KodeGo

Javascript

```
if (brain!=empty)
{
   keepCoding( );
}
else{
   orderCoffee( );
}
```

KodeGo

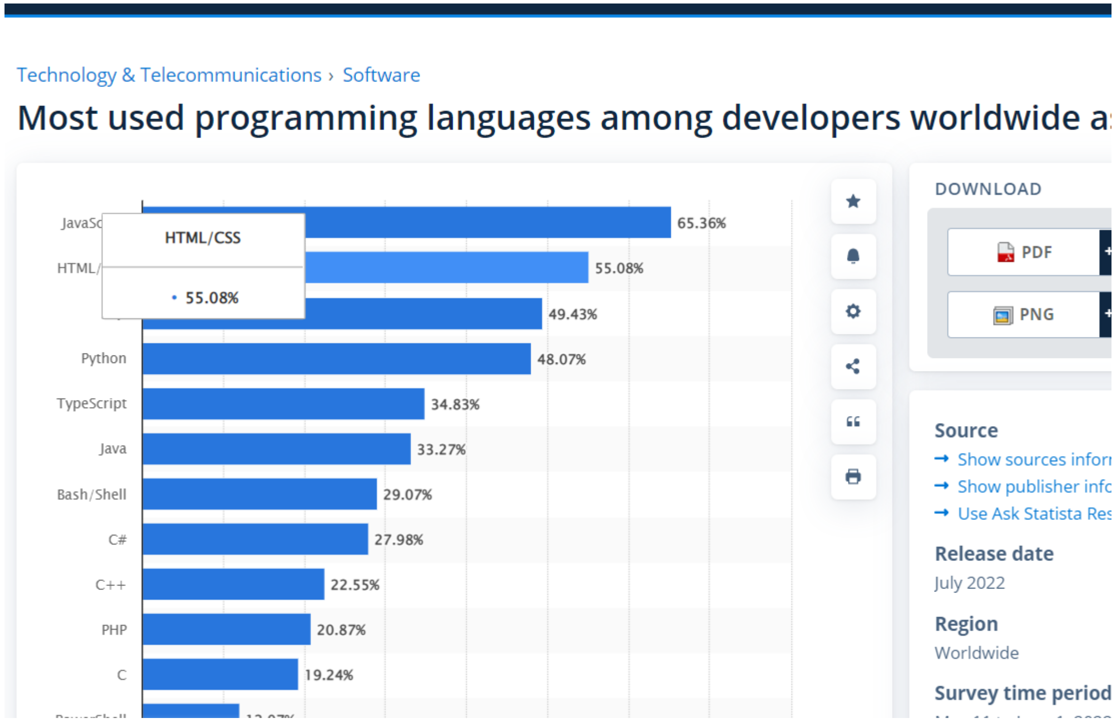
LEARNING OUTCOMES

1. **Understand the function of Javascript**
2. **Apply** three ways to add JavaScript to a web page
3. Positioning of Script inside HTML Document
4. Javascript Syntax

Why Study JavaScript?

JavaScript is one of the 3 languages all web developers must learn:

1. HTML to define the content of web pages
2. CSS to specify the layout of web pages
3. JavaScript to program the behavior of web pages


Technology & Telecommunications › Software

Most used programming languages among developers worldwide a

Source
→ Show sources inforı
→ Show publisher infc
→ Use Ask Statista Res

Release date
July 2022

Region
Worldwide

Survey time period

https://www.statista.com/statistics/793628/worldwide-developer-survey-most-used-languages/

Introduction to JavaScript

JavaScript is a programming language initially designed to interact with elements of web pages.

```
<script>
if(age > 19){
    alert("Adult");
} else{
    alert("Teenager");
}
</script>
```

Browsers use their own JavaScript Engines to execute the JavaScript code. Some commonly used browsers are:

- Chrome uses a V8 engine.
- Firefox uses the SpiderMonkey engine.
- Microsoft Edge uses the ChakraCore engine.
- Safari uses the SquirrelFish engine.

Why JavaScript?

> Easy to use
> In Demand
> Easy to learn

JavaScript Syntax

A JavaScript consists of JavaScript statements that are placed within the `<script></script>` HTML tags in a web page, or within the external JavaScript file having `.js` extension.

Adding JavaScript to Your Web Pages

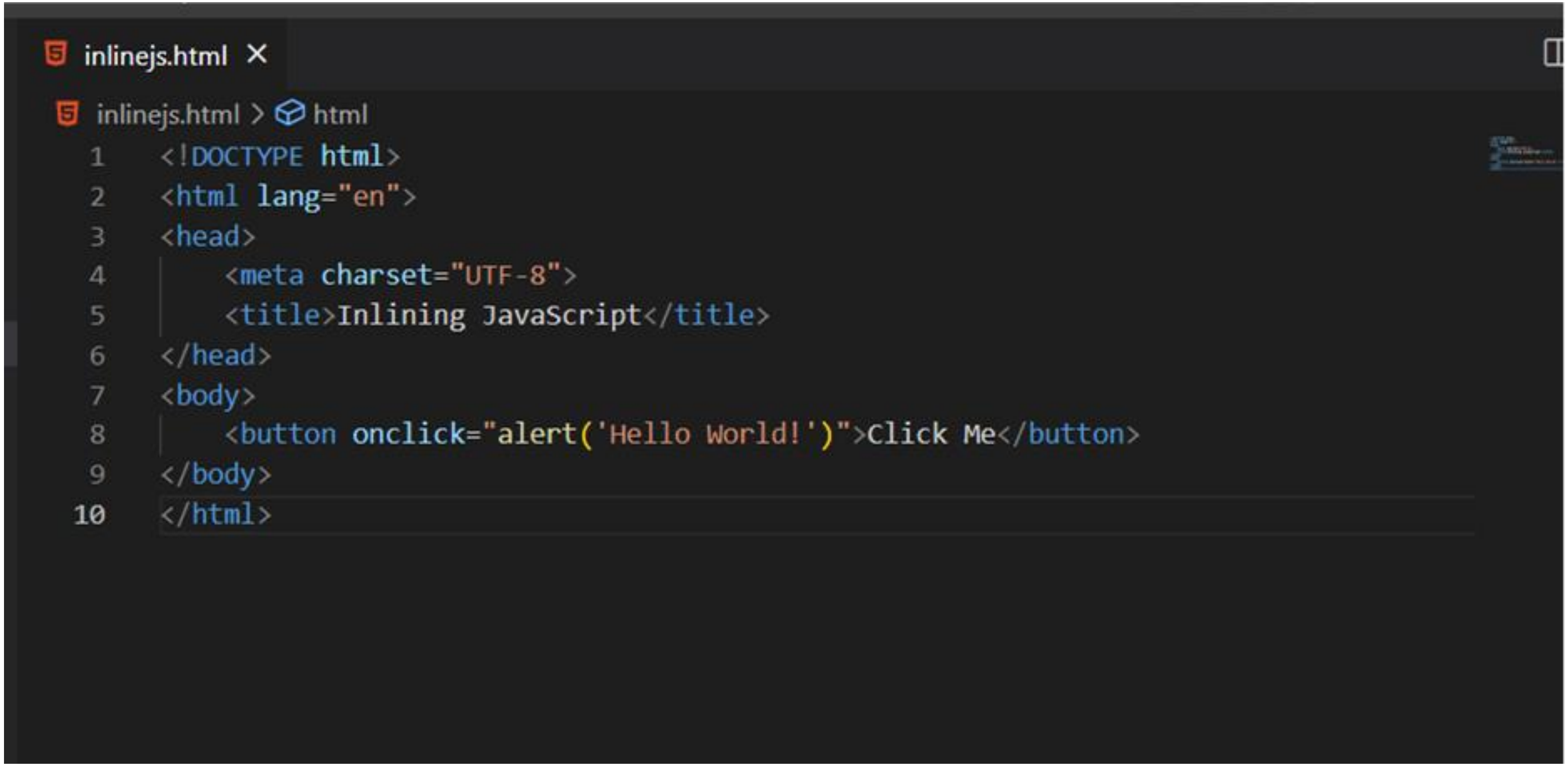
There are typically three ways to add JavaScript to a web page:

- **Embedding** the JavaScript code between a pair of `<script>` and `</script>` tag.
- Creating an **external** JavaScript file with the `.js` extension and then load it within the page through the `src` attribute of the `<script>` tag.
- Placing the JavaScript code directly **inside** an HTML tag using the special tag attributes such as `onclick`, `onmouseover`, `onkeypress`, `onload`, etc.

Placing the JavaScript Code Inline

You can also place JavaScript code inline by inserting it directly inside the HTML tag using the special tag attributes such as `onclick`, `onmouseover`, `onkeypress`, `onload`, etc.

3. Create an .html file (ex inlinejs.html)


inlinejs.html ×

inlinejs.html > html

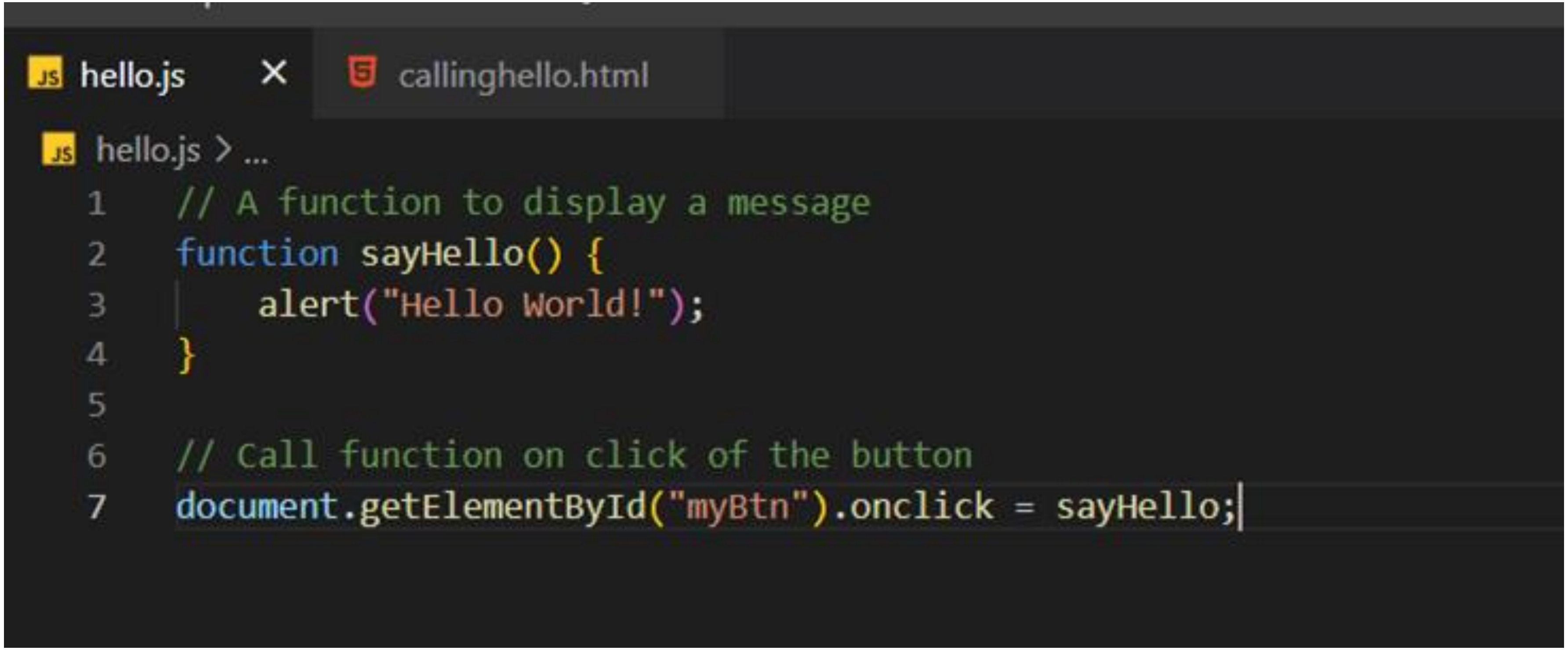
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Inlining JavaScript</title>
</head>
<body>
    <button onclick="alert('Hello World!')">Click Me</button>
</body>
</html>
```

Calling an External JavaScript File

You can also place your JavaScript code into a separate file with a `.js` extension, and then call that file in your document through the `src` attribute of the `<script>` tag, like this:

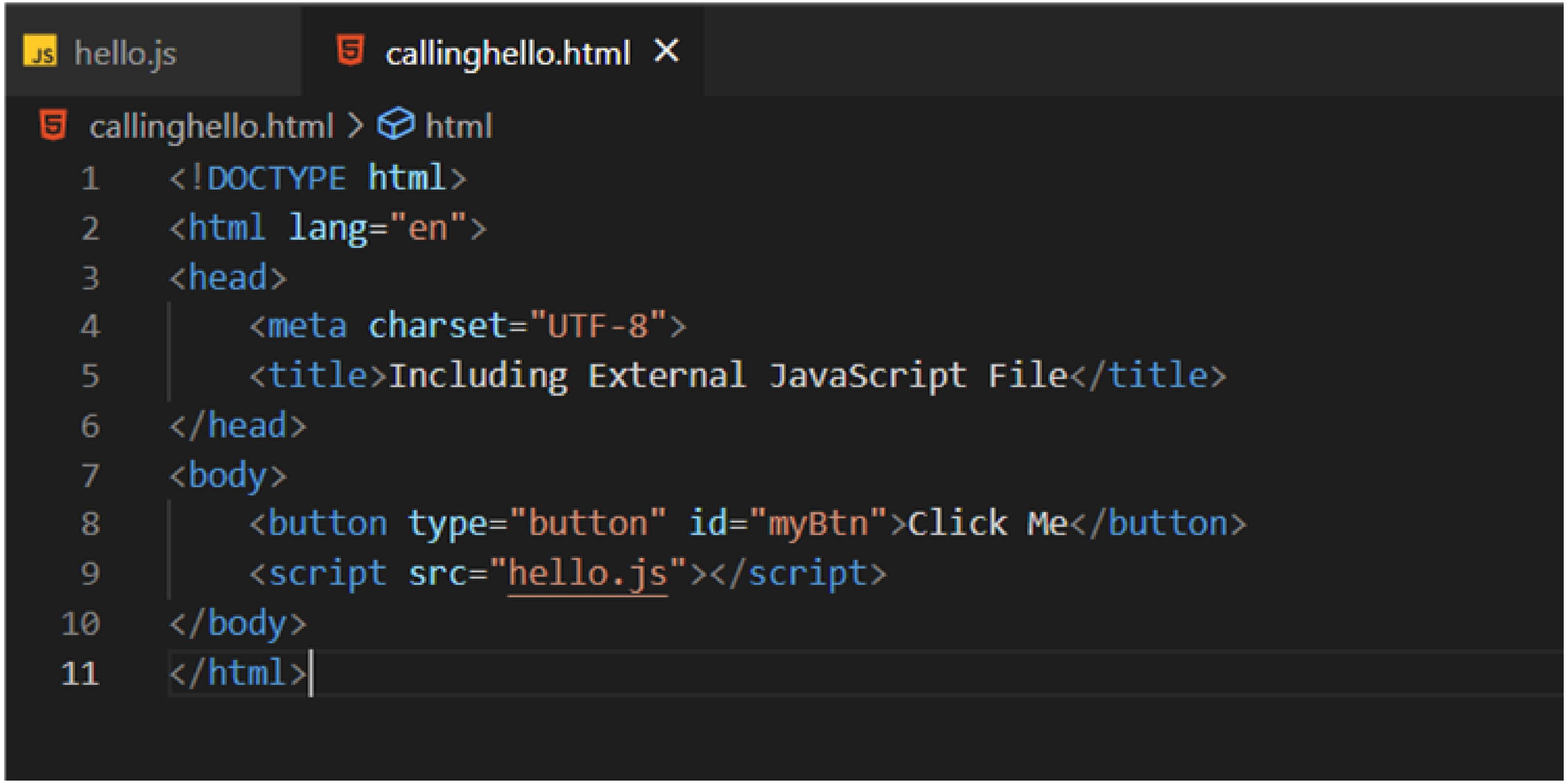
```
<script src="js/hello.js"></script>
```

1. Create a .js file (ex hello.js)



```javascript
// A function to display a message
function sayHello() {
    alert("Hello World!");
}

// Call function on click of the button
document.getElementById("myBtn").onclick = sayHello;
```

2. Create an .html file (ex callinghello.html)

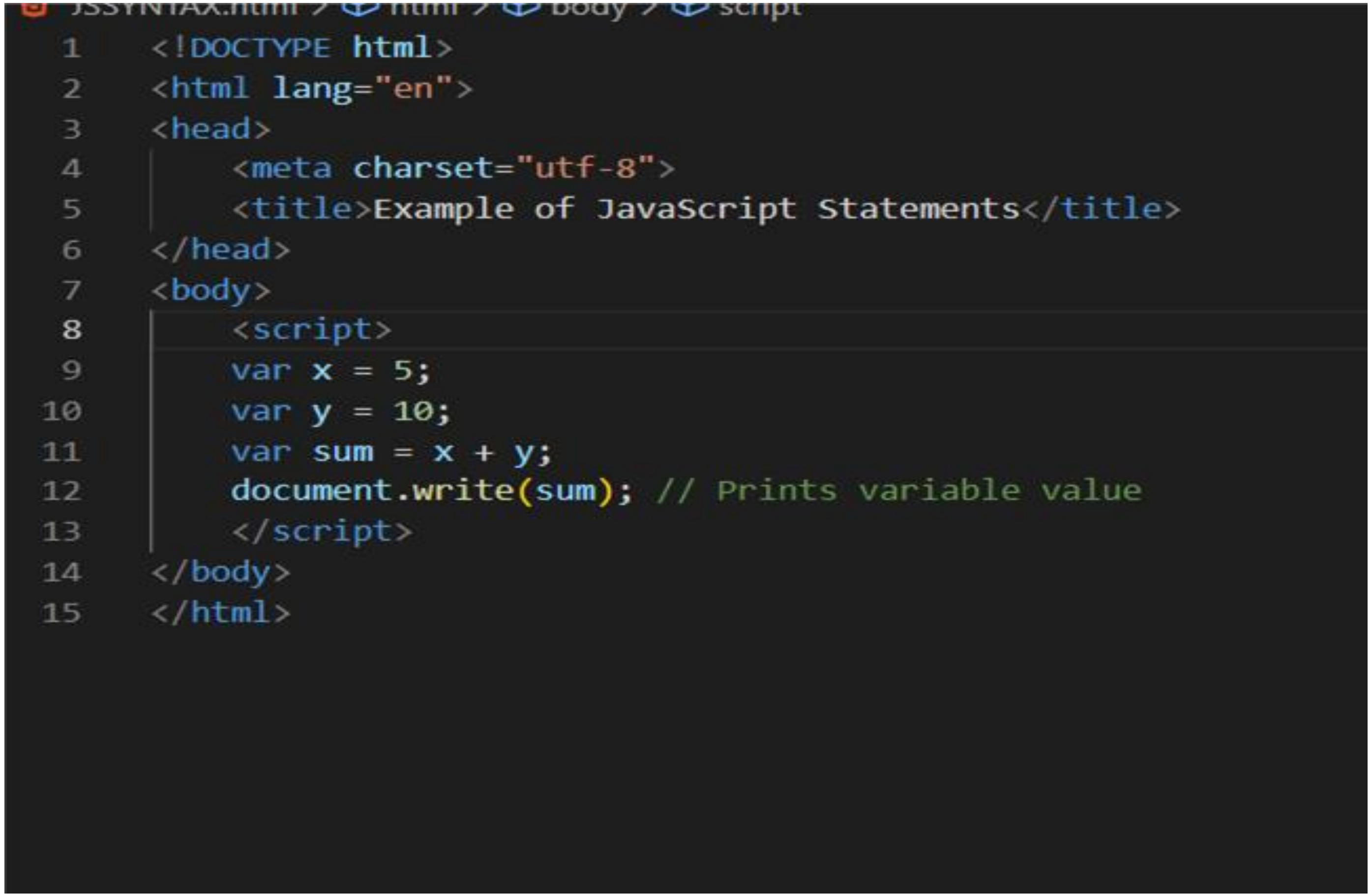


```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Including External JavaScript File</title>
</head>
<body>
    <button type="button" id="myBtn">Click Me</button>
    <script src="hello.js"></script>
</body>
</html>
```

Positioning of Script inside HTML Document

The `<script>` element can be placed in the `<head>`, or `<body>` section of an HTML document.

But ideally, scripts should be placed at the end of the body section, just before the closing `</body>` tag, it will make your web pages load faster, since it prevents obstruction of initial page rendering.

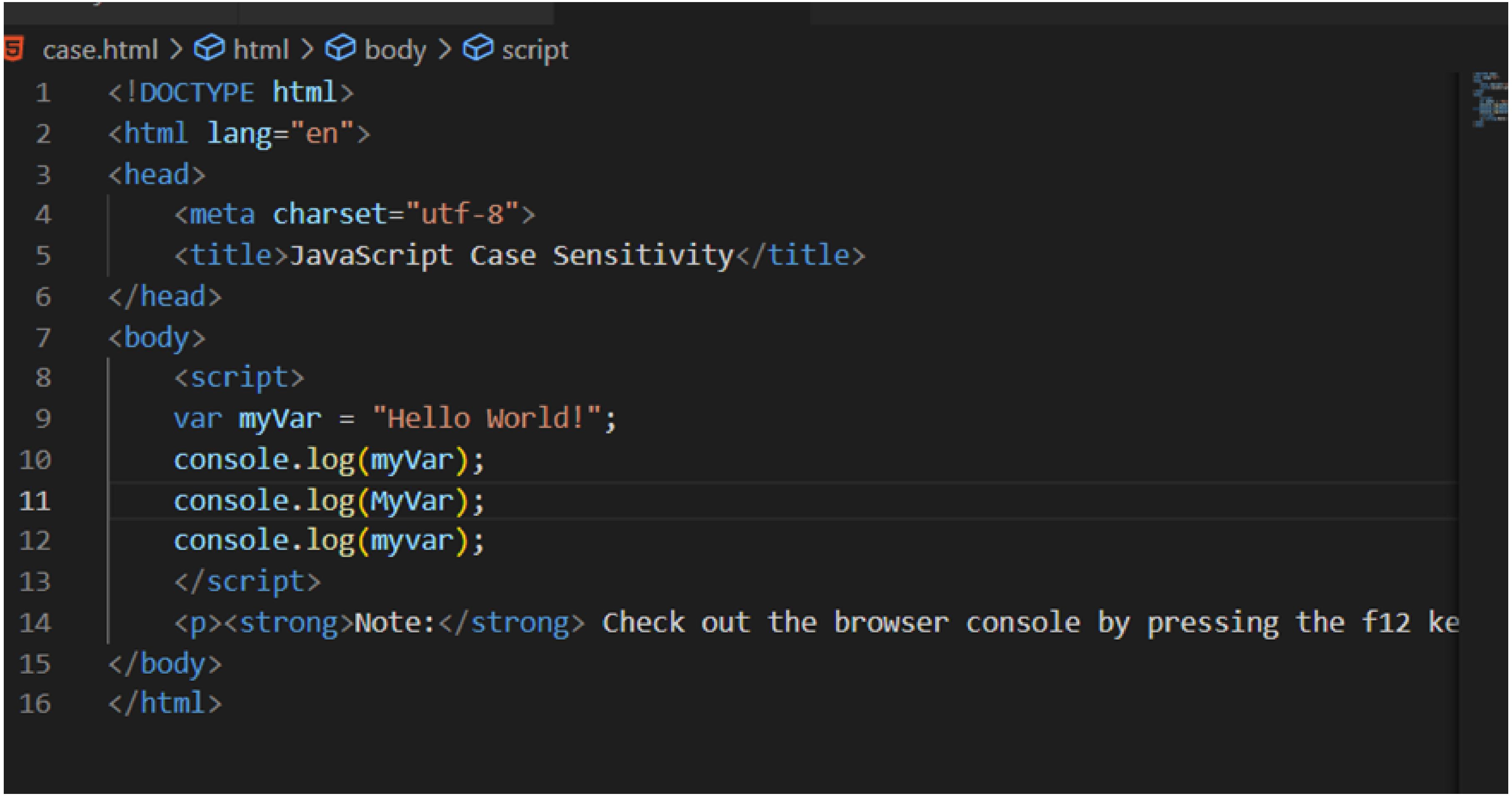


JSSYNTAX.html > html > body > script

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>Example of JavaScript Statements</title>
</head>
<body>
    <script>
    var x = 5;
    var y = 10;
    var sum = x + y;
    document.write(sum); // Prints variable value
    </script>
</body>
</html>
```

Case Sensitivity in JavaScript

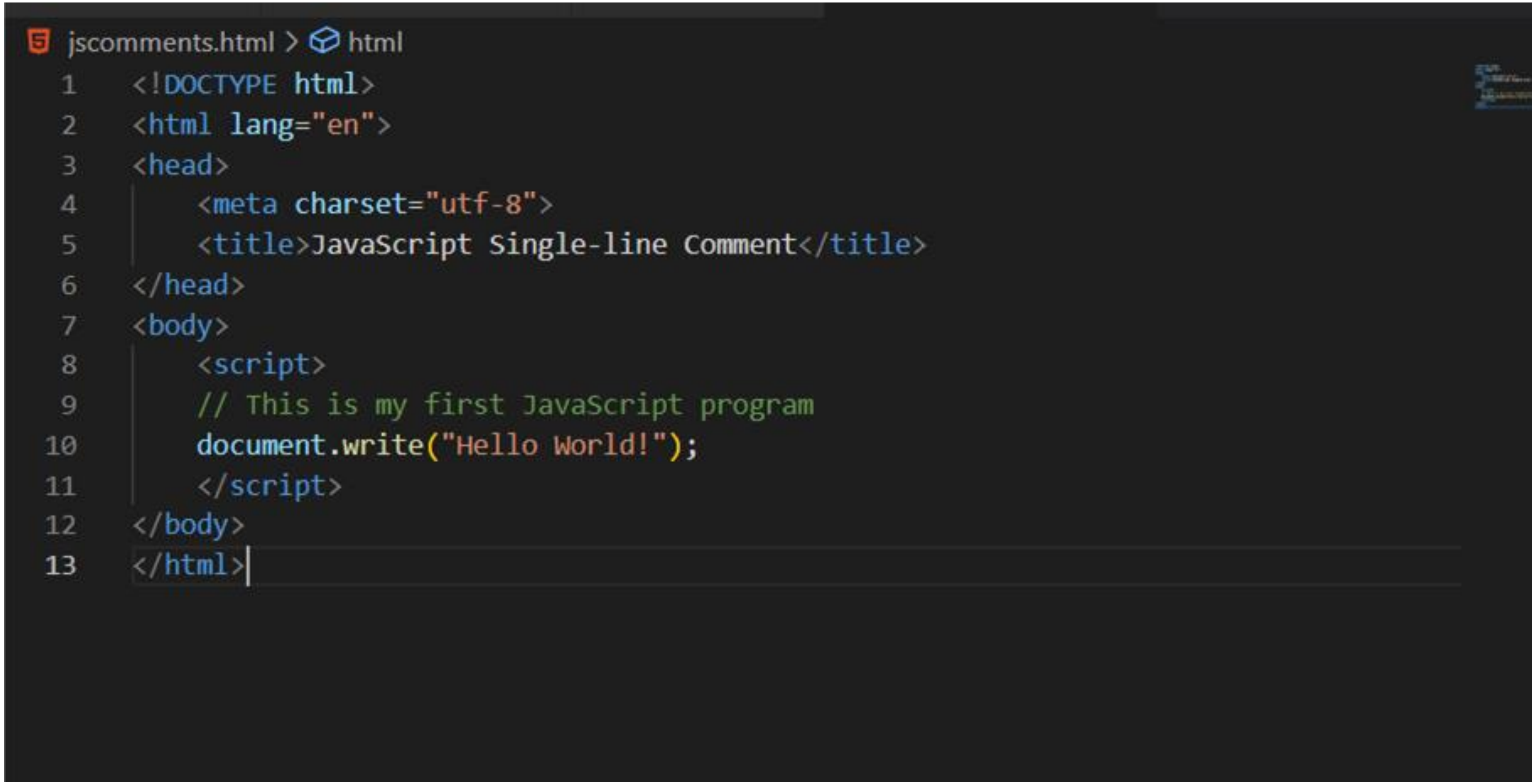
The variable `myVar` must be typed `myVar` not `MyVar` or `myvar`. Similarly, the method name `getElementById()` must be typed with the exact case not as `getElementByID()`.



```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JavaScript Case Sensitivity</title>
</head>
<body>
    <script>
    var myVar = "Hello World!";
    console.log(myVar);
    console.log(MyVar);
    console.log(myvar);
    </script>
    <p><strong>Note:</strong> Check out the browser console by pressing the f12 ke
</body>
</html>
```

JavaScript Comments

Comments are usually added with the purpose of providing extra information pertaining to source code.



```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JavaScript Single-line Comment</title>
</head>
<body>
    <script>
    // This is my first JavaScript program
    document.write("Hello World!");
    </script>
</body>
</html>
```

KodeGo

Thank you!

KodeGo

LEARNING OUTCOMES

Discuss and apply;
variables
data types
alert
document.write
JS events

Create a Simple Conversion Calculator

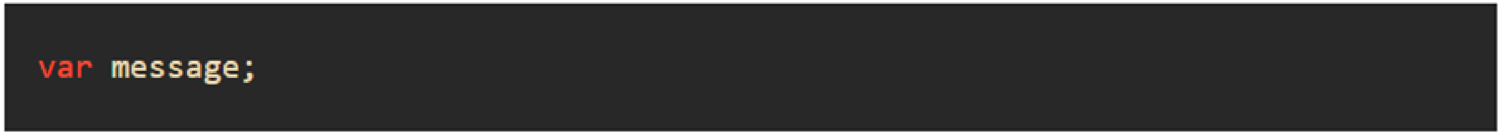
JavaScript Variables

A variable stores the data value that can be changed later on.

```
<script>
  var variable1 = “hello”;
</script>
```

KodeGo

To declare a variable, you use the var keyword followed by the variable name as follows:



```
var message;
```

A variable name can be any valid identifier. By default, the message variable has a special value undefined if you have not assigned a value to it.

A variable name can be any valid identifier. By default, the message variable has a special value undefined if you have not assigned a value to it.

Variable names follow these rules:

- Variable names are case-sensitive. This means that the message and Message are different variables.
- Variable names can only contain letters, numbers, underscores, or dollar signs and cannot contain spaces. Also, variable names must begin with a letter, an underscore (_) or a dollar sign ($).
- Variable names cannot use the reserved words.

Types of Variable

Let variable

The let keyword signals that the variable can be **re-assigned** a different value.

```
<script>
    let variable1 = “hello”;
    variable1 = “hi”;
    //will change the initial value and
    will not have any error
</script>
```

Constant variable

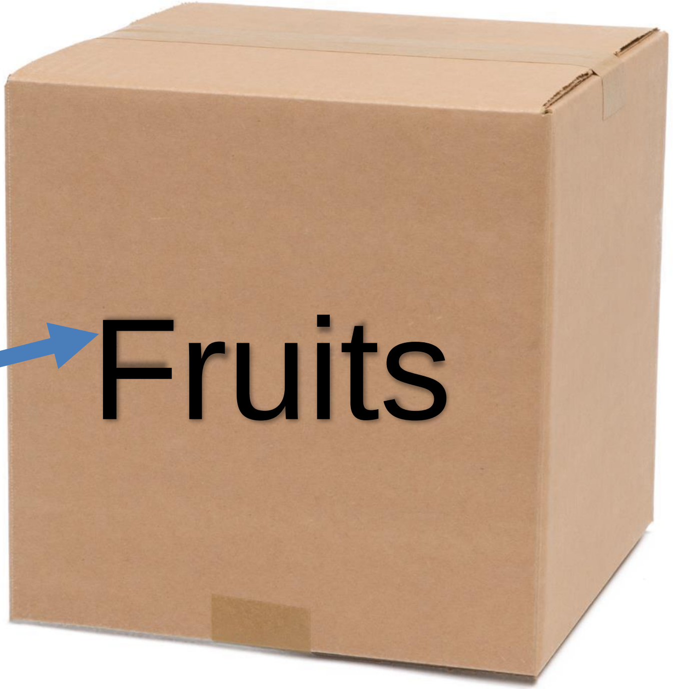
a const variable **cannot be reassigned** because it is constant.

```
<script>
    const variable1 = “hello”;
    variable1 = “hi”;
    //will create an error
</script>
```

KodeGo

Value : mango/ apple/ banana

Variable name



Javascript Data Types

String

String is a series of characters. Strings are written with quotes.

```
<script>
    let variable1 = “string”;
</script>
```

Numbers

Numbers can be written with, or without decimals.

```
<script>
    let variable1 = 12;
</script>
```

KodeGo

Javascript Data Types

Booleans

Booleans can only have two values: **true** and **false**.

```
<script>
    let variable1 = true;
</script>
```

Array

Arrays are written with square brackets. It is used to store **multiple values** inside a **single variable.**

```
<script>
    let variable1 = [“one”, “two”, “t”];
</script>
```

KodeGo

JavaScript Outputs

innerHTML

The **id attribute** defines the HTML element and the **innerHTML property** defines the HTML content

```
<body>
   <p id="demo"></p>
  <script>
  document.getElementById("demo").innerHTML = 5 +
6;
   </script>
 </body>
```

KodeGo

document.write()

For testing purposes, it is convenient to use **document.write():**

```
<script>
  document.write(5 + 6);
</script>
```

JavaScript Outputs

alert( )

Uses and alert box to display data

window.alert( ):

```
<script>
  window.alert(5 + 6);
  </script>
```

JavaScript Events

An event is an action that occurs as per the user’s instruction as input and gives the output in response.

```
<button onclick=”alert(‘Hello World’)”>
   Click Me
</button>
```

JavaScript Events

An event is an action that occurs as per the user's instruction as input and gives the output in response.

```
<body>

<p id="demo"></p>
<button onclick="myFunction()">Click me</button>

<script>
    function myFunction() {
      document.getElementById("demo").innerHTML = "Mabuhay!";
    }
</script>

</body>
```

KodeGo

Different Types of Javascript Events

Event	Description
onclick	The user clicks an HTML element
onmouseover	The user moves the mouse over an HTML element
onmouseout	The user moves the mouse away from an HTML element
onkeydown	The user pushes a keyboard key
onload	The browser has finished loading the page

KodeGo

Javascript Arithmetic Operators

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
**	Exponentiation
/	Division
%	Modulus
++	Increment
- -	Decrement

KodeGo

JavaScript Functions

A JavaScript function is a **block of code** designed to perform a **particular task**.

The Function Data Type

A function is a block of code which will be executed only if it is called.

How to Create a Function in JavaScript

1. Use the keyword **function** followed by the name of the function.
2. After the function name, open and close parentheses.
3. After parenthesis, open and close curly braces.
4. Within curly braces, write your lines of code.

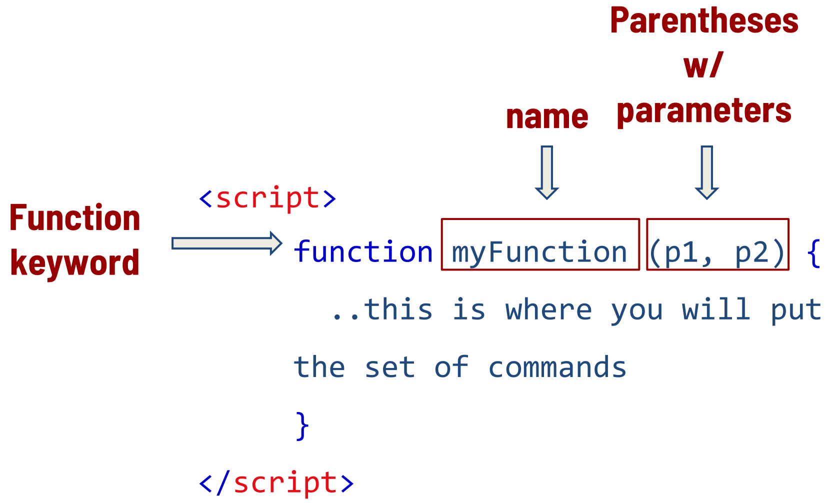
Syntax:

```
function functionname()

{

  lines of code to be executed
```

JavaScript Functions

Function Syntax

Functions are created using the **function** keyword, followed by the **function name**, followed by the **parentheses** which stores **parameters** separated by commas.



```
<script>
    function myFunction (p1, p2) {
      ..this is where you will put
    the set of commands
    }
</script>
```

KodeGo

```
<body>

<p id="demo"></p>
<button onclick="myFunction()">Click me</button>

<script>
    function myFunction() {
      document.getElementById("demo").innerHTML = "Mabuhay!";
    }
</script>

</body>
```

Exercise 6: Create a Simple Conversion Calculator

1. With the topic of your choice create a simple conversion calculator
2. The current unit must be the commonly used unit of measurement
3. Add at least three (3) unit options

Example:

Topic: Length

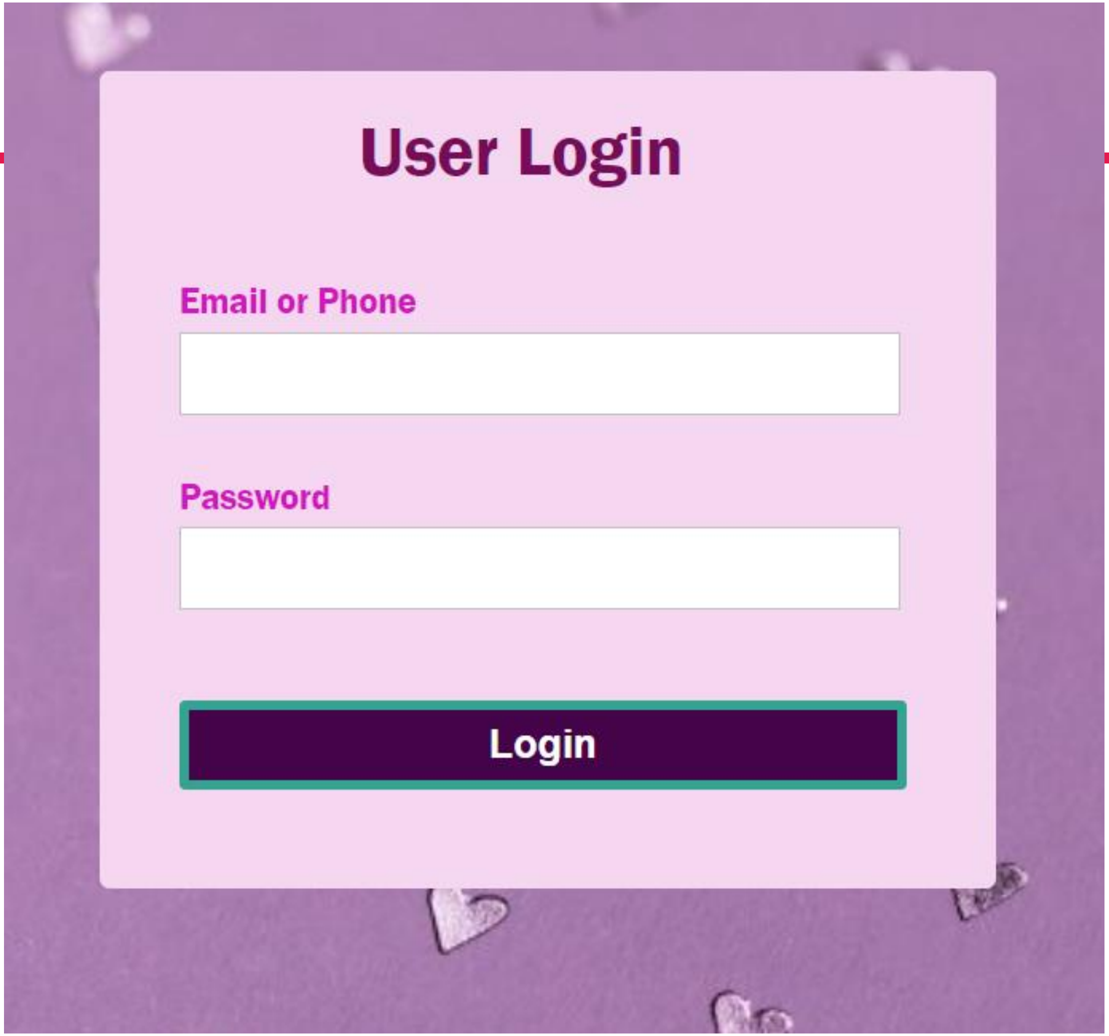
Commonly used unit: Inches

Unit options for conversion: Millimeter, Centimeter and Meter

KodeGo

Thank you!

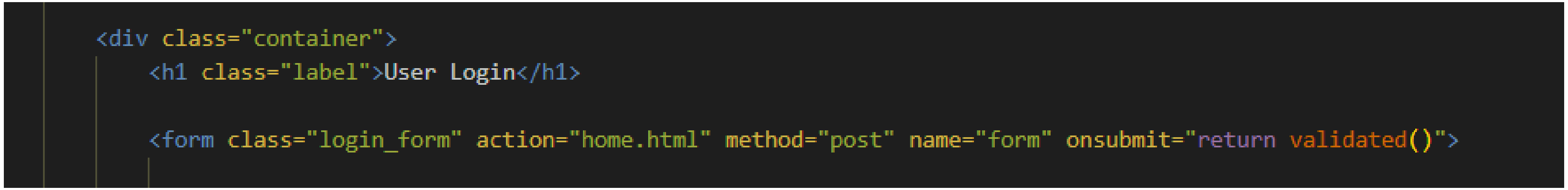
Log in
Form Validation


User Login
Email or Phone
Password
Login

Creating files

1.index.html
2.style.css
3.loginform.js
4. add an image for the background

index.html



```
<div class="container">
    <h1 class="label">User Login</h1>

    <form class="login_form" action="home.html" method="post" name="form" onsubmit="return validated()">
```

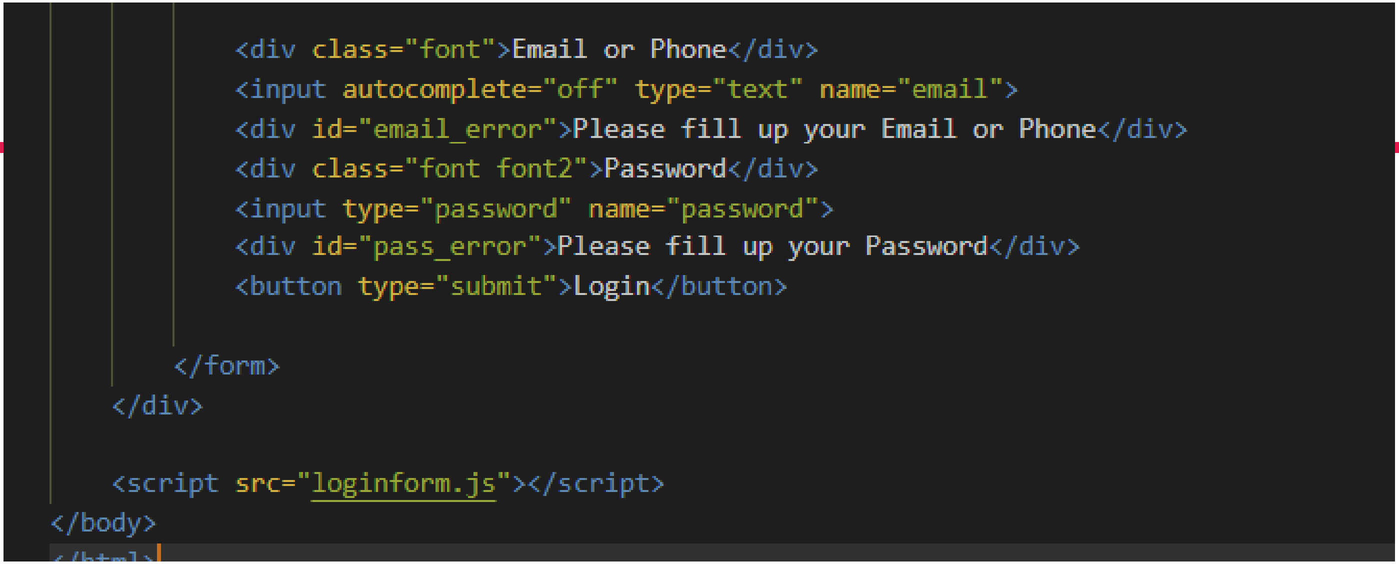
Notes:

action=" " the form data will be sent to this URL (home.html) – after the form validation

method="post" is used to send data to a server to create/update a resource.

return validated() – is a name of function you will create in JavaScript

index.html



```html
            <div class="font">Email or Phone</div>
            <input autocomplete="off" type="text" name="email">
            <div id="email_error">Please fill up your Email or Phone</div>
            <div class="font font2">Password</div>
            <input type="password" name="password">
            <div id="pass_error">Please fill up your Password</div>
            <button type="submit">Login</button>

        </form>
    </div>

    <script src="loginform.js"></script>
</body>
</html>
```

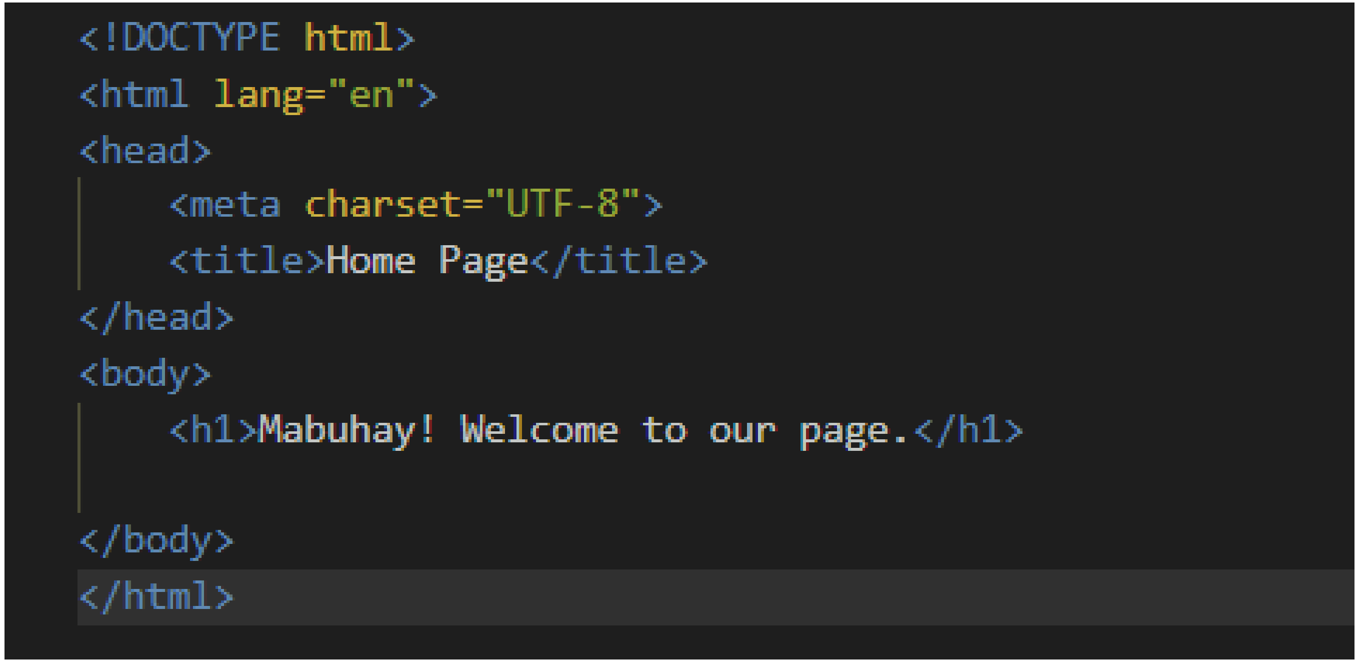
Notes:

Autocomplete allows the browser to predict the value. When a user starts to type in a field, the browser should display options to fill in the field, based on earlier typed values.

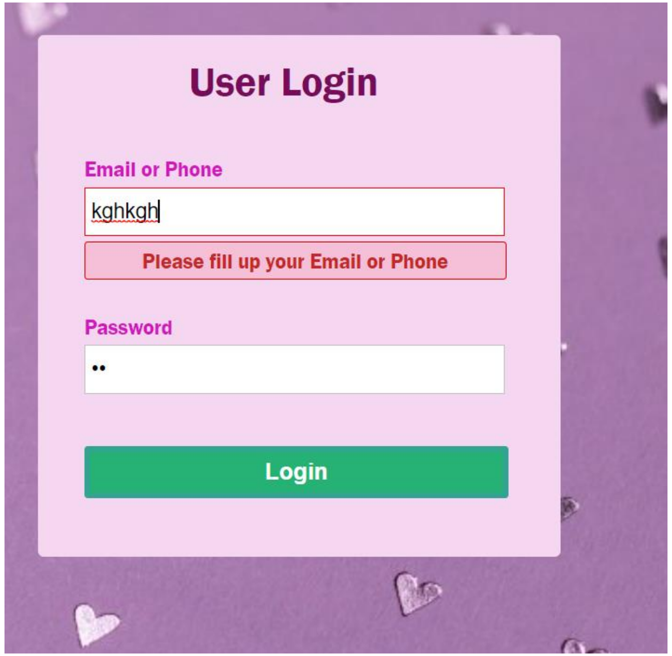
Note: The autocomplete attribute works with the following input types: text, search, url, tel, email, password, datepickers, range, and color

Source: W3schools

home.html



```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Home Page</title>
</head>
<body>
    <h1>Mabuhay! Welcome to our page.</h1>

</body>
</html>
```



User Login

Email or Phone

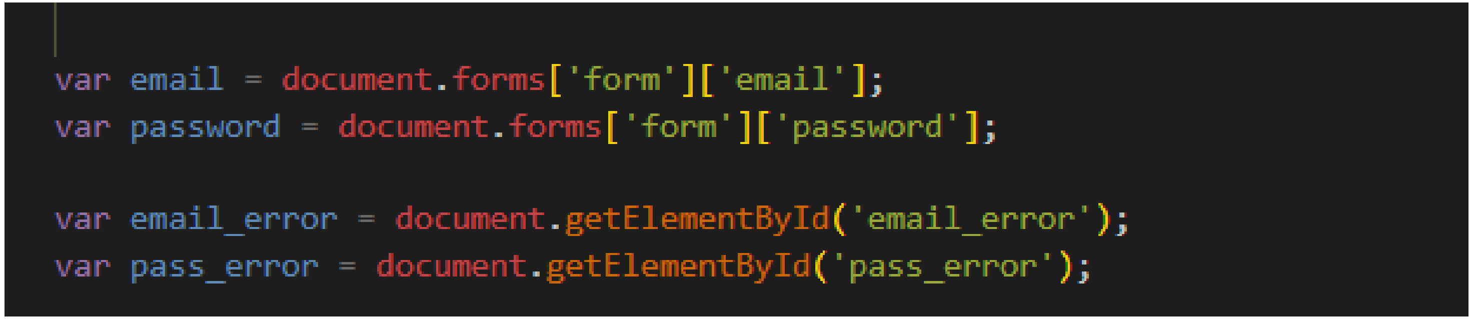
kghkgh

Please fill up your Email or Phone

Password

••

loginform.js



```
var email = document.forms['form']['email'];
var password = document.forms['form']['password'];

var email_error = document.getElementById('email_error');
var pass_error = document.getElementById('pass_error');
```

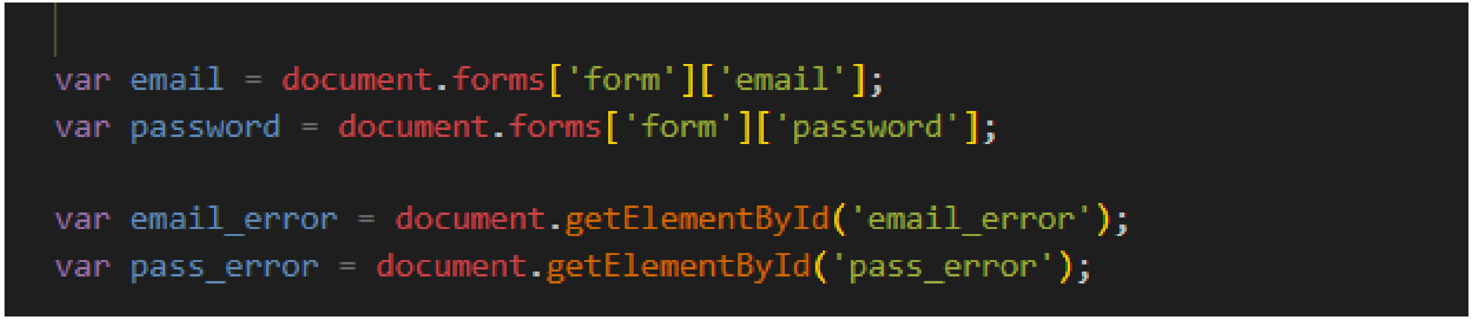
documents.forms - is an object containing all of the forms for that HTML document.

It will return a collection of all the forms within a particular page.

It will provide the value for the form element with the name ‘form’ with the name ‘email’

It will provide the value for the form element with the name ‘form’ with the name ‘password’

loginform.js

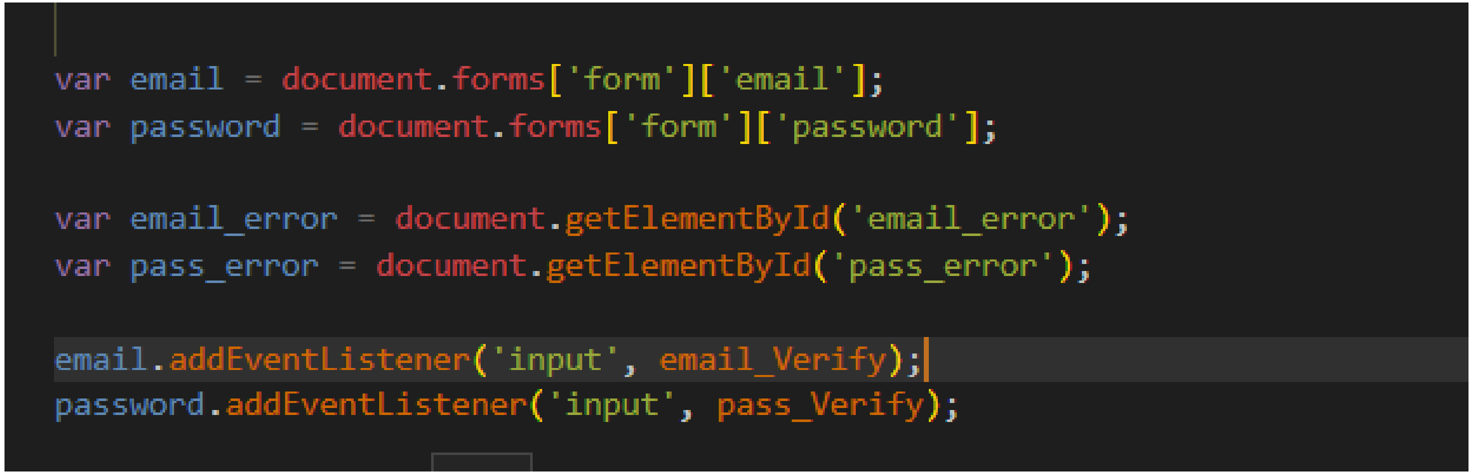


```
var email = document.forms['form']['email'];
var password = document.forms['form']['password'];

var email_error = document.getElementById('email_error');
var pass_error = document.getElementById('pass_error');
```

Document method document.getElementById()

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

loginform.js



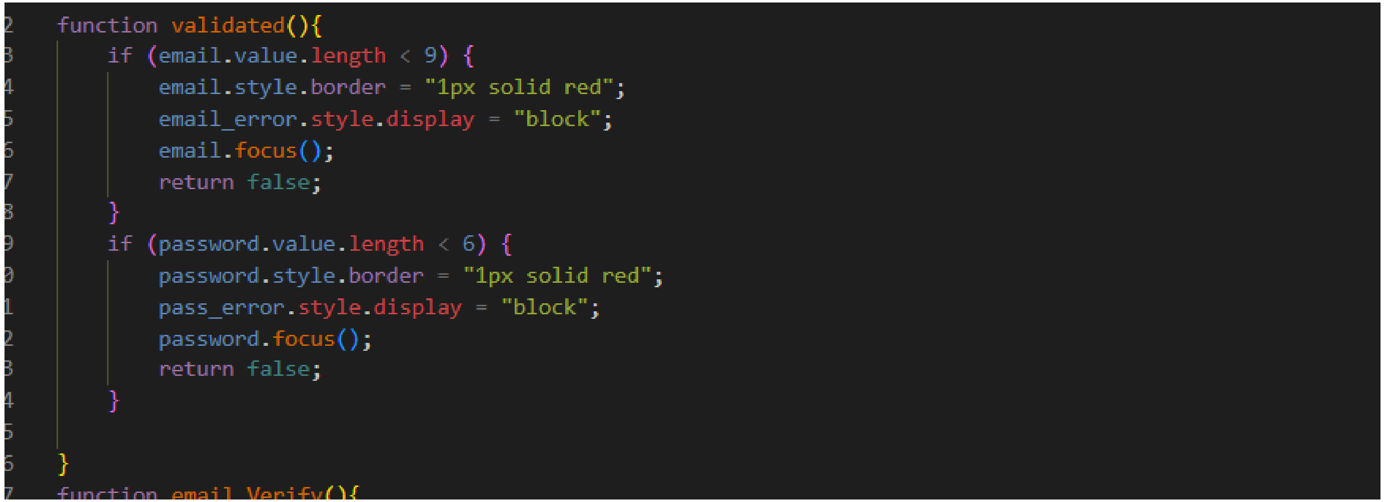
```
var email = document.forms['form']['email'];
var password = document.forms['form']['password'];

var email_error = document.getElementById('email_error');
var pass_error = document.getElementById('pass_error');

email.addEventListener('input', email_Verify);
password.addEventListener('input', pass_Verify);
```

The **addEventListener()** and **onclick** both listen for an event. Both can execute a callback function when a button is clicked.

The addEventListener() method can have multiple event handlers applied to the same element. It doesn't overwrite other event handlers.

https://www.geeksforgeeks.org/difference-between-addeventlistener-and-onclick-in-javascript/

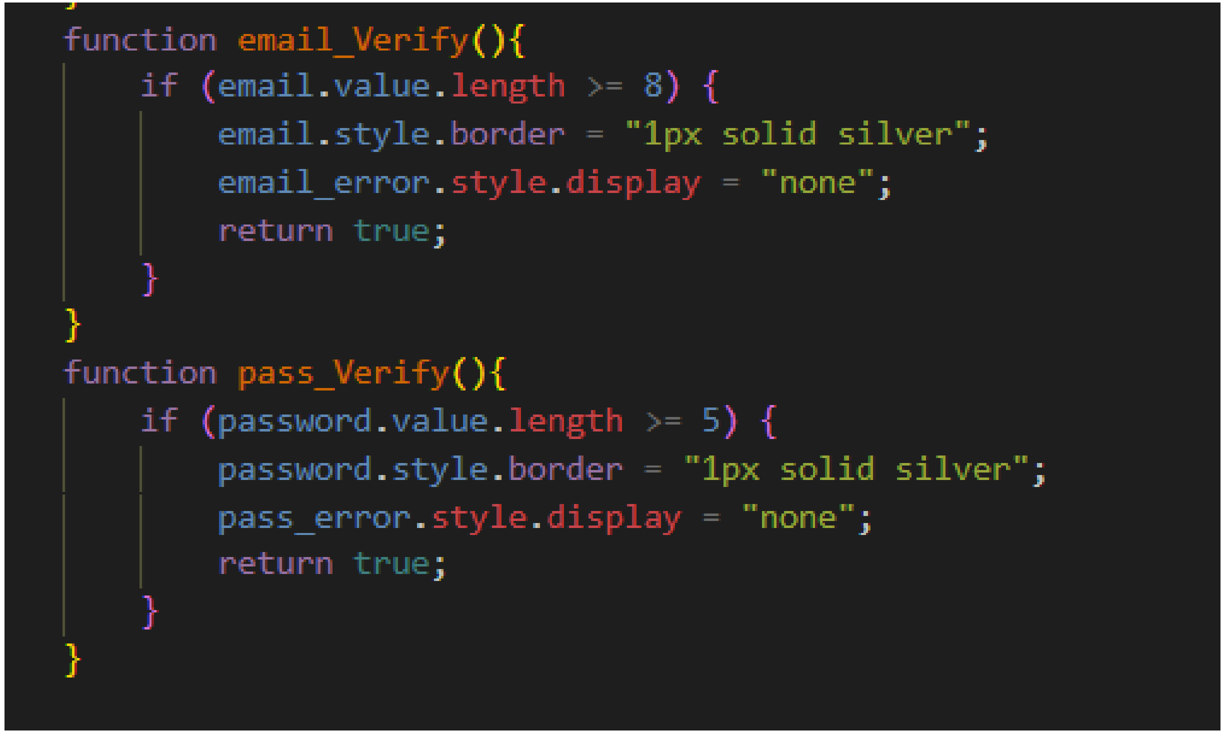
loginform.js



```
function validated(){
    if (email.value.length < 9) {
        email.style.border = "1px solid red";
        email_error.style.display = "block";
        email.focus();
        return false;
    }
    if (password.value.length < 6) {
        password.style.border = "1px solid red";
        pass_error.style.display = "block";
        password.focus();
        return false;
    }

}
function email_Verify(){
```

Create function for email and password errors.

loginform.js



```
function email_Verify(){
    if (email.value.length >= 8) {
        email.style.border = "1px solid silver";
        email_error.style.display = "none";
        return true;
    }
}
function pass_Verify(){
    if (password.value.length >= 5) {
        password.style.border = "1px solid silver";
        pass_error.style.display = "none";
        return true;
    }
}
```

Create function for email and password to validate.

Thank you!

KodeGo

LEARNING OUTCOMES

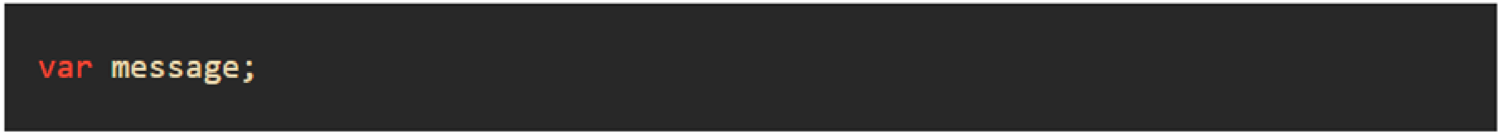
1. **Understand the function of Javascript**
2. **Apply** three ways to add JavaScript to a web page
3. Positioning of Script inside HTML Document
4. Javascript Syntax

JavaScript Data Types (RECAP)

Javascript Variables

A variable is a label that references a value like a number or string. Before using a variable, you need to declare it.

To declare a variable, you use the var keyword followed by the variable name as follows:



```
var message;
```

A variable name can be any valid identifier. By default, the message variable has a special value undefined if you have not assigned a value to it.

A variable name can be any valid identifier. By default, the message variable has a special value undefined if you have not assigned a value to it.

Variable names follow these rules:

- Variable names are case-sensitive. This means that the message and Message are different variables.
- Variable names can only contain letters, numbers, underscores, or dollar signs and cannot contain spaces. Also, variable names must begin with a letter, an underscore (_) or a dollar sign ($).
- Variable names cannot use the reserved words.

JavaScript Let

Variables defined with `let` cannot be Redeclared.

Variables defined with `let` must be Declared before use.

Variables defined with `let` have Block Scope.{}

JavaScript Const

Variables defined with `const` cannot be Redeclared.

Variables defined with `const` cannot be Reassigned.

Variables defined with `const` have Block Scope.{}

Data Types in JavaScript

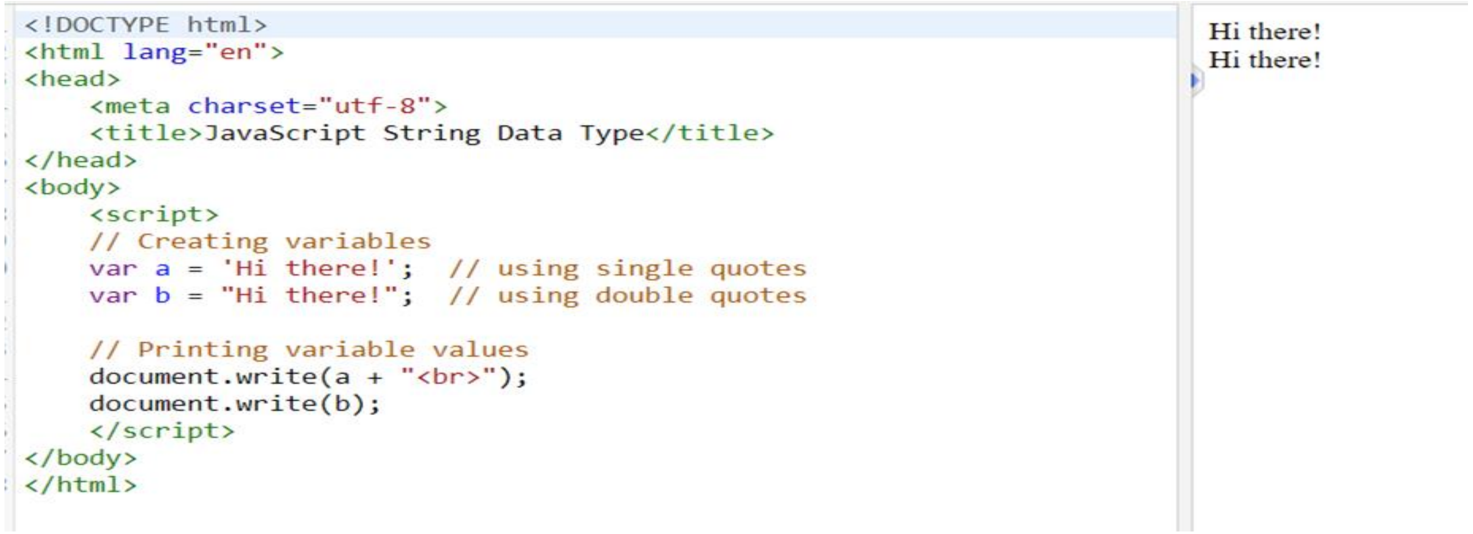
There are six basic data types in JavaScript which can be divided into three main categories: primitive (or *primary*), *composite* (or *reference*), and *special* data types.

String, Number, and Boolean are **primitive data types**.

Object, Array, and Function (which are all types of objects) are **composite data types.**

Whereas Undefined and Null are **special data types.**

The *string* data type is used to represent textual data (i.e. sequences of characters). Strings are created using single or double quotes surrounding one or more characters, as shown below:

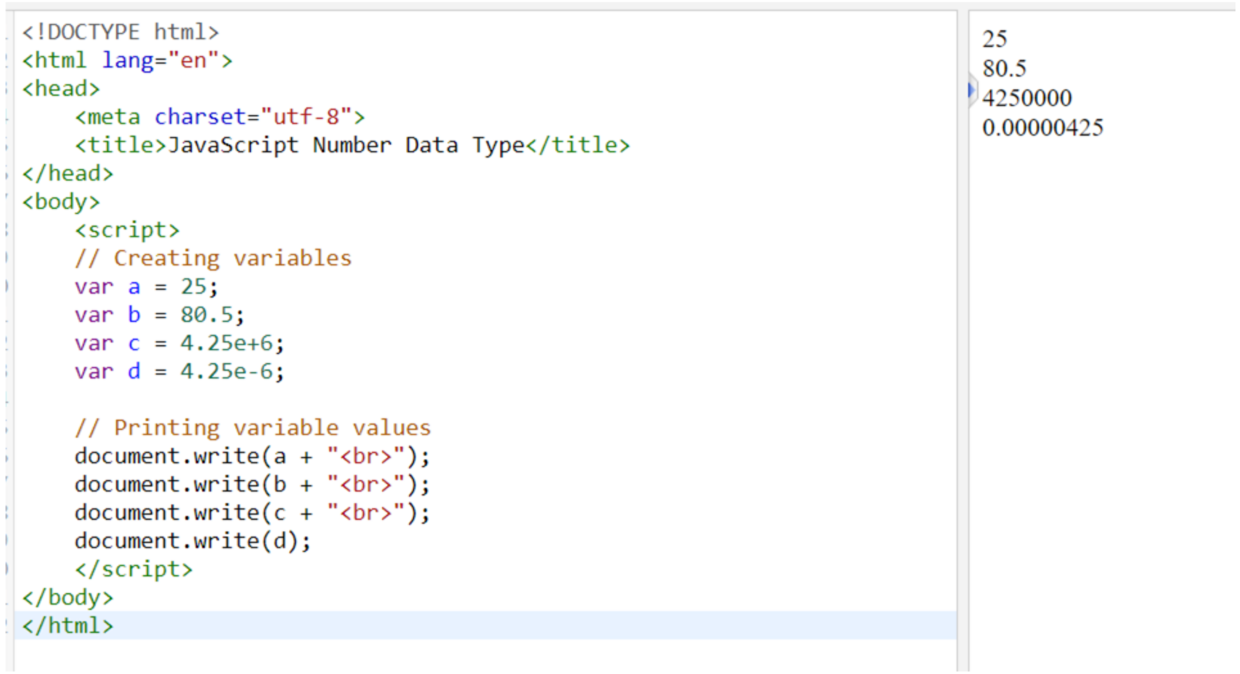


```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JavaScript String Data Type</title>
</head>
<body>
    <script>
    // Creating variables
    var a = 'Hi there!';  // using single quotes
    var b = "Hi there!";  // using double quotes

    // Printing variable values
    document.write(a + "<br>");
    document.write(b);
    </script>
</body>
</html>
```

```
Hi there!
Hi there!
```

1. Try to change the variable names and make it 5 variables. Add CSS.

The *number* data type is used to represent positive or negative numbers with or without decimal place, or numbers written using exponential notation e.g. 1.5e-4 (equivalent to 1.5×10^{-4}).

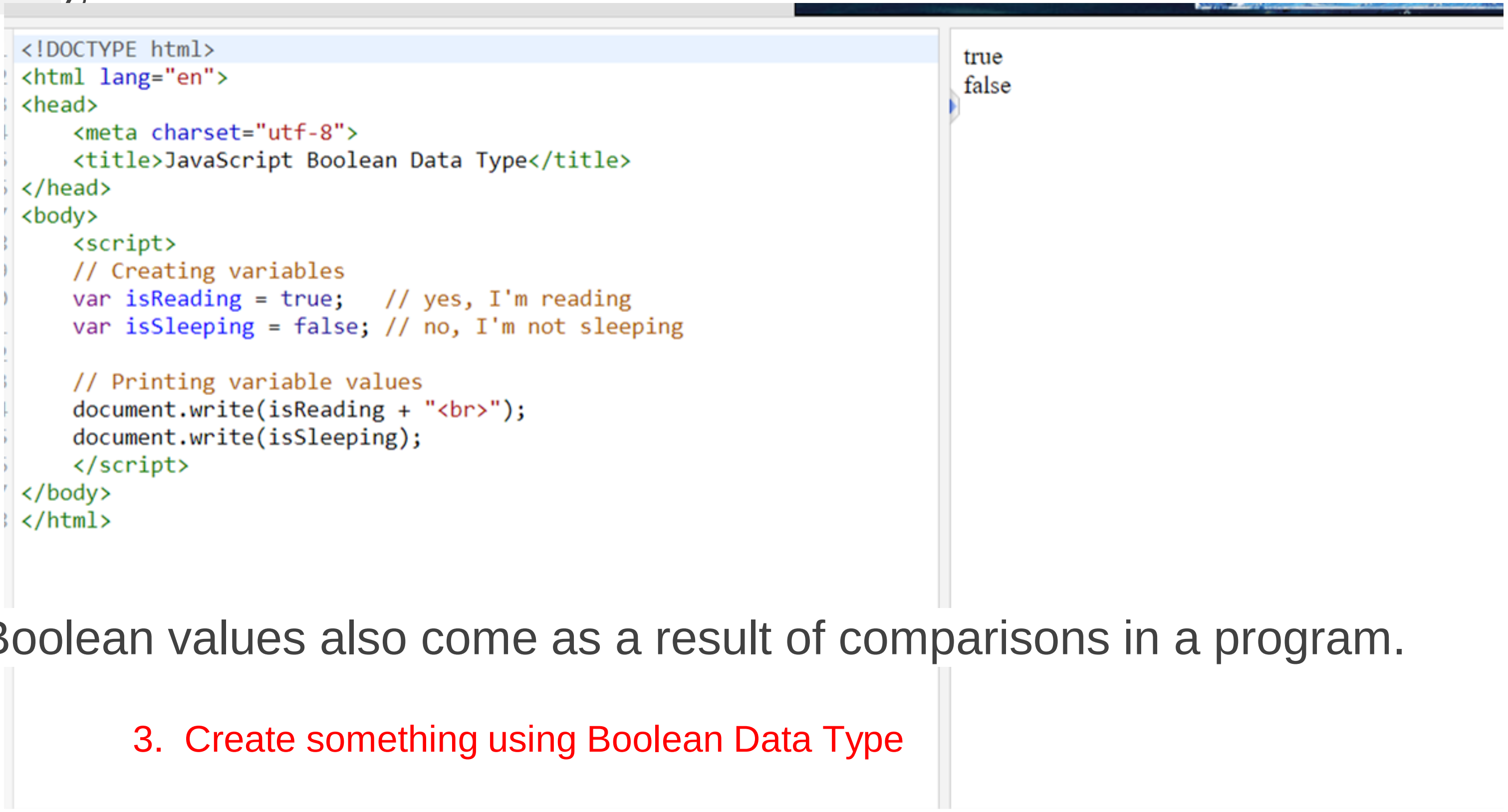


```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JavaScript Number Data Type</title>
</head>
<body>
    <script>
    // Creating variables
    var a = 25;
    var b = 80.5;
    var c = 4.25e+6;
    var d = 4.25e-6;

    // Printing variable values
    document.write(a + "<br>");
    document.write(b + "<br>");
    document.write(c + "<br>");
    document.write(d);
    </script>
</body>
</html>
```

```
25
80.5
4250000
0.00000425
```

2. Display a name of a person with : Age: , Height: , Weight: , Siblings:

The Boolean data type can hold only two values: `true` or `false`. It is typically used to store values like yes (`true`) or no (`false`), on (`true`) or off (`false`), etc. as demonstrated below:



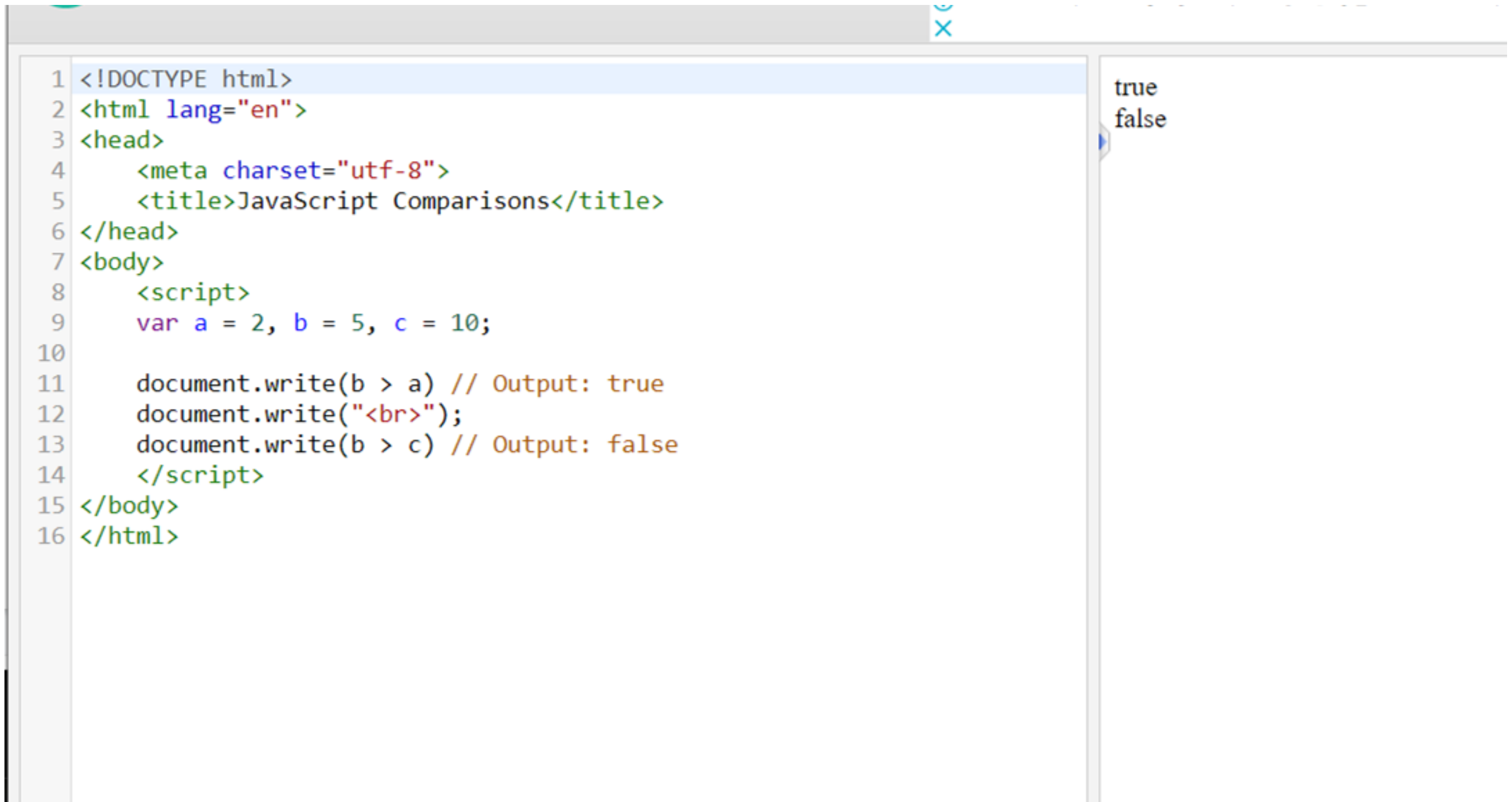
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JavaScript Boolean Data Type</title>
</head>
<body>
    <script>
    // Creating variables
    var isReading = true;    // yes, I'm reading
    var isSleeping = false; // no, I'm not sleeping

    // Printing variable values
    document.write(isReading + "<br>");
    document.write(isSleeping);
    </script>
</body>
</html>
```

```
true
false
```

Boolean values also come as a result of comparisons in a program.

3. Create something using Boolean Data Type

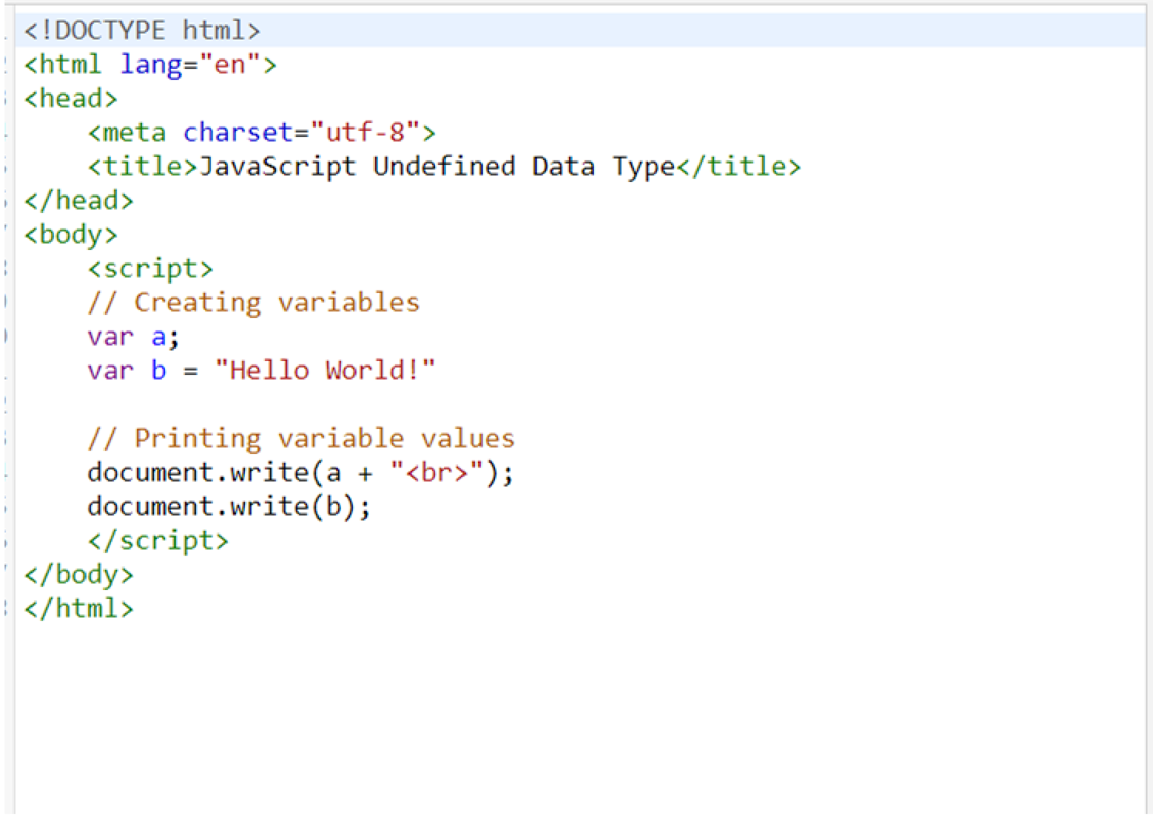
Boolean values also come as a result of comparisons in a program. The following example compares two variables and shows the result in an alert dialog box:



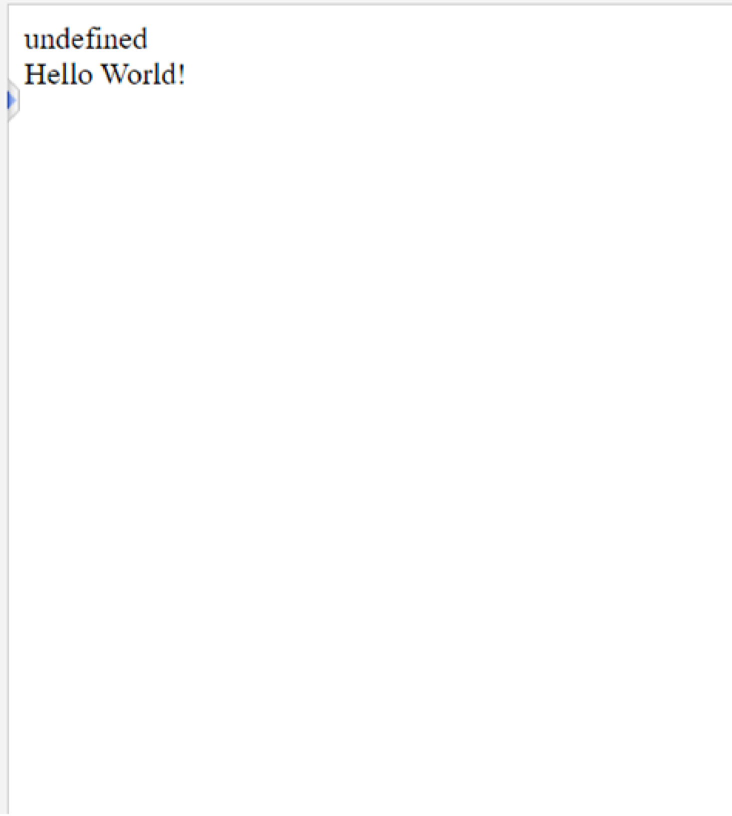
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JavaScript Comparisons</title>
</head>
<body>
    <script>
    var a = 2, b = 5, c = 10;

    document.write(b > a) // Output: true
    document.write("<br>");
    document.write(b > c) // Output: false
    </script>
</body>
</html>
```

```
true
false
```

The **undefined data** type can only have one value-the special value `undefined`. If a variable has been declared, but has not been assigned a value, has the value `undefined`.



```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JavaScript Undefined Data Type</title>
</head>
<body>
    <script>
    // Creating variables
    var a;
    var b = "Hello World!"

    // Printing variable values
    document.write(a + "<br>");
    document.write(b);
    </script>
</body>
</html>
```

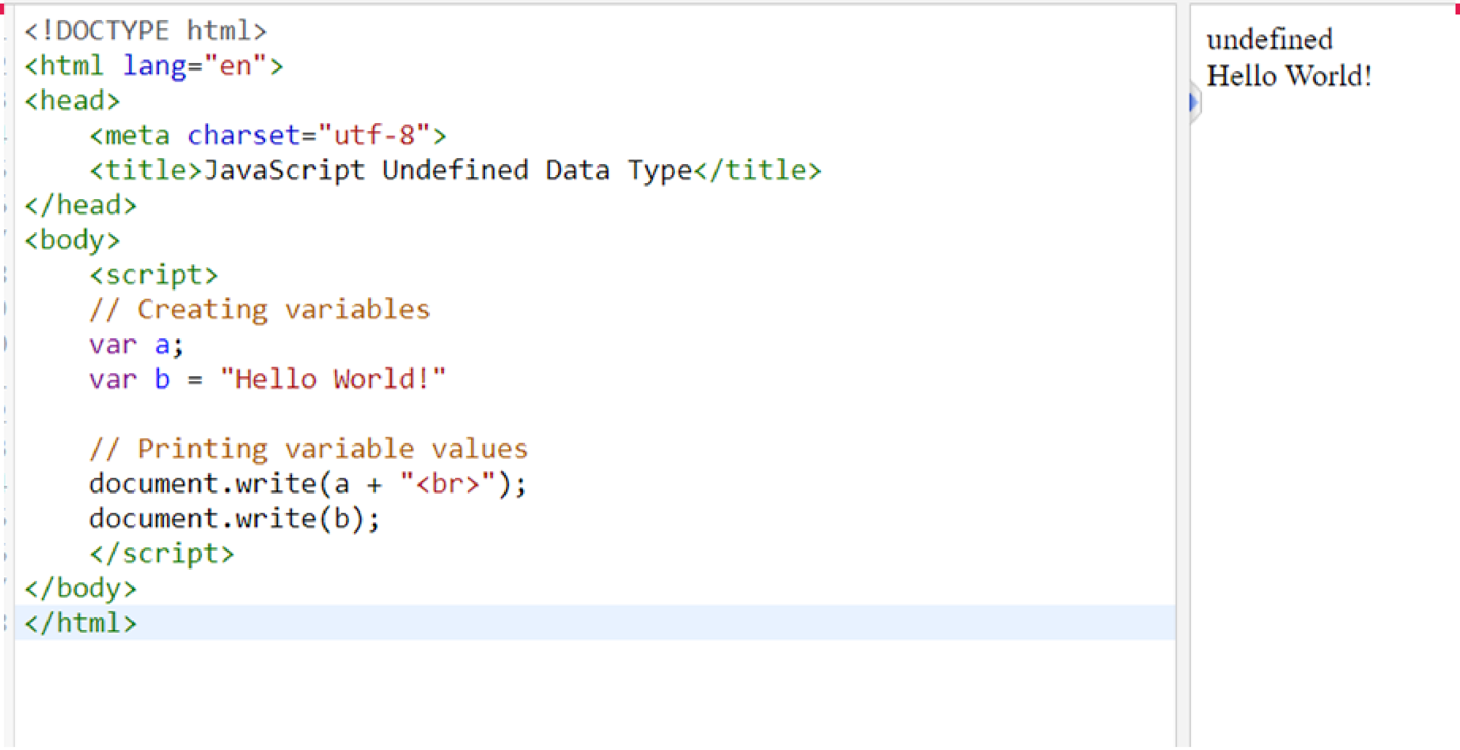


```
undefined
Hello World!
```

The Undefined Data Type

The undefined data type can only have one value-the special value `undefined`. If a variable has been declared, but has not been assigned a value, has the value `undefined`.

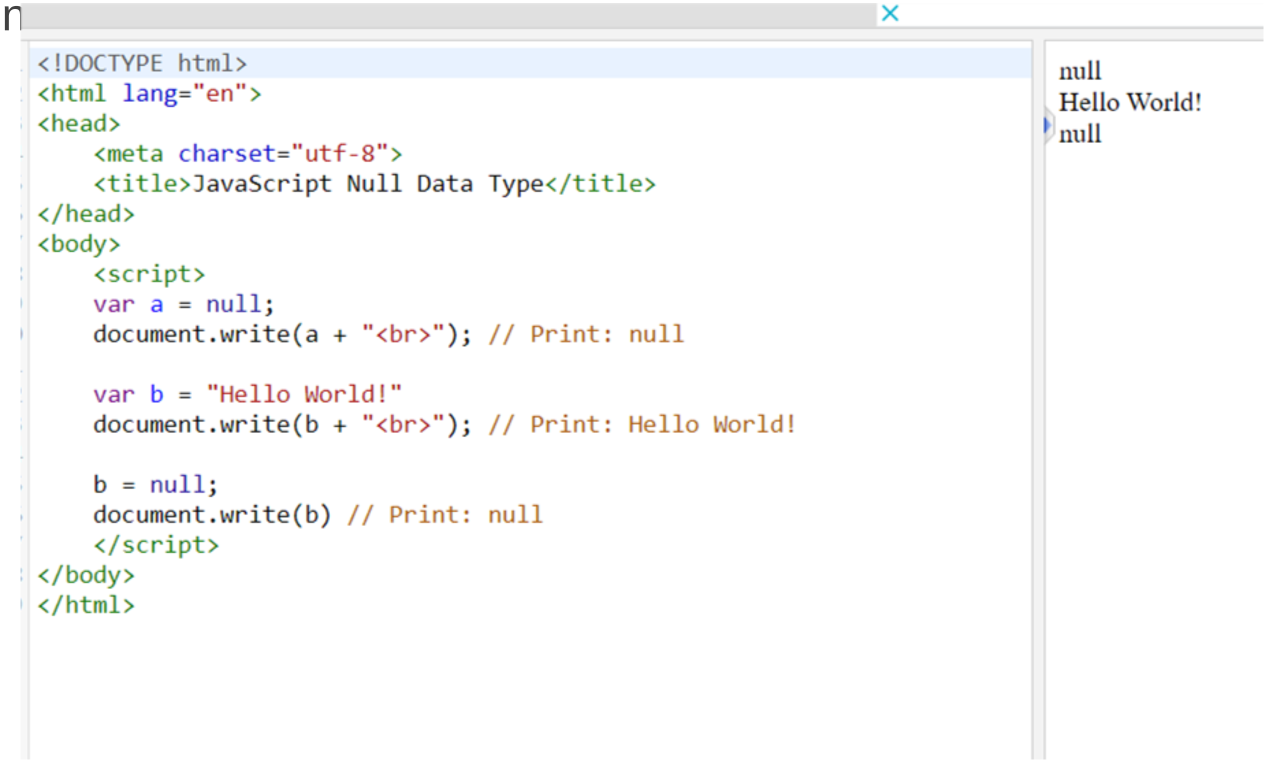
The Undefined Data Type



```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JavaScript Undefined Data Type</title>
</head>
<body>
    <script>
    // Creating variables
    var a;
    var b = "Hello World!"

    // Printing variable values
    document.write(a + "<br>");
    document.write(b);
    </script>
</body>
</html>
```

```
undefined
Hello World!
```

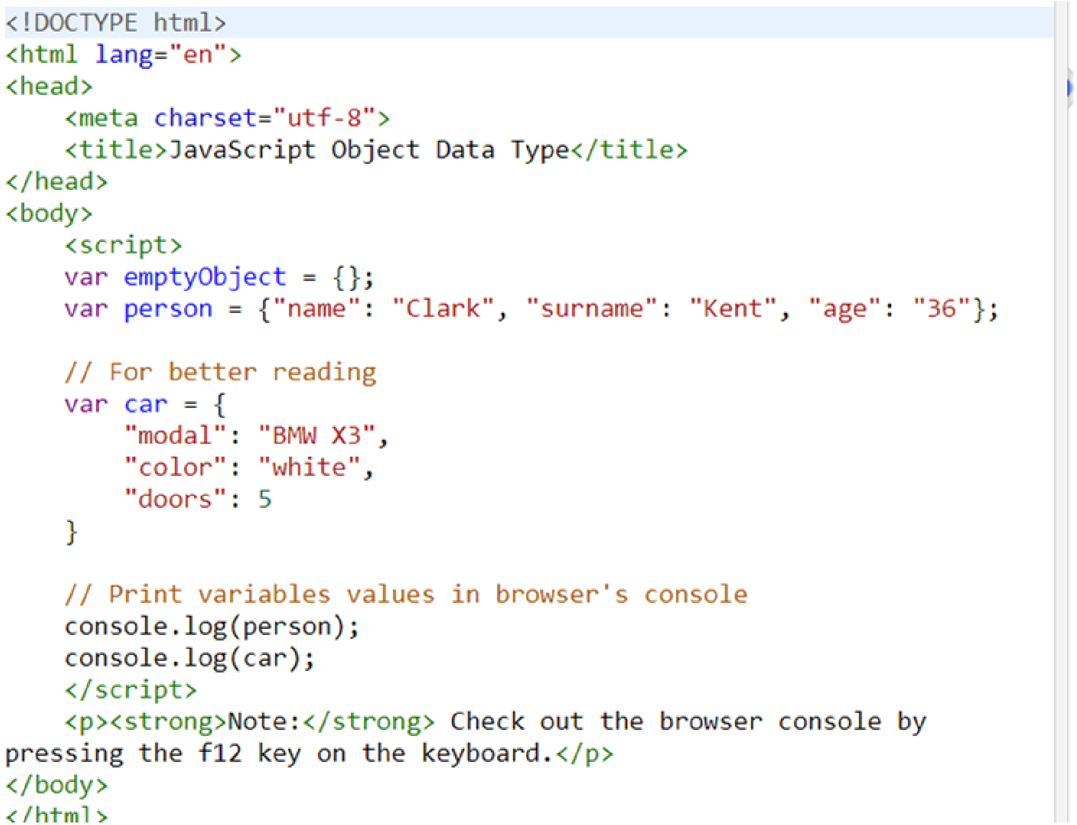
This is another special data type that can have only one value-the `null` value. A `null` value means that there is no value. It is not equivalent to an empty string (`""`) or 0, it is simply nothin



```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JavaScript Null Data Type</title>
</head>
<body>
    <script>
    var a = null;
    document.write(a + "<br>"); // Print: null

    var b = "Hello World!"
    document.write(b + "<br>"); // Print: Hello World!

    b = null;
    document.write(b) // Print: null
    </script>
</body>
</html>
```

```
null
Hello World!
null
```

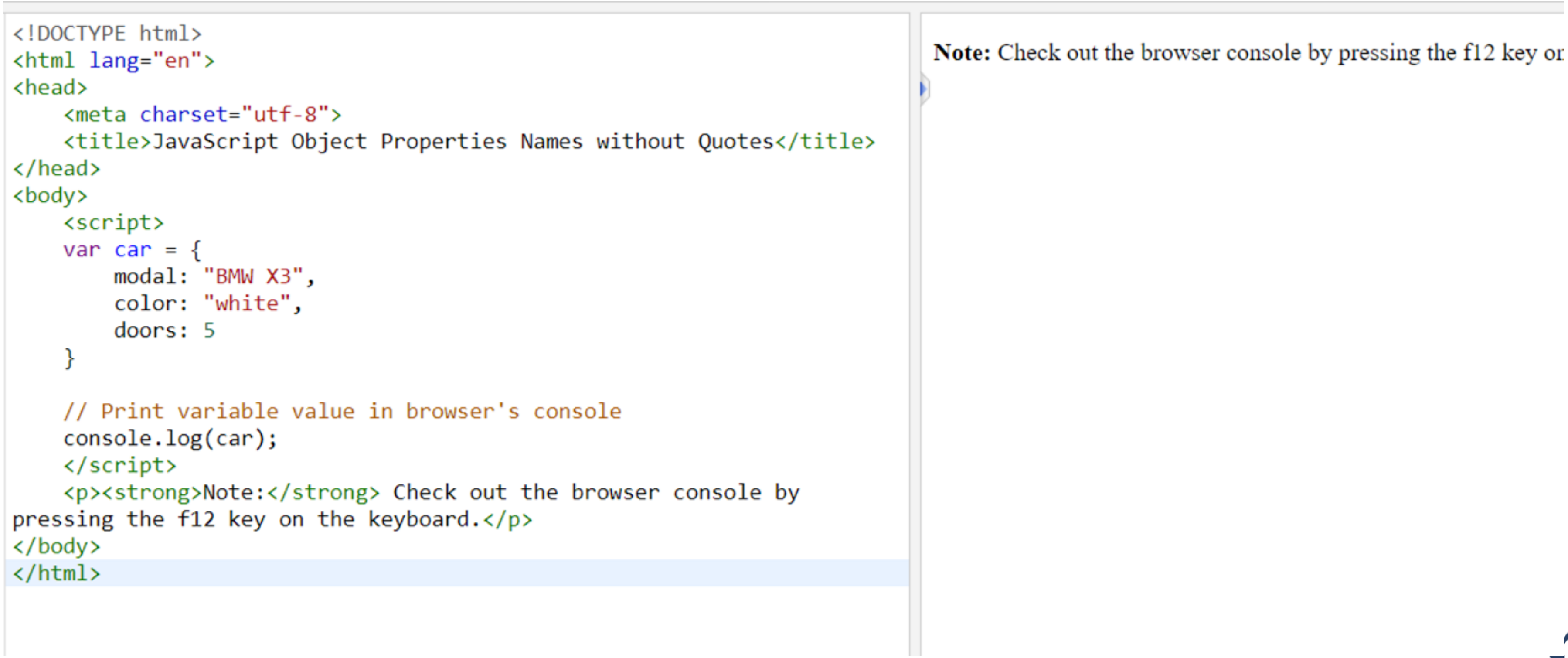
An object contains **properties, defined as a key-value pair.** A property key (name) is always a string, but the value can be any data type, like strings, numbers, booleans, or complex data types like arrays, function and other objects.



```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JavaScript Object Data Type</title>
</head>
<body>
    <script>
    var emptyObject = {};
    var person = {"name": "Clark", "surname": "Kent", "age": "36"};

    // For better reading
    var car = {
        "modal": "BMW X3",
        "color": "white",
        "doors": 5
    }

    // Print variables values in browser's console
    console.log(person);
    console.log(car);
    </script>
    <p><strong>Note:</strong> Check out the browser console by
pressing the f12 key on the keyboard.</p>
</body>
</html>
```

4. Create an object with different variables

You can omit the quotes around property name if the name is a valid JavaScript name. That means quotes are required around `"first-name"` but are optional around `firstname`. So the car object in the above example can also be written as:



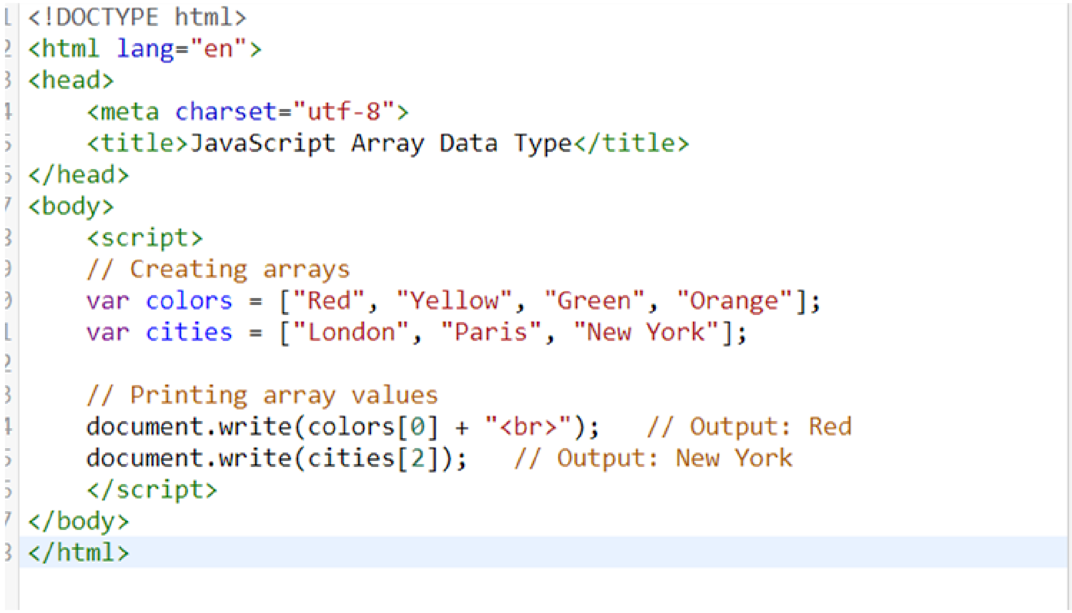
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JavaScript Object Properties Names without Quotes</title>
</head>
<body>
    <script>
    var car = {
        modal: "BMW X3",
        color: "white",
        doors: 5
    }

    // Print variable value in browser's console
    console.log(car);
    </script>
    <p><strong>Note:</strong> Check out the browser console by
pressing the f12 key on the keyboard.</p>
</body>
</html>
```

Note: Check out the browser console by pressing the f12 key or

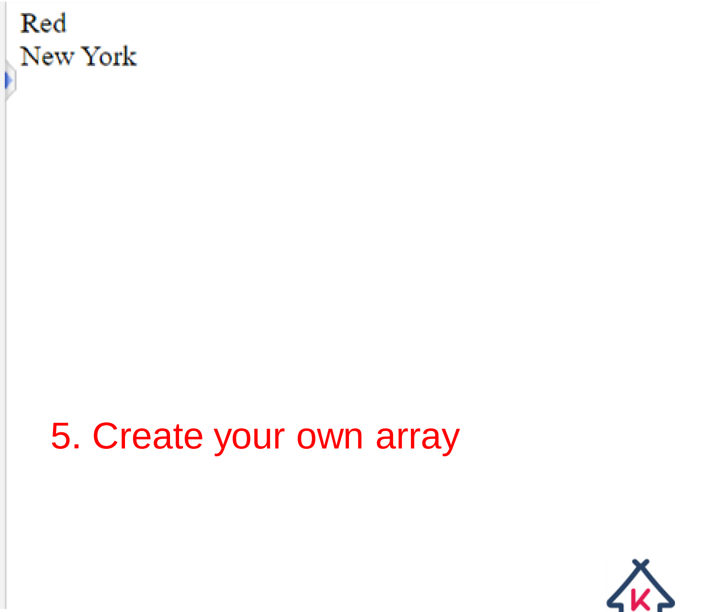
An array is a type of object used for storing multiple values in single variable.

Each value (also called an element) in an array has a numeric position, known as its index, and it may contain data of any data type-numbers, strings, booleans, functions, objects, and even other arrays.

The array index starts from 0, so that the first array element is `arr[0]` not `arr[1]`.



```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JavaScript Array Data Type</title>
</head>
<body>
    <script>
    // Creating arrays
    var colors = ["Red", "Yellow", "Green", "Orange"];
    var cities = ["London", "Paris", "New York"];

    // Printing array values
    document.write(colors[0] + "<br>");   // Output: Red
    document.write(cities[2]);   // Output: New York
    </script>
</body>
</html>
```

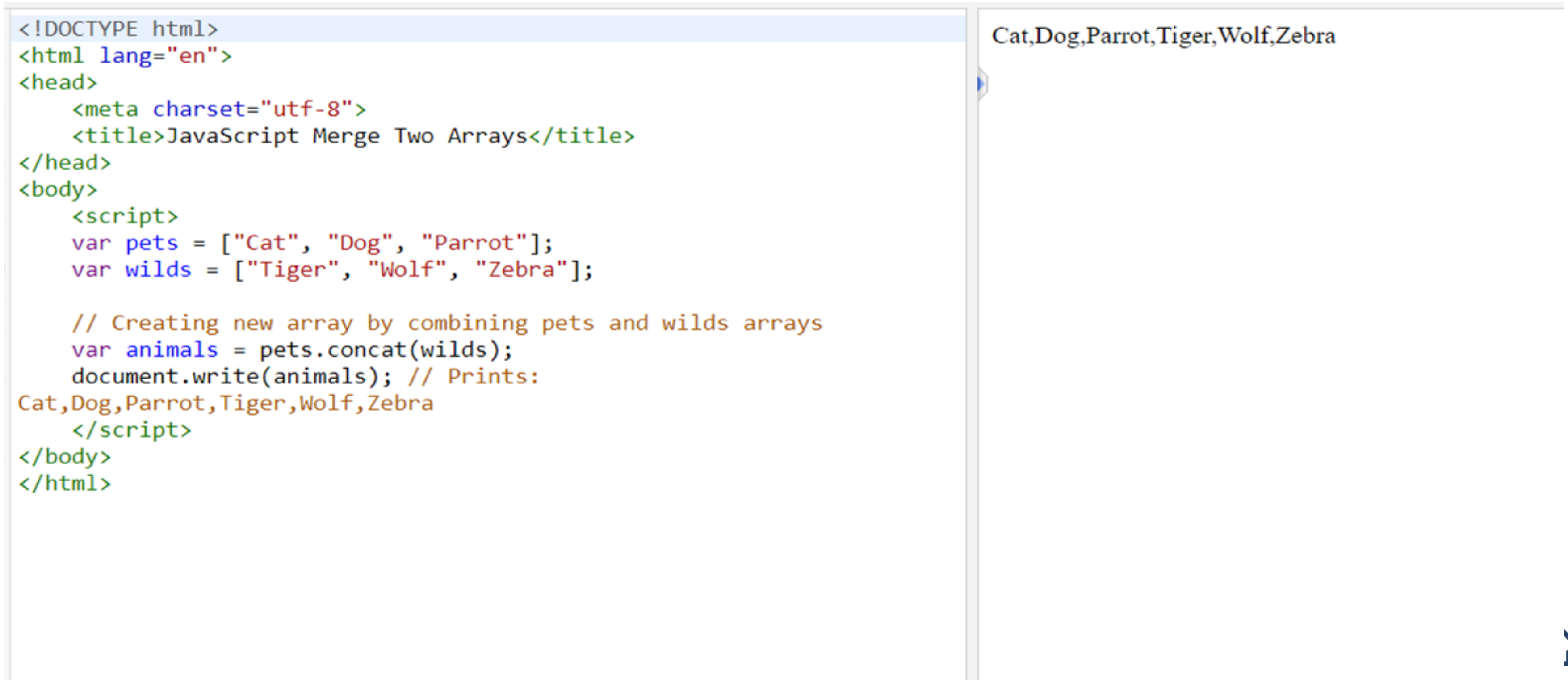


```
Red
New York
```

5. Create your own array

Merging 2 or More Arrays

The `concat()` method can be used to merge or combine two or more arrays. This method does not change the existing arrays, instead it returns a new array. For example:

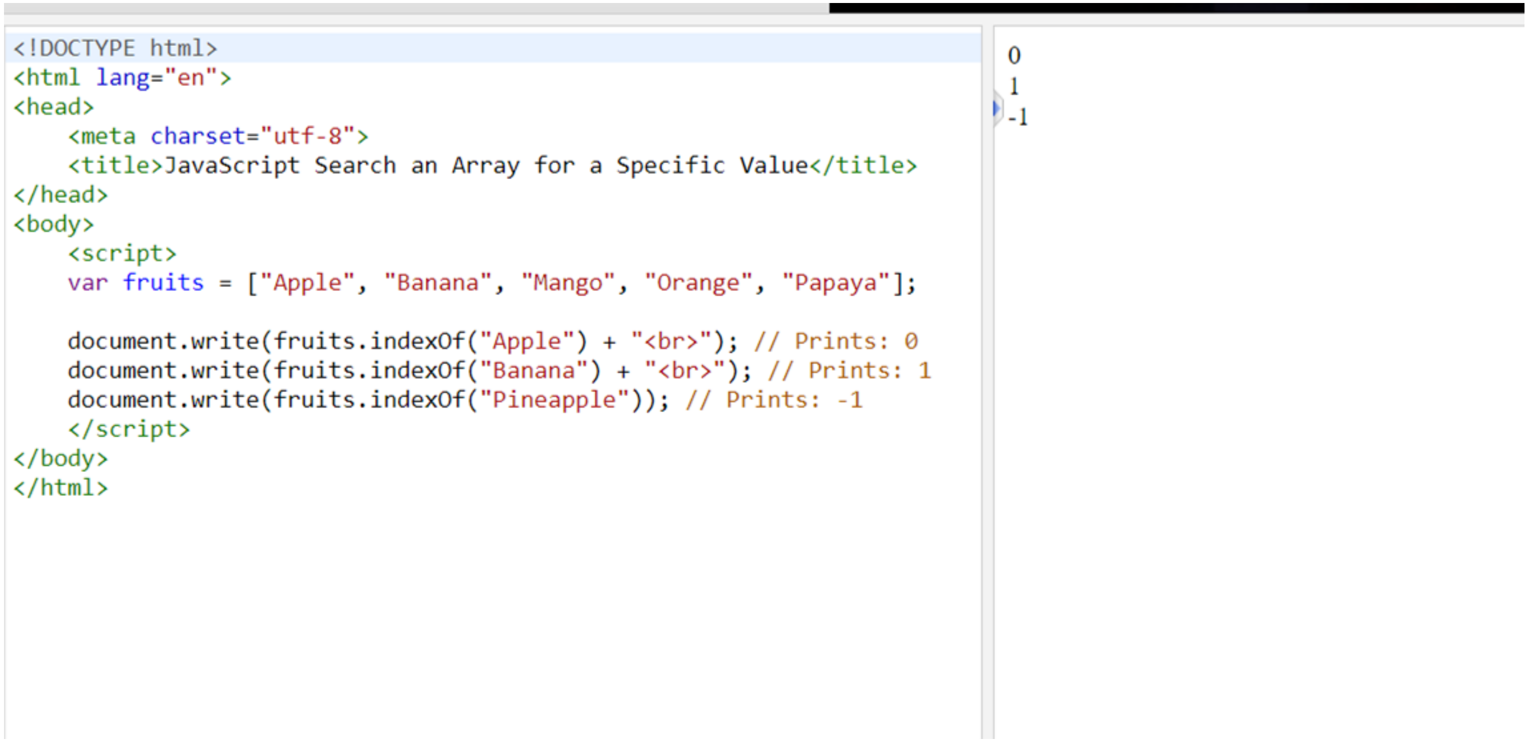


```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JavaScript Merge Two Arrays</title>
</head>
<body>
    <script>
    var pets = ["Cat", "Dog", "Parrot"];
    var wilds = ["Tiger", "Wolf", "Zebra"];

    // Creating new array by combining pets and wilds arrays
    var animals = pets.concat(wilds);
    document.write(animals); // Prints:
Cat,Dog,Parrot,Tiger,Wolf,Zebra
    </script>
</body>
</html>
```

```
Cat,Dog,Parrot,Tiger,Wolf,Zebra
```

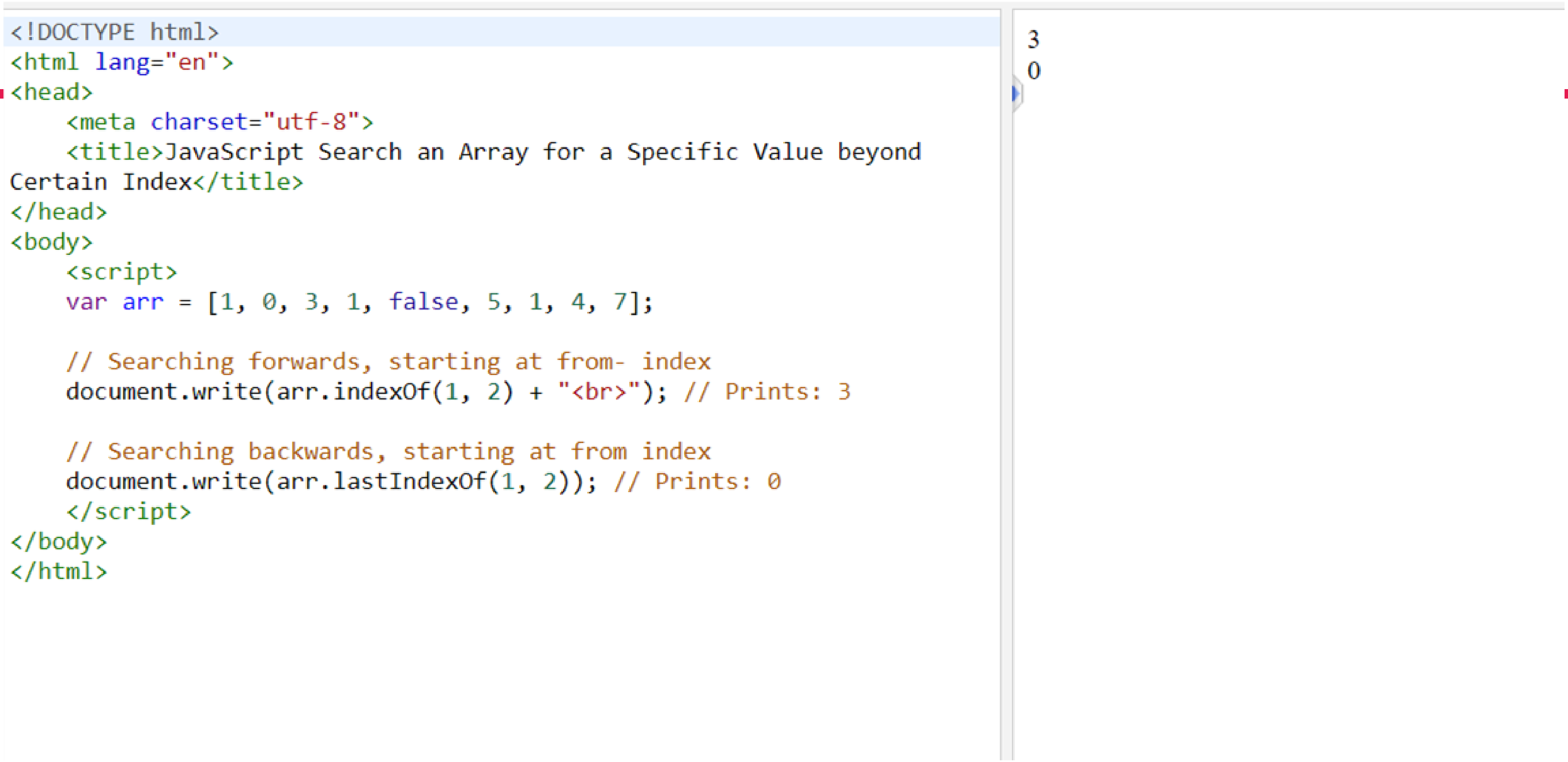
If you want to search an array for a specific value, you can simply use the `indexOf()` and `lastIndexOf()`.

If the value is found, both methods return an index representing the array element.



```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JavaScript Search an Array for a Specific Value</title>
</head>
<body>
    <script>
    var fruits = ["Apple", "Banana", "Mango", "Orange", "Papaya"];

    document.write(fruits.indexOf("Apple") + "<br>"); // Prints: 0
    document.write(fruits.indexOf("Banana") + "<br>"); // Prints: 1
    document.write(fruits.indexOf("Pineapple")); // Prints: -1
    </script>
</body>
</html>
```

```
0
1
-1
```



```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JavaScript Search an Array for a Specific Value beyond
Certain Index</title>
</head>
<body>
    <script>
    var arr = [1, 0, 3, 1, false, 5, 1, 4, 7];

    // Searching forwards, starting at from- index
    document.write(arr.indexOf(1, 2) + "<br>"); // Prints: 3

    // Searching backwards, starting at from index
    document.write(arr.lastIndexOf(1, 2)); // Prints: 0
    </script>
</body>
</html>
```

```
3
0
```

The Function Data Type

A function is a block of code which will be executed only if it is called.

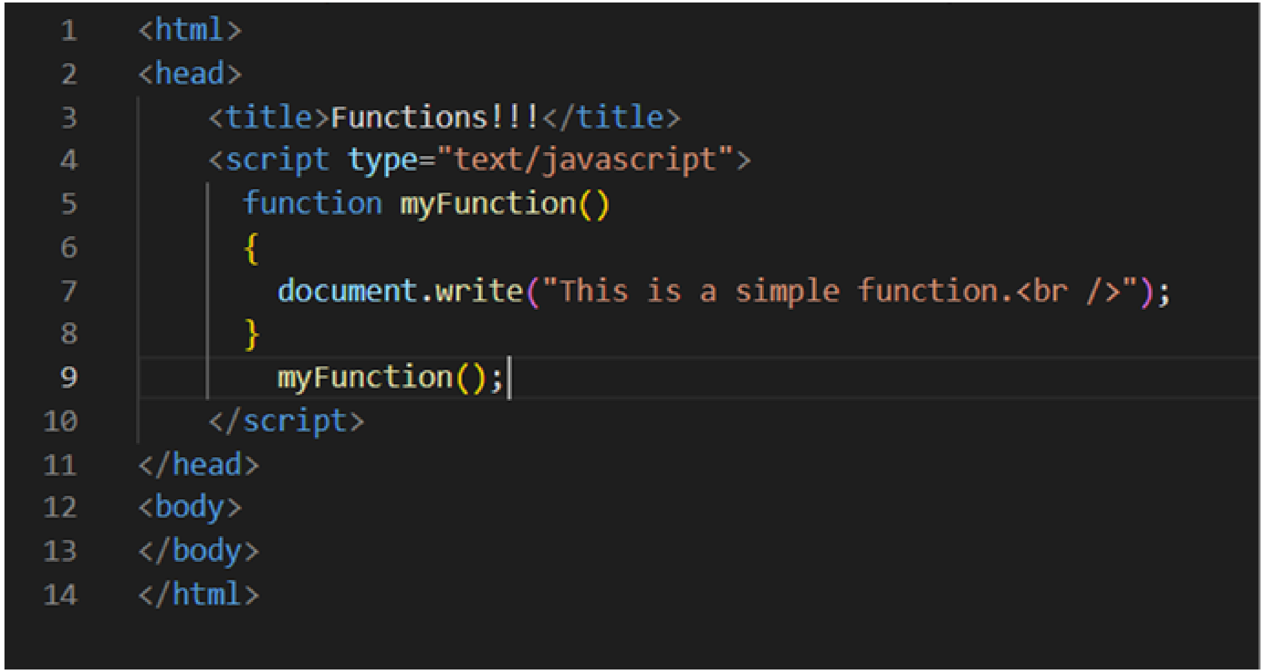
How to Create a Function in JavaScript

1. Use the keyword **function** followed by the name of the function.
2. After the function name, open and close parentheses.
3. After parenthesis, open and close curly braces.
4. Within curly braces, write your lines of code.

Syntax:

```
function functionname()

{

  lines of code to be executed
```

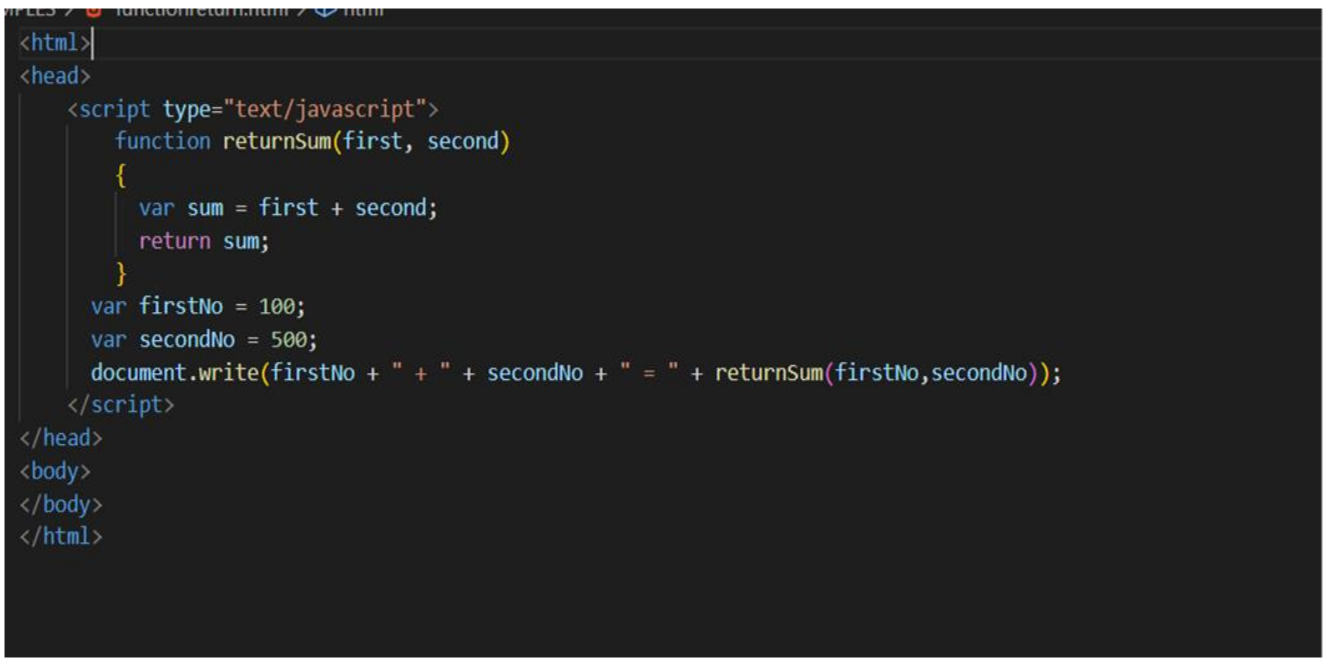
Simple Function



```
<html>
<head>
    <title>Functions!!!</title>
    <script type="text/javascript">
      function myFunction()
      {
        document.write("This is a simple function.<br />");
      }
        myFunction();
    </script>
</head>
<body>
</body>
</html>
```

```
Syntax:

function
functionname(arg1,
arg2)

{

  lines of code to
be executed


   }
```

JavaScript Return Value (JS functions that return values)



```
<html>
<head>
    <script type="text/javascript">
        function returnSum(first, second)
        {
          var sum = first + second;
          return sum;
        }
      var firstNo = 100;
      var secondNo = 500;
      document.write(firstNo + " + " + secondNo + " = " + returnSum(firstNo,secondNo));
    </script>
</head>
<body>
</body>
</html>
```

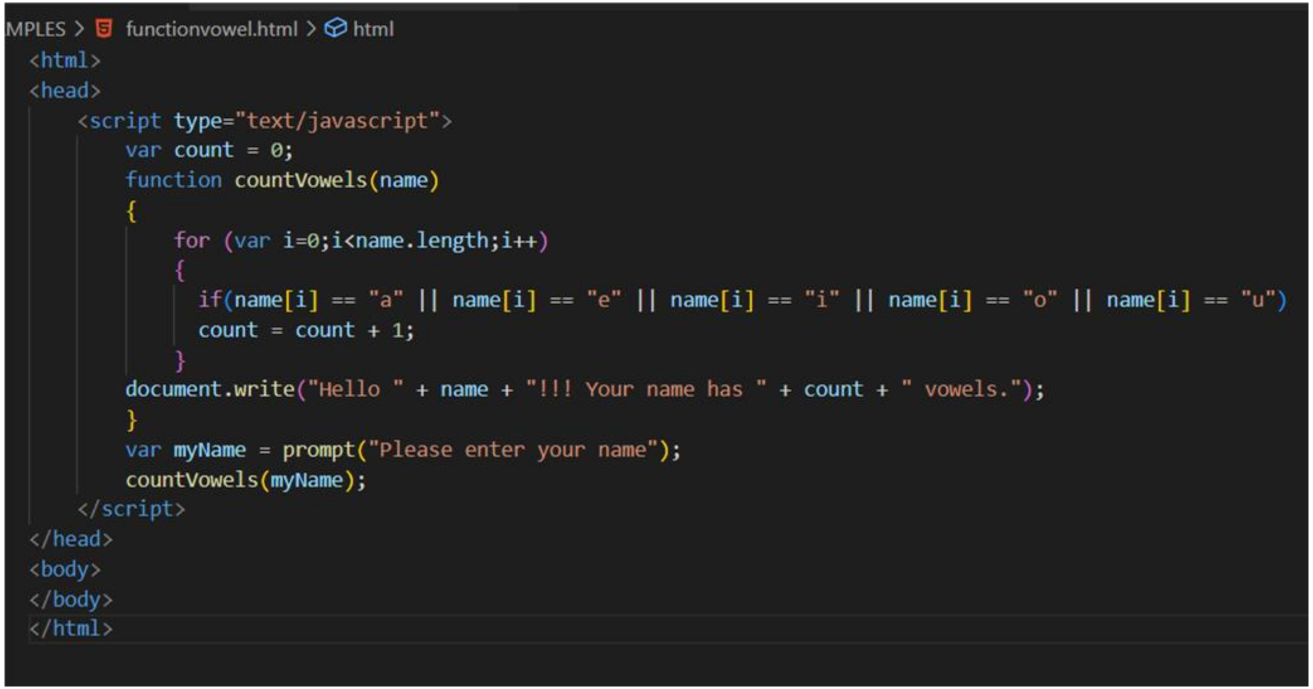
Syntax:

```
function
functionname(arg1, arg2)

{

  lines of code to be
executed

  return value;


   }
```

Function with Arguments

Syntax:

```
function functionname(arg1, arg2)

{

lines of code to be executed

}
```



MPLES > functionvowel.html > html

```
<html>
<head>
    <script type="text/javascript">
        var count = 0;
        function countVowels(name)
        {
            for (var i=0;i<name.length;i++)
            {
              if(name[i] == "a" || name[i] == "e" || name[i] == "i" || name[i] == "o" || name[i] == "u")
              count = count + 1;
            }
        document.write("Hello " + name + "!!! Your name has " + count + " vowels.");
        }
        var myName = prompt("Please enter your name");
        countVowels(myName);
    </script>
</head>
<body>
</body>
</html>
```

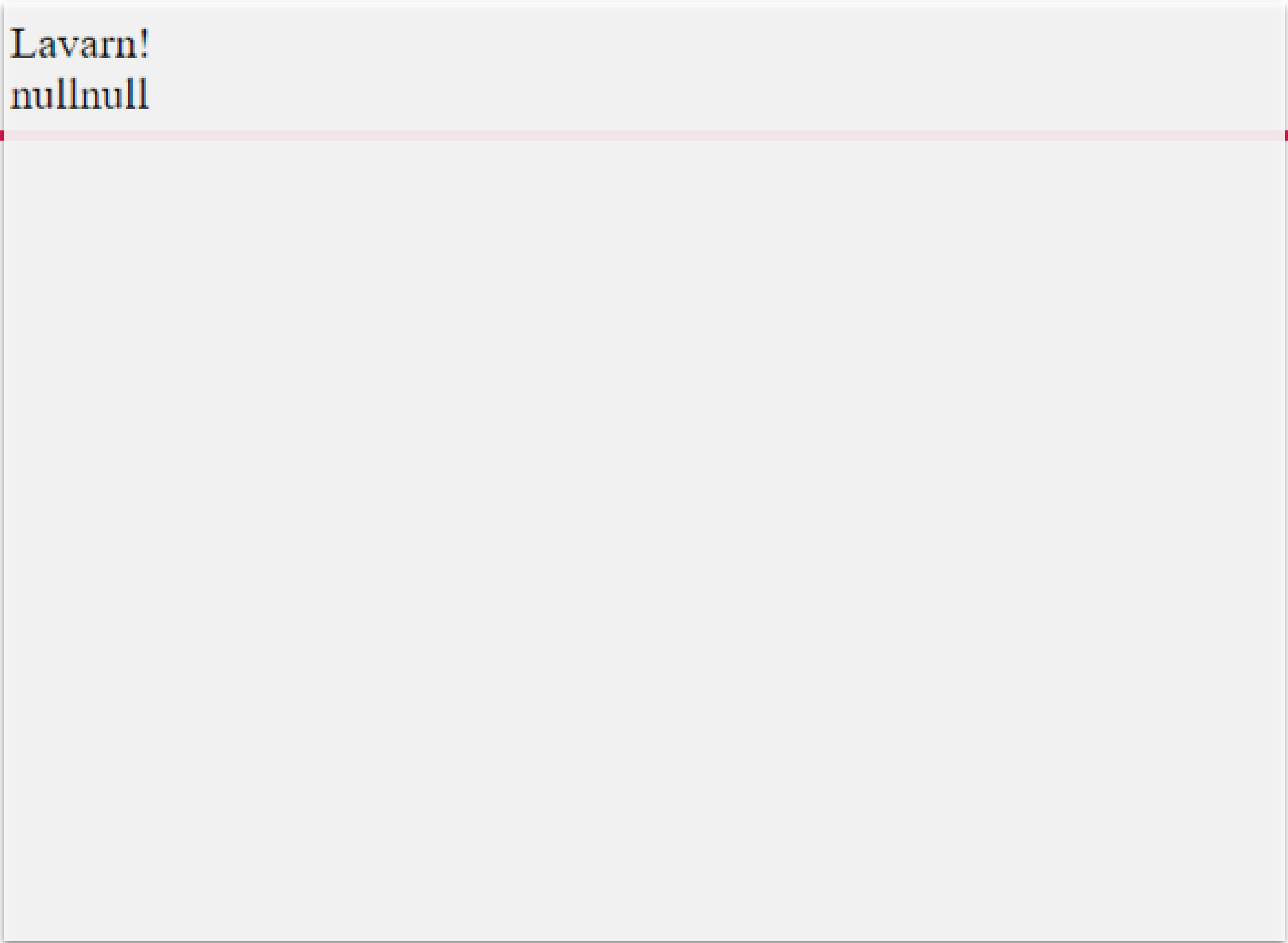
i - increment

The value **i++** is the value of i before the increment. **(pre-increment)**

The value of **++i** is the value of i after the increment. **(post-increment)**

Create source code for the following
Live Server Preview

5.



```
Lavarn!
nullnull
```