JavaScript Introduction

- Introduced in the year 1995.
- Light-weight Object Oriented Scripting Language.
- Enables dynamic interactivity on web pages.

Features-

- Almost every popular web browser supports JavaScript as they provide built-in execution environments.
- The syntax and structure of the JavaScript is almost similar to the C programming language. Thus, it is a structured programming language.
- JavaScript is a weakly typed language, means there is no need to declare the type of a variable before using it.
- It is an interpreted language and case sensitive scripting language.
- JavaScript is supportable by almost every operating system available nowadays, including, Windows, macOS, Linux, etc.

JavaScript Applications

Apart from its features we can build anything on our HTML page with JavaScript that will be supported by every browser.

Some of the JavaScript applications are:

- Client-side validation,
- Dynamic drop-down menus,
- Displaying date and time,
- Displaying pop-up windows and dialog boxes (like an alert dialog box, confirm dialog box and prompt dialog box),
- Displaying clocks, Calendar, etc.

First JavaScript Program

- The **<script>** tag in HTML is used to define the client-side script. The <script> tag contains the scripting statements, or it points to an external script file.

- The value of type attribute is **text/javascript** is the content type that provides information to the browser about the data. It is the default type value.

- The **document.write()** function is used to display dynamic content through JavaScript. Here*, **document*** is an inbuilt object and ***write()*** is the method of the document object.

Example-

```
<script type="text/javascript">
   document.write("Welcome to the world of JavaScript");
</script>
```

document.write()

- **document.write()** in JavaScript is a function that is used to display some text in the browser window.

- **We can concatenate strings by using "+" operator.**

```
document.write("My roll number is " +10);
```

- **We can do arithmetic operations and display them in our result.**

```
document.write("Sum of 5+5= " + (5+5));
```

- **Tags are easily interpreted as HTML elements.**

```
document.write("<h2>This is a HTML content</h2>");
```

Where To Put JavaScript Code

We can put our JavaScript code in various places like:

- Between the body tag of html

- Between the head tag of html

- In .js file (external javaScript)

- In inspection mode (ctrl+shift+i)

Comments in JavaScript-

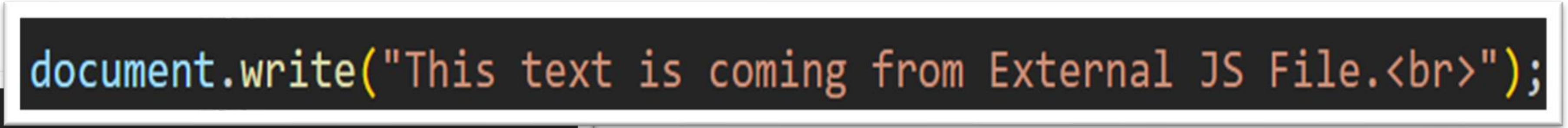
To make any line of JavaScript as a comment, use double slash (**//**). If you want to make multiple lines as a comment, use **/*** and ***/**.

External JavaScript

- We can create external JavaScript file and embed it in many html pages.
- It provides **code re-usability** because single JavaScript file can be used in several html pages.
- OR, a single JavaScript file can be used several times in a same HTML file.
- An external JavaScript file must be saved by **.js extension**.

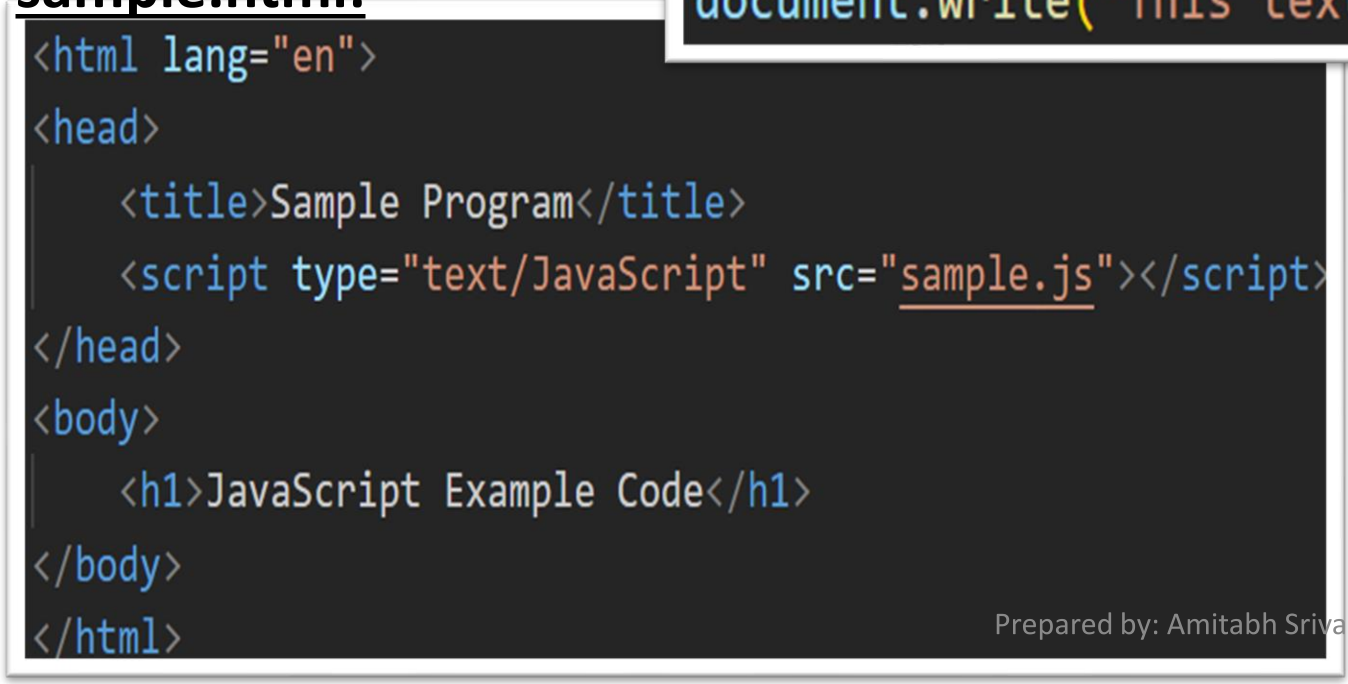
Example-

sample.js:



```
document.write("This text is coming from External JS File.<br>");
```

sample.html:

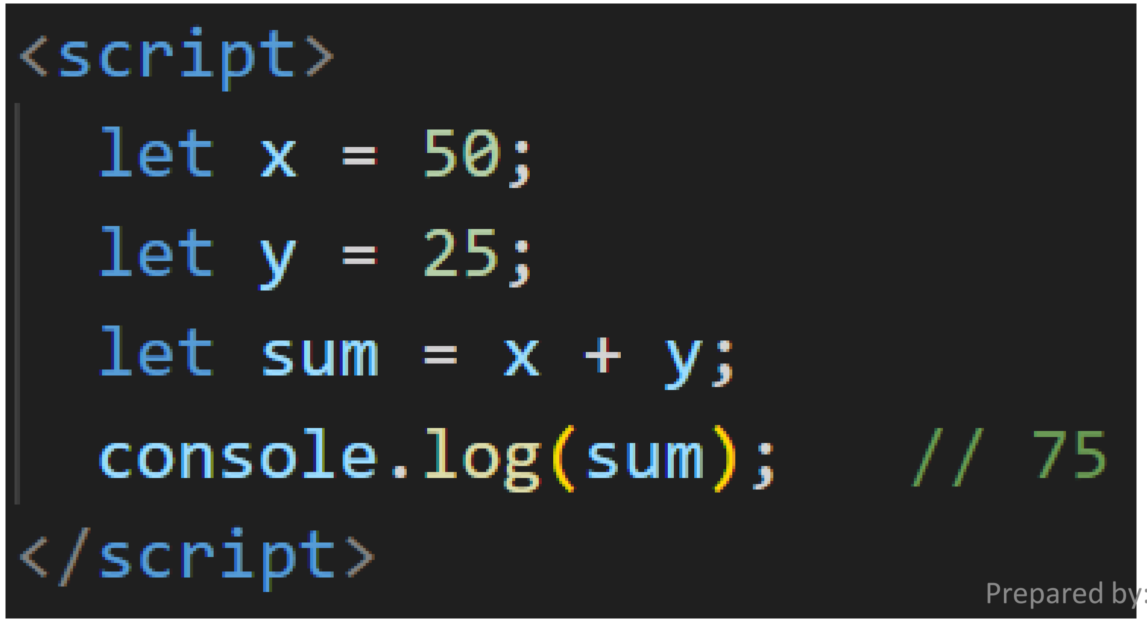


```
<html lang="en">
<head>
    <title>Sample Program</title>
    <script type="text/JavaScript" src="sample.js"></script
</head>
<body>
    <h1>JavaScript Example Code</h1>
</body>
</html>
```

console.log()

- All modern browsers have a web console for debugging.
- To open web console window press **ctrl + shift + j** in the browser.
- The **console.log()** method is used to print messages/variables to the console.

Example-

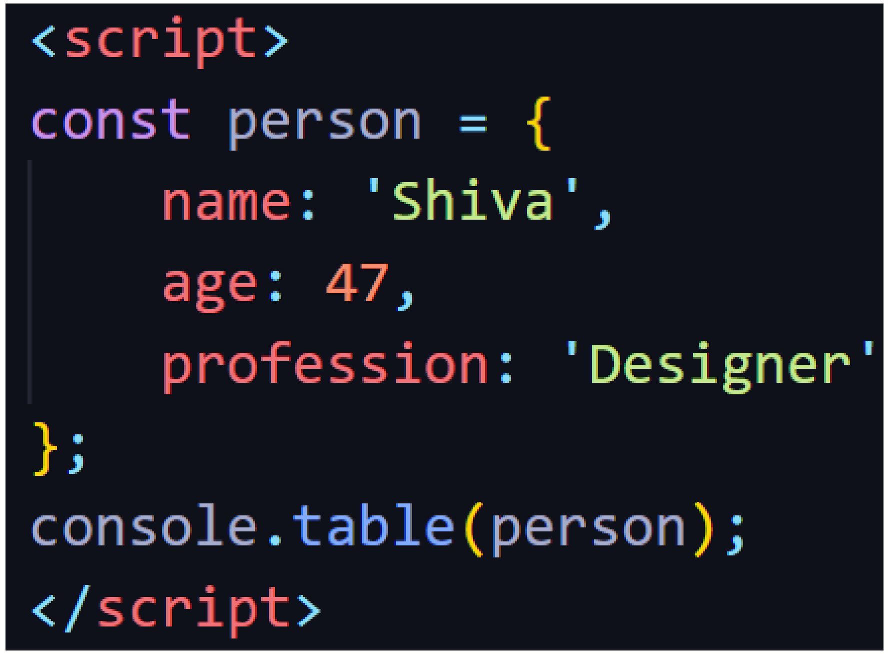


```
<script>
  let x = 50;
  let y = 25;
  let sum = x + y;
  console.log(sum);   // 75
</script>
```

console.table()

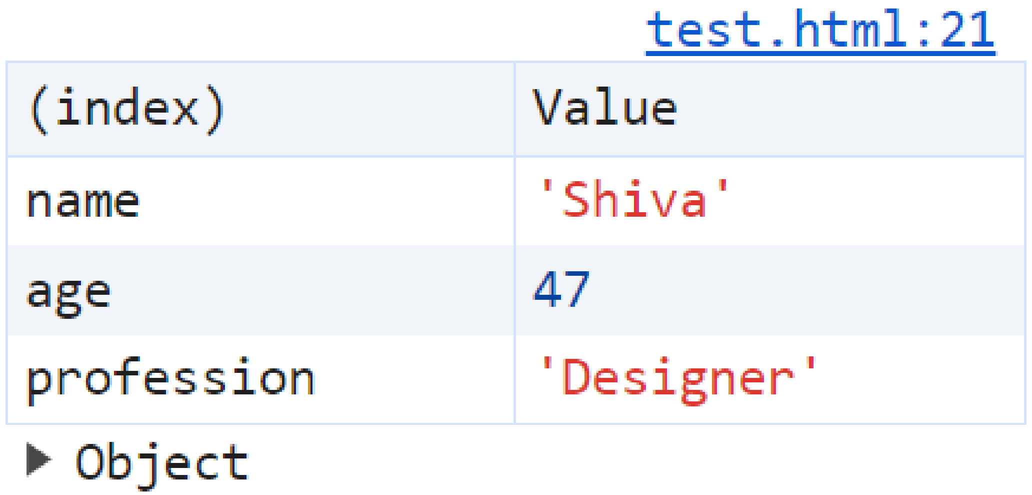
- The **console.table()** method is used to print data in table format to the console.

Example-



```
<script>
const person = {
    name: 'Shiva',
    age: 47,
    profession: 'Designer'
};
console.table(person);
</script>
```

Output-

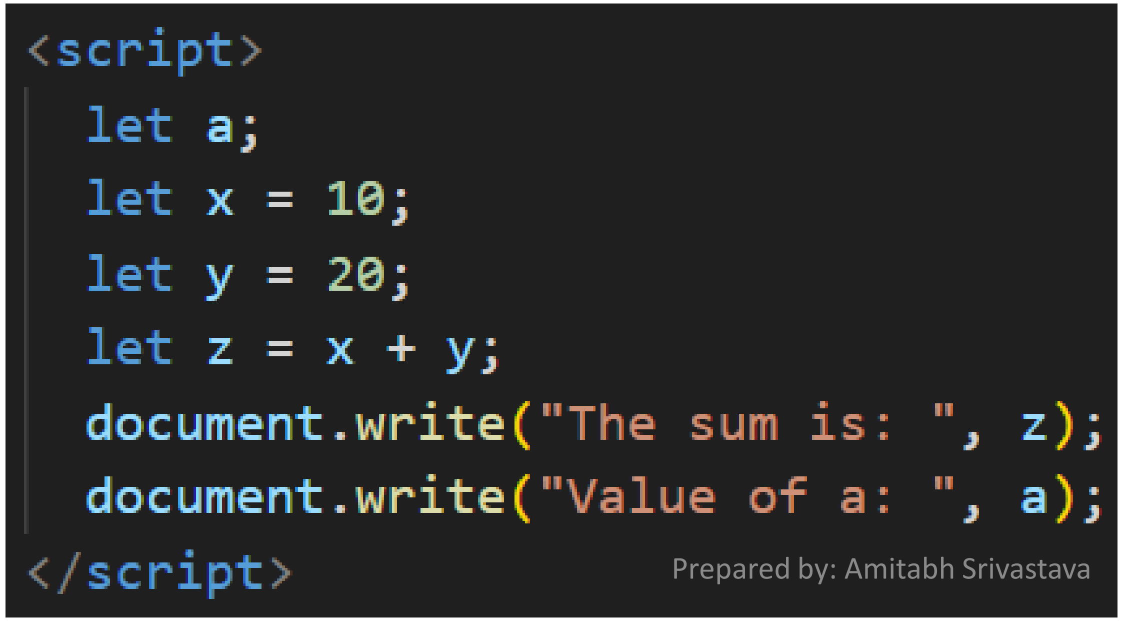


```
                                     test.html:21
(index)                 Value
name                    'Shiva'
age                     47
profession              'Designer'
▶ Object
```

JavaScript Variables

- A JavaScript variable is simply a name of storage location.
- By default, the variable has assigned a special value ′**undefined′** if no value is assigned to it.
- Variable name must start with a letter (a to z or A to Z), underscore (_), or dollar($) sign.
- After first letter we can use digits (0 to 9), for example **‘value1’**.
- JavaScript variables are case sensitive, for example ‘**a**’ and ‘**A**‘ are different variables.

Example-



```
<script>
  let a;
  let x = 10;
  let y = 20;
  let z = x + y;
  document.write("The sum is: ", z);
  document.write("Value of a: ", a);
</script>
```

Prints “undefined”.

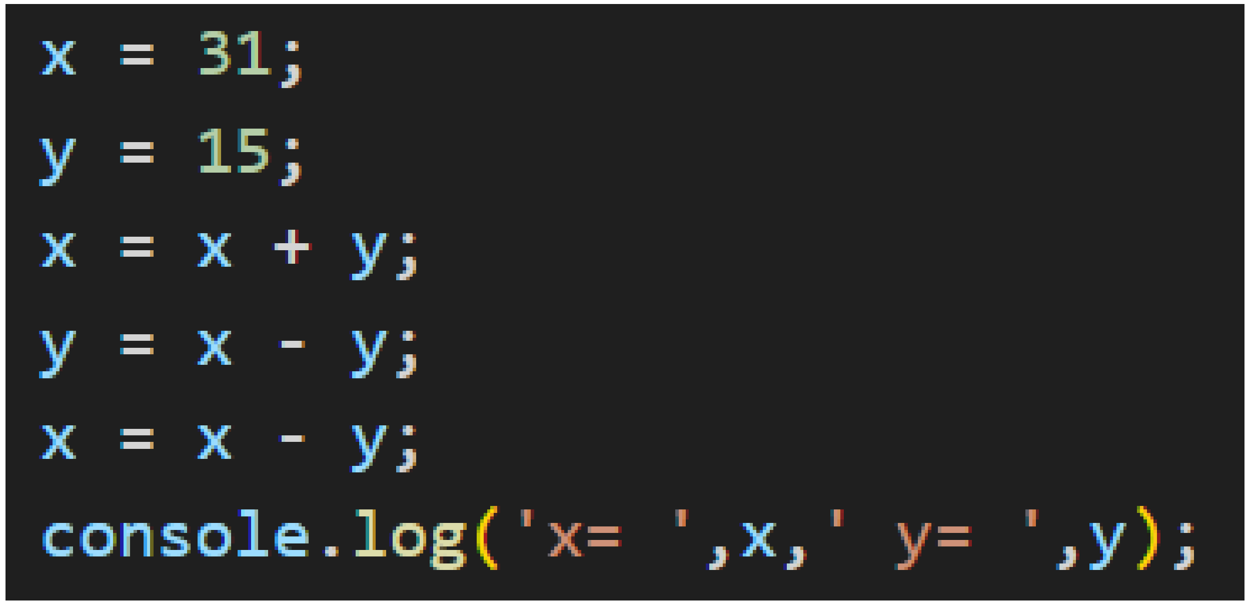
Exercise

Write a JavaScript program to swap two variables without using third variable?

Assume x = 31 and y = 15.

Display your result in the console window.

Solution-

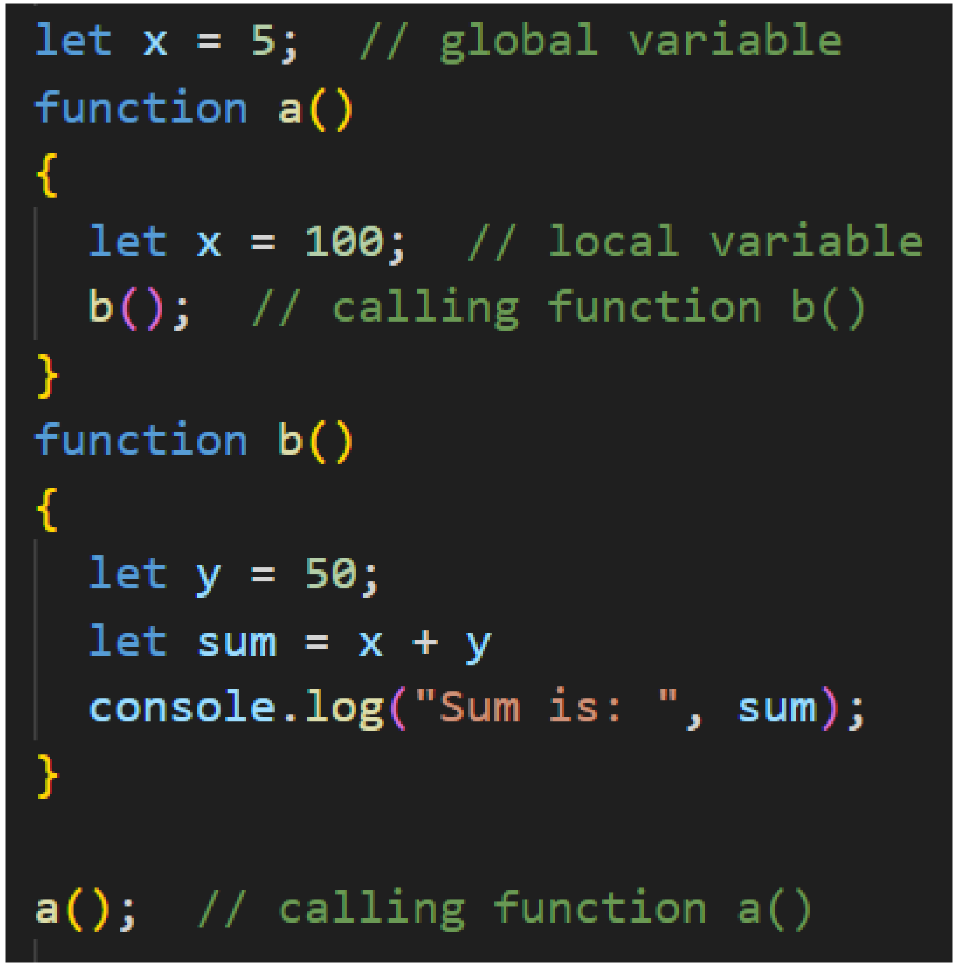


```
x = 31;
y = 15;
x = x + y;
y = x - y;
x = x - y;
console.log('x= ',x,' y= ',y);
```

Local Vs. Global Variables

There are two types of variables in JavaScript: **local variable** and **global variable**.

Example-



```
let x = 5;  // global variable
function a()
{
  let x = 100;  // local variable
  b();  // calling function b()
}
function b()
{
  let y = 50;
  let sum = x + y
  console.log("Sum is: ", sum);
}

a();  // calling function a()
```

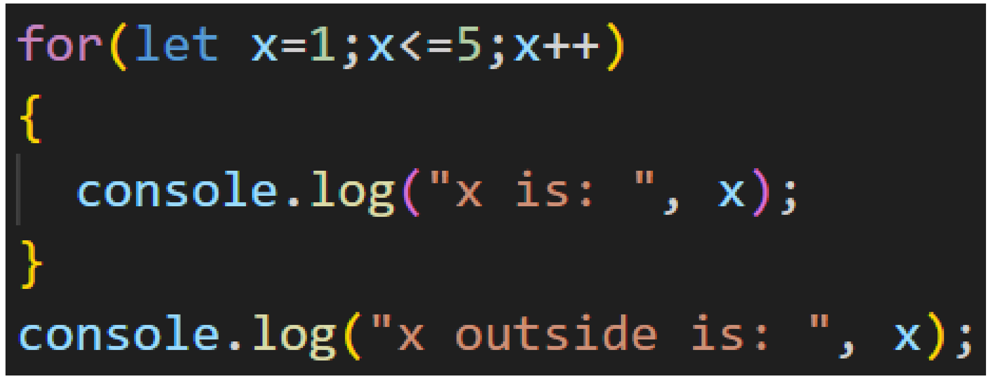
Here, variable **'x'** outside the function will be considered as a global variable and hence function **b()** can easily access it and calculate the sum as-
5 + 50 = 55.

But, variable **'x'** inside function **a()** is a local variable and its scope is limited to that function only and hence function **b()** cannot access the value of a local variable defined under different function.

Let Vs. Var

The scope of **‘let’** is limited to the curly braces while the variables declared using **‘var’** can be access outside its defined scope.

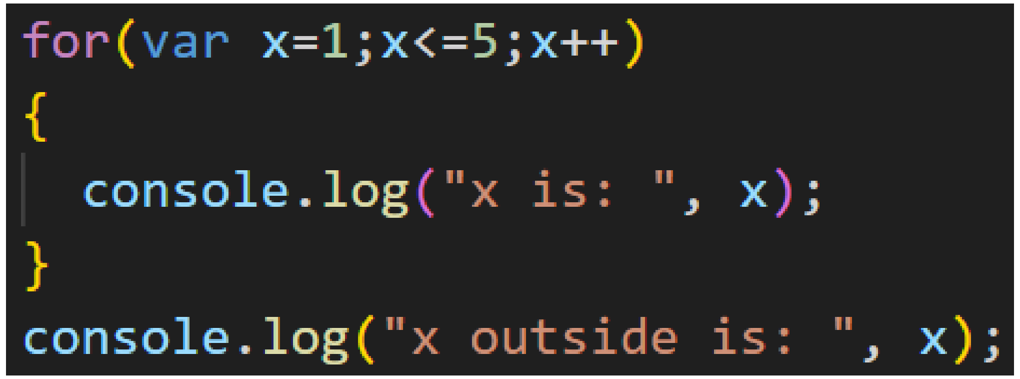
Example-1:



```
for(let x=1;x<=5;x++)
{
  console.log("x is: ", x);
}
console.log("x outside is: ", x);
```

It will give error while accessing **‘x’** outside.

Example-2:



```
for(var x=1;x<=5;x++)
{
  console.log("x is: ", x);
}
console.log("x outside is: ", x);
```

Const Keyword

- The **const** declaration creates a read-only reference to a value.
- The value of a constant cannot change through re-assignment, and it can't be re-declared.
- The following rules are TRUE for a variable declared using the const keyword-
  - Constants cannot be reassigned a value.
  - A constant cannot be re-declared.
  - A Constants must be initialized during its declaration.

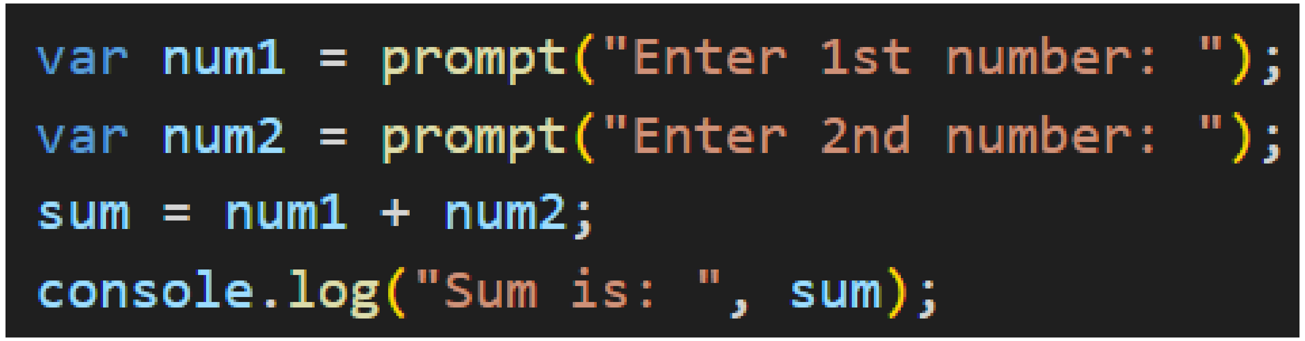
Example-

```
const x = 10
x = 12      // will produce error ‘Assignment to constant variable’
```

Typecasting: string to int

- Typecasting means conversion of one data type into another.

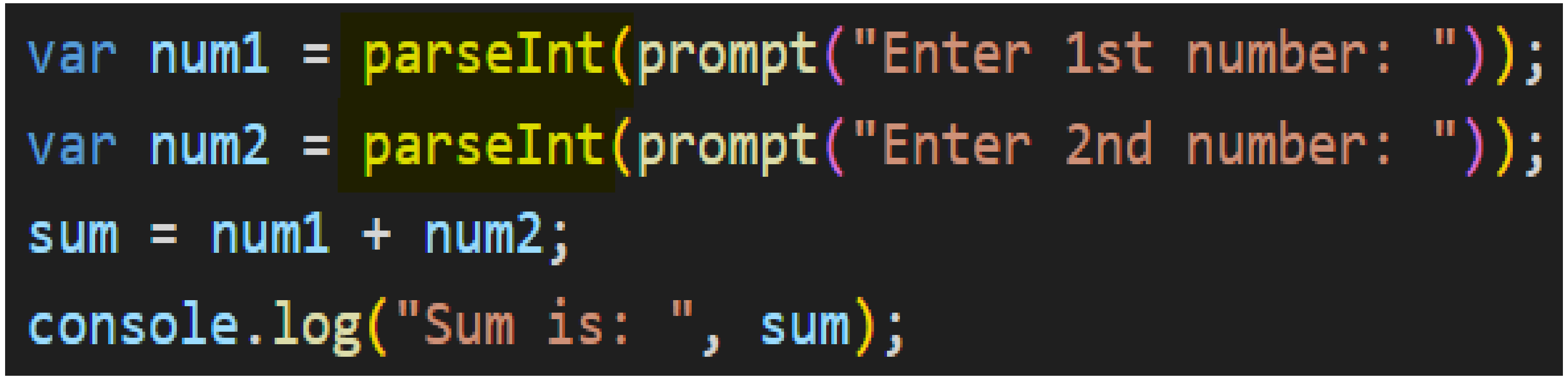
Example-



```
var num1 = prompt("Enter 1st number: ");
var num2 = prompt("Enter 2nd number: ");
sum = num1 + num2;
console.log("Sum is: ", sum);
```

This code will actually do the **concatenation** of numbers instead of performing addition operation.
Because, the **prompt()** method always creates a string variable.

We need to parse/type-cast the values entered by the user from string type to an integer type .

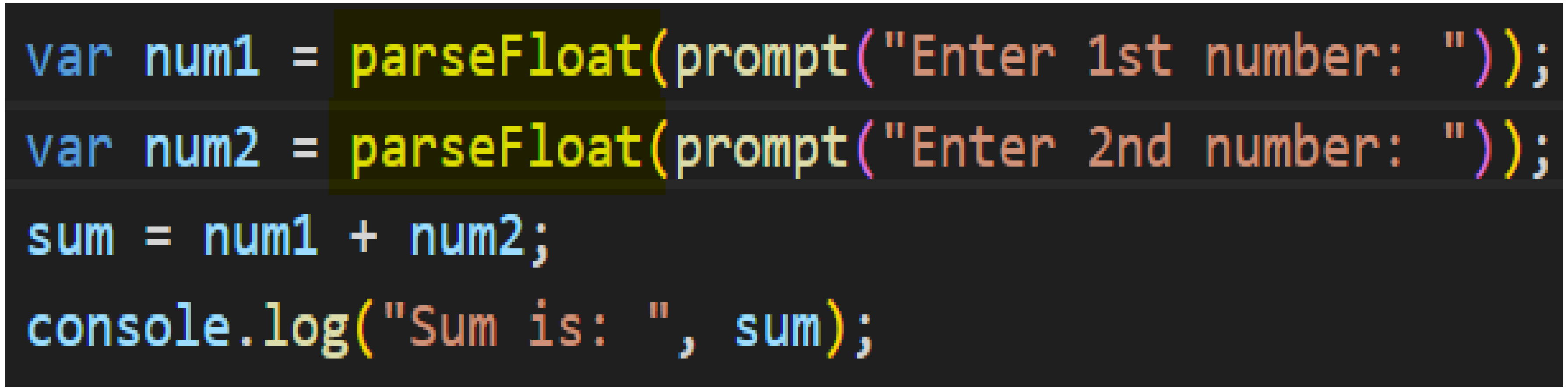


```
var num1 = parseInt(prompt("Enter 1st number: "));
var num2 = parseInt(prompt("Enter 2nd number: "));
sum = num1 + num2;
console.log("Sum is: ", sum);
```

Typecasting: string to float

- Conversion of String to Float type.

Example-

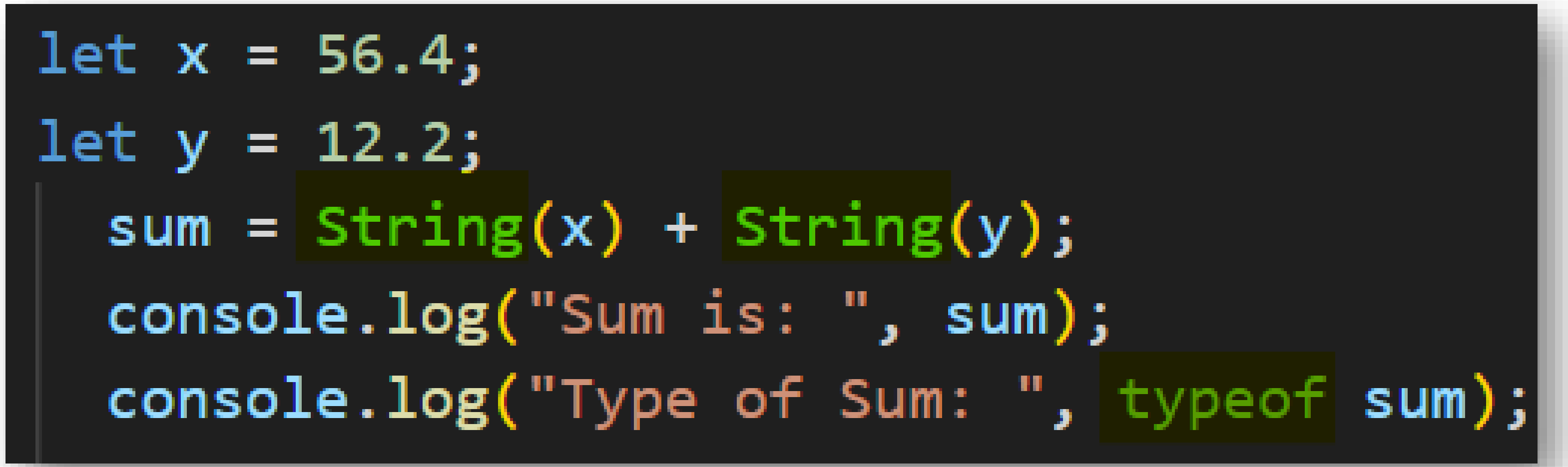


```
var num1 = parseFloat(prompt("Enter 1st number: "));
var num2 = parseFloat(prompt("Enter 2nd number: "));
sum = num1 + num2;
console.log("Sum is: ", sum);
```

Typecasting: number to string

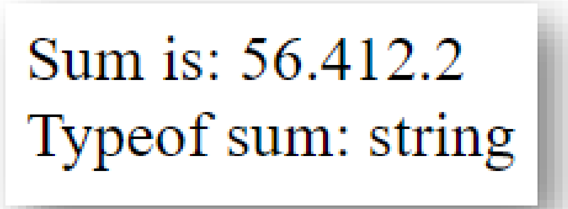
- Conversion of Number (integer/float) to String type.

Example-



```
let x = 56.4;
let y = 12.2;
  sum = String(x) + String(y);
  console.log("Sum is: ", sum);
  console.log("Type of Sum: ", typeof sum);
```

Output-



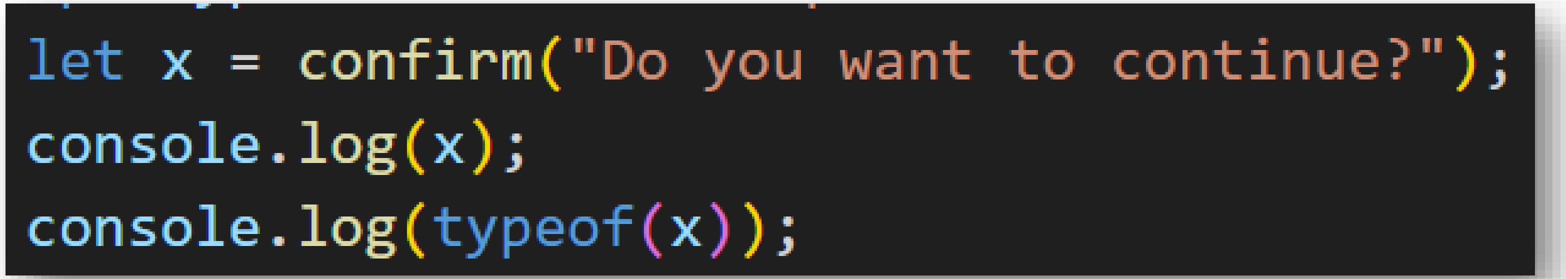
```
Sum is: 56.412.2
Typeof sum: string
```

confirm()

- The **confirm()** function in JavaScript is used to display a modal dialog box with a message and two buttons: **“OK”** and **“Cancel”**.

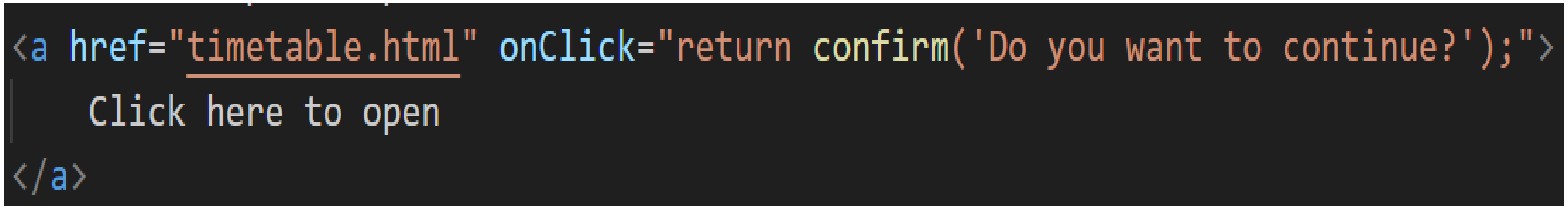
- It is commonly used to obtain user confirmation for an action or decision in a web application.

- **Example-**



```javascript
let x = confirm("Do you want to continue?");
console.log(x);
console.log(typeof(x));
```

- **Example-**

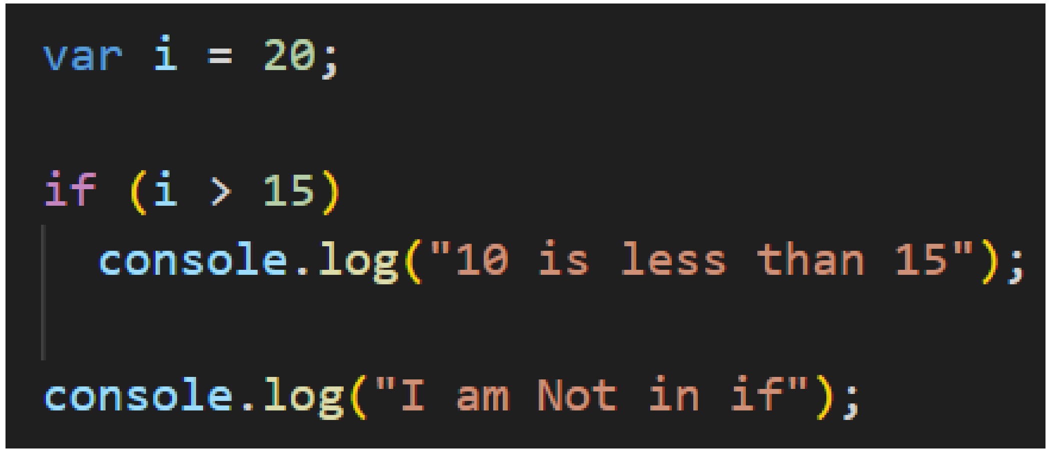


```html
<a href="timetable.html" onClick="return confirm('Do you want to continue?');">
    Click here to open
</a>
```

Conditional Statements: if

- The **conditional statement** will perform some action for the specific condition.
- If the condition meets then the particular block of action will be executed otherwise it will execute another block of action that satisfies that particular condition.
- In JavaScript we have **if…else** blocks to control the flow of our program on condition basis.

Example-



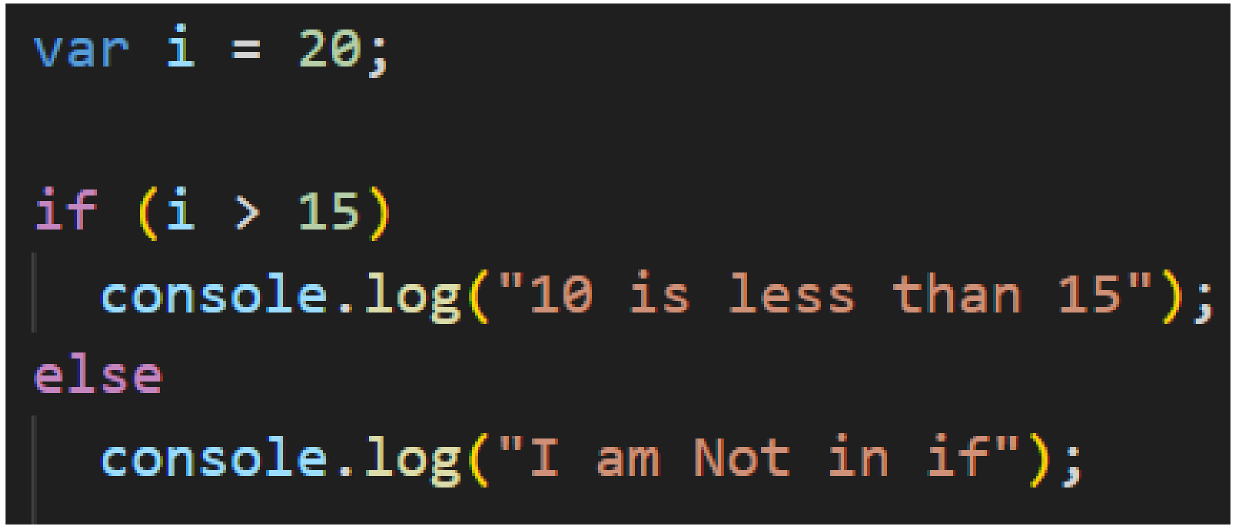
```
var i = 20;

if (i > 15)
  console.log("10 is less than 15");

console.log("I am Not in if");
```

If we do not provide the curly braces '{' and '}' after **if(condition)** then, by default, if statement will consider the immediate one statement to be inside its block.

Conditional Statements: else

- The **if statement** tells us that if a condition is **true**, it will execute a block of statements and if the condition is false, it won't.
- But what if we want to do something else if the condition is **false**. Here comes the else statement.
- We can use the **else statement** with the if statement to execute a block of code when the condition is false.

Example-



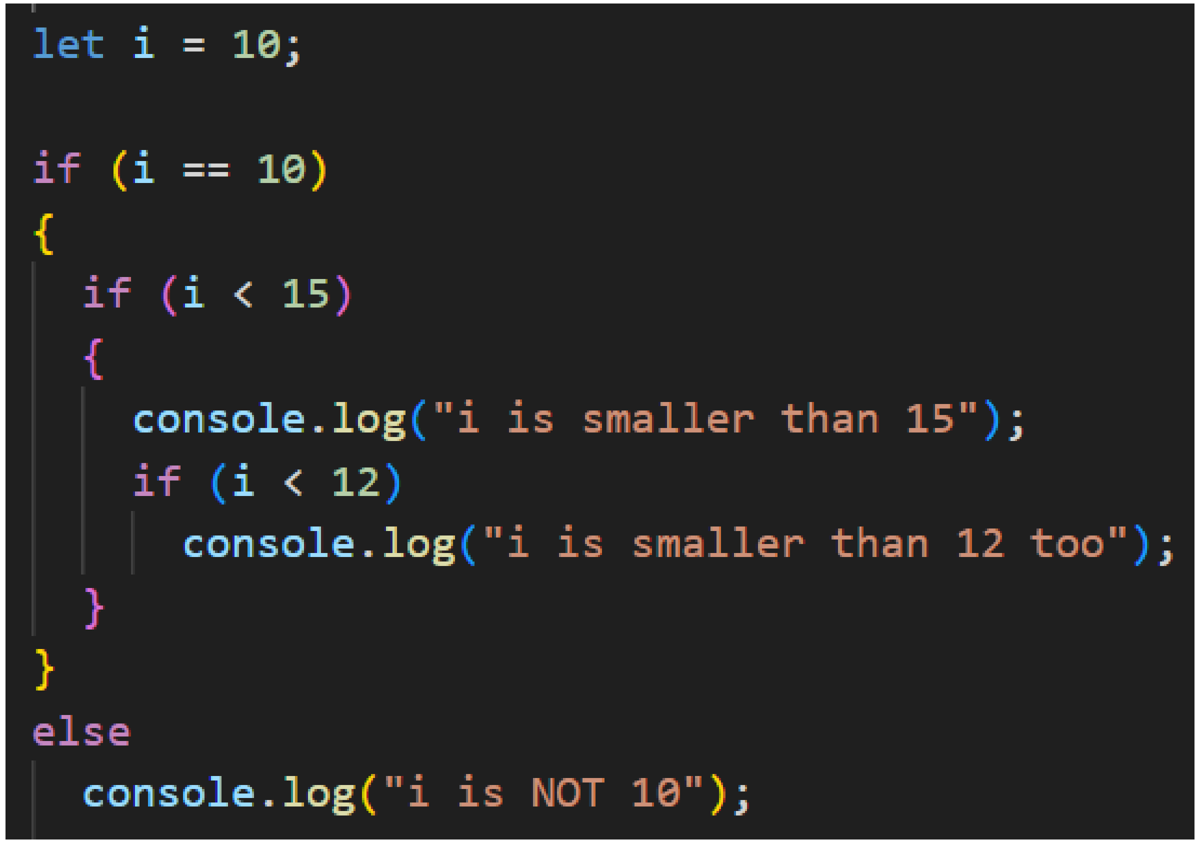
```
var i = 20;

if (i > 15)
  console.log("10 is less than 15");
else
  console.log("I am Not in if");
```

Conditional Statements: nested

A nested **if** is an if statement that is the target of another if or else.

Nested **if statements** mean one if statement inside another if statement.

Example-

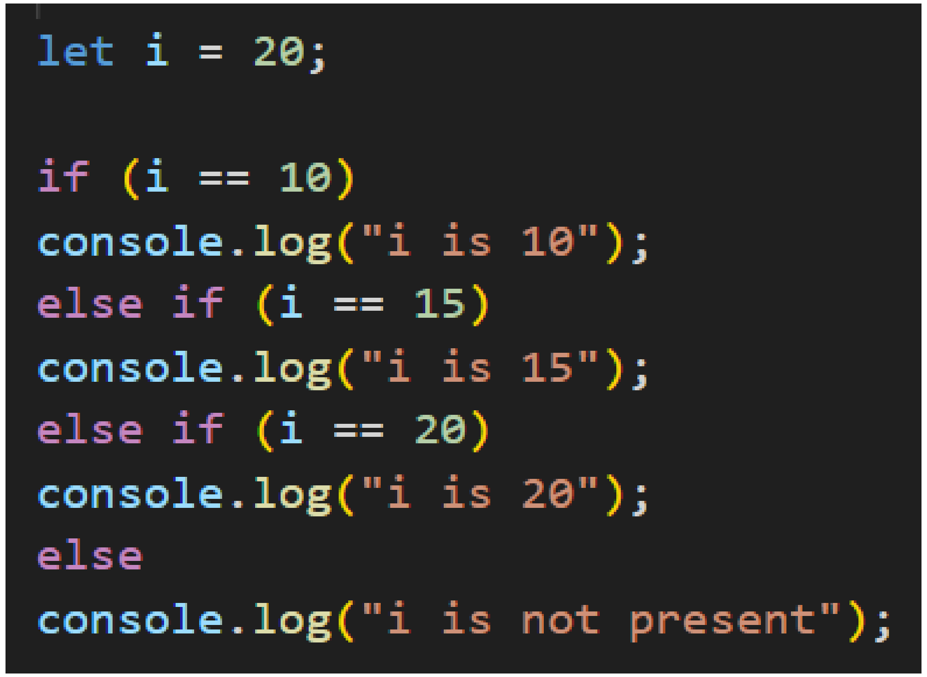


```
let i = 10;

if (i == 10)
{
  if (i < 15)
  {
    console.log("i is smaller than 15");
    if (i < 12)
      console.log("i is smaller than 12 too");
  }
}
else
  console.log("i is NOT 10");
```

Conditional Statements: ladder

- With the help of **if-else-if ladder statement**, we can decide among multiple options.
- In this ladder, if statements are executed from top to down.
- As soon as one of the conditions controlling the **if is true**, the statement associated with that if is executed, and the rest of the ladder is bypassed.
- If none of the conditions is true, then the final **else statement** will be executed.

Example-



```
let i = 20;

if (i == 10)
console.log("i is 10");
else if (i == 15)
console.log("i is 15");
else if (i == 20)
console.log("i is 20");
else
console.log("i is not present");
```

Switch Case

The **switch case** statement is also used for decision making purposes.

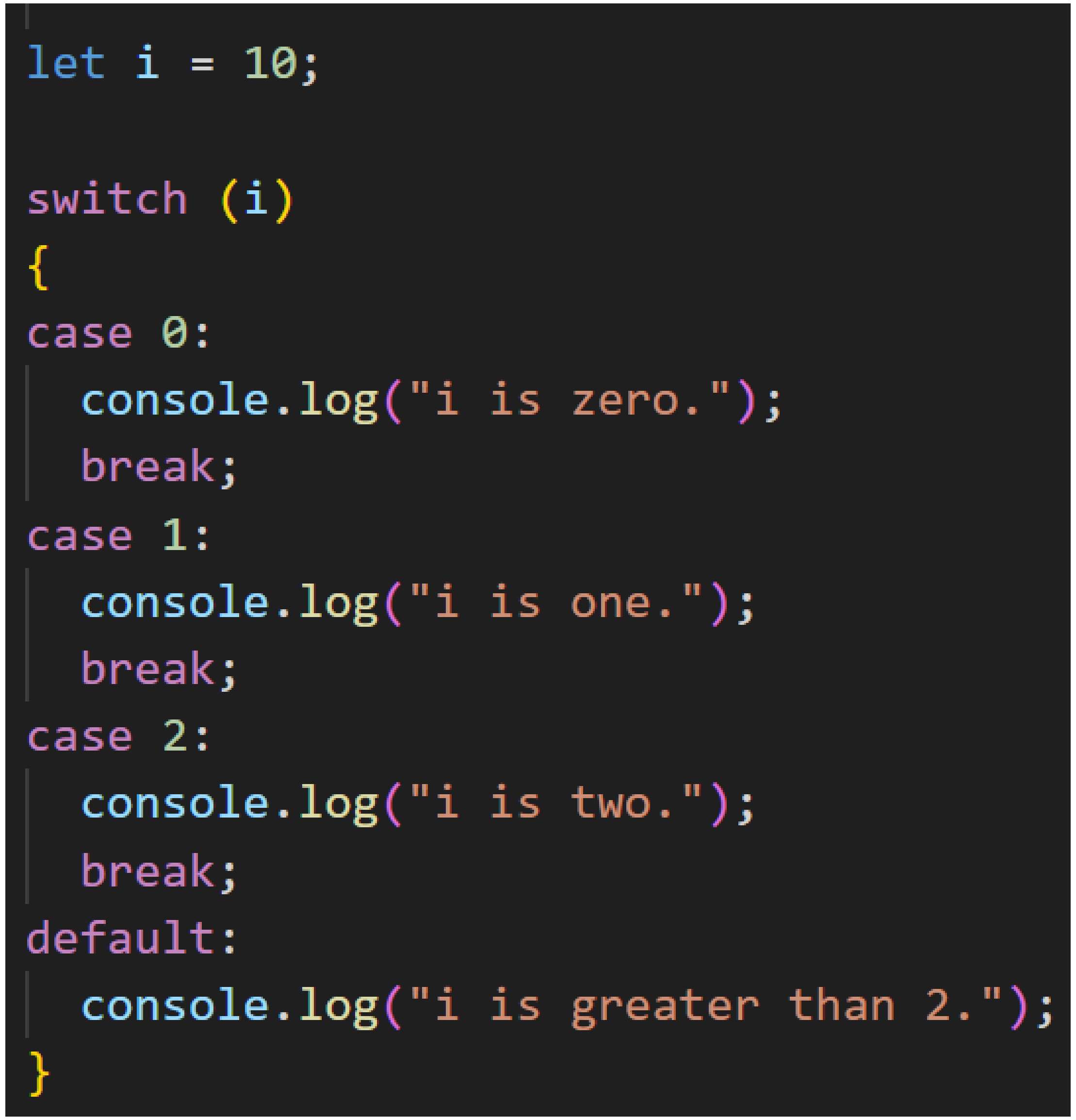
Scenario- Consider a situation when we want to test a variable for hundred different values and based on the test, we want to execute some tasks. Using an if-else statements for this purpose will be less efficient over switch case statements and also it will make the code look messy.

Syntax-

```
switch(expression)
{
     case value1:
          statement1;
          break;
     case value2:
          statement2;
          break;
    .
    .
    case valueN:
         statementN;
         break;
    default:
         statementDefault;
}
```

- **Expression** can be of numbers or strings type.
- **Statements** inside the case executes whose case value matches with the expression. Duplicate case values are not allowed.
- The **default statement** is optional. If the expression passed to switch does not matches with value in any case, then the statement under default will be executed.
- The **break statement** is used inside the switch to terminate a statement sequence.
- The break statement is optional. If omitted, execution will continue on into the next case.

Switch Case: Example

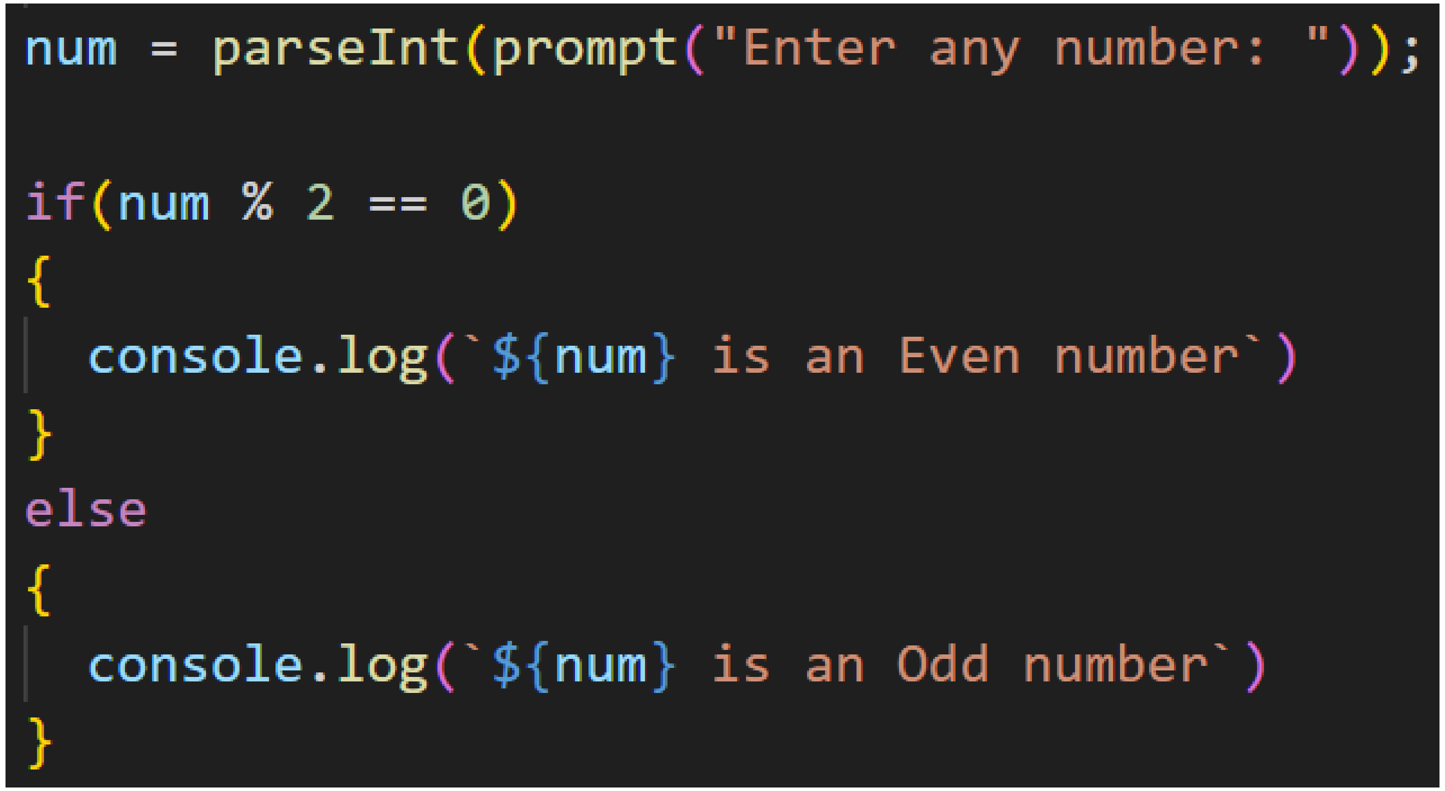


```
let i = 10;

switch (i)
{
case 0:
  console.log("i is zero.");
  break;
case 1:
  console.log("i is one.");
  break;
case 2:
  console.log("i is two.");
  break;
default:
  console.log("i is greater than 2.");
}
```

Exercise-1

Write a program that accepts a number from user and prints if it is an Even or Odd?

Solution-

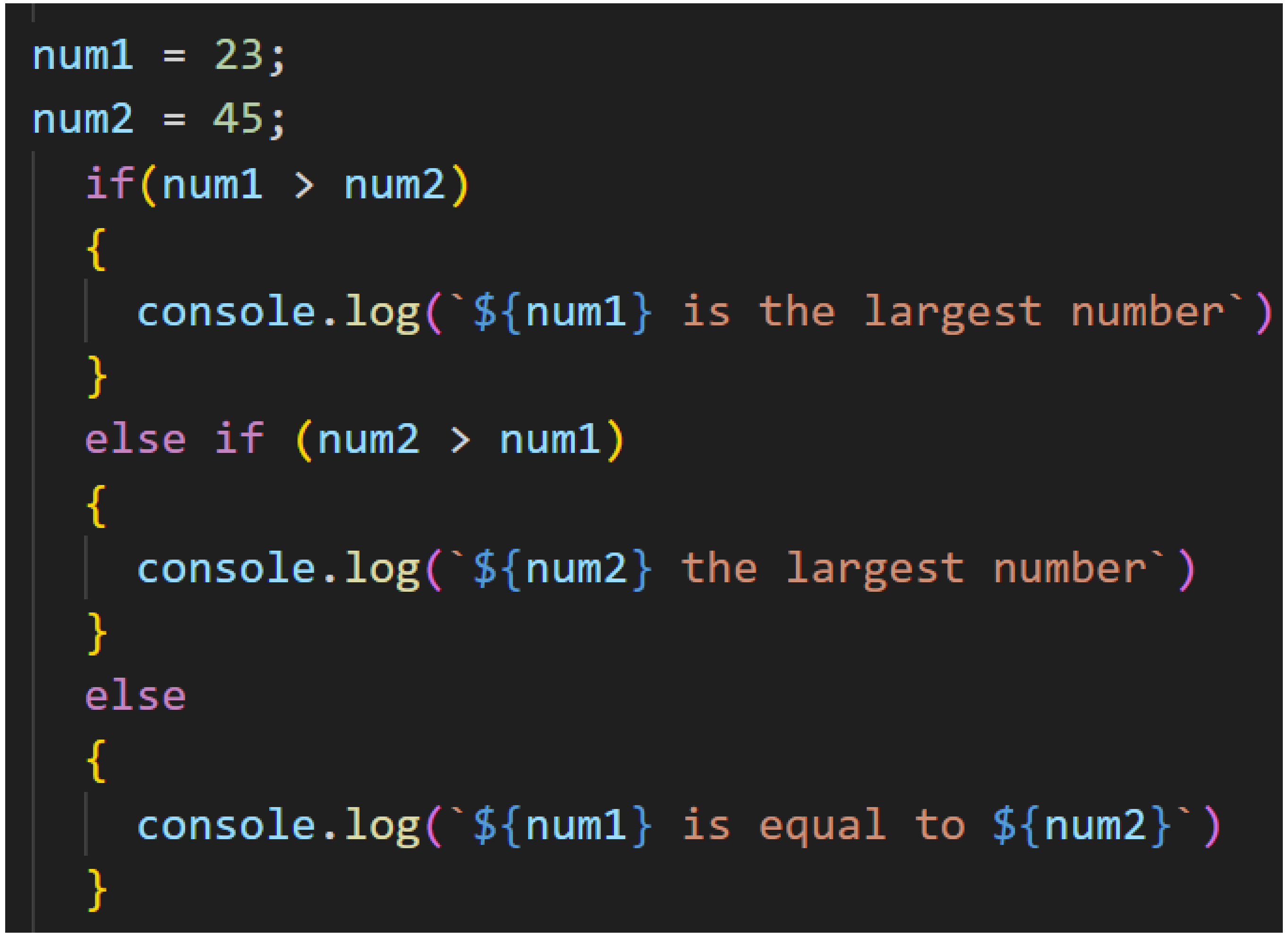


```javascript
num = parseInt(prompt("Enter any number: "));

if(num % 2 == 0)
{
  console.log(`${num} is an Even number`)
}
else
{
  console.log(`${num} is an Odd number`)
}
```

Exercise-2

Write a program to find out largest of two numbers?

Solution-



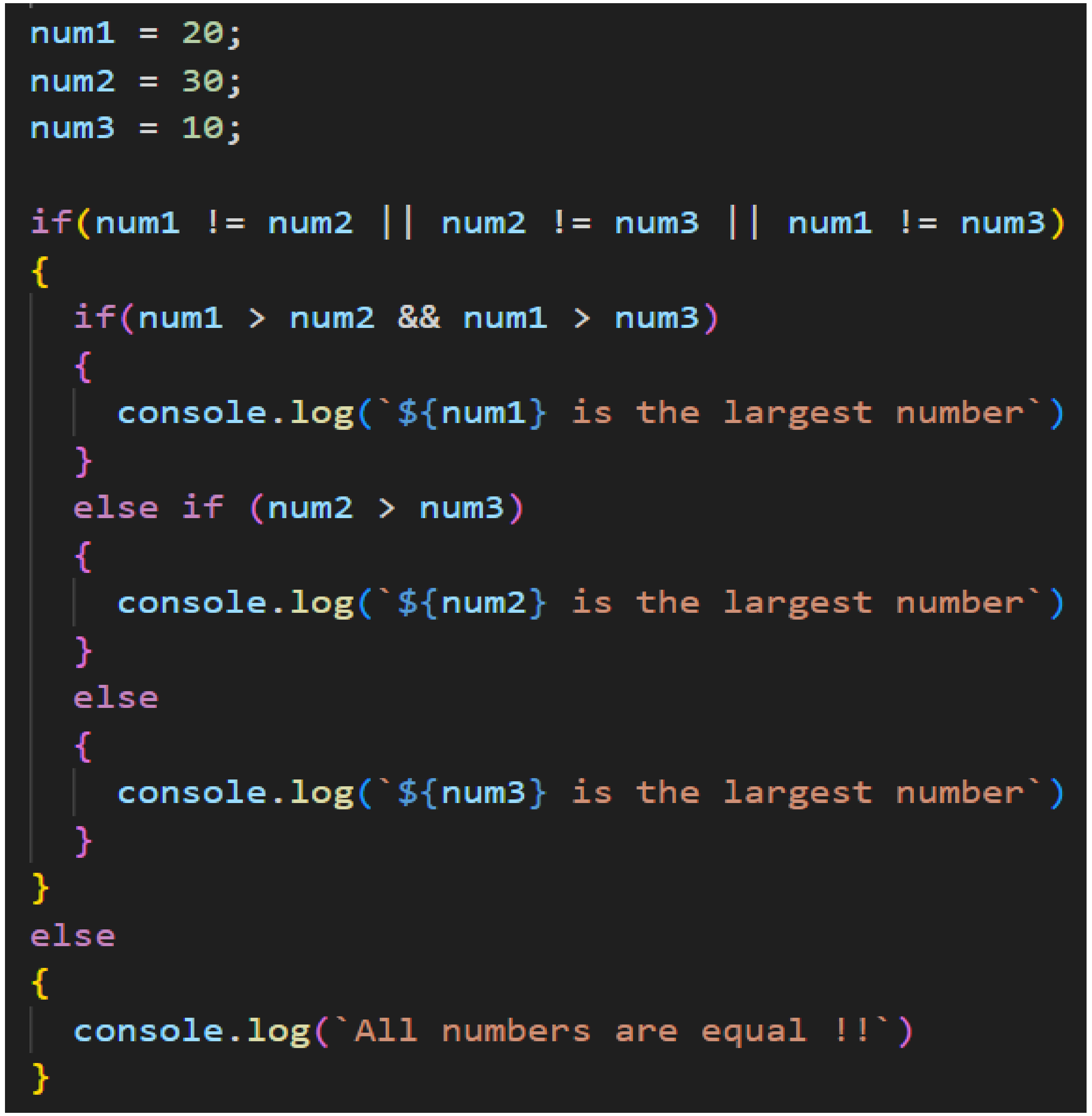
```
num1 = 23;
num2 = 45;
  if(num1 > num2)
  {
    console.log(`${num1} is the largest number`)
  }
  else if (num2 > num1)
  {
    console.log(`${num2} the largest number`)
  }
  else
  {
    console.log(`${num1} is equal to ${num2}`)
  }
```

Exercise-3

Write a program to find out largest of three numbers?

Logic-

- num1 is the largest if num1>num2 and num1>num3.
- num2 is the largest if num2>num3.
- Else print num3 as largest.

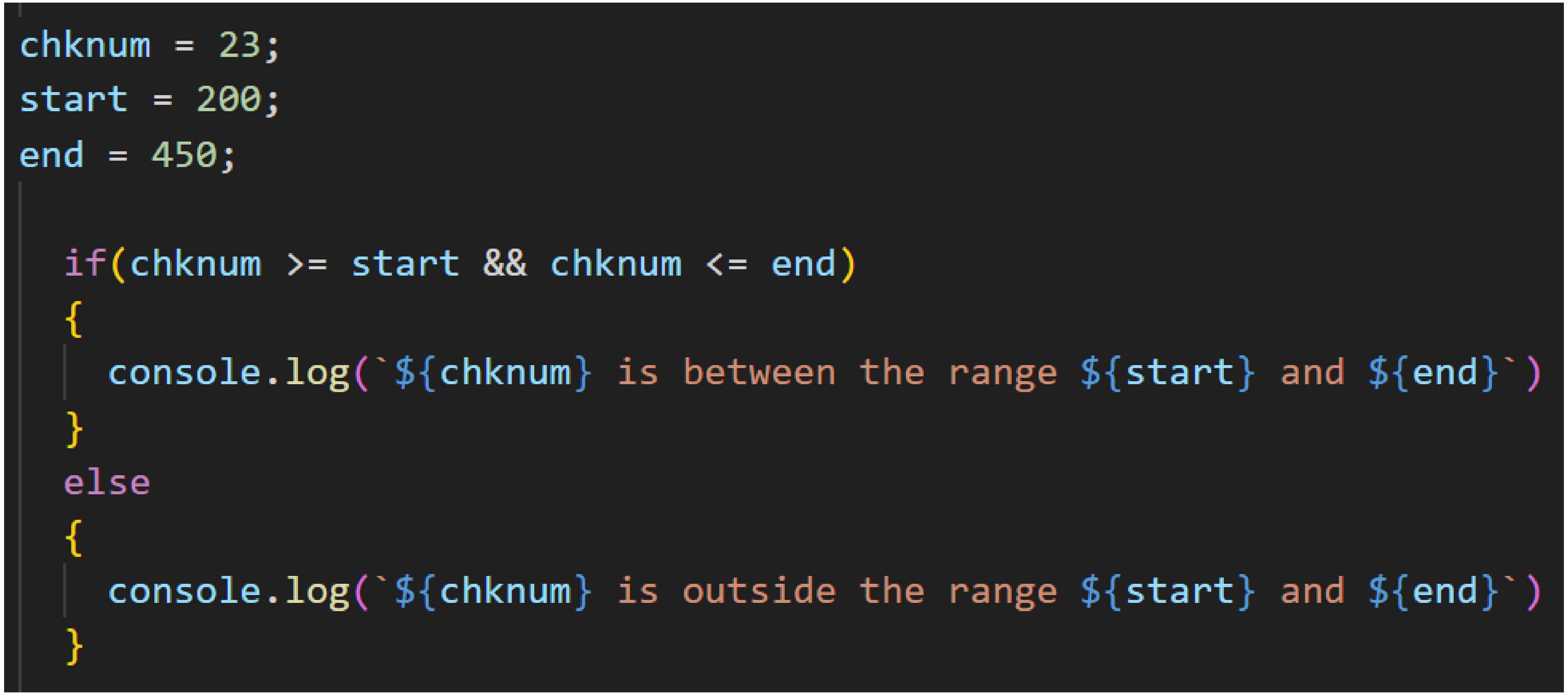


```
num1 = 20;
num2 = 30;
num3 = 10;

if(num1 != num2 || num2 != num3 || num1 != num3)
{
  if(num1 > num2 && num1 > num3)
  {
    console.log(`${num1} is the largest number`)
  }
  else if (num2 > num3)
  {
    console.log(`${num2} is the largest number`)
  }
  else
  {
    console.log(`${num3} is the largest number`)
  }
}
else
{
  console.log(`All numbers are equal !!`)
}
```

Exercise-4

Write a program to check whether a number is between start and end range?

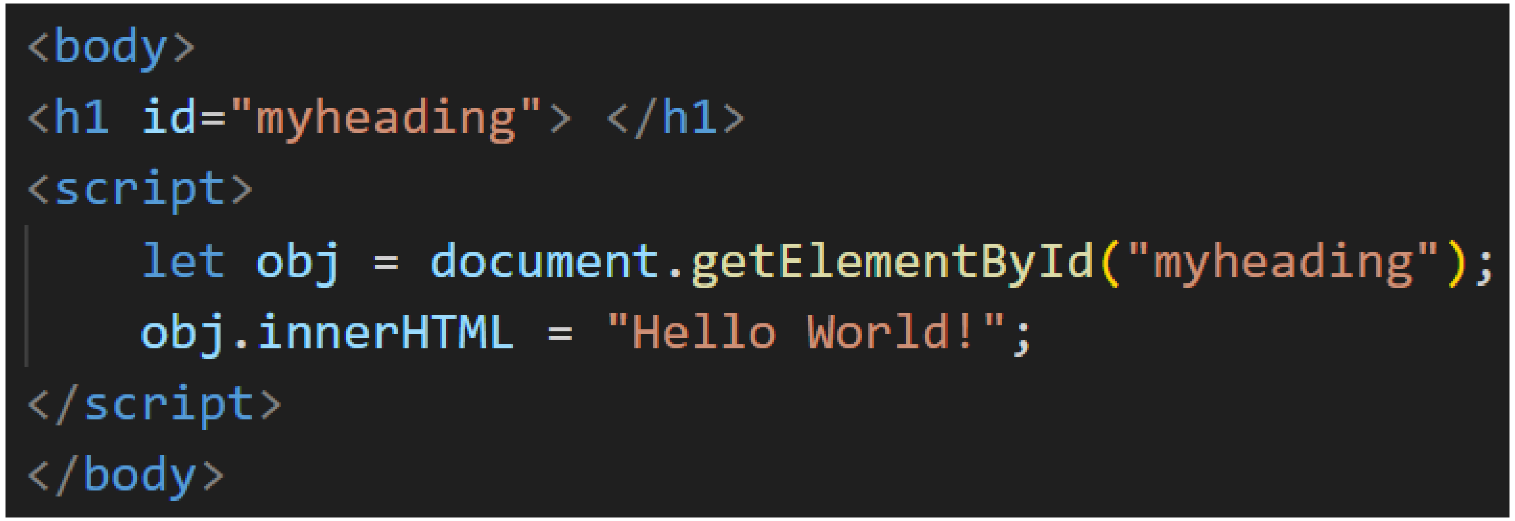
Solution-



```javascript
chknum = 23;
start = 200;
end = 450;

  if(chknum >= start && chknum <= end)
  {
    console.log(`${chknum} is between the range ${start} and ${end}`)
  }
  else
  {
    console.log(`${chknum} is outside the range ${start} and ${end}`)
  }
```

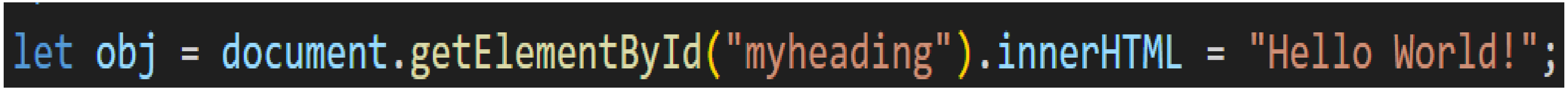
innerHTML Property

- **innerHTML** is used to select the HTML element to set its content through JavaScript.
- To select an HTML element using id attribute, we need to use the method **getElementById()** under **document** object.
- **Example-1:**



```
<body>
<h1 id="myheading"> </h1>
<script>
    let obj = document.getElementById("myheading");
    obj.innerHTML = "Hello World!";
</script>
</body>
```

- **Example-2:**

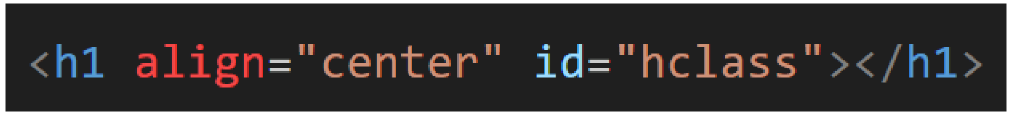


```
let obj = document.getElementById("myheading").innerHTML = "Hello World!";
```

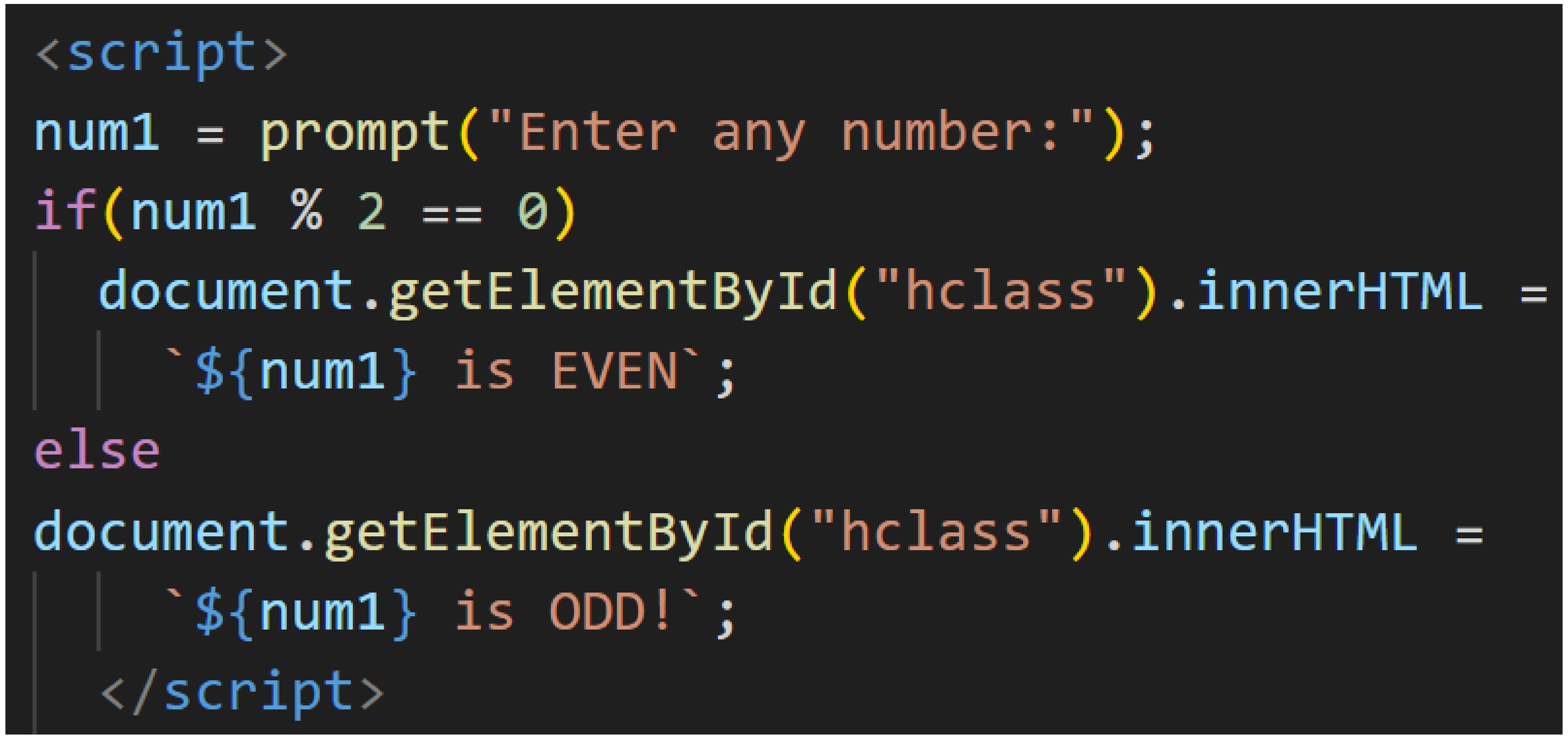
Exercise-1

Write a program to check whether an entered number is EVEN or ODD?

Display your result inside <H1> tag using innerHTML method like, “12 is EVEN” or “3 is ODD”.



```html
<h1 align="center" id="hclass"></h1>
```



```html
<script>
num1 = prompt("Enter any number:");
if(num1 % 2 == 0)
  document.getElementById("hclass").innerHTML =
    `${num1} is EVEN`;
else
document.getElementById("hclass").innerHTML =
    `${num1} is ODD!`;
  </script>
```

Loops

- Looping in programming languages is a feature which facilitates the execution of a set of instructions repeatedly while some condition evaluates to true.
- There are three types of loops in JavaScript.

  1. for loop **2.** while loop **3.** do-while loop

Entry-Controlled loops: In this type of loops the test condition is tested before entering the loop body. **For Loop** and **While Loop** are entry-controlled loops.

Exit-Controlled Loops: In this type of loops the test condition is tested or evaluated at the end of loop body. Therefore, the loop body will execute at least once, irrespective of whether the test condition is true or false. **do – while Loop** is an exit-controlled loop. Let’s explore each of these loops one by one with examples.

While Loop

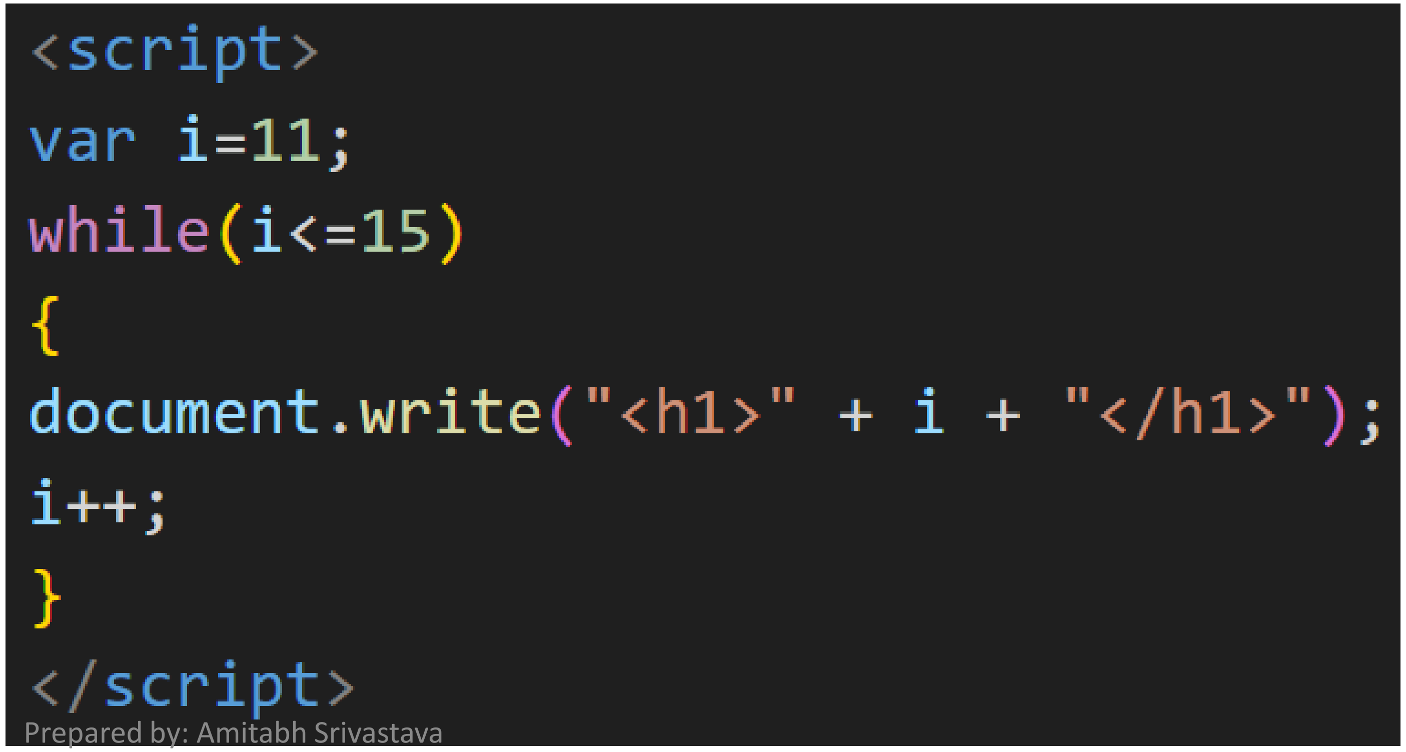
- **While loop** allows code to be executed repeatedly based on a given Boolean condition.

- It should be used if number of iterations is not known.

Syntax-

```
while(Boolean condition)
{
   Statements to be executed...;
}
```

Example-



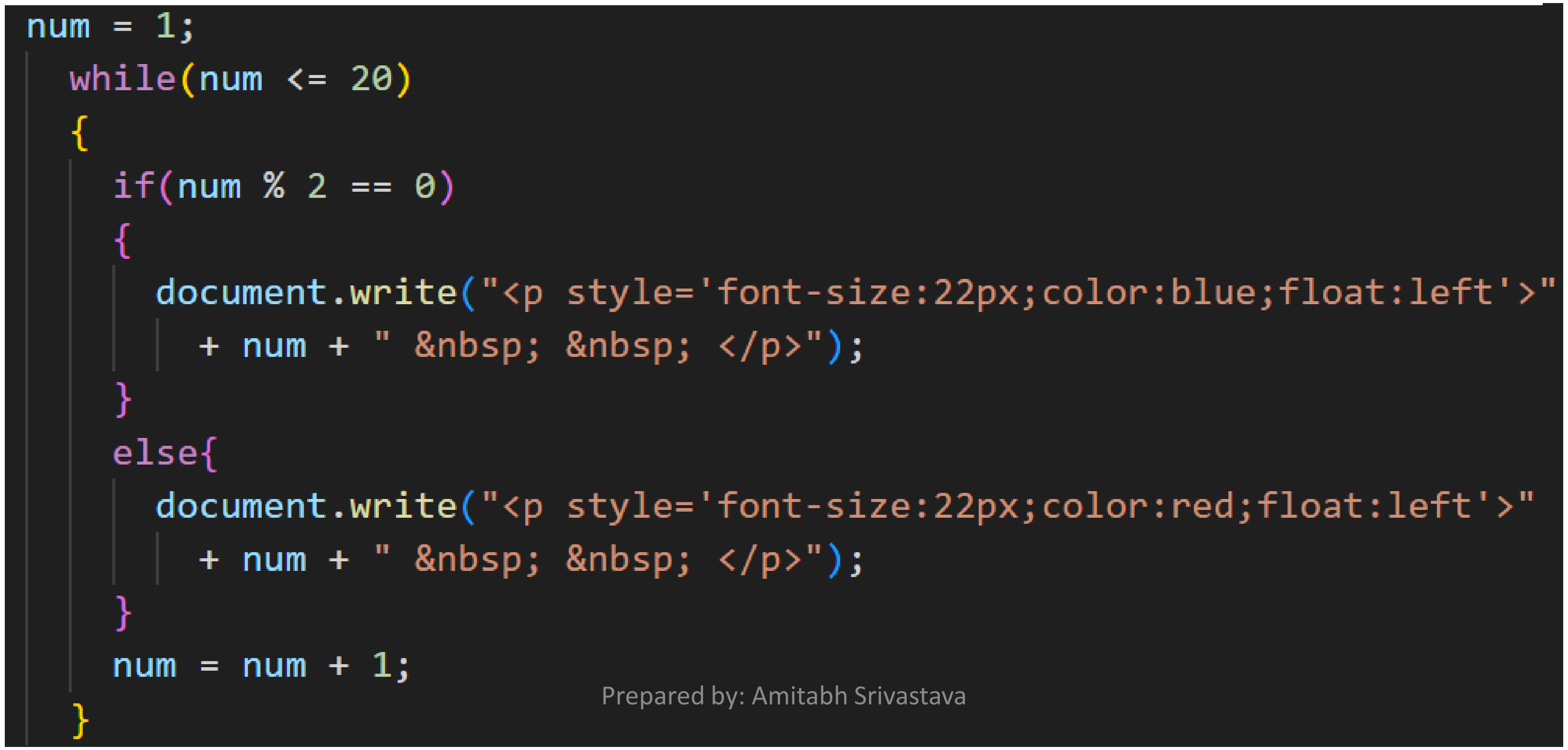
```
<script>
var i=11;
while(i<=15)
{
document.write("<h1>" + i + "</h1>");
i++;
}
</script>
```

While loop: Exercise-1

Write a program to display Even and Odd numbers from 1 to 20 in font-size 22px?

All the numbers should be in same line and Even numbers in Blue color and Odd numbers in Red color. Refer the below output.

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

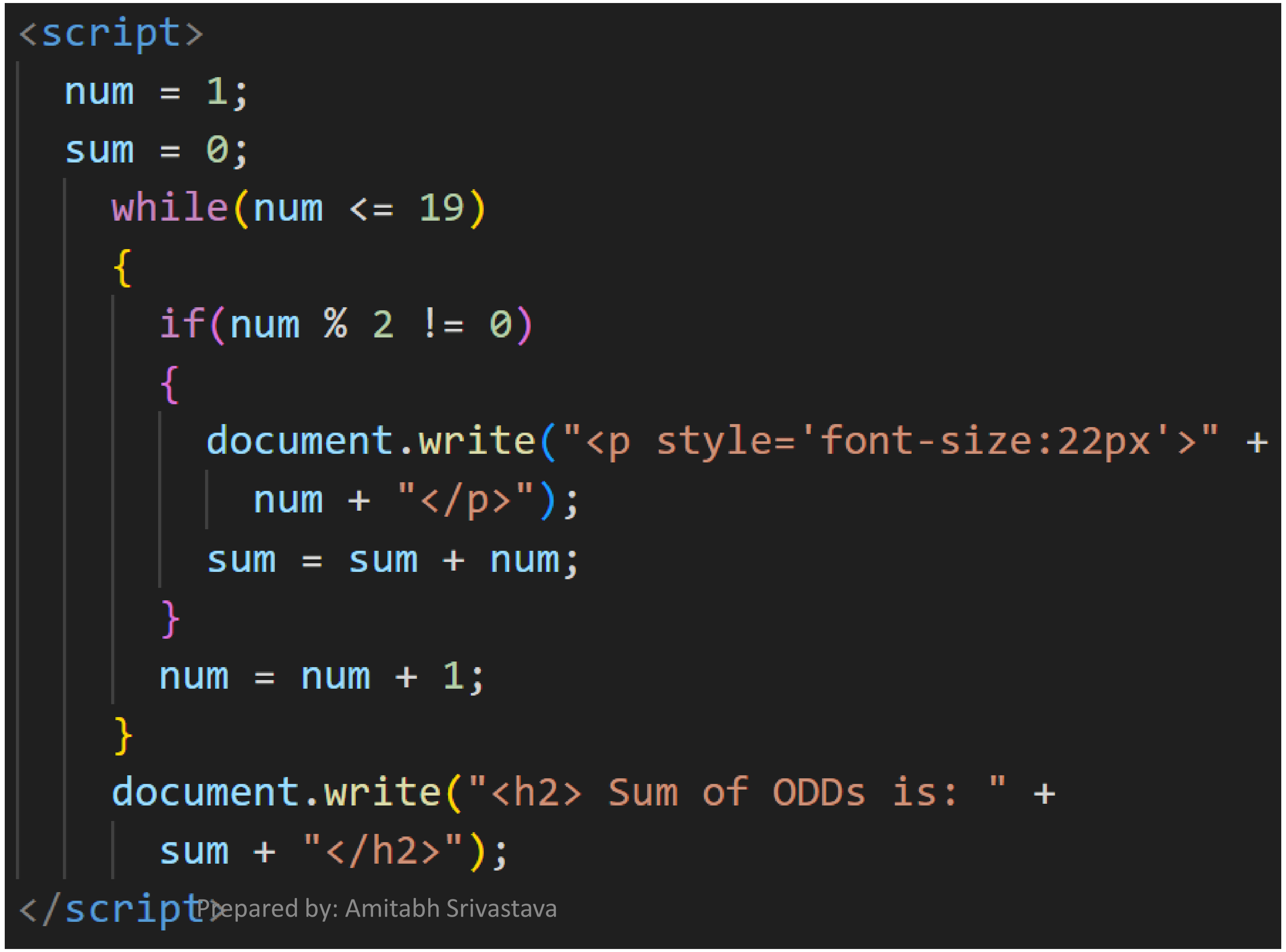


```
num = 1;
  while(num <= 20)
  {
    if(num % 2 == 0)
    {
      document.write("<p style='font-size:22px;color:blue;float:left'>"
        + num + " &nbsp; &nbsp; </p>");
    }
    else{
      document.write("<p style='font-size:22px;color:red;float:left'>"
        + num + " &nbsp; &nbsp; </p>");
    }
    num = num + 1;
  }
```

While loop: Exercise-2

Write a program to display ODD numbers from 1 to 20 in font-size 22px? At the end, also display the sum of all ODD numbers.

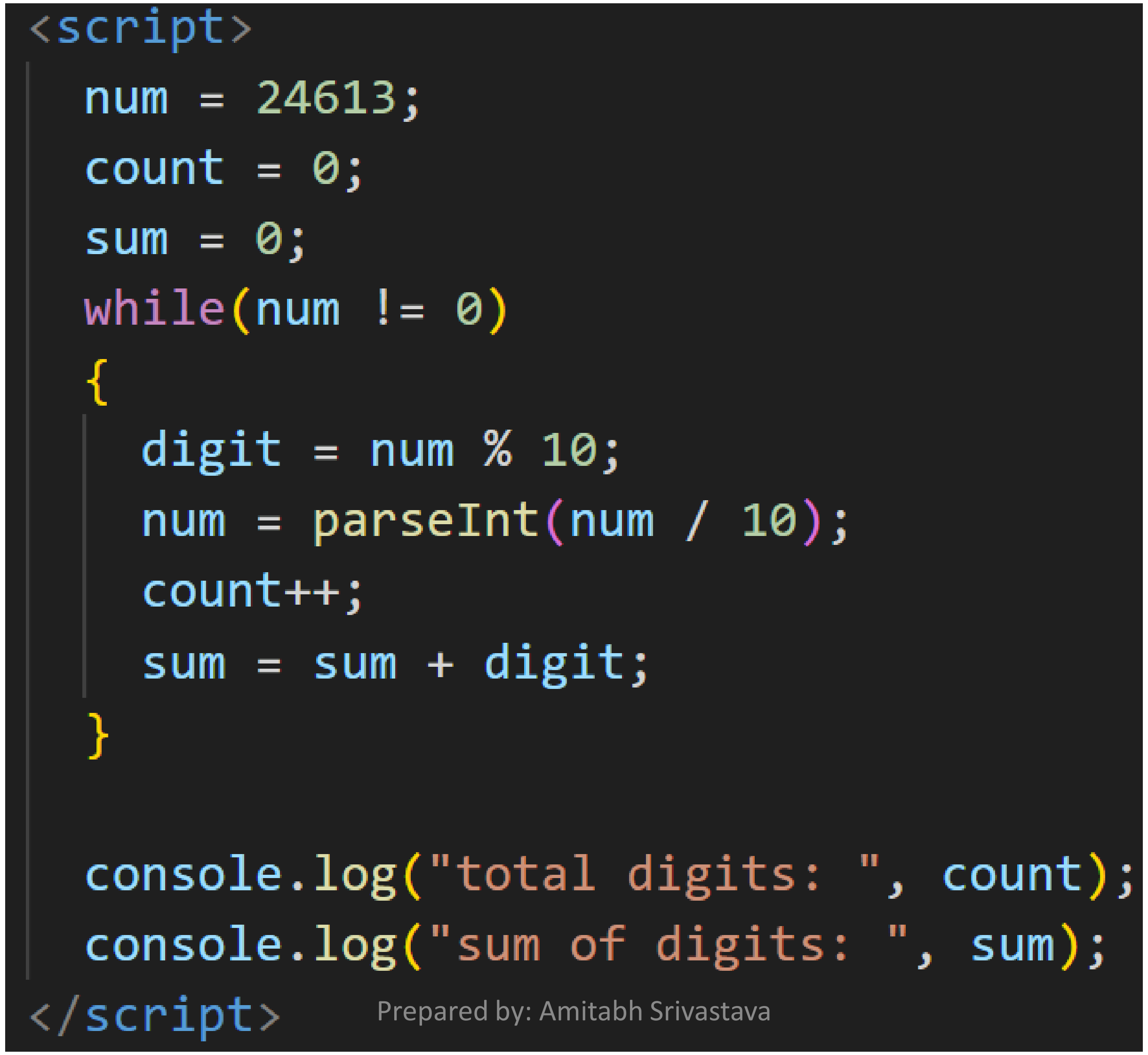
```
1

3

5

7

9

11

13

15

17

19

Sum of ODDs is: 100
```



```
<script>
  num = 1;
  sum = 0;
    while(num <= 19)
    {
      if(num % 2 != 0)
      {
        document.write("<p style='font-size:22px'>" +
          num + "</p>");
        sum = sum + num;
      }
      num = num + 1;
    }
    document.write("<h2> Sum of ODDs is: " +
      sum + "</h2>");
</script>
```

While loop: Exercise-3

Write a program to display the number of digits in a given number? Also display the sum of all digits?



```
<script>
  num = 24613;
  count = 0;
  sum = 0;
  while(num != 0)
  {
    digit = num % 10;
    num = parseInt(num / 10);
    count++;
    sum = sum + digit;
  }

  console.log("total digits: ", count);
  console.log("sum of digits: ", sum);
</script>
```

For Loop

- Unlike a while loop, a **for** statement includes the initialization, condition and increment/decrement in one line thereby providing a shorter, easy to debug structure of looping.

Syntax-

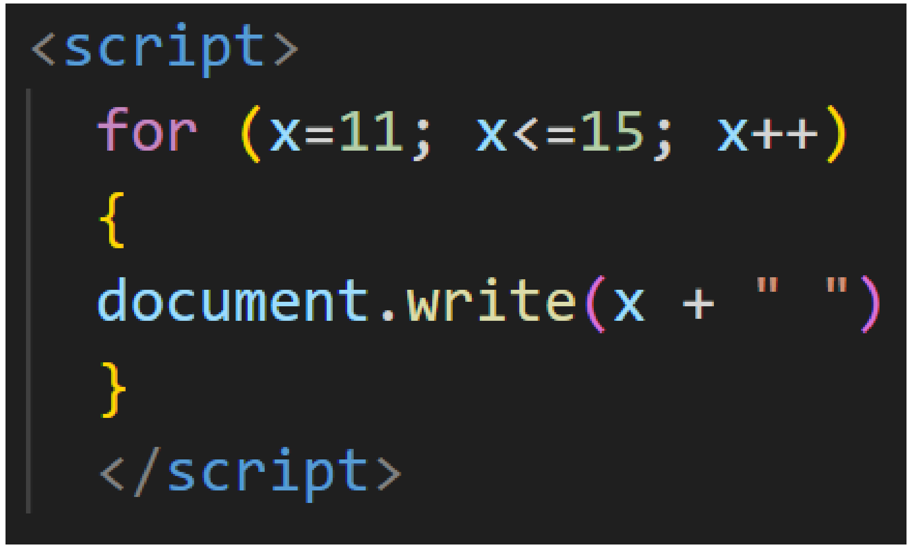
for (initialization; testing condition; increment/decrement)

{

  Statements to be executed…

}

Example-



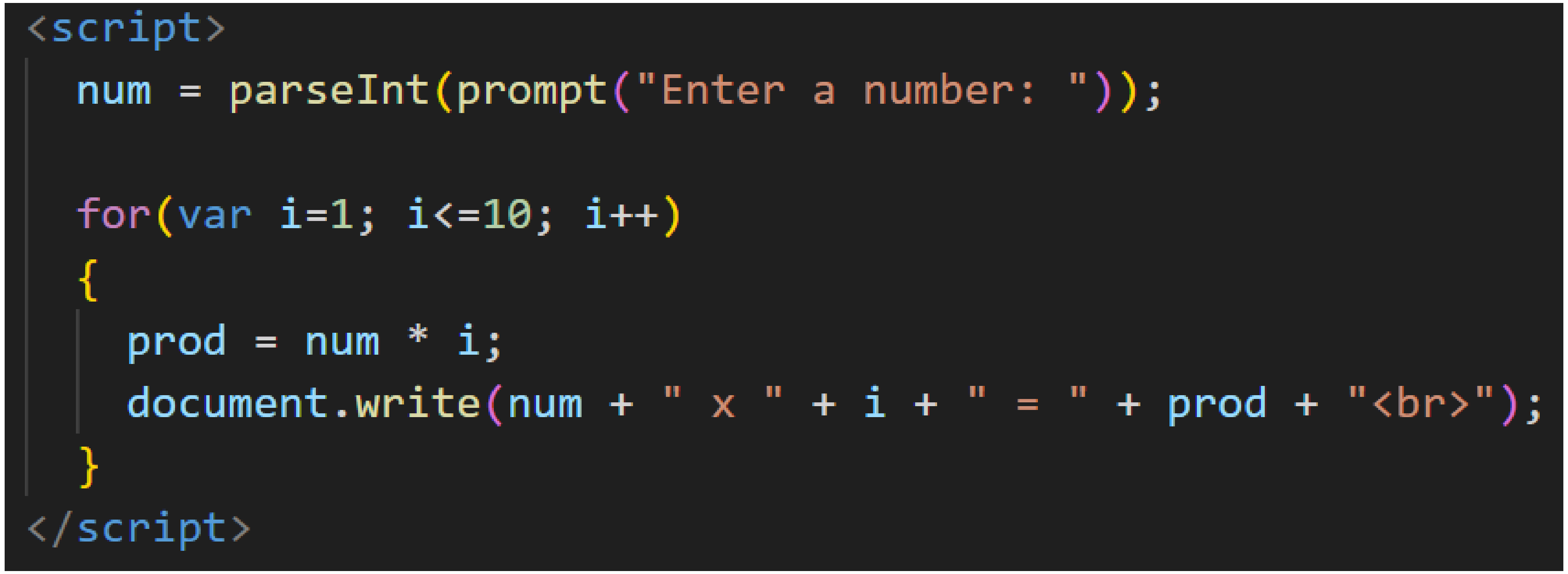
```
<script>
  for (x=11; x<=15; x++)
  {
  document.write(x + " ")
  }
  </script>
```

For loop: Exercise-1

Write a program to display the table of a number entered by the user?

```
23 x 1 = 23
23 x 2 = 46
23 x 3 = 69
23 x 4 = 92
23 x 5 = 115
23 x 6 = 138
23 x 7 = 161
23 x 8 = 184
23 x 9 = 207
23 x 10 = 230
```

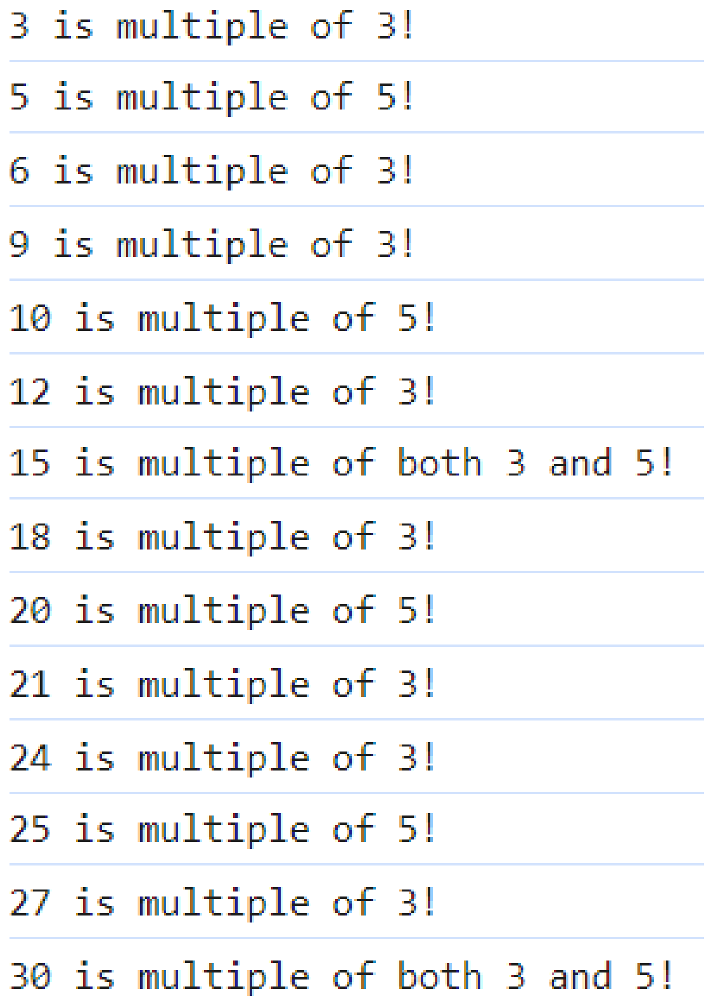
Solution-



```
<script>
  num = parseInt(prompt("Enter a number: "));

  for(var i=1; i<=10; i++)
  {
    prod = num * i;
    document.write(num + " x " + i + " = " + prod + "<br>");
  }
</script>
```

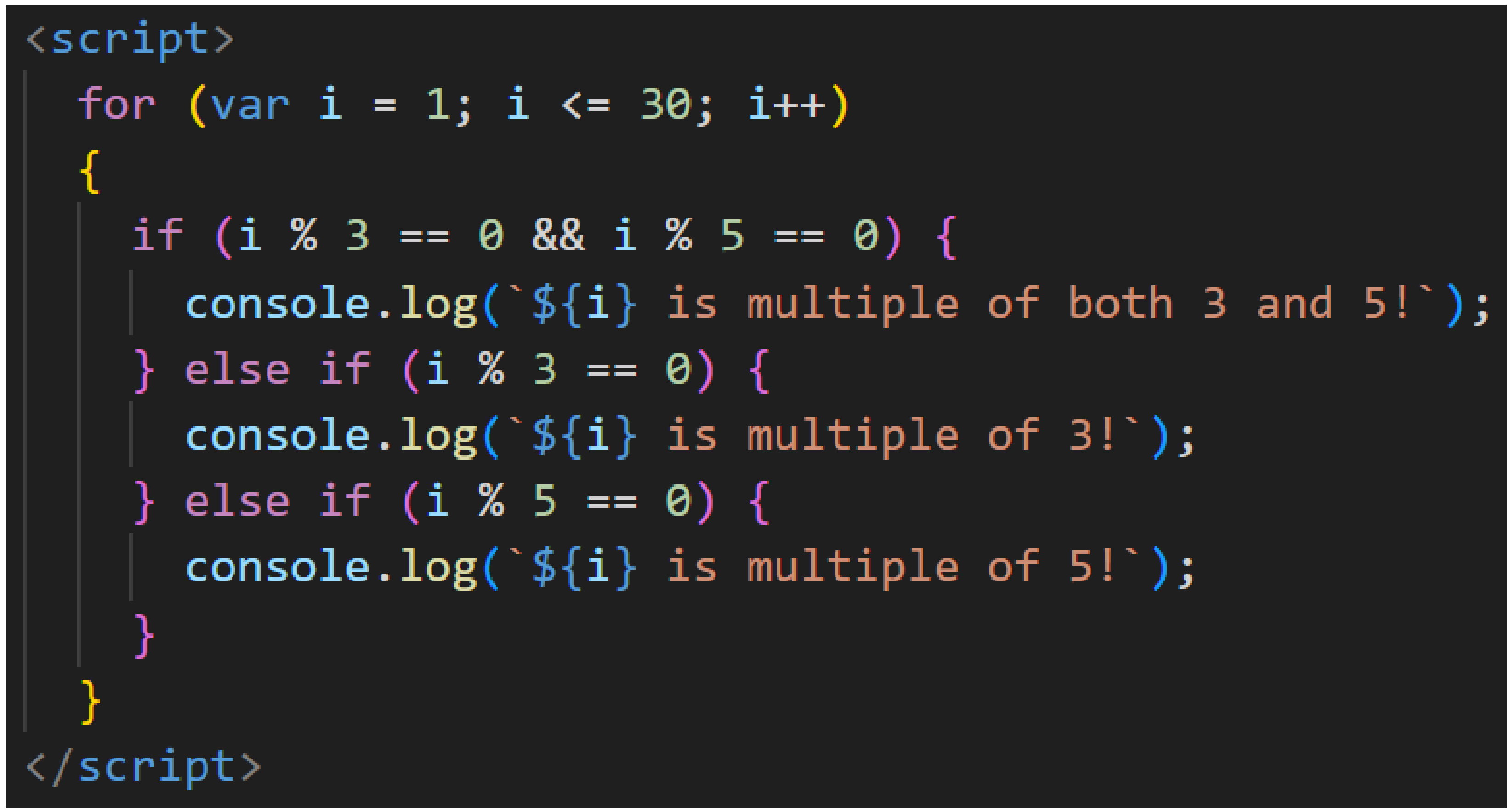
For loop: Exercise-2

Write a program to display all the multiples of 3 and 5 from 1 to 30? Show your result in console window.



```
3 is multiple of 3!
5 is multiple of 5!
6 is multiple of 3!
9 is multiple of 3!
10 is multiple of 5!
12 is multiple of 3!
15 is multiple of both 3 and 5!
18 is multiple of 3!
20 is multiple of 5!
21 is multiple of 3!
24 is multiple of 3!
25 is multiple of 5!
27 is multiple of 3!
30 is multiple of both 3 and 5!
```

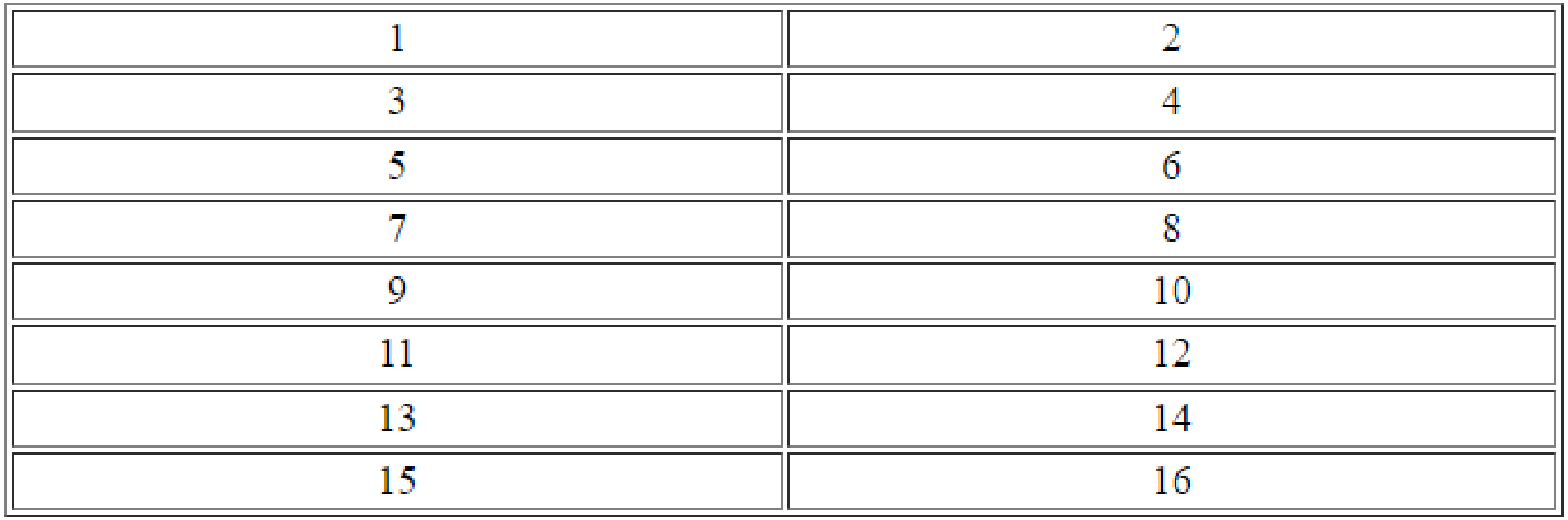
Solution-



```
<script>
  for (var i = 1; i <= 30; i++)
  {
    if (i % 3 == 0 && i % 5 == 0) {
      console.log(`${i} is multiple of both 3 and 5!`);
    } else if (i % 3 == 0) {
      console.log(`${i} is multiple of 3!`);
    } else if (i % 5 == 0) {
      console.log(`${i} is multiple of 5!`);
    }
  }
</script>
```

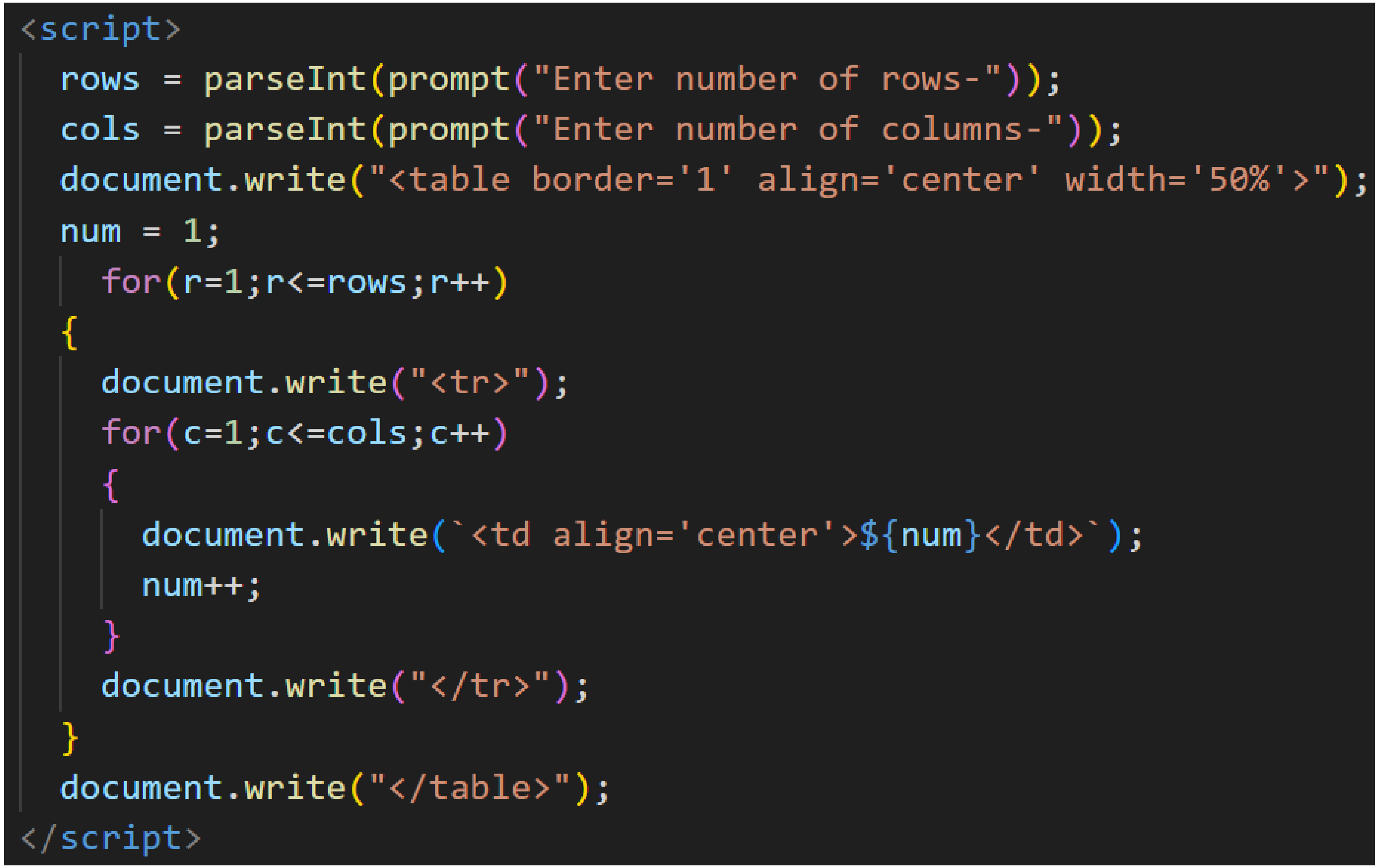
For loop: Exercise-3

Write a program to create a table by taking number of rows and columns from the user? Also print numbers in sequence in every cell. Refer below figure.



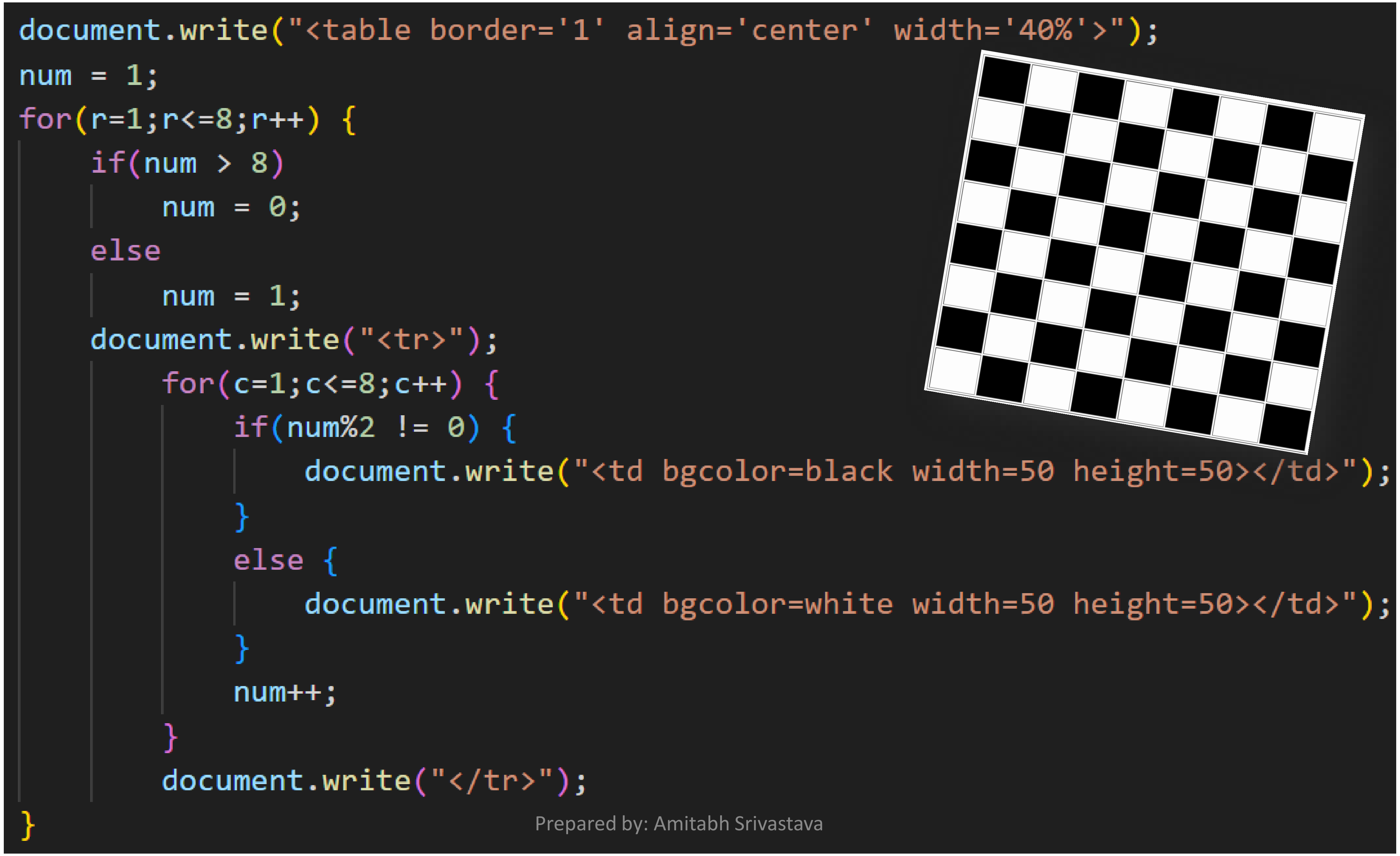
1	2
3	4
5	6
7	8
9	10
11	12
13	14
15	16

For loop: Exercise-3: Code

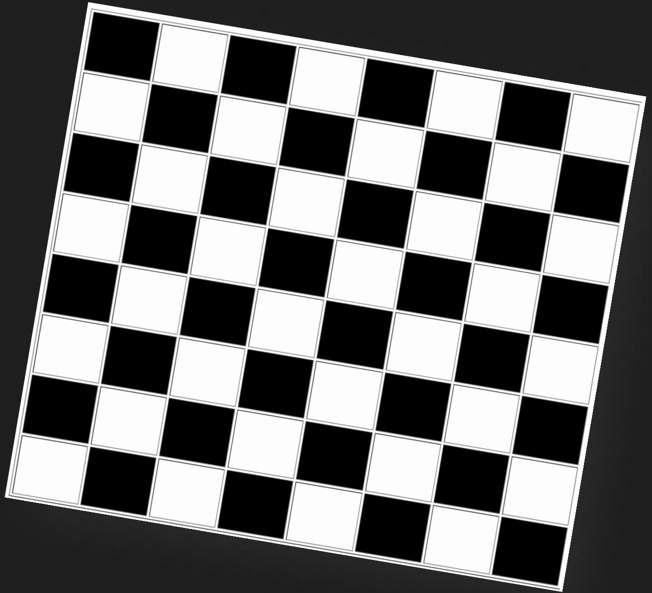


```
<script>
  rows = parseInt(prompt("Enter number of rows-"));
  cols = parseInt(prompt("Enter number of columns-"));
  document.write("<table border='1' align='center' width='50%'>");
  num = 1;
    for(r=1;r<=rows;r++)
  {
    document.write("<tr>");
    for(c=1;c<=cols;c++)
    {
      document.write(`<td align='center'>${num}</td>`);
      num++;
    }
    document.write("</tr>");
  }
  document.write("</table>");
</script>
```

For loop: ChessBoard



```
document.write("<table border='1' align='center' width='40%'>");
num = 1;
for(r=1;r<=8;r++) {
    if(num > 8)
        num = 0;
    else
        num = 1;
    document.write("<tr>");
        for(c=1;c<=8;c++) {
            if(num%2 != 0) {
                document.write("<td bgcolor=black width=50 height=50></td>");
            }
            else {
                document.write("<td bgcolor=white width=50 height=50></td>");
            }
            num++;
        }
        document.write("</tr>");
}
```

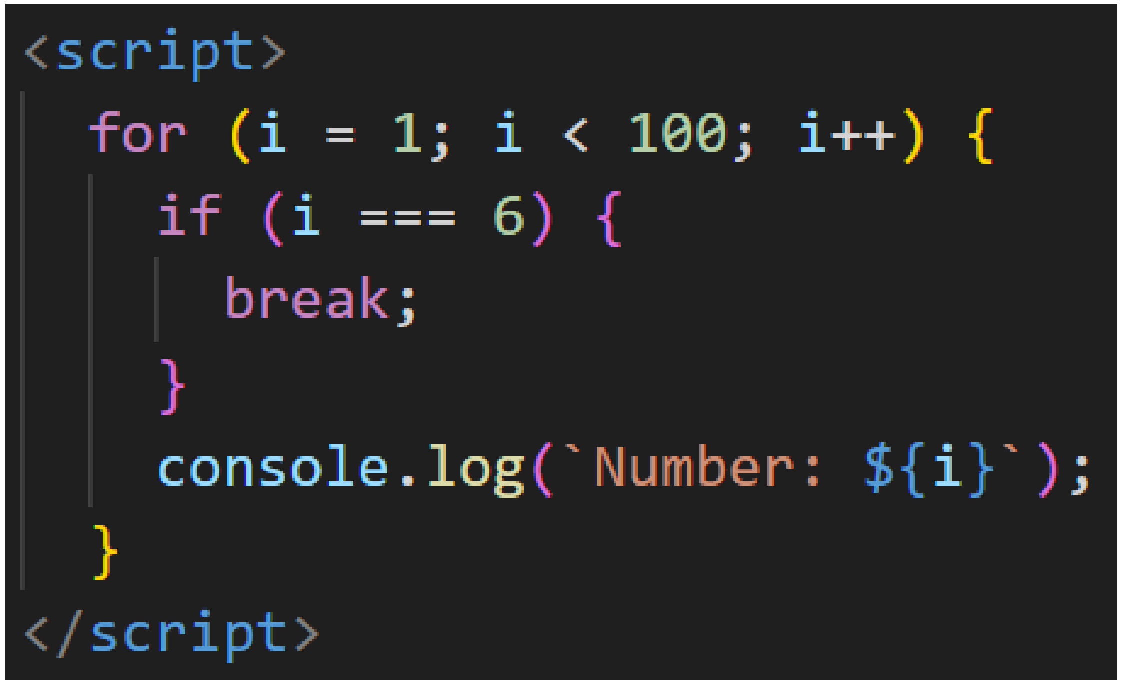


Prepared by: Amitabh Srivastava

Break Statement

- The **break statement** is used to jump out/exit from a loop.
- It breaks the loop and continues executing the code after the loop.
- It is also used to jump out from a **switch()** statement.

Example-



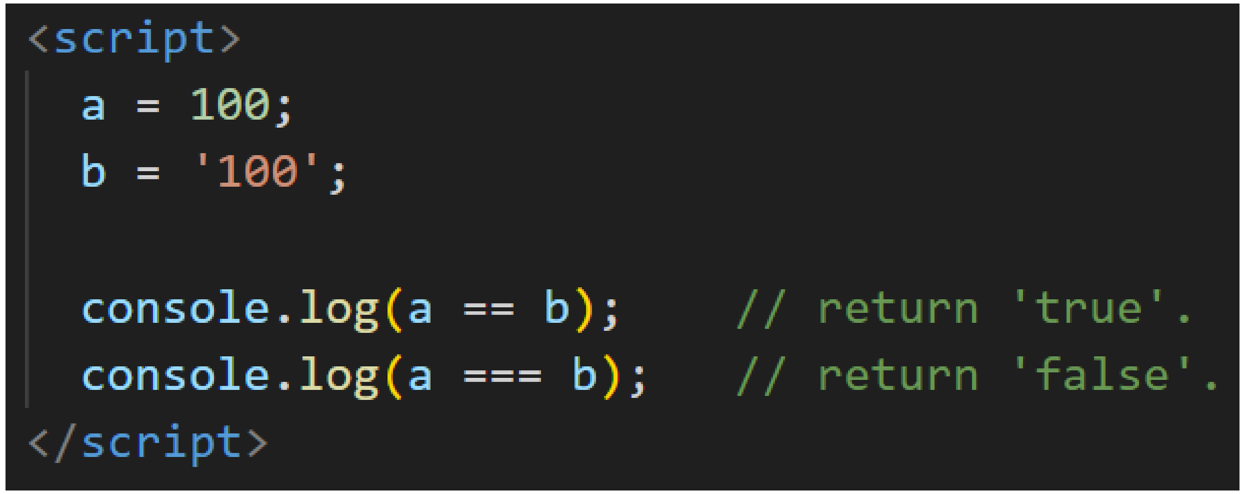
```
<script>
  for (i = 1; i < 100; i++) {
    if (i === 6) {
      break;
    }
    console.log(`Number: ${i}`);
  }
</script>
```

Notice the "===" operator used inside if condition.

== and === Operators

- **Double equals (==)** operator compares the datatypes of the operands and if they are different, then the JavaScript Engine automatically **converts one of the operands to be the same as the other one** in order to make the comparison possible.
- **Triple equals (===)** operator, while comparing the variables, first checks if the **types differ**. If they do, it **returns false**. If the **types match**, then it checks for the value. If the **values are same**, it **returns true**.

Example-

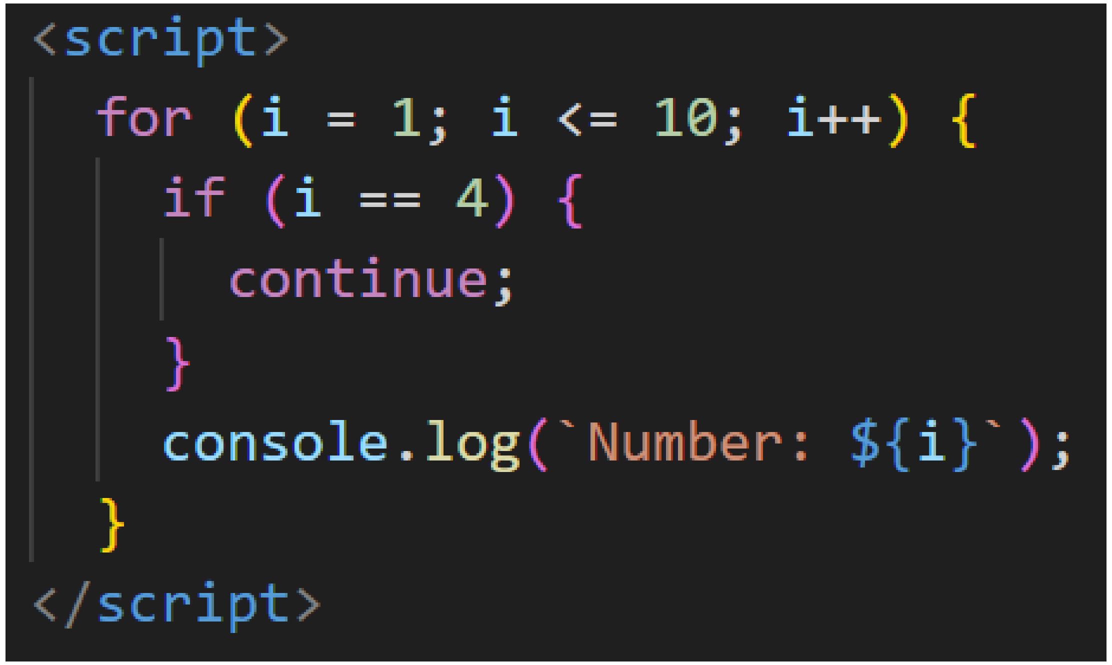


```
<script>
  a = 100;
  b = '100';

  console.log(a == b);    // return 'true'.
  console.log(a === b);   // return 'false'.
</script>
```

Continue Statement

- The **continue statement** jumps over one iteration in the loop.
- It breaks the current iteration of the loop and continues executing the next iteration in the loop.

Example-



```
<script>
  for (i = 1; i <= 10; i++) {
    if (i == 4) {
      continue;
    }
    console.log(`Number: ${i}`);
  }
</script>
```

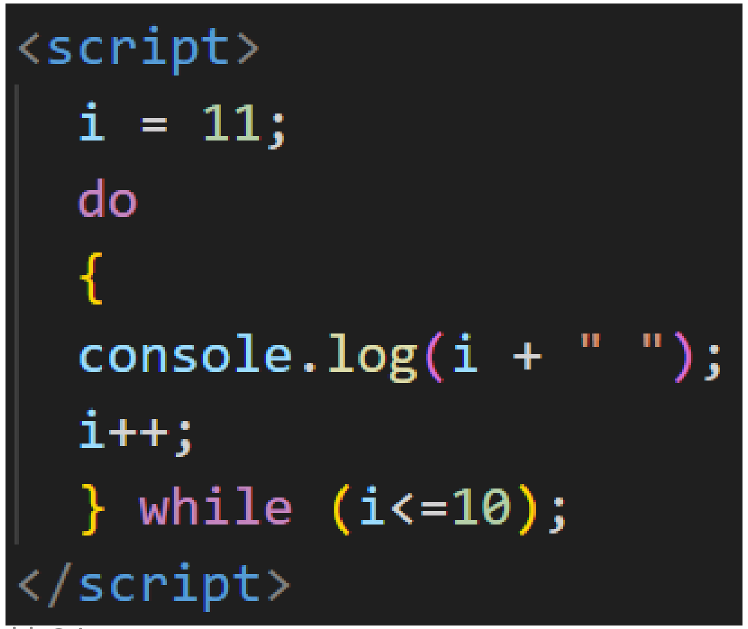
Do-while Loop

- **do-while loop** is similar to while loop with only one difference and that is, it checks for the condition after the first iteration of the loop statements.

Syntax-

```
do
{
    Statements to be executed...
}
while (condition);
```

Example-

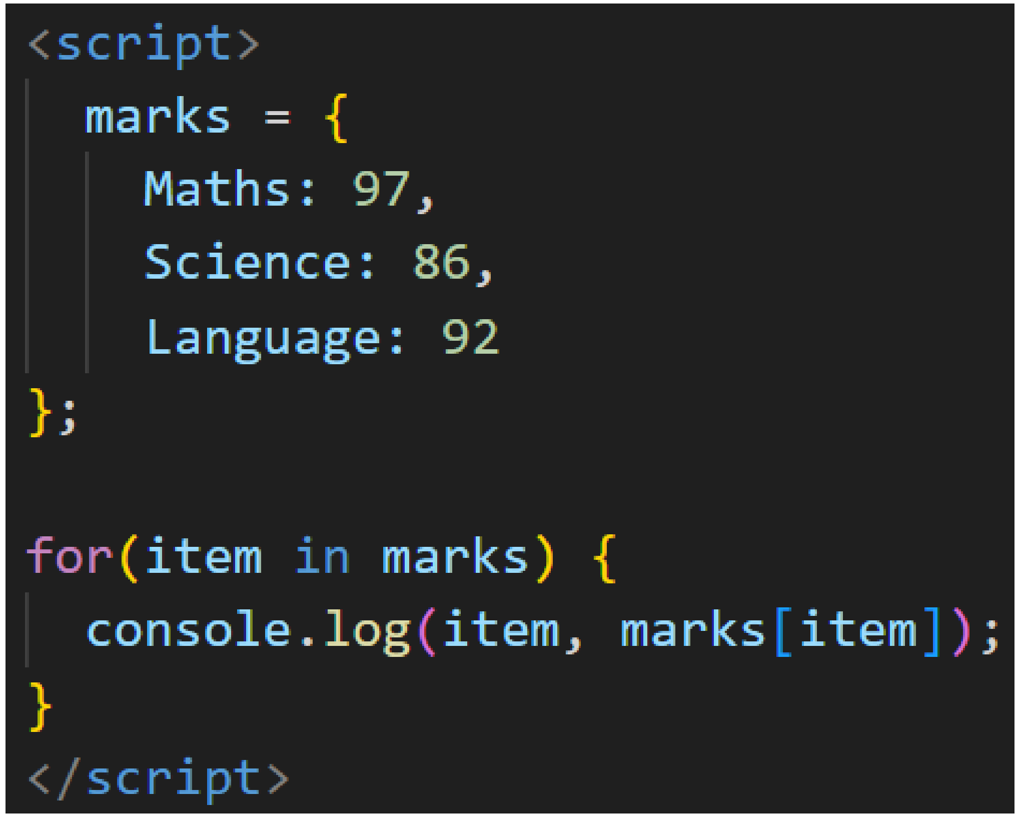


```
<script>
  i = 11;
  do
  {
  console.log(i + " ");
  i++;
  } while (i<=10);
</script>
```

For-in Loop

- For-in loop in JavaScript is used to iterate over properties of an object having values in key-value pair.

Example-



```
<script>
  marks = {
    Maths: 97,
    Science: 86,
    Language: 92
};

for(item in marks) {
  console.log(item, marks[item]);
}
</script>
```

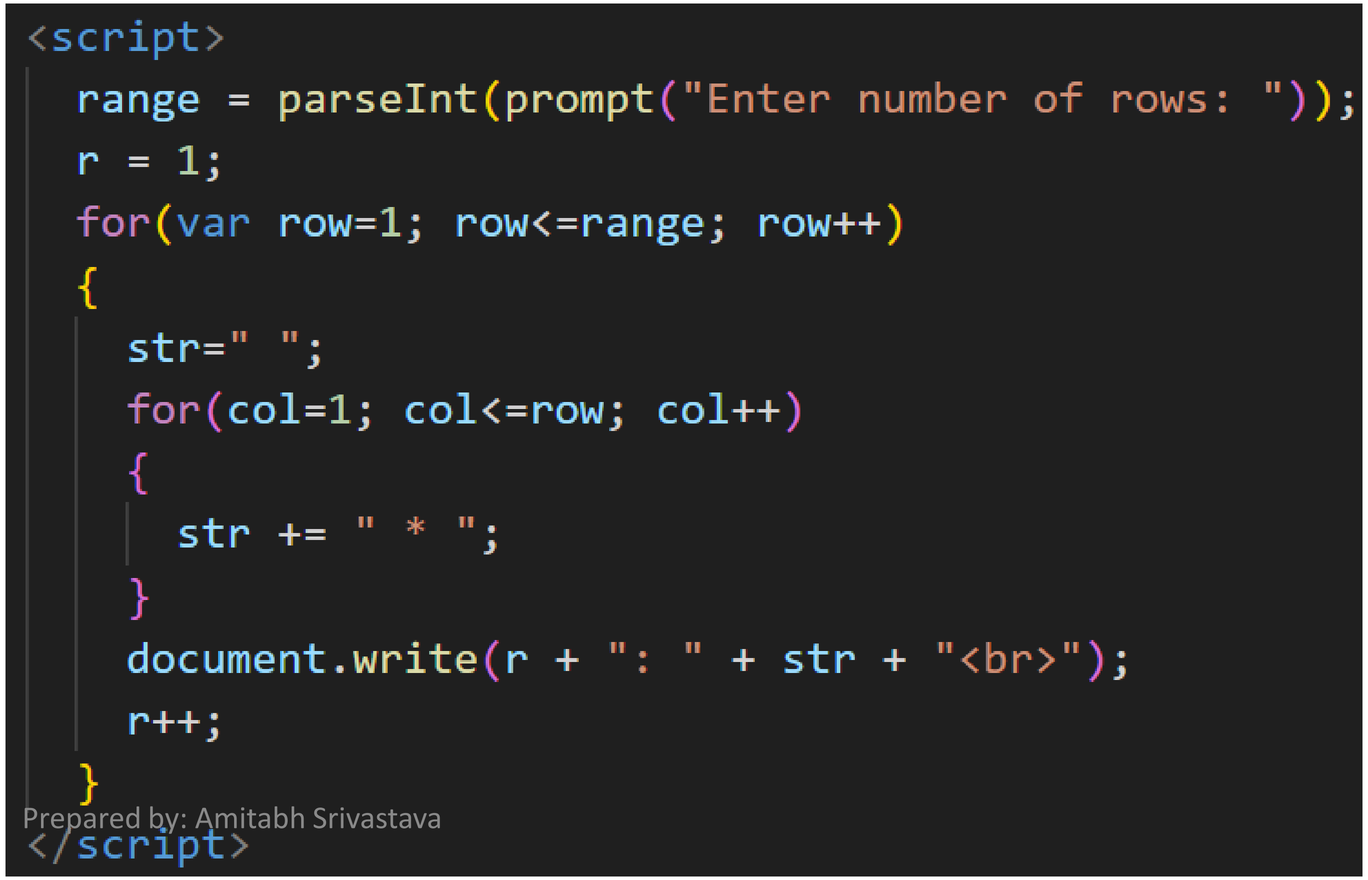
Loops: Practice-1

Write a program to print the following star pattern after asking the no. of rows from the user? Also print the row number at the beginning of each row.

Enter number of rows: 7

```
1: *
2: * *
3: * * *
4: * * * *
5: * * * * *
6: * * * * * *
7: * * * * * * *
```

Solution-



```
<script>
  range = parseInt(prompt("Enter number of rows: "));
  r = 1;
  for(var row=1; row<=range; row++)
  {
    str=" ";
    for(col=1; col<=row; col++)
    {
      str += " * ";
    }
    document.write(r + ": " + str + "<br>");
    r++;
  }
</script>
```

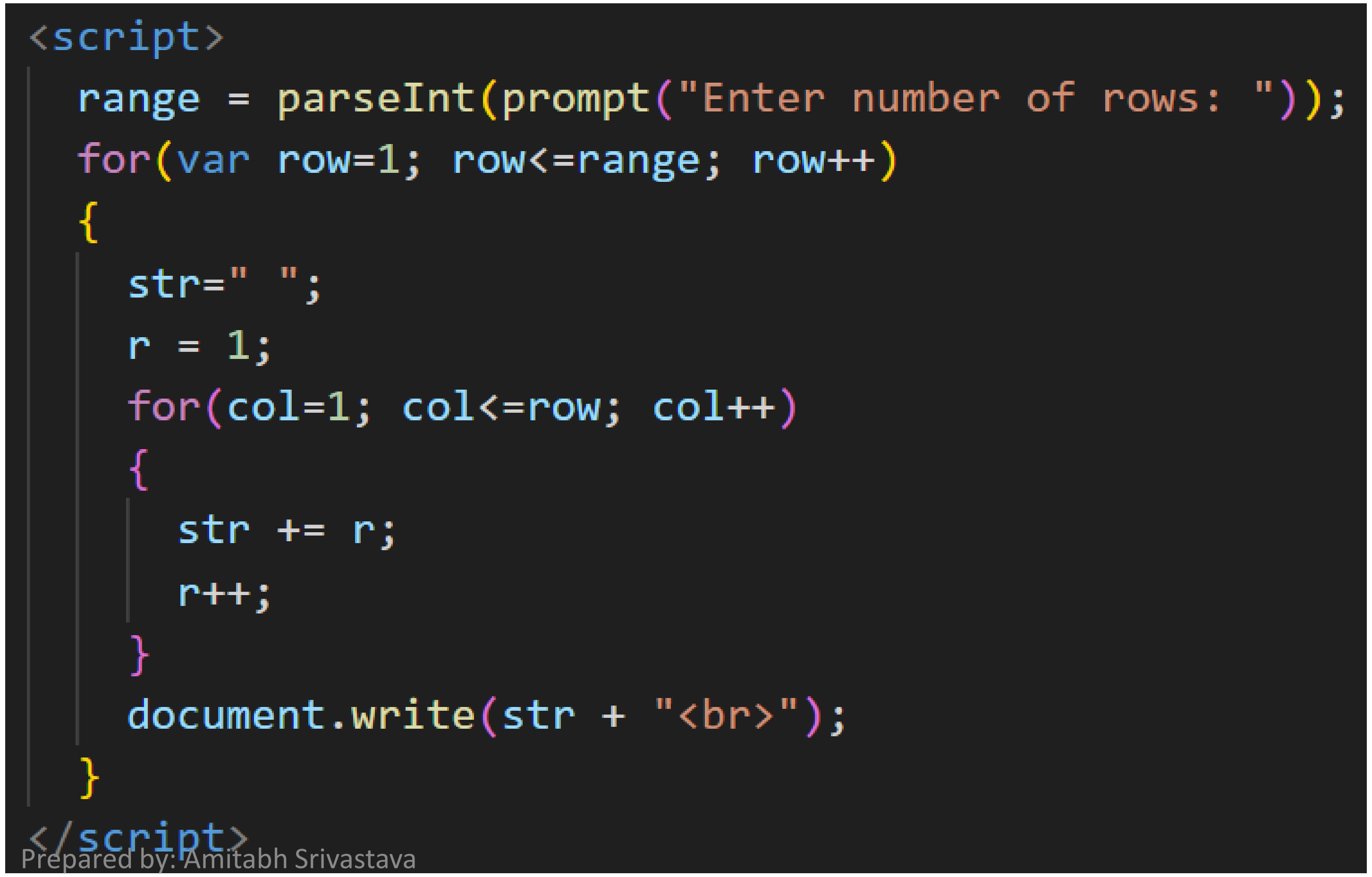
Loops: Practice-2

Write a program to print the following number's pattern after asking the no. of rows from the user?

Enter number of rows: 7

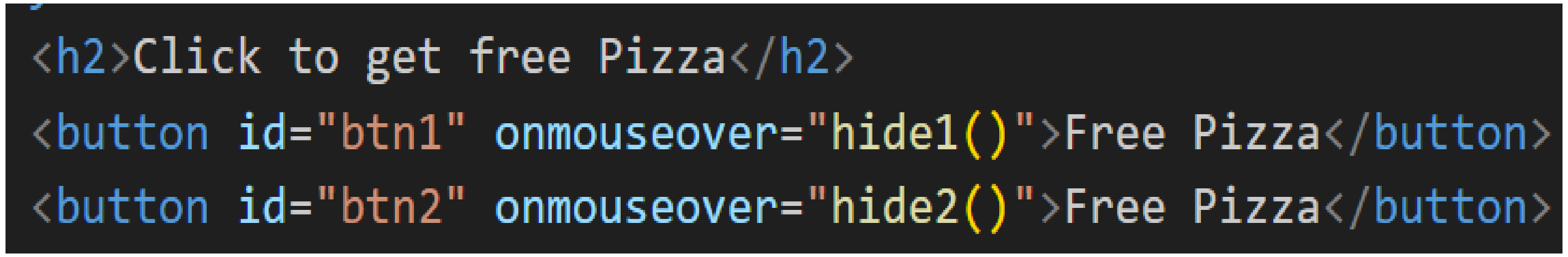
```
1
12
123
1234
12345
123456
1234567
```

Solution-

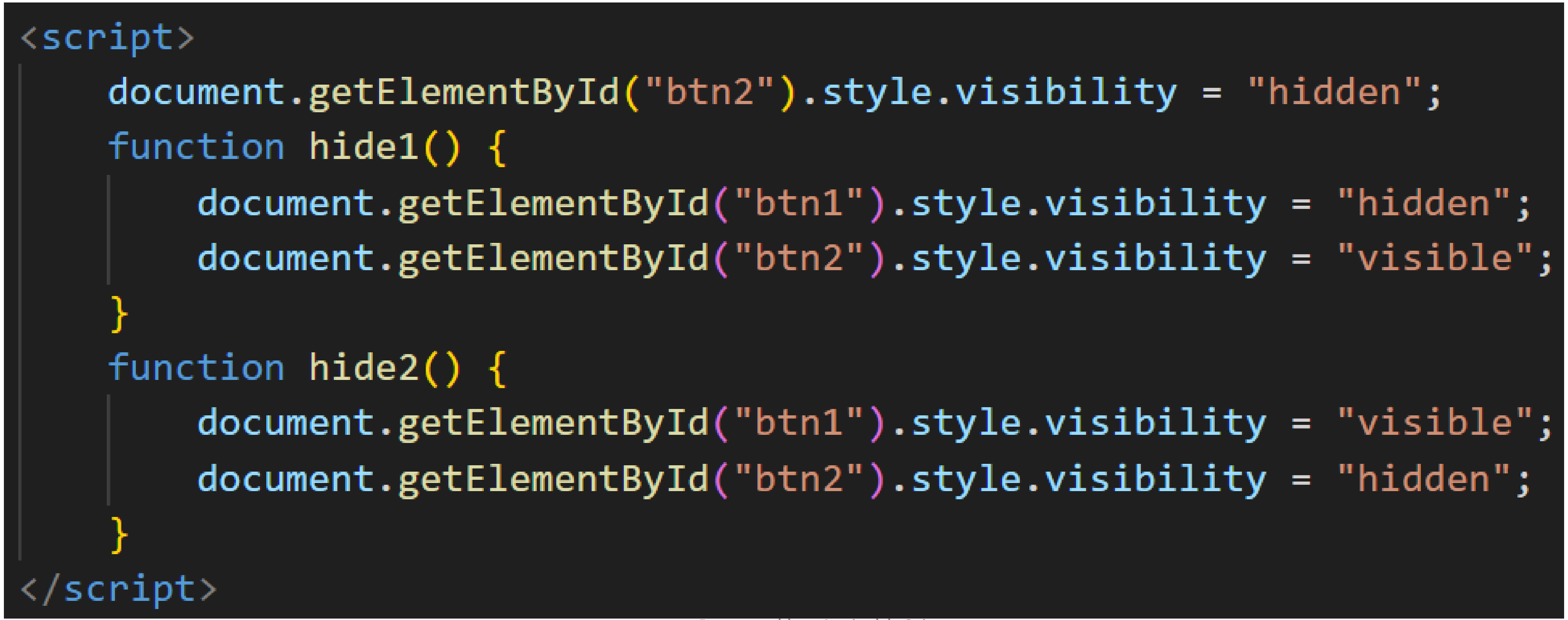


```
<script>
  range = parseInt(prompt("Enter number of rows: "));
  for(var row=1; row<=range; row++)
  {
    str=" ";
    r = 1;
    for(col=1; col<=row; col++)
    {
      str += r;
      r++;
    }
    document.write(str + "<br>");
  }
</script>
```

Style using JavaScript: Example-1

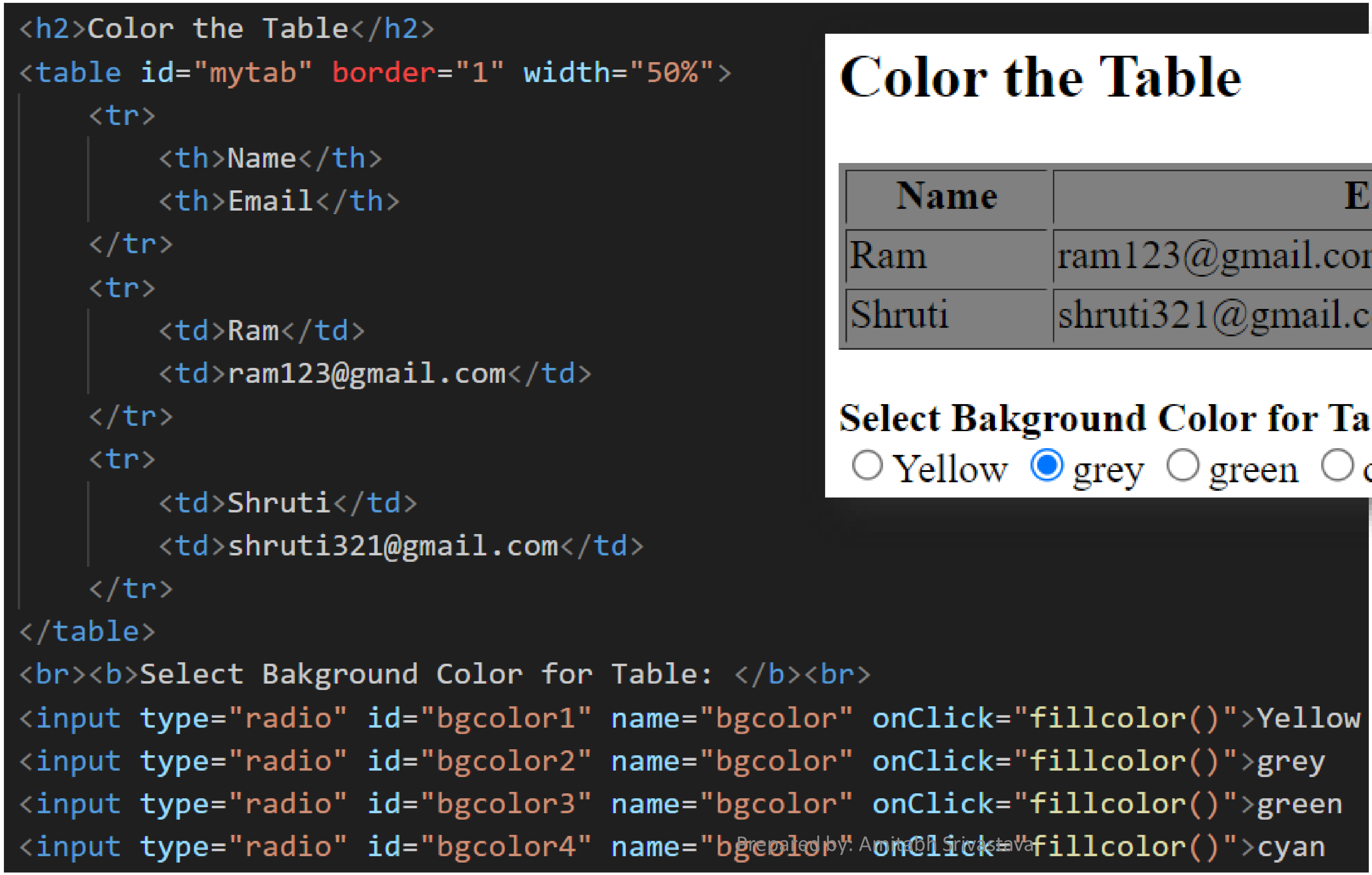


```html
<h2>Click to get free Pizza</h2>
<button id="btn1" onmouseover="hide1()">Free Pizza</button>
<button id="btn2" onmouseover="hide2()">Free Pizza</button>
```

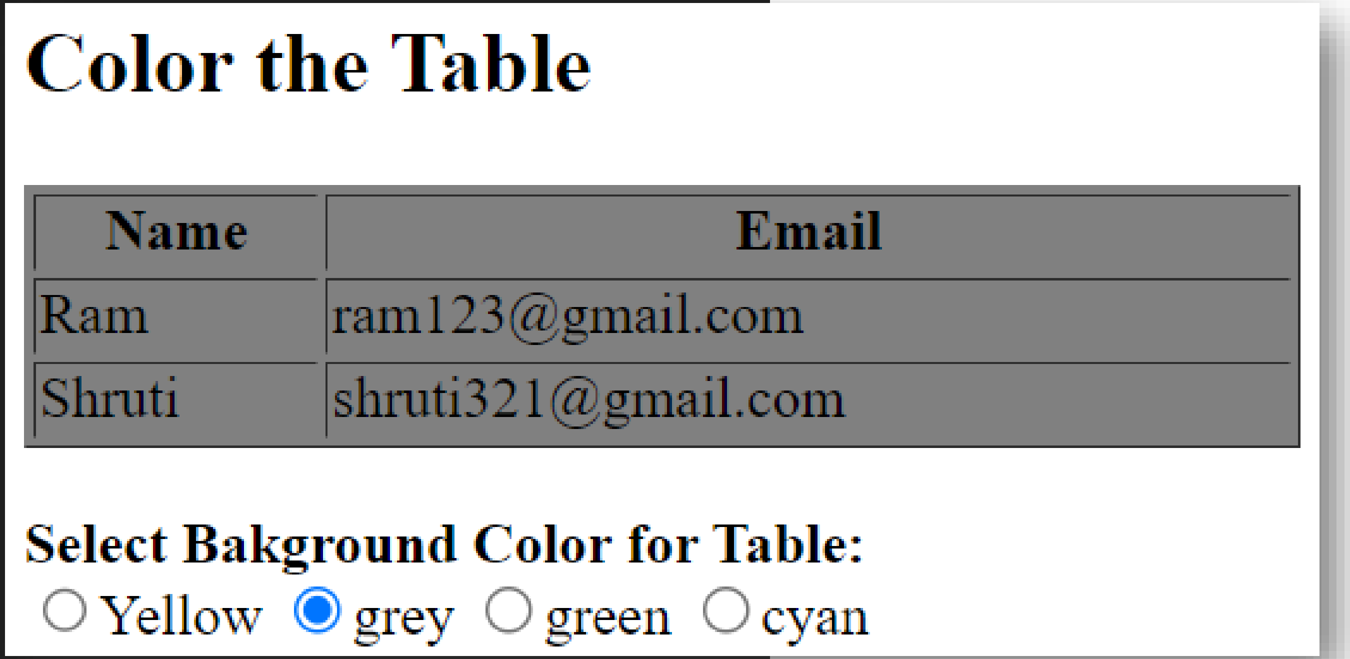


```html
<script>
    document.getElementById("btn2").style.visibility = "hidden";
    function hide1() {
        document.getElementById("btn1").style.visibility = "hidden";
        document.getElementById("btn2").style.visibility = "visible";
    }
    function hide2() {
        document.getElementById("btn1").style.visibility = "visible";
        document.getElementById("btn2").style.visibility = "hidden";
    }
</script>
```

Style using JavaScript: Example-2



```
<h2>Color the Table</h2>
<table id="mytab" border="1" width="50%">
    <tr>
        <th>Name</th>
        <th>Email</th>
    </tr>
    <tr>
        <td>Ram</td>
        <td>ram123@gmail.com</td>
    </tr>
    <tr>
        <td>Shruti</td>
        <td>shruti321@gmail.com</td>
    </tr>
</table>
<br><b>Select Bakground Color for Table: </b><br>
<input type="radio" id="bgcolor1" name="bgcolor" onClick="fillcolor()">Yellow
<input type="radio" id="bgcolor2" name="bgcolor" onClick="fillcolor()">grey
<input type="radio" id="bgcolor3" name="bgcolor" onClick="fillcolor()">green
<input type="radio" id="bgcolor4" name="bgcolor" onClick="fillcolor()">cyan
```

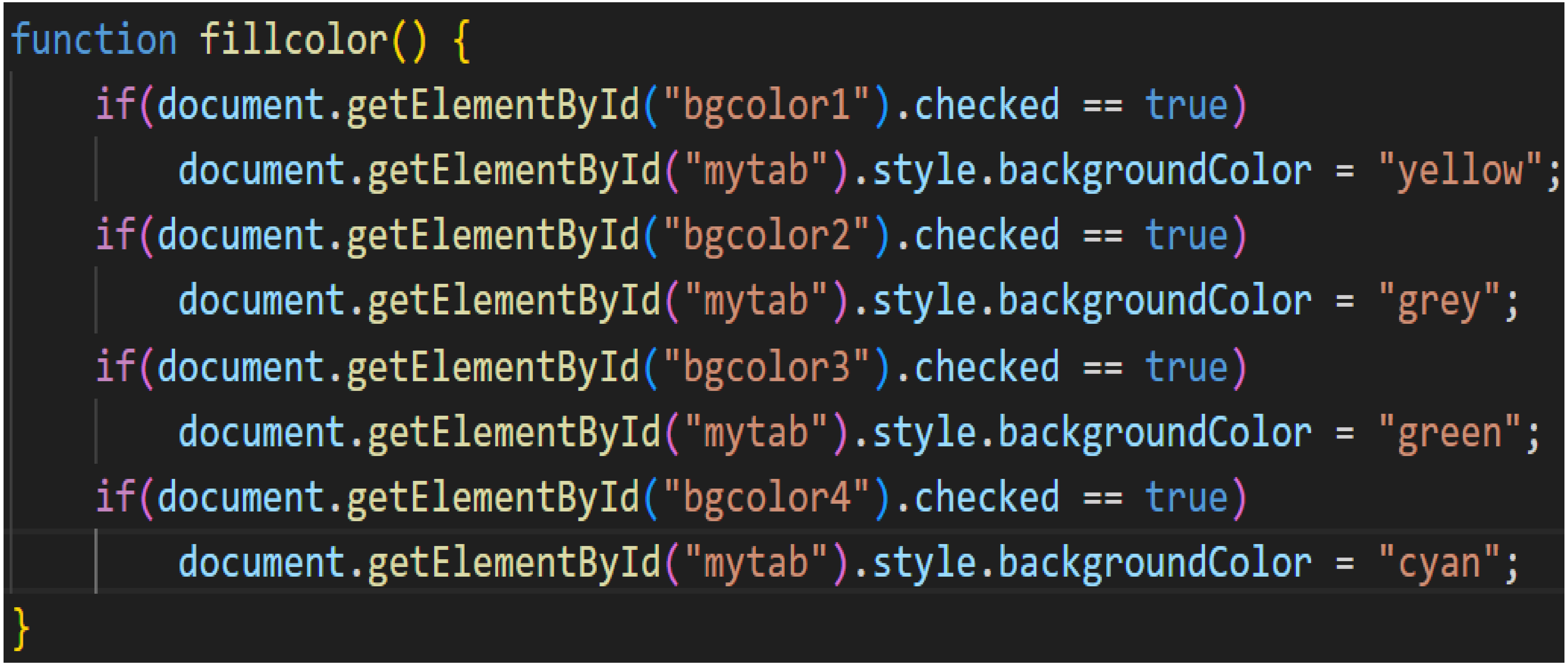


Color the Table

Name	Email
Ram	ram123@gmail.com
Shruti	shruti321@gmail.com

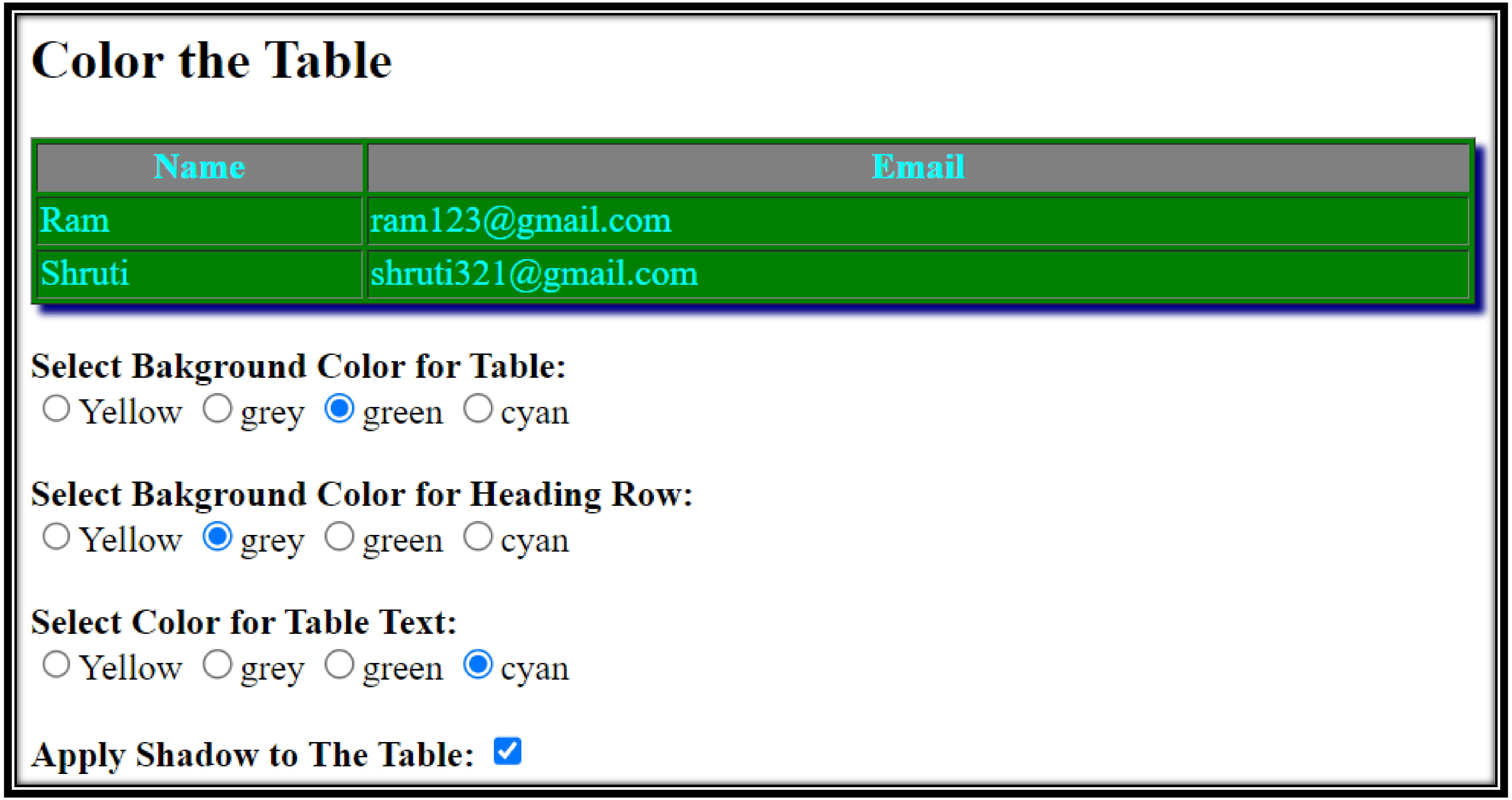
Select Bakground Color for Table:
○ Yellow [✓ grey] ○ green ○ cyan

Example-2...cont.



```
function fillcolor() {
    if(document.getElementById("bgcolor1").checked == true)
        document.getElementById("mytab").style.backgroundColor = "yellow";
    if(document.getElementById("bgcolor2").checked == true)
        document.getElementById("mytab").style.backgroundColor = "grey";
    if(document.getElementById("bgcolor3").checked == true)
        document.getElementById("mytab").style.backgroundColor = "green";
    if(document.getElementById("bgcolor4").checked == true)
        document.getElementById("mytab").style.backgroundColor = "cyan";
}
```

Do It Yourself - 1



Color the Table

Name	Email
Ram	ram123@gmail.com
Shruti	shruti321@gmail.com

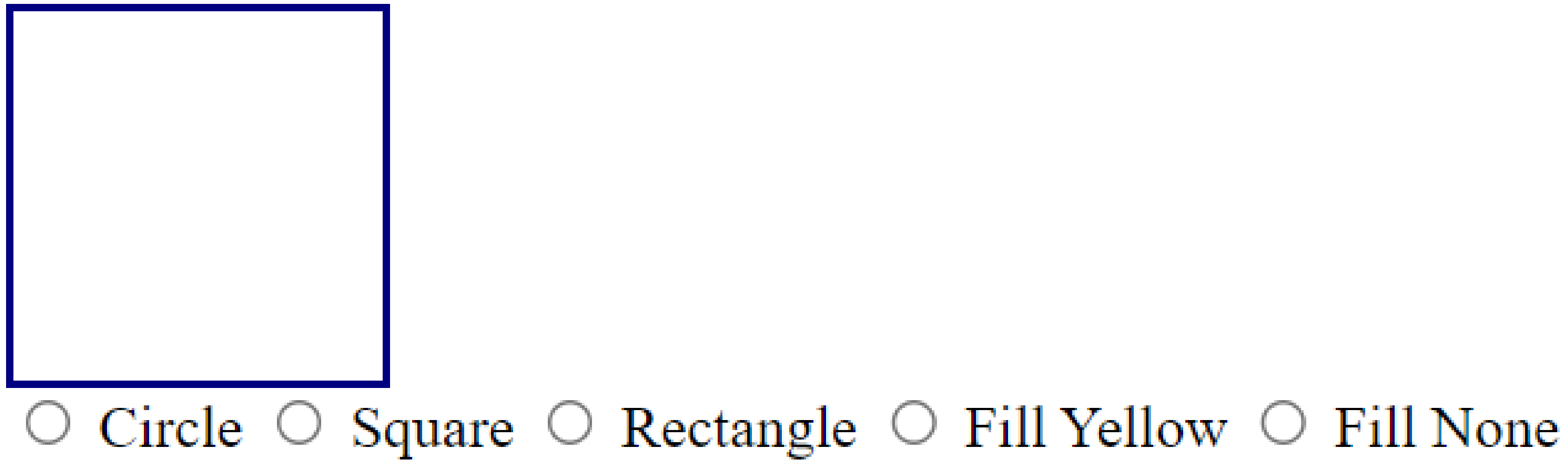
Select Bakground Color for Table:
○ Yellow ○ grey [✓ green] ○ cyan

Select Bakground Color for Heading Row:
○ Yellow [✓ grey] ○ green ○ cyan

Select Color for Table Text:
○ Yellow ○ grey ○ green [✓ cyan]

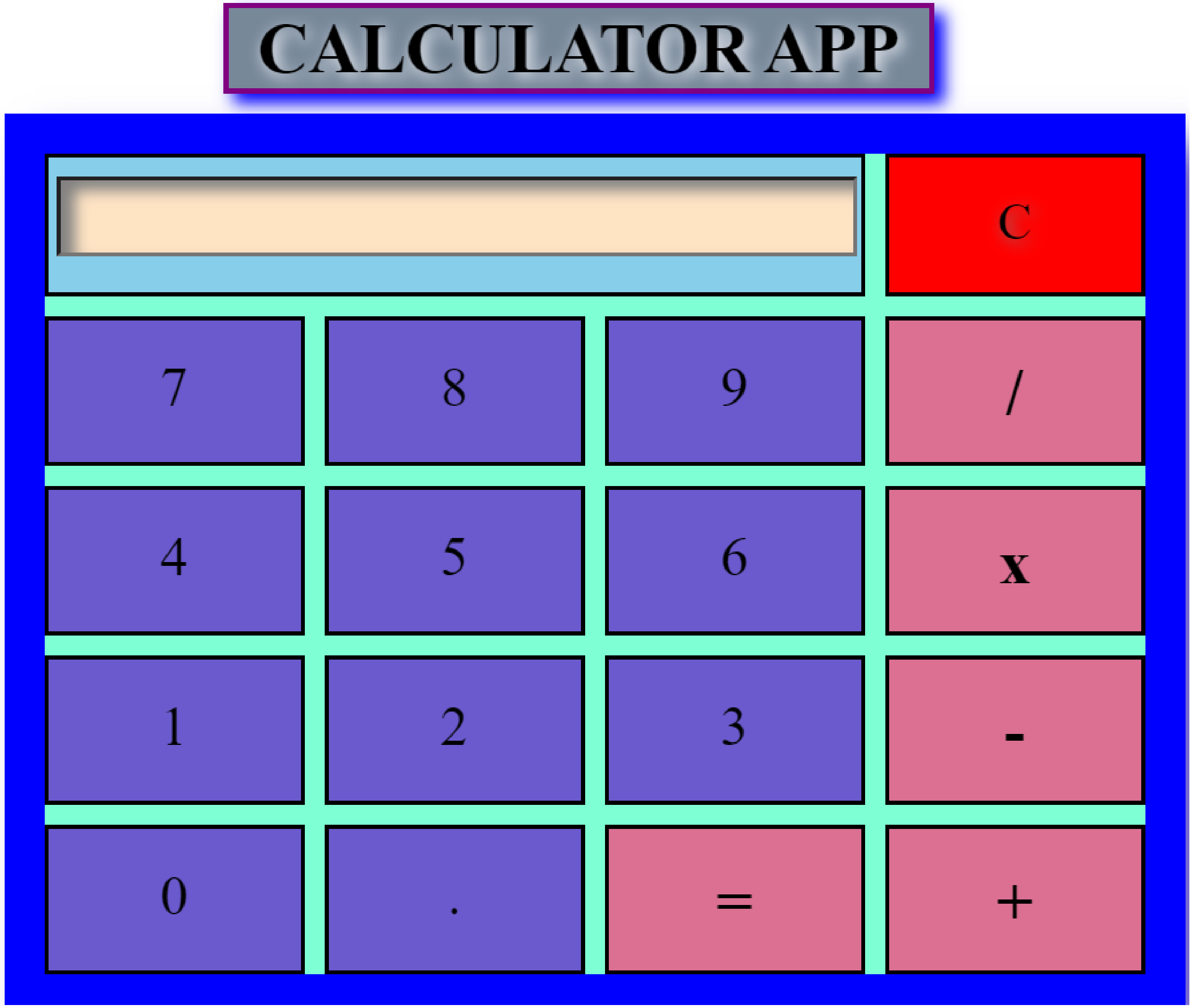
Apply Shadow to The Table: ✓

Do It Yourself - 2

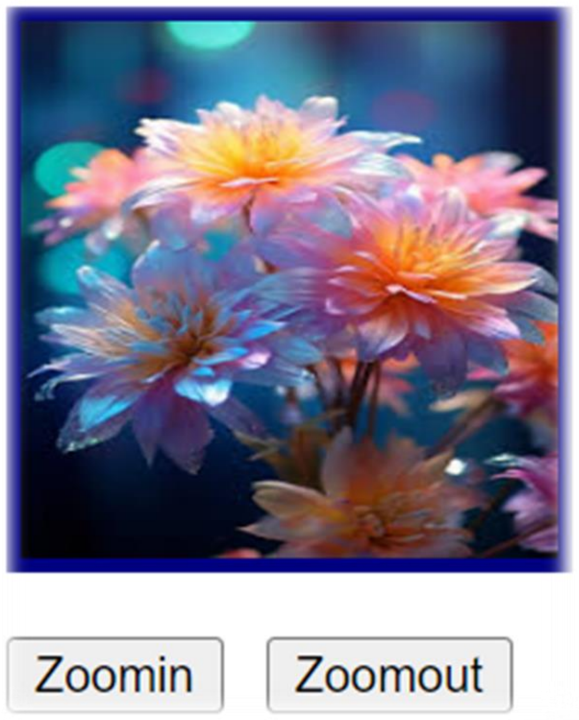


○ Circle ○ Square ○ Rectangle ○ Fill Yellow ○ Fill None

Do It Yourself - 3



Do It Yourself - 4



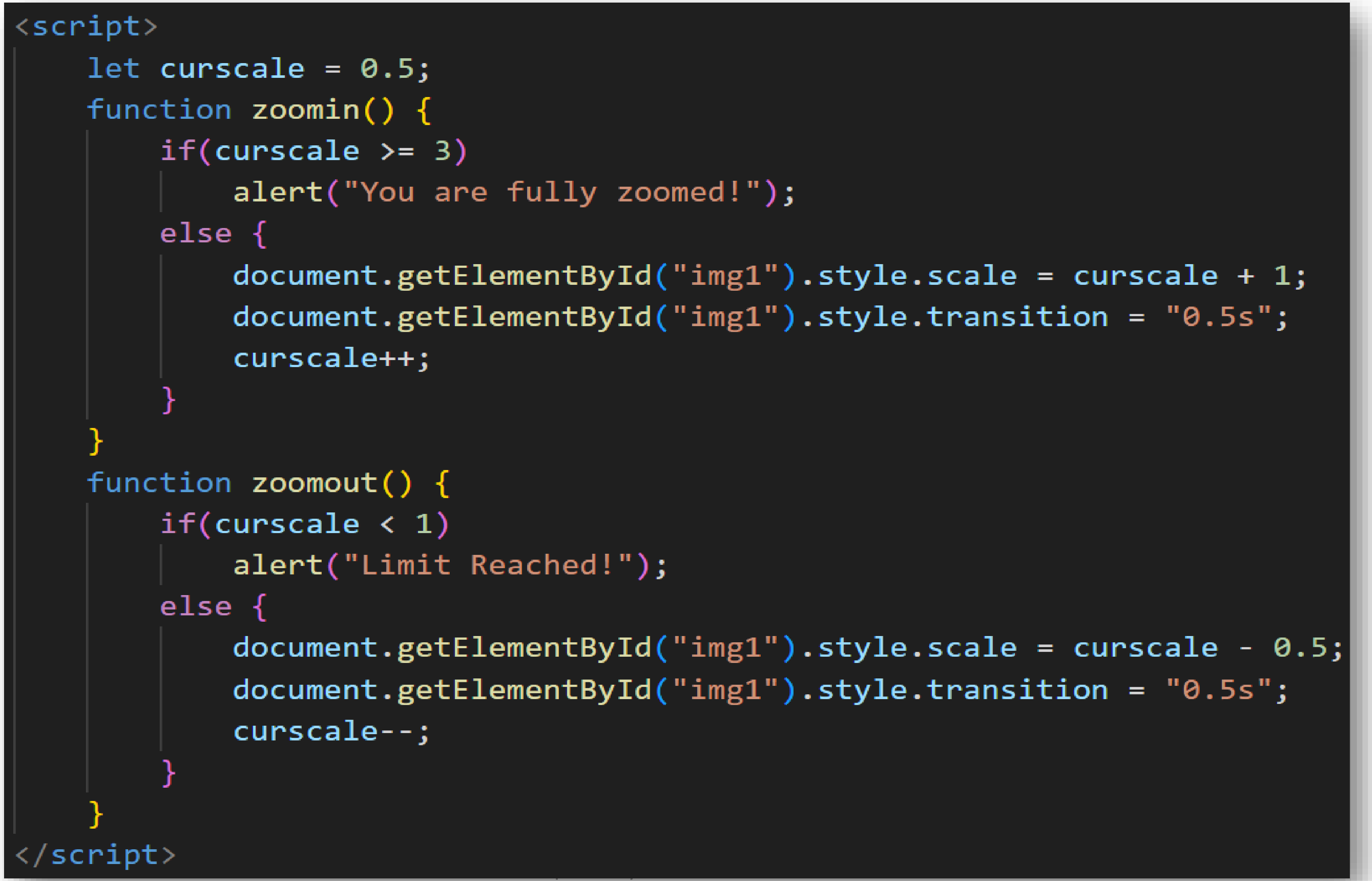
HTML-

```
<body>
<div>
    <img id="img1" src="flower1.jpg" height="150" width="150">
</div>
<br><br>
<button onclick="zoomin()">
    Zoomin </button> &nbsp;
<button onclick="zoomout()">
    Zoomout </button>
</body>
```

CSS-

```
<style>
 div {
    display: inline-flex;
    overflow: hidden;
    border: 4px solid navy;
 }
</style>
```

Do It Yourself – 4 (JS)



```
<script>
    let curscale = 0.5;
    function zoomin() {
        if(curscale >= 3)
            alert("You are fully zoomed!");
        else {
            document.getElementById("img1").style.scale = curscale + 1;
            document.getElementById("img1").style.transition = "0.5s";
            curscale++;
        }
    }
    function zoomout() {
        if(curscale < 1)
            alert("Limit Reached!");
        else {
            document.getElementById("img1").style.scale = curscale - 0.5;
            document.getElementById("img1").style.transition = "0.5s";
            curscale--;
        }
    }
</script>
```

Methods of String Object

- **charAt()-** returns character at given index.

```
var str = new String("JavaScript");
document.write("<b>Char At:</b> " + str.charAt(4));
```

- **charCodeAt()-** returns the ASCII value of the character at given index.

```
document.write("<b>CharCode At:</b> " + str.charCodeAt(2)+"<br>");
```

- **indexOf()-** returns the index of given character. If not found, returns **-1**.

```
 document.write("<b>Index of:</b> " + str.indexOf("Z")+"<br>");
```

- **toLowerCase()-** converts the string to lowercase.

```
document.write("<b>Lower Case:</b> " + str.toLowerCase()+"<br>");
```

- **toUpperCase()-** converts the string to uppercase.

```
document.write("<b>Upper Case:</b> " + str.toUpperCase()+"<br>");
```

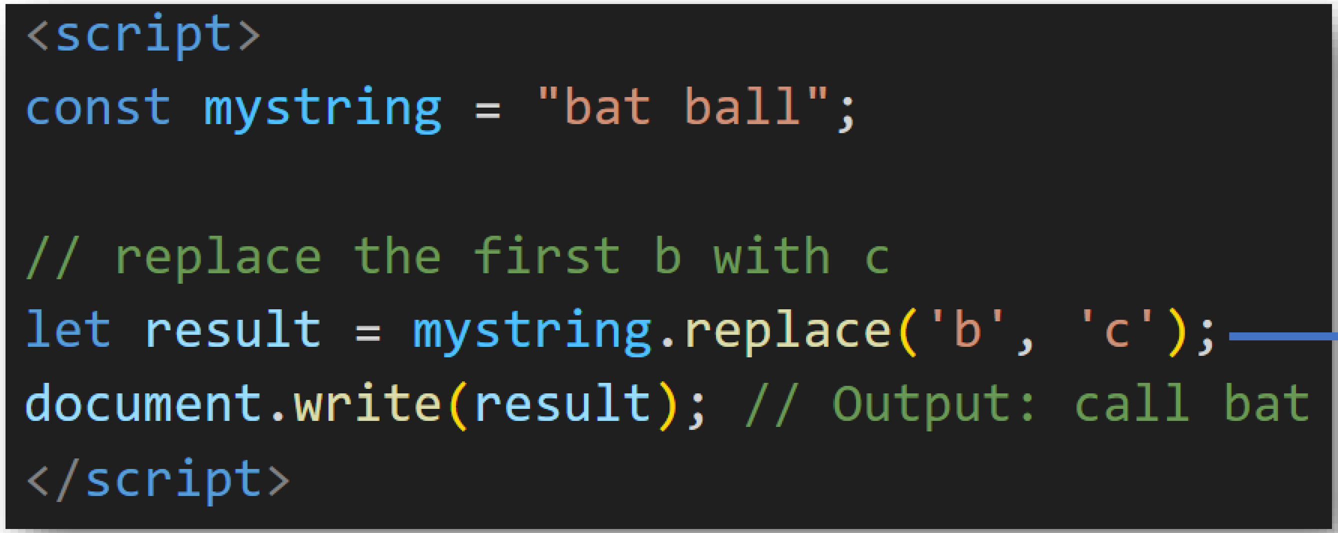
Replace()

- The **replace()** method returns a new string with the specified string/regex replaced.

Syntax-

source_string.replace(original_string, new_string);

Example-

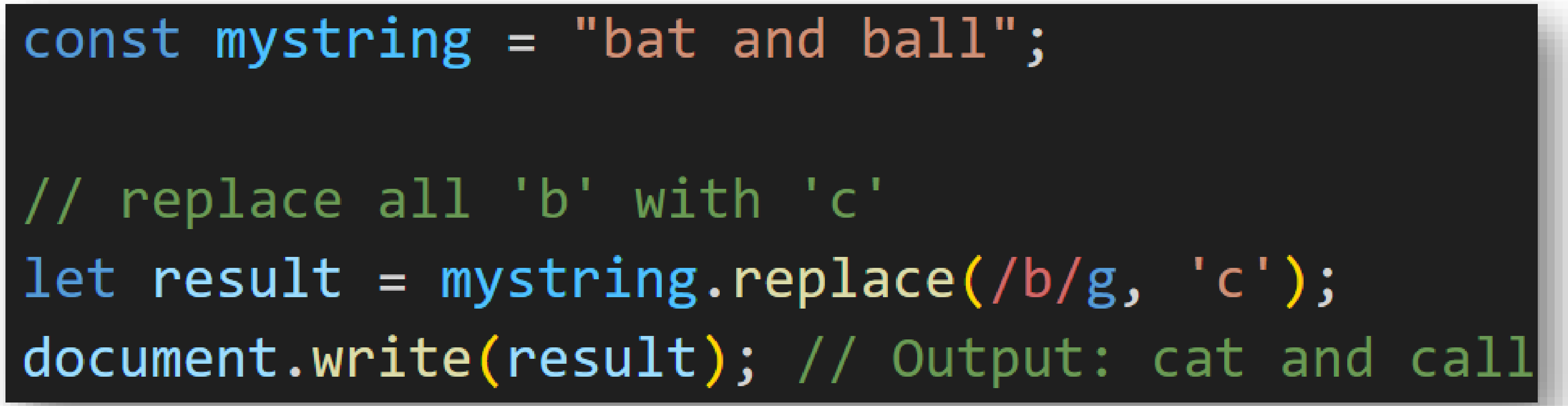


```
<script>
const mystring = "bat ball";

// replace the first b with c
let result = mystring.replace('b', 'c');
document.write(result); // Output: call bat
</script>
```

replaces only 1st occurrence.

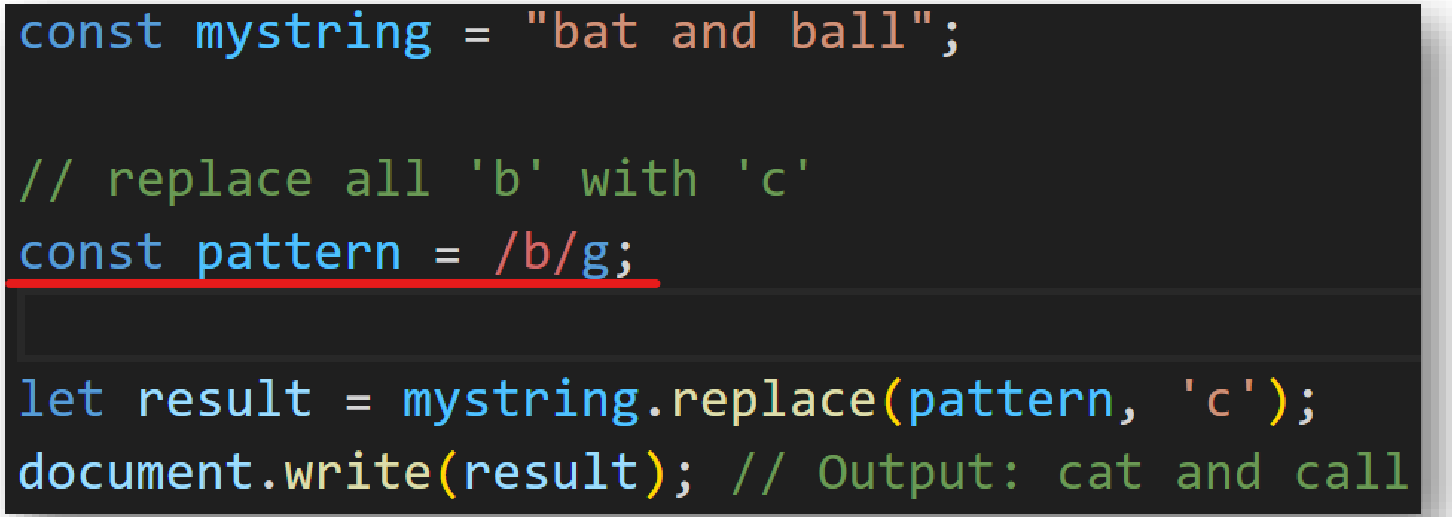
Replace()- all occurrences 'g'

Example:1-



```
const mystring = "bat and ball";

// replace all 'b' with 'c'
let result = mystring.replace(/b/g, 'c');
document.write(result); // Output: cat and call
```

Example:2-

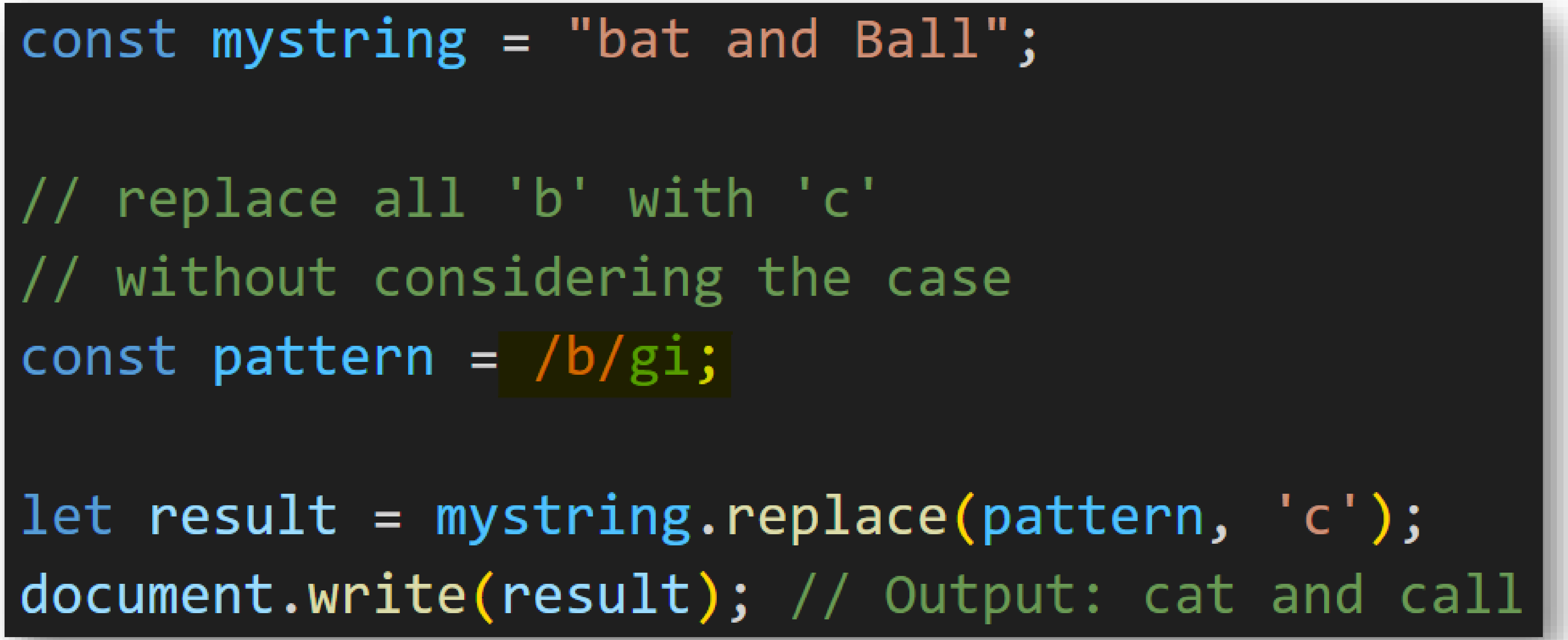


```
const mystring = "bat and ball";

// replace all 'b' with 'c'
const pattern = /b/g;

let result = mystring.replace(pattern, 'c');
document.write(result); // Output: cat and call
```

Replace()- ignore case ‘i’

- **replace()** is always case-sensitive. To ignore case, use ‘i’ option in Regex pattern.

Example:1-



```
const mystring = "bat and Ball";

// replace all 'b' with 'c'
// without considering the case
const pattern = /b/gi;

let result = mystring.replace(pattern, 'c');
document.write(result); // Output: cat and call
```

Functions

- A function is a set of statements that take inputs, do some specific computation, and produces output.
- Instead of writing the same code again and again at different places, we can write such repeated statements inside a function and call that function.
- In JavaScript there are built-in functions as well as user-defined functions.
- To create/define a function, the keyword **function** is used.

Syntax-

```
function function_Name(Parameter1, Parameter2, ..)
{
 statement...1;
 statement...2;
 statement...N;
 [return statement / value;]  // optional
}
```

Calling A Function

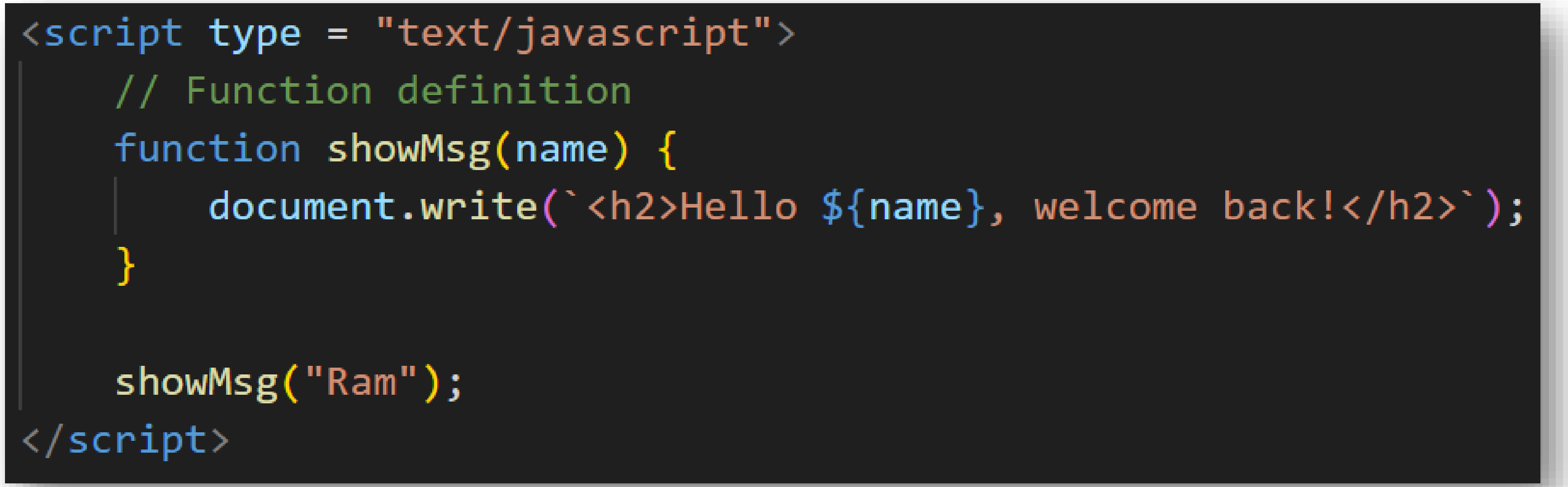
- After defining a function, the next step is to call it to make use of the defined function.
- We can call a function by using the function name separated by the value of parameters enclosed between parenthesis and a semicolon at the end.

Syntax-

functionName(Value1, Value2, ..);

Remember- The number of parameters in defined function and in calling function must be same.

Example-

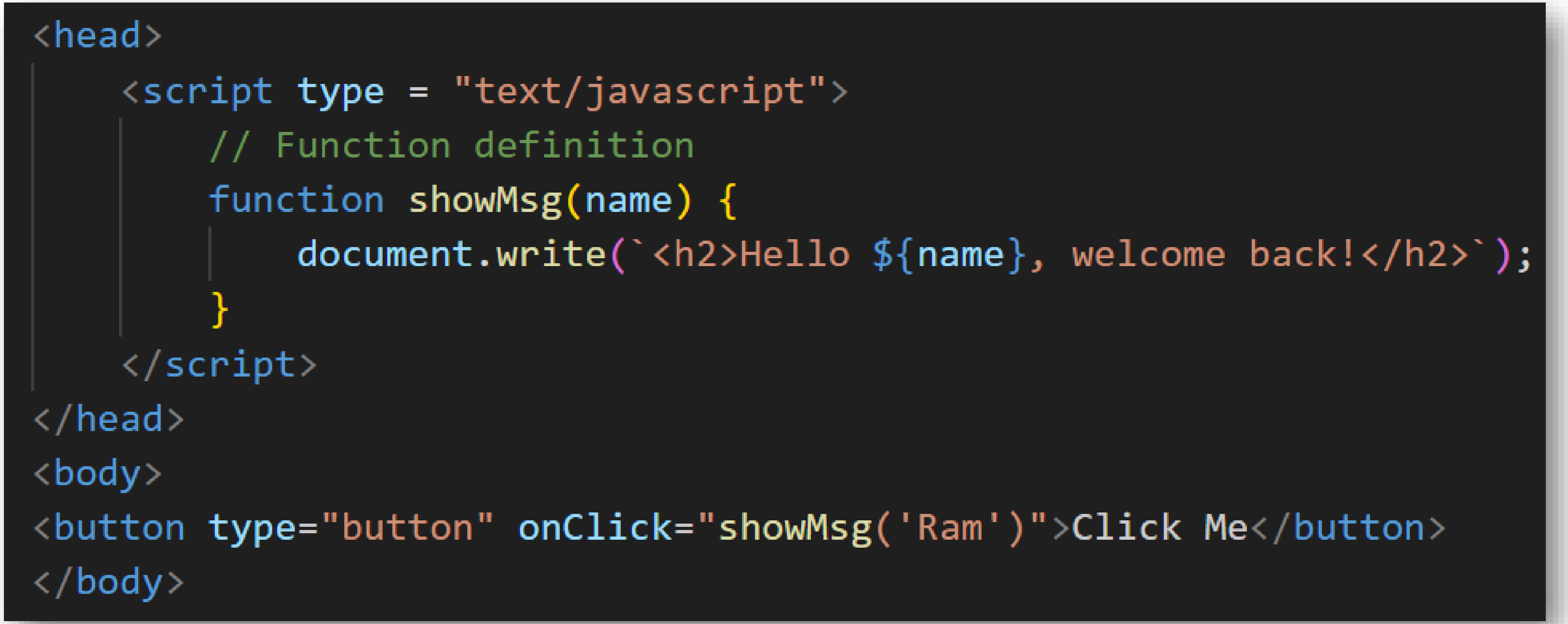


```
<script type = "text/javascript">
    // Function definition
    function showMsg(name) {
        document.write(`<h2>Hello ${name}, welcome back!</h2>`);
    }

    showMsg("Ram");
</script>
```

Calling Function on Event

- We can also do that when user clicks on a button only then a specific function should run.

Example-

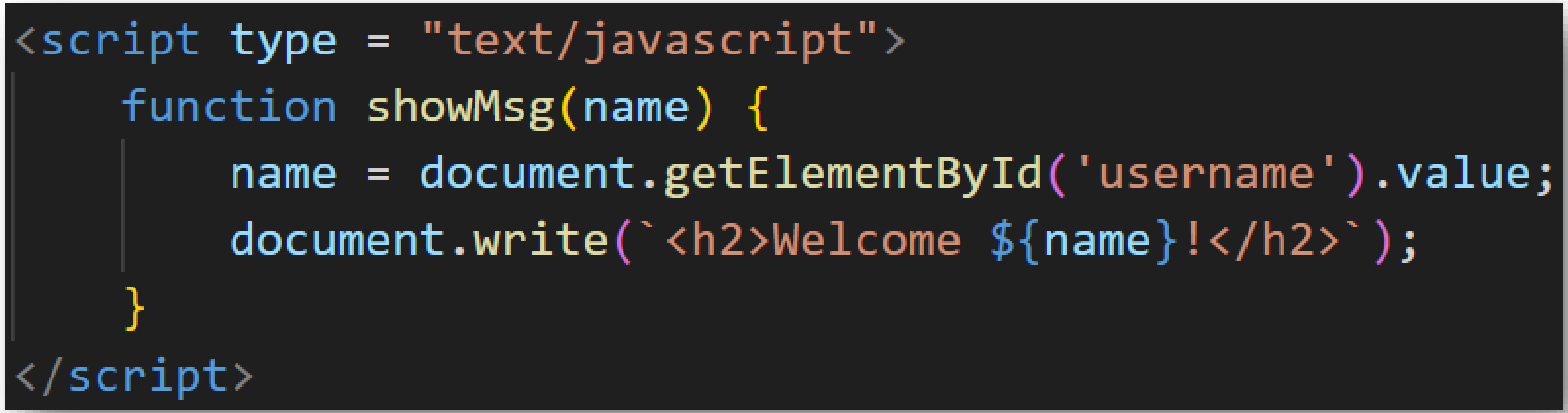


```
<head>
    <script type = "text/javascript">
        // Function definition
        function showMsg(name) {
            document.write(`<h2>Hello ${name}, welcome back!</h2>`);
        }
    </script>
</head>
<body>
<button type="button" onClick="showMsg('Ram')">Click Me</button>
</body>
```

getElementById()

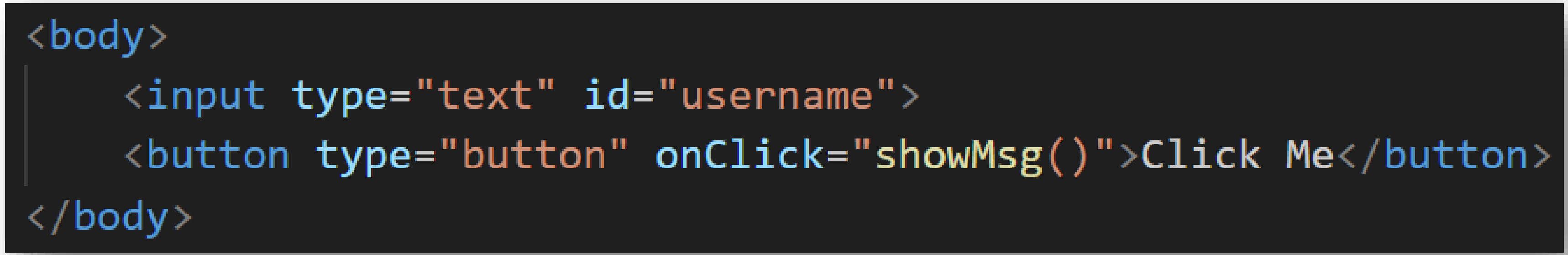
Scenario-

- An HTML has a text field and we want that when visitor fill its name in that text field and clicks on the button, only then **showMsg()** function should call with a welcome message containing the entered name by the user/visitor.



```
<script type = "text/javascript">
    function showMsg(name) {
        name = document.getElementById('username').value;
        document.write(`<h2>Welcome ${name}!</h2>`);
    }
</script>
```

- We need to pass the value entered in a text field from HTML form to JavaScript function.
- JavaScript has **getElementById()** method to collect the value of specific form element.



```
<body>
    <input type="text" id="username">
    <button type="button" onClick="showMsg()">Click Me</button>
</body>
```

JavaScript Events

- The change in the state of an object is known as an **Event**.

- When JavaScript code is included in HTML, it reacts over these events and allow us to execute required code or set of statements.

- This process of reacting over the events is called **Event Handling**.

- JavaScript handles the HTML events via **Event Handlers**.

- Events are part of the **Document Object Model (DOM)** and every HTML element contains a set of events which can trigger JavaScript Code.

- **For example**, when user clicks on an object in a browser, the **“OnClick”** Event Handler is called. So you can easily add some code or set of codes to this **“OnClick”** event which you want to execute when this event occurs.

JavaScript Event Types

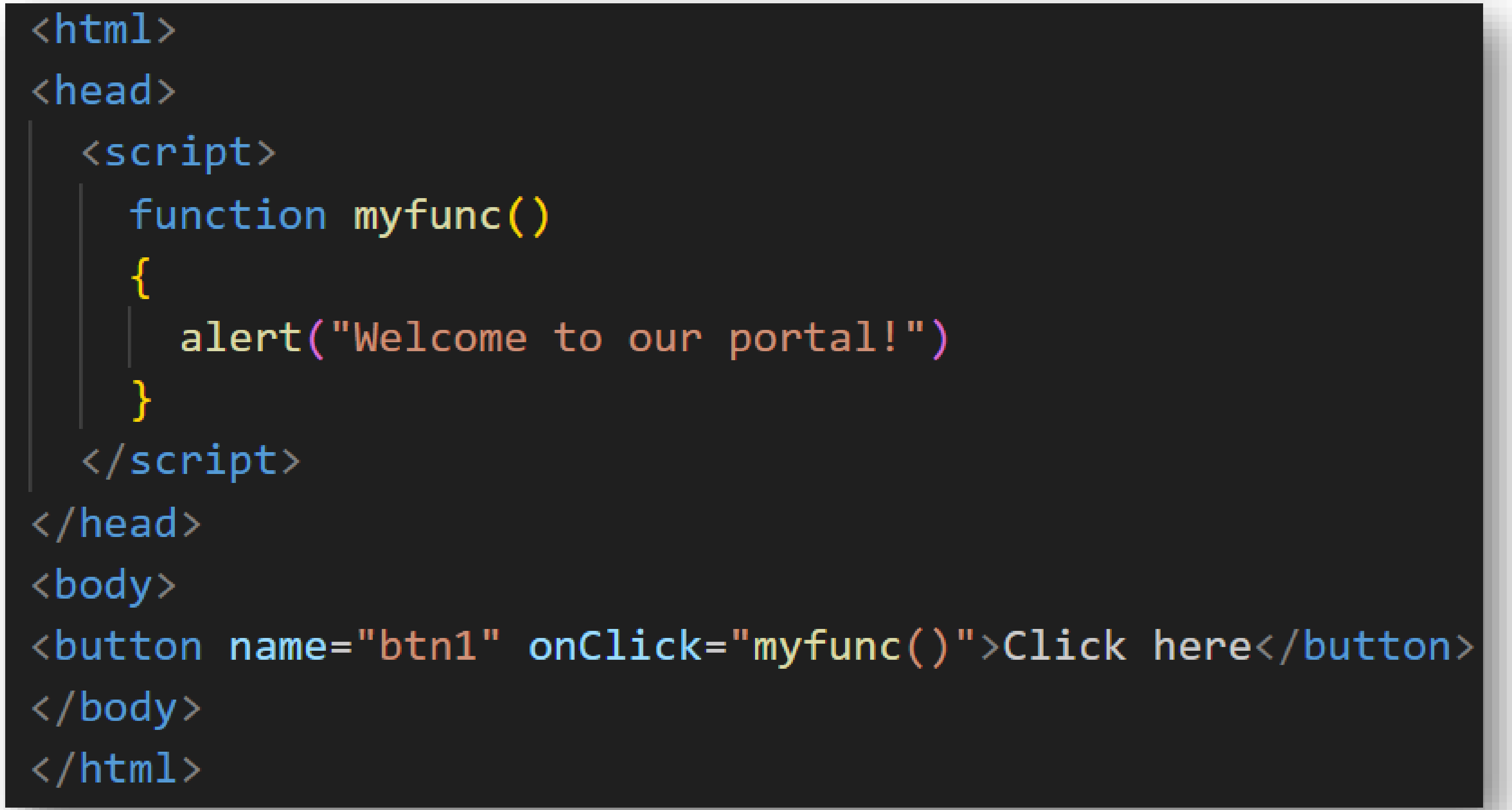
- **Mouse events-**
  - onClick, onMouseOver, onMouseOut, etc.

- **Keyboard events-**
  - onKeyUp, onKeyDown.

- **Form events-**
  - onFocus, onBlur, onSubmit, etc.

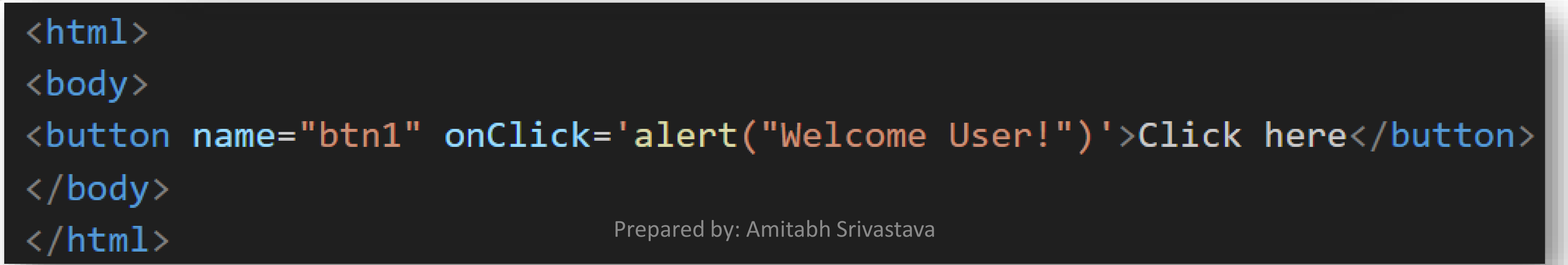
- **Document evets-**
  - onLoad, onResize, etc.

JavaScript Event: Example-1



```
<html>
<head>
  <script>
    function myfunc()
    {
      alert("Welcome to our portal!")
    }
  </script>
</head>
<body>
<button name="btn1" onClick="myfunc()">Click here</button>
</body>
</html>
```

More simple way



```
<html>
<body>
<button name="btn1" onClick='alert("Welcome User!")'>Click here</button>
</body>
</html>
```

Multiple Statements in One Event

- Multiple lines of code can also be write inside the **onClick** event attribute by separating them using a semicolon.

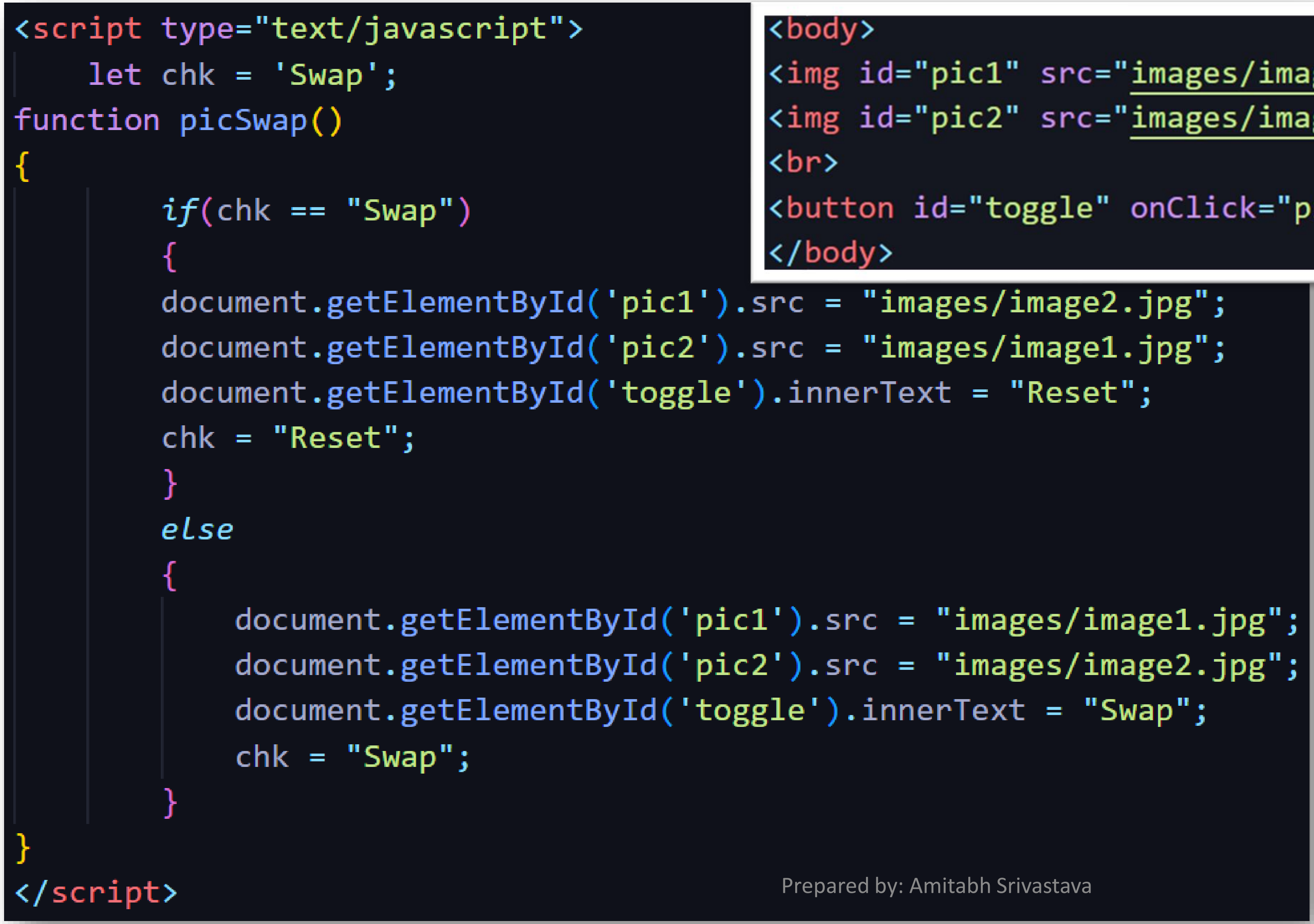
Example-

```
onClick='x=2+5;alert("Sum of 2 and 5 is: "+x)'
```

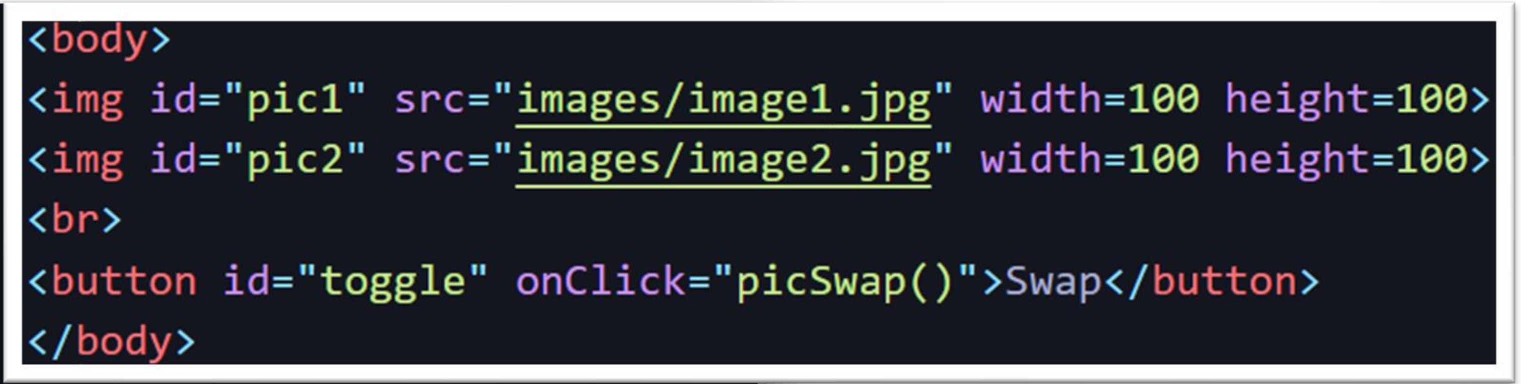
OR, by using back-tick-

```
onClick='x=3+5;alert(`Sum of 3 and 8 is: ${x}`)'
```

Example Code-1: Image Swap

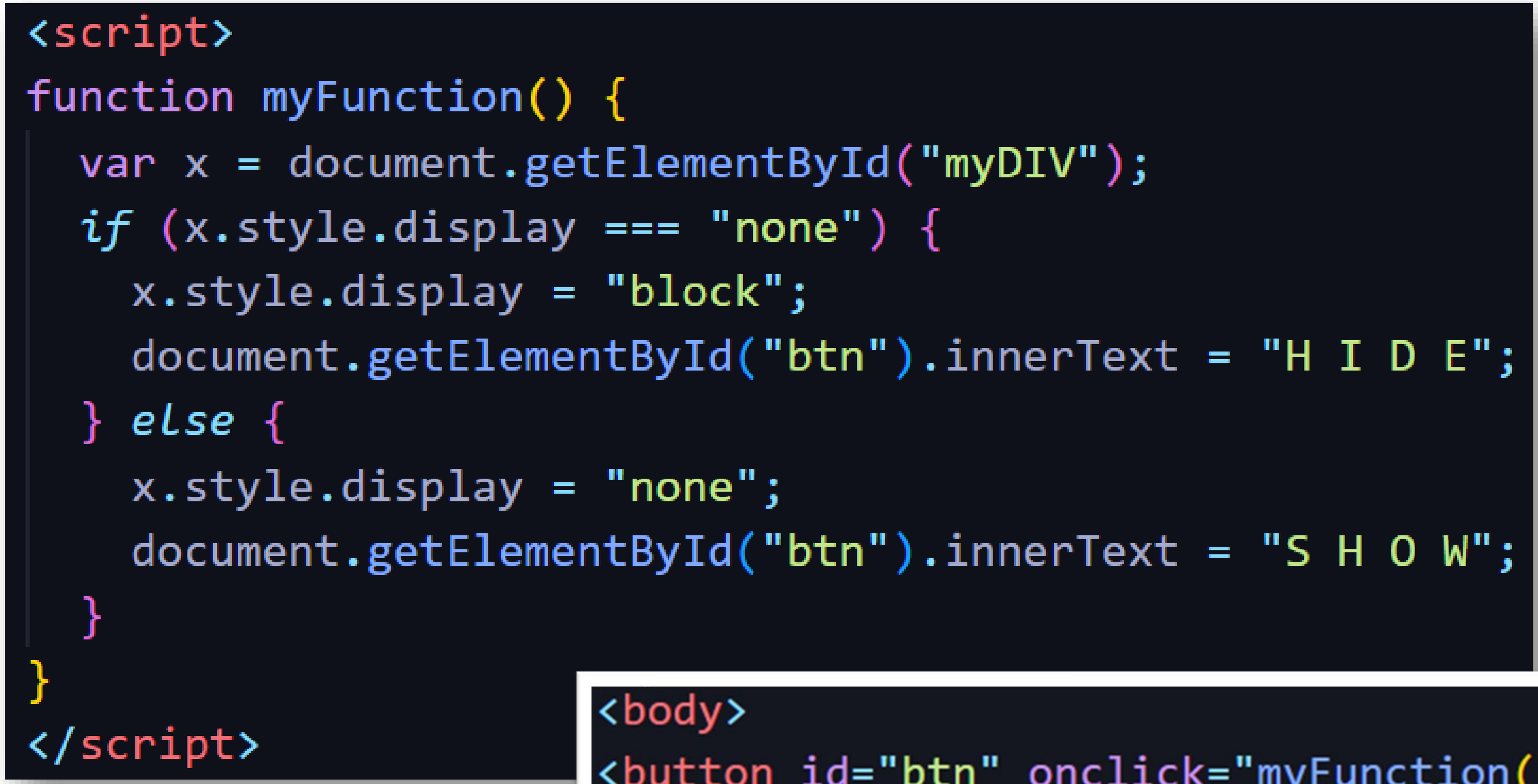


```
<script type="text/javascript">
    let chk = 'Swap';
function picSwap()
{
        if(chk == "Swap")
        {
        document.getElementById('pic1').src = "images/image2.jpg";
        document.getElementById('pic2').src = "images/image1.jpg";
        document.getElementById('toggle').innerText = "Reset";
        chk = "Reset";
        }
        else
        {
            document.getElementById('pic1').src = "images/image1.jpg";
            document.getElementById('pic2').src = "images/image2.jpg";
            document.getElementById('toggle').innerText = "Swap";
            chk = "Swap";
        }
}
</script>
```

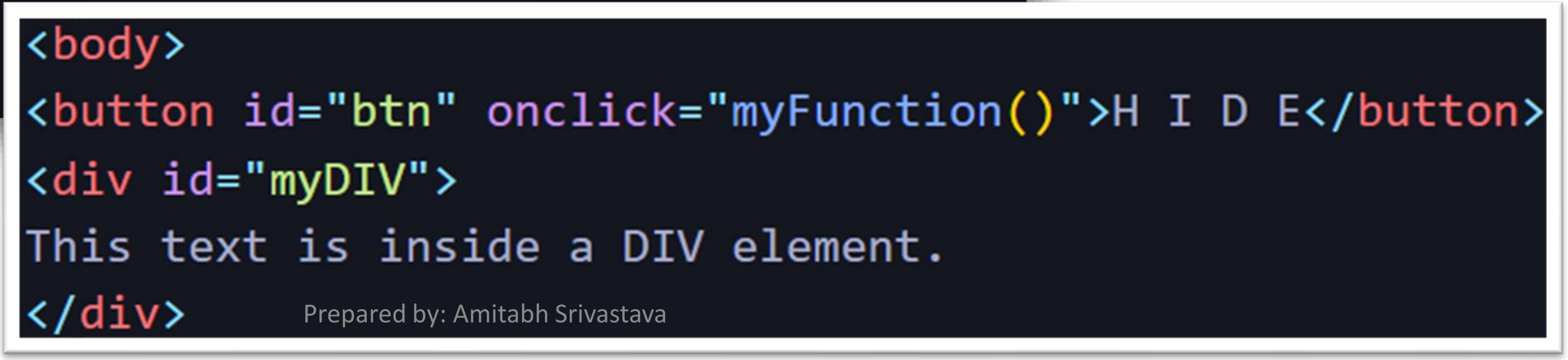


```
<body>
<img id="pic1" src="images/image1.jpg" width=100 height=100>
<img id="pic2" src="images/image2.jpg" width=100 height=100>
<br>
<button id="toggle" onClick="picSwap()">Swap</button>
</body>
```

Example Code-2: Hide/Show Data

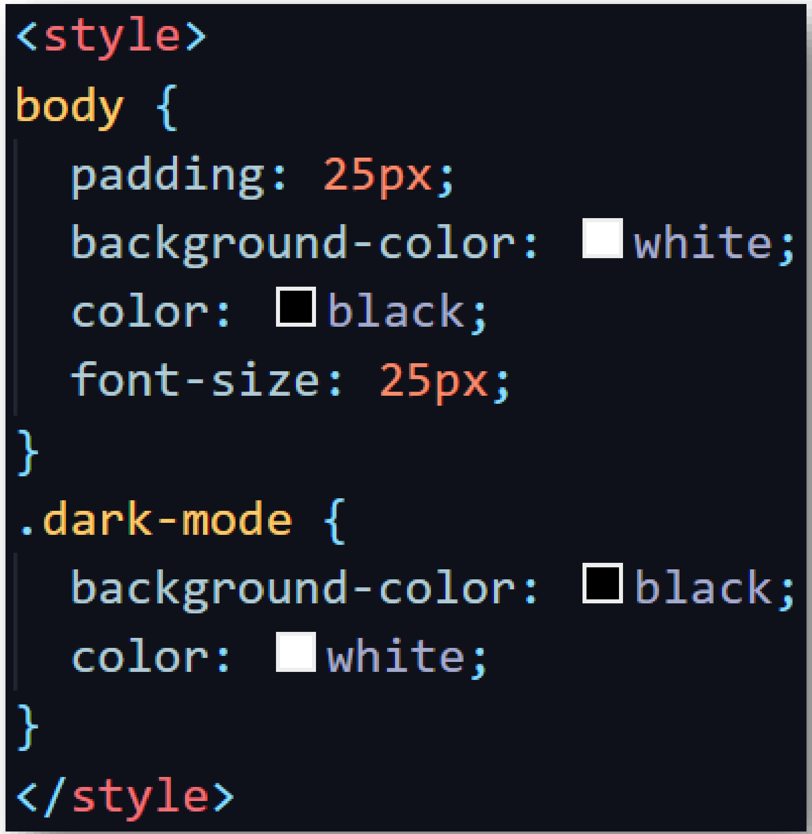


```
<script>
function myFunction() {
  var x = document.getElementById("myDIV");
  if (x.style.display === "none") {
    x.style.display = "block";
    document.getElementById("btn").innerText = "H I D E";
  } else {
    x.style.display = "none";
    document.getElementById("btn").innerText = "S H O W";
  }
}
</script>
```

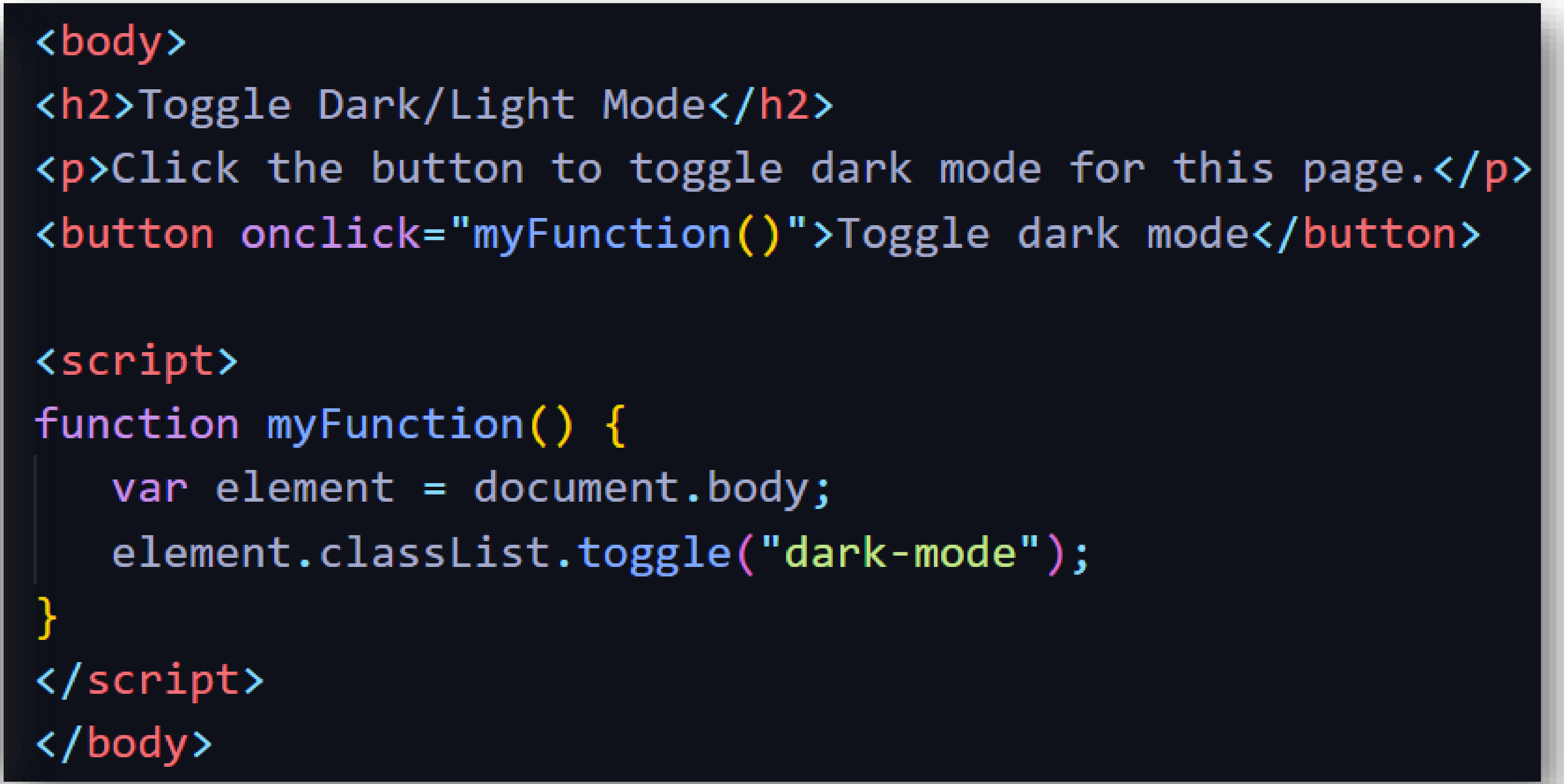


```
<body>
<button id="btn" onclick="myFunction()">H I D E</button>
<div id="myDIV">
This text is inside a DIV element.
</div>
```

Example Code-3: Toggle Dark Mode



```
<style>
body {
  padding: 25px;
  background-color: white;
  color: black;
  font-size: 25px;
}
.dark-mode {
  background-color: black;
  color: white;
}
</style>
```



```
<body>
<h2>Toggle Dark/Light Mode</h2>
<p>Click the button to toggle dark mode for this page.</p>
<button onclick="myFunction()">Toggle dark mode</button>

<script>
function myFunction() {
   var element = document.body;
   element.classList.toggle("dark-mode");
}
</script>
</body>
```