JAVASCRIPT

What is JavaScript?

JavaScript is a programming language designed for Web pages

What is a script

- A program or sequence of instructions that is interpreted or carried out by another program
- A program embedded in an HTML document
- Scripts + HTML → DHTML (dynamic HTML)

Web Pages Layers

Web pages have 3 layers…

1. Structural/Content Layer (XHTML)
2. Presentational Layer (CSS)
   - How things look
3. Behavioral Layer (JavaScript and DOM)
   - How websites behave

What is Java Script ?

- JavaScript is a client-side scripting language.
- A scripting language is a lightweight programming language.
- JavaScript is programming code that can be inserted into HTML pages.
- JavaScript inserted into HTML pages, can be executed by all modern web browsers.
- Java Script can enhance the dynamics and interactive features of your page by allowing you to perform calculations, check forms, write interactive games, add special effects, customize graphics selections, create security passwords and more.

What can it be used for

- Text animation
- graphic animation
- HTML forms submission
- client-side forms data validation
- web site navigation

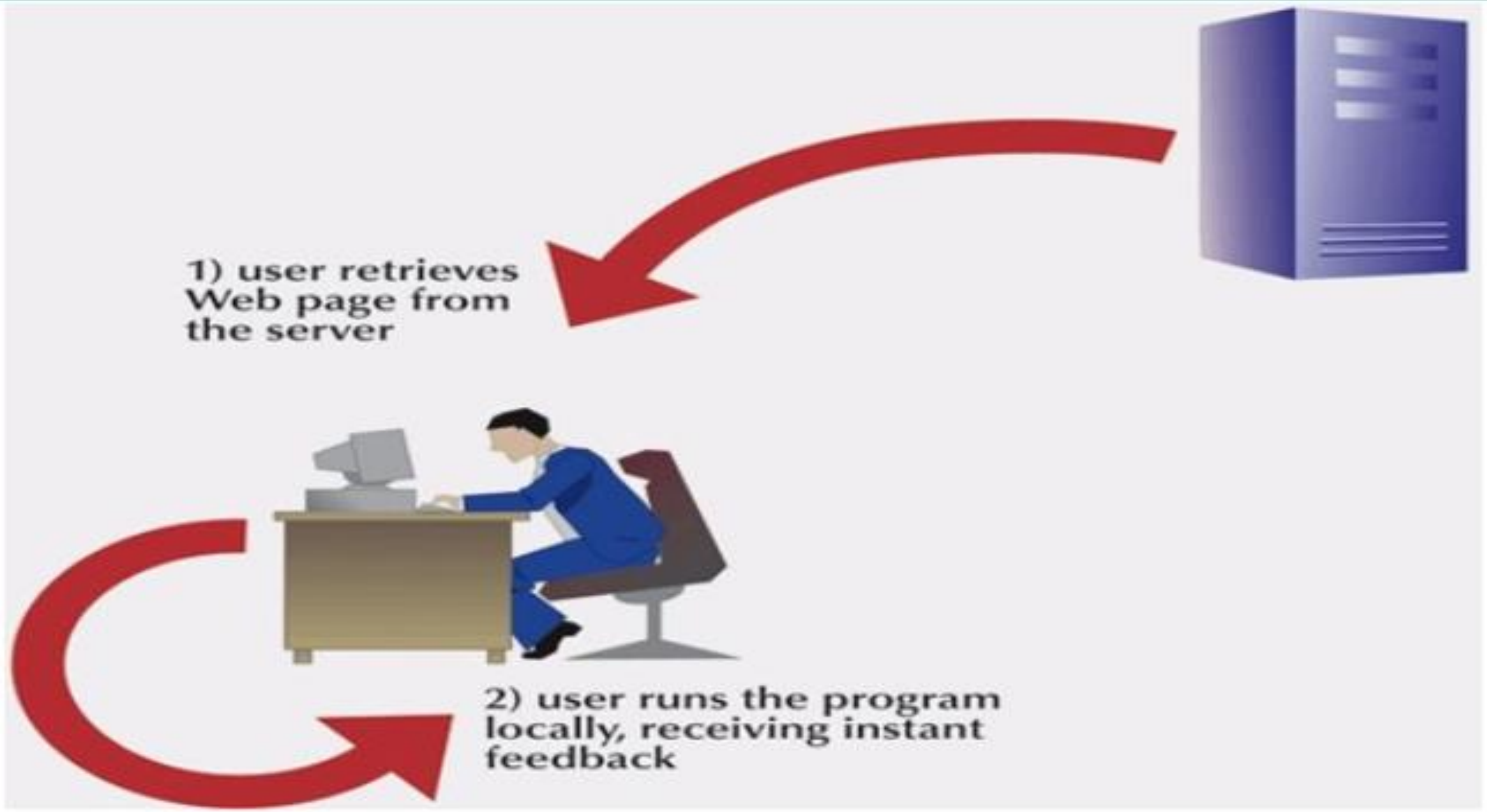
Introduction

JavaScript is used in web pages for:

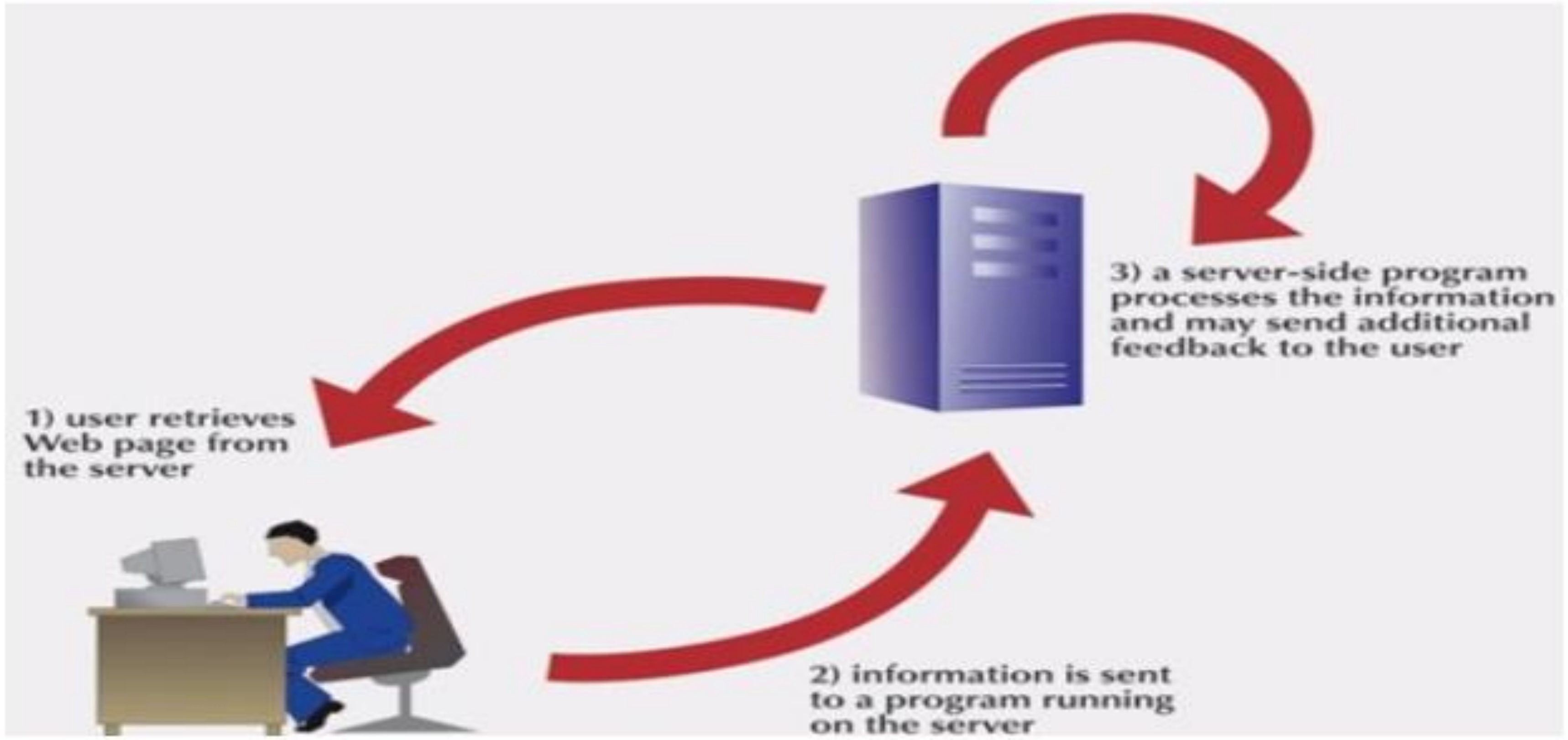
- Dynamics: mouse clicks, pop up windows, and animations
- Client-side execution: validating input, processing requests

- It avoids Client/Server communication and traffic
- JavaScript is executed on client-side
- JavaScript is simple, powerful, and interpretive language and requires only a web browser

Client-Side Programming

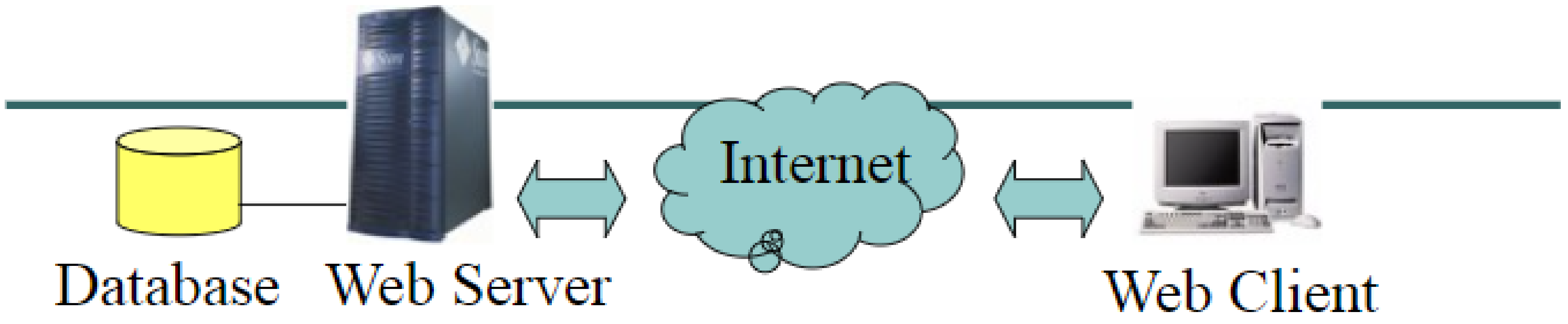

1) user retrieves Web page from the server
2) user runs the program locally, receiving instant feedback

Server-Side Programming


3) a server-side program processes the information and may send additional feedback to the user
1) user retrieves Web page from the server
2) information is sent to a program running on the server

Client side scripting	Server side scripting
• Used when the users browser already has all the code	• Used to create dynamic pages
• The Web Browser executes the client side scripting	• The Web Server executes the server side scripting
• Cannot be used to connect to the databases on the web server	• Used to connect to the databases that reside on the web server
• Can’t access the file system that resides at the web server	• Can access the file system residing at the web server
• Response from a client-side script is faster as compared to a server-side script	• Response from a server-side script is slower as compared to a client-side script

Server side and Client side Programming



Server-side Programming

- CGI
- PHP
- ASP
- Java Servlet, ...

Client-side Programming

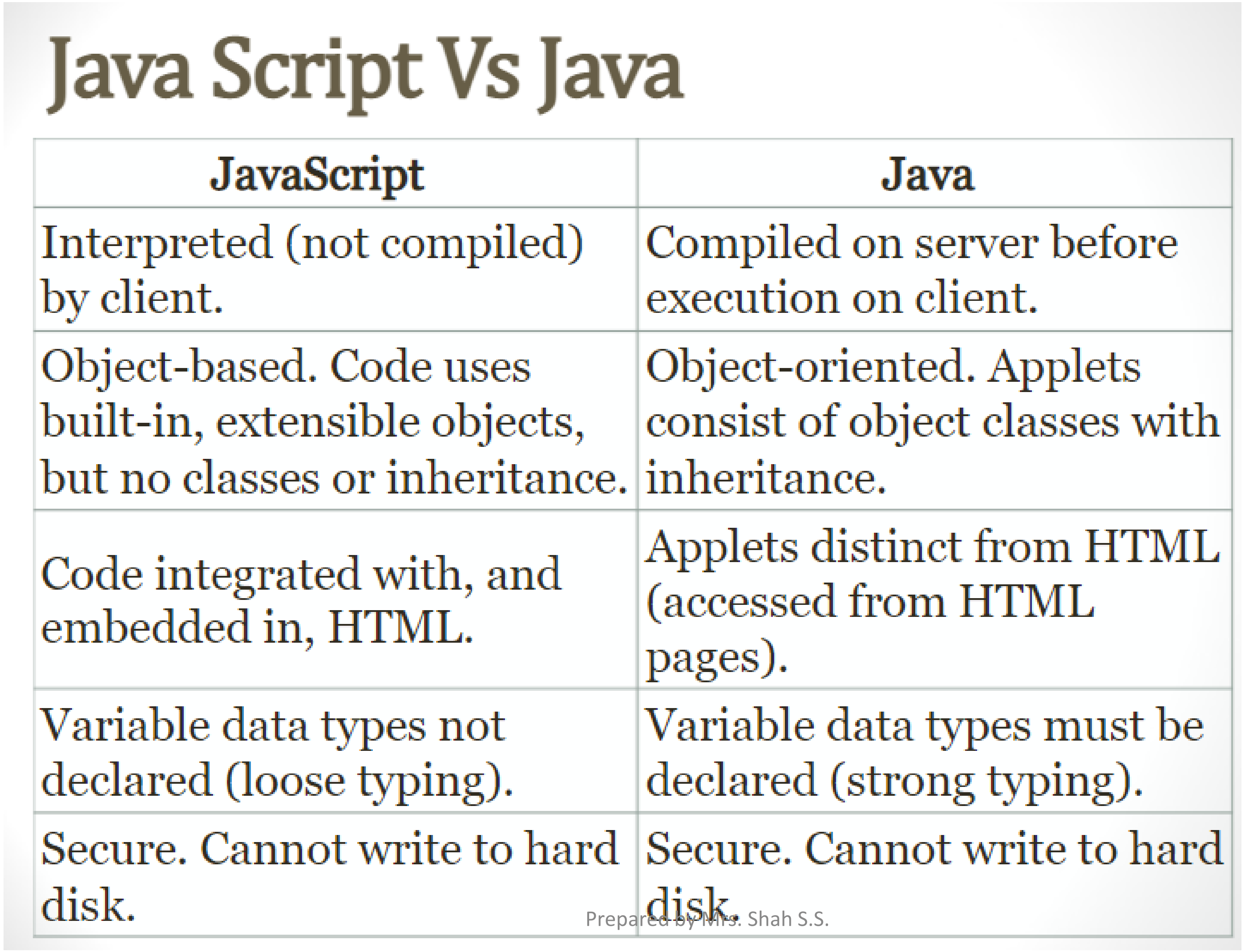
- XHTML
- Javascript
- Dreamweaver
- Flash
- XML ...

Advantages of Javascript

- **Speed**
- Client-side JavaScript is extremely quick since it very well may be run promptly inside the client-side program. Except if outside assets are required, JavaScript is unhindered by network calls to a backend server.
- **Simplicity**
- JavaScript is relatively simple to learn and implement.
- **Popularity**
- javascript is used everywhere on the web.

Disadvantages of JavaScript

- **Client-Side Security** - Since JavaScript code is executed on the client-side, bugs and oversights can now and then be taken advantage of for malicious purposes. Along these lines, certain individuals decide to cripple JavaScript completely.
- **Browser Support** - While server-side scripts generally produce similar results, various programs here and there decipher JavaScript code in an unexpected way.



Java Script Vs Java

JavaScript	Java
Interpreted (not compiled) by client.	Compiled on server before execution on client.
Object-based. Code uses built-in, extensible objects, but no classes or inheritance.	Object-oriented. Applets consist of object classes with inheritance.
Code integrated with, and embedded in, HTML.	Applets distinct from HTML (accessed from HTML pages).
Variable data types not declared (loose typing).	Variable data types must be declared (strong typing).
Secure. Cannot write to hard disk.	Secure. Cannot write to hard disk.

The <script>...</script> tag

- The code for the script is contained in the <script>…</script> tag

```
<script type="text/javascript">
              .
              .
              .
</script>
```

How to use/implement Java Script?

- We can implement Java script in our web page by following three ways-
  1. Inside the head tag
  2. Within the body tag
  3. In an external file (with extension .js)

Implementing Java Script

1. **Inside HEAD Tag:**

Syntax:

```
<HTML>
      <HEAD>
            <SCRIPT TYPE= “TEXT/JAVASCRIPT”>
                    <!- -
                                Java Script Code
                    // - ->
              </SCRIPT>
      </HEAD>
      <BODY>

      </BODY>
</HTML>
```

Implementing Java Script

2. **Within BODY Tag:**

Syntax:

```
<HTML>
     <HEAD>
     </HEAD>
     <BODY>
          <SCRIPT TYPE= “TEXT/JAVASCRIPT”>
                 <!- -
      java script code
                 // - ->
          </SCRIPT>
     </BODY>
</HTML>
```

Implementing Java Script

3. **In an External Link:**

Syntax:

```
<HTML>
      <HEAD>
            <SCRIPT SRC= “myscript.js”>
                    </SCRIPT>
      </HEAD>
      <BODY>
        <input TYPE=“Button” onclick=“msg()” value=“Message”>
      </BODY>
</HTML>
```

Myscript.js:

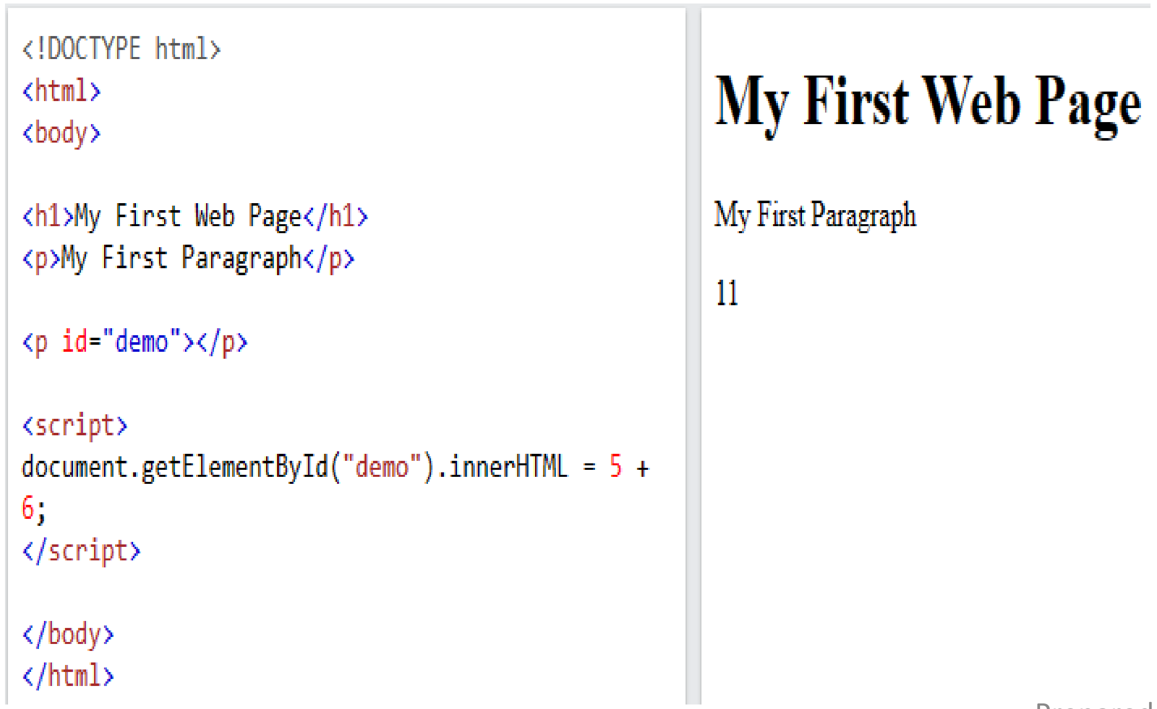
```
Function msg()
{ alert("Hello") }
```

JavaScript Display Possibilities

JavaScript can "display" data in different ways:

Writing into an HTML element, using innerHTML.
Writing into the HTML output using document.write().
Writing into an alert box, using window.alert().
Writing into the browser console, using console.log().

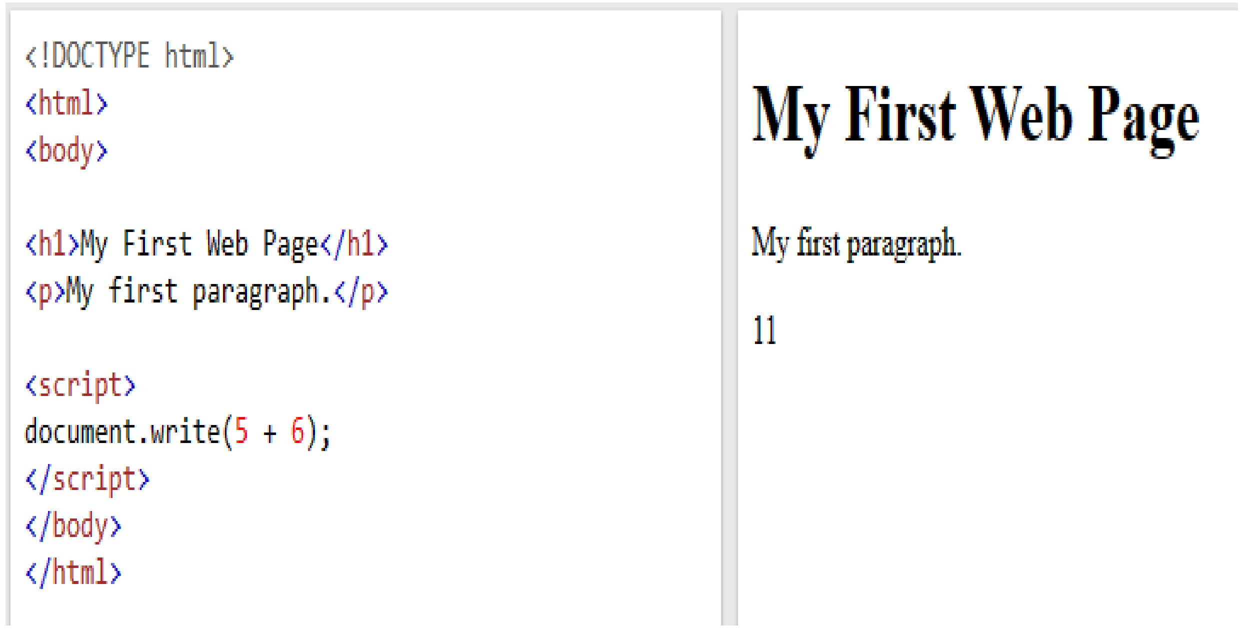
Using innerHTML



```
<!DOCTYPE html>
<html>
<body>

<h1>My First Web Page</h1>
<p>My First Paragraph</p>

<p id="demo"></p>

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

</body>
</html>
```

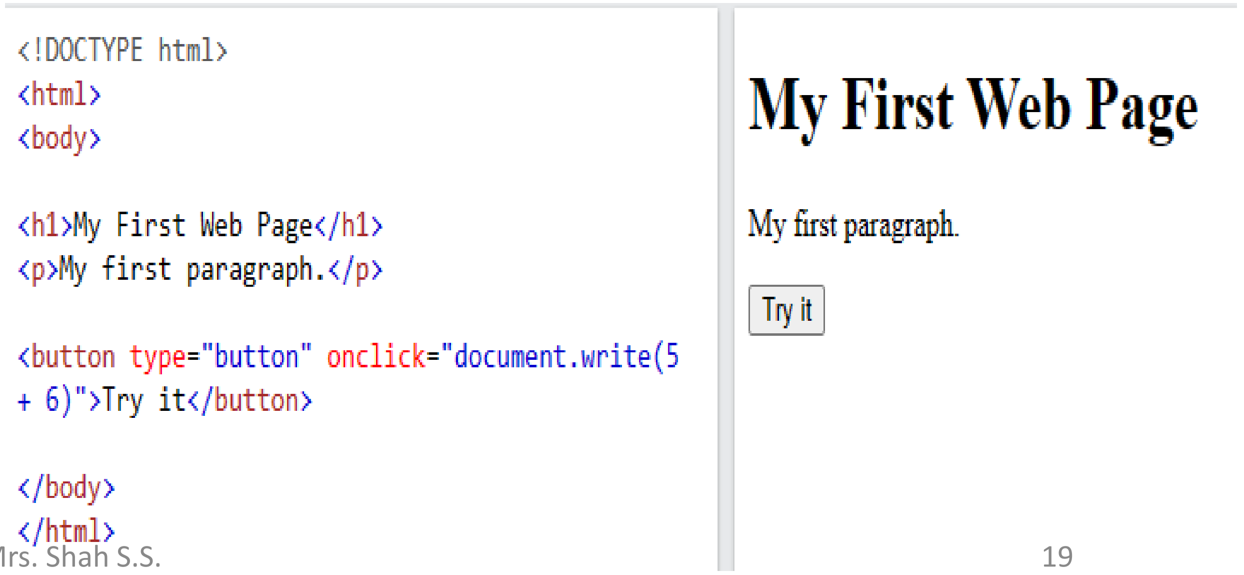
```
My First Web Page

My First Paragraph

11
```

Using document.write()



```
<!DOCTYPE html>
<html>
<body>

<h1>My First Web Page</h1>
<p>My first paragraph.</p>

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

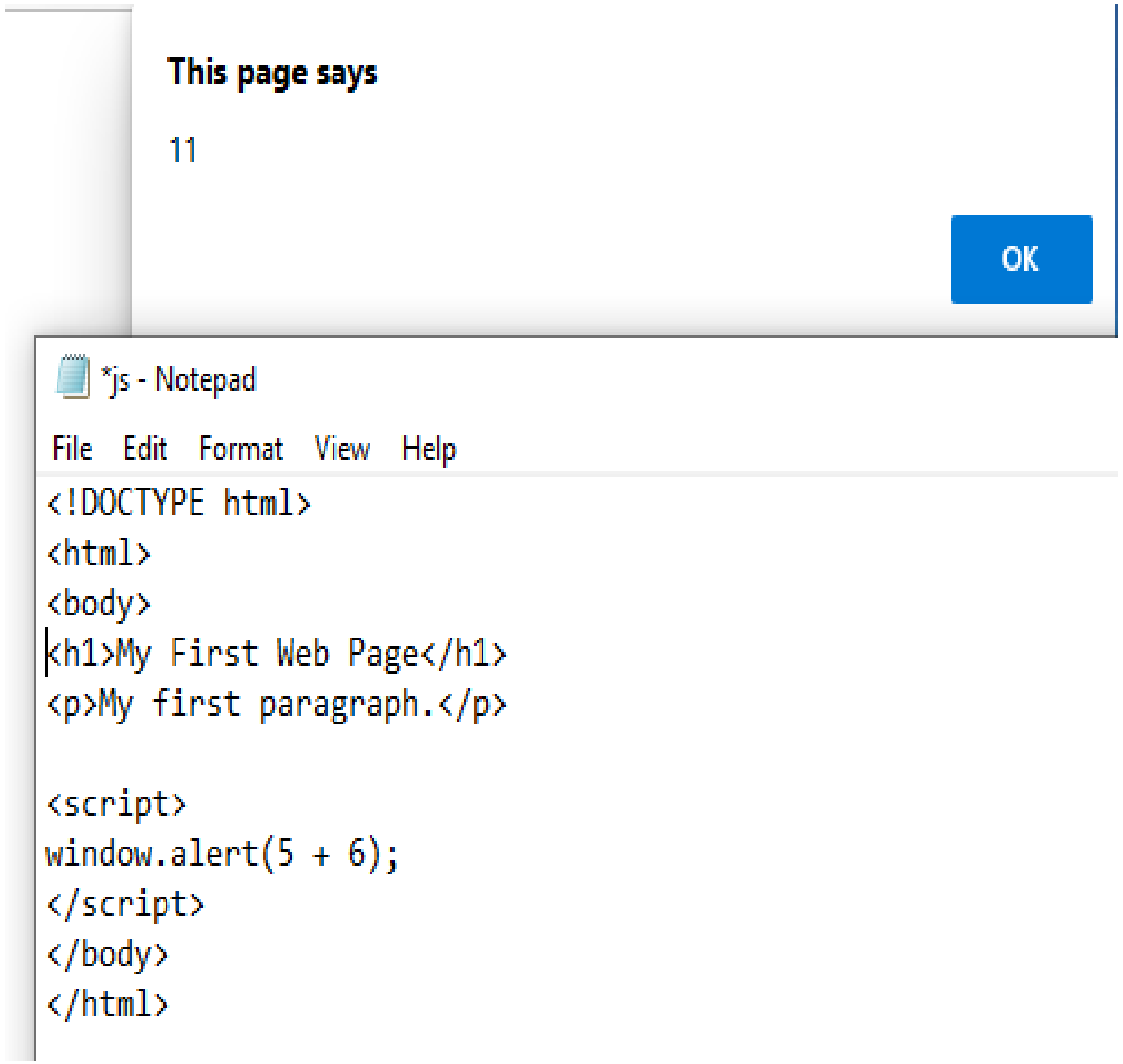
```
My First Web Page

My first paragraph.

11
```



```
<!DOCTYPE html>
<html>
<body>

<h1>My First Web Page</h1>
<p>My first paragraph.</p>

<button type="button" onclick="document.write(5
+ 6)">Try it</button>

</body>
</html>
```

```
My First Web Page

My first paragraph.

Try it
```

Using window.alert()

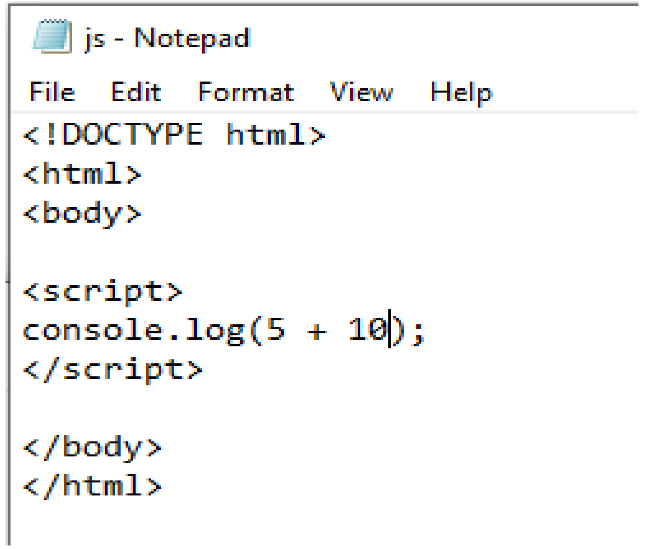

This page says

11

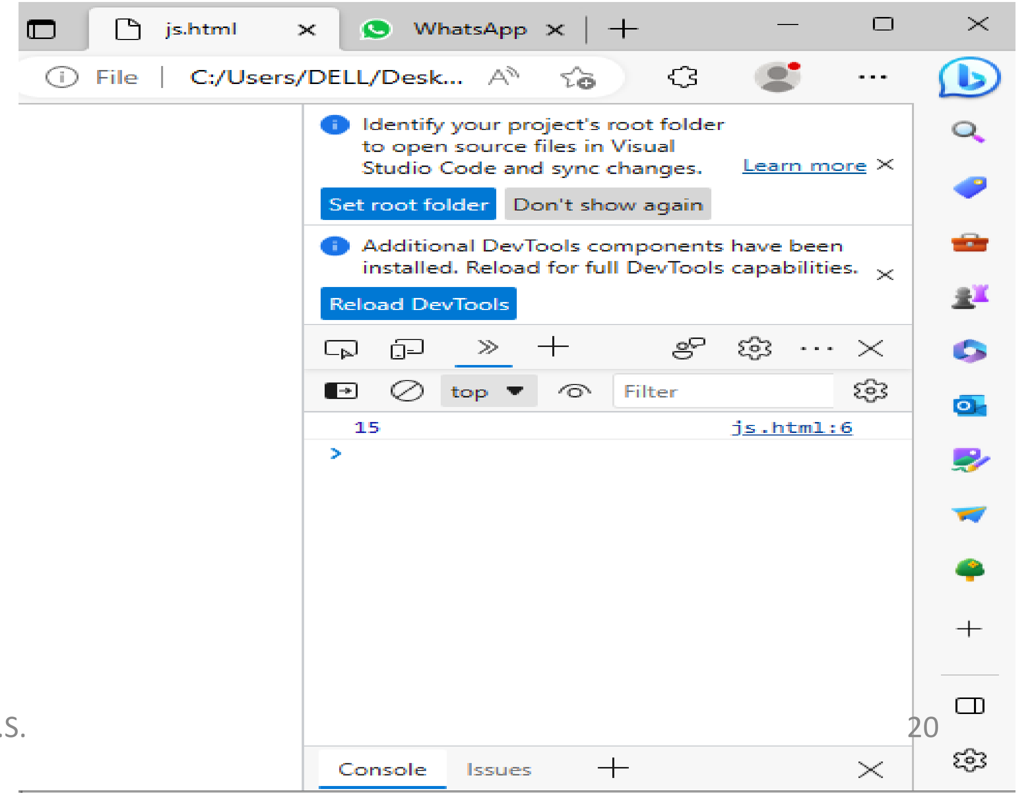
*js - Notepad

```
<!DOCTYPE html>
<html>
<body>
<h1>My First Web Page</h1>
<p>My first paragraph.</p>

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

Using console.log()


js - Notepad

```
<!DOCTYPE html>
<html>
<body>

<script>
console.log(5 + 10);
</script>

</body>
</html>
```



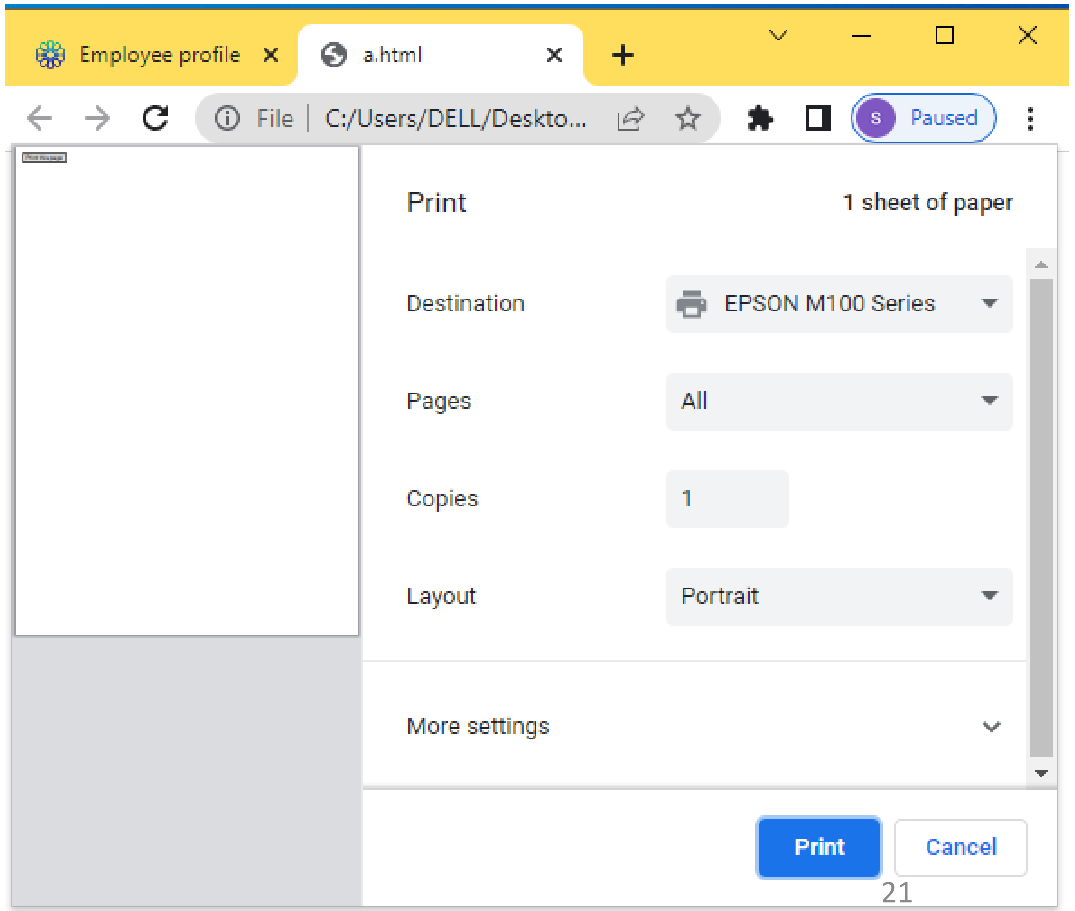
Identify your project's root folder to open source files in Visual Studio Code and sync changes.

Additional DevTools components have been installed. Reload for full DevTools capabilities.

```
15                                   js.html:6
>
```

JavaScript Print

```
<!DOCTYPE html>
<html>
<body>

<button onclick="window.print()">Print this page</button>

</body>
</html>
```



Employee profile
a.html
File | C:/Users/DELL/Deskto...
Paused

Print 1 sheet of paper

Destination EPSON M100 Series

Pages All

Copies 1

Layout Portrait

More settings

Print Cancel

JavaScript Data Types

```
// Numbers:
let length = 16;
let weight = 7.5;

// Strings:
let color = "Yellow";
let lastName = "Johnson";

// Booleans
let x = true;
let y = false;
 // Exponential Notation

  let y = 123e5;    // 12300000
  let z = 123e-5;   // 0.00123
```

```
//BigInt
   let x = BigInt("123456789012345678901234567890");

 // Object:
const person = {firstName:"John", lastName:"Doe"};

// Array object:
const cars = ["Saab", "Volvo", "BMW"];

// Date object:
const date = new Date("2022-03-25");
```

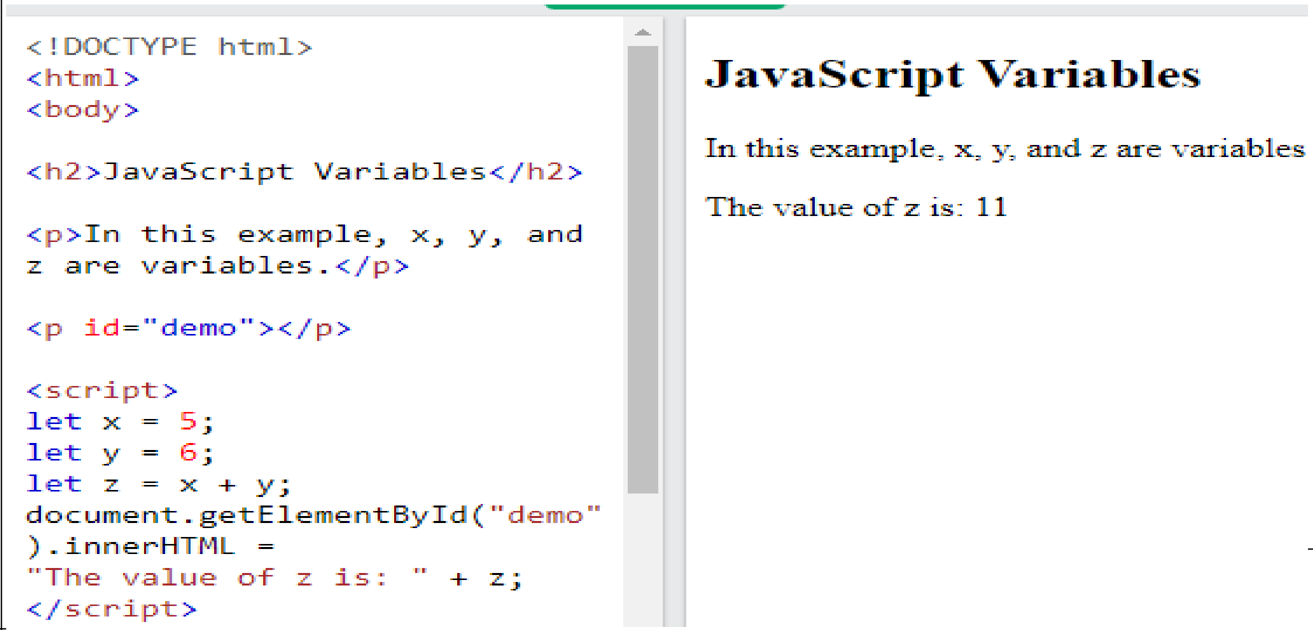
JavaScript Variables

Variables are containers for storing data (storing data values).

- **Things To Remember While Naming A Variable**
- Variable names should be short and easy to remember.
- They should be descriptive enough to tell you what the variable represents.
- They should not be too generic or too specific to avoid confusion.
- They should not include any special characters like $, %, or @, except underscore.
- They should not contain spaces.
- They should not start with a number.
- Start a variable name with a letter or underscore(_).
- Javascript variables are case-sensitive, i.e., x is not equal to X.
- They should be unique and not used by other variables in the code.
- They could either be camel-cased or lowercase.
- They should not start with a capital letter.

4 Ways to Declare a JavaScript Variable:

- •Using `var`
- •Using `let`
- •Using `const`
- •Using nothing

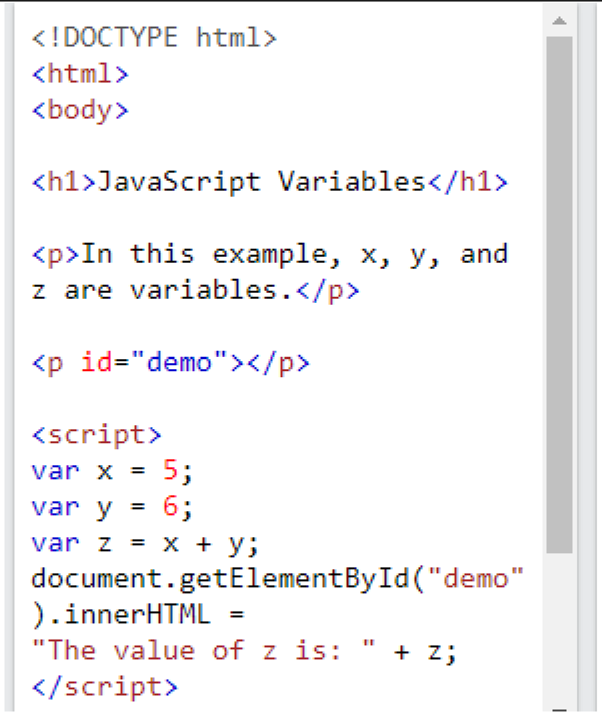


```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Variables</h2>

<p>In this example, x, y, and
z are variables.</p>

<p id="demo"></p>

<script>
let x = 5;
let y = 6;
let z = x + y;
document.getElementById("demo"
).innerHTML =
"The value of z is: " + z;
</script>
```

JavaScript Variables

In this example, x, y, and z are variables

The value of z is: 11

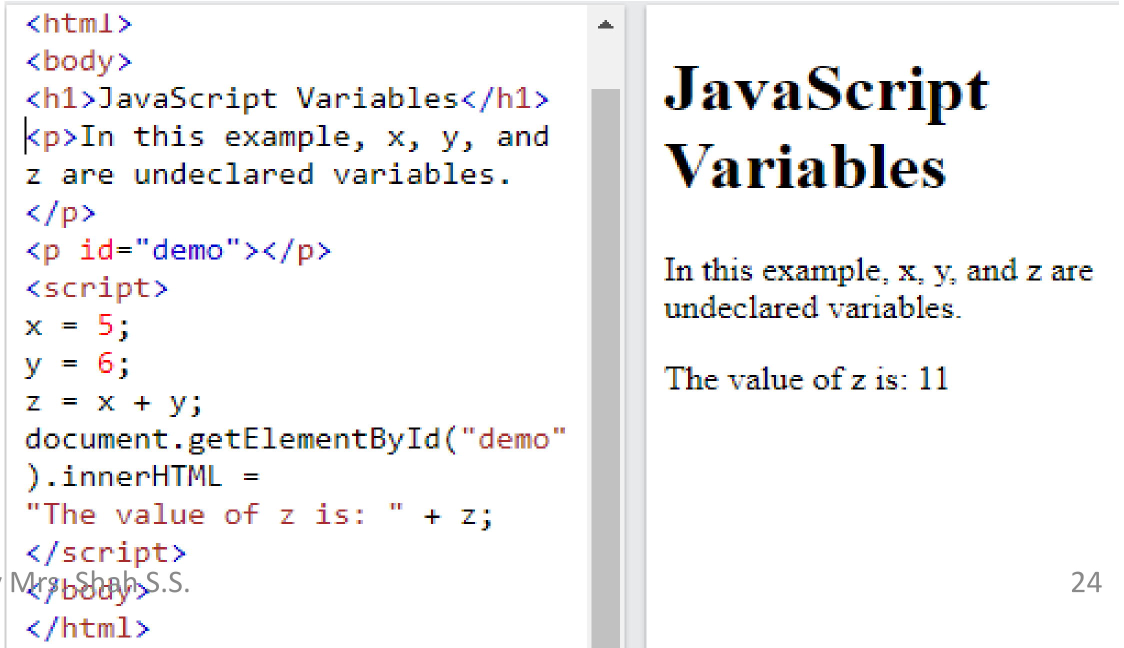


```
<!DOCTYPE html>
<html>
<body>

<h1>JavaScript Variables</h1>

<p>In this example, x, y, and
z are variables.</p>

<p id="demo"></p>

<script>
var x = 5;
var y = 6;
var z = x + y;
document.getElementById("demo"
).innerHTML =
"The value of z is: " + z;
</script>
```

JavaScript Variables

In this example, x, y, and z are variables.

The value of z is: 11



```
<html>
<body>
<h1>JavaScript Variables</h1>
<p>In this example, x, y, and
z are undeclared variables.
</p>
<p id="demo"></p>
<script>
x = 5;
y = 6;
z = x + y;
document.getElementById("demo"
).innerHTML =
"The value of z is: " + z;
</script>
</body>
</html>
```

JavaScript Variables

In this example, x, y, and z are undeclared variables.

The value of z is: 11

JavaScript Let

The `let` keyword was introduced in (2015).
Variables defined with `let` **can not be redeclared**.
Variables defined with `let` **must be declared before use**.
Variables defined with `let` **have block scope**.

Block Scope

`let` and `const` these two keywords provide **Block Scope** in JavaScript.
Variables declared inside a { } block cannot be accessed from outside the block:

Example

```
{
  let x = 2;
}
// x can NOT be used here
```

Variables declared with the `var` keyword can NOT have block scope.
Variables declared inside a { } block can be accessed from outside the block.

Example

```
{
  var x = 2;
}
// x CAN be used here
```

Cannot be Redeclared

Variables defined with `let` **can not be redeclared**.
You can not accidentally redeclare a variable declared with `let`.

With `let` you can **not** do this:

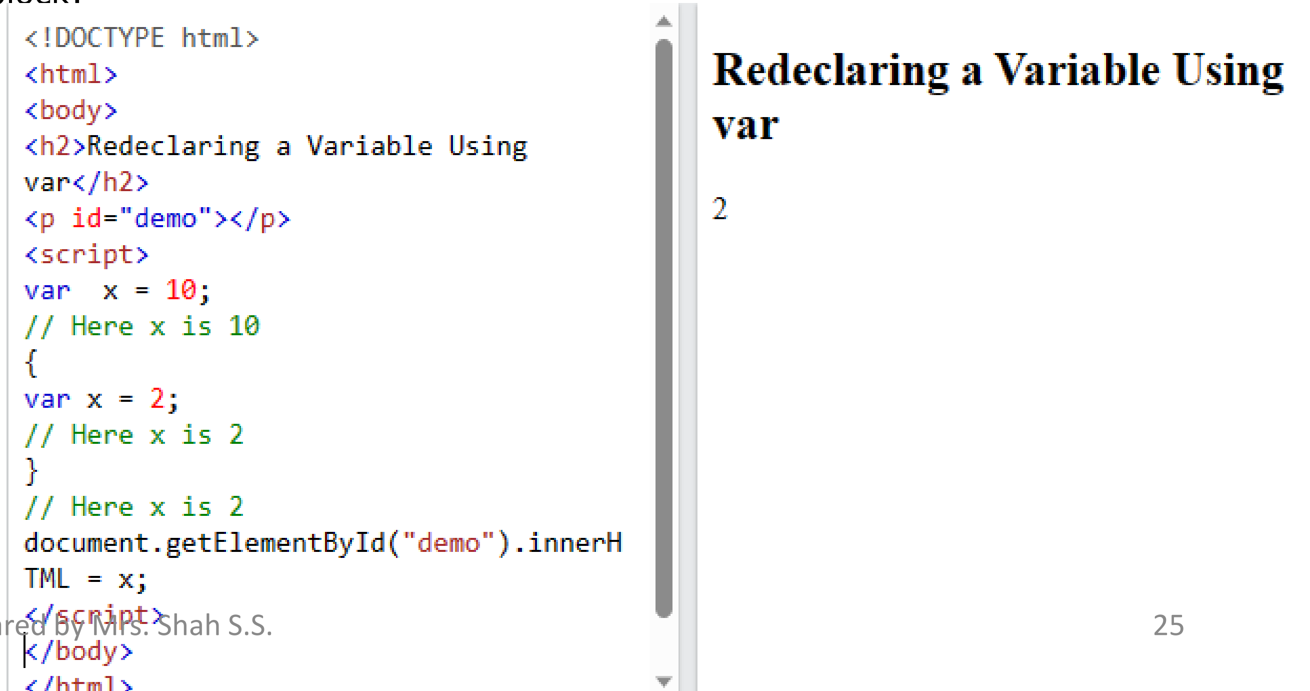
```
let x = "John Doe";
let x = 0;
```

With `var` you can:

```
var x = "John Doe";
var x = 0;
```

Redeclaring Variables

Redeclaring a variable using the `var` keyword can impose problems.
Redeclaring a variable inside a block will also redeclare the variable outside the block:



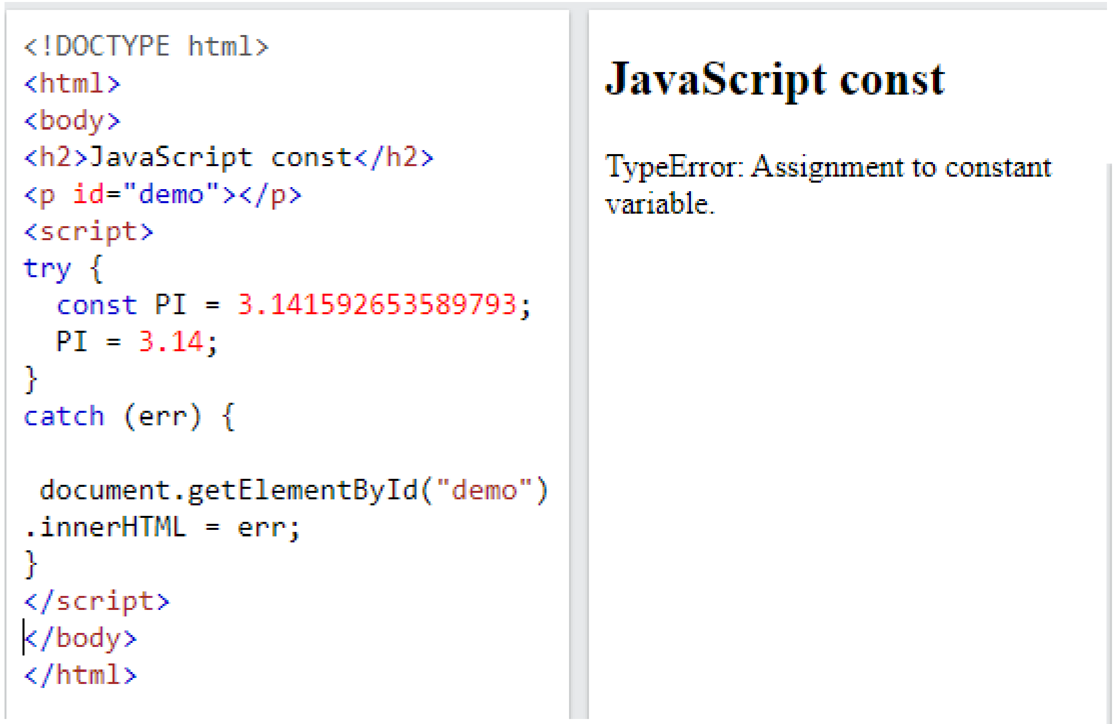
```
<!DOCTYPE html>
<html>
<body>
<h2>Redeclaring a Variable Using
var</h2>
<p id="demo"></p>
<script>
var  x = 10;
// Here x is 10
{
var x = 2;
// Here x is 2
}
// Here x is 2
document.getElementById("demo").innerH
TML = x;
</script>
</body>
</html>
```

```
Redeclaring a Variable Using
var

2
```

JavaScript Const

The `const` keyword was introduced in (2015).
Variables defined with `const` cannot be Redeclared.
Variables defined with `const` cannot be Reassigned.
Variables defined with `const` have Block Scope.

Cannot be Reassigned

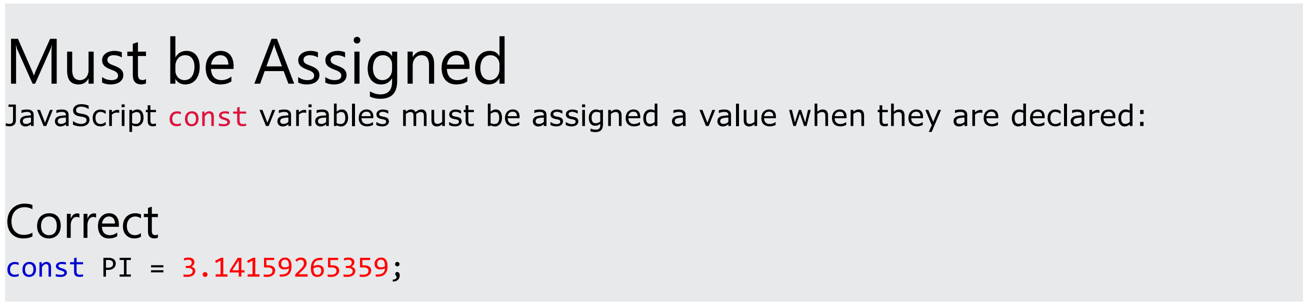


```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript const</h2>
<p id="demo"></p>
<script>
try {
  const PI = 3.141592653589793;
  PI = 3.14;
}
catch (err) {

 document.getElementById("demo")
.innerHTML = err;
}
</script>
</body>
</html>
```

JavaScript const

TypeError: Assignment to constant variable.

Must be Assigned



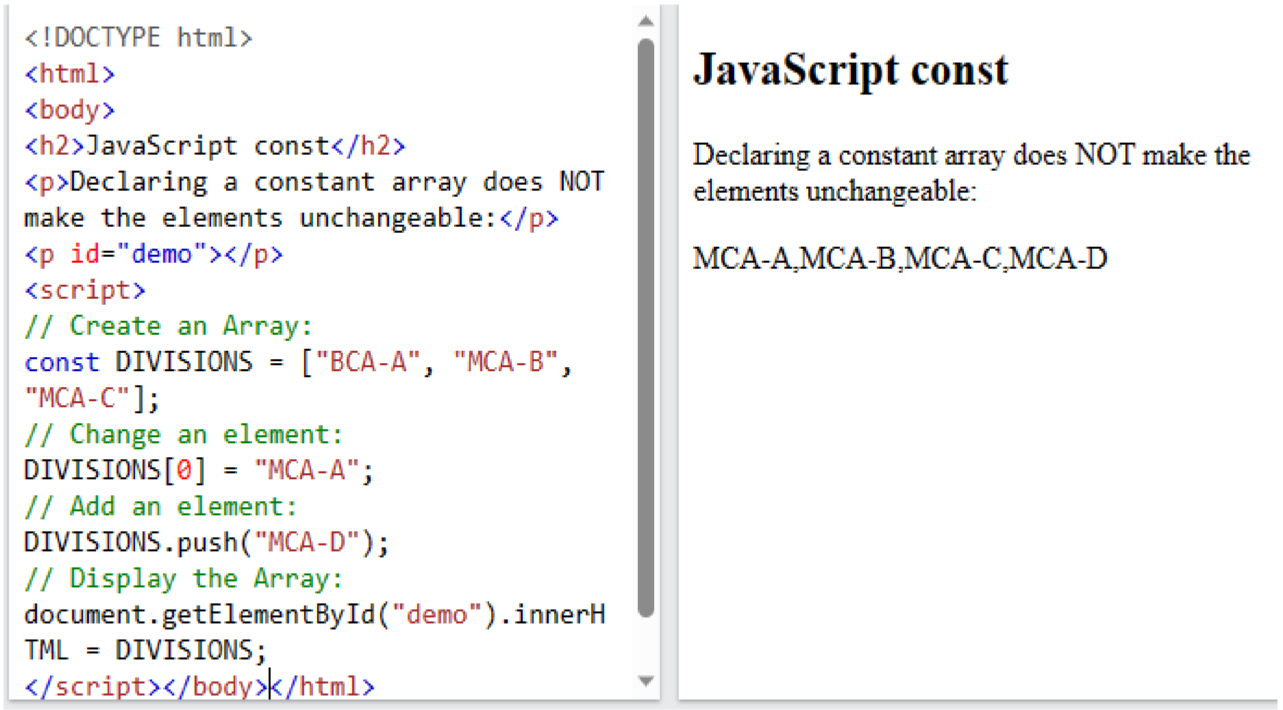
Must be Assigned

JavaScript `const` variables must be assigned a value when they are declared:

Correct

```
const PI = 3.14159265359;
```

Constant Arrays



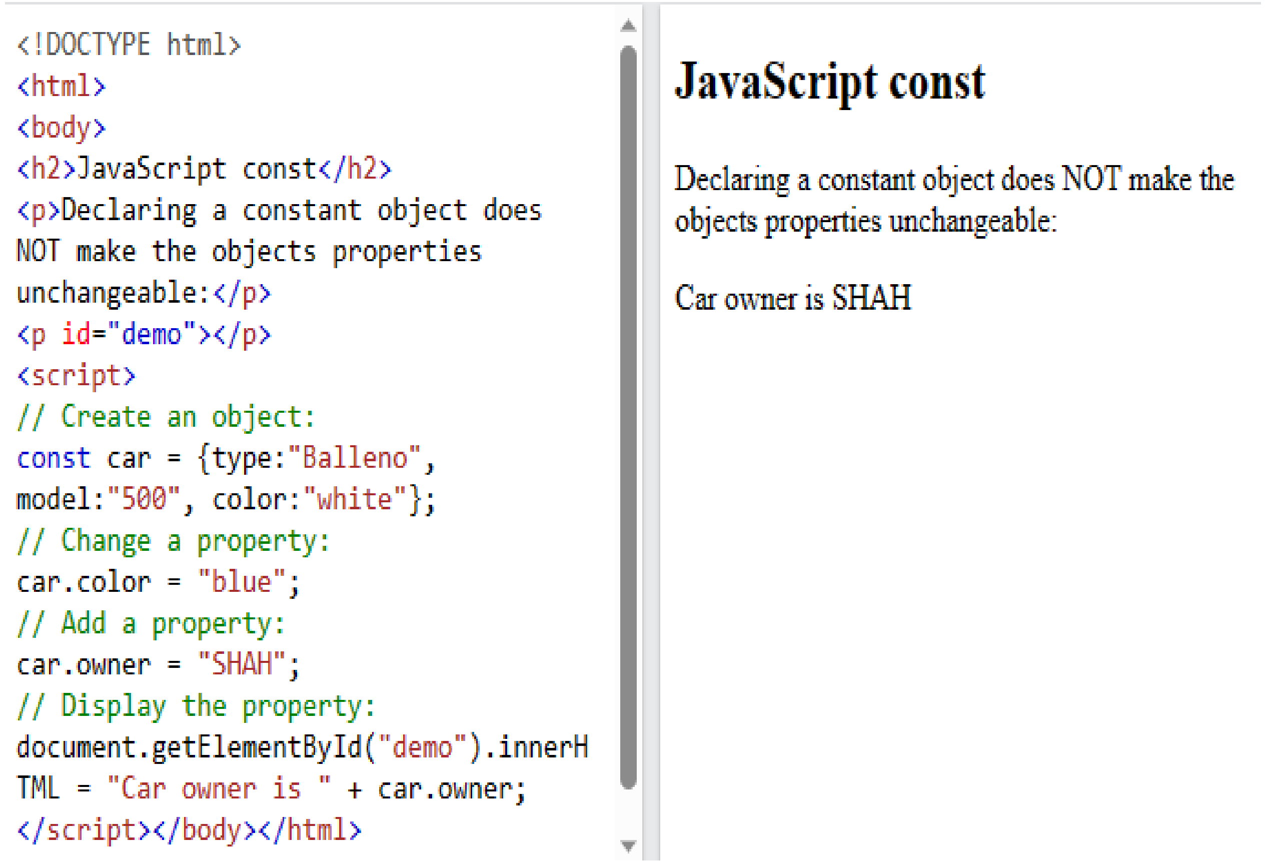
```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript const</h2>
<p>Declaring a constant array does NOT
make the elements unchangeable:</p>
<p id="demo"></p>
<script>
// Create an Array:
const DIVISIONS = ["BCA-A", "MCA-B",
"MCA-C"];
// Change an element:
DIVISIONS[0] = "MCA-A";
// Add an element:
DIVISIONS.push("MCA-D");
// Display the Array:
document.getElementById("demo").innerH
TML = DIVISIONS;
</script></body></html>
```

JavaScript const

Declaring a constant array does NOT make the elements unchangeable:

MCA-A,MCA-B,MCA-C,MCA-D

Constant Objects

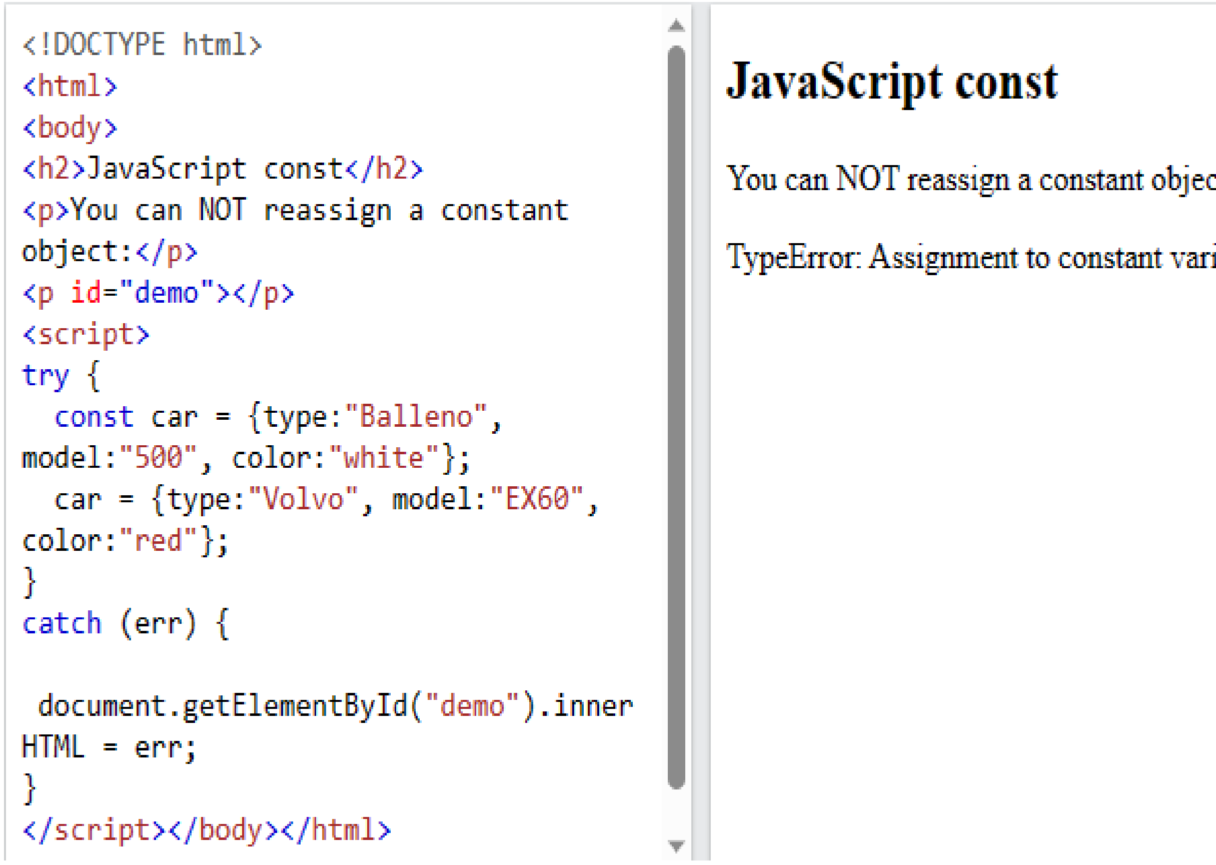


```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript const</h2>
<p>Declaring a constant object does
NOT make the objects properties
unchangeable:</p>
<p id="demo"></p>
<script>
// Create an object:
const car = {type:"Balleno",
model:"500", color:"white"};
// Change a property:
car.color = "blue";
// Add a property:
car.owner = "SHAH";
// Display the property:
document.getElementById("demo").innerH
TML = "Car owner is " + car.owner;
</script></body></html>
```

JavaScript const

Declaring a constant object does NOT make the objects properties unchangeable:

Car owner is SHAH



```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript const</h2>
<p>You can NOT reassign a constant
object:</p>
<p id="demo"></p>
<script>
try {
  const car = {type:"Balleno",
model:"500", color:"white"};
  car = {type:"Volvo", model:"EX60",
color:"red"};
}
catch (err) {

 document.getElementById("demo").inner
HTML = err;
}
</script></body></html>
```

JavaScript const

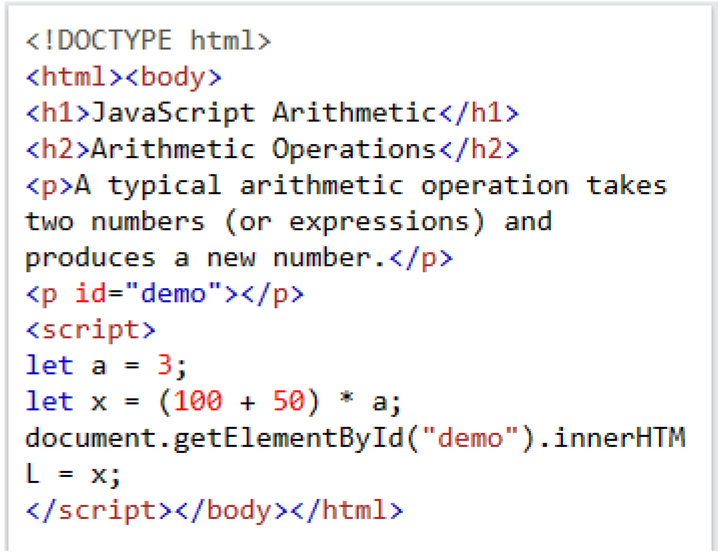
You can NOT reassign a constant objec

TypeError: Assignment to constant vari

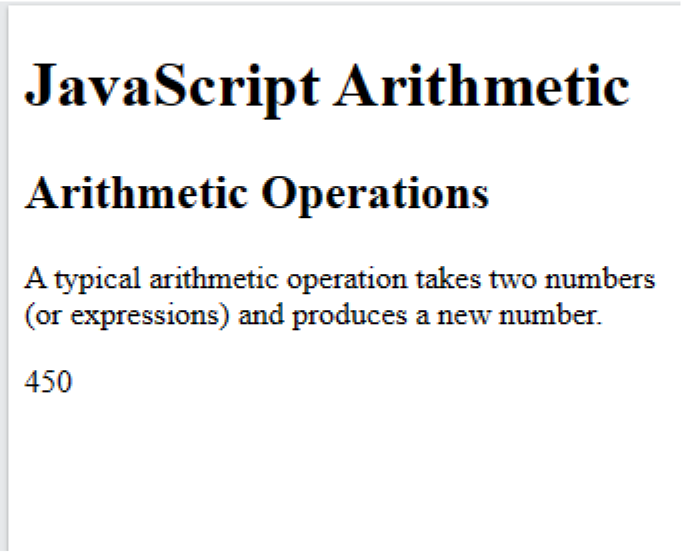
- **Types of JavaScript Operators**
- There are different types of JavaScript operators:
- Arithmetic Operators
- Assignment Operators
- Comparison Operators
- String Operators
- Logical Operators
- Bitwise Operators
- Ternary Operators
- Type Operators

Arithmetic Operators

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
**	Exponentiation (ES2016)
/	Division
%	Modulus (Division Remainder)
++	Increment
--	Decrement



```
<!DOCTYPE html>
<html><body>
<h1>JavaScript Arithmetic</h1>
<h2>Arithmetic Operations</h2>
<p>A typical arithmetic operation takes
two numbers (or expressions) and
produces a new number.</p>
<p id="demo"></p>
<script>
let a = 3;
let x = (100 + 50) * a;
document.getElementById("demo").innerHTM
L = x;
</script></body></html>
```

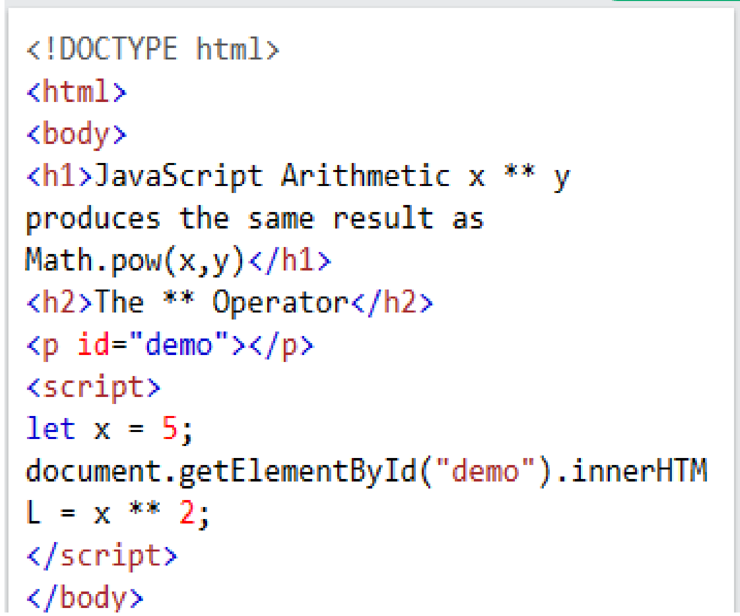


JavaScript Arithmetic

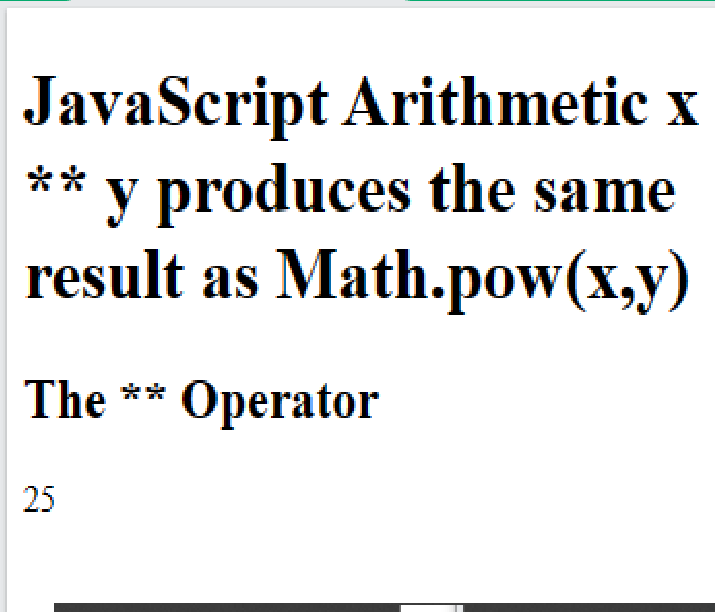
Arithmetic Operations

A typical arithmetic operation takes two numbers (or expressions) and produces a new number.

450



```
<!DOCTYPE html>
<html>
<body>
<h1>JavaScript Arithmetic x ** y
produces the same result as
Math.pow(x,y)</h1>
<h2>The ** Operator</h2>
<p id="demo"></p>
<script>
let x = 5;
document.getElementById("demo").innerHTM
L = x ** 2;
</script>
</body>
```



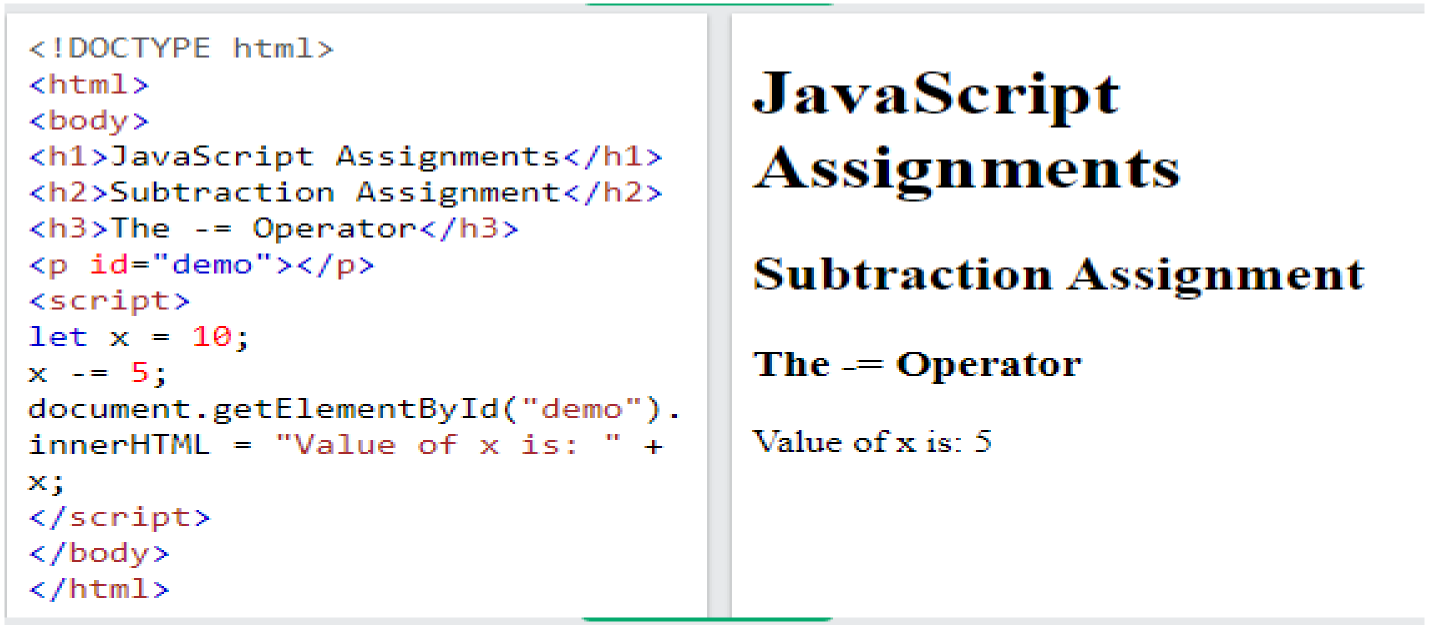
JavaScript Arithmetic x ** y produces the same result as Math.pow(x,y)

The ** Operator

25

JavaScript Assignment Operators

Operator	Example	Same As
=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y
**=	x **= y	x = x ** y



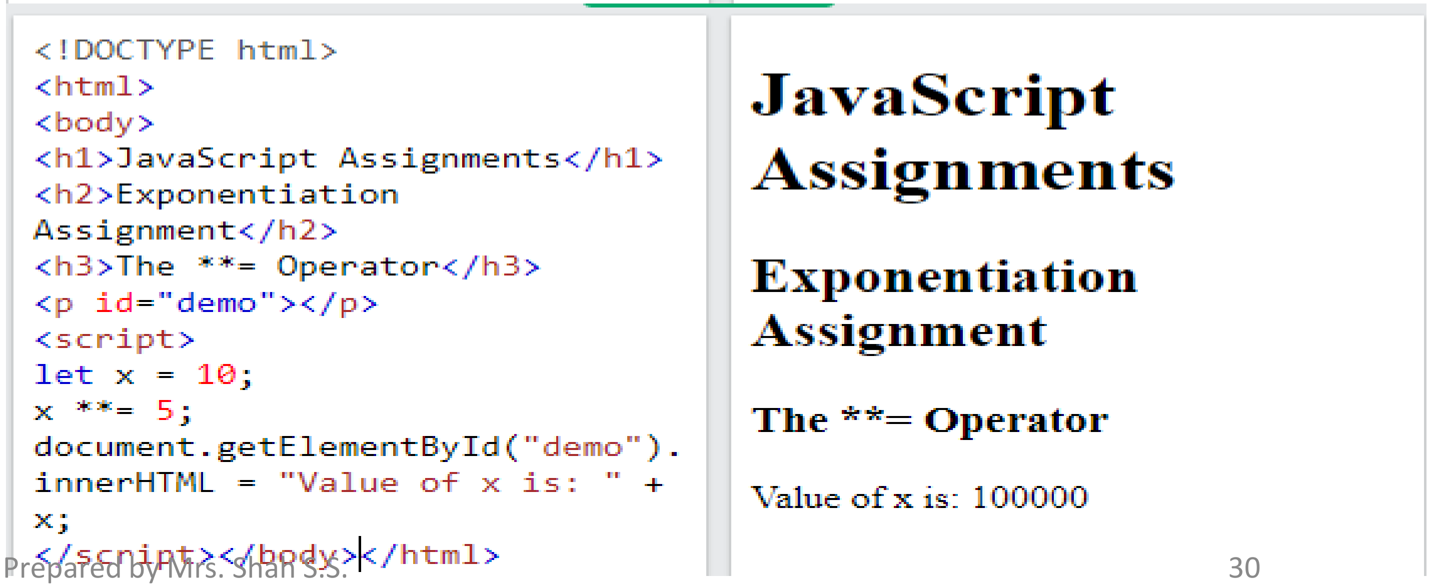
```
<!DOCTYPE html>
<html>
<body>
<h1>JavaScript Assignments</h1>
<h2>Subtraction Assignment</h2>
<h3>The -= Operator</h3>
<p id="demo"></p>
<script>
let x = 10;
x -= 5;
document.getElementById("demo").
innerHTML = "Value of x is: " +
x;
</script>
</body>
</html>
```

JavaScript Assignments

Subtraction Assignment

The -= Operator

Value of x is: 5



```
<!DOCTYPE html>
<html>
<body>
<h1>JavaScript Assignments</h1>
<h2>Exponentiation
Assignment</h2>
<h3>The **= Operator</h3>
<p id="demo"></p>
<script>
let x = 10;
x **= 5;
document.getElementById("demo").
innerHTML = "Value of x is: " +
x;
</script></body></html>
```

JavaScript Assignments

Exponentiation Assignment

The **= Operator

Value of x is: 100000

Shift Assignment Operators

Operator	Example	Same As
<<=	x <<= y	x = x << y
>>=	x >>= y	x = x >> y
>>>=	x >>>= y	x = x >>> y

Bitwise Assignment Operators

Operator	Example	Same As
&=	x &= y	x = x & y
^=	x ^= y	x = x ^ y
|=	x |= y	x = x | y

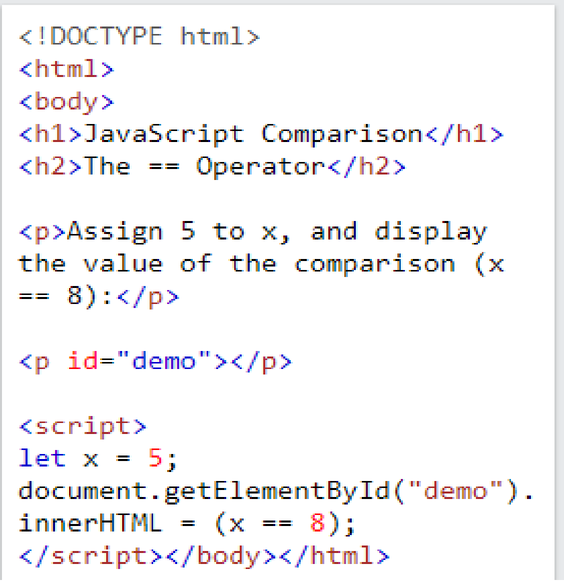
Logical Assignment Operators

Operator	Example	Same As
&&=	x &&= y	x = x && (x = y)
||=	x ||= y	x = x || (x = y)
??=	x ??= y	x = x ?? (x = y)

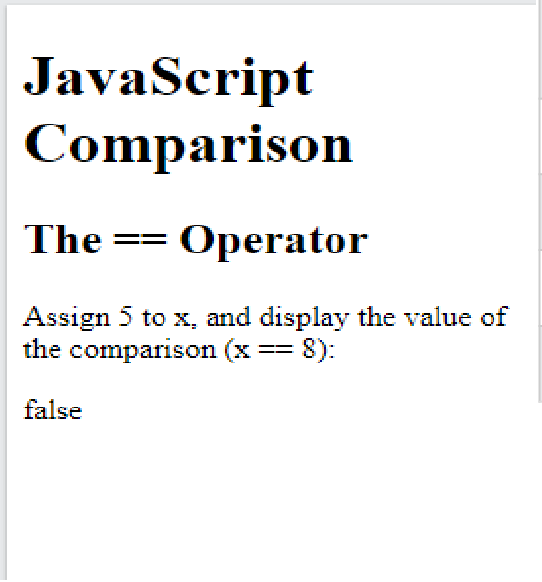
JavaScript Comparison Operators

Operator	Description
==	equal to
===	equal value and equal type
!=	not equal
!==	not equal value or not equal type
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to
?	ternary operator

Operator	Description	Comparing	Returns
==	equal to	x == 8	false
		x == 5	true
		x == "5"	true
===	equal value and equal type	x === 5	true
		x === "5"	false
!=	not equal	x != 8	true
!==	not equal value or not equal type	x !== 5	false
		x !== "5"	true
		x !== 8	true
>	greater than	x > 8	false
<	less than	x < 8	true
>=	greater than or equal to	x >= 8	false
<=	less than or equal to	x <= 8	true



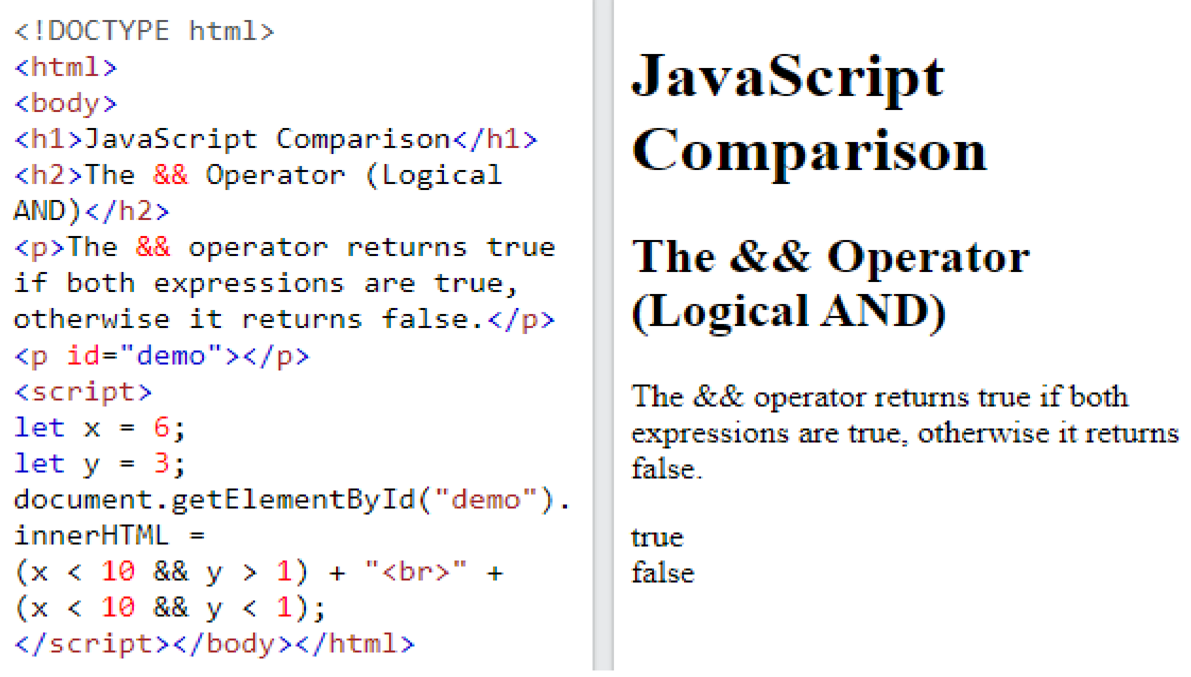
```
<!DOCTYPE html>
<html>
<body>
<h1>JavaScript Comparison</h1>
<h2>The == Operator</h2>

<p>Assign 5 to x, and display
the value of the comparison (x
== 8):</p>

<p id="demo"></p>

<script>
let x = 5;
document.getElementById("demo").
innerHTML = (x == 8);
</script></body></html>
```



```
JavaScript
Comparison

The == Operator

Assign 5 to x, and display the value of
the comparison (x == 8):

false
```

Logical Operators

Operator	Description	Example
&&	and	(x < 10 && y > 1) is true
||	or	(x == 5 || y == 5) is false
!	not	!(x == y) is true



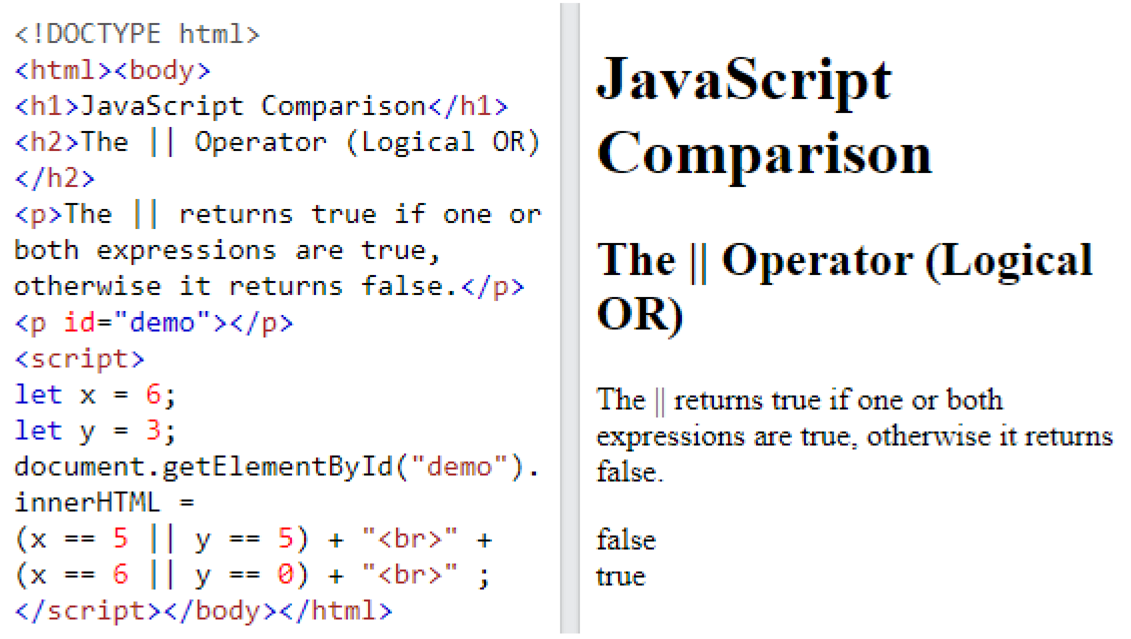
```
<!DOCTYPE html>
<html>
<body>
<h1>JavaScript Comparison</h1>
<h2>The && Operator (Logical
AND)</h2>
<p>The && operator returns true
if both expressions are true,
otherwise it returns false.</p>
<p id="demo"></p>
<script>
let x = 6;
let y = 3;
document.getElementById("demo").
innerHTML =
(x < 10 && y > 1) + "<br>" +
(x < 10 && y < 1);
</script></body></html>
```

JavaScript Comparison

The && Operator (Logical AND)

The && operator returns true if both expressions are true, otherwise it returns false.

```
true
false
```



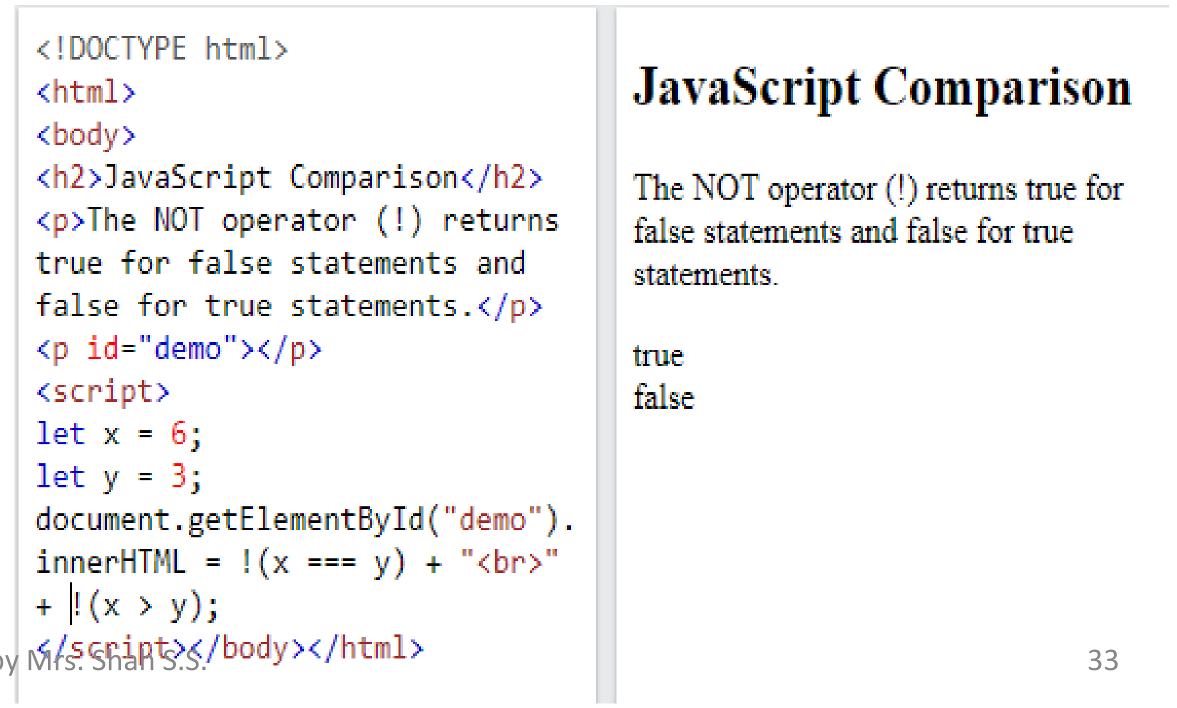
```
<!DOCTYPE html>
<html><body>
<h1>JavaScript Comparison</h1>
<h2>The || Operator (Logical OR)
</h2>
<p>The || returns true if one or
both expressions are true,
otherwise it returns false.</p>
<p id="demo"></p>
<script>
let x = 6;
let y = 3;
document.getElementById("demo").
innerHTML =
(x == 5 || y == 5) + "<br>" +
(x == 6 || y == 0) + "<br>" ;
</script></body></html>
```

JavaScript Comparison

The || Operator (Logical OR)

The || returns true if one or both expressions are true, otherwise it returns false.

```
false
true
```



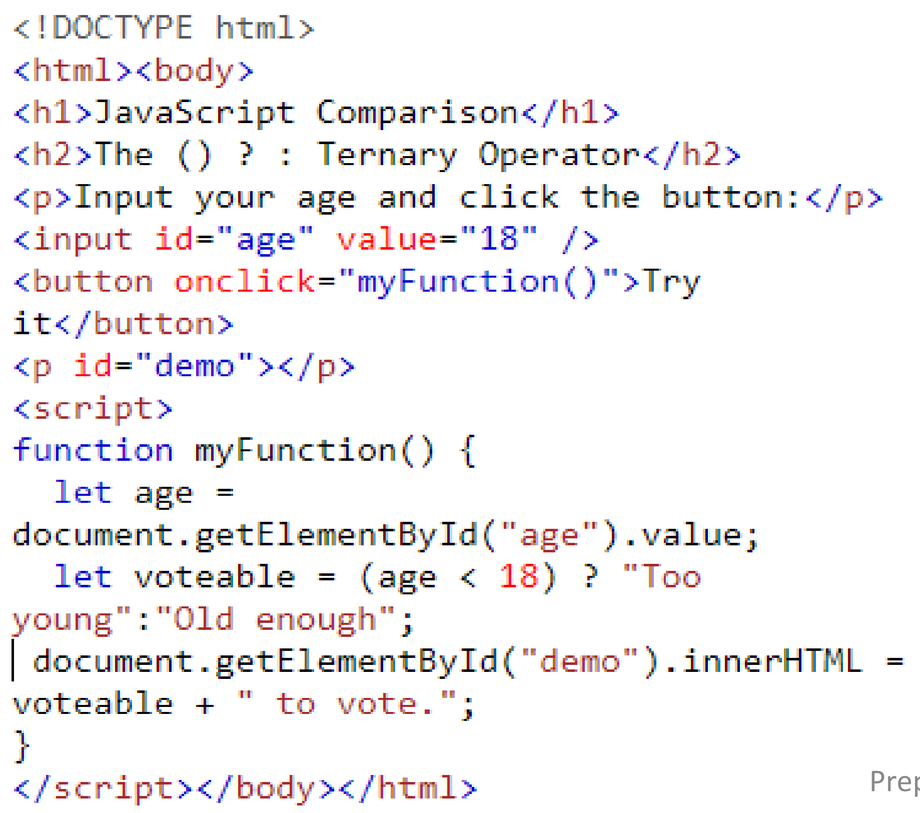
```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript Comparison</h2>
<p>The NOT operator (!) returns
true for false statements and
false for true statements.</p>
<p id="demo"></p>
<script>
let x = 6;
let y = 3;
document.getElementById("demo").
innerHTML = !(x === y) + "<br>"
+ !(x > y);
</script></body></html>
```

JavaScript Comparison

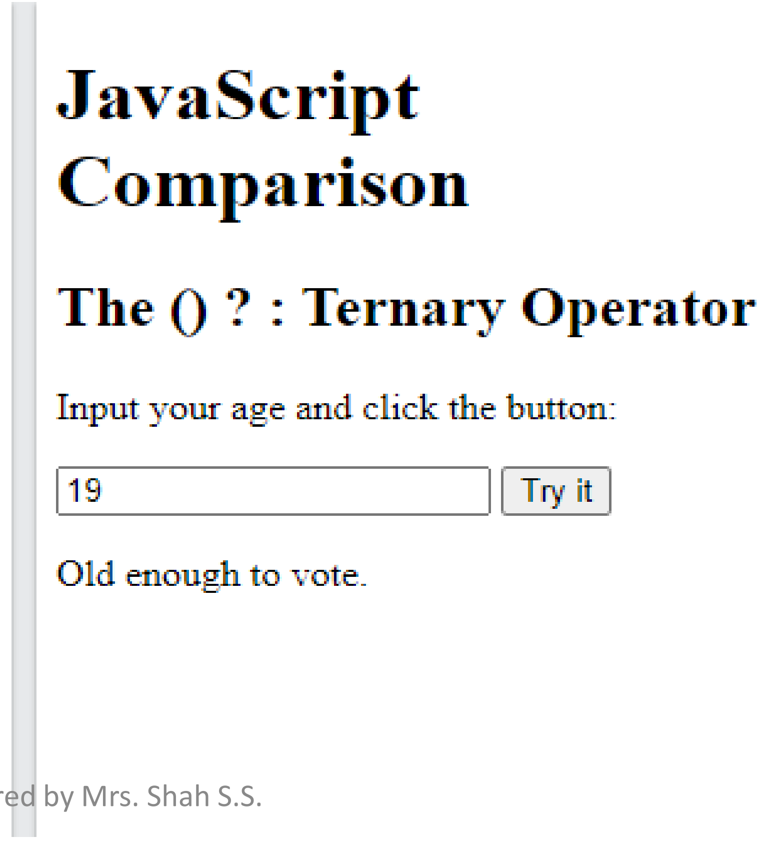
The NOT operator (!) returns true for false statements and false for true statements.

```
true
false
```

- Conditional (Ternary) Operator
- JavaScript also contains a conditional operator that assigns a value to a variable based on some condition.
- Syntax
- *variablename* = (*condition*) ? *value1*:*value2*



```
<!DOCTYPE html>
<html><body>
<h1>JavaScript Comparison</h1>
<h2>The () ? : Ternary Operator</h2>
<p>Input your age and click the button:</p>
<input id="age" value="18" />
<button onclick="myFunction()">Try
it</button>
<p id="demo"></p>
<script>
function myFunction() {
  let age =
document.getElementById("age").value;
  let voteable = (age < 18) ? "Too
young":"Old enough";
 document.getElementById("demo").innerHTML =
voteable + " to vote.";
}
</script></body></html>
```



JavaScript Comparison

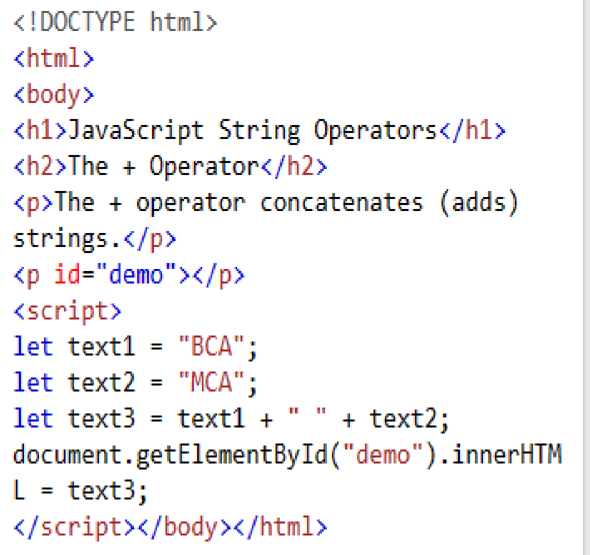
The () ? : Ternary Operator

Input your age and click the button:

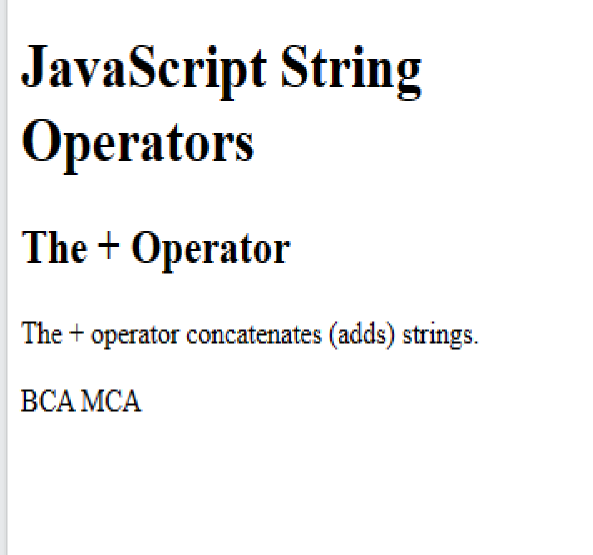
19 Try it

Old enough to vote.

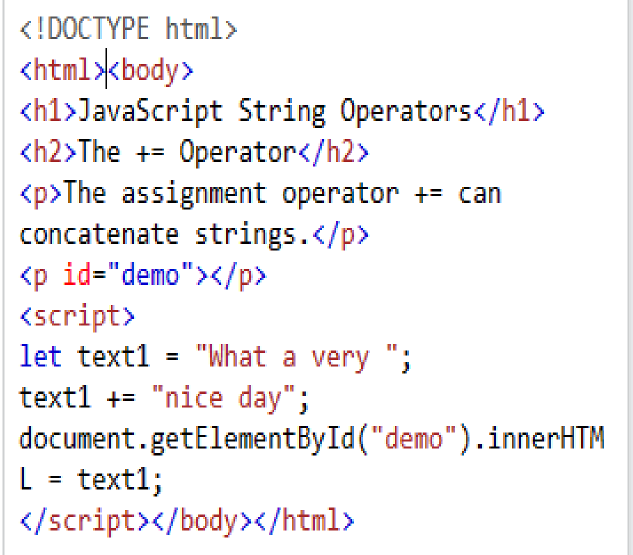
Adding JavaScript Strings



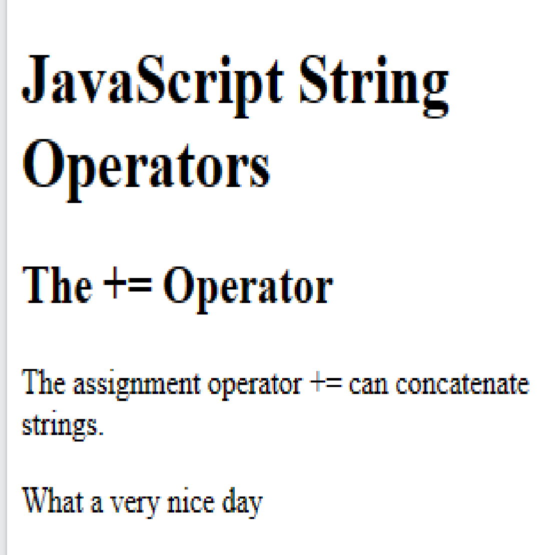
```
<!DOCTYPE html>
<html>
<body>
<h1>JavaScript String Operators</h1>
<h2>The + Operator</h2>
<p>The + operator concatenates (adds)
strings.</p>
<p id="demo"></p>
<script>
let text1 = "BCA";
let text2 = "MCA";
let text3 = text1 + " " + text2;
document.getElementById("demo").innerHTM
L = text3;
</script></body></html>
```



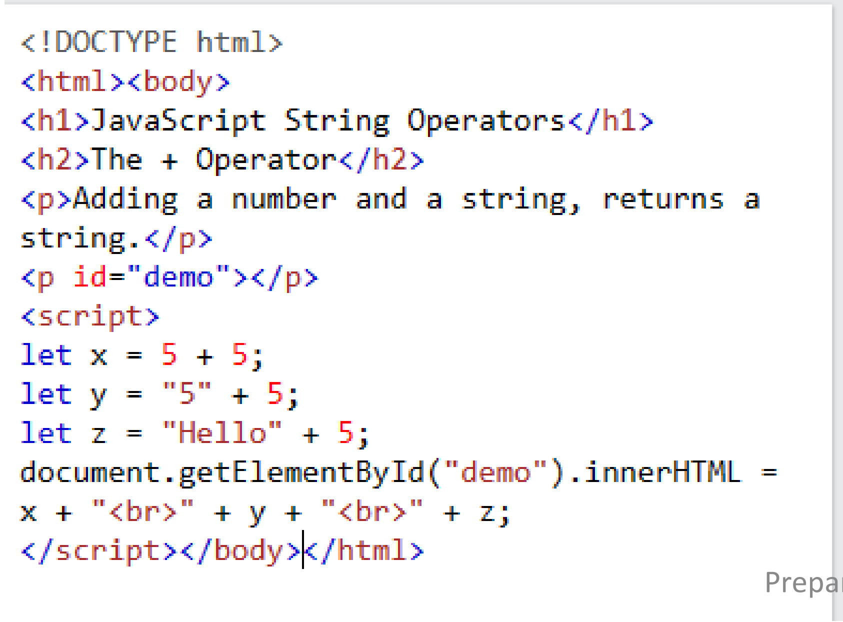
```
JavaScript String
Operators

The + Operator

The + operator concatenates (adds) strings.

BCA MCA
```



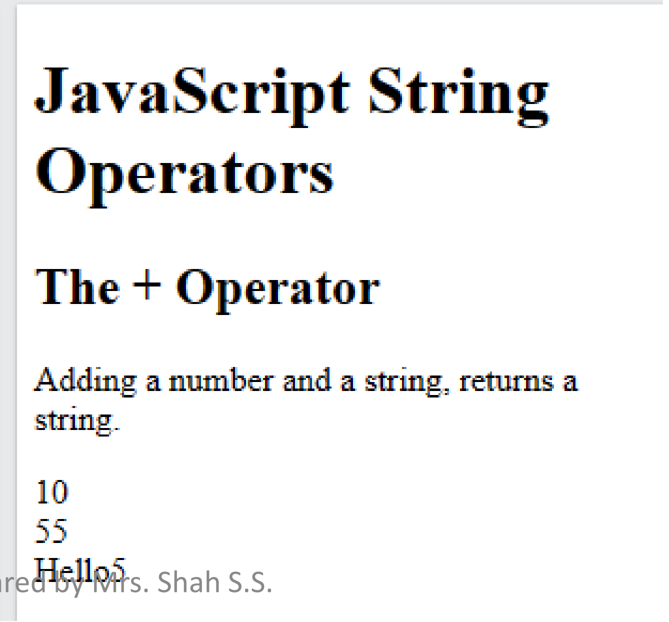
```
<!DOCTYPE html>
<html><body>
<h1>JavaScript String Operators</h1>
<h2>The += Operator</h2>
<p>The assignment operator += can
concatenate strings.</p>
<p id="demo"></p>
<script>
let text1 = "What a very ";
text1 += "nice day";
document.getElementById("demo").innerHTM
L = text1;
</script></body></html>
```



```
JavaScript String
Operators

The += Operator

The assignment operator += can concatenate
strings.

What a very nice day
```



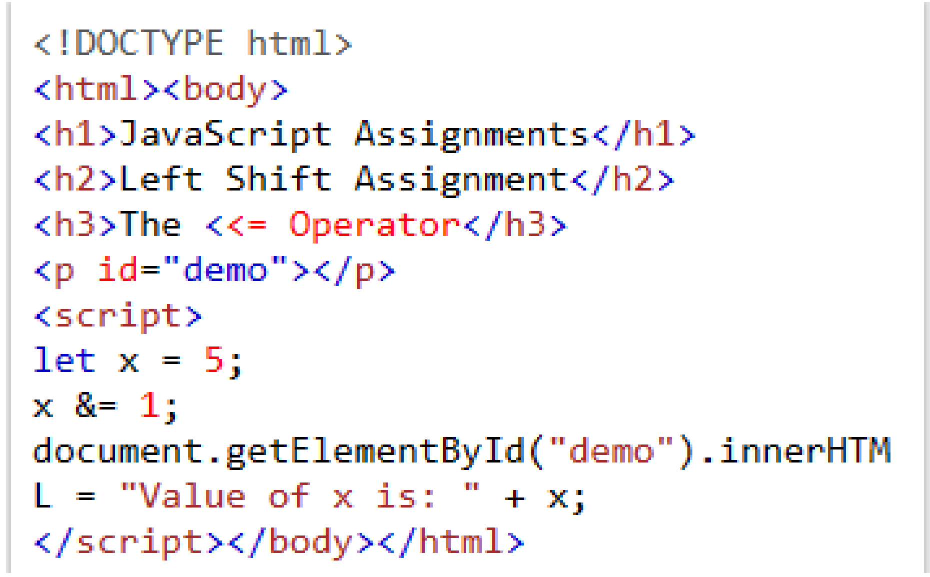
```
<!DOCTYPE html>
<html><body>
<h1>JavaScript String Operators</h1>
<h2>The + Operator</h2>
<p>Adding a number and a string, returns a
string.</p>
<p id="demo"></p>
<script>
let x = 5 + 5;
let y = "5" + 5;
let z = "Hello" + 5;
document.getElementById("demo").innerHTML =
x + "<br>" + y + "<br>" + z;
</script></body></html>
```



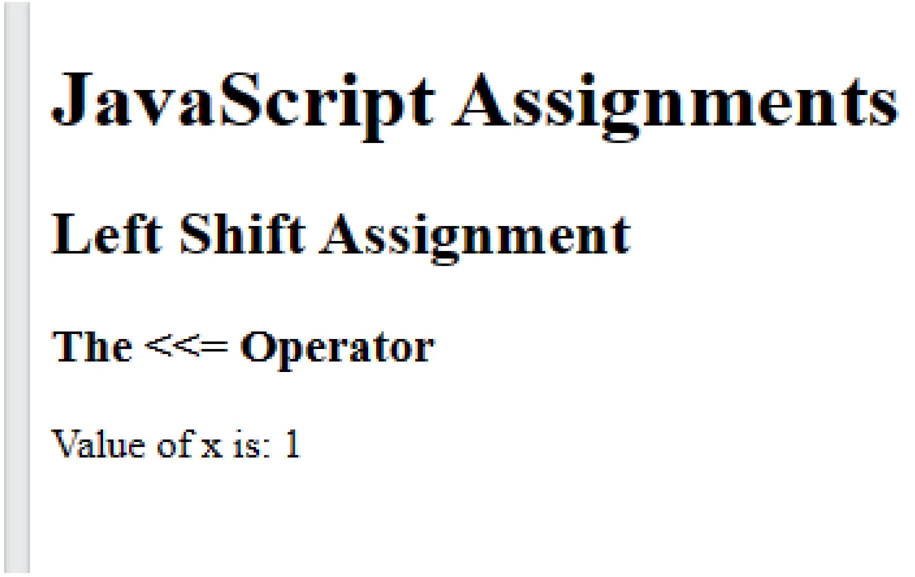
```
JavaScript String
Operators

The + Operator

Adding a number and a string, returns a
string.

10
55
Hello5
```

JavaScript Bitwise Operators

Operator	Description	Example	Same as	Result	Decimal
&	AND	5 & 1	0101 & 0001	0001	1
|	OR	5 | 1	0101 | 0001	0101	5
~	NOT	~ 5	~0101	1010	10
^	XOR	5 ^ 1	0101 ^ 0001	0100	4
<<	left shift	5 << 1	0101 << 1	1010	10
>>	right shift	5 >> 1	0101 >> 1	0010	2
>>>	unsigned right shift	5 >>> 1	0101 >>> 1	0010	2



```
<!DOCTYPE html>
<html><body>
<h1>JavaScript Assignments</h1>
<h2>Left Shift Assignment</h2>
<h3>The <<= Operator</h3>
<p id="demo"></p>
<script>
let x = 5;
x &= 1;
document.getElementById("demo").innerHTM
L = "Value of x is: " + x;
</script></body></html>
```

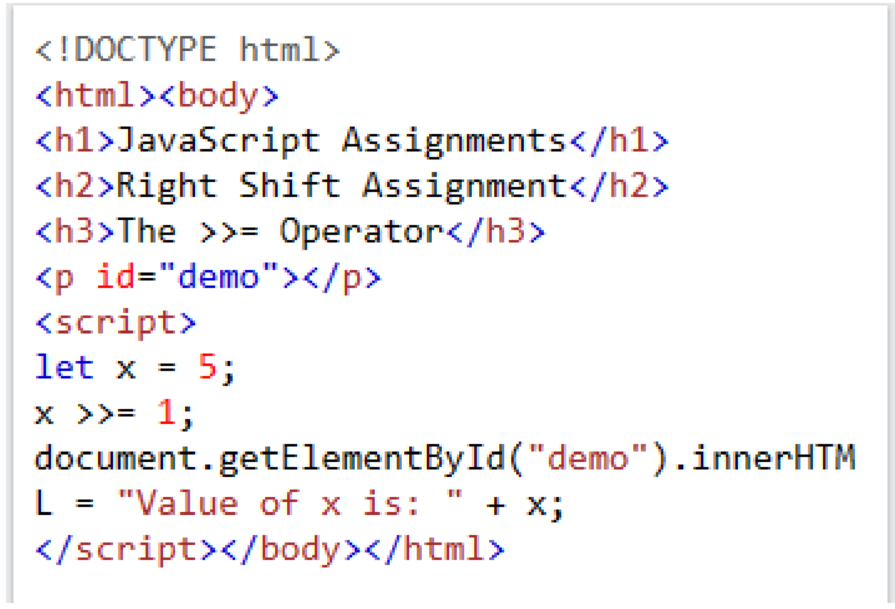


JavaScript Assignments

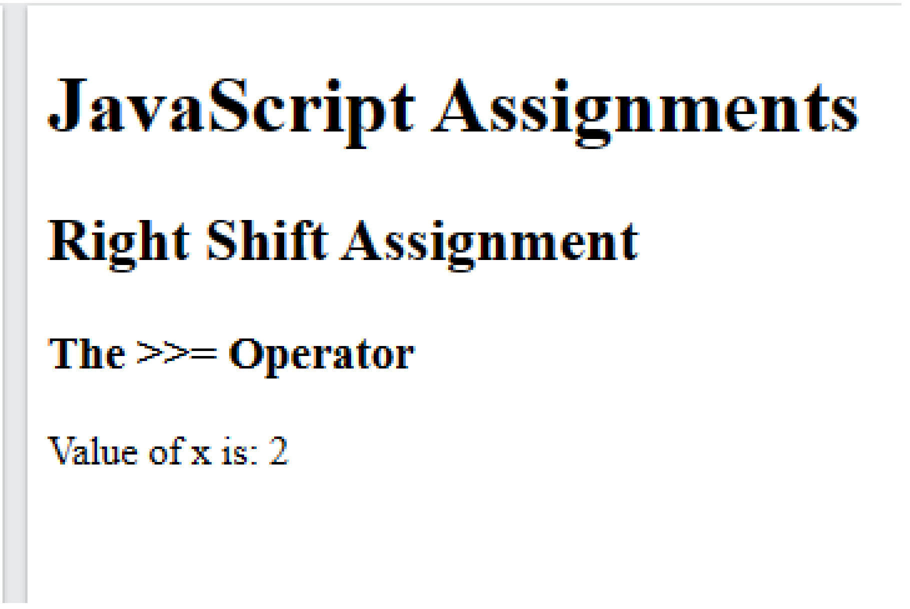
Left Shift Assignment

The <<= Operator

Value of x is: 1



```
<!DOCTYPE html>
<html><body>
<h1>JavaScript Assignments</h1>
<h2>Right Shift Assignment</h2>
<h3>The >>= Operator</h3>
<p id="demo"></p>
<script>
let x = 5;
x >>= 1;
document.getElementById("demo").innerHTM
L = "Value of x is: " + x;
</script></body></html>
```



JavaScript Assignments

Right Shift Assignment

The >>= Operator

Value of x is: 2

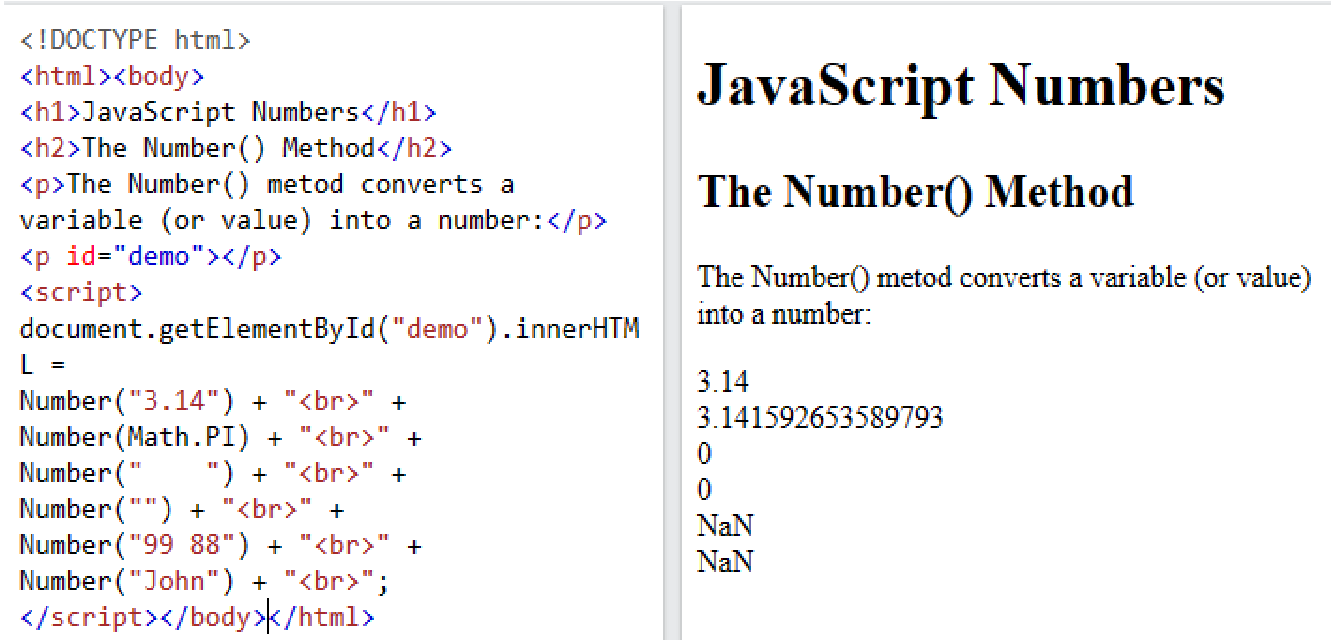
JavaScript Type Operators

Operator	Description
typeof	Returns the type of a variable
instanceof	Returns true if an object is an instance of an object type

- Converting Strings to Numbers
- Converting Numbers to Strings
- Converting Dates to Numbers
- Converting Numbers to Dates
- Converting Booleans to Numbers
- Converting Numbers to Booleans

Converting Strings to Numbers

The global method `Number()` converts a variable (or a value) into a number.
A numeric string (like "3.14") converts to a number (like 3.14).
An empty string (like "") converts to 0.
A non numeric string (like "John") converts to `NaN` (Not a Number).



```
<!DOCTYPE html>
<html><body>
<h1>JavaScript Numbers</h1>
<h2>The Number() Method</h2>
<p>The Number() metod converts a
variable (or value) into a number:</p>
<p id="demo"></p>
<script>
document.getElementById("demo").innerHTM
L =
Number("3.14") + "<br>" +
Number(Math.PI) + "<br>" +
Number("    ") + "<br>" +
Number("") + "<br>" +
Number("99 88") + "<br>" +
Number("John") + "<br>";
</script></body></html>
```

JavaScript Numbers

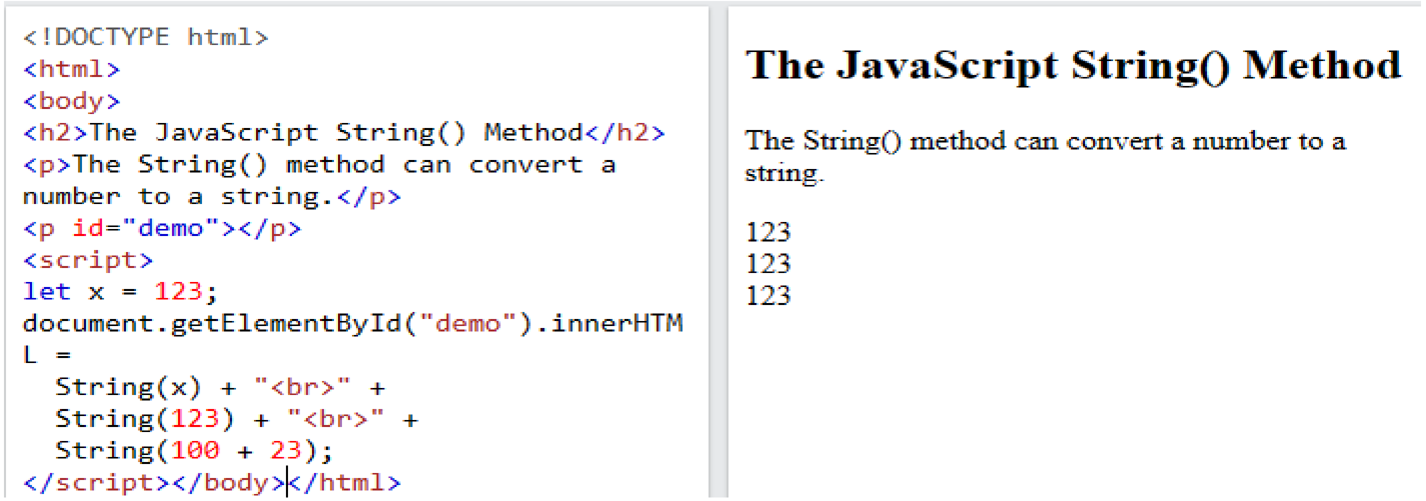
The Number() Method

The Number() metod converts a variable (or value) into a number:

```
3.14
3.141592653589793
0
0
NaN
NaN
```

Converting Numbers to Strings

The global method String() can convert numbers to strings.
It can be used on any type of numbers, literals, variables, or expressions:



```
<!DOCTYPE html>
<html>
<body>
<h2>The JavaScript String() Method</h2>
<p>The String() method can convert a number to a string.</p>
<p id="demo"></p>
<script>
let x = 123;
document.getElementById("demo").innerHTML =
  String(x) + "<br>" +
  String(123) + "<br>" +
  String(100 + 23);
</script></body></html>
```

```
The JavaScript String() Method

The String() method can convert a number to a string.

123
123
123
```

Converting Dates to Numbers

The global method Number() can be used to convert dates to numbers.

```
d = new Date();
Number(d)          // returns 1404568027739
```

Converting Dates to Strings

The global method String() can convert dates to strings.

```
String(Date())  // returns "Thu Jul 17 2014 15:38:19 GMT+0200 (W. Europe Daylight Time)"
```

Converting Booleans to Numbers

The global method Number() can also convert booleans to numbers.

```
Number(false)     // returns 0
Number(true)      // returns 1
```

Converting Booleans to Strings

The global method String() can convert booleans to strings.

```
String(false)      // returns "false"
String(true)       // returns "true"
```

Conditional Statements

In JavaScript we have the following conditional statements:

- Use `if` to specify a block of code to be executed, if a specified condition is true
- Use `else` to specify a block of code to be executed, if the same condition is false
- Use `else if` to specify a new condition to test, if the first condition is false
- Use `switch` to specify many alternative blocks of code to be executed

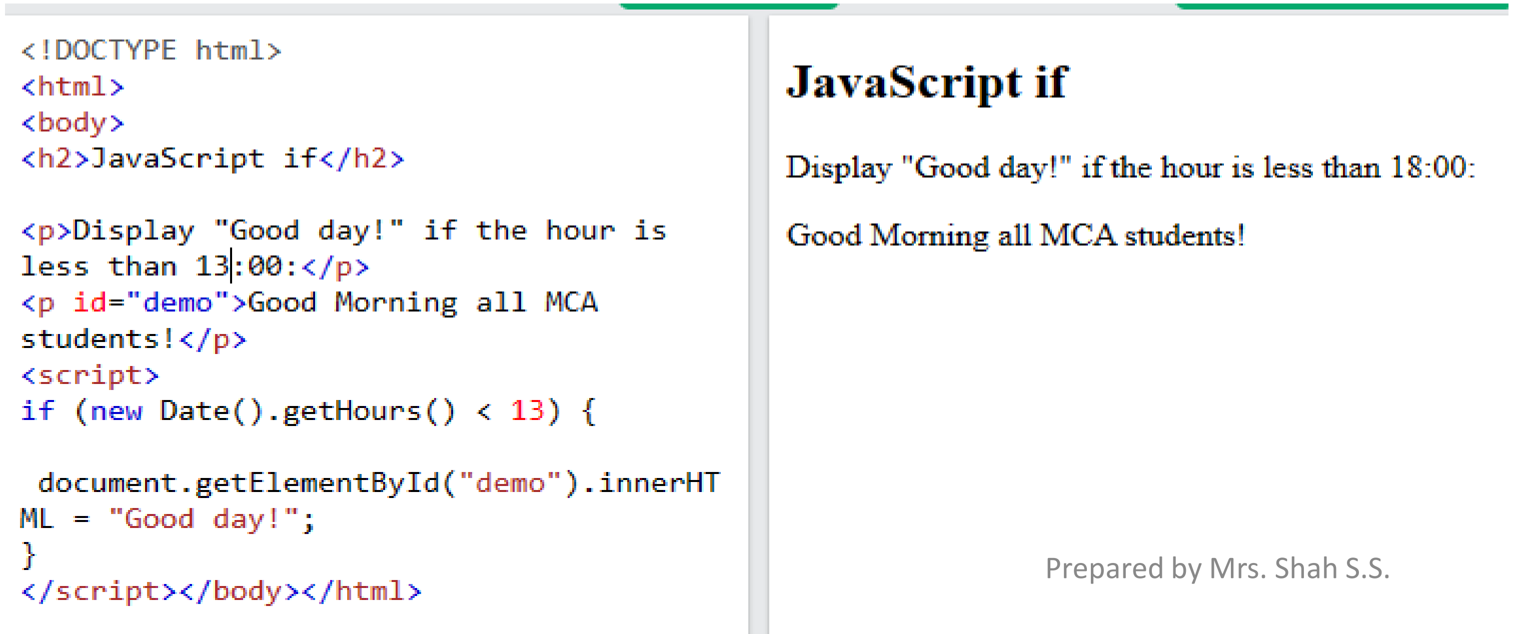
The if Statement

Use the `if` statement to specify a block of JavaScript code to be executed if a condition is true.

Syntax

```
if (condition) {
 //  block of code to be executed if the condition is true
}
```

Note that `if` is in lowercase letters. Uppercase letters (If or IF) will generate a JavaScript error.



```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript if</h2>

<p>Display "Good day!" if the hour is
less than 13:00:</p>
<p id="demo">Good Morning all MCA
students!</p>
<script>
if (new Date().getHours() < 13) {

 document.getElementById("demo").innerHT
ML = "Good day!";
}
</script></body></html>
```

JavaScript if

Display "Good day!" if the hour is less than 18:00:

Good Morning all MCA students!

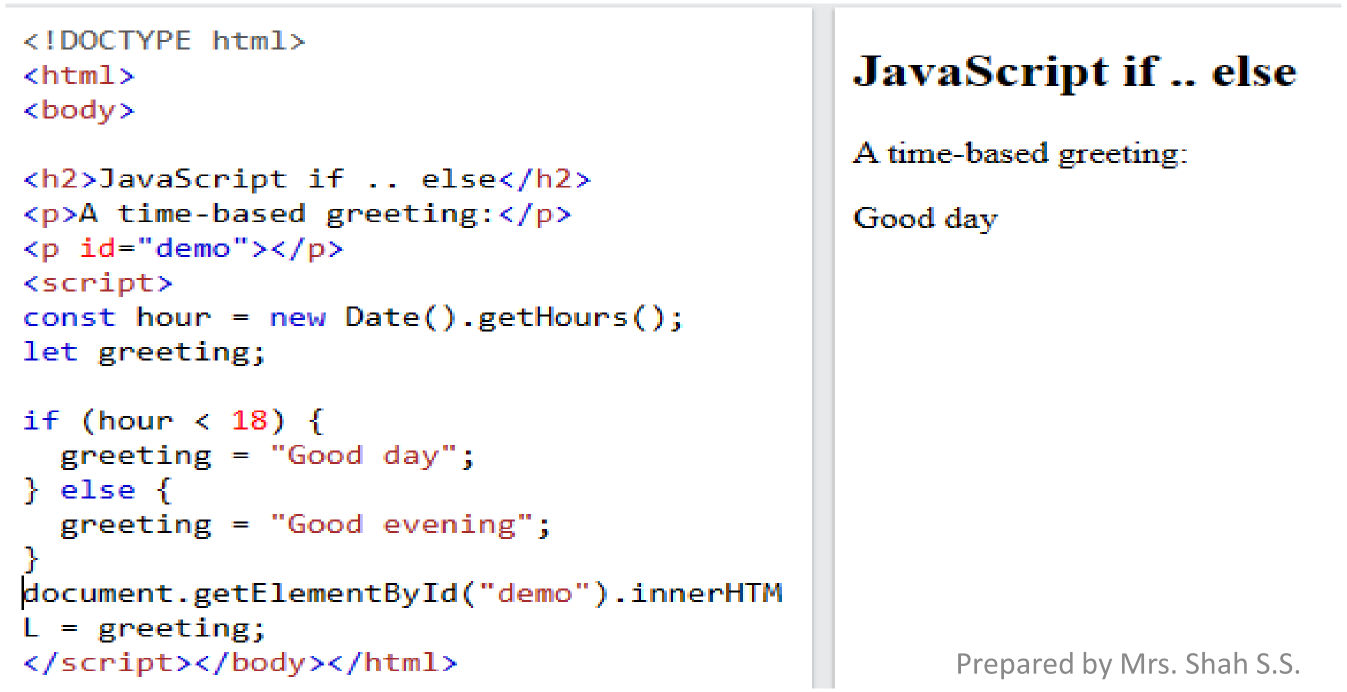
The else Statement

Use the `else` statement to specify a block of code to be executed if the condition is false.

```
if (condition) {
 //  block of code to be executed if the condition is true
} else {
 //  block of code to be executed if the condition is false
}
```

Example

If the hour is less than 18, create a "Good day" greeting, otherwise "Good evening":



```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript if .. else</h2>
<p>A time-based greeting:</p>
<p id="demo"></p>
<script>
const hour = new Date().getHours();
let greeting;

if (hour < 18) {
  greeting = "Good day";
} else {
  greeting = "Good evening";
}
document.getElementById("demo").innerHTM
L = greeting;
</script></body></html>
```

```
JavaScript if .. else

A time-based greeting:

Good day
```

The else if Statement

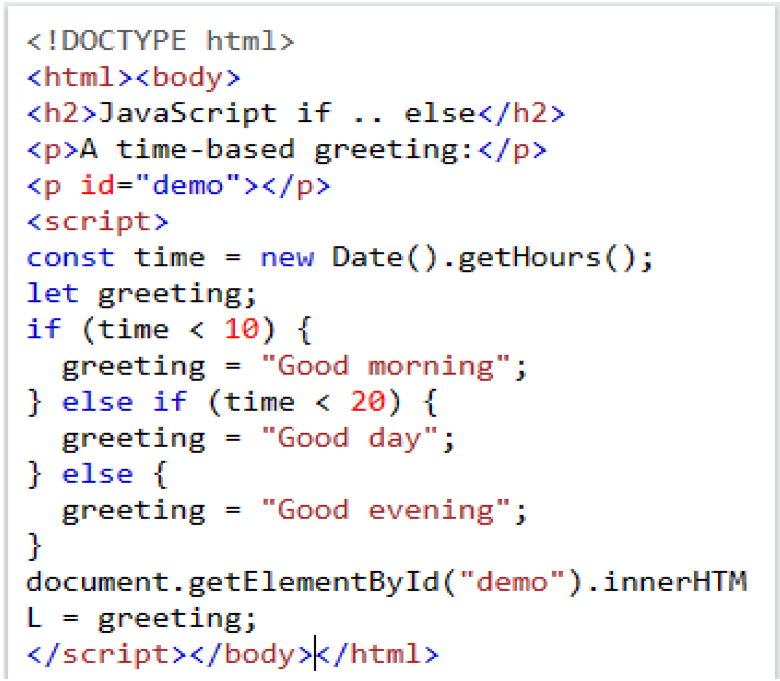
Use the `else if` statement to specify a new condition if the first condition is false.

Syntax

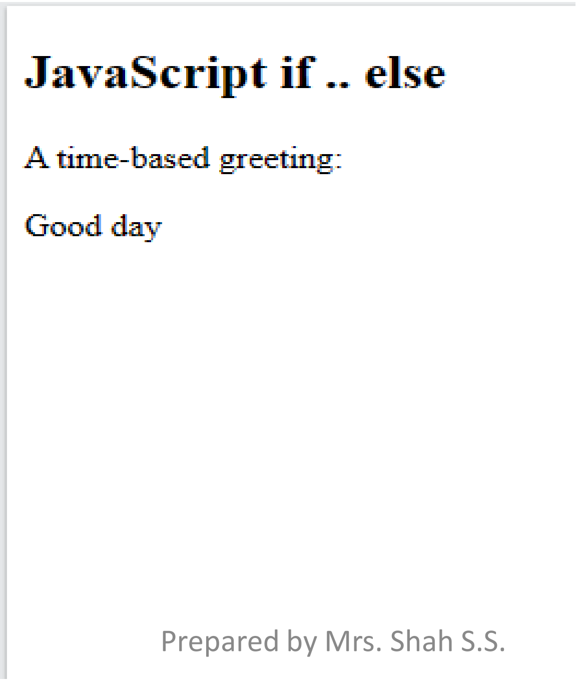
```
if (condition1) {
 //  block of code to be executed if condition1 is true
} else if (condition2) {
 //  block of code to be executed if the condition1 is false and condition2 is true
} else {
 //  block of code to be executed if the condition1 is false and condition2 is false
}
```

Example

If time is less than 10:00, create a "Good morning" greeting, if not, but time is less than 20:00, create a "Good day" greeting, otherwise a "Good evening":



```
<!DOCTYPE html>
<html><body>
<h2>JavaScript if .. else</h2>
<p>A time-based greeting:</p>
<p id="demo"></p>
<script>
const time = new Date().getHours();
let greeting;
if (time < 10) {
  greeting = "Good morning";
} else if (time < 20) {
  greeting = "Good day";
} else {
  greeting = "Good evening";
}
document.getElementById("demo").innerHTM
L = greeting;
</script></body></html>
```



```
JavaScript if .. else

A time-based greeting:

Good day
```

The JavaScript Switch Statement

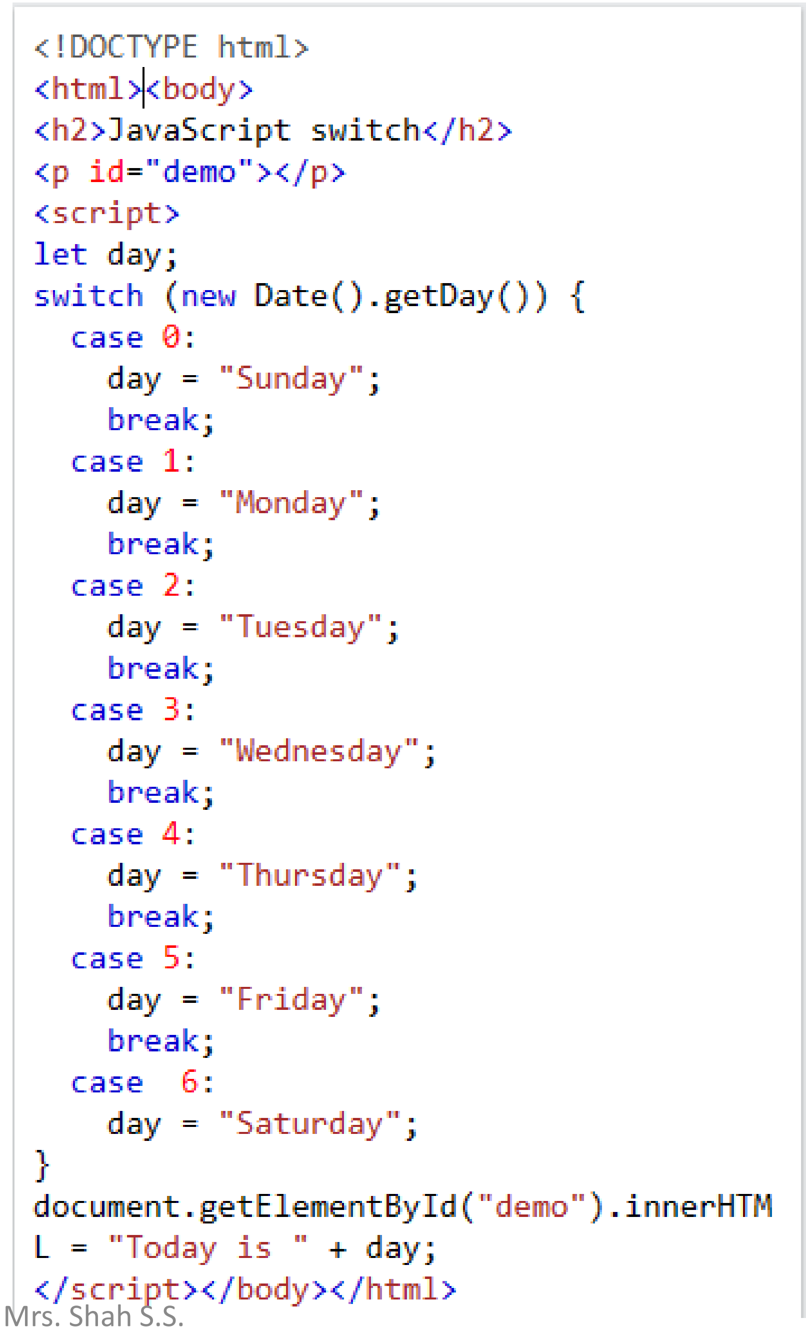
Use the `switch` statement to select one of many code blocks to be executed.

Syntax

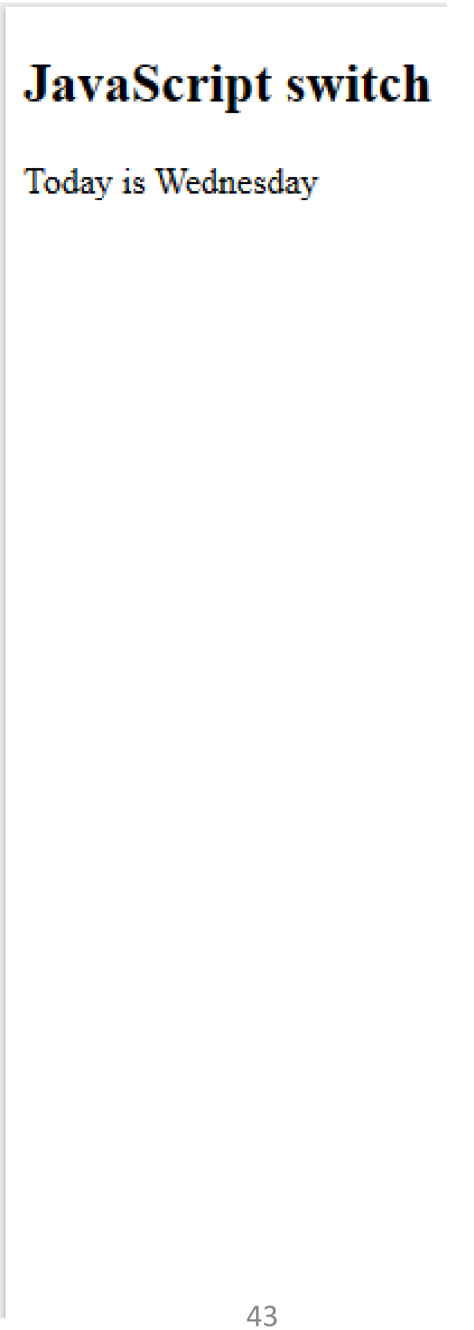
```
switch(expression) {
 case x:
  // code block
  break;
 case y:
  // code block
  break;
 default:
   // code block
}
```

This is how it works:

- *The switch expression is evaluated once.*
- *The value of the expression is compared with the values of each case.*
- *If there is a match, the associated block of code is executed.*
- *If there is no match, the default code block is executed.*



```
<!DOCTYPE html>
<html><body>
<h2>JavaScript switch</h2>
<p id="demo"></p>
<script>
let day;
switch (new Date().getDay()) {
  case 0:
    day = "Sunday";
    break;
  case 1:
    day = "Monday";
    break;
  case 2:
    day = "Tuesday";
    break;
  case 3:
    day = "Wednesday";
    break;
  case 4:
    day = "Thursday";
    break;
  case 5:
    day = "Friday";
    break;
  case  6:
    day = "Saturday";
}
document.getElementById("demo").innerHTM
L = "Today is " + day;
</script></body></html>
```



```
JavaScript switch

Today is Wednesday
```

JavaScript Loops

Loops are handy, if you want to run the same code over and over again, each time with a different value.

Different Kinds of Loops

JavaScript supports different kinds of loops:

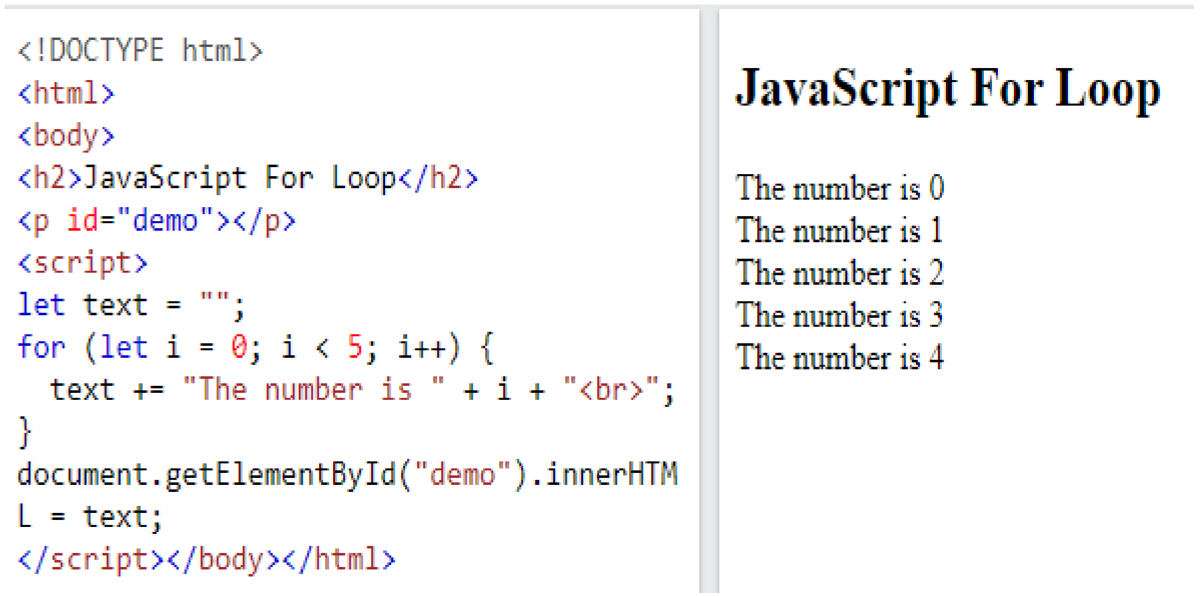
- `for` - loops through a block of code a number of times
- `for/in` - loops through the properties of an object
- `for/of` - loops through the values of an iterable object
- `while` - loops through a block of code while a specified condition is true
- `do/while` - also loops through a block of code while a specified condition is true

The For Loop

The for statement creates a loop with 3 optional expressions:

```
for (expression 1; expression 2; expression 3) {
  // code block to be executed
}
```

Expression 1 is executed (one time) before the execution of the code block.
Expression 2 defines the condition for executing the code block.
Expression 3 is executed (every time) after the code block has been executed.



```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript For Loop</h2>
<p id="demo"></p>
<script>
let text = "";
for (let i = 0; i < 5; i++) {
  text += "The number is " + i + "<br>";
}
document.getElementById("demo").innerHTM
L = text;
</script></body></html>
```

JavaScript For Loop

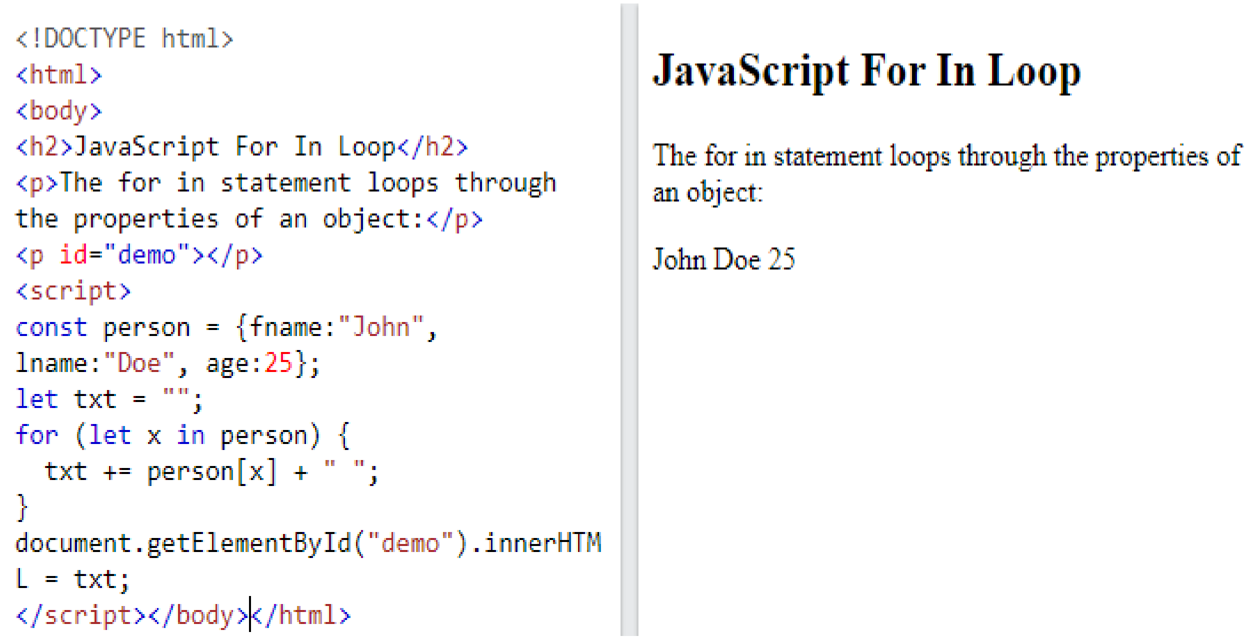
```
The number is 0
The number is 1
The number is 2
The number is 3
The number is 4
```

The For In Loop

The JavaScript for in statement loops through the properties of an Object:

Syntax

```
for (key in object) {
  // code block to be executed
}
```



```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript For In Loop</h2>
<p>The for in statement loops through
the properties of an object:</p>
<p id="demo"></p>
<script>
const person = {fname:"John",
lname:"Doe", age:25};
let txt = "";
for (let x in person) {
  txt += person[x] + " ";
}
document.getElementById("demo").innerHTM
L = txt;
</script></body></html>
```

JavaScript For In Loop

The for in statement loops through the properties of an object:

John Doe 25

The For Of Loop

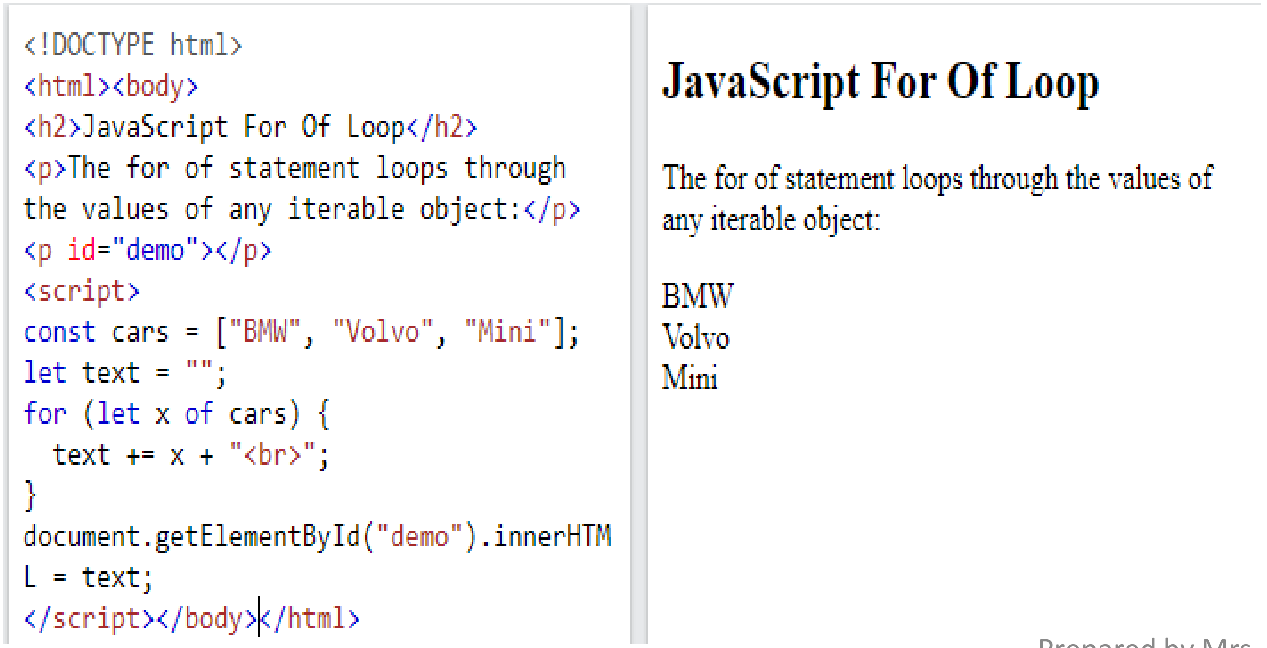
The JavaScript `for of` statement loops through the values of an iterable object.
It lets you loop over iterable data structures such as Arrays, Strings, Maps, NodeLists, and more:

Syntax

```
for (variable of iterable) {
  // code block to be executed
}
```

variable - For every iteration the value of the next property is assigned to the variable. *Variable* can be declared with `const`, `let`, or `var`.
iterable - An object that has iterable properties.

Looping over an Array



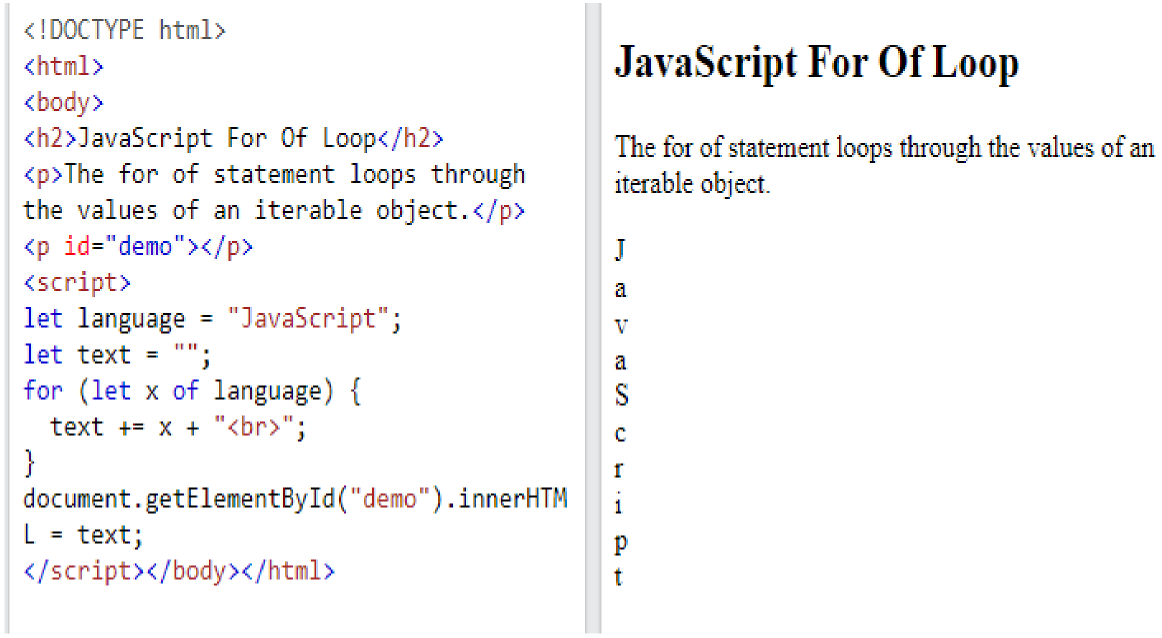
```
<!DOCTYPE html>
<html><body>
<h2>JavaScript For Of Loop</h2>
<p>The for of statement loops through
the values of any iterable object:</p>
<p id="demo"></p>
<script>
const cars = ["BMW", "Volvo", "Mini"];
let text = "";
for (let x of cars) {
  text += x + "<br>";
}
document.getElementById("demo").innerHTM
L = text;
</script></body></html>
```

JavaScript For Of Loop

The for of statement loops through the values of any iterable object:

```
BMW
Volvo
Mini
```

Looping over a String



```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript For Of Loop</h2>
<p>The for of statement loops through
the values of an iterable object.</p>
<p id="demo"></p>
<script>
let language = "JavaScript";
let text = "";
for (let x of language) {
  text += x + "<br>";
}
document.getElementById("demo").innerHTM
L = text;
</script></body></html>
```

JavaScript For Of Loop

The for of statement loops through the values of an iterable object.

```
J
a
v
a
S
c
r
i
p
t
```

The While Loop

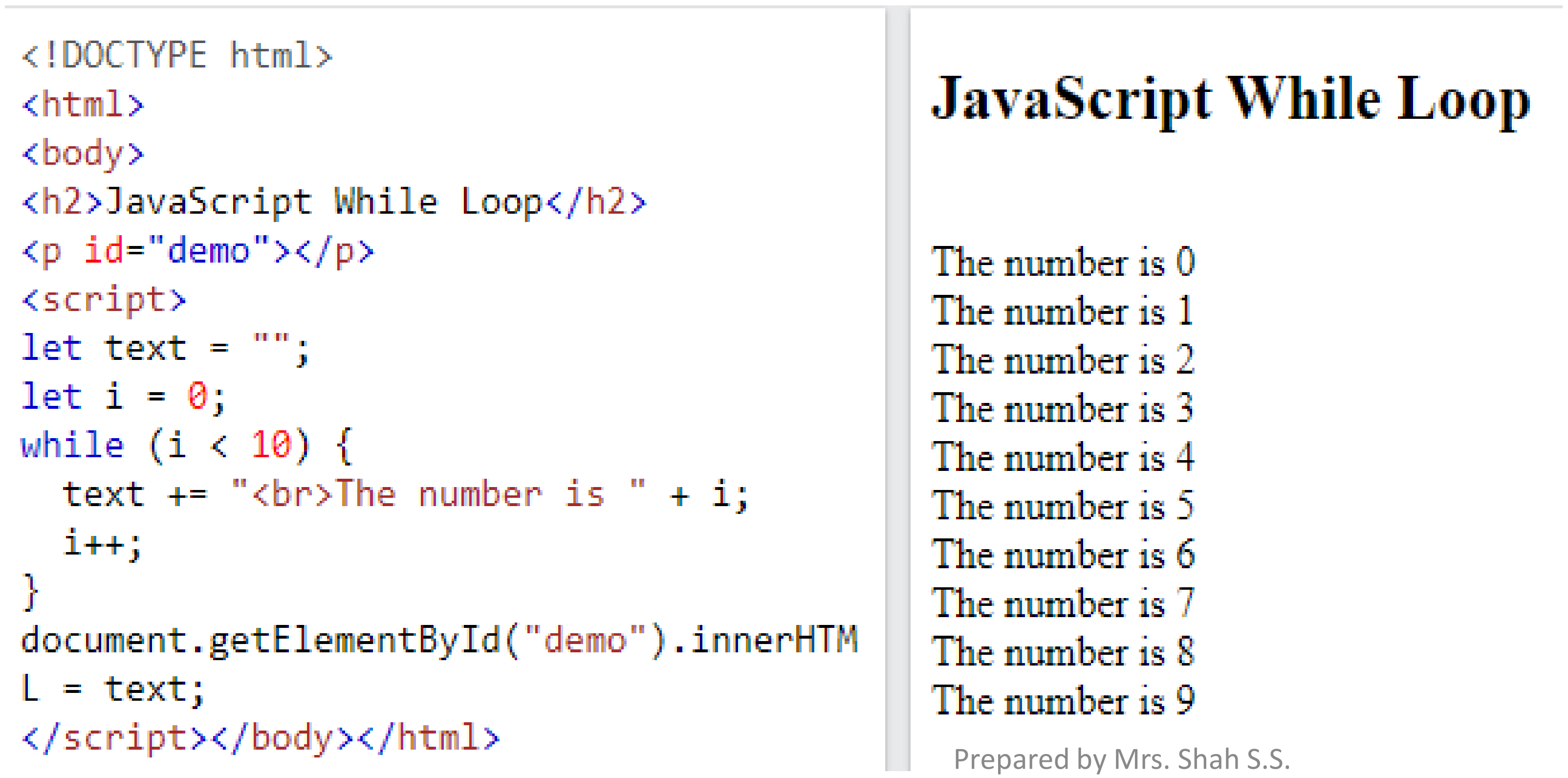
The `while` loop loops through a block of code as long as a specified condition is true.

Syntax

```
while (condition) {
  // code block to be executed
}
```

Example

In the following example, the code in the loop will run, over and over again, as long as a variable (i) is less than 10:



```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript While Loop</h2>
<p id="demo"></p>
<script>
let text = "";
let i = 0;
while (i < 10) {
  text += "<br>The number is " + i;
  i++;
}
document.getElementById("demo").innerHTM
L = text;
</script></body></html>
```

JavaScript While Loop

```
The number is 0
The number is 1
The number is 2
The number is 3
The number is 4
The number is 5
The number is 6
The number is 7
The number is 8
The number is 9
```

The Do While Loop

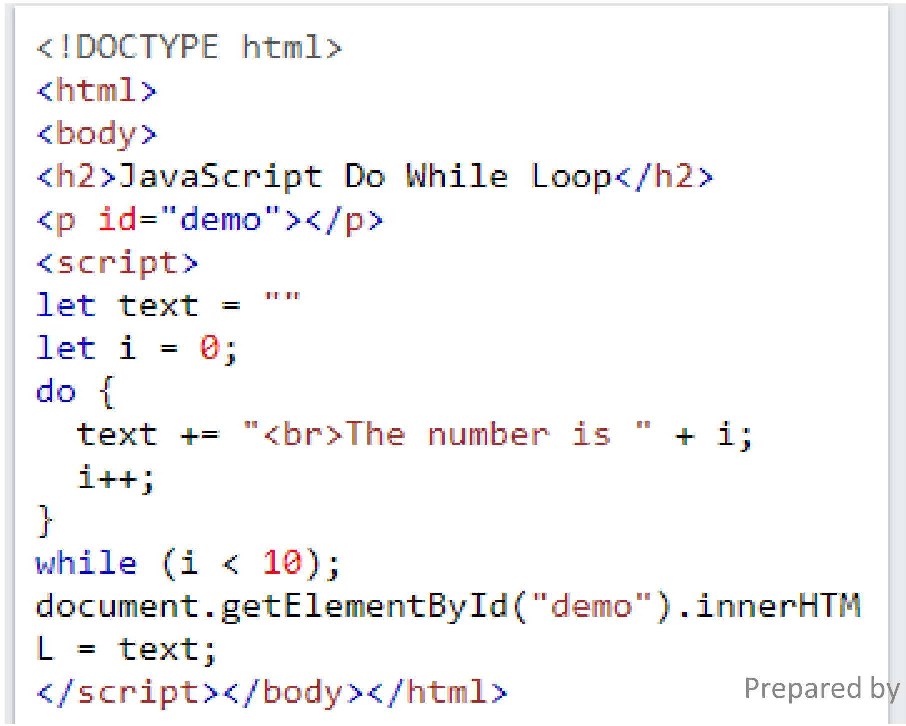
The `do while` loop is a variant of the while loop. This loop will execute the code block once, before checking if the condition is true, then it will repeat the loop as long as the condition is true.

Syntax

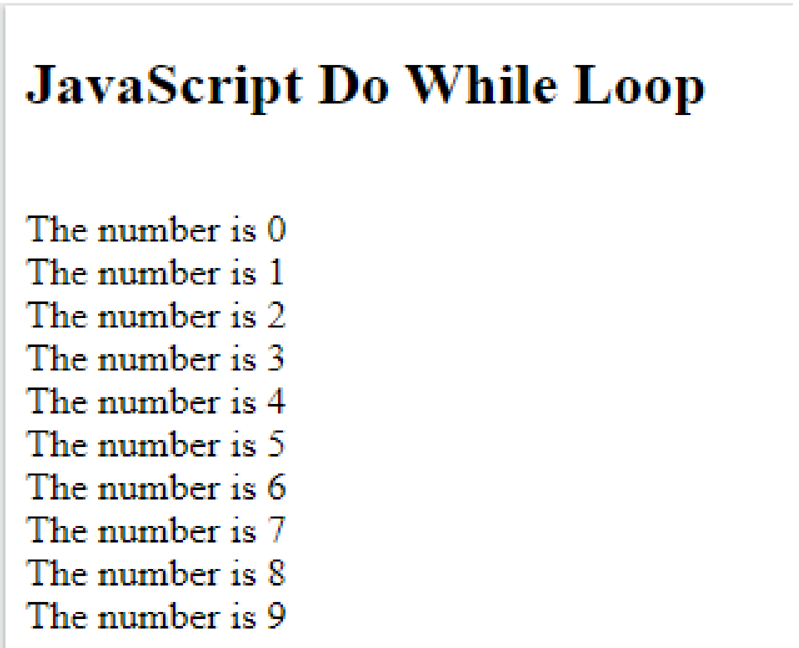
```
do {
  // code block to be executed
}
while (condition);
```

Example

The example below uses a `do while` loop. The loop will always be executed at least once, even if the condition is false, because the code block is executed before the condition is tested:



```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript Do While Loop</h2>
<p id="demo"></p>
<script>
let text = ""
let i = 0;
do {
  text += "<br>The number is " + i;
  i++;
}
while (i < 10);
document.getElementById("demo").innerHTM
L = text;
</script></body></html>
```



```
JavaScript Do While Loop

The number is 0
The number is 1
The number is 2
The number is 3
The number is 4
The number is 5
The number is 6
The number is 7
The number is 8
The number is 9
```

popup boxes: Alert box, Confirm box, and Prompt box.

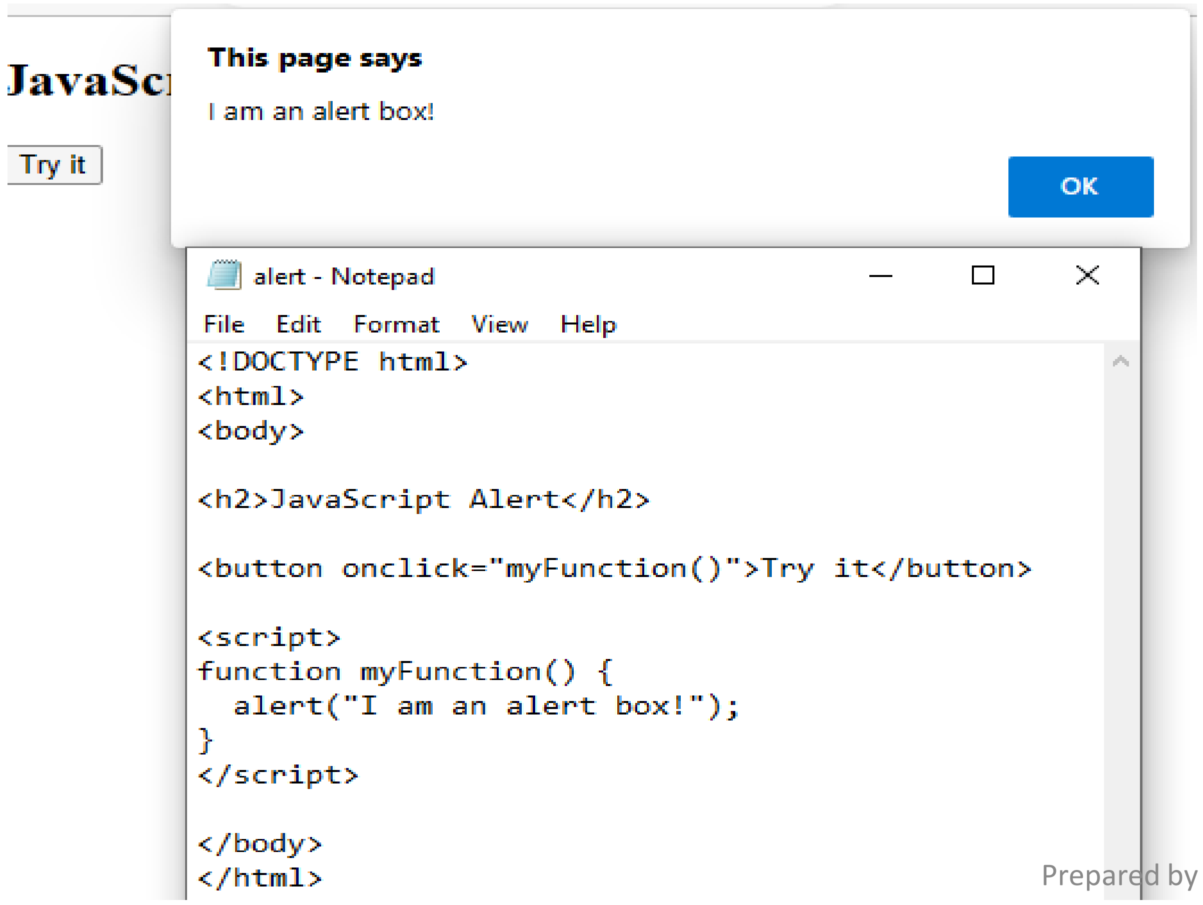
Alert Box

An alert box is often used if you want to make sure information comes through to the user.
When an alert box pops up, the user will have to click "OK" to proceed.

Syntax

`window.alert("`*`sometext`*`");`

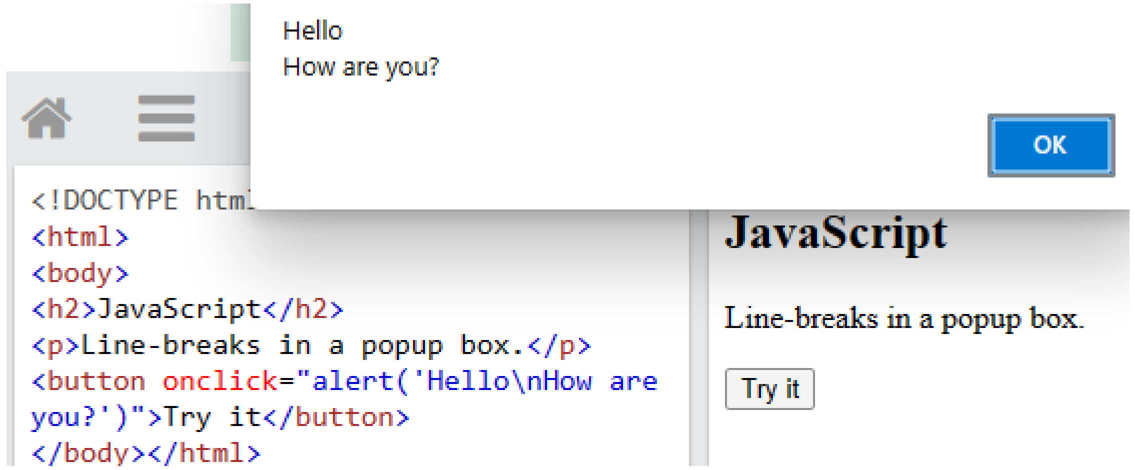
The `window.alert()` method can be written without the window prefix.



```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Alert</h2>

<button onclick="myFunction()">Try it</button>

<script>
function myFunction() {
  alert("I am an alert box!");
}
</script>

</body>
</html>
```

Line Breaks

To display line breaks inside a popup box, use a back-slash followed by the character n.



```
<!DOCTYPE html
<html>
<body>
<h2>JavaScript</h2>
<p>Line-breaks in a popup box.</p>
<button onclick="alert('Hello\nHow are
you?')">Try it</button>
</body></html>
```

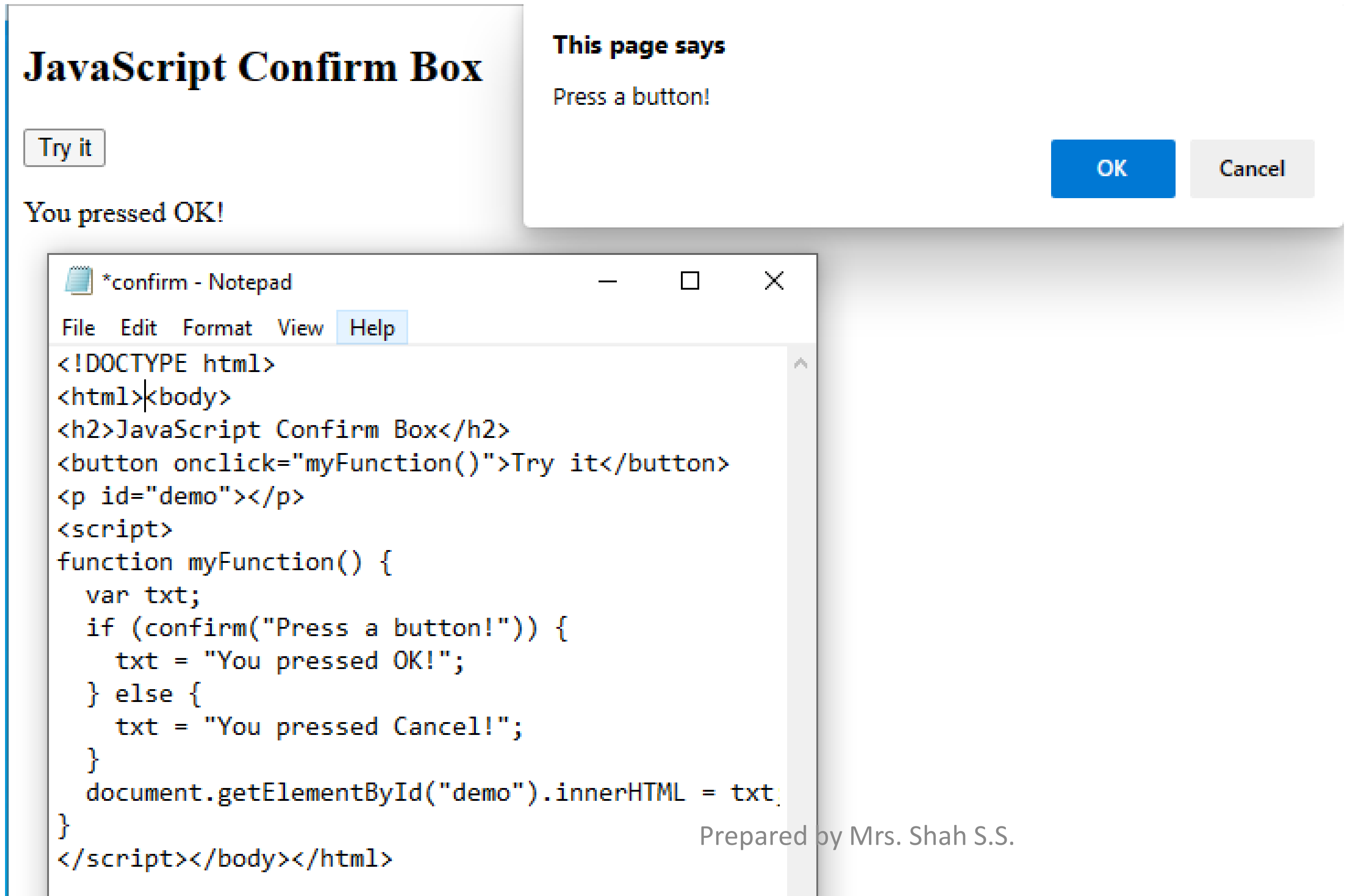
Confirm Box

A confirm box is often used if you want the user to verify or accept something.
When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed.
If the user clicks "OK", the box returns **true**. If the user clicks "Cancel", the box returns **false**.

Syntax

`window.confirm("`*`sometext`*`");`

The `window.confirm()` method can be written without the window prefix.



```
<!DOCTYPE html>
<html><body>
<h2>JavaScript Confirm Box</h2>
<button onclick="myFunction()">Try it</button>
<p id="demo"></p>
<script>
function myFunction() {
  var txt;
  if (confirm("Press a button!")) {
    txt = "You pressed OK!";
  } else {
    txt = "You pressed Cancel!";
  }
  document.getElementById("demo").innerHTML = txt;
}
</script></body></html>
```

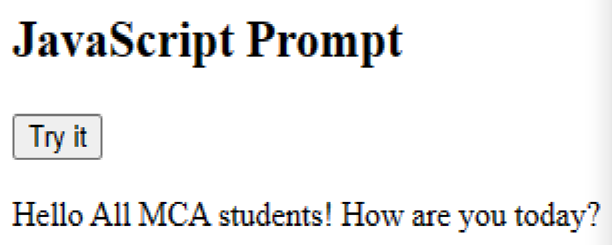
Prompt Box

A prompt box is often used if you want the user to input a value before entering a page.
When a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed after entering an input value.
If the user clicks "OK" the box returns the input value. If the user clicks "Cancel" the box returns null.

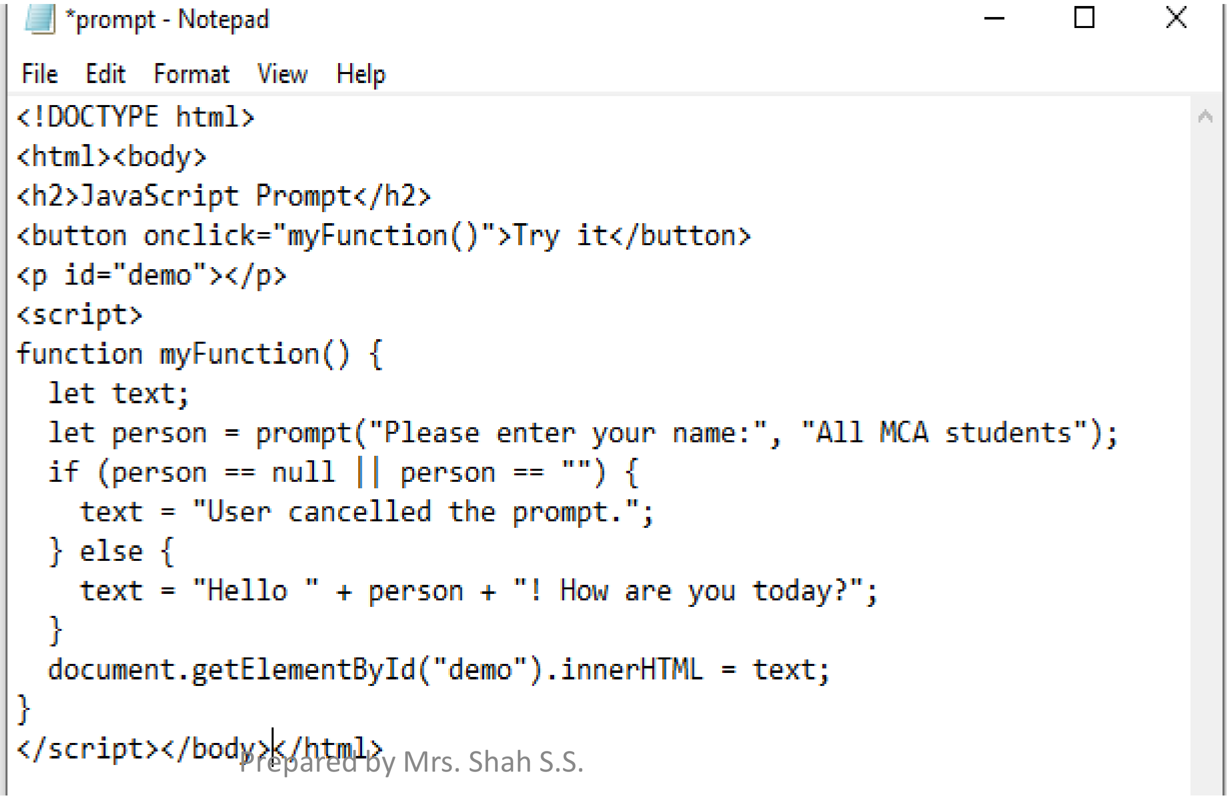
Syntax

`window.prompt("`*`sometext`*`","`*`defaultText`*`");`

The `window.prompt()` method can be written without the window prefix.



Hello All MCA students! How are you today?



```
<!DOCTYPE html>
<html><body>
<h2>JavaScript Prompt</h2>
<button onclick="myFunction()">Try it</button>
<p id="demo"></p>
<script>
function myFunction() {
  let text;
  let person = prompt("Please enter your name:", "All MCA students");
  if (person == null || person == "") {
    text = "User cancelled the prompt.";
  } else {
    text = "Hello " + person + "! How are you today?";
  }
  document.getElementById("demo").innerHTML = text;
}
</script></body></html>
```

JavaScript Events

HTML events are **"things"** that happen to HTML elements.
When JavaScript is used in HTML pages, JavaScript can **"react"** on these events.

HTML Events

An HTML event can be something the browser does, or something a user does.
Here are some examples of HTML events:

- An HTML web page has finished loading
- An HTML input field was changed
- An HTML button was clicked

HTML allows event handler attributes, **with JavaScript code**, to be added to HTML elements.
With single quotes:

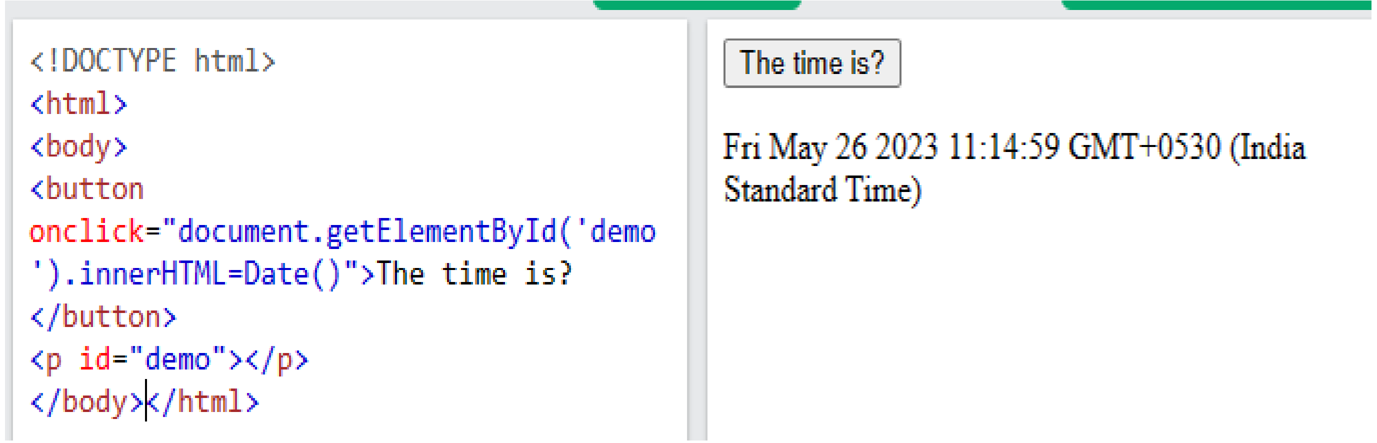
```
<element event='some JavaScript'>
```

With double quotes:

```
<element event="some JavaScript">
```

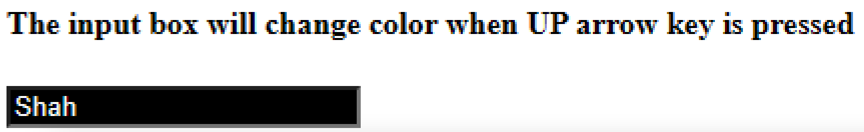
In the following example, an `onclick` attribute (with code), is added to a `<button>` element:

Event	Description
onchange	An HTML element has been changed
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

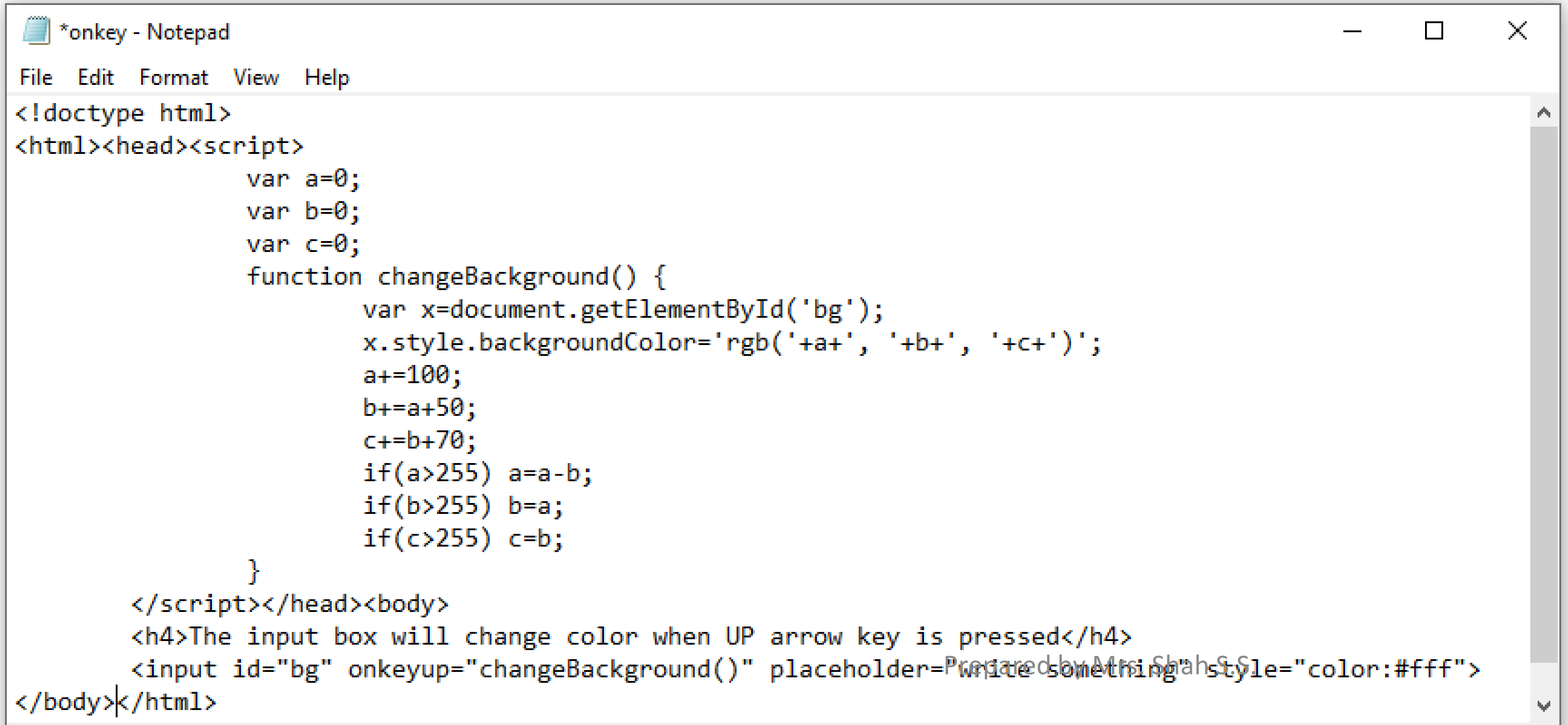


```
<!DOCTYPE html>
<html>
<body>
<button
onclick="document.getElementById('demo
').innerHTML=Date()">The time is?
</button>
<p id="demo"></p>
</body></html>
```

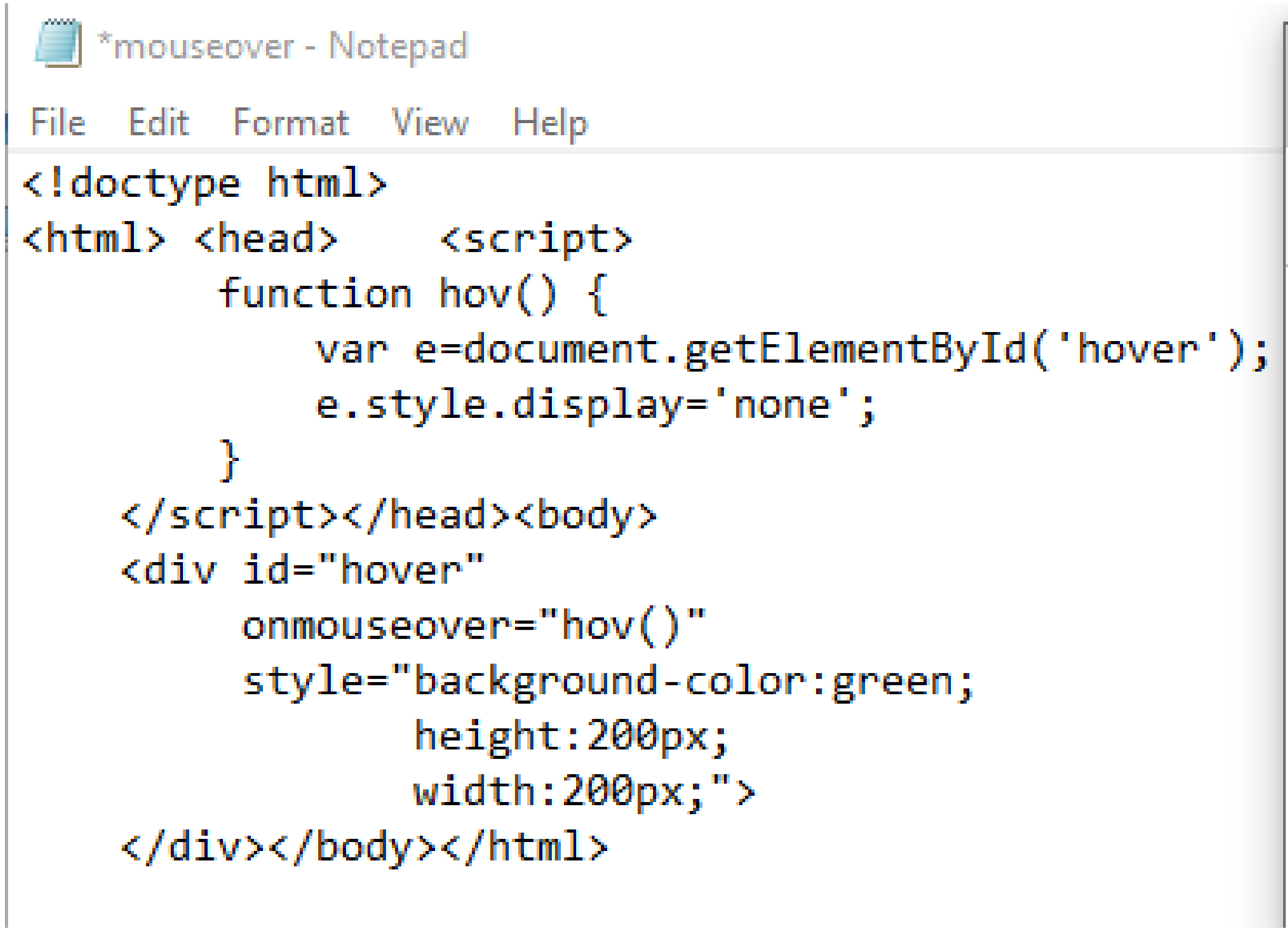
```
Fri May 26 2023 11:14:59 GMT+0530 (India
Standard Time)
```



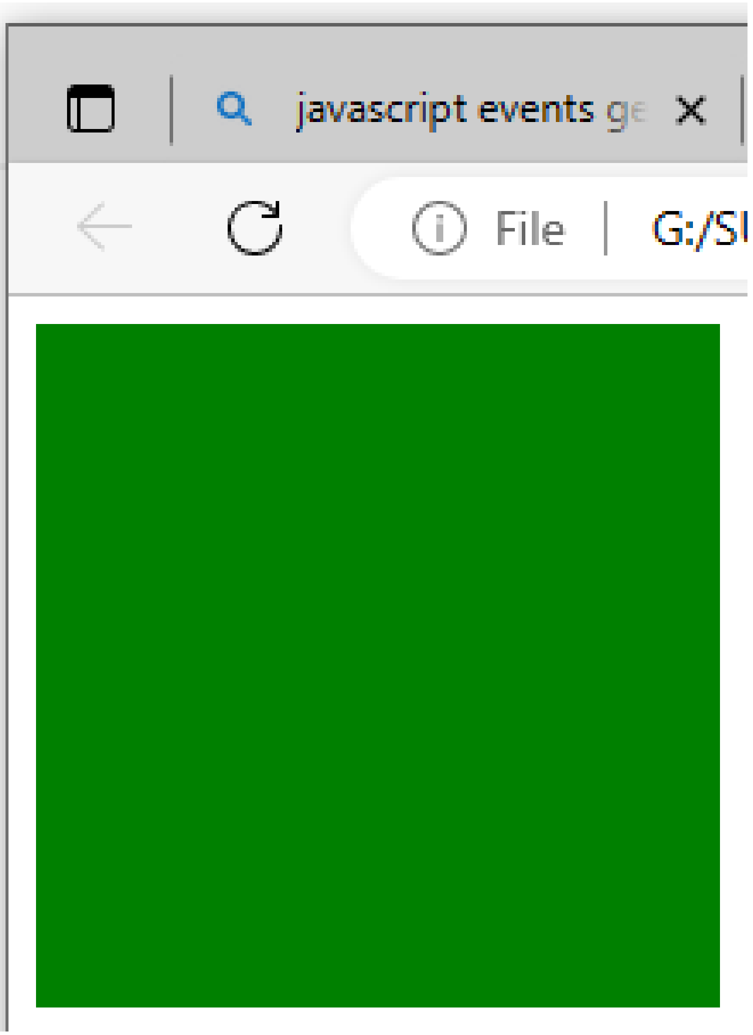
The input box will change color when UP arrow key is pressed

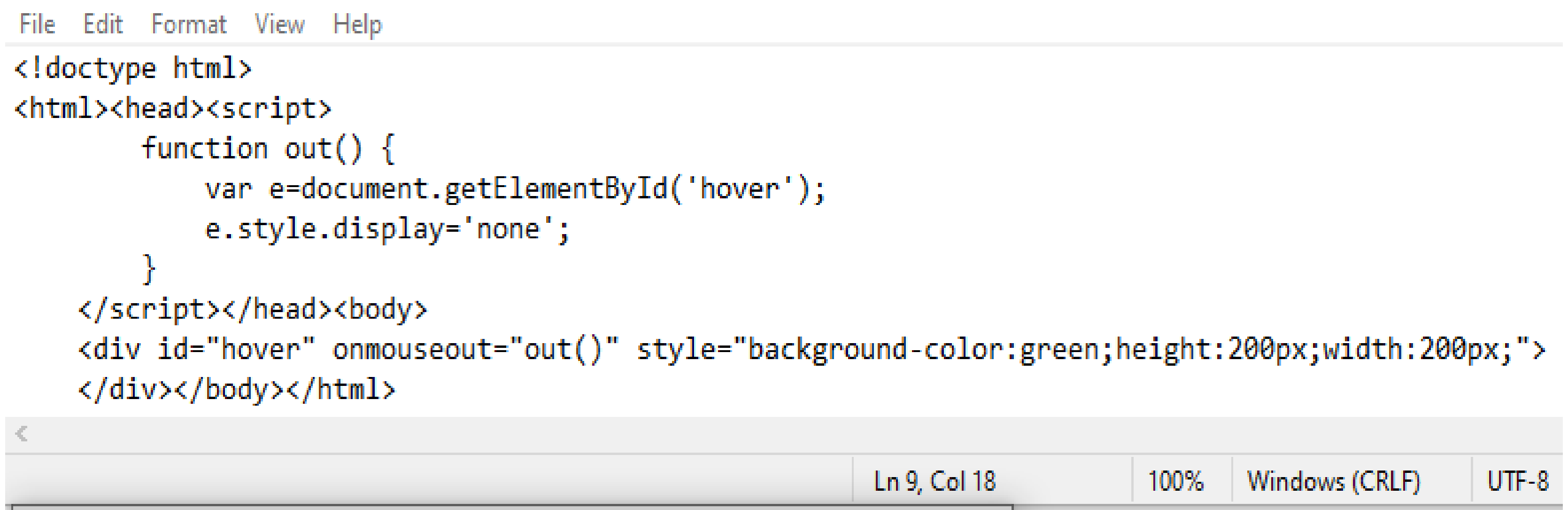


```
<!doctype html>
<html><head><script>
                var a=0;
                var b=0;
                var c=0;
                function changeBackground() {
                        var x=document.getElementById('bg');
                        x.style.backgroundColor='rgb('+a+', '+b+', '+c+')';
                        a+=100;
                        b+=a+50;
                        c+=b+70;
                        if(a>255) a=a-b;
                        if(b>255) b=a;
                        if(c>255) c=b;
                }
        </script></head><body>
        <h4>The input box will change color when UP arrow key is pressed</h4>
        <input id="bg" onkeyup="changeBackground()" placeholder="write something" style="color:#fff">
</body></html>
```

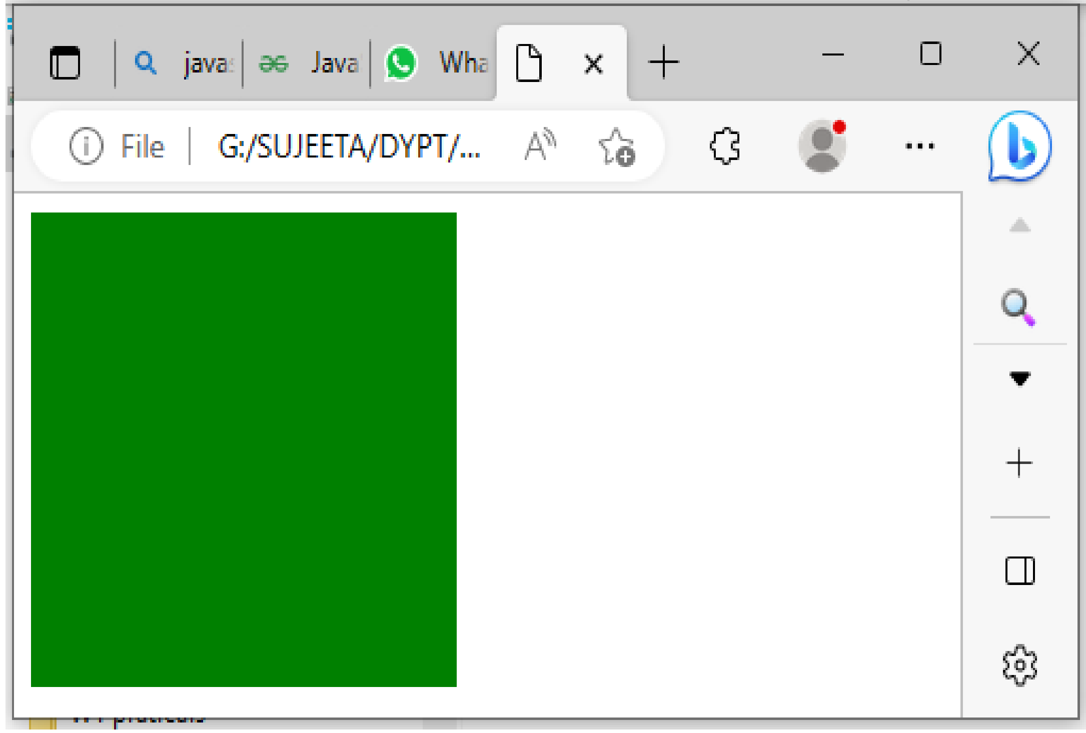


```
<!doctype html>
<html> <head>    <script>
        function hov() {
            var e=document.getElementById('hover');
            e.style.display='none';
        }
    </script></head><body>
    <div id="hover"
         onmouseover="hov()"
         style="background-color:green;
                height:200px;
                width:200px;">
    </div></body></html>
```

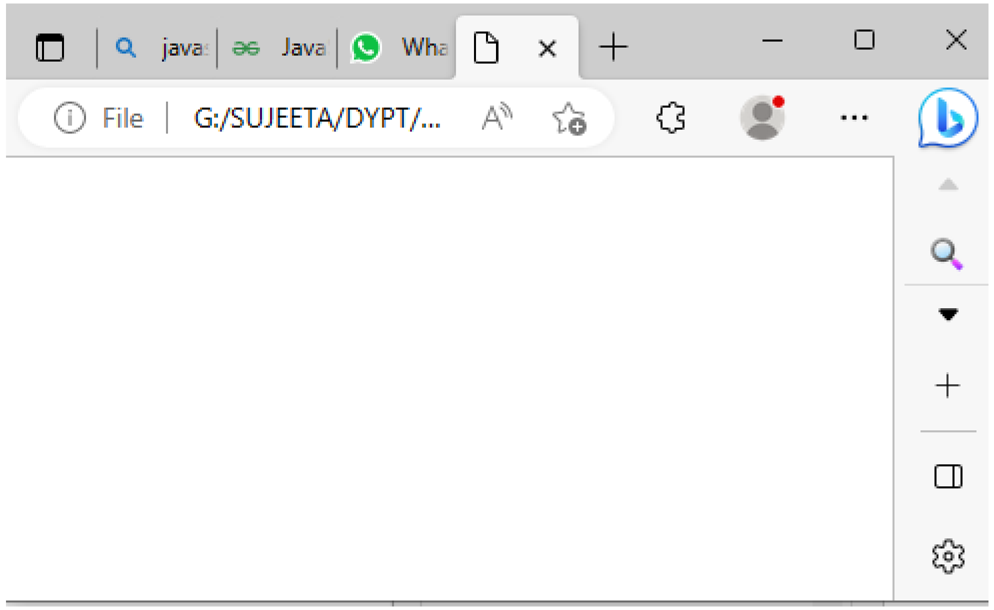


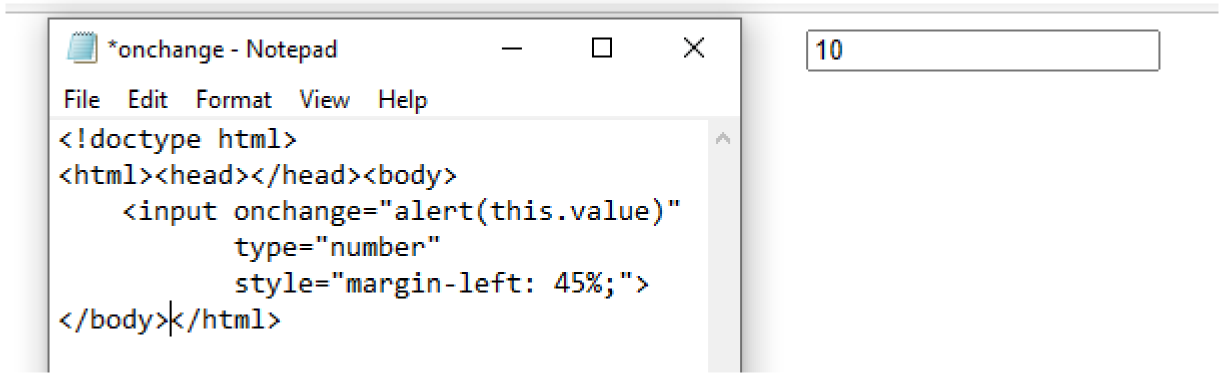


```
<!doctype html>
<html><head><script>
        function out() {
            var e=document.getElementById('hover');
            e.style.display='none';
        }
    </script></head><body>
    <div id="hover" onmouseout="out()" style="background-color:green;height:200px;width:200px;">
    </div></body></html>
```



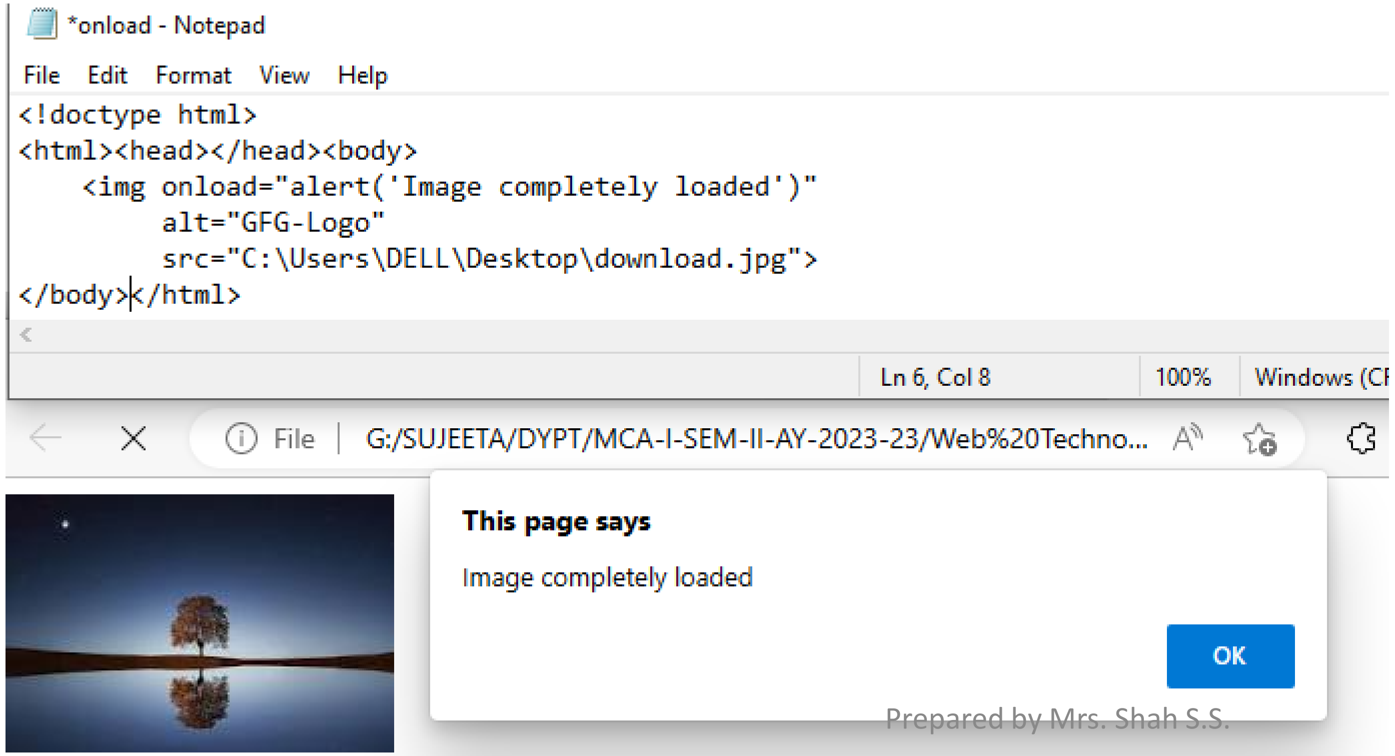
After Mouseout





```
<!doctype html>
<html><head></head><body>
    <input onchange="alert(this.value)"
           type="number"
           style="margin-left: 45%;">
</body></html>
```

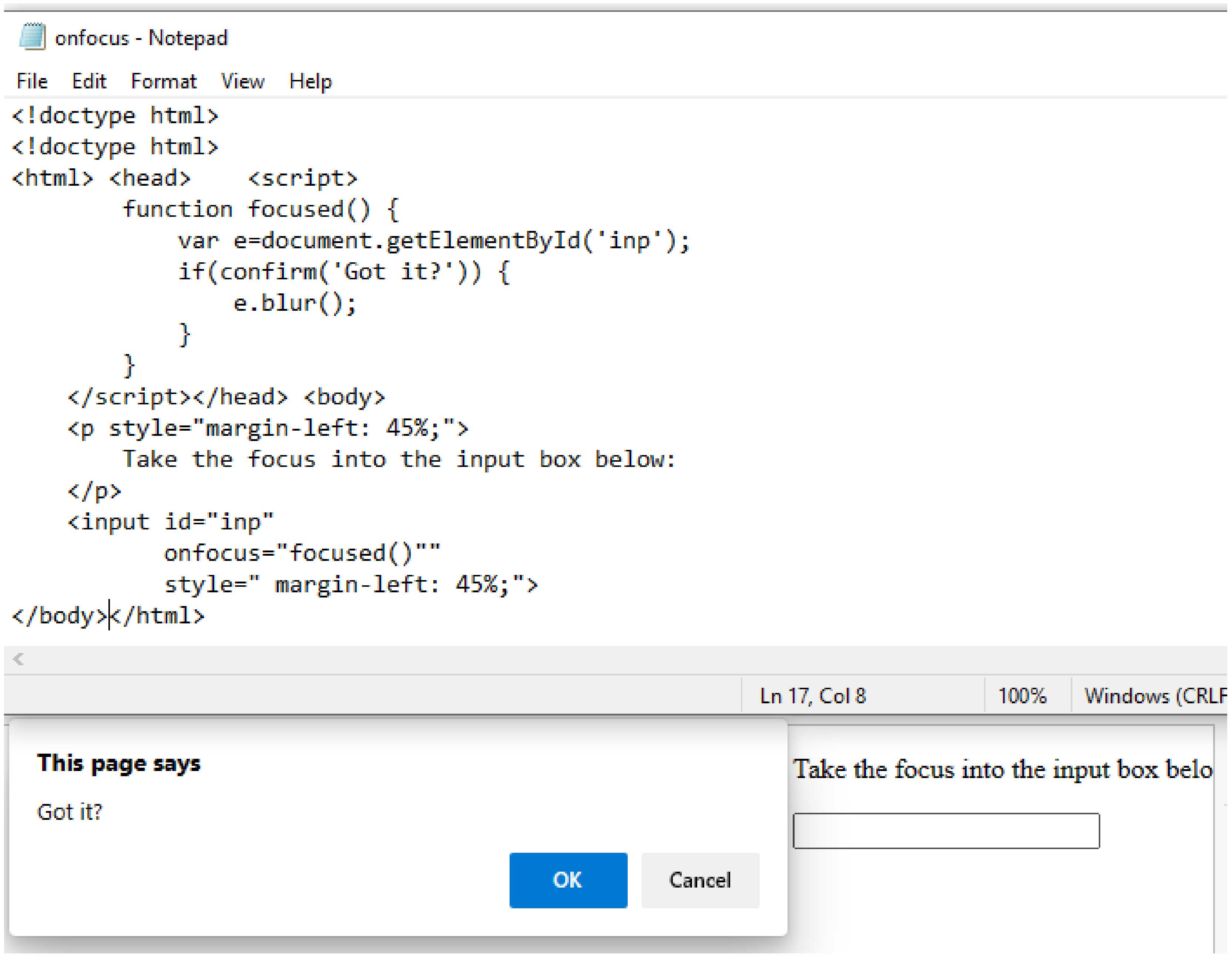
```
10
```


*onload - Notepad
File Edit Format View Help
Ln 6, Col 8
100%
Windows (CR
File
G:/SUJEETA/DYPT/MCA-I-SEM-II-AY-2023-23/Web%20Techno...
OK

```
<!doctype html>
<html><head></head><body>
    <img onload="alert('Image completely loaded')"
         alt="GFG-Logo"
         src="C:\Users\DELL\Desktop\download.jpg">
</body></html>
```

This page says

Image completely loaded

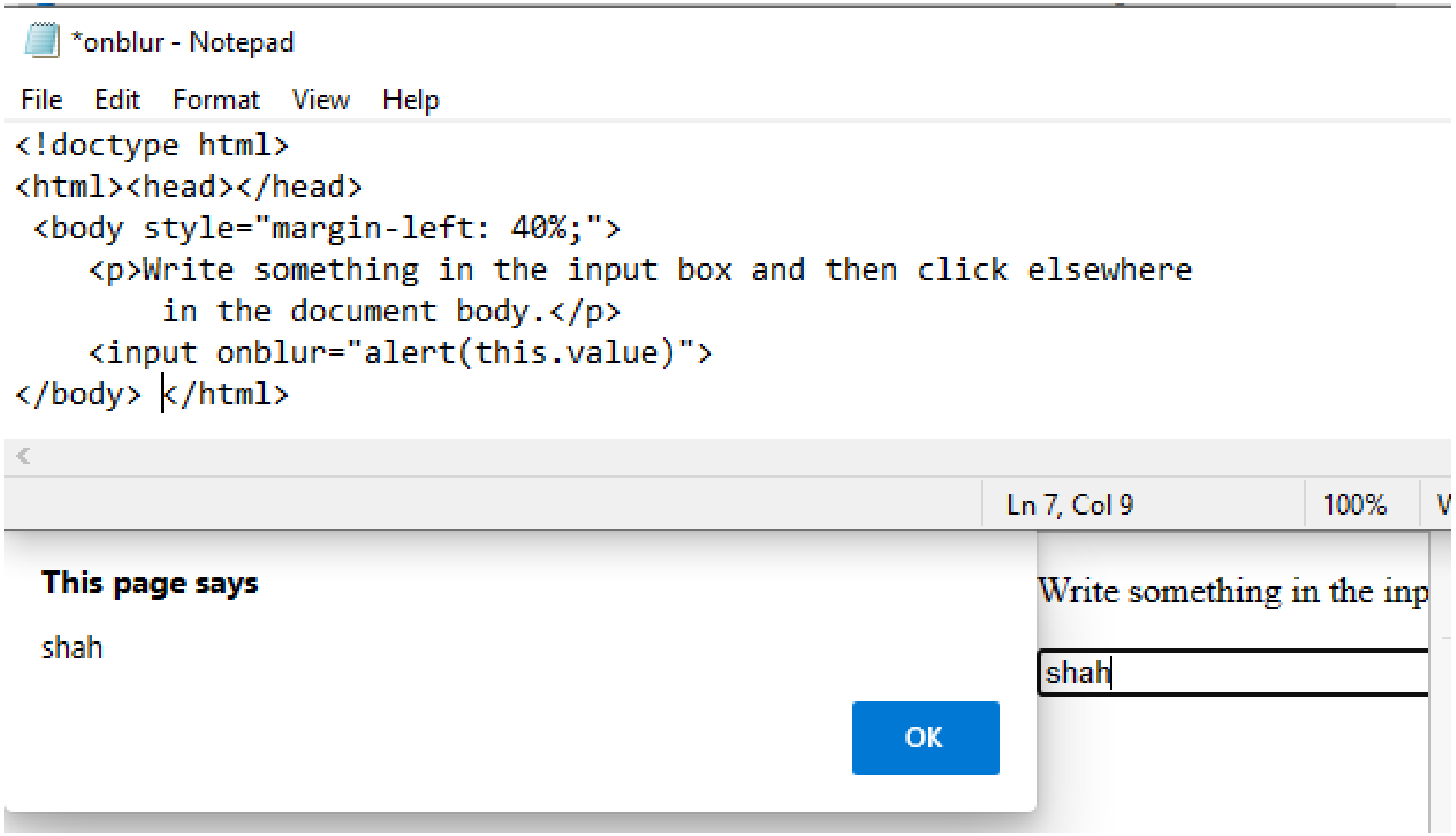


```
<!doctype html>
<!doctype html>
<html> <head>    <script>
        function focused() {
            var e=document.getElementById('inp');
            if(confirm('Got it?')) {
                e.blur();
            }
        }
    </script></head> <body>
    <p style="margin-left: 45%;">
        Take the focus into the input box below:
    </p>
    <input id="inp"
           onfocus="focused()""
           style=" margin-left: 45%;">
</body></html>
```

This page says

Got it?

Take the focus into the input box belo



```
<!doctype html>
<html><head></head>
 <body style="margin-left: 40%;">
    <p>Write something in the input box and then click elsewhere
        in the document body.</p>
    <input onblur="alert(this.value)">
</body> </html>
```

JavaScript Arrays

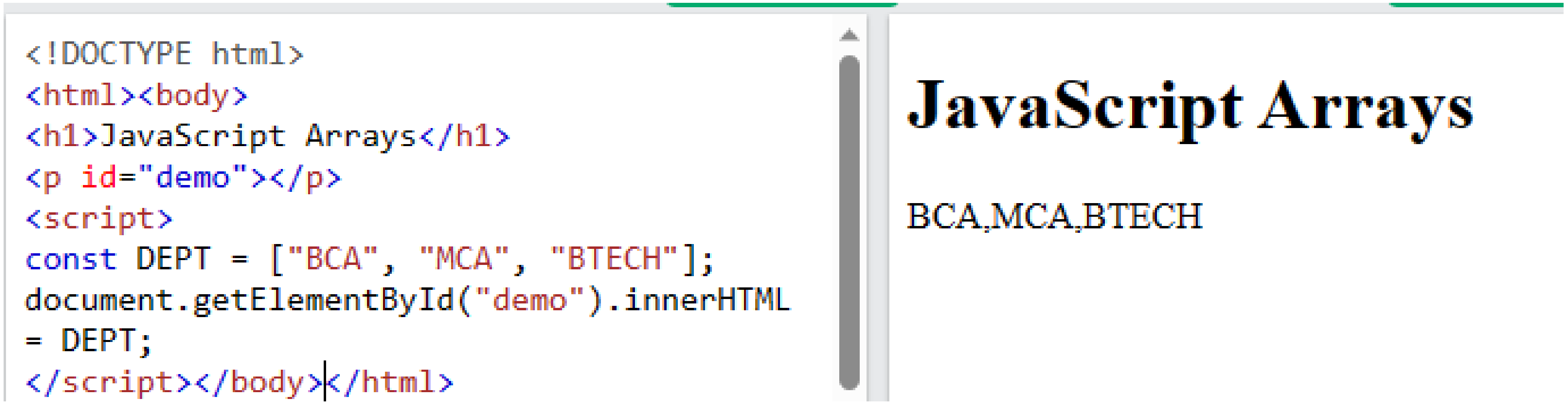
An array is a special variable, which can hold more than one value:

Creating an Array/Access the Full Array

Using an array literal is the easiest way to create a JavaScript Array.

Syntax:

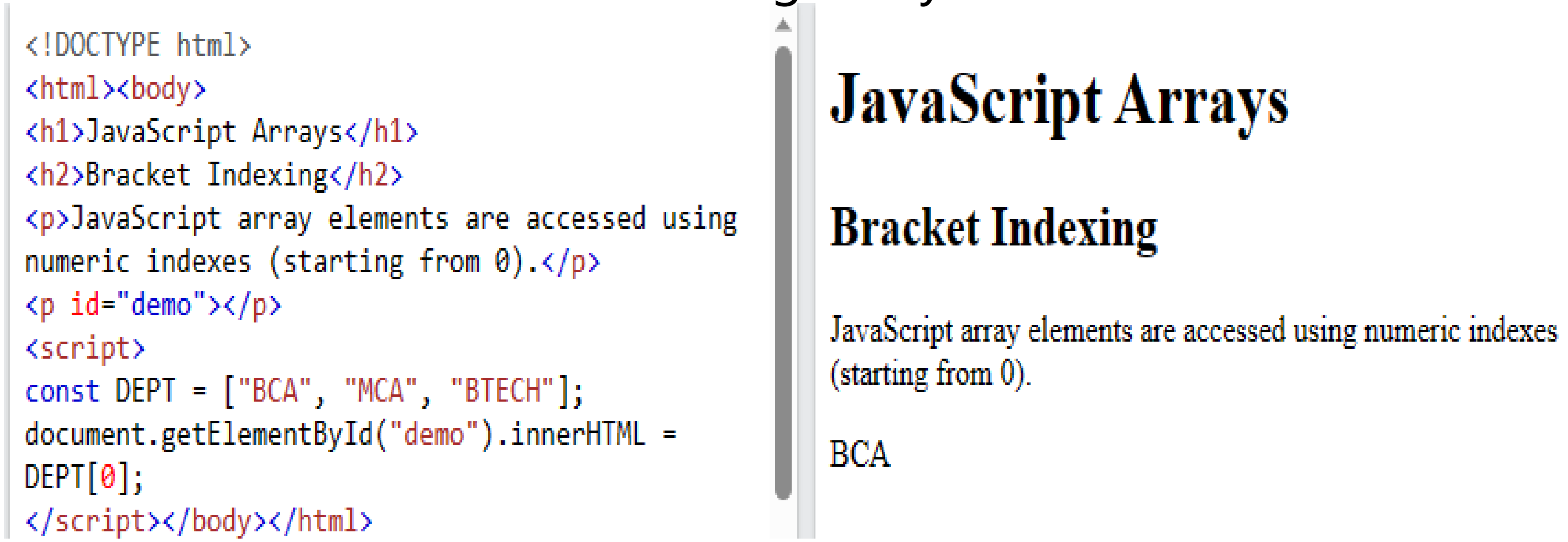
```
const array_name = [item1, item2, ...];
```



```
<!DOCTYPE html>
<html><body>
<h1>JavaScript Arrays</h1>
<p id="demo"></p>
<script>
const DEPT = ["BCA", "MCA", "BTECH"];
document.getElementById("demo").innerHTML
= DEPT;
</script></body></html>
```

```
JavaScript Arrays

BCA,MCA,BTECH
```

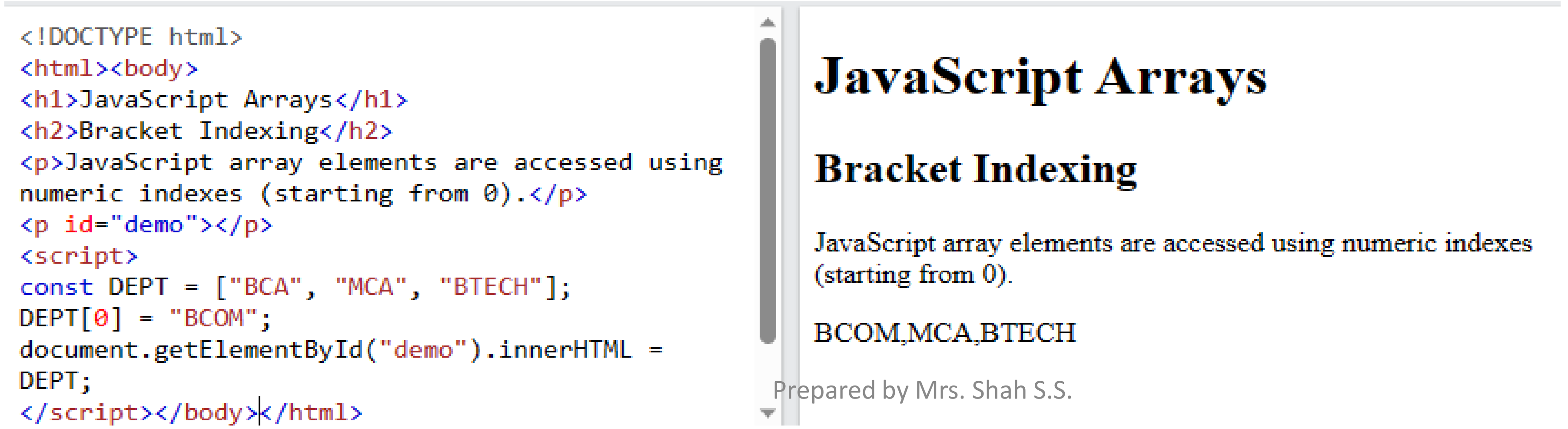
Accessing Array Elements



```
<!DOCTYPE html>
<html><body>
<h1>JavaScript Arrays</h1>
<h2>Bracket Indexing</h2>
<p>JavaScript array elements are accessed using
numeric indexes (starting from 0).</p>
<p id="demo"></p>
<script>
const DEPT = ["BCA", "MCA", "BTECH"];
document.getElementById("demo").innerHTML =
DEPT[0];
</script></body></html>
```

```
JavaScript Arrays

Bracket Indexing

JavaScript array elements are accessed using numeric indexes
(starting from 0).

BCA
```

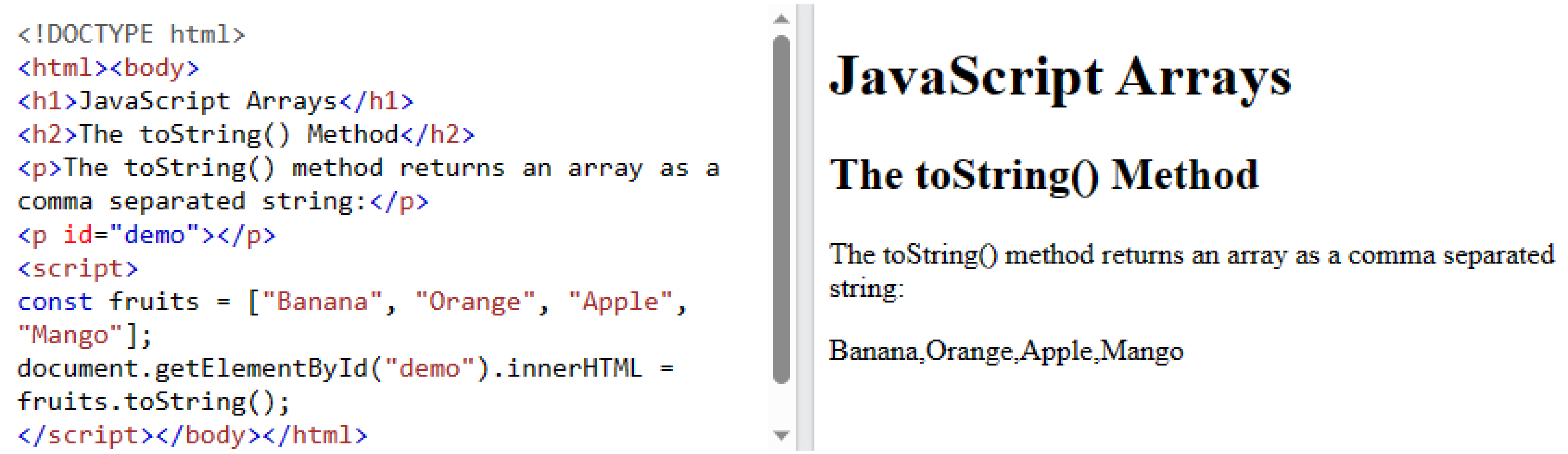
Changing an Array Element



```
<!DOCTYPE html>
<html><body>
<h1>JavaScript Arrays</h1>
<h2>Bracket Indexing</h2>
<p>JavaScript array elements are accessed using
numeric indexes (starting from 0).</p>
<p id="demo"></p>
<script>
const DEPT = ["BCA", "MCA", "BTECH"];
DEPT[0] = "BCOM";
document.getElementById("demo").innerHTML =
DEPT;
</script></body></html>
```

```
JavaScript Arrays

Bracket Indexing

JavaScript array elements are accessed using numeric indexes
(starting from 0).

BCOM,MCA,BTECH
```

Converting an Array to a String



```
<!DOCTYPE html>
<html><body>
<h1>JavaScript Arrays</h1>
<h2>The toString() Method</h2>
<p>The toString() method returns an array as a
comma separated string:</p>
<p id="demo"></p>
<script>
const fruits = ["Banana", "Orange", "Apple",
"Mango"];
document.getElementById("demo").innerHTML =
fruits.toString();
</script></body></html>
```

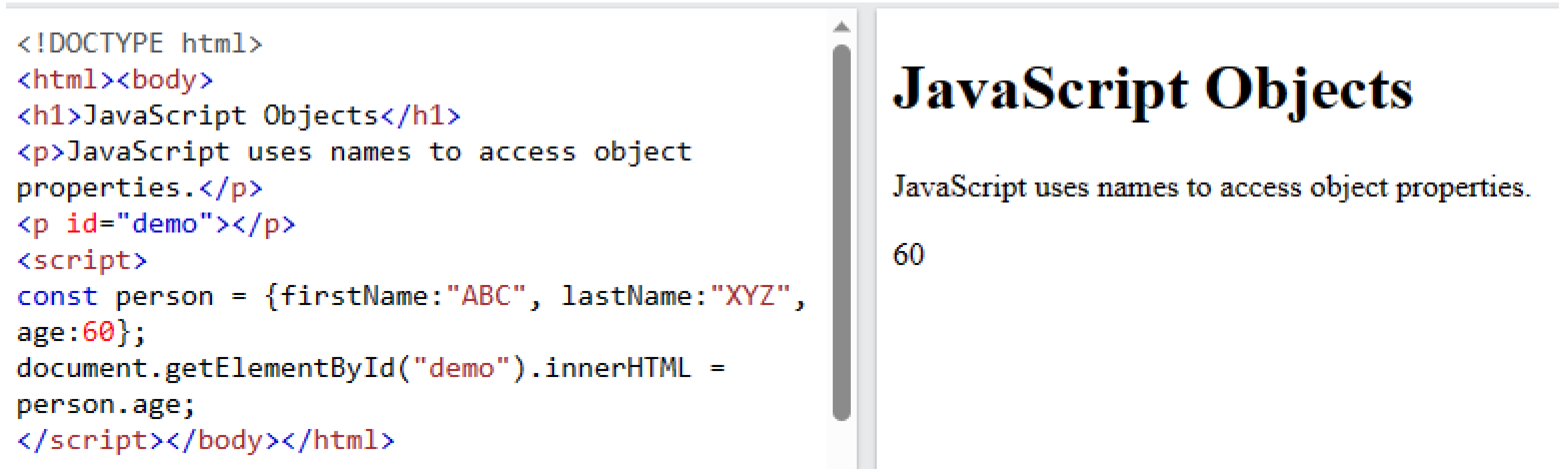
JavaScript Arrays

The toString() Method

The toString() method returns an array as a comma separated string:

```
Banana,Orange,Apple,Mango
```

Arrays are Objects

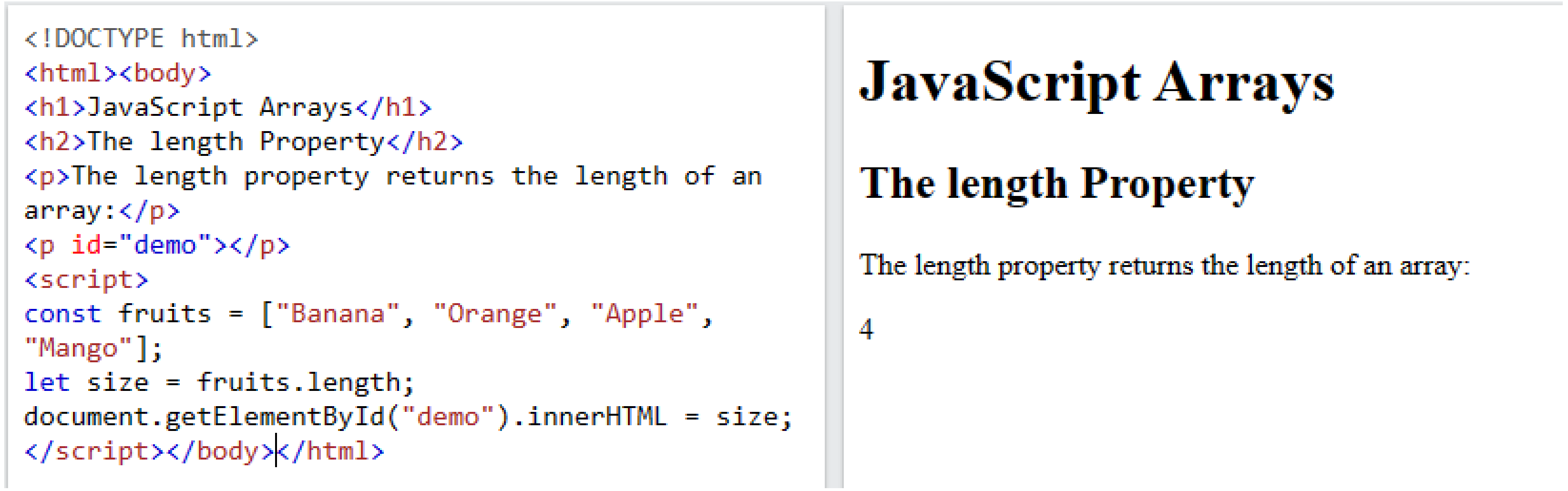


```
<!DOCTYPE html>
<html><body>
<h1>JavaScript Objects</h1>
<p>JavaScript uses names to access object
properties.</p>
<p id="demo"></p>
<script>
const person = {firstName:"ABC", lastName:"XYZ",
age:60};
document.getElementById("demo").innerHTML =
person.age;
</script></body></html>
```

JavaScript Objects

JavaScript uses names to access object properties.

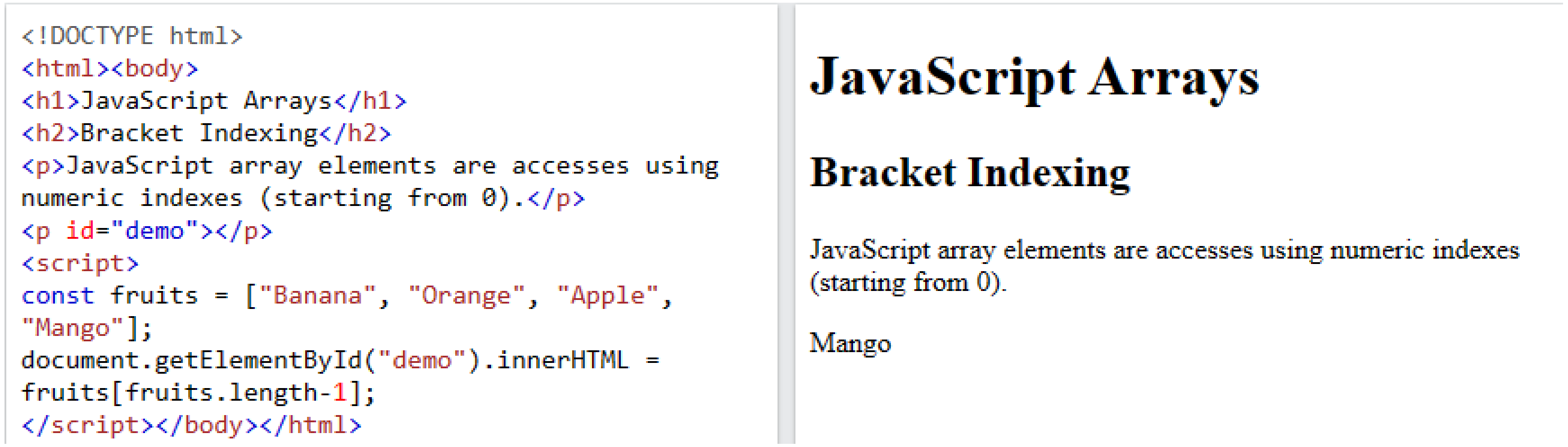
```
60
```



```
<!DOCTYPE html>
<html><body>
<h1>JavaScript Arrays</h1>
<h2>The length Property</h2>
<p>The length property returns the length of an
array:</p>
<p id="demo"></p>
<script>
const fruits = ["Banana", "Orange", "Apple",
"Mango"];
let size = fruits.length;
document.getElementById("demo").innerHTML = size;
</script></body></html>
```

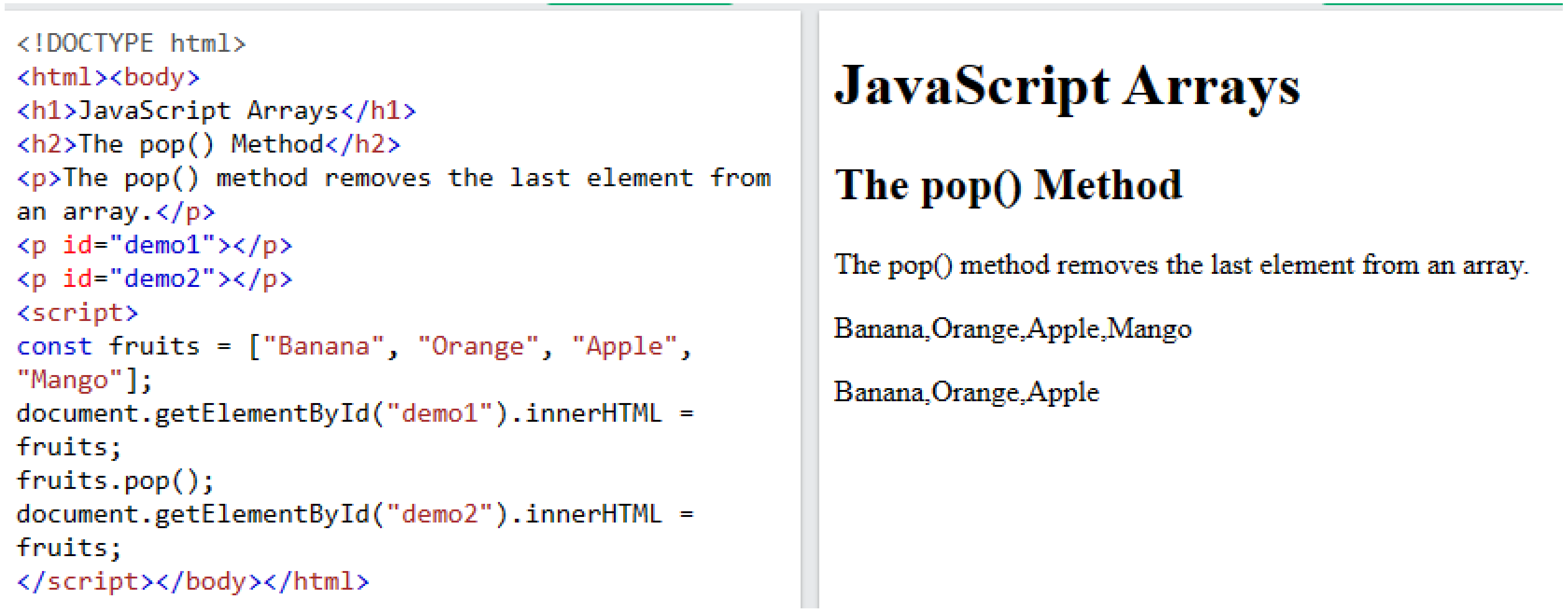
```
JavaScript Arrays

The length Property

The length property returns the length of an array:

4
```

Accessing the Last Array Element



```
<!DOCTYPE html>
<html><body>
<h1>JavaScript Arrays</h1>
<h2>Bracket Indexing</h2>
<p>JavaScript array elements are accesses using
numeric indexes (starting from 0).</p>
<p id="demo"></p>
<script>
const fruits = ["Banana", "Orange", "Apple",
"Mango"];
document.getElementById("demo").innerHTML =
fruits[fruits.length-1];
</script></body></html>
```

```
JavaScript Arrays

Bracket Indexing

JavaScript array elements are accesses using numeric indexes
(starting from 0).

Mango
```



```
<!DOCTYPE html>
<html><body>
<h1>JavaScript Arrays</h1>
<h2>The pop() Method</h2>
<p>The pop() method removes the last element from
an array.</p>
<p id="demo1"></p>
<p id="demo2"></p>
<script>
const fruits = ["Banana", "Orange", "Apple",
"Mango"];
document.getElementById("demo1").innerHTML =
fruits;
fruits.pop();
document.getElementById("demo2").innerHTML =
fruits;
</script></body></html>
```

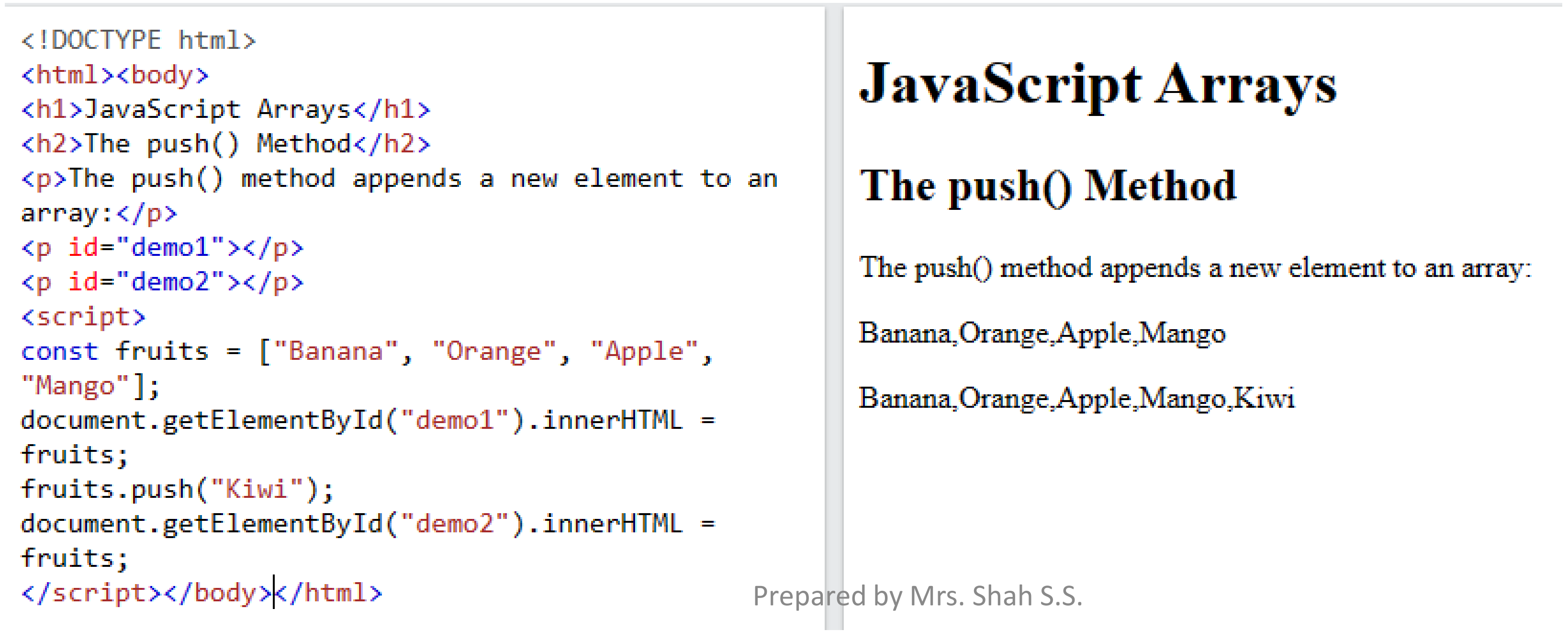
JavaScript Arrays

The pop() Method

The pop() method removes the last element from an array.

Banana,Orange,Apple,Mango

Banana,Orange,Apple



```
<!DOCTYPE html>
<html><body>
<h1>JavaScript Arrays</h1>
<h2>The push() Method</h2>
<p>The push() method appends a new element to an
array:</p>
<p id="demo1"></p>
<p id="demo2"></p>
<script>
const fruits = ["Banana", "Orange", "Apple",
"Mango"];
document.getElementById("demo1").innerHTML =
fruits;
fruits.push("Kiwi");
document.getElementById("demo2").innerHTML =
fruits;
</script></body></html>
```

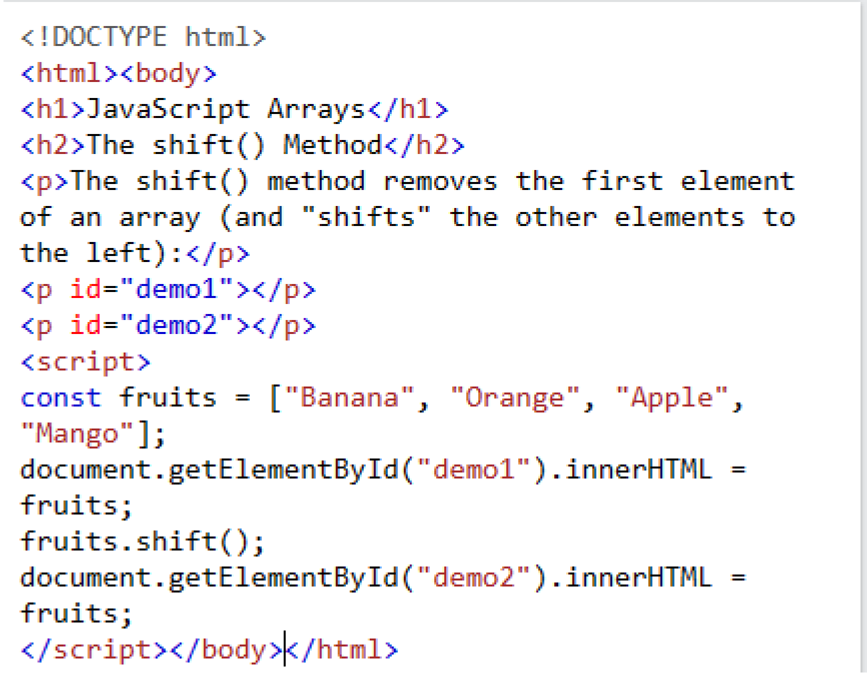
JavaScript Arrays

The push() Method

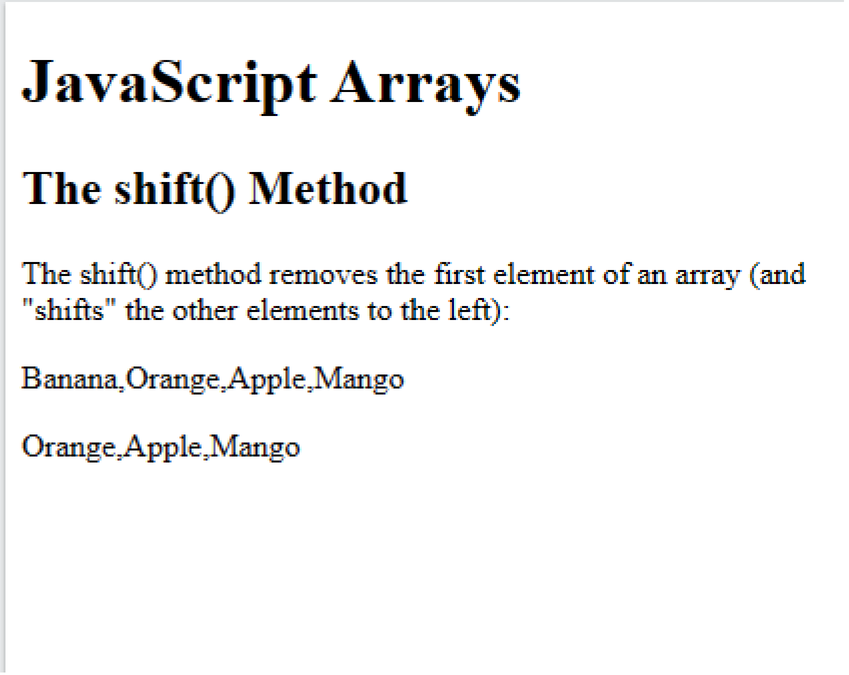
The push() method appends a new element to an array:

Banana,Orange,Apple,Mango

Banana,Orange,Apple,Mango,Kiwi



```
<!DOCTYPE html>
<html><body>
<h1>JavaScript Arrays</h1>
<h2>The shift() Method</h2>
<p>The shift() method removes the first element
of an array (and "shifts" the other elements to
the left):</p>
<p id="demo1"></p>
<p id="demo2"></p>
<script>
const fruits = ["Banana", "Orange", "Apple",
"Mango"];
document.getElementById("demo1").innerHTML =
fruits;
fruits.shift();
document.getElementById("demo2").innerHTML =
fruits;
</script></body></html>
```

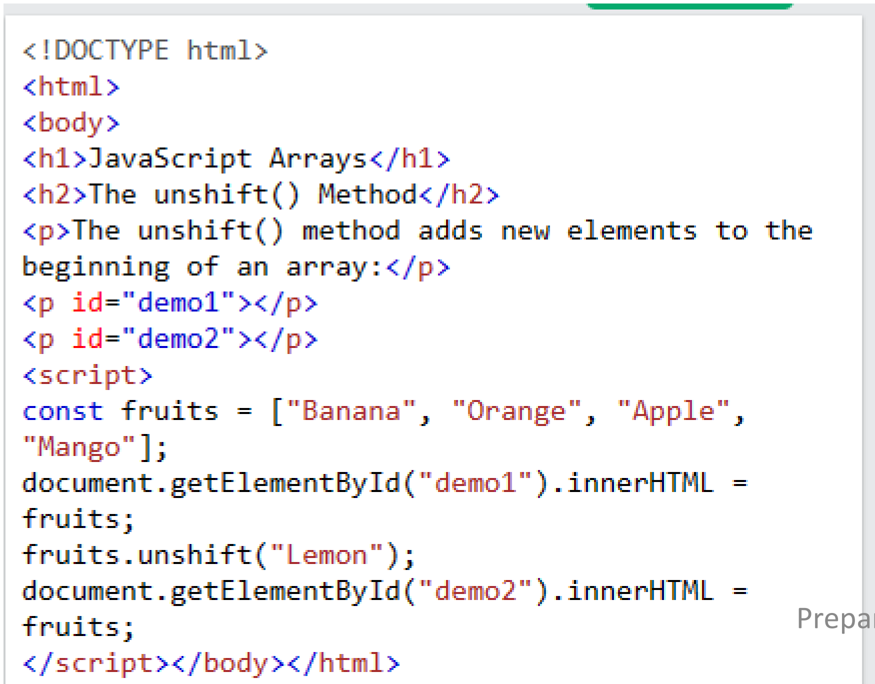


JavaScript Arrays

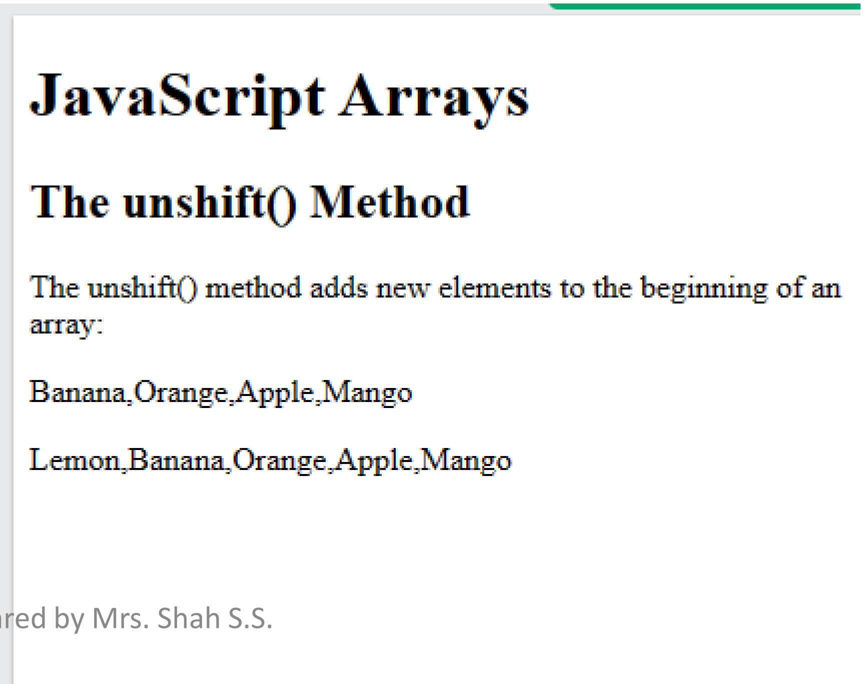
The shift() Method

The shift() method removes the first element of an array (and "shifts" the other elements to the left):

```
Banana,Orange,Apple,Mango

Orange,Apple,Mango
```



```
<!DOCTYPE html>
<html>
<body>
<h1>JavaScript Arrays</h1>
<h2>The unshift() Method</h2>
<p>The unshift() method adds new elements to the
beginning of an array:</p>
<p id="demo1"></p>
<p id="demo2"></p>
<script>
const fruits = ["Banana", "Orange", "Apple",
"Mango"];
document.getElementById("demo1").innerHTML =
fruits;
fruits.unshift("Lemon");
document.getElementById("demo2").innerHTML =
fruits;
</script></body></html>
```

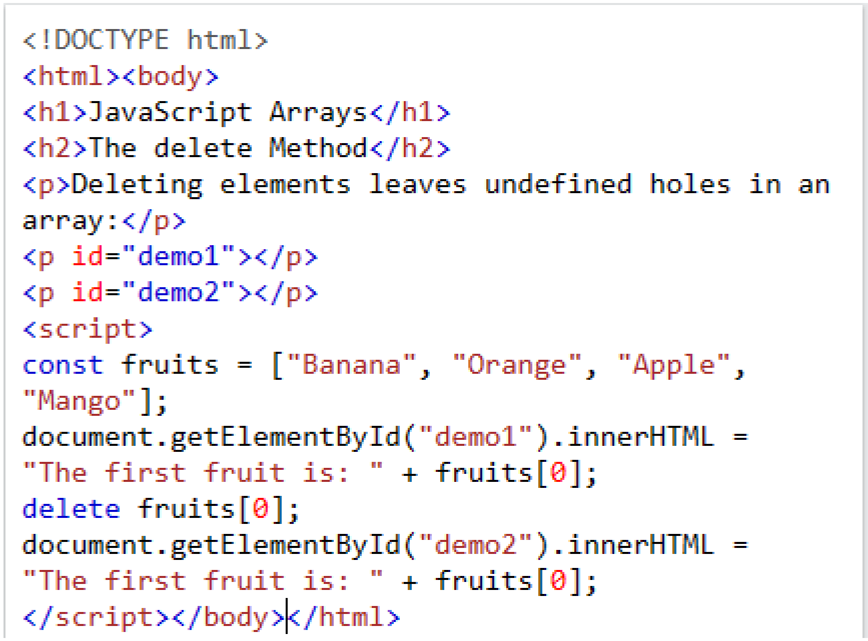


JavaScript Arrays

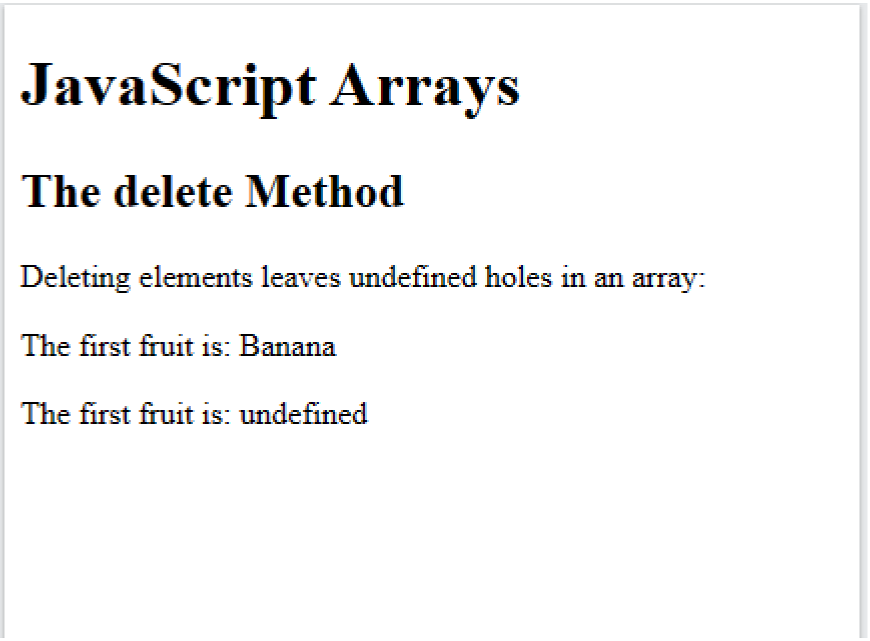
The unshift() Method

The unshift() method adds new elements to the beginning of an array:

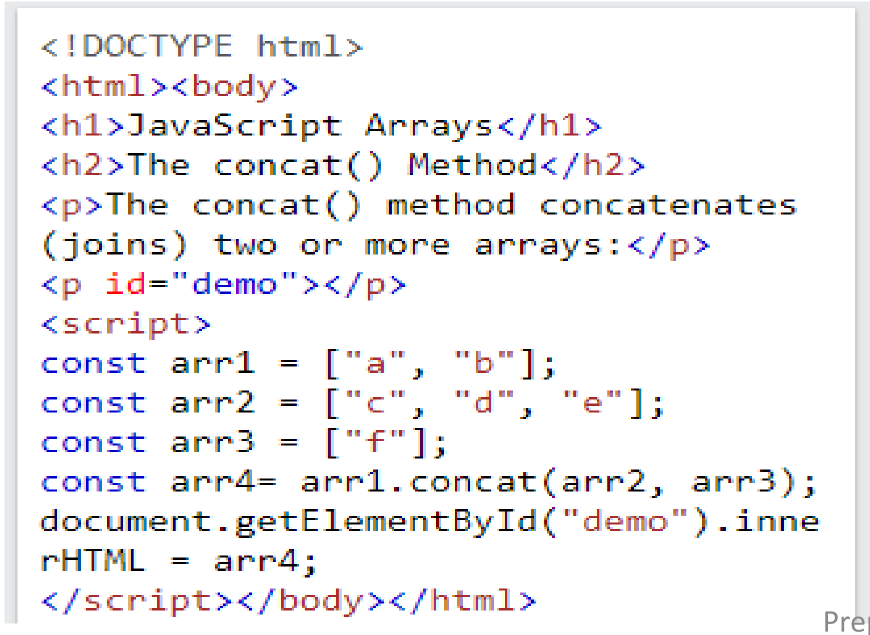
```
Banana,Orange,Apple,Mango

Lemon,Banana,Orange,Apple,Mango
```



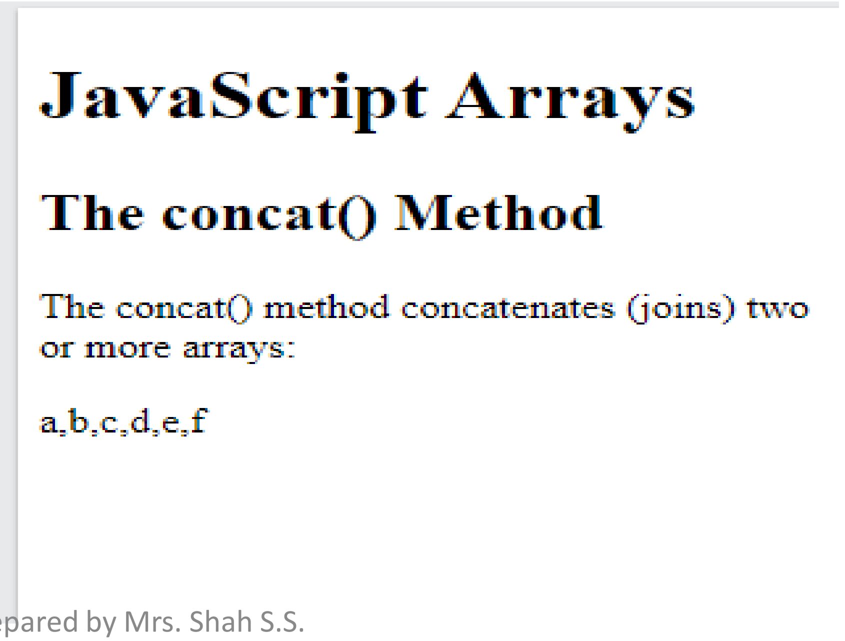
```
<!DOCTYPE html>
<html><body>
<h1>JavaScript Arrays</h1>
<h2>The delete Method</h2>
<p>Deleting elements leaves undefined holes in an
array:</p>
<p id="demo1"></p>
<p id="demo2"></p>
<script>
const fruits = ["Banana", "Orange", "Apple",
"Mango"];
document.getElementById("demo1").innerHTML =
"The first fruit is: " + fruits[0];
delete fruits[0];
document.getElementById("demo2").innerHTML =
"The first fruit is: " + fruits[0];
</script></body></html>
```



```
JavaScript Arrays

The delete Method

Deleting elements leaves undefined holes in an array:

The first fruit is: Banana

The first fruit is: undefined
```



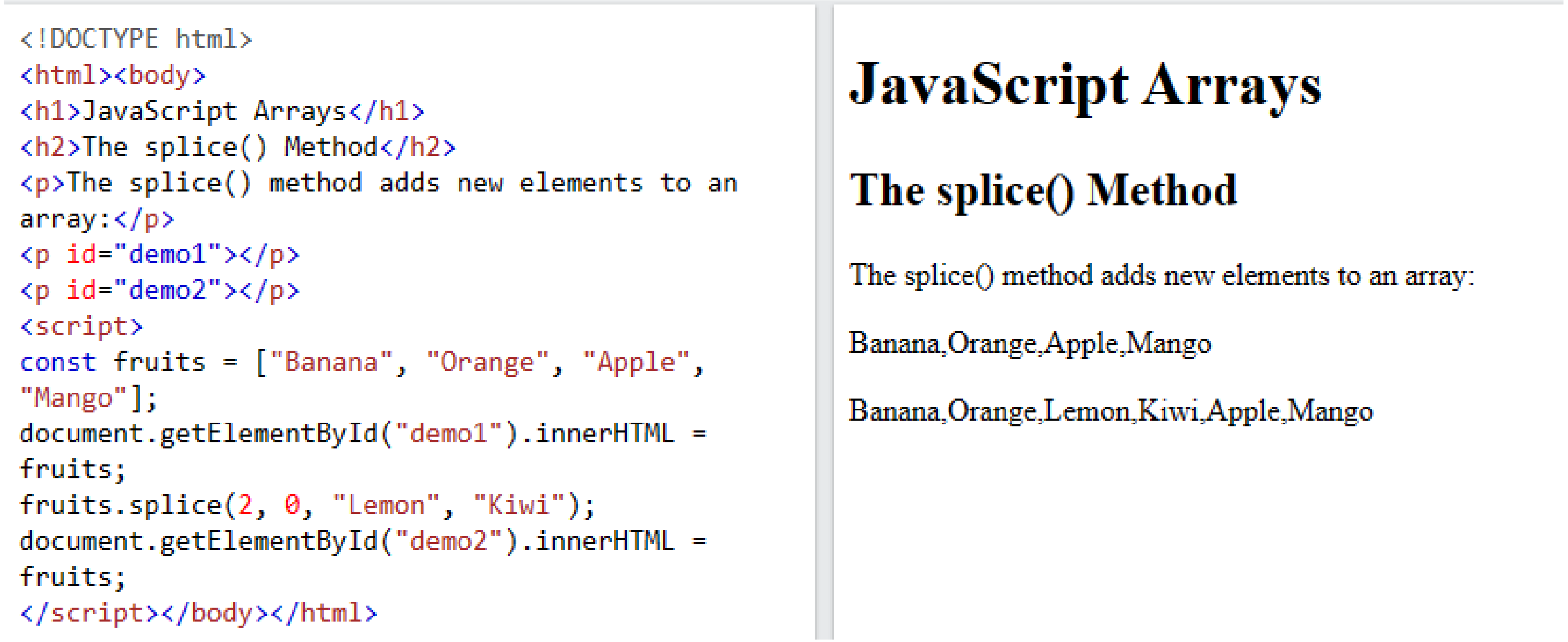
```
<!DOCTYPE html>
<html><body>
<h1>JavaScript Arrays</h1>
<h2>The concat() Method</h2>
<p>The concat() method concatenates
(joins) two or more arrays:</p>
<p id="demo"></p>
<script>
const arr1 = ["a", "b"];
const arr2 = ["c", "d", "e"];
const arr3 = ["f"];
const arr4= arr1.concat(arr2, arr3);
document.getElementById("demo").inne
rHTML = arr4;
</script></body></html>
```



```
JavaScript Arrays

The concat() Method

The concat() method concatenates (joins) two
or more arrays:

a,b,c,d,e,f
```

Splicing and Slicing Arrays

The `splice()` method adds new items to an array.
The `slice()` method slices out a piece of an array.



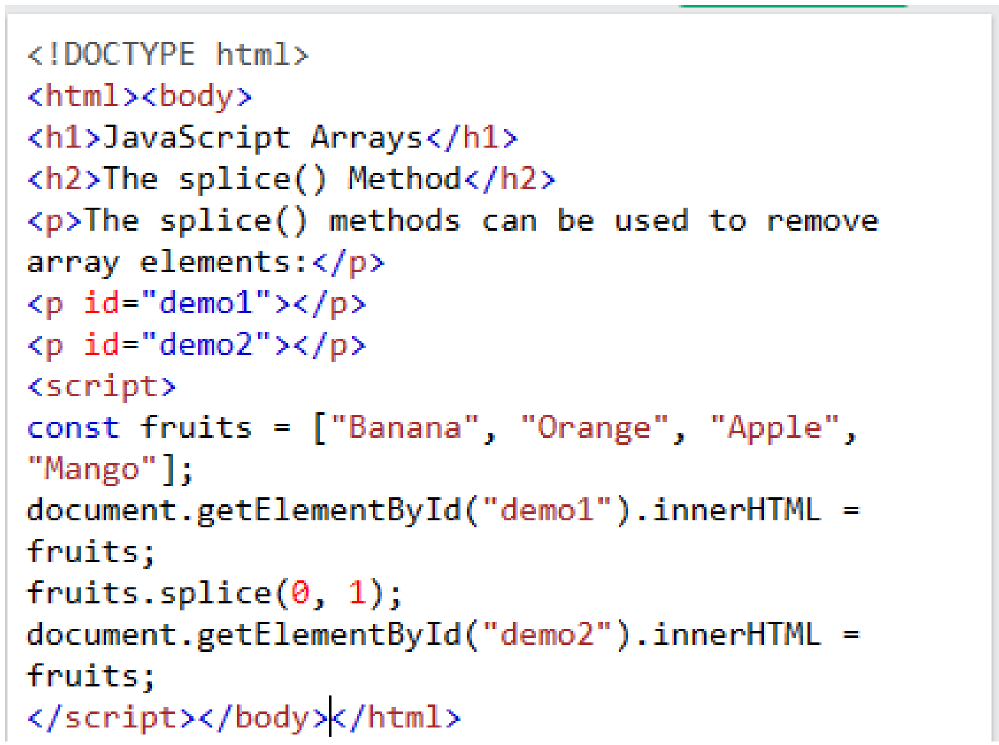
```
<!DOCTYPE html>
<html><body>
<h1>JavaScript Arrays</h1>
<h2>The splice() Method</h2>
<p>The splice() method adds new elements to an
array:</p>
<p id="demo1"></p>
<p id="demo2"></p>
<script>
const fruits = ["Banana", "Orange", "Apple",
"Mango"];
document.getElementById("demo1").innerHTML =
fruits;
fruits.splice(2, 0, "Lemon", "Kiwi");
document.getElementById("demo2").innerHTML =
fruits;
</script></body></html>
```

```
JavaScript Arrays

The splice() Method

The splice() method adds new elements to an array:

Banana,Orange,Apple,Mango

Banana,Orange,Lemon,Kiwi,Apple,Mango
```

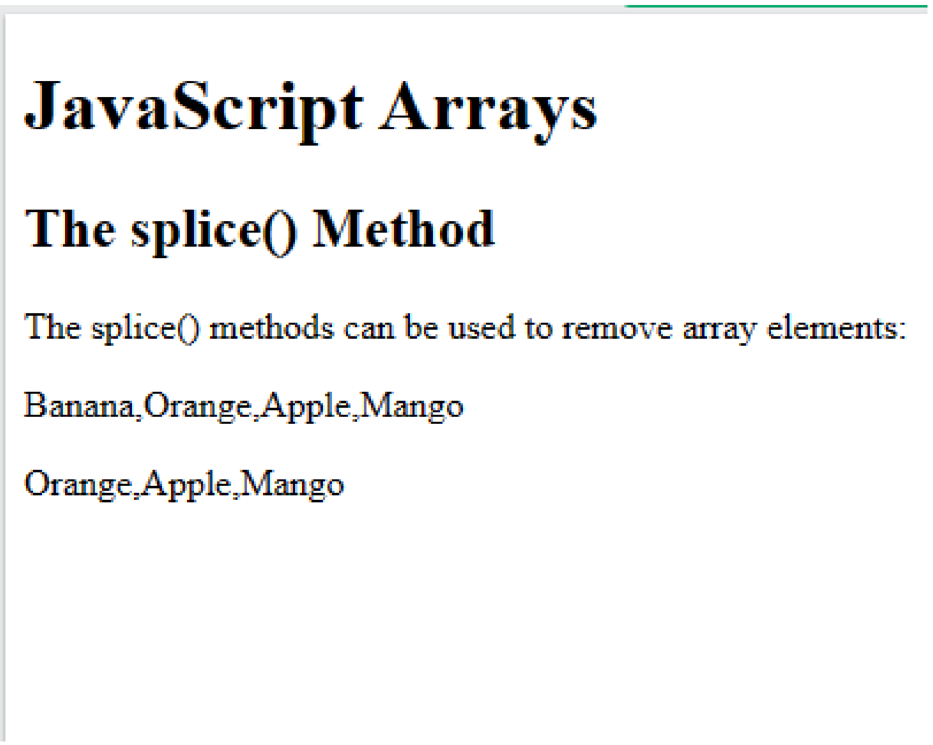
The first parameter (2) defines the position **where** new elements should be **added** (spliced in).

The second parameter (0) defines **how many** elements should be **removed**.

Using splice() to Remove Elements



```
<!DOCTYPE html>
<html><body>
<h1>JavaScript Arrays</h1>
<h2>The splice() Method</h2>
<p>The splice() methods can be used to remove
array elements:</p>
<p id="demo1"></p>
<p id="demo2"></p>
<script>
const fruits = ["Banana", "Orange", "Apple",
"Mango"];
document.getElementById("demo1").innerHTML =
fruits;
fruits.splice(0, 1);
document.getElementById("demo2").innerHTML =
fruits;
</script></body></html>
```



JavaScript Arrays

The splice() Method

The splice() methods can be used to remove array elements:

Banana,Orange,Apple,Mango

Orange,Apple,Mango

The first parameter (0) defines the position where new elements should be **added** (spliced in).

The second parameter (1) defines **how many** elements should be **removed**.

JavaScript Objects

Real Life Objects, Properties, and Methods

In real life, a car is an **object**.
A car has **properties** like weight and color, and **methods** like start and stop:

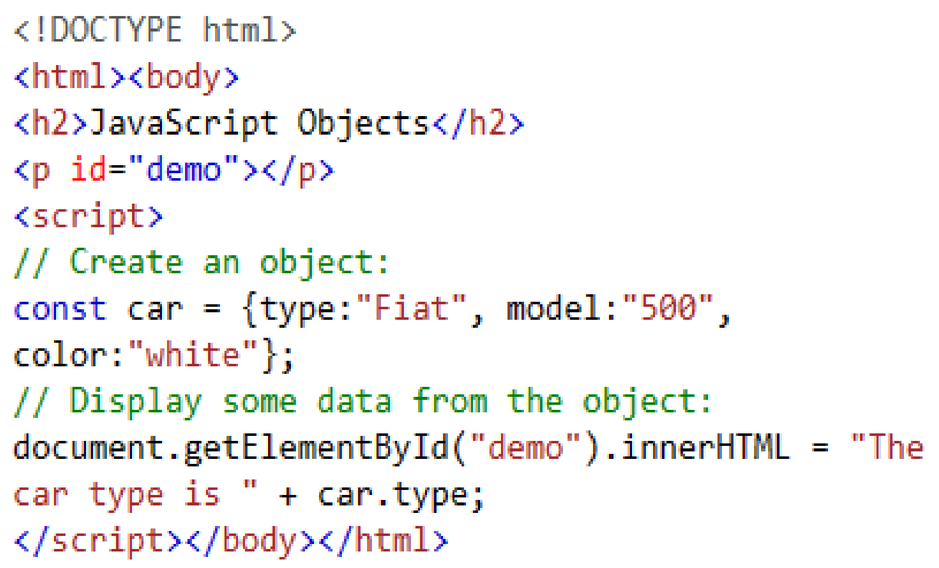
All cars have the same **properties**, but the property **values** differ from car to car.
All cars have the same **methods**, but the methods are performed **at different times**.

In JavaScript, the `this` keyword refers to an **object**.
Which object depends on how `this` is being invoked (used or called).

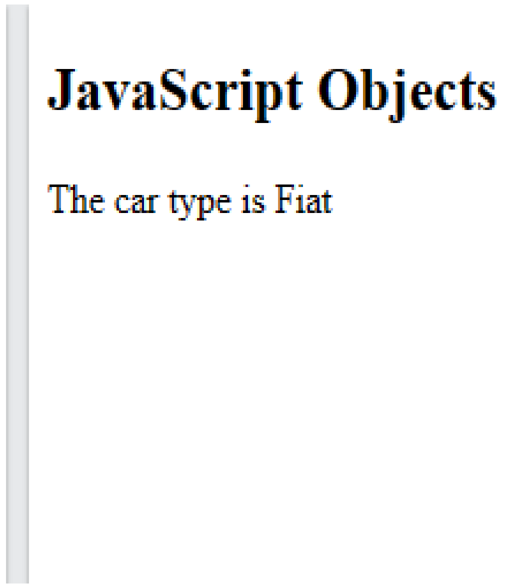
Object	Properties	Methods

	car.name = Fiat
	car.model = 500
	car.weight = 850kg
	car.color = white
		car.start()
		car.drive()
		car.brake()
		car.stop()

Objects are variables too. But objects can contain many values.



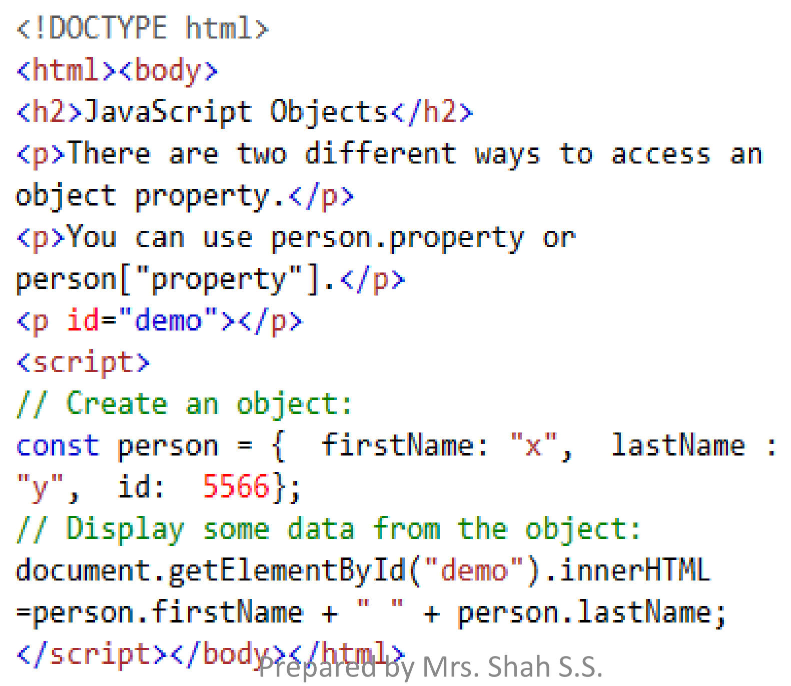
```
<!DOCTYPE html>
<html><body>
<h2>JavaScript Objects</h2>
<p id="demo"></p>
<script>
// Create an object:
const car = {type:"Fiat", model:"500",
color:"white"};
// Display some data from the object:
document.getElementById("demo").innerHTML = "The
car type is " + car.type;
</script></body></html>
```



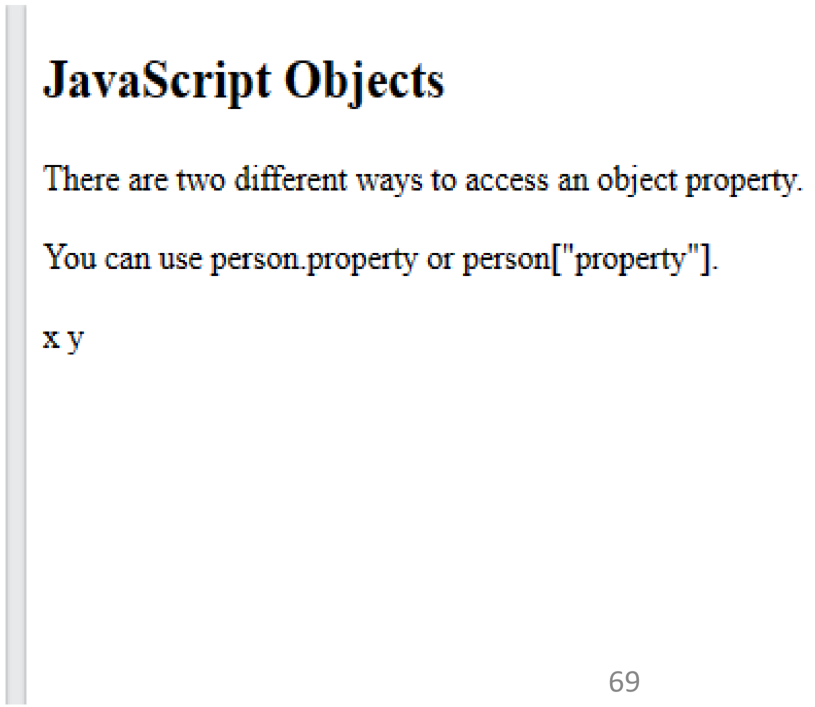
JavaScript Objects

The car type is Fiat

Accessing Object Properties
You can access object properties in two ways:
objectName.propertyName
or
objectName["propertyName"]



```
<!DOCTYPE html>
<html><body>
<h2>JavaScript Objects</h2>
<p>There are two different ways to access an
object property.</p>
<p>You can use person.property or
person["property"].</p>
<p id="demo"></p>
<script>
// Create an object:
const person = {  firstName: "x",  lastName :
"y",  id:  5566};
// Display some data from the object:
document.getElementById("demo").innerHTML
=person.firstName + " " + person.lastName;
</script></body></html>
```



JavaScript Objects

There are two different ways to access an object property.

You can use person.property or person["property"].

x y

JavaScript Functions

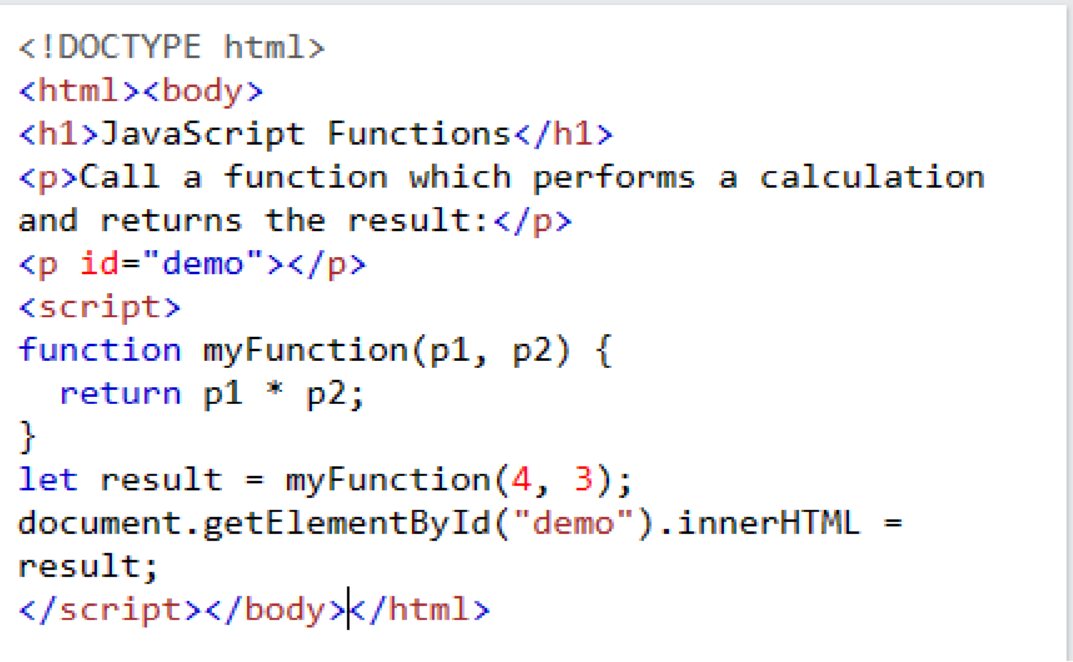
A JavaScript function is a block of code designed to perform a particular task.
A JavaScript function is executed when "something" invokes it (calls it).

JavaScript Function Syntax

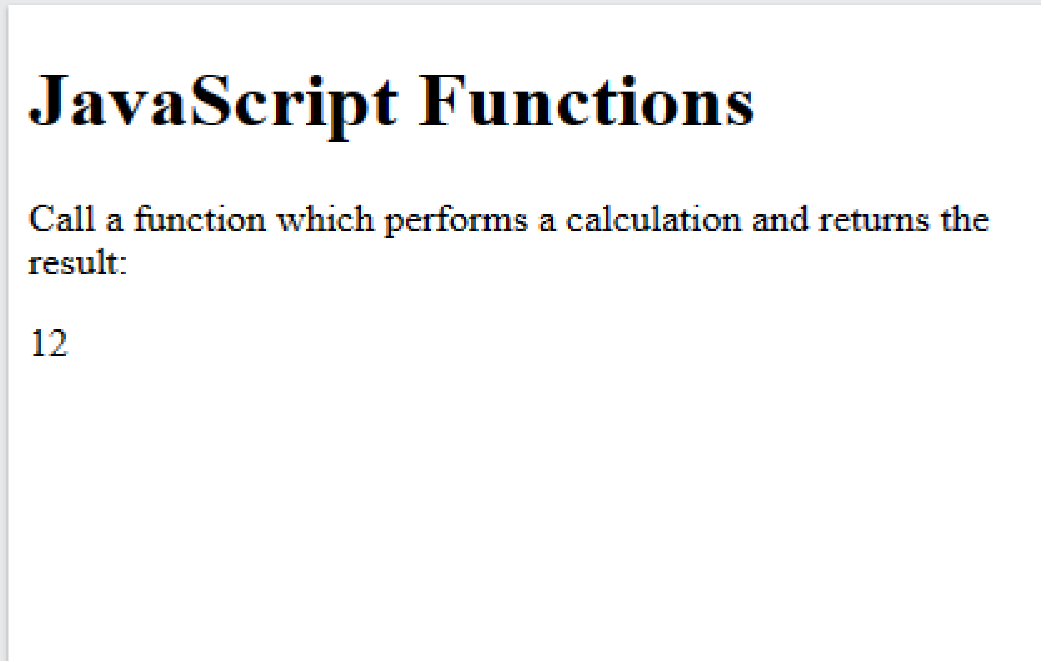
A JavaScript function is defined with the `function` keyword, followed by a **name**, followed by parentheses **()**.
Function names can contain letters, digits, underscores, and dollar signs (same rules as variables).
The parentheses may include parameter names separated by commas:
(parameter1, parameter2, ...)
The code to be executed, by the function, is placed inside curly brackets: **{}**

```
function name(parameter1, parameter2, parameter3) {
  // code to be executed
}
```

Function **parameters** are listed inside the parentheses () in the function definition.
Function **arguments** are the **values** received by the function when it is invoked.
Inside the function, the arguments (the parameters) behave as local variables.



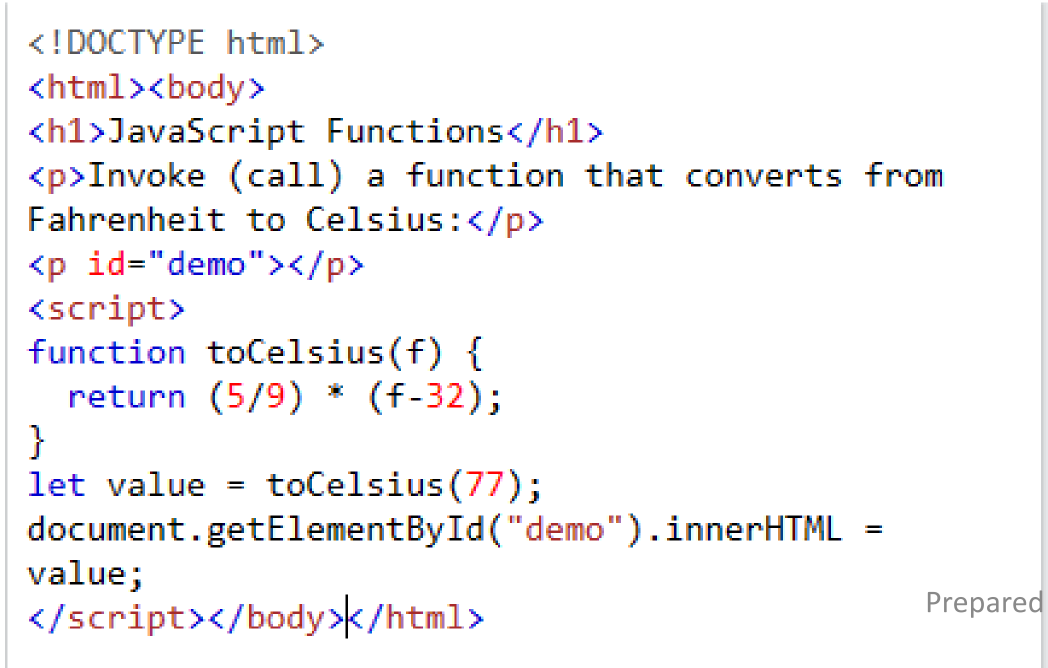
```
<!DOCTYPE html>
<html><body>
<h1>JavaScript Functions</h1>
<p>Call a function which performs a calculation
and returns the result:</p>
<p id="demo"></p>
<script>
function myFunction(p1, p2) {
  return p1 * p2;
}
let result = myFunction(4, 3);
document.getElementById("demo").innerHTML =
result;
</script></body></html>
```



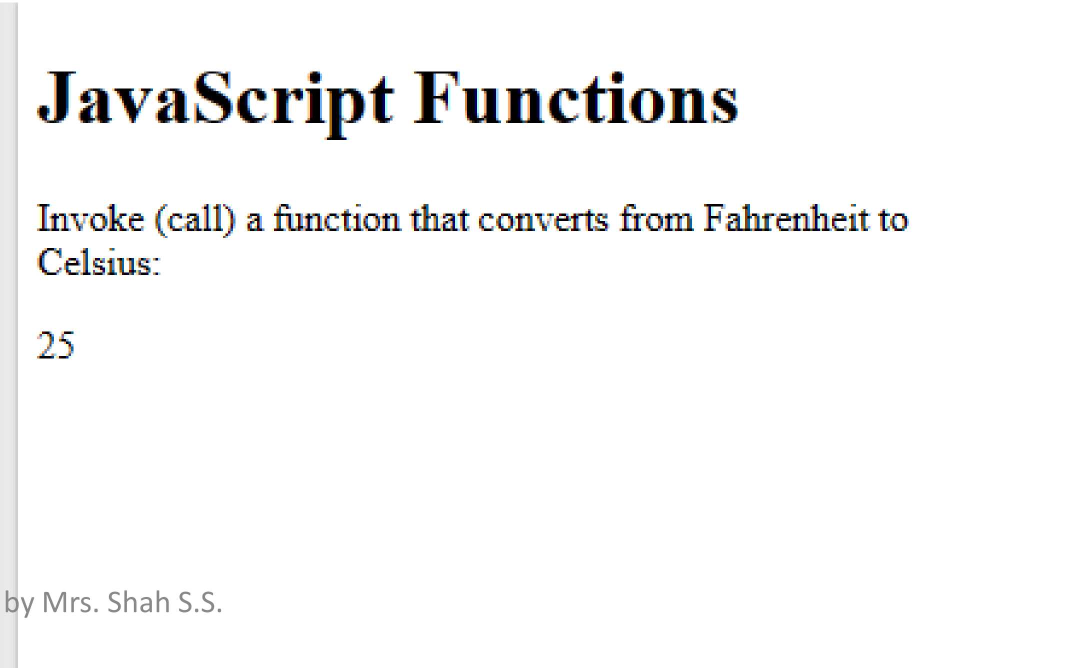
JavaScript Functions

Call a function which performs a calculation and returns the result:

12



```
<!DOCTYPE html>
<html><body>
<h1>JavaScript Functions</h1>
<p>Invoke (call) a function that converts from
Fahrenheit to Celsius:</p>
<p id="demo"></p>
<script>
function toCelsius(f) {
  return (5/9) * (f-32);
}
let value = toCelsius(77);
document.getElementById("demo").innerHTML =
value;
</script></body></html>
```



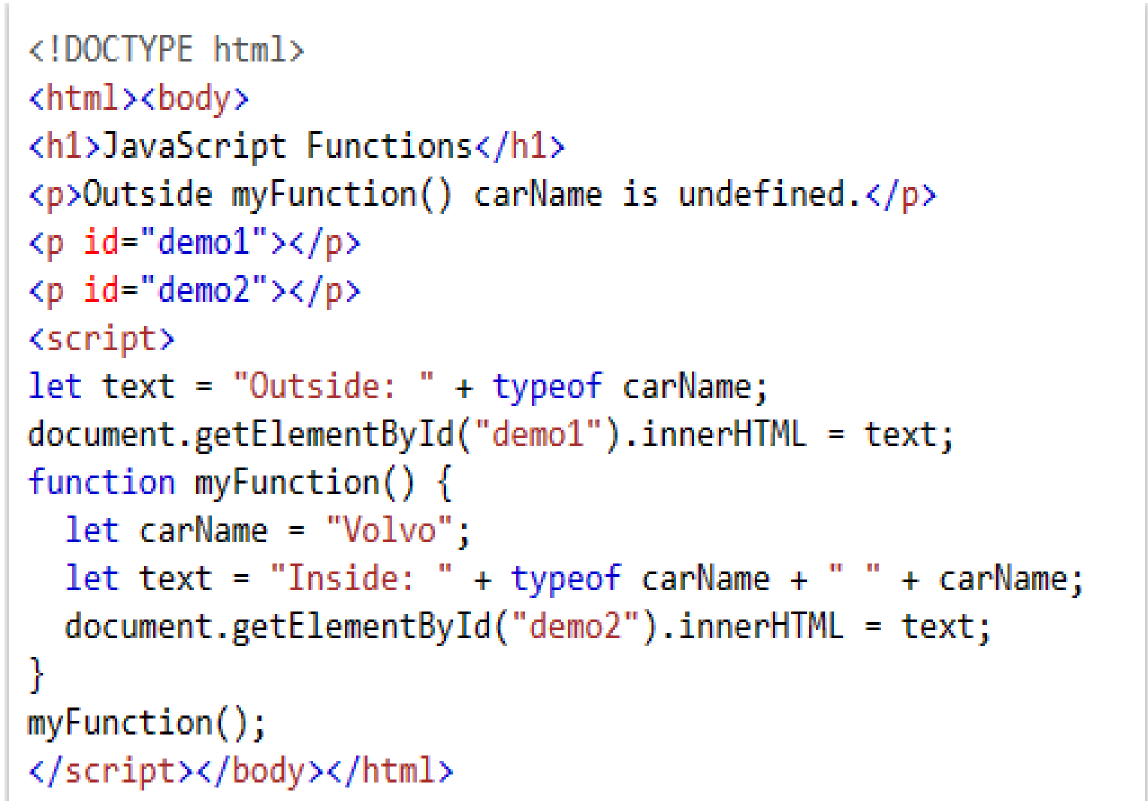
JavaScript Functions

Invoke (call) a function that converts from Fahrenheit to Celsius:

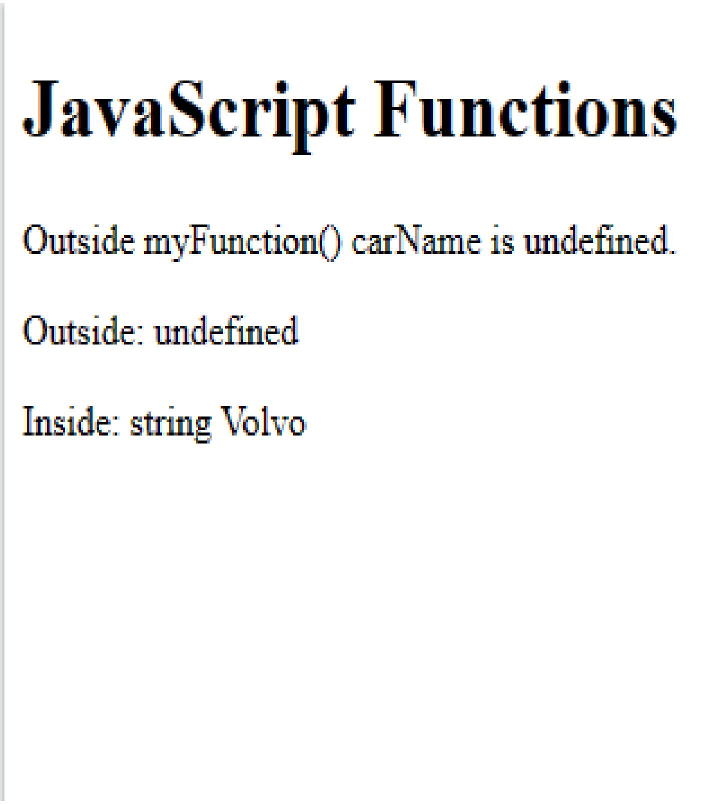
25

Local Variables

Variables declared within a JavaScript function, become **LOCAL** to the function.
Local variables can only be accessed from within the function.



```
<!DOCTYPE html>
<html><body>
<h1>JavaScript Functions</h1>
<p>Outside myFunction() carName is undefined.</p>
<p id="demo1"></p>
<p id="demo2"></p>
<script>
let text = "Outside: " + typeof carName;
document.getElementById("demo1").innerHTML = text;
function myFunction() {
  let carName = "Volvo";
  let text = "Inside: " + typeof carName + " " + carName;
  document.getElementById("demo2").innerHTML = text;
}
myFunction();
</script></body></html>
```



```
JavaScript Functions

Outside myFunction() carName is undefined.

Outside: undefined

Inside: string Volvo
```

Since local variables are only recognized inside their functions, variables with the same name can be used in different functions.
Local variables are created when a function starts, and deleted when the function is completed.

JavaSript in realtime

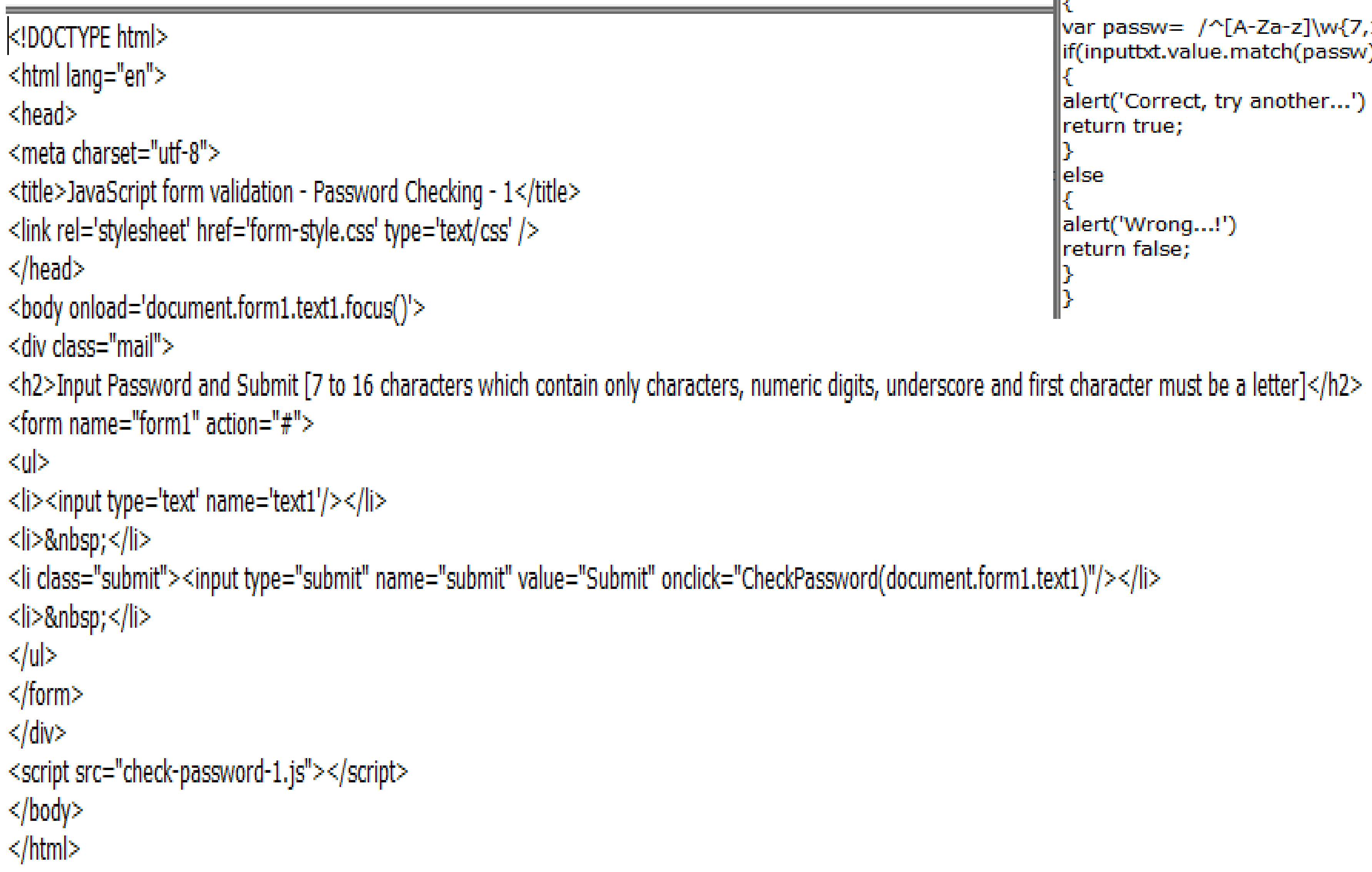
Presentations-RevealJS, BespokeJS

Web Development-create a dynamic and interactive web page.

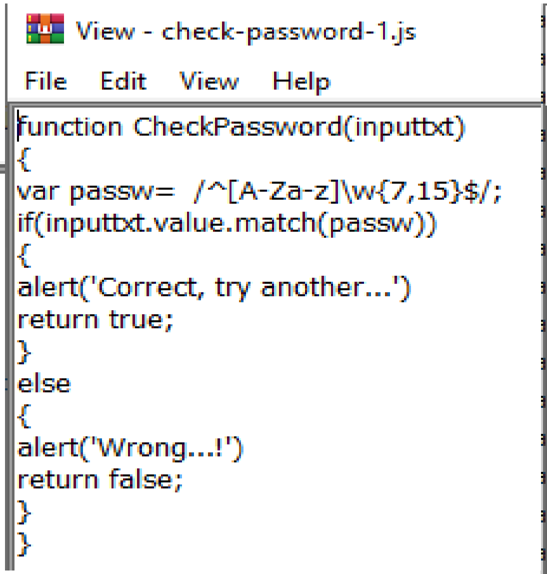
Server Applications-Node.js

Web Applications-React Native serves the purpose of mobile app building, ReactJS is responsible for building user interfaces

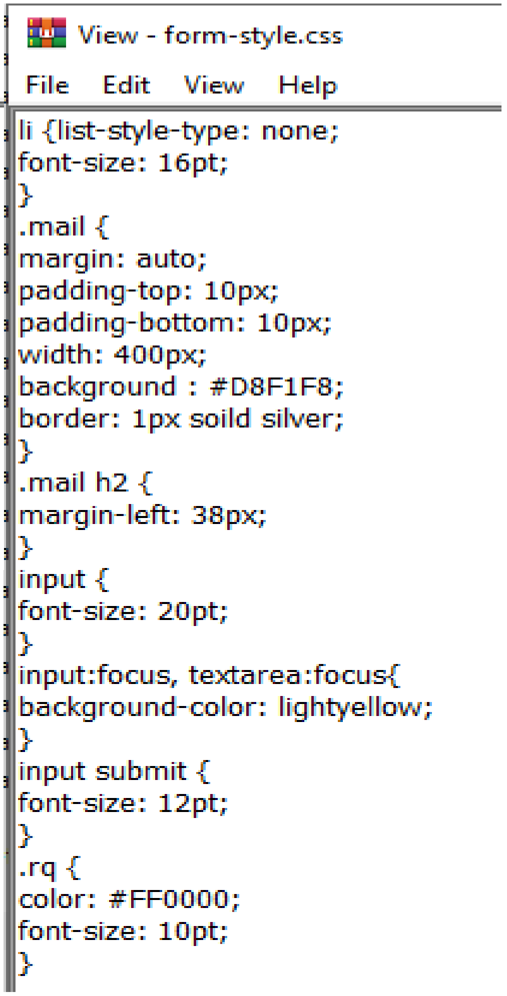
Games-EaselJS library



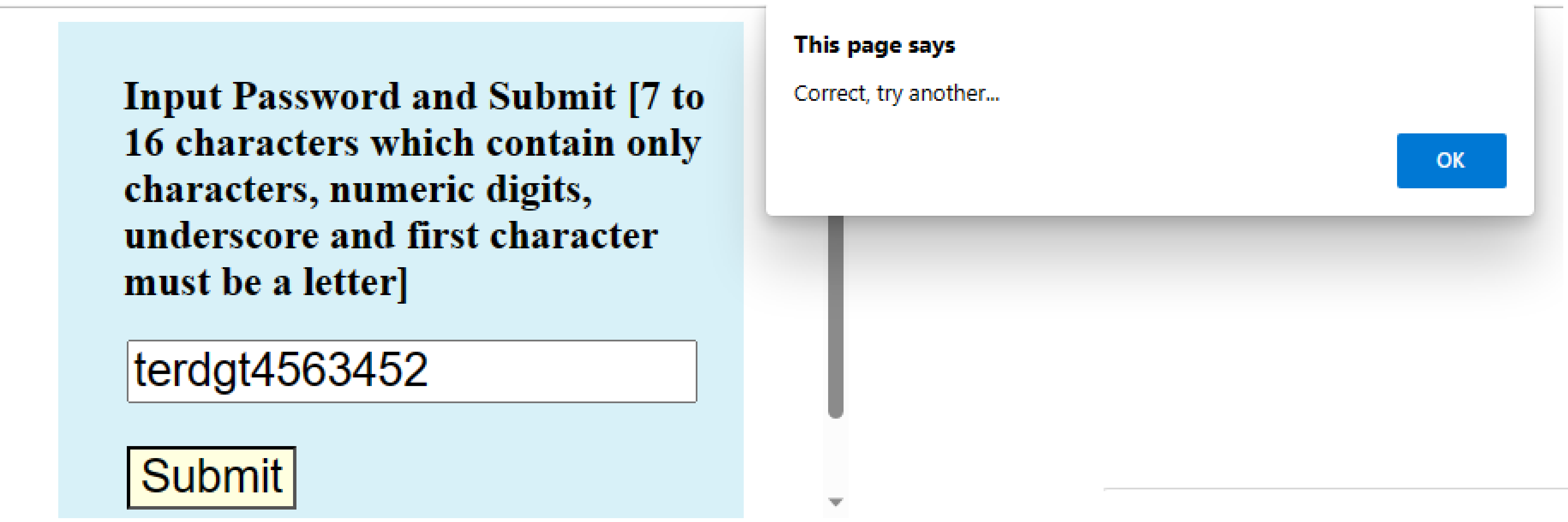
```html
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>JavaScript form validation - Password Checking - 1</title>
<link rel='stylesheet' href='form-style.css' type='text/css' />
</head>
<body onload='document.form1.text1.focus()'>
<div class="mail">
<h2>Input Password and Submit [7 to 16 characters which contain only characters, numeric digits, underscore and first character must be a letter]</h2>
<form name="form1" action="#">
<ul>
<li><input type='text' name='text1'/></li>
<li>&nbsp;</li>
<li class="submit"><input type="submit" name="submit" value="Submit" onclick="CheckPassword(document.form1.text1)"/></li>
<li>&nbsp;</li>
</ul>
</form>
</div>
<script src="check-password-1.js"></script>
</body>
</html>
```



```javascript
function CheckPassword(inputtxt)
{
var passw=  /^[A-Za-z]\w{7,15}$/;
if(inputtxt.value.match(passw))
{
alert('Correct, try another...')
return true;
}
else
{
alert('Wrong...!')
return false;
}
}
```



```css
li {list-style-type: none;
font-size: 16pt;
}
.mail {
margin: auto;
padding-top: 10px;
padding-bottom: 10px;
width: 400px;
background : #D8F1F8;
border: 1px soild silver;
}
.mail h2 {
margin-left: 38px;
}
input {
font-size: 20pt;
}
input:focus, textarea:focus{
background-color: lightyellow;
}
input submit {
font-size: 12pt;
}
.rq {
color: #FF0000;
font-size: 10pt;
}
```



Input Password and Submit [7 to 16 characters which contain only characters, numeric digits, underscore and first character must be a letter]

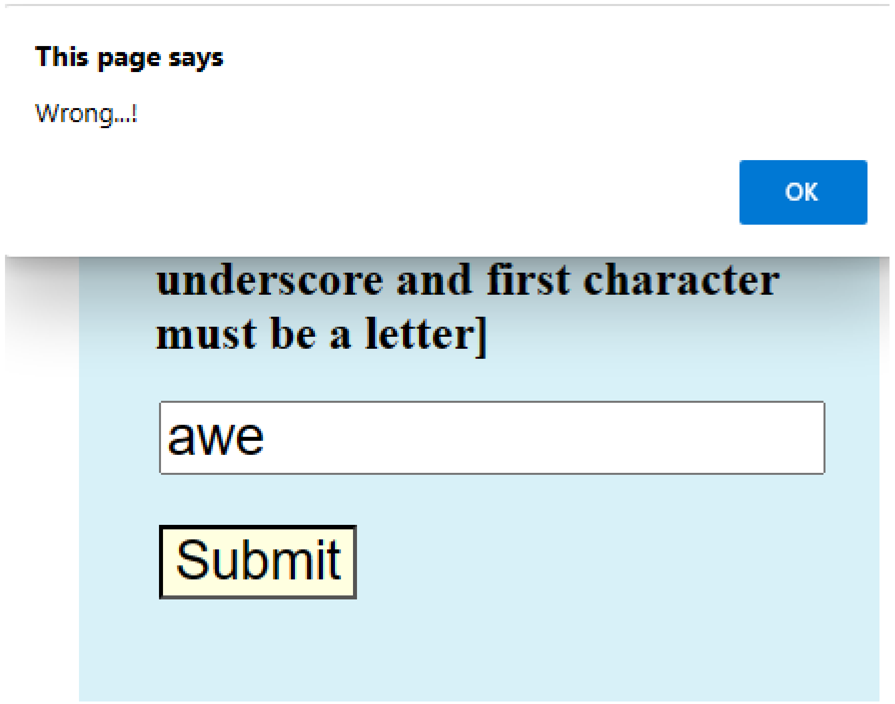
terdgt4563452

Submit

This page says

Correct, try another...

OK



This page says

Wrong...!

OK

underscore and first character must be a letter]

awe

Submit