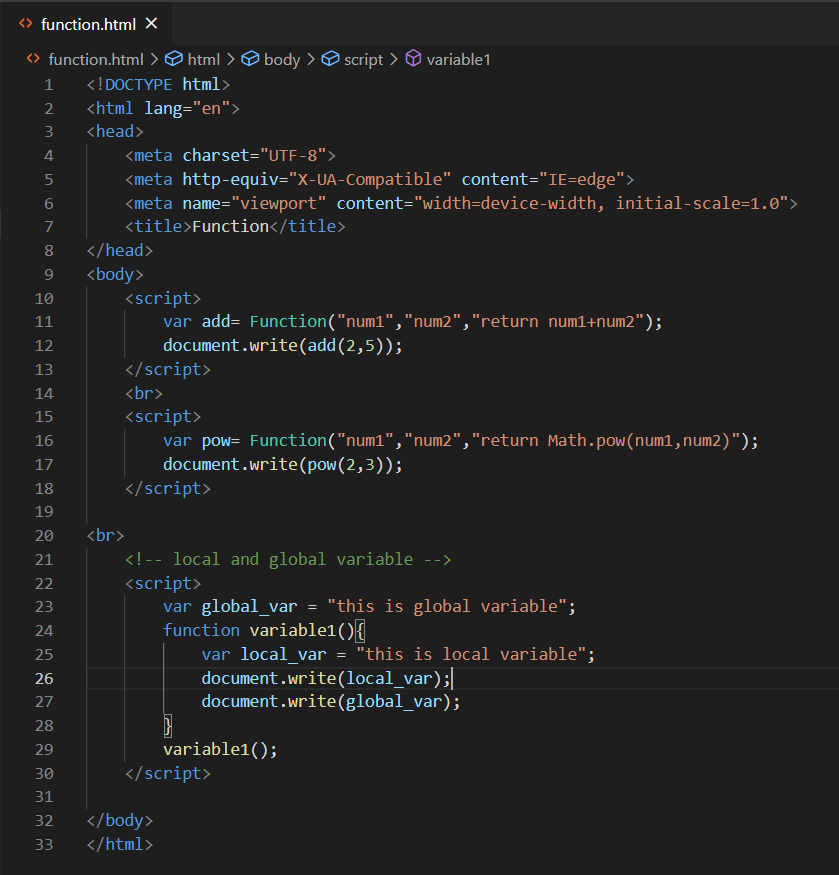
Array:

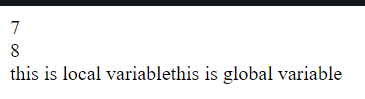




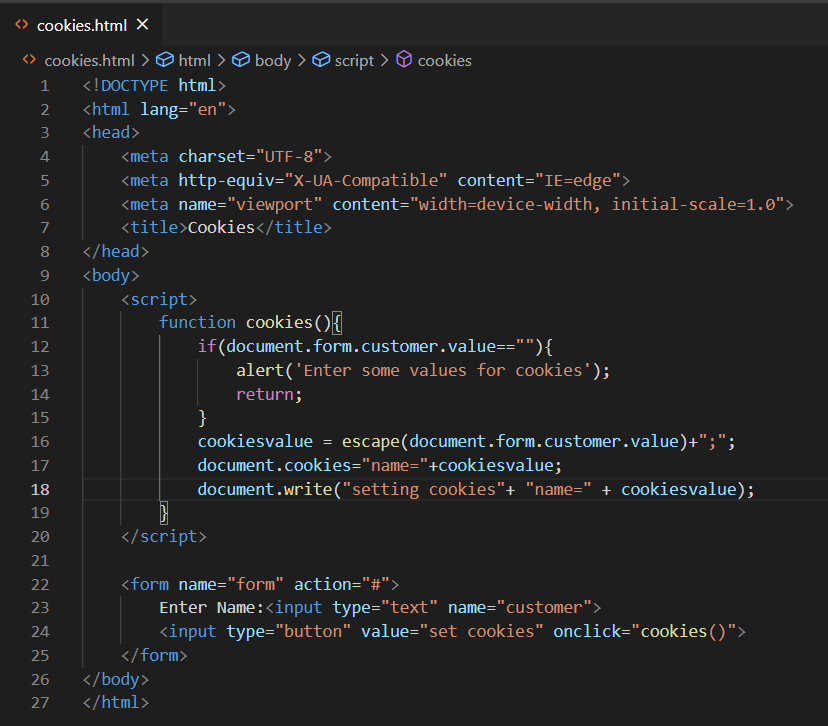
Function:

* You can reuse code: Define the code once, and use it many times.
* You can use the same code many times with different arguments, to produce different results.





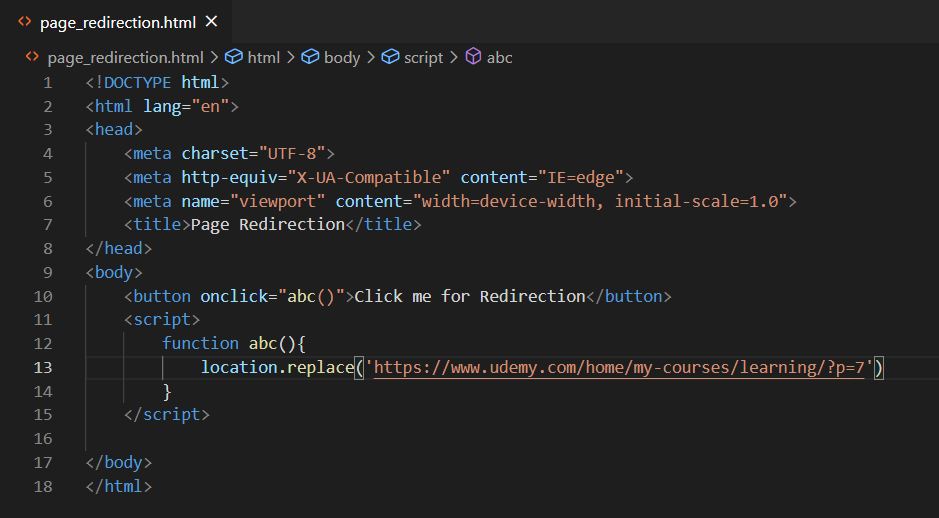
Cookies:

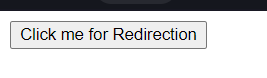


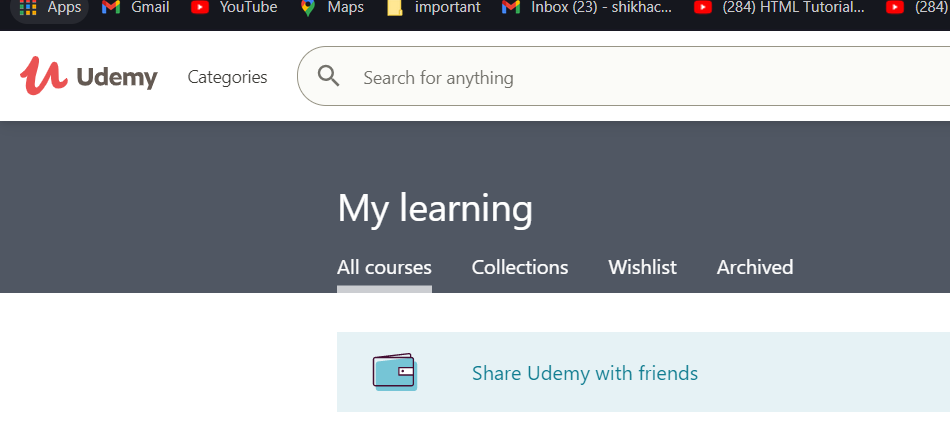




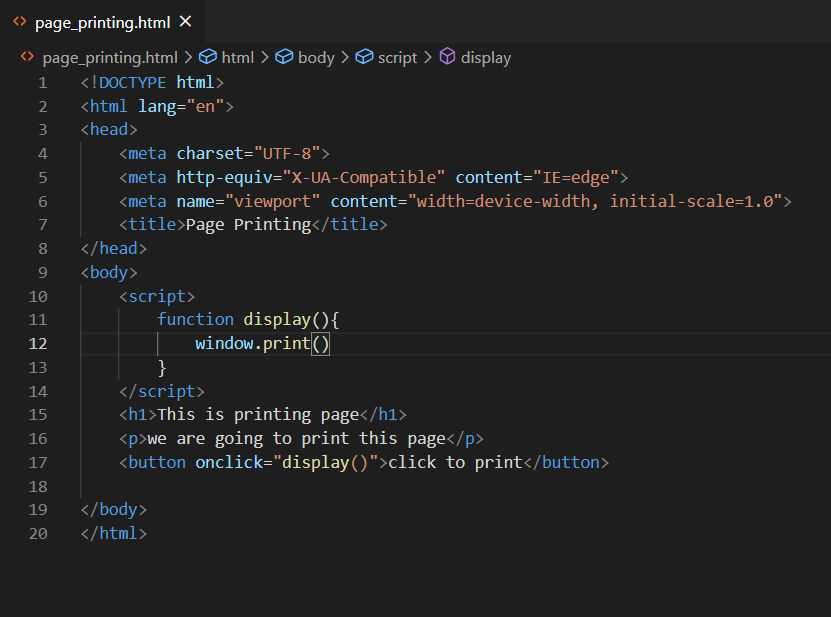
Page Redirection





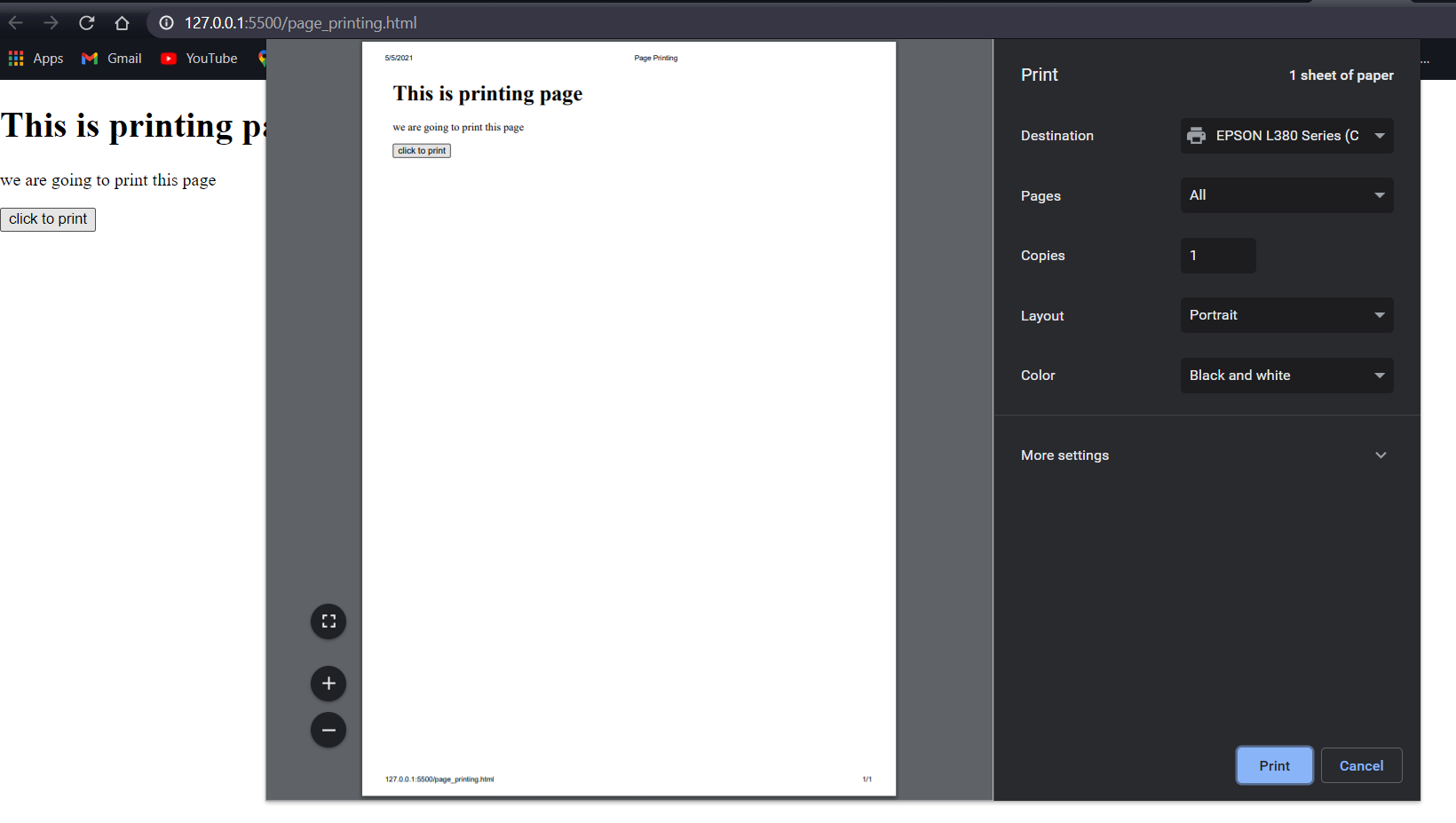


Page Printing:





After click on button

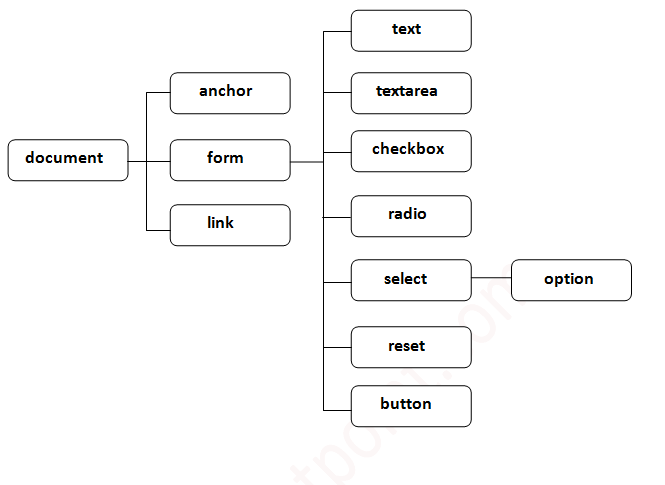


Document Object Model

The **document object** represents the whole html document.

When html document is loaded in the browser, it becomes a document object. It is the **root element** that represents the html document. It has properties and methods. By the help of document object, we can add dynamic content to our web page.

## Properties of document object

Let's see the properties of document object that can be accessed and modified by the document object.

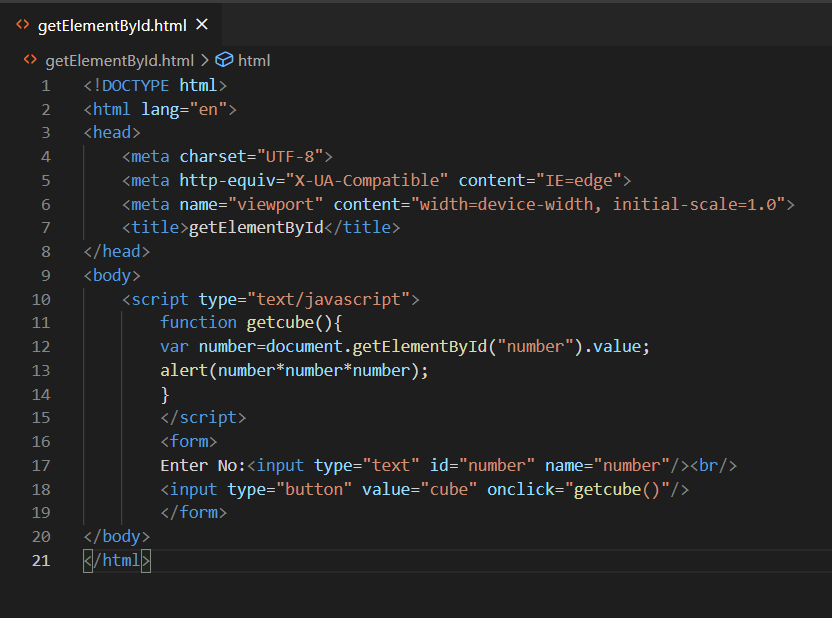
# 

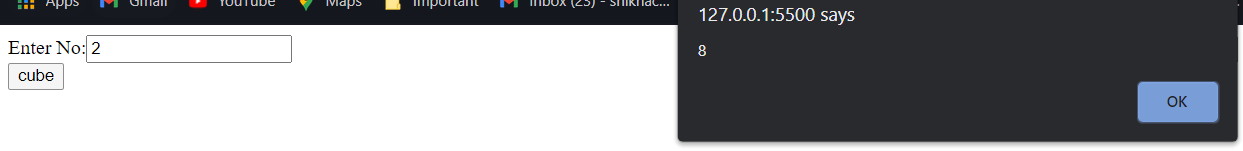
# 

# document.getElementById() method:

The **document.getElementById()** method returns the element of specified id.

**document.form1.name.value** to get the value of the input value. Instead of this, we can use document.getElementById() method to get value of the input text. But we need to define id for the input field.





### getElementsByClassName(), querySelector() and querySelectorAll() Methods

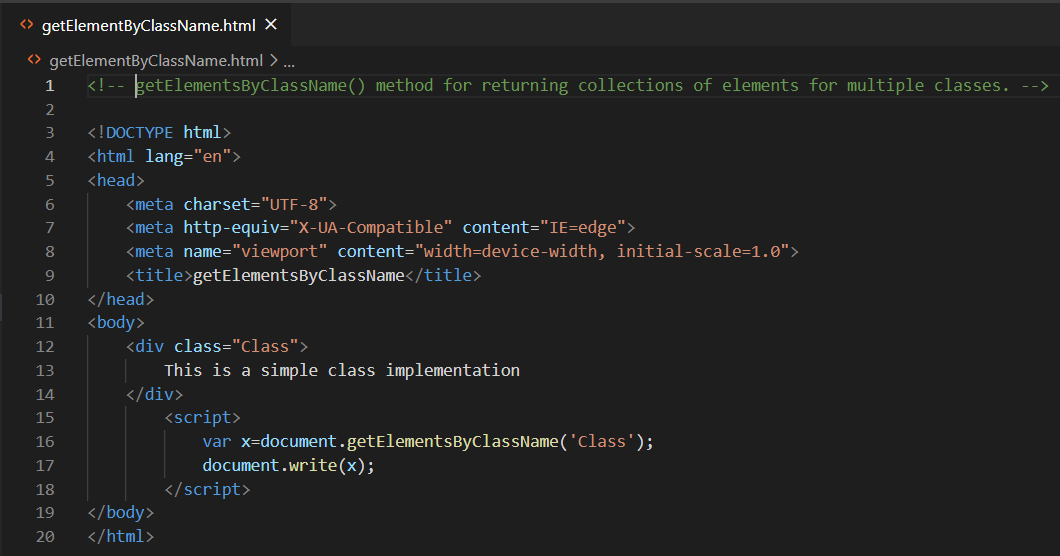
**getElementsByClassName():** It matches the elements with the specified class name, and returns a set of the matched elements. The returned elements are live HTML collection of elements. These live elements can be further updated if any changes are made in the Document Object Model.

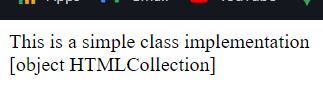
**querySelector():** It returns only a single element that matches the specified classname. If it does not find any matching element, it returns null.

The main point to understand is that all the above-described methods return either one element or a list, but the getELementsByClassName() method serves the **dynamic** updation, and the other two methods serve for the **static**

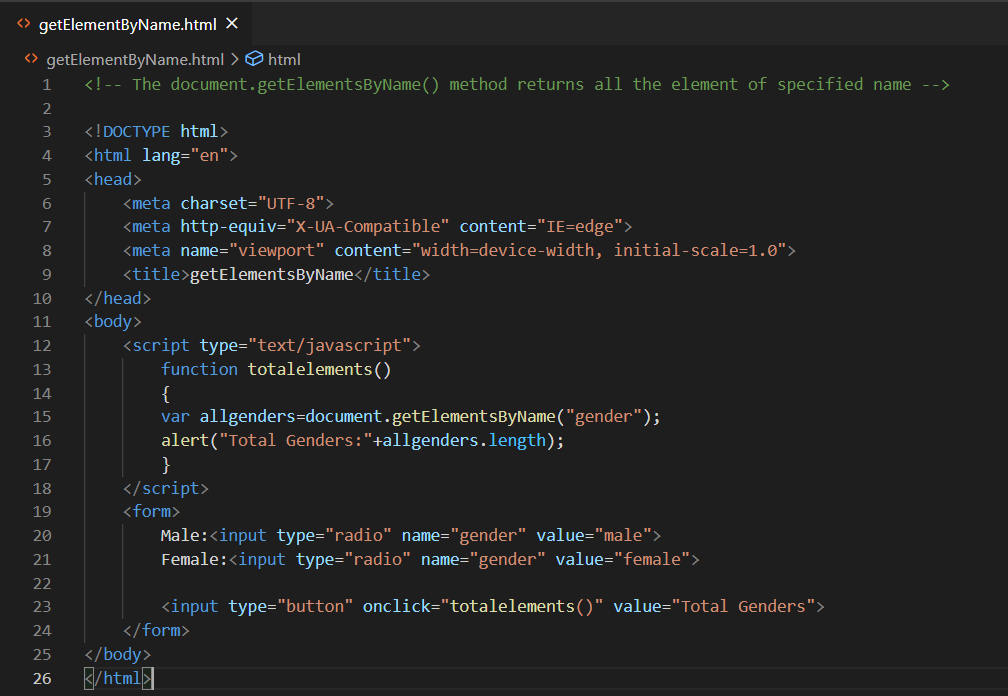
.

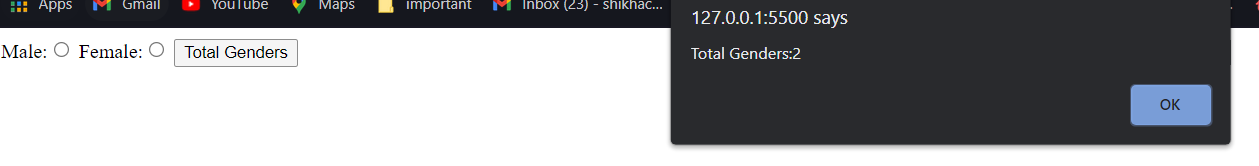
# document.getElementByClassName() method:



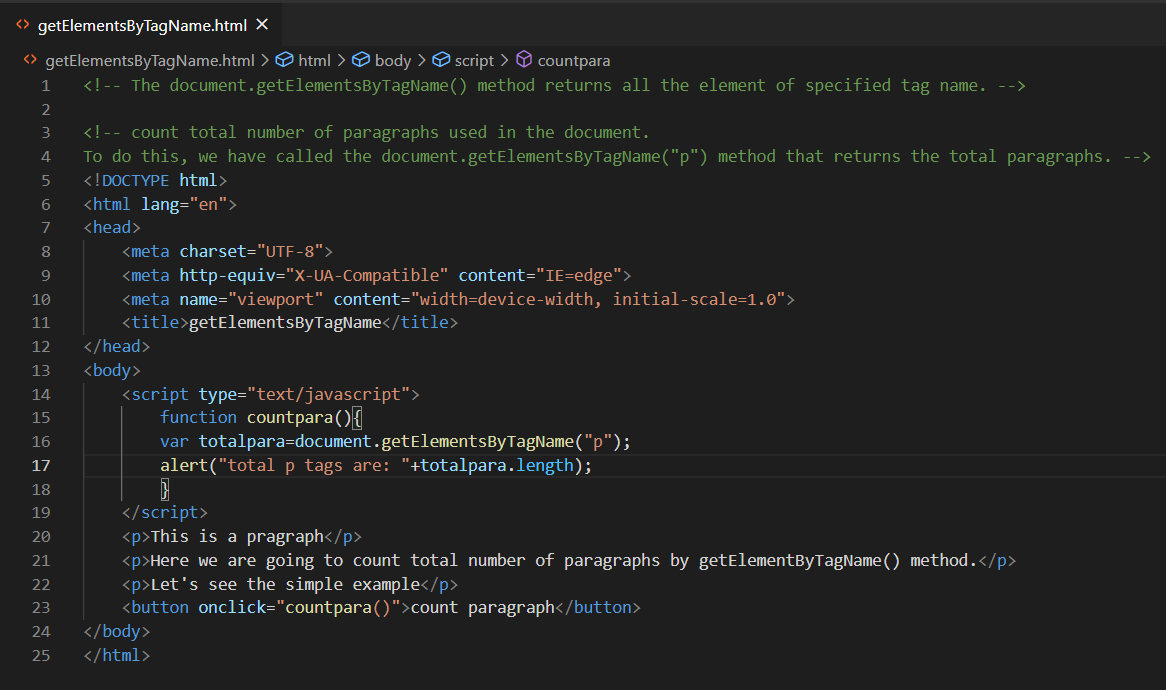


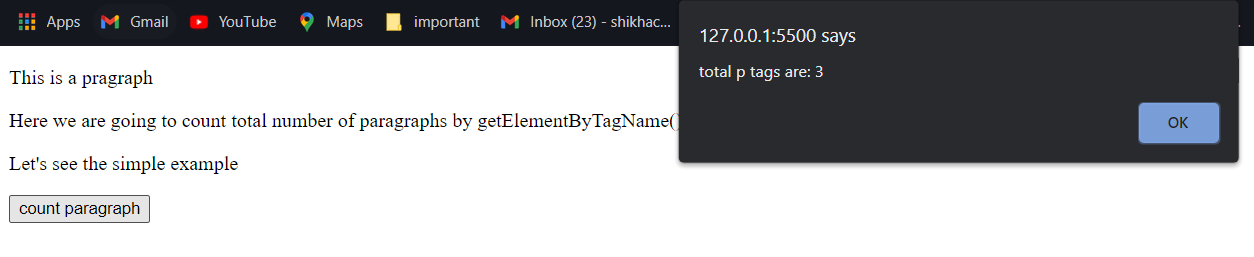
# document.getElementByName() method:





# document.getElementByTagName() method:

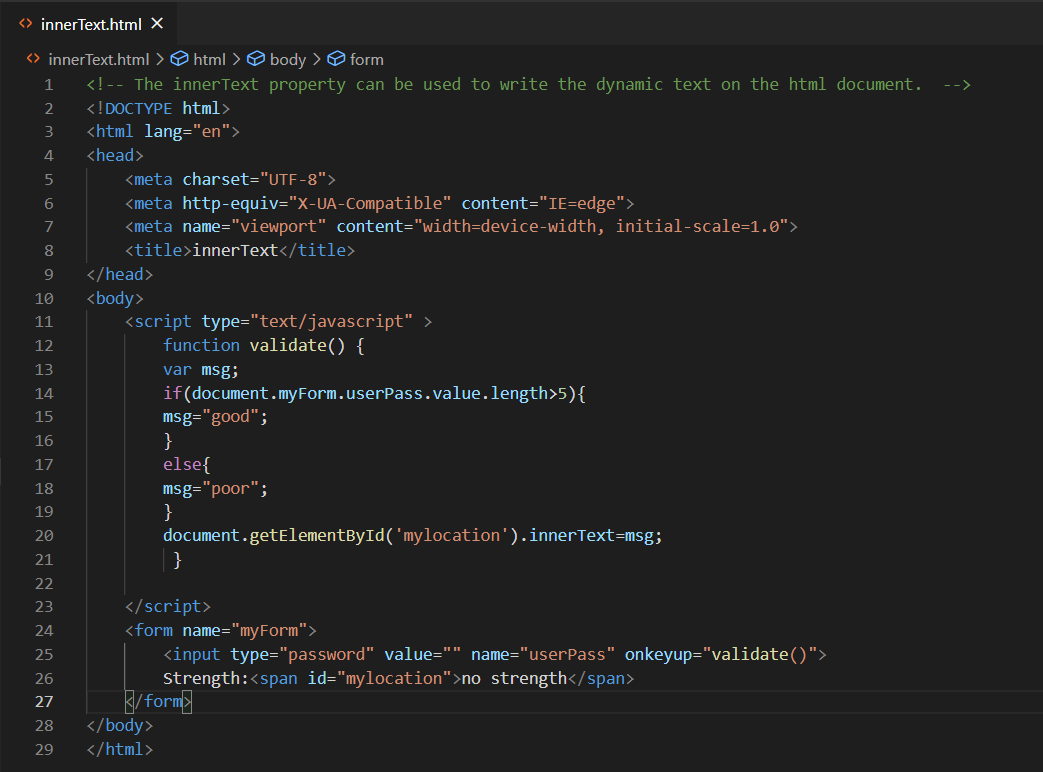




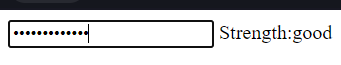
**Javascript - innerText**

The **innerText** property can be used to write the dynamic text on the html document. Here, text will not be interpreted as html text but a normal text.

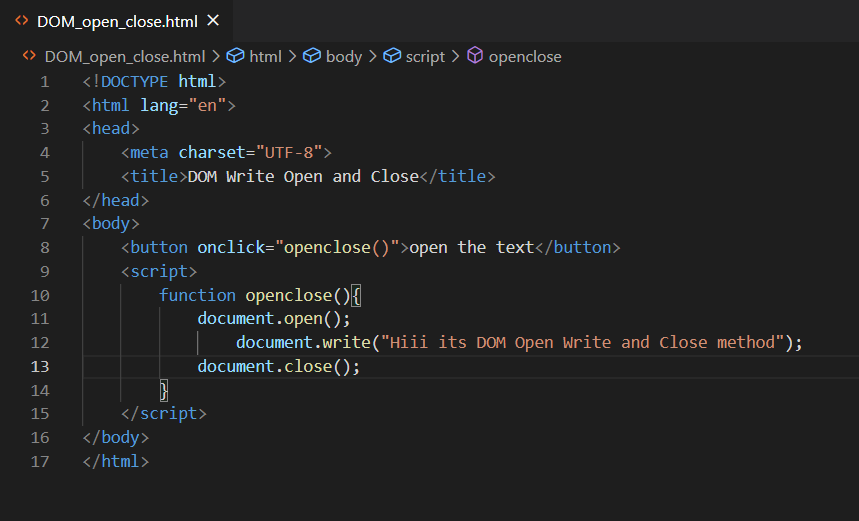
It is used mostly in the web pages to generate the dynamic content such as writing the validation message, password strength etc.

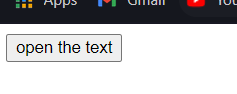


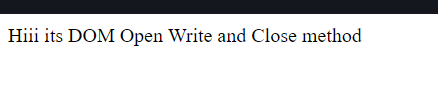




**Write Open Close**







**JavaScript OOPs**

**Object**  : Any entity that has state and behavior is known as an object

**Class** : CLASS are a blueprint or a set of instructions to build a specific type of object

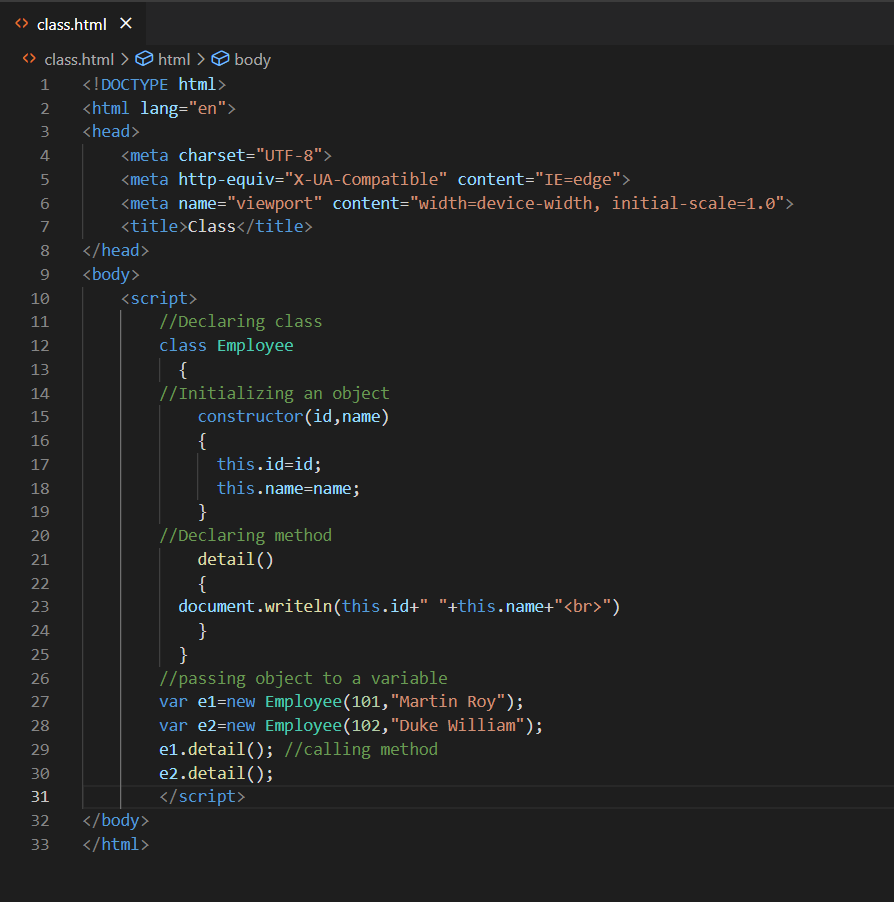
**Inheritance**: The JavaScript inheritance is a mechanism that allows us to create new classes on the basis of already existing classes

**Polymorphism** : The polymorphism is a core concept of an object-oriented paradigm that provides a way to perform a single action in different forms. It provides an ability to call the same method on different JavaScript objects.

**Abstraction** : An abstraction is a way of hiding the implementation details and showing only the functionality to the users. In other words, it ignores the irrelevant details and shows only the required one.

**Encapsulation** : The JavaScript Encapsulation is a process of binding the data (i.e. variables) with the functions acting on that data.

Class:





object:

There are 3 ways to create objects.

1. **By object literal**

<script>

Emp = {id:102,name:"Shyam Kumar",salary:40000}

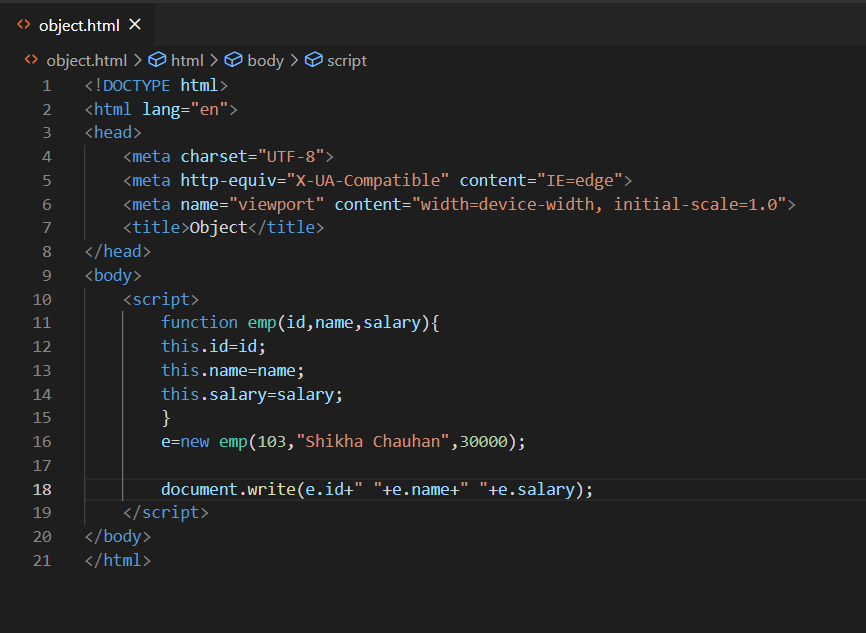
document.write(emp.id+" "+emp.name+" "+emp.salary);

</script>

1. **By creating instance of Object directly (using new keyword)**

var objectname= new Object();

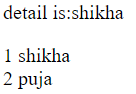
1. **By using an object constructor (using new keyword)**



Constructor:

* **this keyword** refers to the current object.





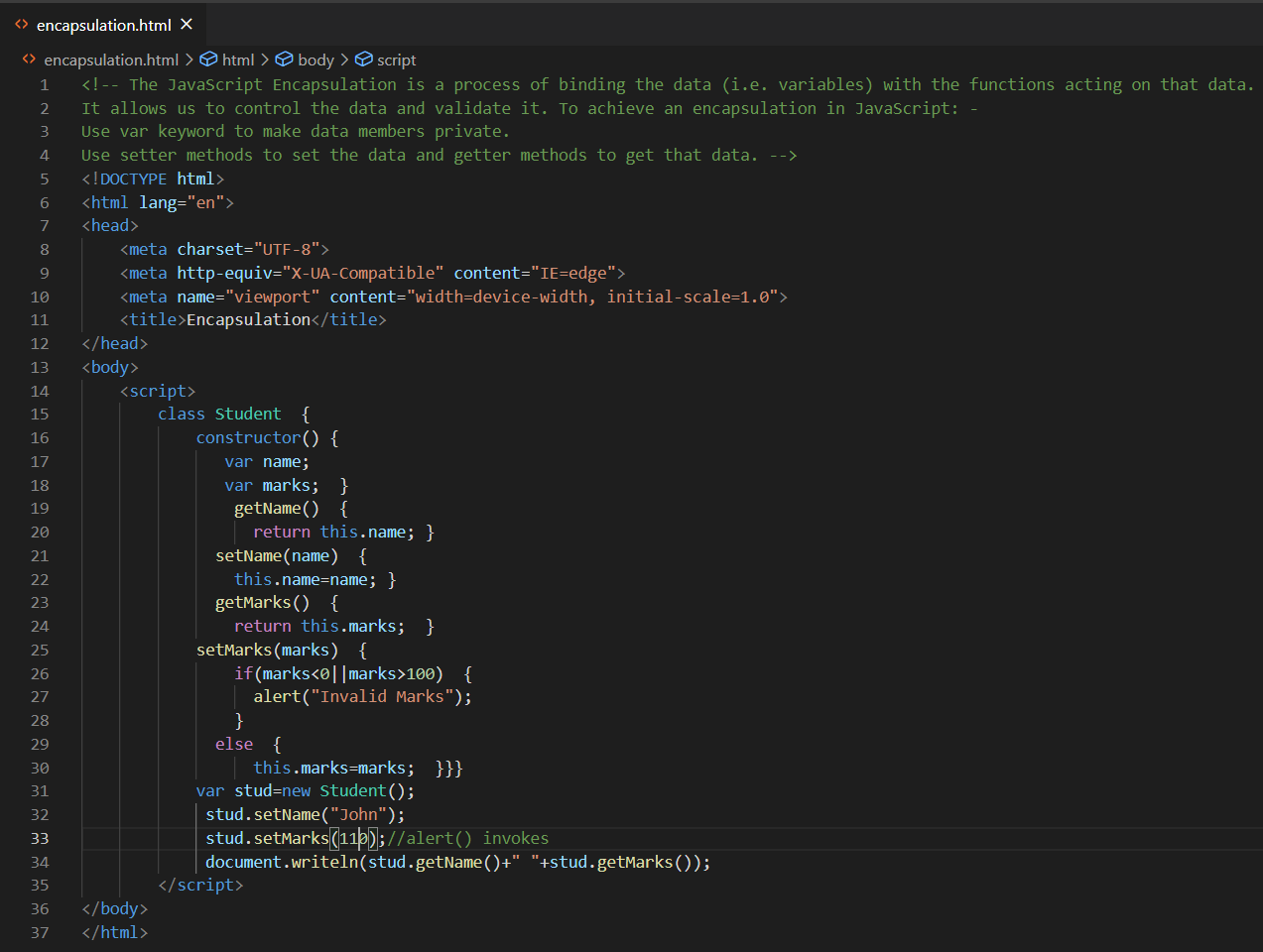
**Encapsulation:**

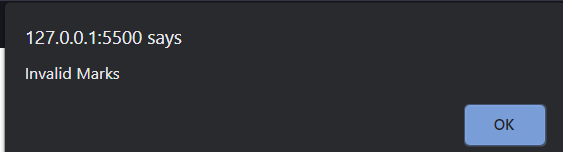
The JavaScript Encapsulation is a process of binding the data (i.e. variables) with the functions acting on that data. It allows us to control the data and validate it. To achieve an encapsulation in JavaScript: -

* Use var keyword to make data members private.
* Use setter methods to set the data and getter methods to get that data.

The encapsulation allows us to handle an object using the following properties:

**Read/Write** - Here, we use setter methods to write the data and getter methods read that data.





# Inheritance:

The JavaScript inheritance is a mechanism that allows us to create new classes on the basis of already existing classes. It provides flexibility to the child class to reuse the methods and variables of a parent class.

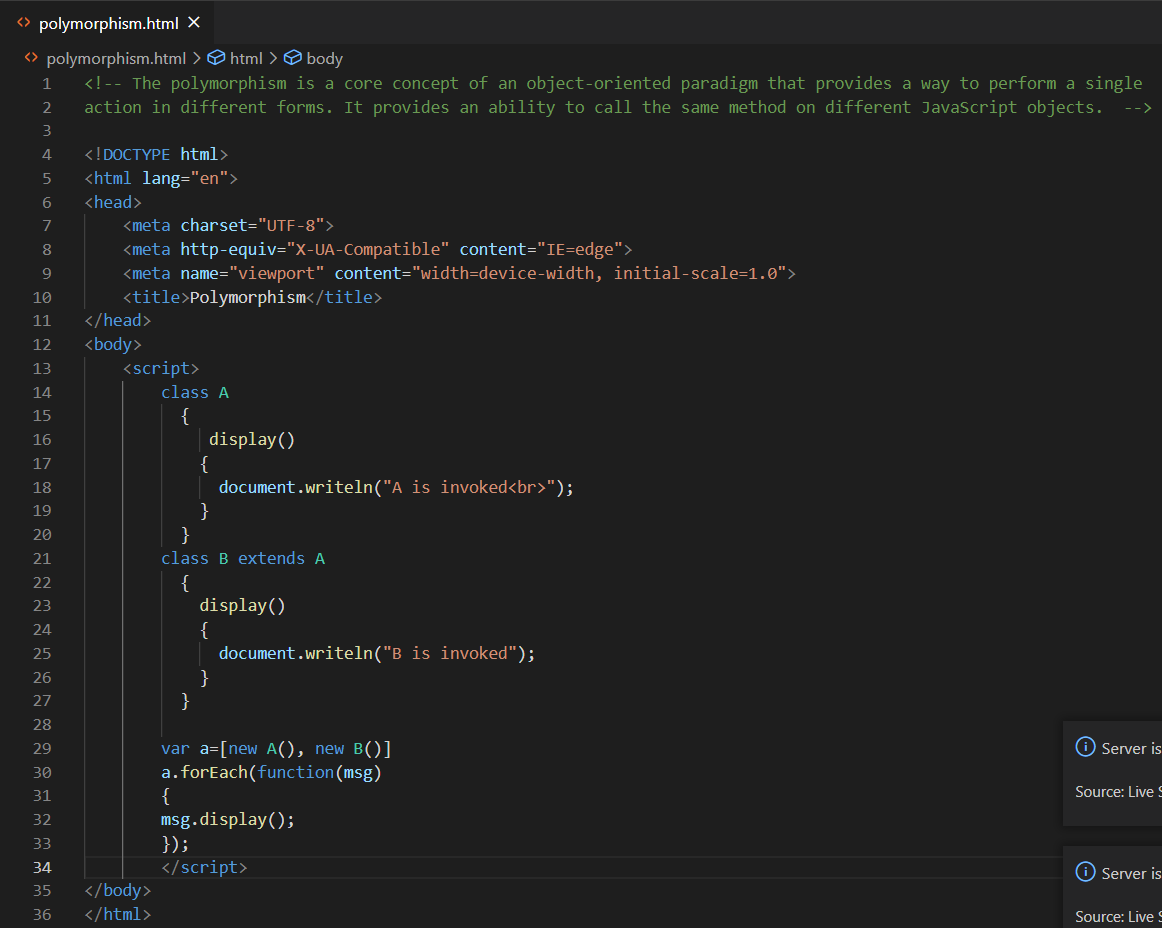
The JavaScript **extends** keyword is used to create a child class on the basis of a parent class. It facilitates child class to acquire all the properties and behavior of its parent class.

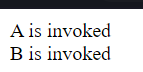




# Polymorphism:

The polymorphism is a core concept of an object-oriented paradigm that provides a way to perform a single action in different forms. It provides an ability to call the same method on different JavaScript objects.



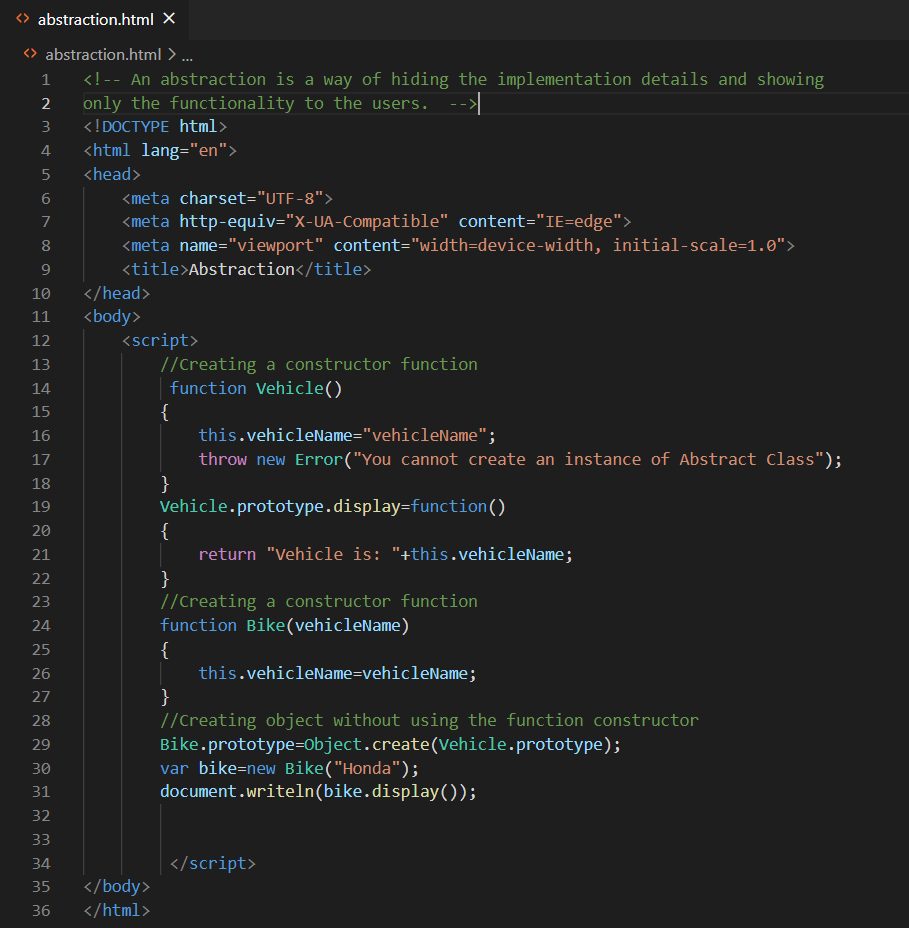


# Abstraction:

An abstraction is a way of hiding the implementation details and showing only the functionality to the users. In other words, it ignores the irrelevant details and shows only the required one.

## Points to remember

* We cannot create an instance of Abstract Class.
* It reduces the duplication of code.



Vehicle is: Honda

**Var and Let and const keyword:**

**var** i = 5;  
for (**var** i = 0; i < 10; i++) {  
  // some statements  
}  
// Here i is 10

=================================================================

**let** i = 5;  
for (**let** i = 0; i < 10; i++) {  
  // some statements  
}  
// Here i is 5

In the first example, using **var**, the variable declared in the loop redeclares the variable outside the loop.

In the second example, using **let** the variable declared in the loop does not redeclare the variable outside the loop.

When **let** is used to declare the i variable in a loop, the i variable will only be visible within the loop.

The keyword **const** is a little misleading.

It does NOT define a constant value. It defines a constant reference to a value.

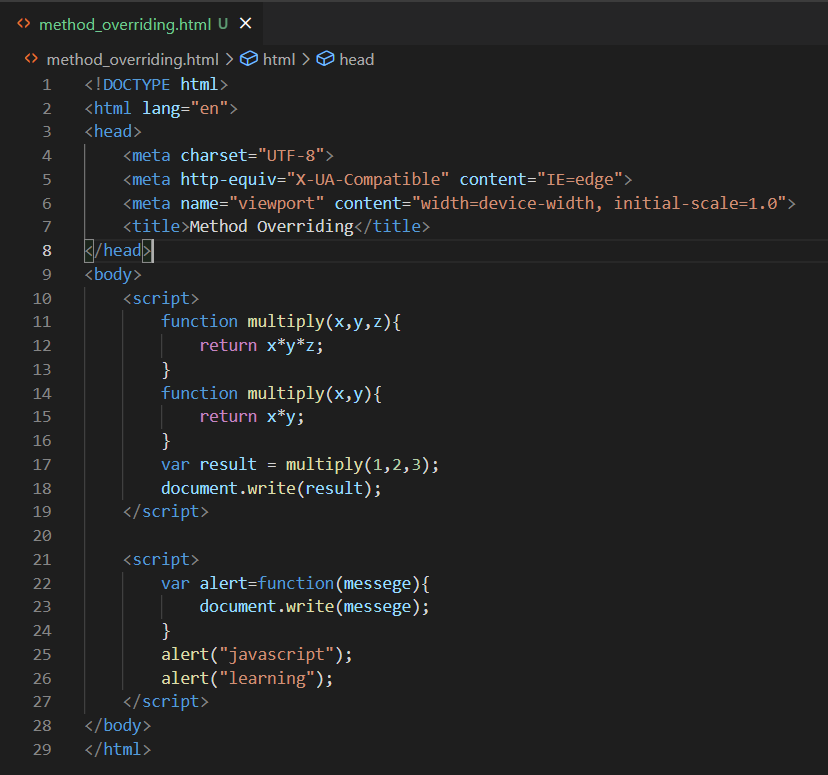
Because of this, we cannot change constant primitive values, but we can change the properties of constant objects.

You can change the properties of a constant object:

// You can create a const object:  
const car = {type:"Fiat", model:"500", color:"white"};  
  
// You can change a property:  
car.color = "red";  
  
// You can add a property:  
car.owner = "Johnson";

**Advanced JavaScript**

Method overriding:





Getter and setter property:

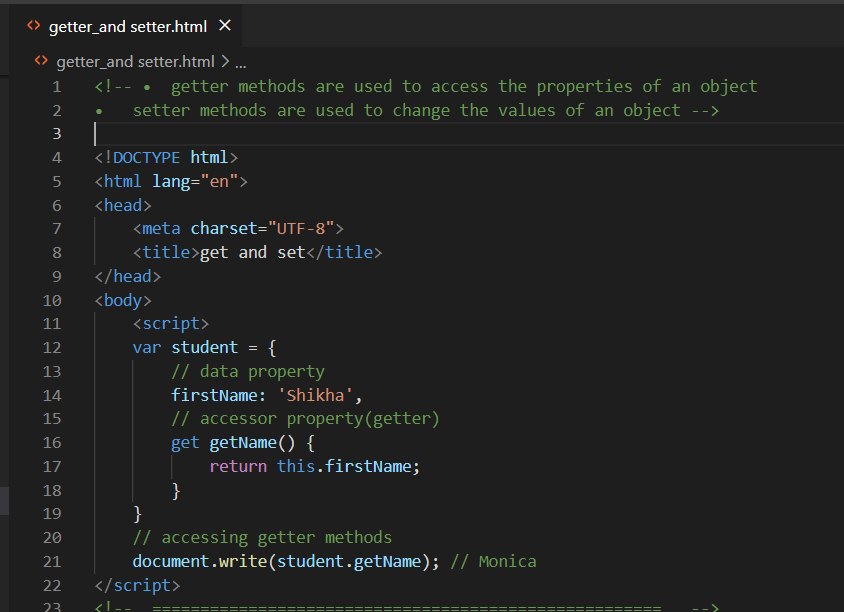
* getter methods are used to access the properties of an object
* setter methods are used to change the values of an object

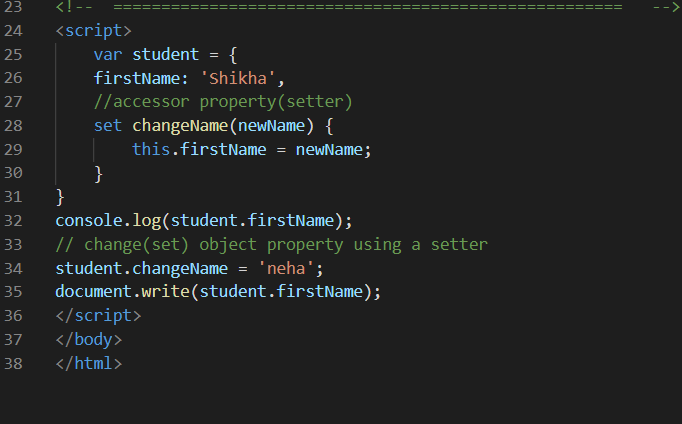
Object.defineProperty() is used to access and change the property of an object

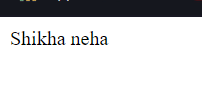
Object.defineProperty(obj, prop, descriptor)

The Object.defineProperty() method takes three arguments.

* The first argument is the objectName.
* The second argument is the name of the property.
* The third argument is an object that describes the property.





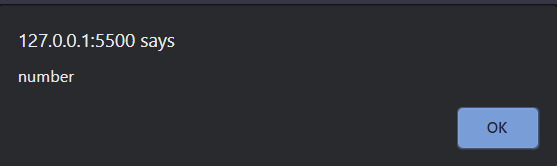


**Map:**

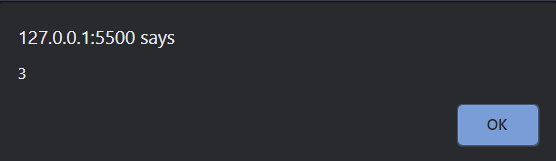
The JavaScript Map object is used to map keys to values. It stores each element as key-value pair. It operates the elements such as search, update and delete on the basis of specified key.

* A map object cannot contain the duplicate keys.
* A map object can contain the duplicate values.
* The key and value can be of any type



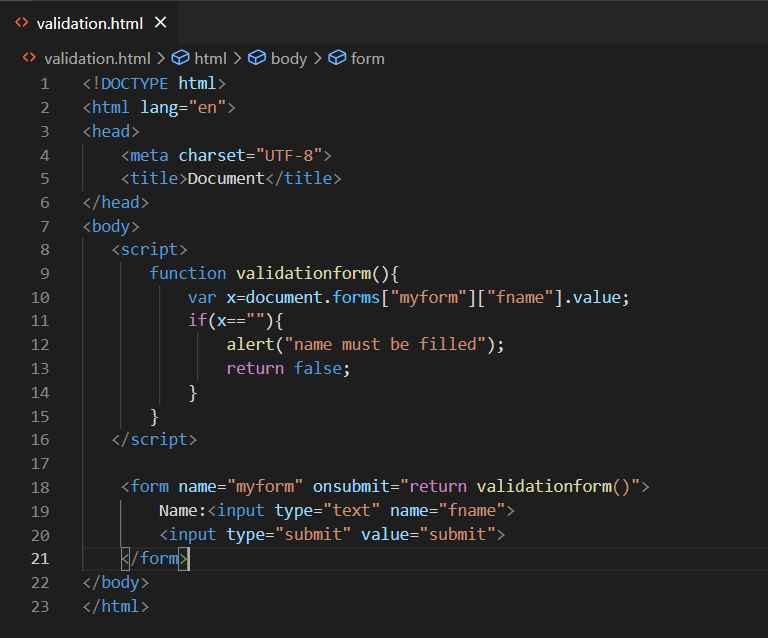


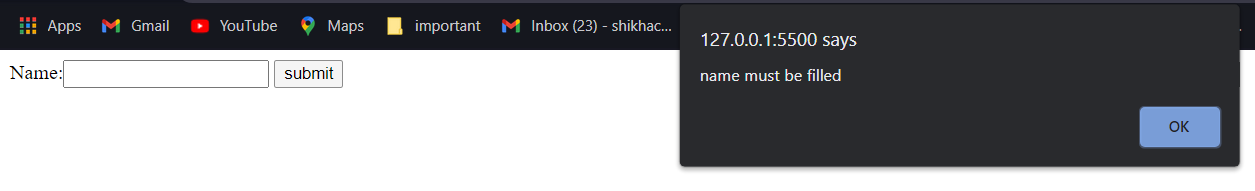




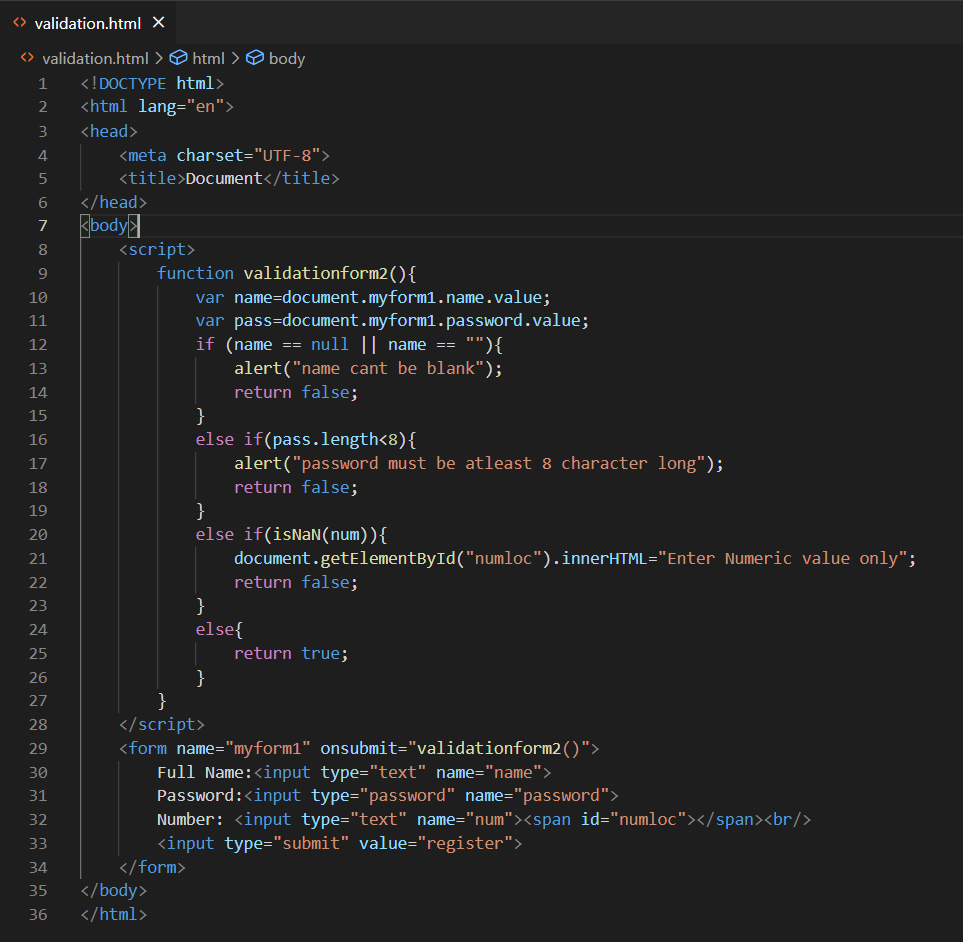
**Validation:**

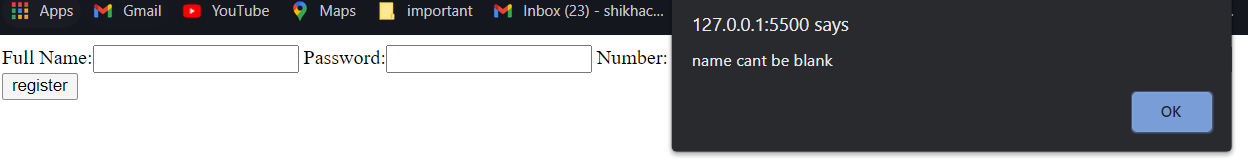
**Example1:**

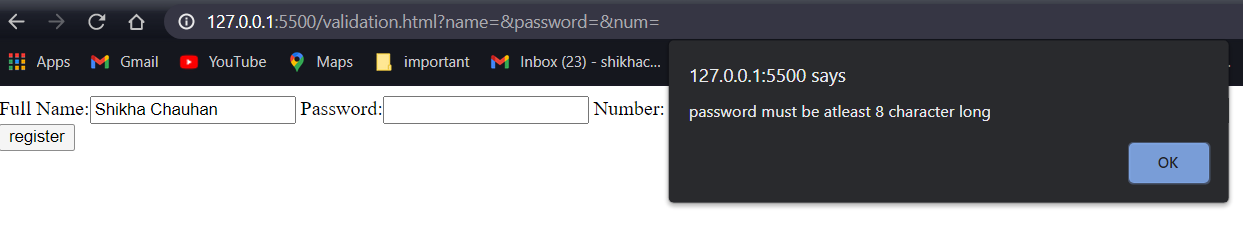




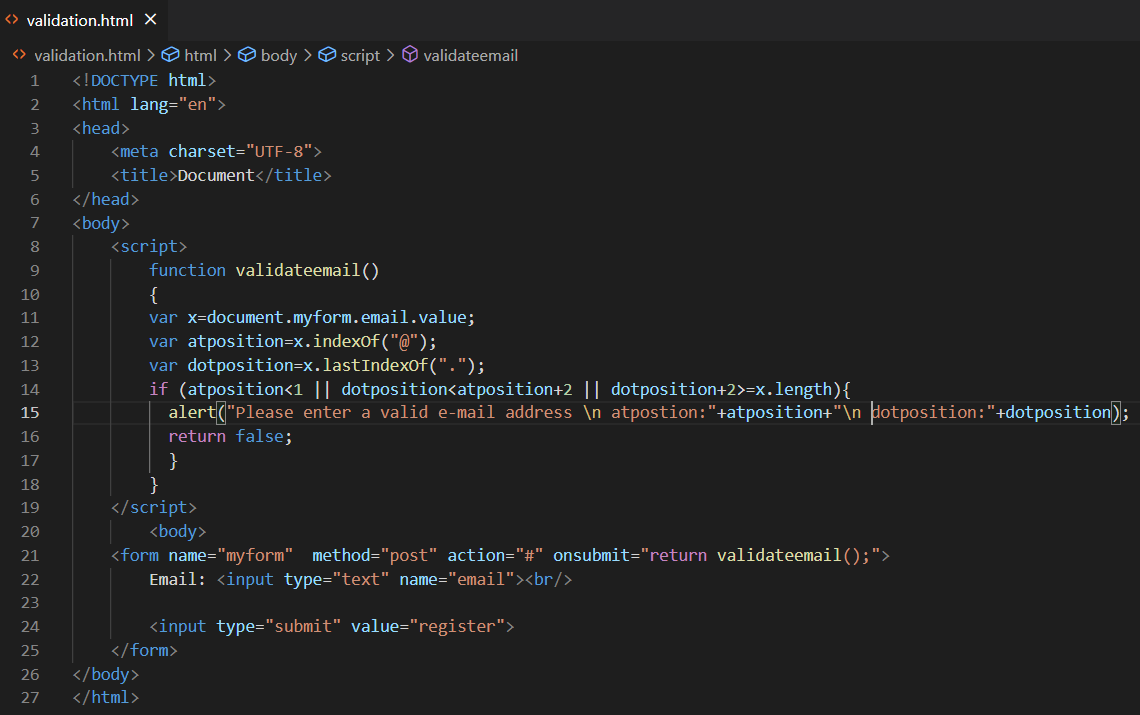
**Example2:**

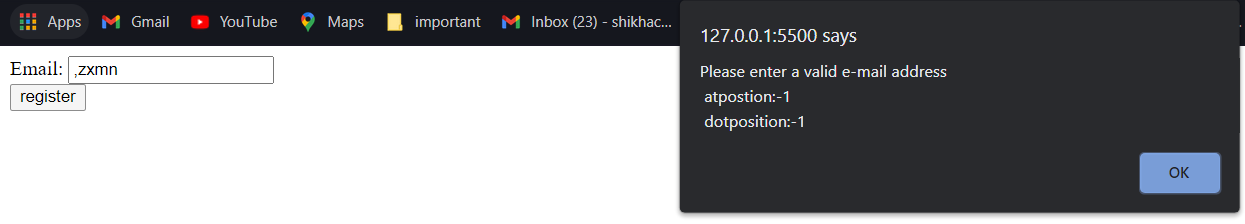






**Example 3:**





## Features in ES6:

* [The let keyword](https://www.w3schools.com/js/js_es6.asp#mark_let)
* [The const keyword](https://www.w3schools.com/js/js_es6.asp#mark_const)
* [JavaScript Arrow Functions](https://www.w3schools.com/js/js_es6.asp#mark_arrow)
* [JavaScript For/of](https://www.w3schools.com/js/js_es6.asp#mark_forof)
* [JavaScript Classes](https://www.w3schools.com/js/js_es6.asp#mark_class)
* [JavaScript Promises](https://www.w3schools.com/js/js_es6.asp#mark_promise)
* [JavaScript Symbol](https://www.w3schools.com/js/js_es6.asp#mark_symbol)
* [Default Parameters](https://www.w3schools.com/js/js_es6.asp#mark_param)
* [Function Rest Parameter](https://www.w3schools.com/js/js_es6.asp#mark_rest)
* [Array.find()](https://www.w3schools.com/js/js_es6.asp#mark_array_find)
* [Array.findIndex()](https://www.w3schools.com/js/js_es6.asp#mark_array_findIndex)
* [New Math Methods](https://www.w3schools.com/js/js_es6.asp#mark_math_methods)
* [New Number Properties](https://www.w3schools.com/js/js_es6.asp#mark_number_properties)
* [New Number Methods](https://www.w3schools.com/js/js_es6.asp#mark_number_methods)
* [New Global Methods](https://www.w3schools.com/js/js_es6.asp#mark_global_methods)
* JavaScript Modules

**Let**: keyword allows you to declare a variable with block scope.

var x = 10;

// Here x is 10

{

  let x = 2;

  // Here x is 2

}

// Here x is 10

**Const**: keyword allows you to declare a constant (a JavaScript variable with a constant value).

Constants are similar to let variables, except that the value cannot be changed.

var x = 10;

// Here x is 10

{

  const x = 2;

  // Here x is 2

}

// Here x is 10

## Arrow Functions

Arrow functions allows a short syntax for writing function expressions.

You don't need the function keyword, the return keyword, and the **curly brackets**.

// ES5

var x = function(x, y) {

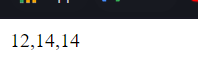
   return x \* y;

}

// ES6

const x = (x, y) => x \* y;

## 



## The For/Of Loop

The JavaScript **for/of** statement loops through the values of an iterable objects.

for/of lets you loop over data structures that are iterable such as Arrays, Strings, Maps and more.

var cars = ["BMW", "Volvo", "Mini"];

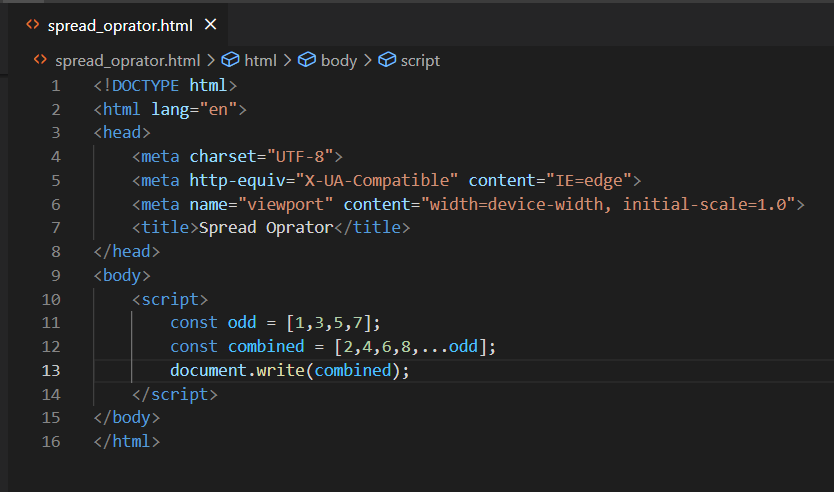
var x;

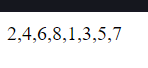
for (x of cars) {

  document.write(x + "<br >");

}

Spread Oprators:





Generators: