UNIT 4: Client Side Scripting [15 hrs]

By: Yuba Raj Devkota

Unit 4: Client-Side Scripting 15 LHs

Introduction; Adding JavaScript to a Page; Output; Comments; Variables and Data Types; Operators; Control Statements; Functions; Arrays; Classes and Objects; Built-in Objects; Event Handling and Form Validation, Error Handling, Handling Cookies; DOM; BOM; Basics of jQuery, React, and AngularJS, AJAX, and JSON.

What is Scripting Language?

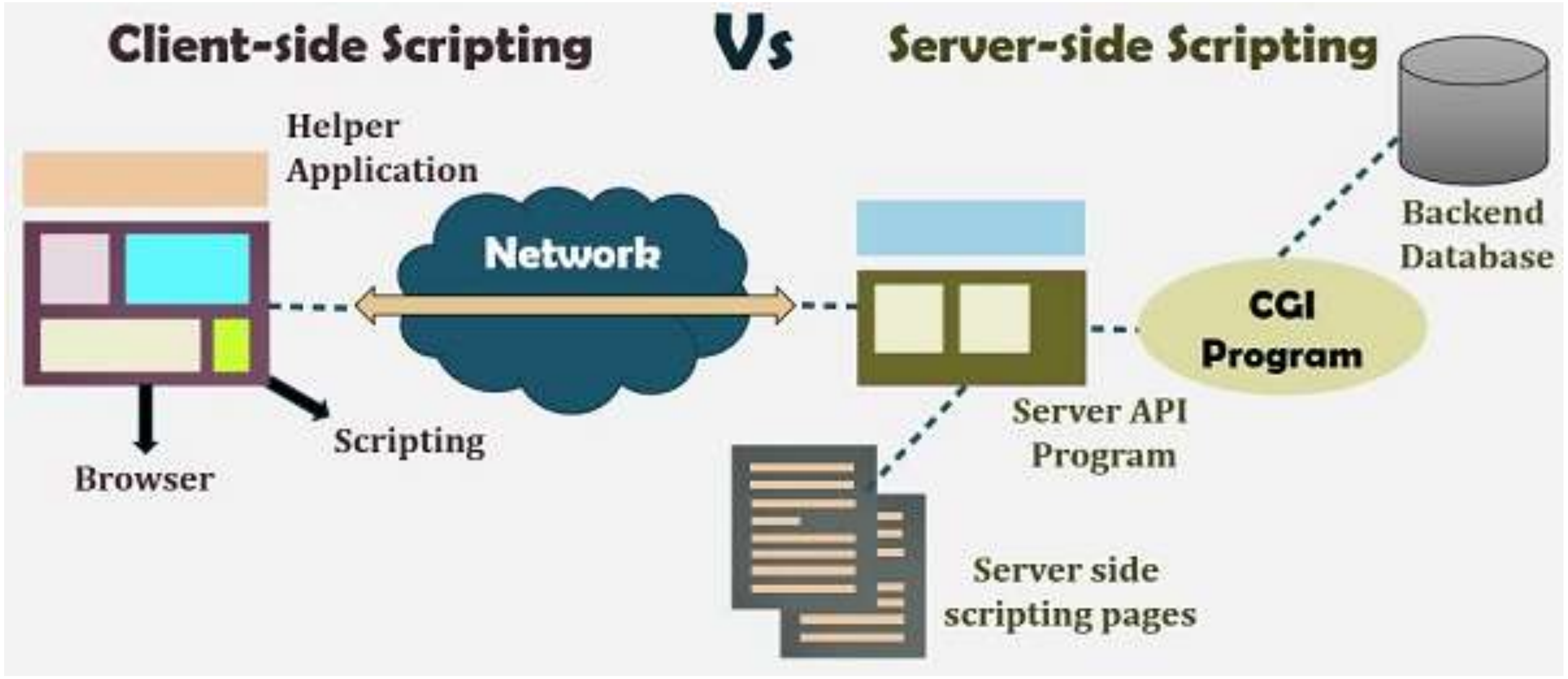
- The scripting language is basically a language where instructions are written for a run time environment. They do not require the compilation step and are rather interpreted.
- A scripting language is a programming language designed for integrating and communicating with other programming languages.
- Some of the Scripting Languages are:
  - **bash**: used in linux platform.
  - **Node js**: Framework to write network applications using Javascript. Some example includes IBM, LinkedIn, Microsoft, Netflix, PayPal, Yahoo for real-time web applications.
  - **Ruby**: Ruby's flexibility has allowed developers to create innovative software. It is a scripting language which is great for web development.
  - **Python**: It is easy, free and open source. It supports procedure-oriented programming and object-oriented programming. Python is an interpreted language with dynamic semantics and huge lines of code are scripted and is currently the most hyped language among developers.
  - **Perl**: A scripting language with innovative features to make it different and popular. Found on all windows and Linux servers. It helps in text manipulation tasks. High traffic websites that use Perl extensively include priceline.com, IMDB.

Scripting Language VS Programming Languages

- Basically, all scripting languages are programming languages.
- The theoretical difference between the two is that scripting languages do not require the compilation step and are rather interpreted. For example, normally, a C program needs to be compiled before running whereas normally, a scripting language like JavaScript or PHP need not be compiled.
- Generally, compiled programs run faster than interpreted programs because they are first converted native machine code.
- Also, compilers read and analyze the code only once, and report the errors collectively that the code might have, but the interpreter will read and analyze the code statements each time it meets them and halts at that very instance if there is some error.
- Some scripting languages traditionally used without an explicit compilation step are JavaScript, PHP, Python, VBScript.
- Some programming languages traditionally used with an explicit compilation step are C, C++.

Client Side Scripting / Javascript

- What is Client Side Scripting vs Server Side Scripting?



Need of Client Scripting Language

The two main benefits of client-side scripting are:

- The user’s actions will result in an immediate response because they don’t require a trip to the server.
- Fewer resources are used and needed on the web-server.

The two huge advances in client-side scripting are **Ajax** and **jQuery**. Ajax is not a tool or a programing language, but a concept. The concept is to call the server directly from the client without doing a PostBack. The main advantage of Ajax is to save and retrieve data from a database, bypassing the web server, which is known as a **CallBack**.

jQuery is a multi-browser JavaScript library designed to simplify client-side scripting and serves as is a wrapper around JavaScript. jQuery is free open source software.

About Javascript

- Javascript is the most popular scripting language in the world
- It helps you developing great front end as well as back end software using different javascript based frameworks like jQuery, NodeJS etc
- You do not need any special software for javascript because all main web browsers support it.
- You can also create web applications, Mobile Apps and Games using Javascript.
- A very good thing about Javascript is that, in Javascript you can save time by using the already created Libraries and Frameworks

Application of JavaScript

JavaScript (js) is a light-weight object-oriented programming language which is used by several websites for scripting the webpages. There are following features of JavaScript:

- All popular web browsers support JavaScript
- JavaScript follows the syntax and structure of the C programming language. It is a light-weighted and interpreted language.
- It is a case-sensitive language.
- JavaScript is supportable in several operating systems including, Windows, macOS, etc.

Application of JavaScript

- Client-side validation,
- Dynamic drop-down menus,
- Displaying date and time,
- Displaying pop-up windows and dialog boxes
- Displaying clocks etc.

JavaScript Syntax

- Declaring and Assigning Variables

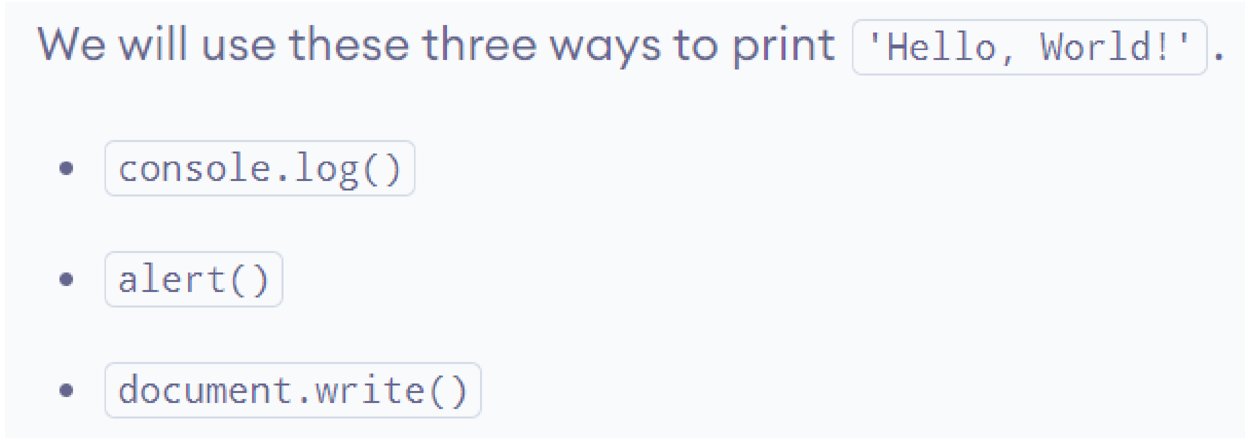
  -
    ```
    var x, y, z;      // How to declare variables
    x = 5; y = 6;     // How to assign values
    z = x + y;        // How to compute values
    ```
- Numbers are written with or without decimals:
  - 10.50 or 1001 both are fine
- Strings are text, written within double or single quotes:
  - “Ram Prasad“ or ‘Ram Prasad‘ both are fine
- JavaScript uses arithmetic operators (+ - * /) to compute values:
  - (5 + 6) * 10
- All JavaScript identifiers are **case sensitive**.
  - The variables lastName and lastname, are two different variables:
- Code after double slashes // or between /* and */ is treated as a comment. Comments are ignored, and will not be executed:

  -
    ```
    var x = 5;   // It will be executed
    // var x = 6;   It will NOT be executed
    ```

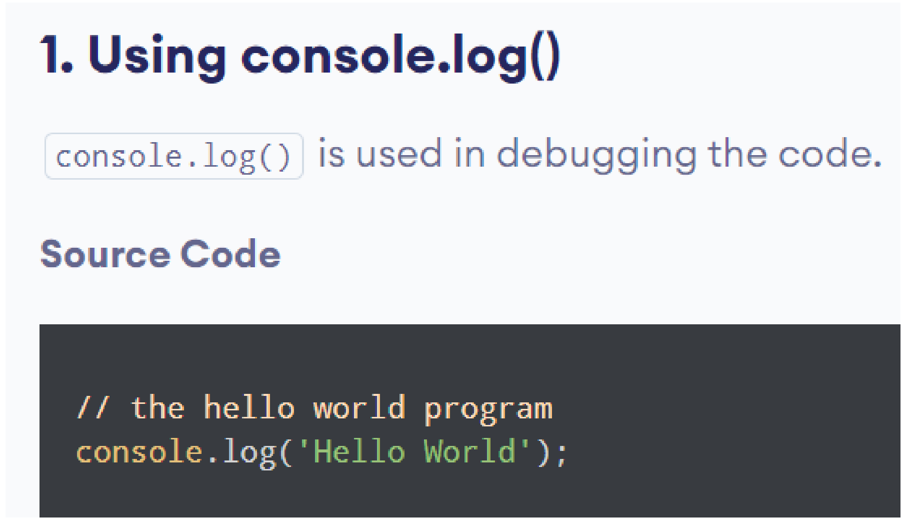
Lab Exercises

17. Add two numbers entered by user and print the output.
18. Calculate the area of tringle when all sides are known. [s=(a+b+c)/2] and [area = square root (s*(s-a)*(s-b)*(s-c))]
19. Find whether the given number is odd or even
20. write a JavaScript Program to find the largest number among three numbers.
21. Find the Factorial of a Number using JavaScript

Print Hello World


We will use these three ways to print `'Hello, World!'`.

- `console.log()`
- `alert()`
- `document.write()`


1. Using console.log()

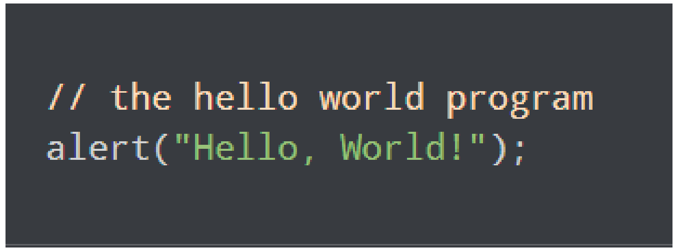
`console.log()` is used in debugging the code.

Source Code

```
// the hello world program
console.log('Hello World');
```

2. Using alert()

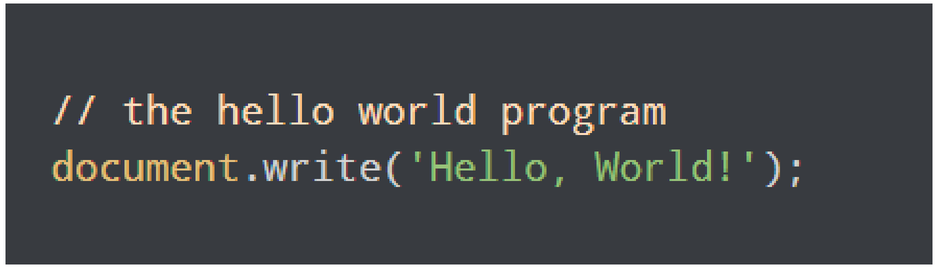
This method displays an alert box over the current window with the specified message.



```
// the hello world program
alert("Hello, World!");
```

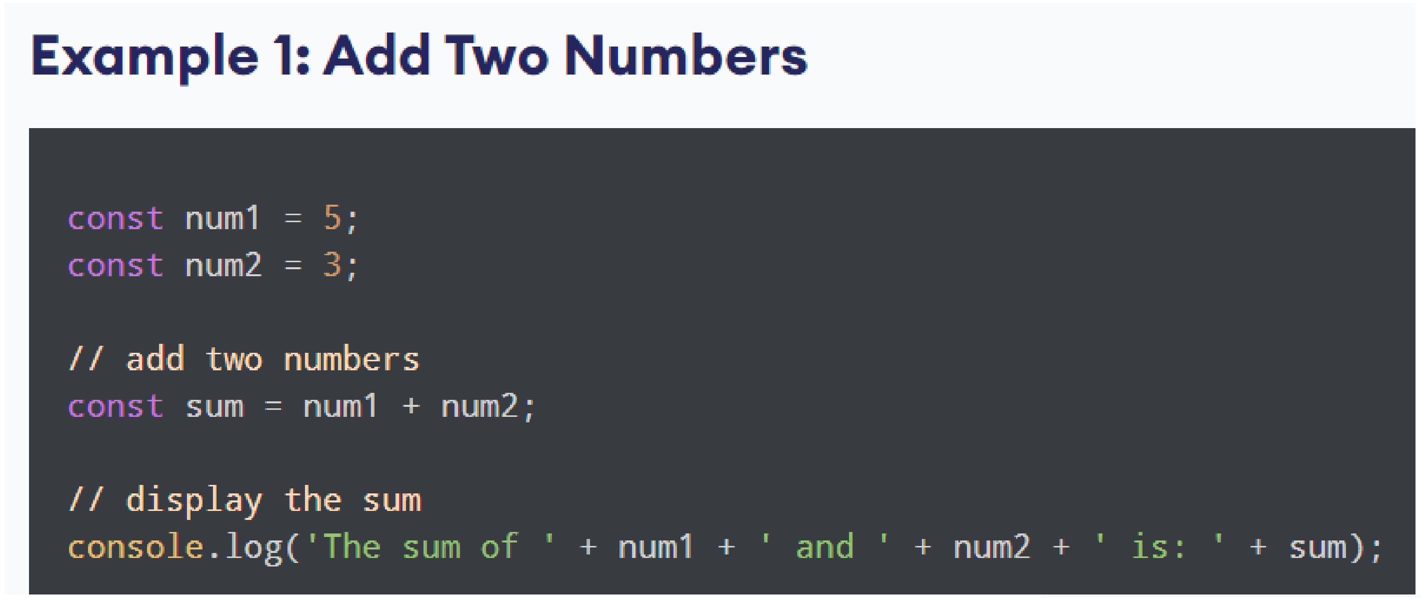
3. Using document.write()

This method used when you want to print the content to the HTML document.



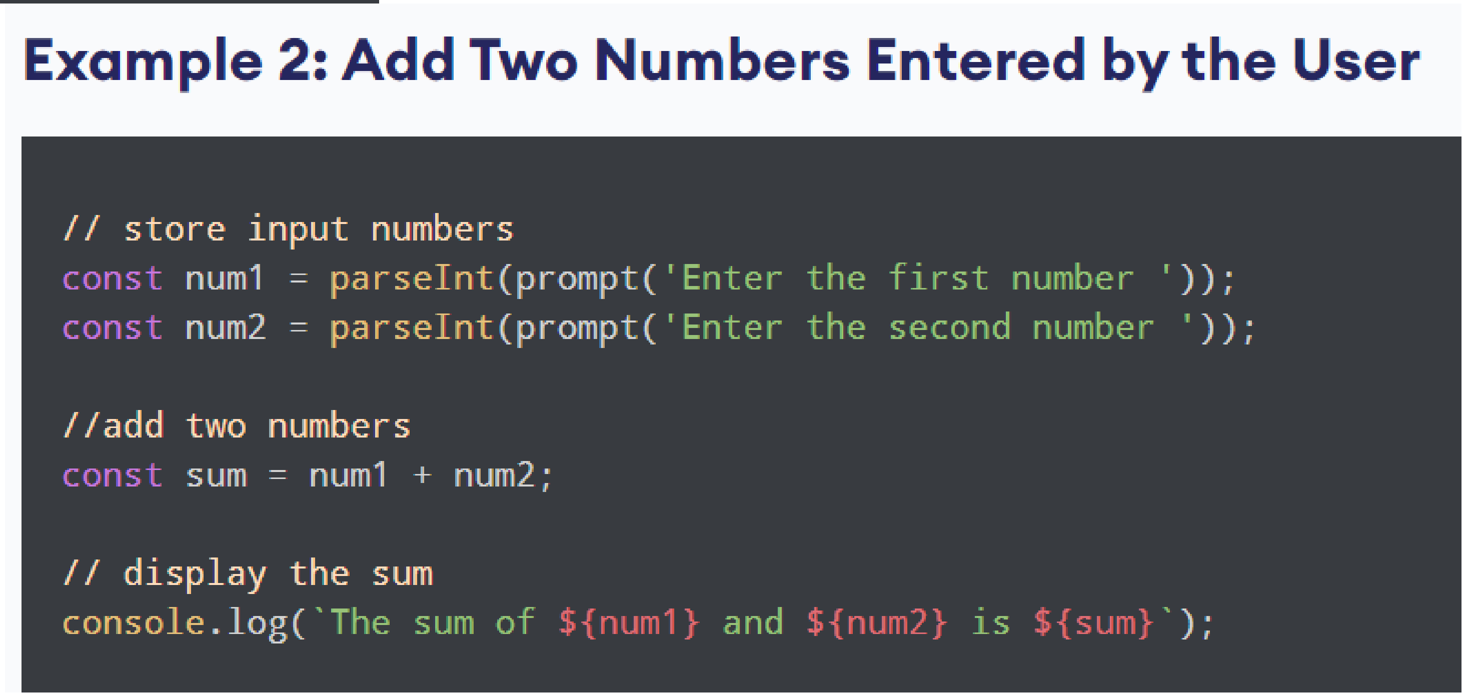
```
// the hello world program
document.write('Hello, World!');
```

Add Two Numbers in JavaScript



Example 1: Add Two Numbers

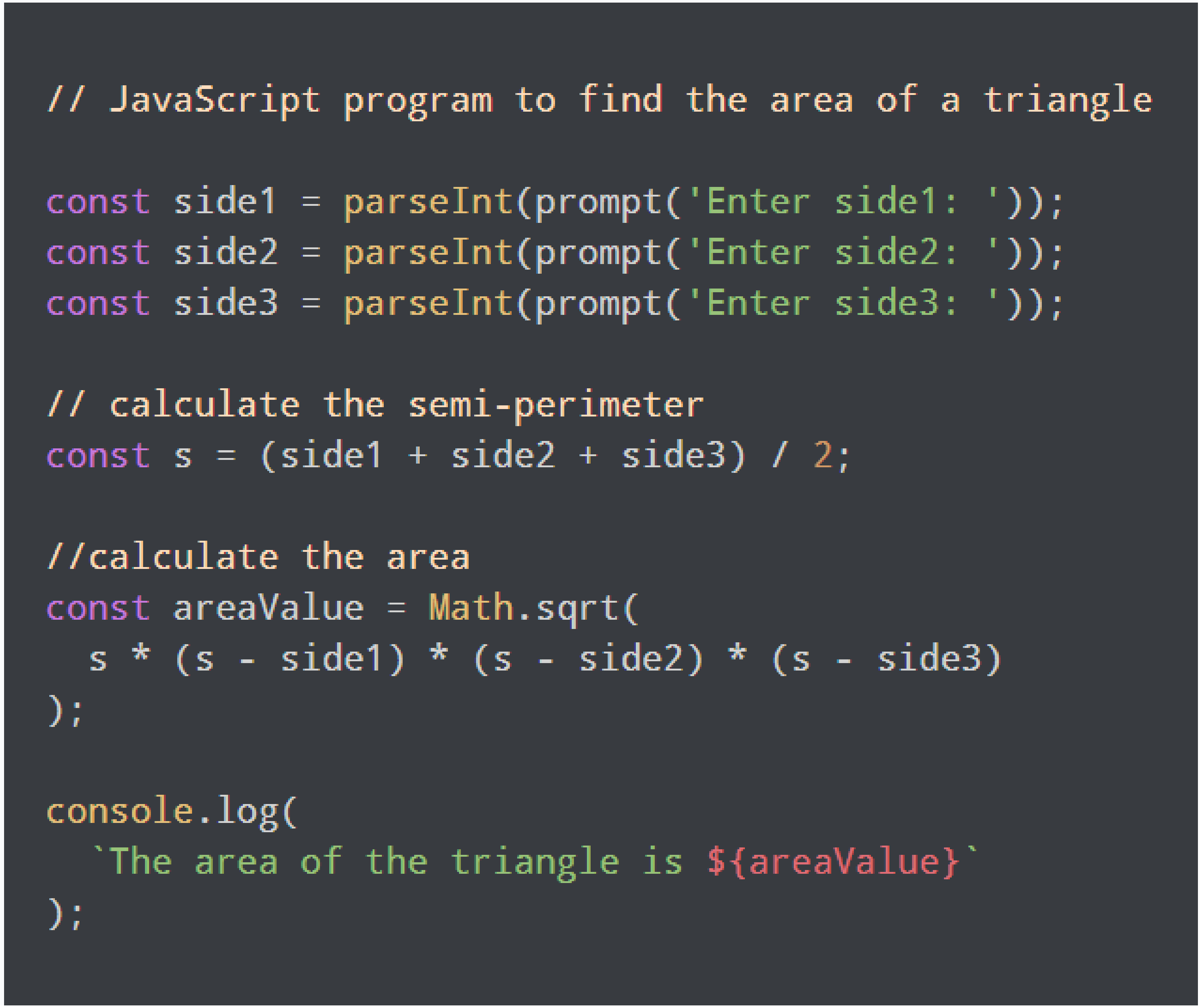
```javascript
const num1 = 5;
const num2 = 3;

// add two numbers
const sum = num1 + num2;

// display the sum
console.log('The sum of ' + num1 + ' and ' + num2 + ' is: ' + sum);
```



Example 2: Add Two Numbers Entered by the User

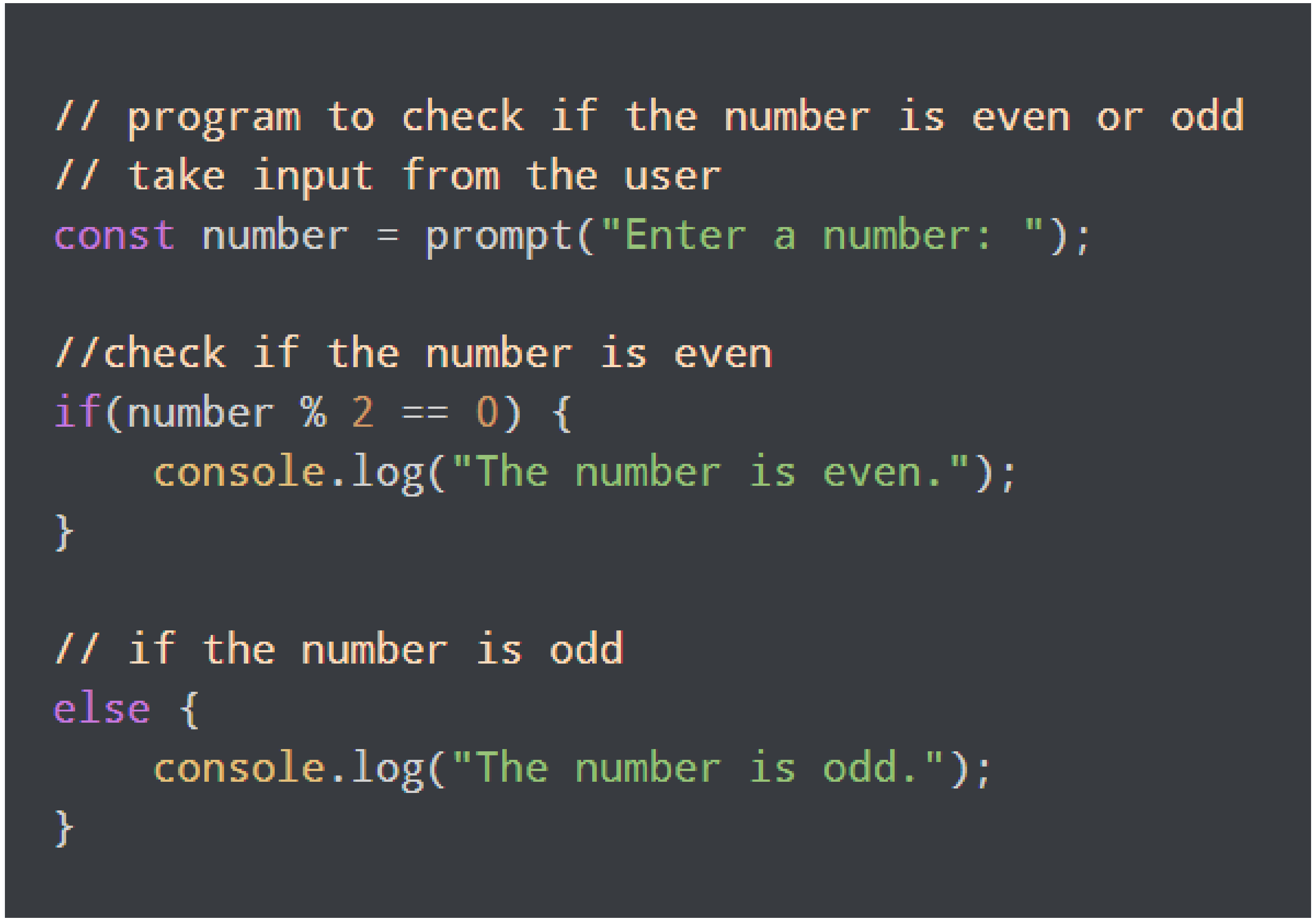
```javascript
// store input numbers
const num1 = parseInt(prompt('Enter the first number '));
const num2 = parseInt(prompt('Enter the second number '));

//add two numbers
const sum = num1 + num2;

// display the sum
console.log(`The sum of ${num1} and ${num2} is ${sum}`);
```

Area of tringle when all sides are given



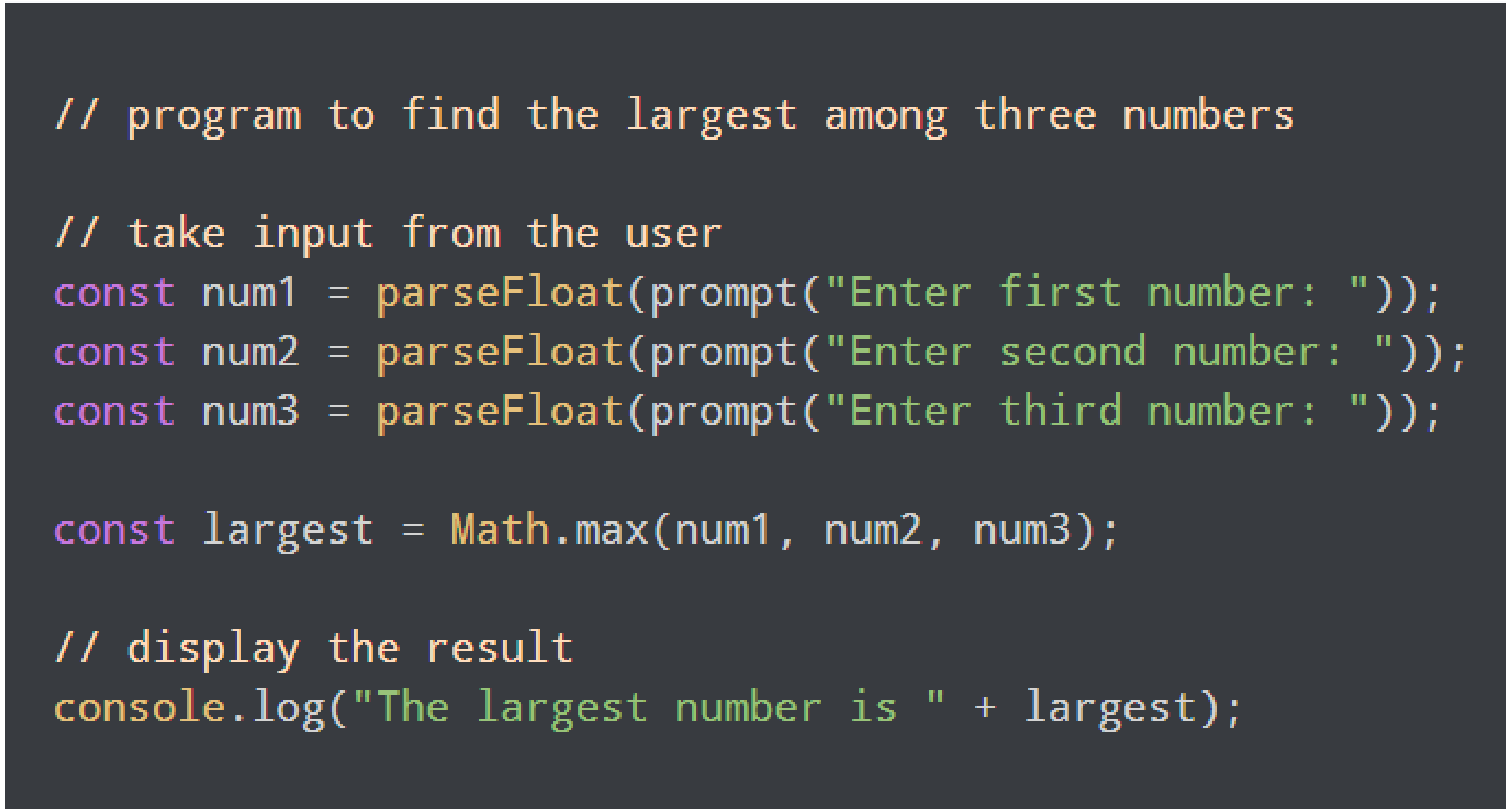
```javascript
// JavaScript program to find the area of a triangle

const side1 = parseInt(prompt('Enter side1: '));
const side2 = parseInt(prompt('Enter side2: '));
const side3 = parseInt(prompt('Enter side3: '));

// calculate the semi-perimeter
const s = (side1 + side2 + side3) / 2;

//calculate the area
const areaValue = Math.sqrt(
  s * (s - side1) * (s - side2) * (s - side3)
);

console.log(
  `The area of the triangle is ${areaValue}`
);
```

Odd or Even number



```
// program to check if the number is even or odd
// take input from the user
const number = prompt("Enter a number: ");

//check if the number is even
if(number % 2 == 0) {
    console.log("The number is even.");
}

// if the number is odd
else {
    console.log("The number is odd.");
}
```

Largest among three numbers



```
// program to find the largest among three numbers

// take input from the user
const num1 = parseFloat(prompt("Enter first number: "));
const num2 = parseFloat(prompt("Enter second number: "));
const num3 = parseFloat(prompt("Enter third number: "));

const largest = Math.max(num1, num2, num3);

// display the result
console.log("The largest number is " + largest);
```

Where to put JavaScript code in HTML?

3 Places to put JavaScript code

- Internal
  - Between the body tag of html
  - Between the head tag of html
- External
  - In .js file (external javaScript)

1) code between the body tag

```
<html>
<body>
<script type="text/javascript">
 alert("Hello Javatpoint");
</script>
</body>
</html>
```

2) JavaScript Example : code between the head tag

```
<html>
<head>
<script type="text/javascript">
function msg(){
 alert("Hello Javatpoint");
}
</script>
</head>
<body>
<p>Welcome to JavaScript</p>
<form>
<input type="button" value="click" onclick="msg()"/>

</form>
</body>
</html>
```

3) External js file

message.js

```
function msg(){
 alert("Hello Javatpoint");
}
```

index.html

```
<html>
<head>
<script type="text/javascript" src="message.js"></script>
</head>
<body>
<p>Welcome to JavaScript</p>
<form>
<input type="button" value="click" onclick="msg()"/>
</form>
</body>
</html>
```

Advantages of External JavaScript

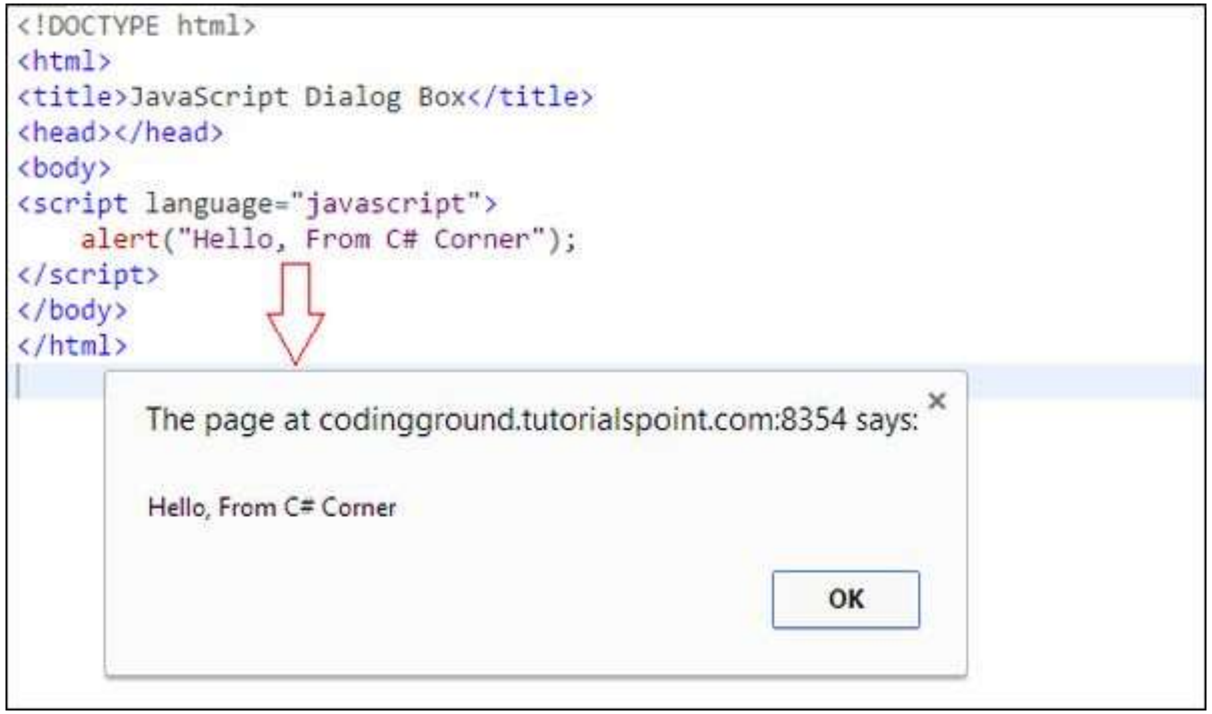
1.It helps in the reusability of code in more than one HTML file.
2.It allows easy code readability.
3.It is time-efficient as web browsers cache the external js files, which further reduces the page loading time.
4.It enables both web designers and coders to work with html and js files parallelly and separately, i.e., without facing any code conflictions.
5.The length of the code reduces as only we need to specify the location of the js file.

JavaScript Popup Boxes

JavaScript has three kind of 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.

```
alert("I am an alert box!");
```



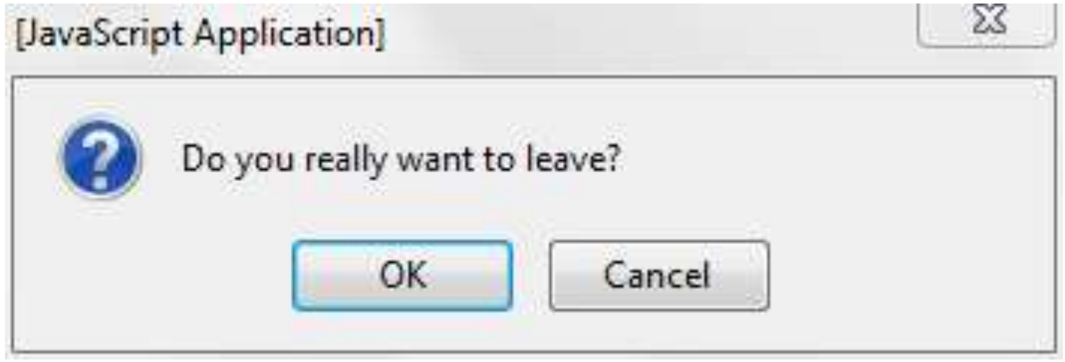
```
<!DOCTYPE html>
<html>
<title>JavaScript Dialog Box</title>
<head></head>
<body>
<script language="javascript">
    alert("Hello, From C# Corner");
</script>
</body>
</html>
```

The page at codingground.tutorialspoint.com:8354 says:

Hello, From C# Corner

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.

```
if (confirm("Press a button!")) {
  txt = "You pressed OK!";
} else {
  txt = "You pressed Cancel!";
}
```

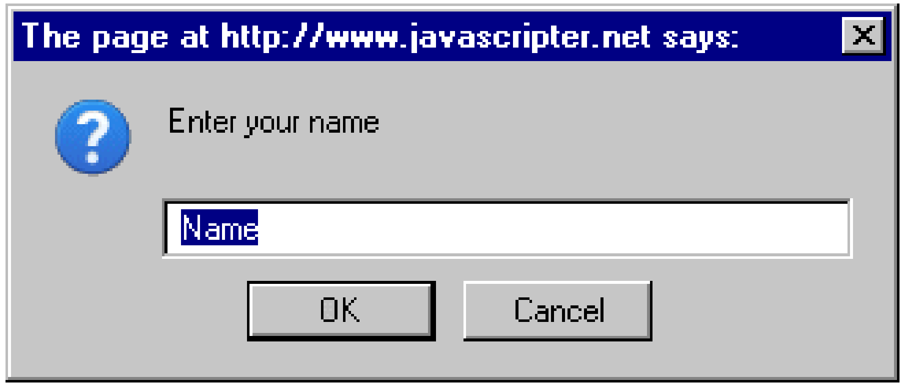


[JavaScript Application]

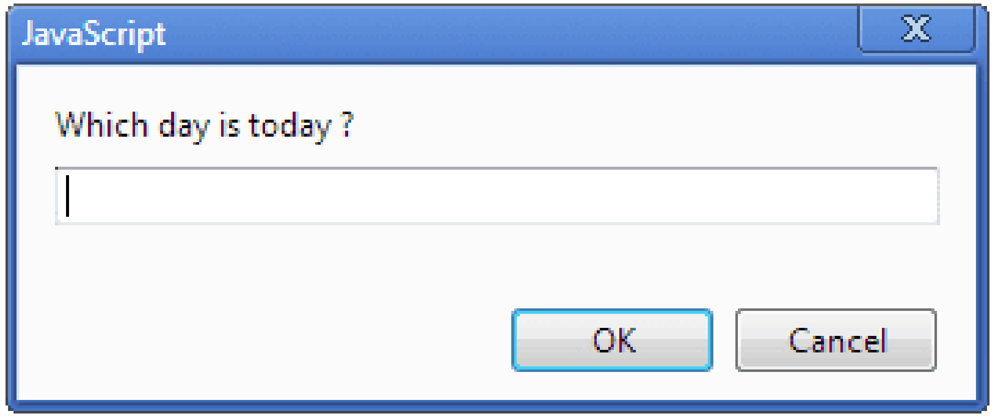
Do you really want to leave?

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.

```
var person = prompt("Please enter your name", "Harry Potter");

if (person == null || person == "") {
  txt = "User cancelled the prompt.";
} else {
  txt = "Hello " + person + "! How are you today?";
}
```


The page at http://www.javascripter.net says:

Enter your name


JavaScript

Which day is today ?

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).
- Syntax

```
function myFunction(p1, p2) {
        return p1 * p2;   // The function returns the product of p1 and p2
}
```

Example

Calculate the product of two numbers, and return the result:

```
var x = myFunction(4, 3);   // Function is called, return value will
end up in x

function myFunction(a, b) {
  return a * b;          // Function returns the product of a and b
}
```

The result in x will be: 12

Conditional Statement in JavaScript

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 select one of many blocks of code to be executed

The **if** statement specifies a block of code to be executed if a condition is true:

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

The **else** statement specifies 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
}
```

The **else if** statement specifies a new condition if the first condition is false:

```
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
}
```

If the time is less than 20:00, create a "Good day" greeting, otherwise "Good evening":

```
var time = new Date().getHours();
if (time < 20) {
  greeting = "Good day";
} else {
  greeting = "Good evening";
}
```

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":

```
var time = new Date().getHours();
if (time < 10) {
  greeting = "Good morning";
} else if (time < 20) {
  greeting = "Good day";
} else {
  greeting = "Good evening";
}
```

Question:

What is the difference between If-else-if ladder and Nested if else?

JavaScript switch Statement

- The switch statement executes a block of code depending on different cases.
- The switch statement is often used together with a break or a default keyword (or both). These are both optional:
  - The **break** keyword breaks out of the switch block. This will stop the execution of more execution of code and/or case testing inside the block. If break is omitted, the next code block in the switch statement is executed.
  - The **default** keyword specifies some code to run if there is no case match. There can only be one default keyword in a switch. Although this is optional, it is recommended that you use it, as it takes care of unexpected cases.

Syntax:

```
switch(expression) {
  case n:
    code block
    break;
  case n:
    code block
    break;
  default:
    default code block
}
```

Example:

```
var text;
switch (new Date().getDay()) {
  case 6:
    text = "Today is Saturday";
    break;
  case 0:
    text = "Today is Sunday";
    break;
  default:
    text = "Looking forward to the Weekend";
}
```

JavaScript Loops

- For Loop

  -
    ```
    var i;
    for (i = 0; i < cars.length; i++) {
      text += cars[i] + "<br>";
    }
    ```
- While Loop

  -
    ```
    while (i < 10) {
      text += "The number is " + i;
      i++;
    }
    ```
- Do While Loop

  -
    ```
    do {
      text += "The number is " + i;
      i++;
    }
    while (i < 10);
    ```

Create a loop that runs as long as i is less than 10, but increase i with 2 each time.

```
var i = 0;
while (i < 10) {
  console.log(i);
  i = i+2;
}
```

Create a loop that runs through each item in the fruits array.

```
var fruits = ["Apple", "Banana", "Orange"];
for (x of fruits) {
  console.log(x);
}
```

Javascript Array

JavaScript array is an object that represents a collection of similar type of elements. There are 3 ways to construct array in JavaScript

- By array literal
- By creating instance of Array directly (using new keyword)
- By using an Array constructor (using new keyword)

1) JavaScript array literal

Syntax: var arrayname=[value1,value2.....valueN];
values are contained inside [ ] and separated by , (comma).

The .length property returns the length of an array.

```
<script>
        var emp=[“Messi",“Ronaldo",“Pele"];

        for (i=0;i<emp.length;i++){
                document.write(emp[i] + "<br/>");
        }
</script>
```

Output of the above example

```
Messi
Ronaldo
Pele
```

2) JavaScript Array directly (new keyword)

```
var arrayname=new Array();
```

Here, new keyword is used to create instance of array.

```
<script>
        var i;
        var emp = new Array();
        emp[0] = "Arun";
        emp[1] = "Varun";
        emp[2] = "John";

        for (i=0;i<emp.length;i++){
                document.write(emp[i] + "<br>");
        }
</script>
```

Output

```
Arun
Varun
John
```

3) JavaScript array constructor (new keyword)

Here, you need to create instance of array by passing arguments in constructor so that we don't have to provide value explicitly.

```
<script>
        var emp=new Array("Jai","Vijay","Smith");
        for (i=0;i<emp.length;i++){
                document.write(emp[i] + "<br>");
        }
</script>
```

Output

```
Jai
Vijay
Smith
```

Array Methods

- Converting Arrays to Strings: The JavaScript method toString() converts an array to a string of (comma separated) array values.
  - var fruits = ["Banana", "Orange", "Apple", "Mango"];
  - document.getElementById("demo").innerHTML = fruits.toString();
  - Result:Banana,Orange,Apple,Mango

- Popping and Pushing: Popping items out of an array, or pushing items into an array.
  - var fruits = ["Banana", "Orange", "Apple", "Mango"];
  - fruits.pop();   // Removes the last element ("Mango") from fruits

  - var fruits = ["Banana", "Orange", "Apple", "Mango"];
  - fruits.push("Kiwi");   // Adds a new element ("Kiwi") to fruits

- Changing Elements: Array elements are accessed using their index number:
  - var fruits = ["Banana", "Orange", "Apple", "Mango"];
  - fruits[0] = "Kiwi";   // Changes the first element of fruits to "Kiwi"

String Methods

- Finding String Length: The length property returns the length of a string:
  - var txt = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
  - var sln = txt.length;

- Finding a String in a String: The indexOf() method returns the index of (the position of) the first occurrence of a specified text in a string:
  - var str = "Please locate where 'locate' occurs!";
  - var pos = str.indexOf("locate");

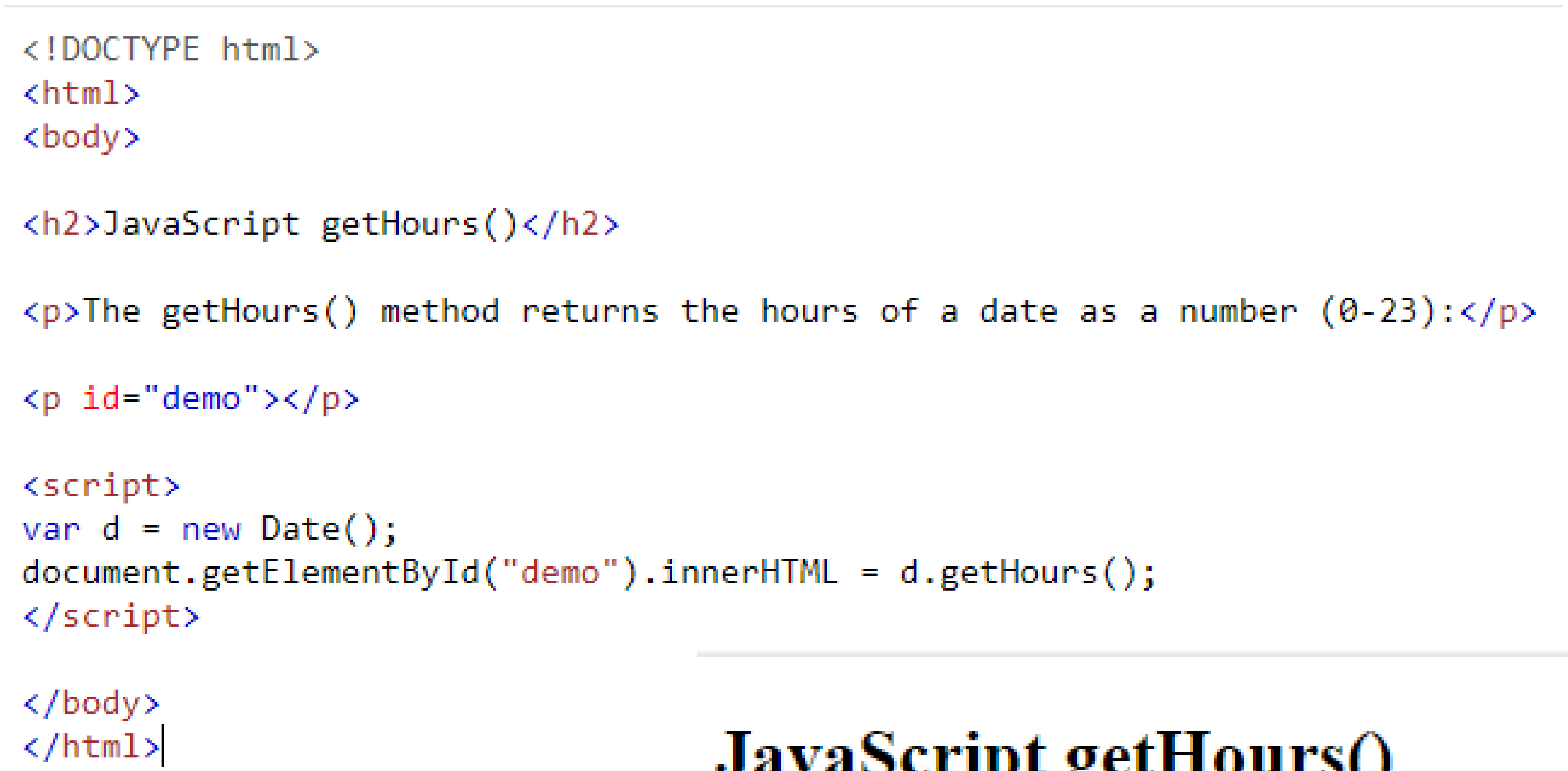
- The substring()/substr() Method:
  - var str = "Apple, Banana, Kiwi";
  - var res = str.substring(7, 13);
  - var res1 = str.substr(7, 6);
  - The result of res & res1 will be: Banana

- concat() joins two or more strings:
  - var text1 = "Hello";
  - var text2 = "World";
  - var text3 = text1.concat(" ", text2);

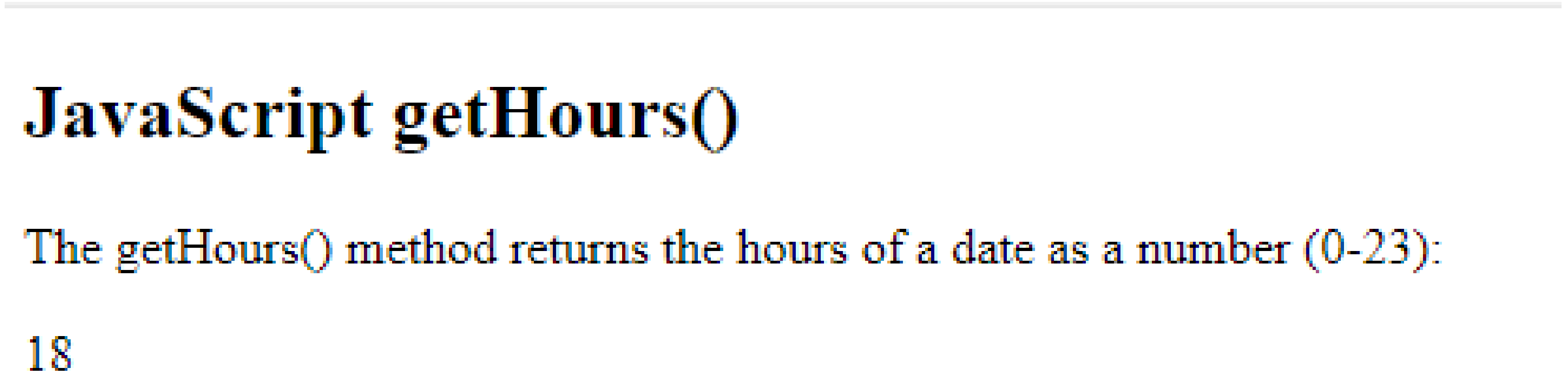
Date Functions in Javascript

- Date objects are created with the new Date() constructor.
- var d = new Date();
- By default, JavaScript will use the browser's time zone and display a date as a full text string:
- **Wed Sep 02 2020 18:23:56 GMT+0545 (Nepal Time)**
- JavaScript Get Date Methods

getFullYear()	Get the **year** as a four digit number (yyyy)
getMonth()	Get the **month** as a number (0-11)
getDate()	Get the **day** as a number (1-31)
getHours()	Get the **hour** (0-23)
getMinutes()	Get the **minute** (0-59)
getSeconds()	Get the **second** (0-59)
getMilliseconds()	Get the **millisecond** (0-999)
getTime()	Get the time (milliseconds since January 1, 1970)
getDay()	Get the weekday as a number (0-6)



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

<h2>JavaScript getHours()</h2>

<p>The getHours() method returns the hours of a date as a number (0-23):</p>

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

<script>
var d = new Date();
document.getElementById("demo").innerHTML = d.getHours();
</script>

</body>
</html>
```



JavaScript getHours()

The getHours() method returns the hours of a date as a number (0-23):

18

JS HTML DOM (Document Object Model)

- The HTML DOM is a standard **object** model and **programming interface** for HTML. It defines:
  - The HTML elements as **objects**
  - The **properties** of all HTML elements
  - The **methods** to access all HTML elements
  - The **events** for all HTML elements
- In other words: **The HTML DOM is a standard for how to get, change, add, or delete HTML elements.**

Changing HTML Content

The easiest way to modify the content of an HTML element is by using the innerHTML property.

To change the content of an HTML element, use this syntax:

```
document.getElementById(id).innerHTML = new HTML
```

```
<html>
<body>

<p id="p1">Hello World!</p>

<script>
document.getElementById("p1").innerHTML = "New text!";
</script>

</body>
</html>
```

Changing the Value of an Attribute

To change the value of an HTML attribute, use this syntax:
document.getElementById(*id*).*attribute = new value*

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

<img id="myImage" src="smiley.gif">

<script>
document.getElementById("myImage").src = "landscape.jpg";
</script>

</body>
</html>
```

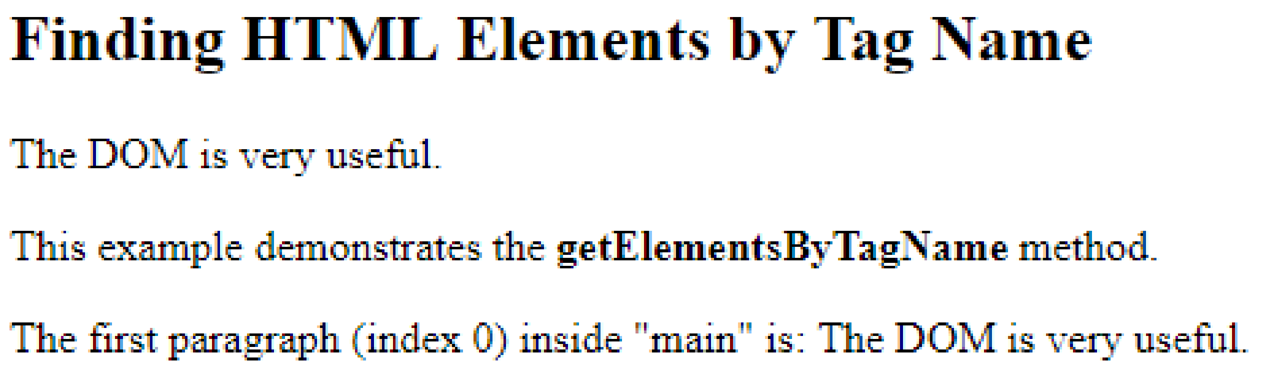
Finding HTML Elements using DOM

- Often, with JavaScript, you want to manipulate HTML elements. To do so, you have to find the elements first. There are several ways to do this:
  - Finding HTML elements by id
  - Finding HTML elements by tag name
  - Finding HTML elements by class name
  - Finding HTML elements by CSS selectors
  - Finding HTML elements by HTML object collections

The easiest way to find an HTML element in the DOM, is by using the element id, Tag Name or Class Name

```
var x = document.getElementById("main");
var y = x.getElementsByTagName("p");
var x = document.getElementsByClassName("intro");
```

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

<h2>Finding HTML Elements by Tag Name</h2>

<div id="main">
<p>The DOM is very useful.</p>
<p>This example demonstrates the <b>getElementsByTagName</b> method.</p>
</div>

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

<script>
var x = document.getElementById("main");
var y = x.getElementsByTagName("p");
document.getElementById("demo").innerHTML =
'The first paragraph (index 0) inside "main" is: ' + y[0].innerHTML;
</script>

</body>
</html>
```



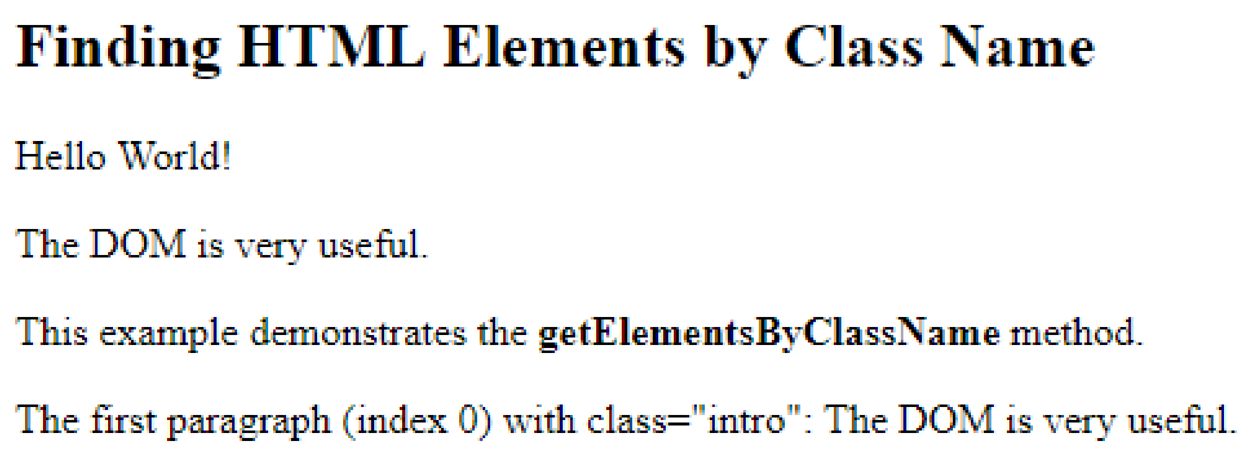
Finding HTML Elements by Tag Name

The DOM is very useful.

This example demonstrates the **getElementsByTagName** method.

The first paragraph (index 0) inside "main" is: The DOM is very useful.

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

<h2>Finding HTML Elements by Class Name</h2>

<p>Hello World!</p>

<p class="intro">The DOM is very useful.</p>
<p class="intro">This example demonstrates the <b>getElementsByClassName</b>
method.</p>

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

<script>
var x = document.getElementsByClassName("intro");
document.getElementById("demo").innerHTML =
'The first paragraph (index 0) with class="intro": ' + x[0].innerHTML;
</script>

</body>
</html>
```



Finding HTML Elements by Class Name

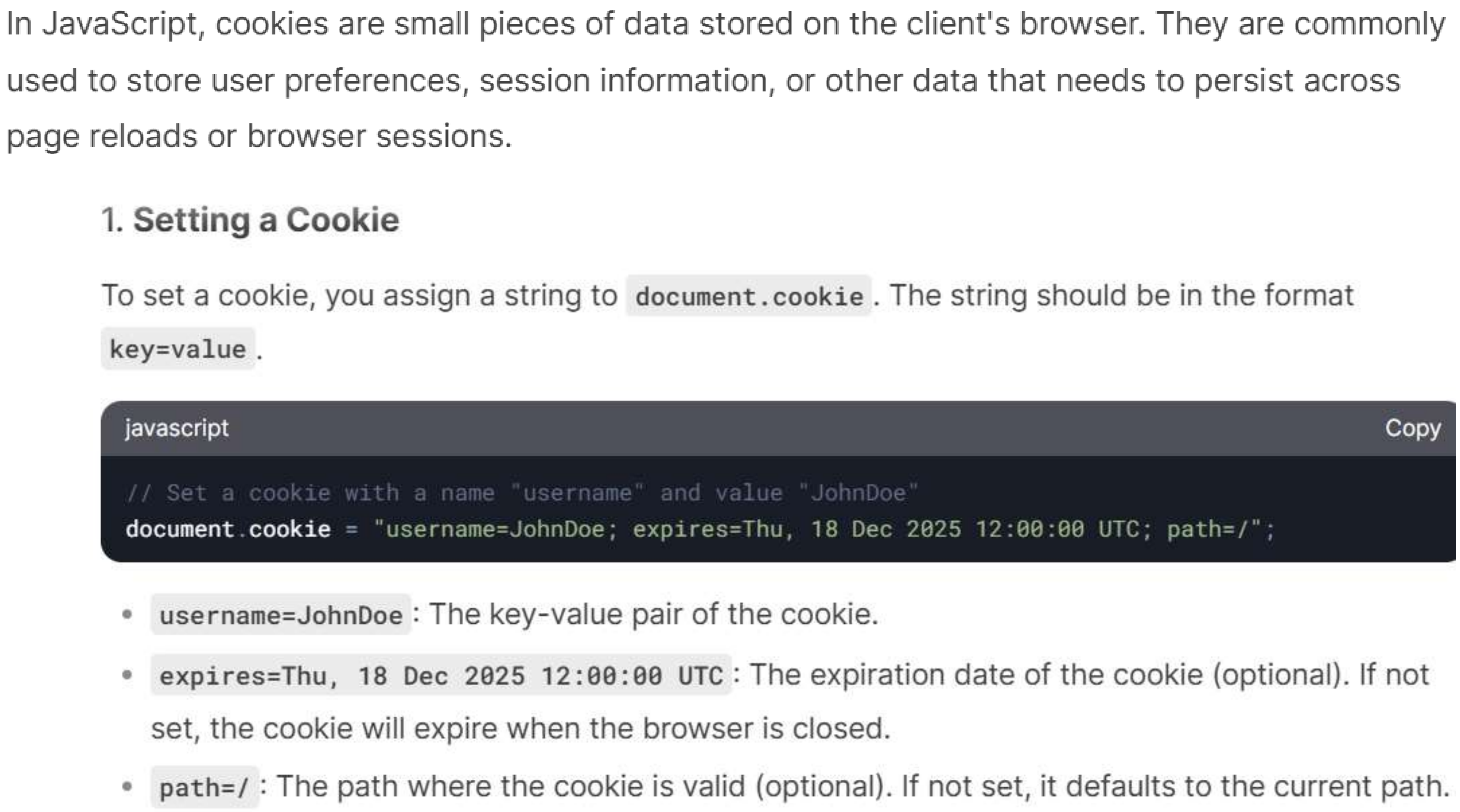
Hello World!

The DOM is very useful.

This example demonstrates the **getElementsByClassName** method.

The first paragraph (index 0) with class="intro": The DOM is very useful.

Cookies in JavaScript



In JavaScript, cookies are small pieces of data stored on the client's browser. They are commonly used to store user preferences, session information, or other data that needs to persist across page reloads or browser sessions.

1. Setting a Cookie

To set a cookie, you assign a string to `document.cookie`. The string should be in the format `key=value`.

javascript

```javascript
// Set a cookie with a name "username" and value "JohnDoe"
document.cookie = "username=JohnDoe; expires=Thu, 18 Dec 2025 12:00:00 UTC; path=/";
```

- `username=JohnDoe`: The key-value pair of the cookie.
- `expires=Thu, 18 Dec 2025 12:00:00 UTC`: The expiration date of the cookie (optional). If not set, the cookie will expire when the browser is closed.
- `path=/`: The path where the cookie is valid (optional). If not set, it defaults to the current path.

2. **Reading a Cookie**

To read a cookie, you access `document.cookie`, which returns all cookies as a string. You can then parse this string to get the value of a specific cookie.

3. **Deleting a Cookie**

To delete a cookie, you set its expiration date to a past date.

```javascript
// Delete the "username" cookie
document.cookie = "username=; expires=Thu, 01 Jan 1970 00:00:00 UTC; path=/;";
```

- Setting `expires` to a past date removes the cookie.

Notes:

- Cookies are limited in size (typically 4KB per cookie).
- They are sent with every HTTP request to the server, which can impact performance if overused.
- For more secure and modern storage options, consider using `localStorage` or `sessionStorage`.

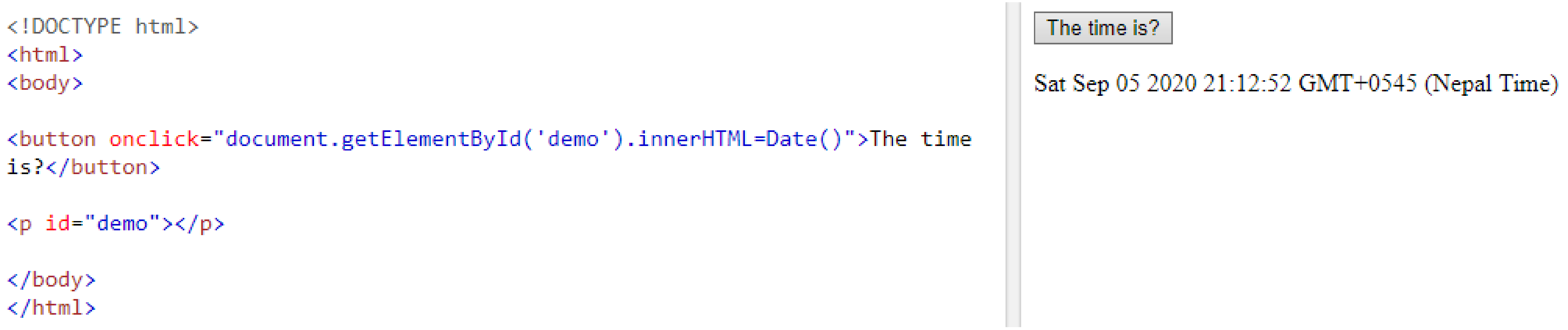
```
// Set a cookie
document.cookie = "username=JohnDoe; expires=Thu, 18 Dec 2025 12:00:00 UTC; path=/";

// Read the cookie
function getCookie(name) {
    const cookieString = document.cookie;
    const cookies = cookieString.split('; ');
    for (let cookie of cookies) {
        const [cookieName, cookieValue] = cookie.split('=');
        if (cookieName === name) {
            return cookieValue;
        }
    }
    return null;
}

const username = getCookie("username");
console.log(username); // Output: JohnDoe

// Delete the cookie
document.cookie = "username=; expires=Thu, 01 Jan 1970 00:00:00 UTC; path=/;";
console.log(getCookie("username")); // Output: null (cookie is deleted)
```

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>
<input type="button" value="setCookie" onclick="set
Cookie()">
<input type="button" value="getCookie" onclick="ge
tCookie()">
    <script>
    function setCookie()
    {
        document.cookie="username=Duke Martin";
    }
    function getCookie()
    {
        if(document.cookie.length!=0)
        {
        alert(document.cookie);
        }
        else
        {
        alert("Cookie not available");
        }
    }
    </script>

</body>
</html>
```

Cookie Example

JavaScript 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

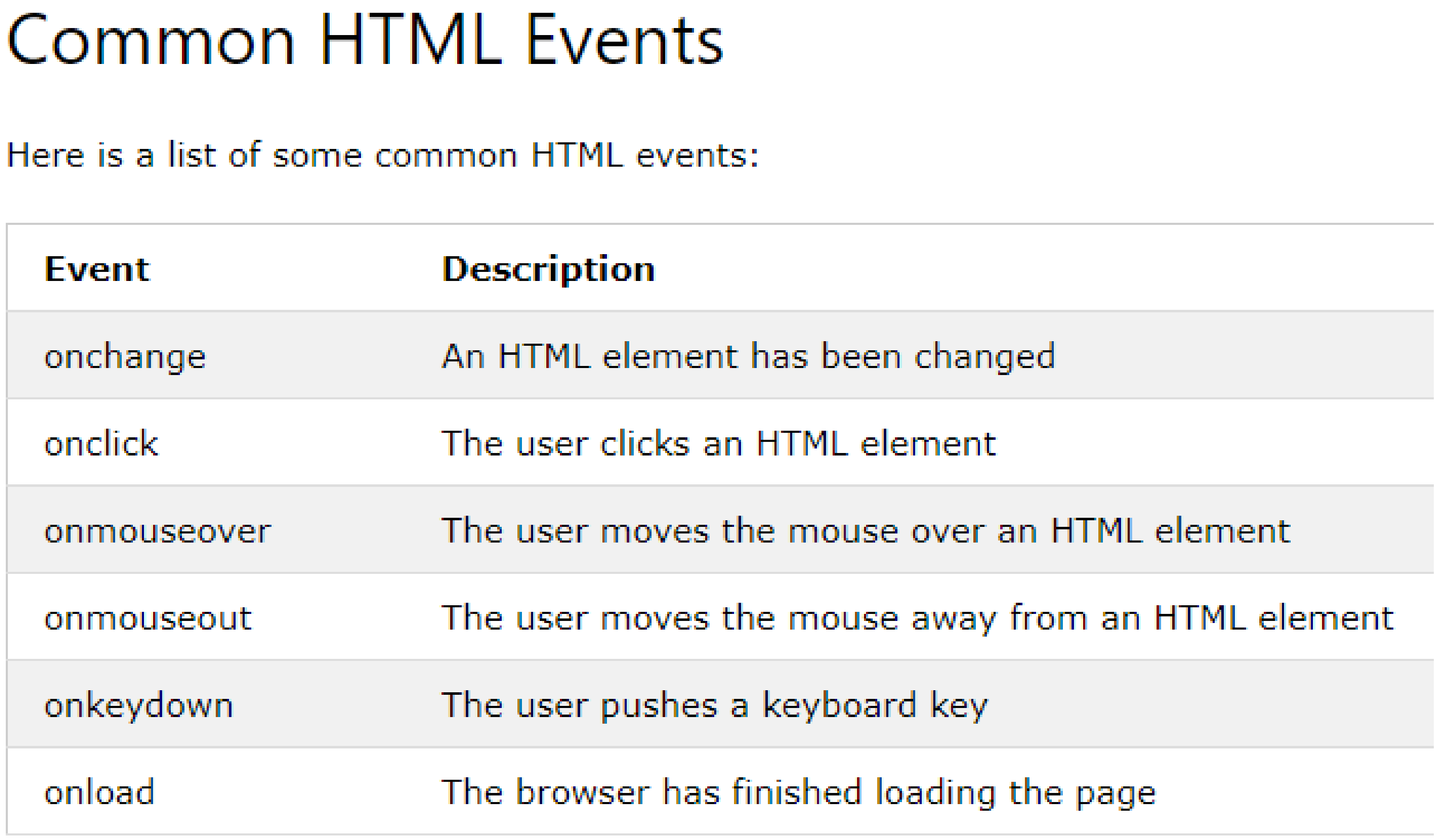


```
<!DOCTYPE html>
<html>
<body>

<button onclick="document.getElementById('demo').innerHTML=Date()">The time
is?</button>

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

</body>
</html>
```

```
Sat Sep 05 2020 21:12:52 GMT+0545 (Nepal Time)
```

```
<!DOCTYPE html>
<html>
<body>
<h1 onclick="this.innerHTML='Ooops!'">Click on this text!</h1>
</body>
</html>
```

Click on this text! **Ooops!**



Common HTML Events

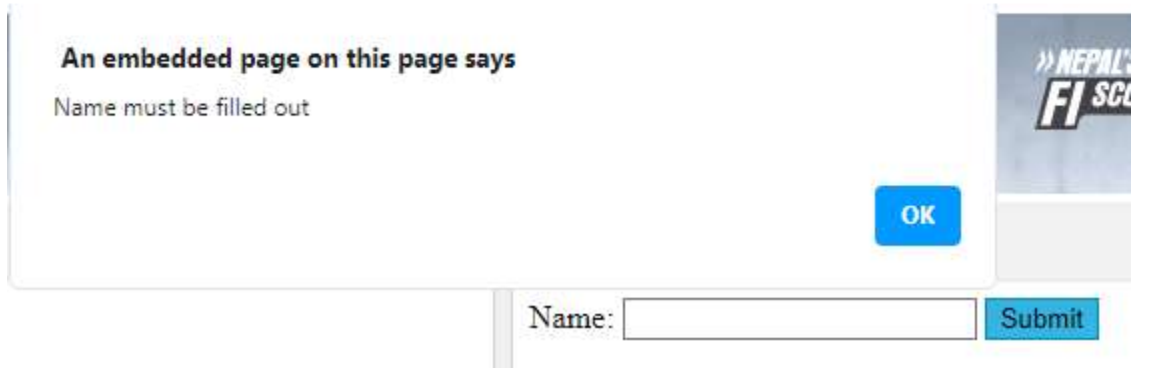
Here is a list of some common HTML events:

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

JavaScript Form Validation

JavaScript Example

```
function validateForm() {
  var x = document.forms["myForm"]["fname"].value;
  if (x == "") {
    alert("Name must be filled out");
    return false;
  }
}
```

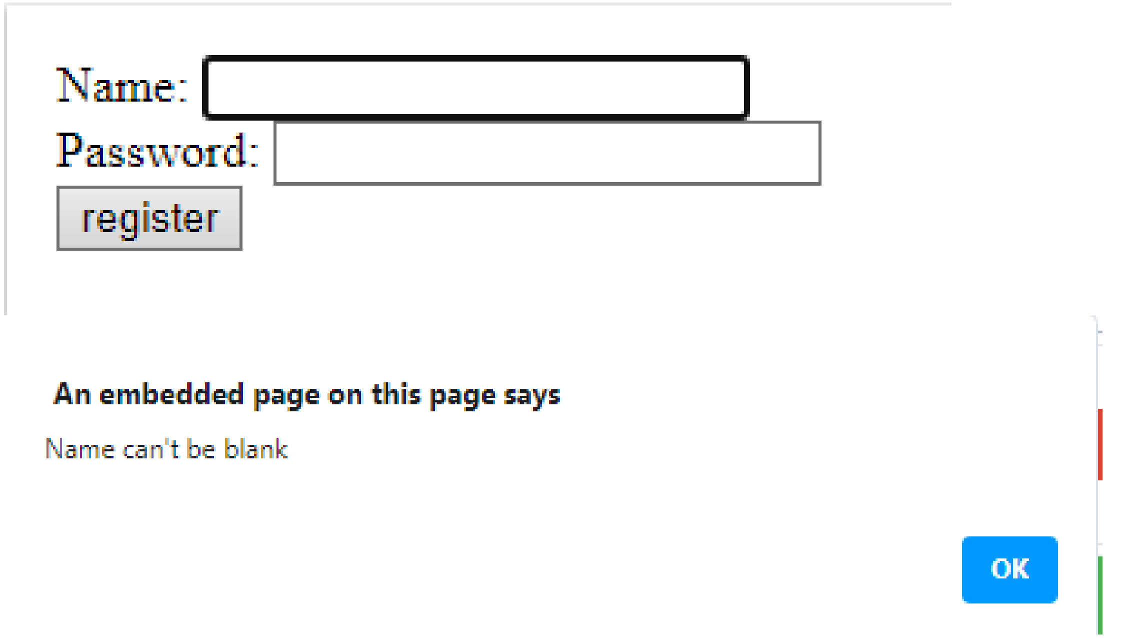

An embedded page on this page says

Name must be filled out

HTML Form Example

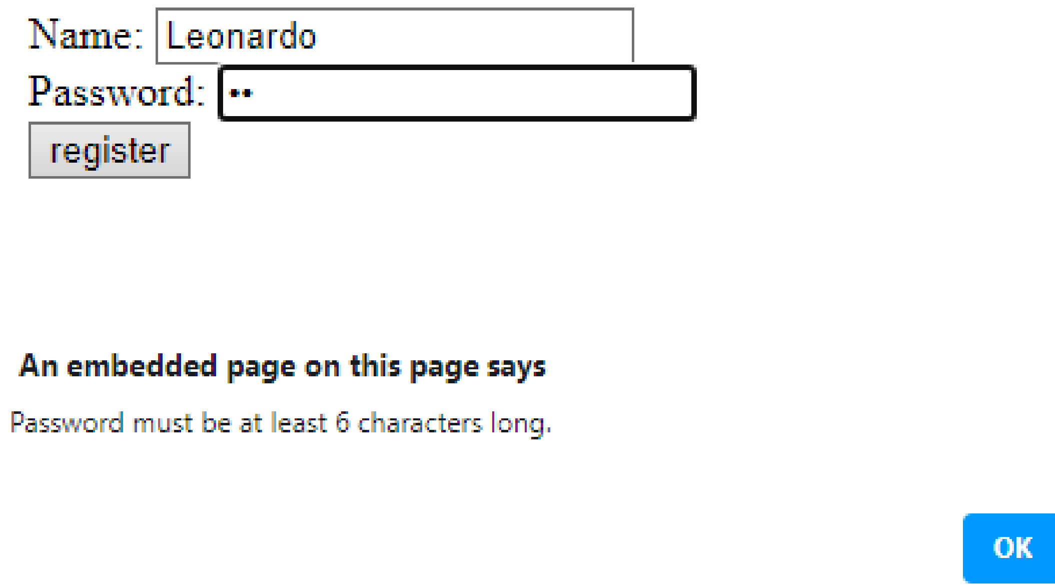
```
<form name="myForm" action="/action_page.php" onsubmit="return validateForm()" method="post">
Name: <input type="text" name="fname">
<input type="submit" value="Submit">
</form>
```

```
<html>
<body>
<script>
function validateform(){
var name=document.myform.name.value;
var password=document.myform.password.value;

if (name==null || name==""){
  alert("Name can't be blank");
  return false;
}else if(password.length<6){
  alert("Password must be at least 6 characters long.");
  return false;
  }
}
</script>
<body>
<form name="myform" method="post"
action="http://www.javatpoint.com/javascriptpages/valid.jsp"
onsubmit="return validateform()" >
Name: <input type="text" name="name"><br/>
Password: <input type="password" name="password"><br/>
<input type="submit" value="register">
</form>
</body>
</html>
```



An embedded page on this page says

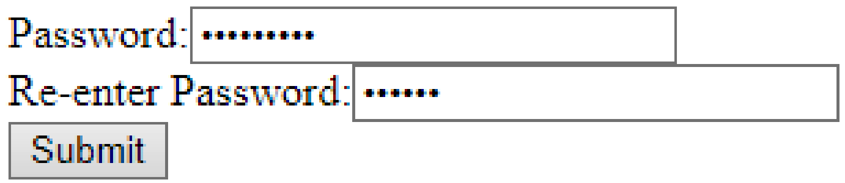
Name can't be blank

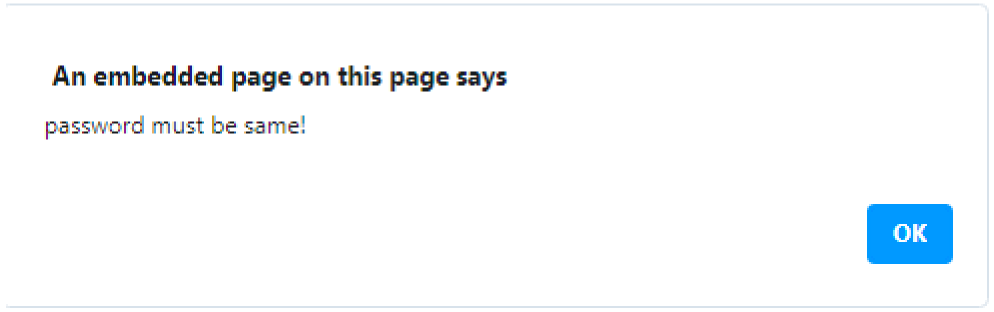


An embedded page on this page says

Password must be at least 6 characters long.

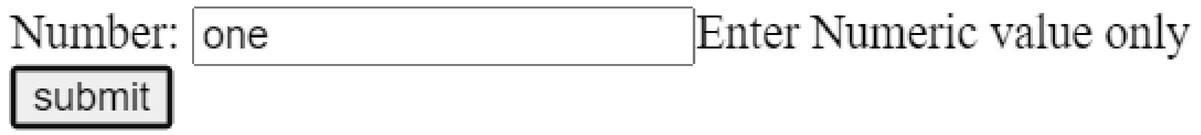
```
<html>
<head>
<script type="text/javascript">
function matchpass(){
var firstpassword=document.f1.password.value;
var secondpassword=document.f1.password2.value;

if(firstpassword==secondpassword){
return true;
}
else{
alert("password must be same!");
return false;
}
}
</script>
</head>
<body>

<form name="f1"
action="http://www.javatpoint.com/javascriptpages/valid.jsp"
onsubmit="return matchpass()">
Password:<input type="password" name="password" /><br/>
Re-enter Password:<input type="password" name="password2"/><br/>
<input type="submit">
</form>
```


Password: •••••••••
Re-enter Password: ••••••


An embedded page on this page says
password must be same!

JavaScript Number Validation

```
<!DOCTYPE html>
<html>
<head>
<script>
function validate(){
var num=document.myform.num.value;
if (isNaN(num)){

document.getElementById("numloc").innerHTML=
"Enter Numeric value only";
  return false;
}else{
  return true;
  }
}
</script>
</head>
```



Number: one Enter Numeric value only

```
<body>
<form name="myform"
action="http://www.javatpoint.com/javascriptpages/valid.jsp"
onsubmit="return validate()" >
Number: <input type="text" name="num"><span
id="numloc"></span><br/>
<input type="submit" value="submit">
</form>

</body>
</html>
```

JavaScript email validation

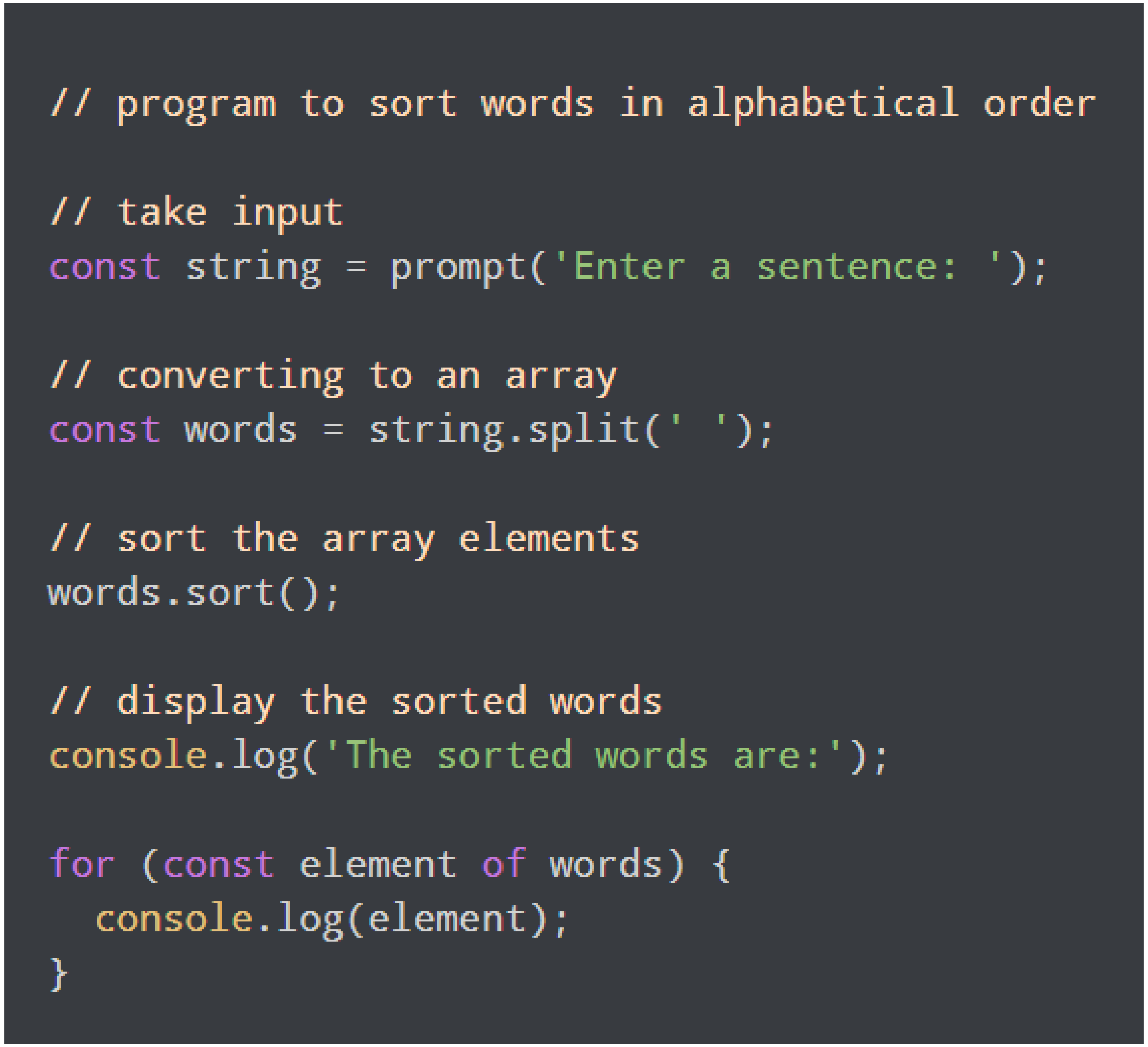
```
<html>
<body>
<script>
function validateemail()
{
var x=document.myform.email.value;
var atposition=x.indexOf("@");
var dotposition=x.lastIndexOf(".");
if (atposition<1 || dotposition<atposition+2 ||
dotposition+2>=x.length){
  alert("Please enter a valid e-mail address”);
  return false;
  }
}
</script>
<body>
<form name="myform"  method="post"
action="http://www.javatpoint.com/javascriptpages/valid.jsp"
onsubmit="return validateemail();">
Email: <input type="text" name="email"><br/>

<input type="submit" value="register">
</form>
</body>
</html>
```

•email id must contain the @ and . character
•There must be at least one character before and after the @.
•There must be at least two characters after . (dot).

Lab exercises

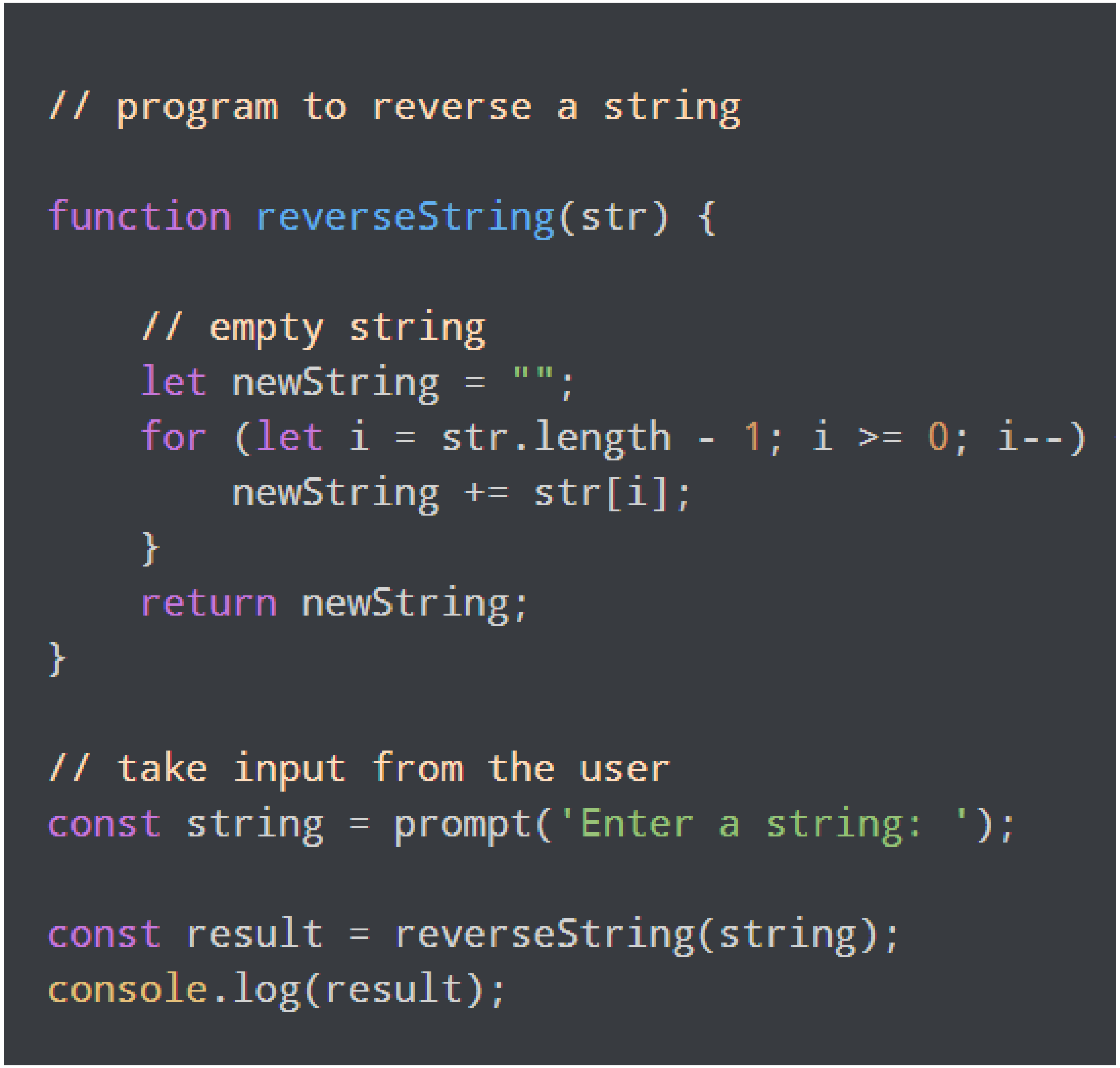
22. Write a JavaScript Program to sort words in alphabetical order.
23. Write a Program to reverse a String
24. Write a Program to display Date and Time.
25. Write a Program to Insert Item in an Array
26. Write a Program to Remove Duplicates from an Array.
27. Write a Program to validate a form having Name, Phone Number, Email and Address.

Sort string in alphabetical order



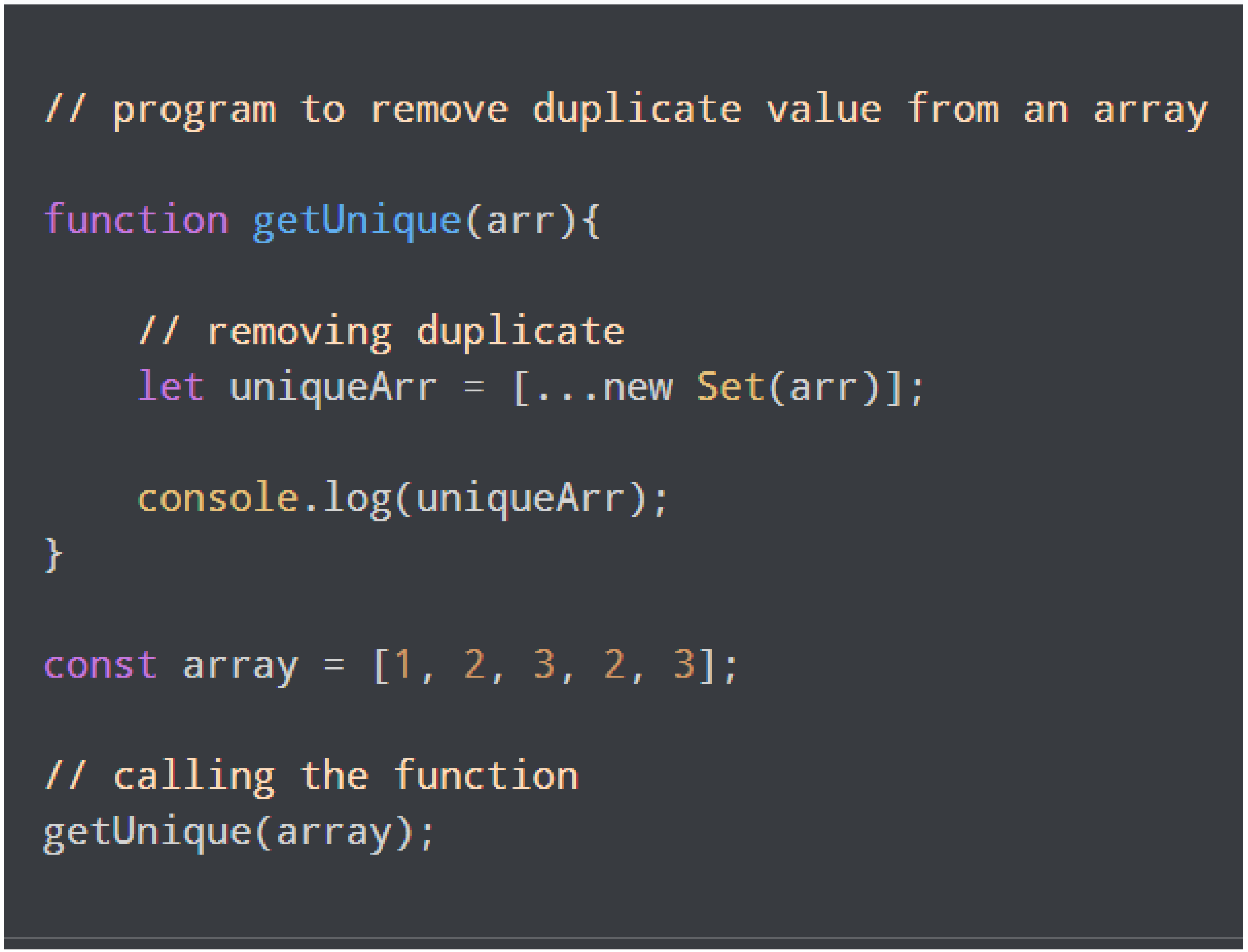
```
// program to sort words in alphabetical order

// take input
const string = prompt('Enter a sentence: ');

// converting to an array
const words = string.split(' ');

// sort the array elements
words.sort();

// display the sorted words
console.log('The sorted words are:');

for (const element of words) {
  console.log(element);
}
```

Reverse a String



```javascript
// program to reverse a string

function reverseString(str) {

    // empty string
    let newString = "";
    for (let i = str.length - 1; i >= 0; i--)
        newString += str[i];
    }
    return newString;
}

// take input from the user
const string = prompt('Enter a string: ');

const result = reverseString(string);
console.log(result);
```

Remove Duplicates from an Array



```
// program to remove duplicate value from an array

function getUnique(arr){

    // removing duplicate
    let uniqueArr = [...new Set(arr)];

    console.log(uniqueArr);
}

const array = [1, 2, 3, 2, 3];

// calling the function
getUnique(array);
```

What is AJAX?

AJAX stands for **Asynchronous JavaScript and XML**. It is a technique used in web development to create asynchronous web applications. With AJAX, web applications can send and retrieve data from a server asynchronously (in the background) without interfering with the display and behavior of the existing page. This allows for dynamic updates to web pages without requiring a full page reload.

AJAX is not a programming language but a combination of:

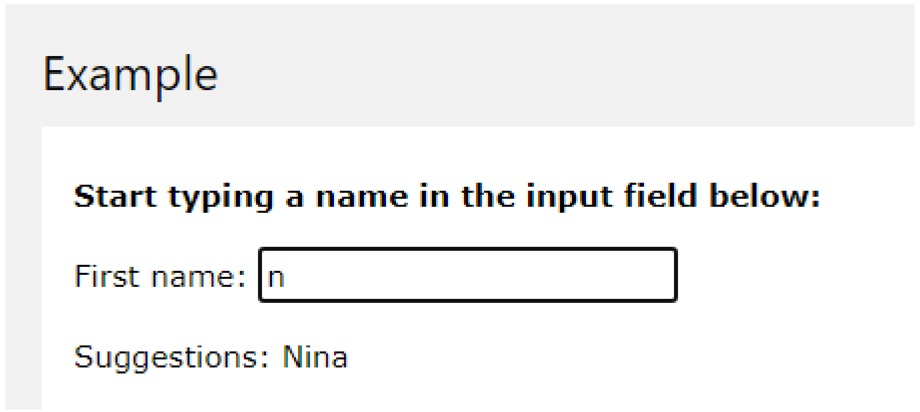
- **JavaScript**: Handles the logic and interaction.
- **XMLHttpRequest (XHR)**: Used to communicate with the server.
- **HTML/CSS**: For displaying and styling the content.
- **JSON/XML**: Common data formats for transferring data between the server and client.

Key Points:

- AJAX allows for seamless updates to a webpage without reloading.
- Modern web development often uses the `fetch` API or libraries like Axios instead of `XMLHttpRequest`.
- AJAX is widely used in single-page applications (SPAs) like Gmail, Facebook, and Twitter.

Applications of AJAX

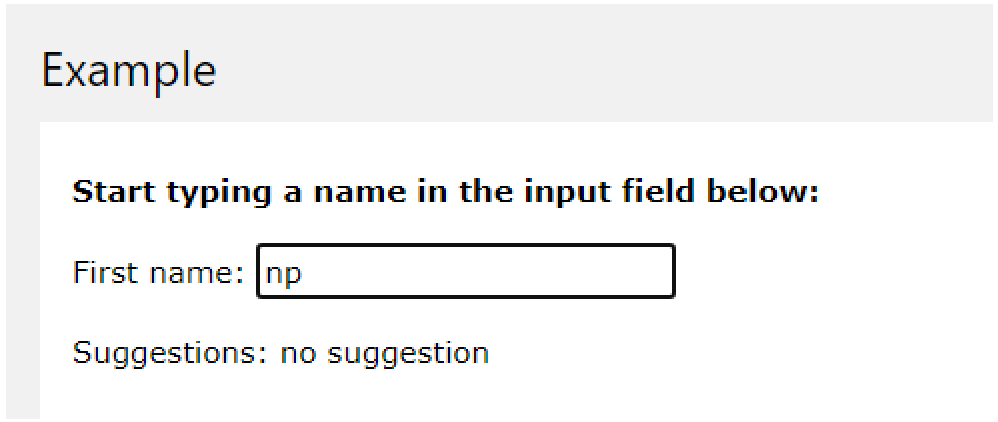
1. **Dynamic Content Loading**: Load content without refreshing the page (e.g., infinite scroll).
2. **Form Submission**: Submit forms without reloading the page.
3. **Auto-Suggestions**: Provide search suggestions as the user types.
4. **Real-Time Updates**: Update content like notifications, chat messages, or stock prices in real-time.
5. **Interactive Web Applications**: Create responsive and interactive user interfaces.


Example

Start typing a name in the input field below:

First name: n

Suggestions: Nina

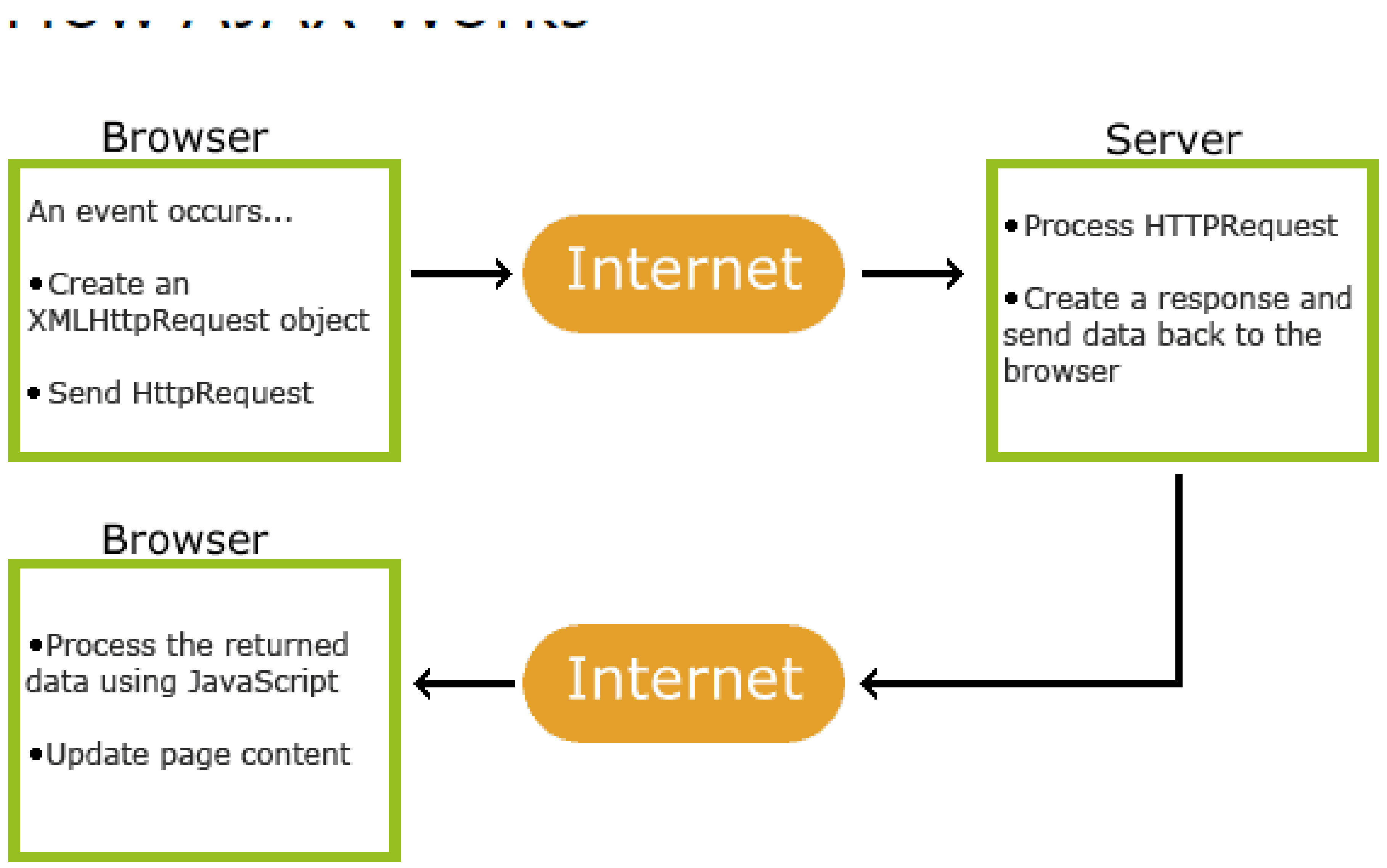

Example

Start typing a name in the input field below:

First name: np

Suggestions: no suggestion

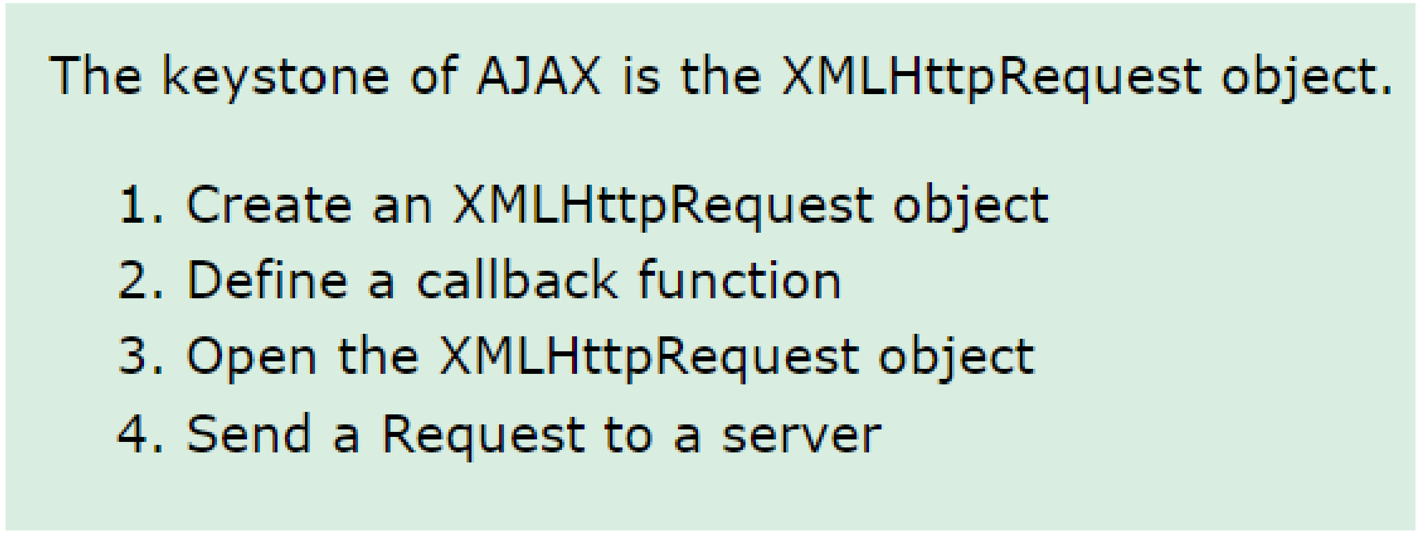
How AJAX Works


Browser
An event occurs...
• Create an XMLHttpRequest object
• Send HttpRequest
Internet
Server
• Process HTTPRequest
• Create a response and send data back to the browser
Browser
• Process the returned data using JavaScript
• Update page content
Internet

The XMLHttpRequest Object

All modern browsers support the `XMLHttpRequest` object.

The `XMLHttpRequest` object can be used to exchange data with a web server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.

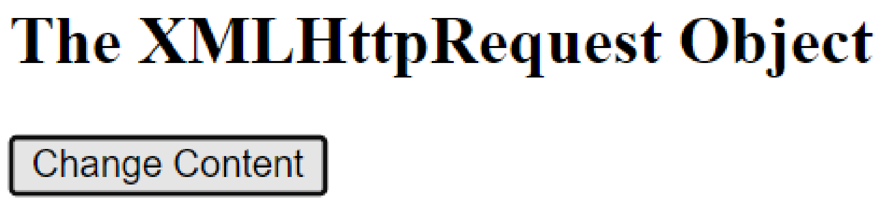


The keystone of AJAX is the XMLHttpRequest object.

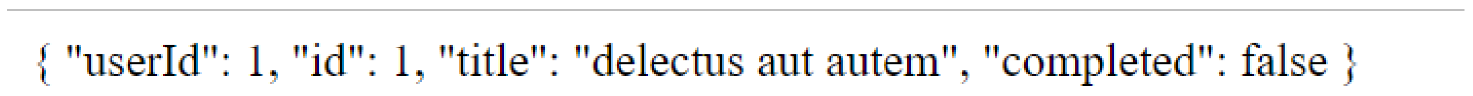
1. Create an XMLHttpRequest object
2. Define a callback function
3. Open the XMLHttpRequest object
4. Send a Request to a server

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

<div id="demo">
<h2>The XMLHttpRequest
Object</h2>
<button type="button"
onclick="loadDoc()">Change
Content</button>
</div>
```

```html
<script>
function loadDoc() {
  const xhttp = new XMLHttpRequest();
  xhttp.onload = function() {
    document.getElementById("demo").innerHTML =
    this.responseText;
  }
  var url = "https://jsonplaceholder.typicode.com/todos/1";
  xhttp.open("GET", url, true);
  xhttp.send();
}
</script>

</body>
</html>
```



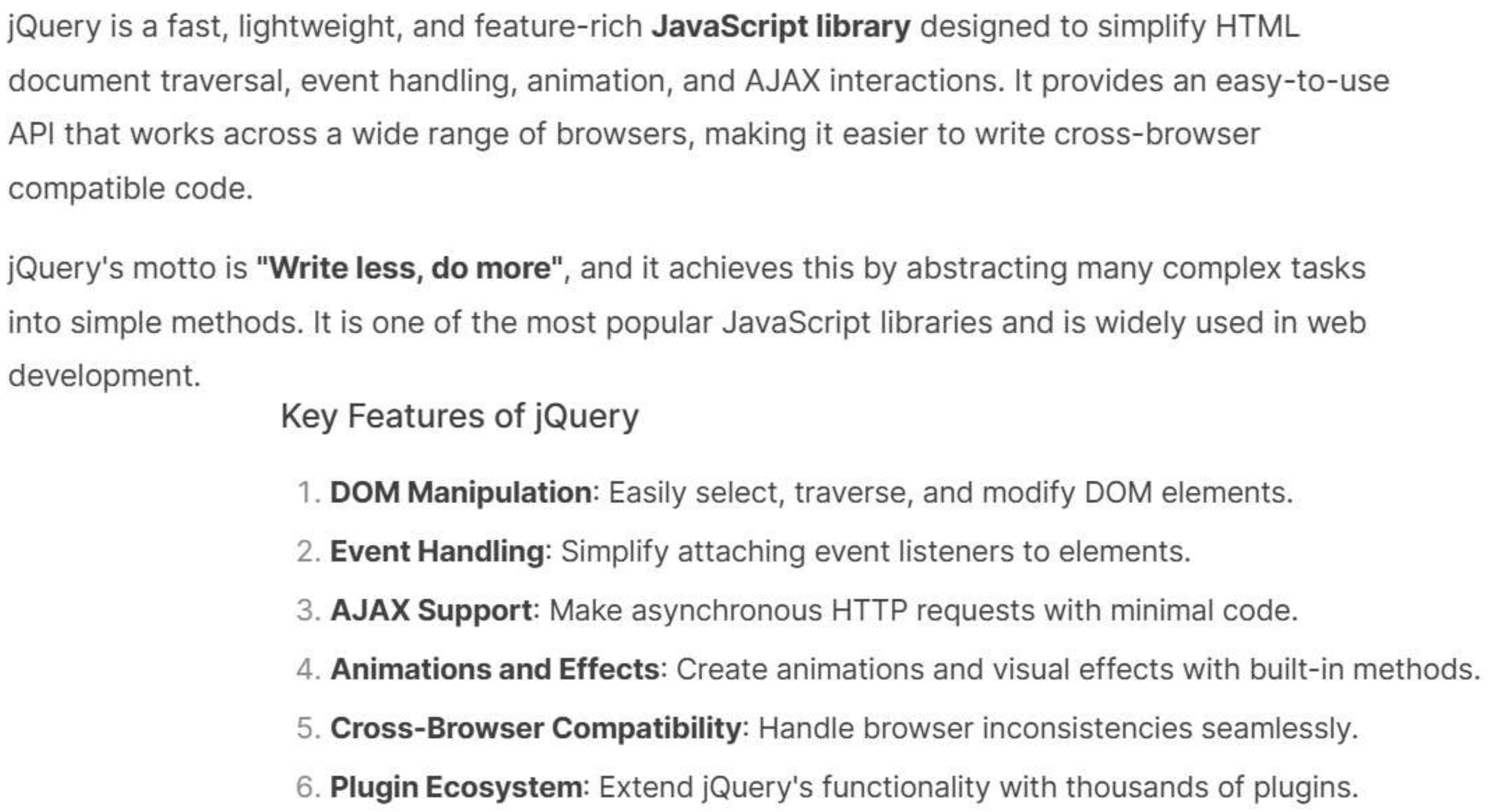
The XMLHttpRequest Object



```
{ "userId": 1, "id": 1, "title": "delectus aut autem", "completed": false }
```

JQuery

What is jQuery?



jQuery is a fast, lightweight, and feature-rich **JavaScript library** designed to simplify HTML document traversal, event handling, animation, and AJAX interactions. It provides an easy-to-use API that works across a wide range of browsers, making it easier to write cross-browser compatible code.

jQuery's motto is **"Write less, do more"**, and it achieves this by abstracting many complex tasks into simple methods. It is one of the most popular JavaScript libraries and is widely used in web development.

Key Features of jQuery

1. **DOM Manipulation**: Easily select, traverse, and modify DOM elements.
2. **Event Handling**: Simplify attaching event listeners to elements.
3. **AJAX Support**: Make asynchronous HTTP requests with minimal code.
4. **Animations and Effects**: Create animations and visual effects with built-in methods.
5. **Cross-Browser Compatibility**: Handle browser inconsistencies seamlessly.
6. **Plugin Ecosystem**: Extend jQuery's functionality with thousands of plugins.

Why jQuery?

There are lots of other JavaScript libraries out there, but jQuery is probably the most popular, and also the most extendable.

Many of the biggest companies on the Web use jQuery, such as:

- Google
- Microsoft
- IBM
- Netflix

Adding jQuery to Your Web Pages

There are several ways to start using jQuery on your web site. You can:

- Download the jQuery library from jQuery.com
- Include jQuery from a CDN, like Google

Downloading jQuery

The jQuery library is a single JavaScript file, and you reference it with the HTML `<script>` tag (notice that the `<script>` tag should be inside the `<head>` section):

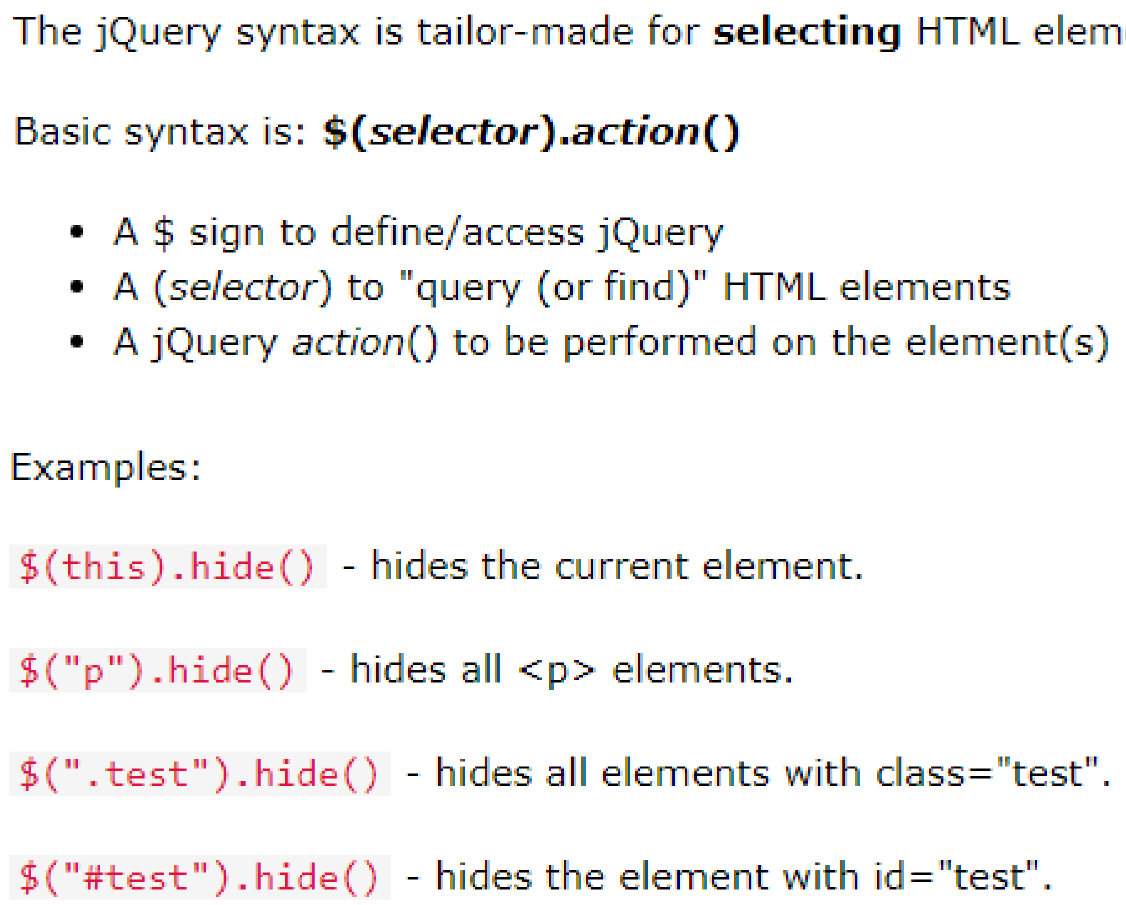
```
<head>
<script src="jquery-3.5.1.min.js"></script>
</head>
```

jQuery CDN

Google CDN:

```
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
</head>
```

jQuery Syntax



The jQuery syntax is tailor-made for **selecting** HTML elements and performing some **action** on the element(s).

Basic syntax is: **$(*selector*).*action*()**

- A $ sign to define/access jQuery
- A (*selector*) to "query (or find)" HTML elements
- A jQuery *action*() to be performed on the element(s)

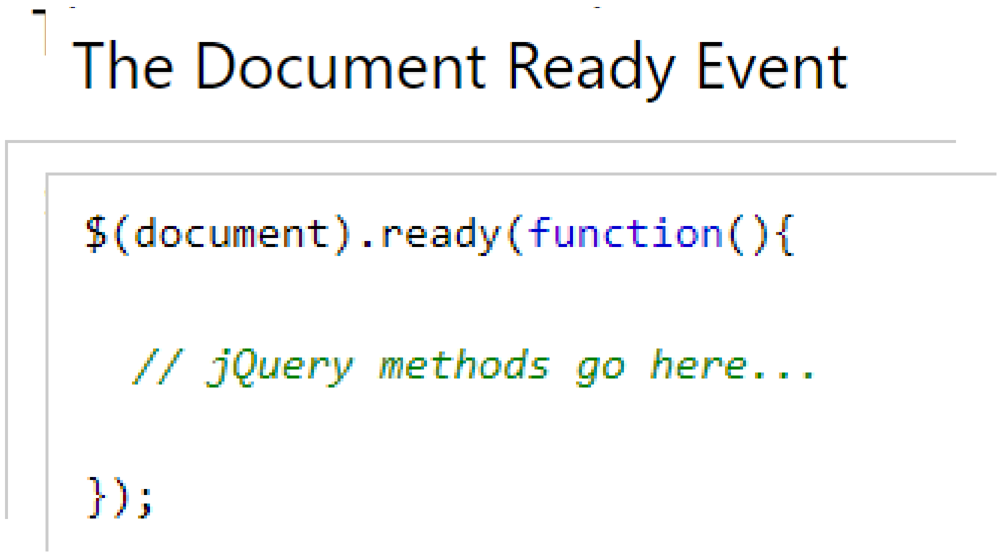
Examples:

`$(this).hide()` - hides the current element.

`$("p").hide()` - hides all <p> elements.

`$(".test").hide()` - hides all elements with class="test".

`$("#test").hide()` - hides the element with id="test".



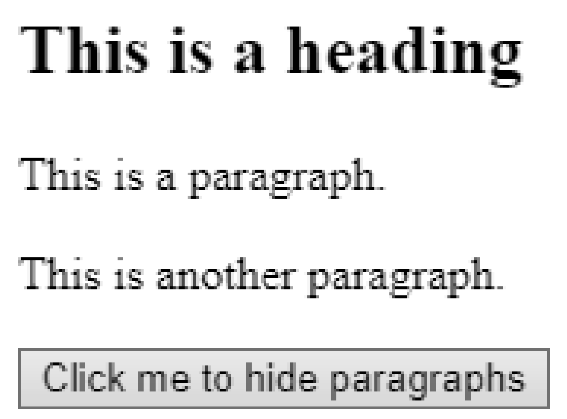
The Document Ready Event

```
$(document).ready(function(){

  // jQuery methods go here...

});
```

This is to prevent any jQuery code from running before the document is finished loading (is ready).

```html
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js">
</script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("p").hide();
  });
});
</script>
</head>
<body>

<h2>This is a heading</h2>

<p>This is a paragraph.</p>
<p>This is another paragraph.</p>

<button>Click me to hide paragraphs</button>

</body>
</html>
```

Syntax	Description
$("*")	Selects all elements
$(this)	Selects the current HTML element
$("p.intro")	Selects all <p> elements with class="intro"
$("p:first")	Selects the first <p> element
$("ul li:first")	Selects the first <li> element of the first <ul>
$("ul li:first-child")	Selects the first <li> element of every <ul>
$("[href]")	Selects all elements with an href attribute



This is a heading

This is a paragraph.

This is another paragraph.

Click me to hide paragraphs

Applications of jQuery

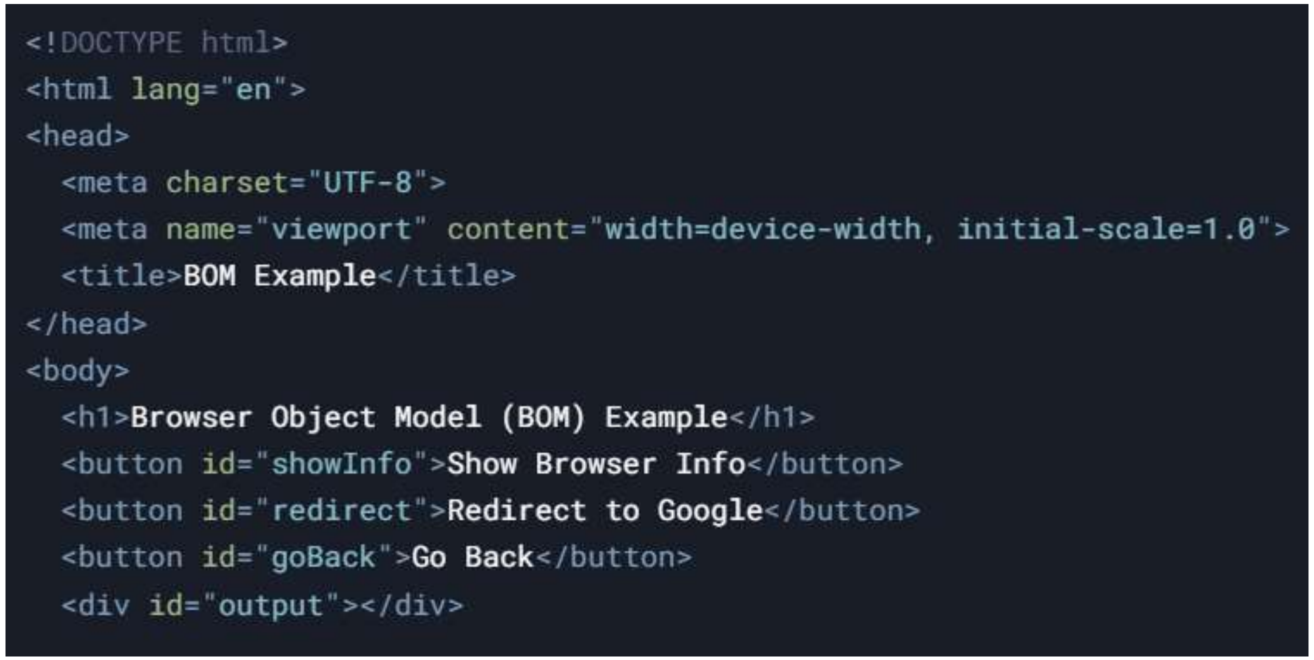
1. **DOM Manipulation**:
   - Easily select and modify HTML elements.
   - Example: Change the content or style of an element.
2. **Event Handling**:
   - Attach event listeners like `click`, `hover`, `submit`, etc.
   - Example: Show a message when a button is clicked.
3. **AJAX Requests**:
   - Simplify making asynchronous requests to the server.
   - Example: Load data from a server without refreshing the page.
4. **Animations and Effects**:
   - Create animations like fading, sliding, and toggling.
   - Example: Fade out an element when a button is clicked.
5. **Form Validation**:
   - Validate user input before submitting a form.
   - Example: Ensure an email field contains a valid email address.
6. **Dynamic Content Loading**:
   - Load content dynamically based on user interaction.
   - Example: Load more posts when the user scrolls to the bottom of the page.
7. **Creating Interactive UI Components**:
   - Build interactive components like accordions, sliders, and modals.
   - Example: Create a dropdown menu that expands on hover.

What is BOM in javascript? Give an example

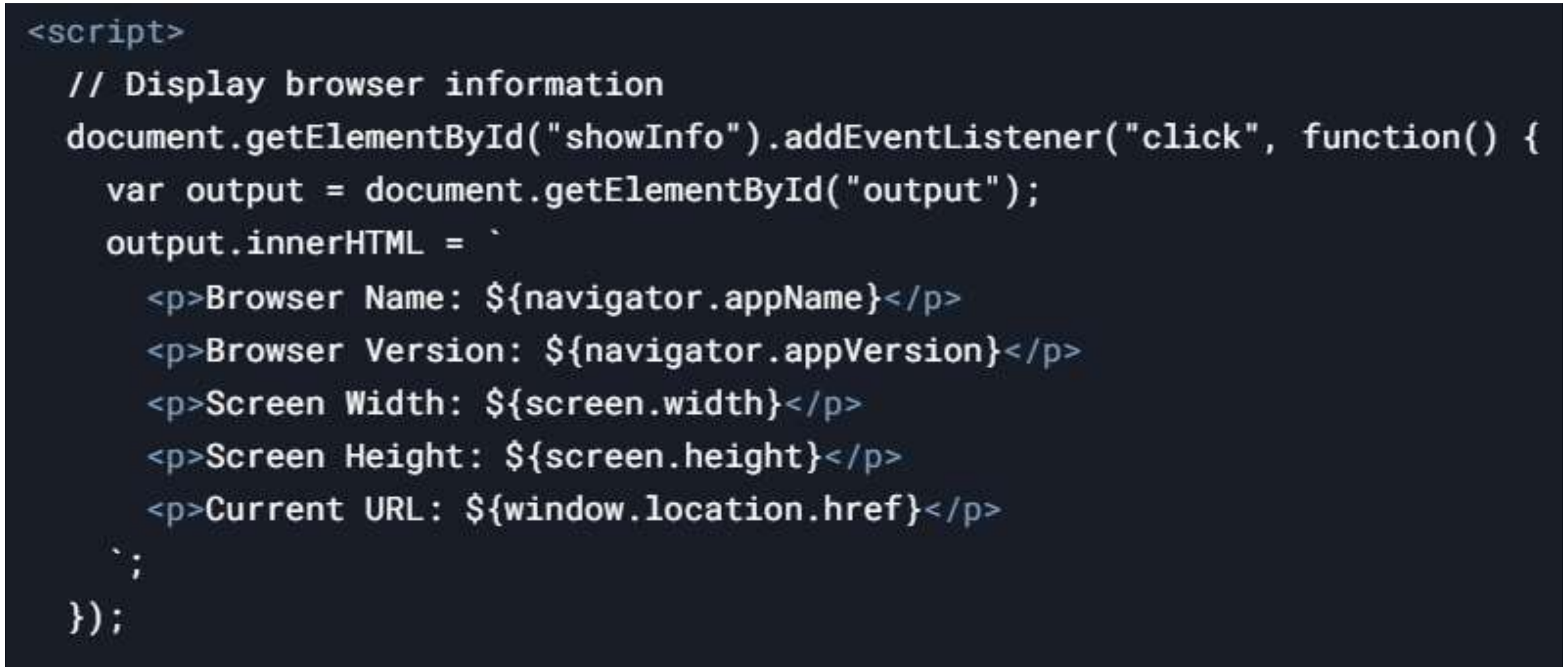
BOM stands for **Browser Object Model**. It is a set of objects provided by the browser that allows JavaScript to interact with the browser itself. Unlike the **Document Object Model (DOM)**, which deals with the structure and content of the document (HTML), the BOM deals with the browser window and its components.

The BOM includes objects such as:

- `window` : The global object representing the browser window.
- `navigator` : Provides information about the browser and its capabilities.
- `screen` : Provides information about the user's screen.
- `location` : Provides information about the current URL and allows navigation to new URLs.
- `history` : Allows interaction with the browser's session history (back/forward navigation).
- `document` : Part of both BOM and DOM, represents the loaded webpage.



```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>BOM Example</title>
</head>
<body>
  <h1>Browser Object Model (BOM) Example</h1>
  <button id="showInfo">Show Browser Info</button>
  <button id="redirect">Redirect to Google</button>
  <button id="goBack">Go Back</button>
  <div id="output"></div>
```



```html
<script>
  // Display browser information
  document.getElementById("showInfo").addEventListener("click", function() {
    var output = document.getElementById("output");
    output.innerHTML = `
      <p>Browser Name: ${navigator.appName}</p>
      <p>Browser Version: ${navigator.appVersion}</p>
      <p>Screen Width: ${screen.width}</p>
      <p>Screen Height: ${screen.height}</p>
      <p>Current URL: ${window.location.href}</p>
    `;
  });
```

```
    // Redirect to Google
    document.getElementById("redirect").addEventListener("click", function() {
      window.location.href = "https://www.google.com";
    });

    // Go back in history
    document.getElementById("goBack").addEventListener("click", function() {
      window.history.back();
    });
  </script>
</body>
</html>
```

Sample Output:

1. **Before Clicking "Show Browser Info"**:

```
Browser Object Model (BOM) Example
[Show Browser Info] [Redirect to Google] [Go Back]
```

2. **After Clicking "Show Browser Info"**:

```
Browser Object Model (BOM) Example
[Show Browser Info] [Redirect to Google] [Go Back]

Browser Name: Netscape
Browser Version: 5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
rome/91.0.4472.124 Safari/537.36
Screen Width: 1920
Screen Height: 1080
Current URL: http://example.com/bom-example.html
```

3. **After Clicking "Redirect to Google"**:
   - The browser navigates to `https://www.google.com`.
4. **After Clicking "Go Back"**:
   - The browser navigates back to the previous page in the history.

Key Points:

- The BOM is not standardized, so its behavior may vary slightly across browsers.
- The `window` object is the root of the BOM and is also the global object in JavaScript.
- BOM is essential for tasks like handling browser windows, navigating history, and accessing browser information.

REACT / React.js

React (also known as **React.js** or **ReactJS**) is an open-source JavaScript library for building user interfaces (UIs) or UI components. It is maintained by Facebook (now Meta) and a community of individual developers and companies. React is primarily used for building single-page applications (SPAs) where the content is dynamically rendered without requiring a page reload.

React follows a component-based architecture, allowing developers to create reusable UI components. It uses a virtual DOM (Document Object Model) to optimize rendering performance, making it faster and more efficient compared to directly manipulating the real DOM.

Applications of React

1. **Single-Page Applications (SPAs):** React is widely used for building SPAs where the content is dynamically updated without refreshing the page.
2. **Mobile Applications:** React Native, a framework based on React, is used to build cross-platform mobile apps.
3. **Dynamic Web Applications:** React is ideal for applications that require frequent data updates, such as social media platforms, e-commerce sites, and dashboards.
4. **Reusable UI Components:** React's component-based structure allows developers to create reusable UI elements, improving code maintainability.
5. **Progressive Web Apps (PWAs):** React can be used to build PWAs that offer a native app-like experience on the web.

```
// Import React and ReactDOM
import React, { useState } from 'react';
import ReactDOM from 'react-dom';

// Define a functional component
function App() {
  // Use state to manage the message
  const [message, setMessage] = useState('Hello, World!');

  // Function to update the message
  const updateMessage = () => {
    setMessage('Hello, React!');
  };

  return (
    <div>
      <h1>{message}</h1>
      <button onClick={updateMessage}>Click Me</button>
    </div>
  );
}

// Render the App component into the root element
ReactDOM.render(<App />, document.getElementById('root'));
```

Sample Output

1. **Initial Output:**

   ```
   Hello, World!
   [Click Me Button]
   ```

2. **After Clicking the Button:**

   ```
   Hello, React!
   [Click Me Button]
   ```

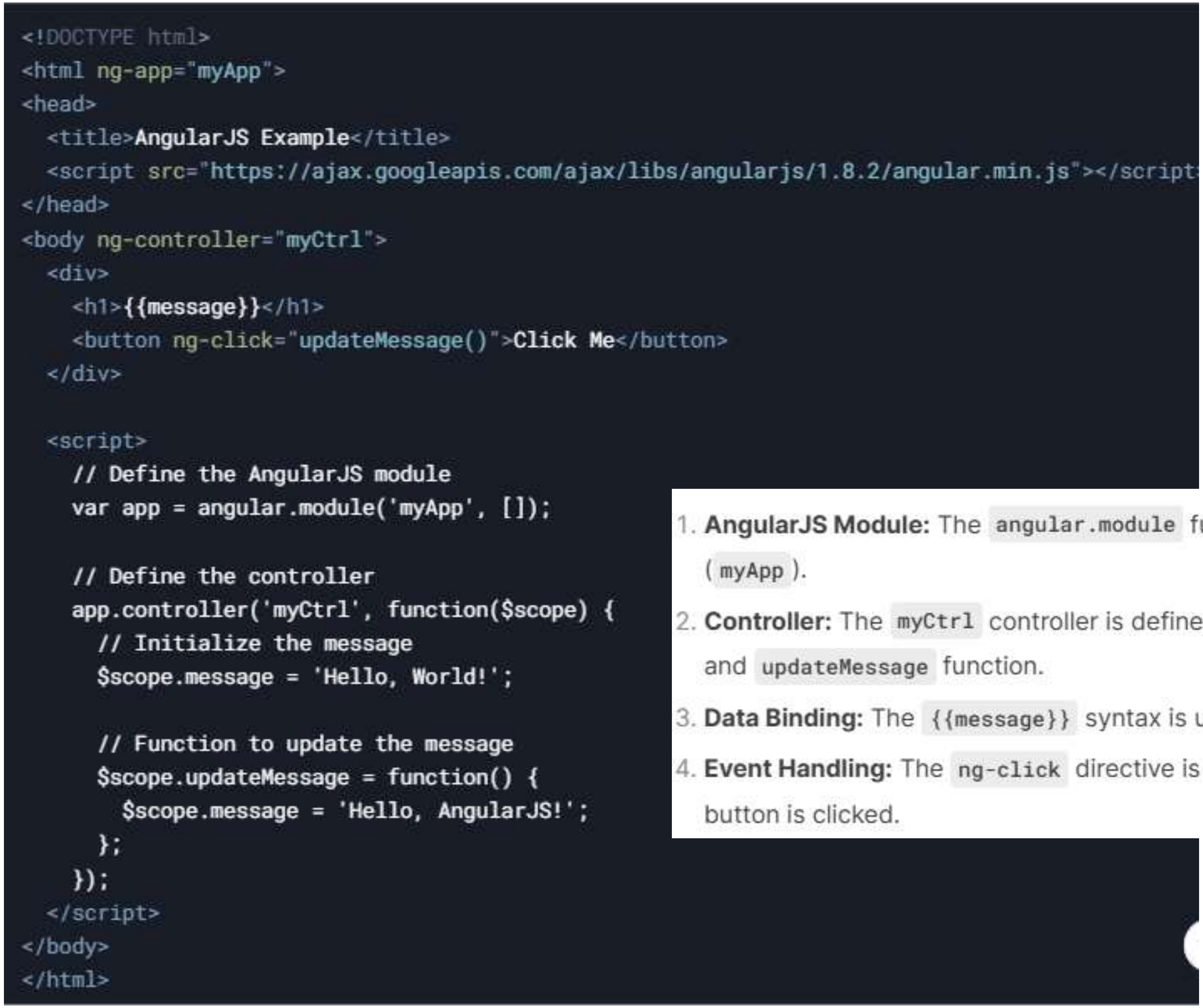
Angular JS

AngularJS is an open-source JavaScript framework developed by Google for building dynamic web applications. It is based on the **Model-View-Controller (MVC)** architecture and is designed to simplify both the development and testing of web applications. AngularJS extends HTML with additional attributes and binds data to HTML with expressions, making it a powerful tool for creating rich, interactive web applications.

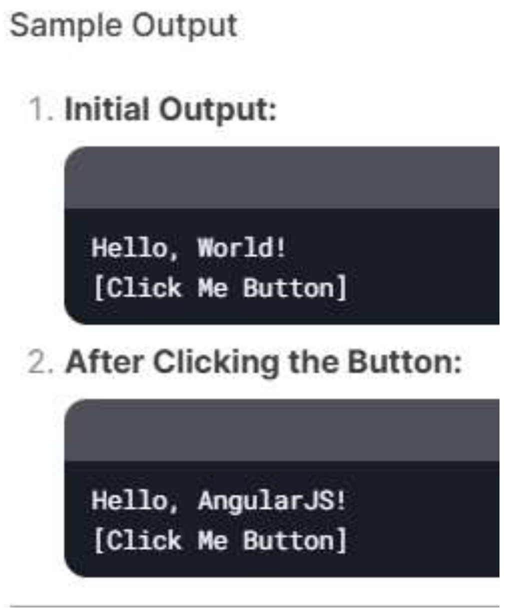
AngularJS is often referred to as **Angular 1.x** to distinguish it from the newer versions of Angular (Angular 2+), which are complete rewrites of the framework.

Applications of AngularJS

1. **Single-Page Applications (SPAs):** AngularJS is widely used for building SPAs where the content is dynamically updated without requiring a page reload.
2. **Dynamic Web Applications:** It is ideal for applications that require real-time data binding and interactivity.
3. **Enterprise-Level Applications:** AngularJS is often used in large-scale applications due to its robust structure and features.
4. **Form Validation:** AngularJS provides built-in support for form validation, making it easier to handle user input.
5. **Custom Directives:** Developers can create custom HTML elements and attributes using AngularJS directives.



```html
<!DOCTYPE html>
<html ng-app="myApp">
<head>
  <title>AngularJS Example</title>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script
</head>
<body ng-controller="myCtrl">
  <div>
    <h1>{{message}}</h1>
    <button ng-click="updateMessage()">Click Me</button>
  </div>

  <script>
    // Define the AngularJS module
    var app = angular.module('myApp', []);

    // Define the controller
    app.controller('myCtrl', function($scope) {
      // Initialize the message
      $scope.message = 'Hello, World!';

      // Function to update the message
      $scope.updateMessage = function() {
        $scope.message = 'Hello, AngularJS!';
      };
    });
  </script>
</body>
</html>
```



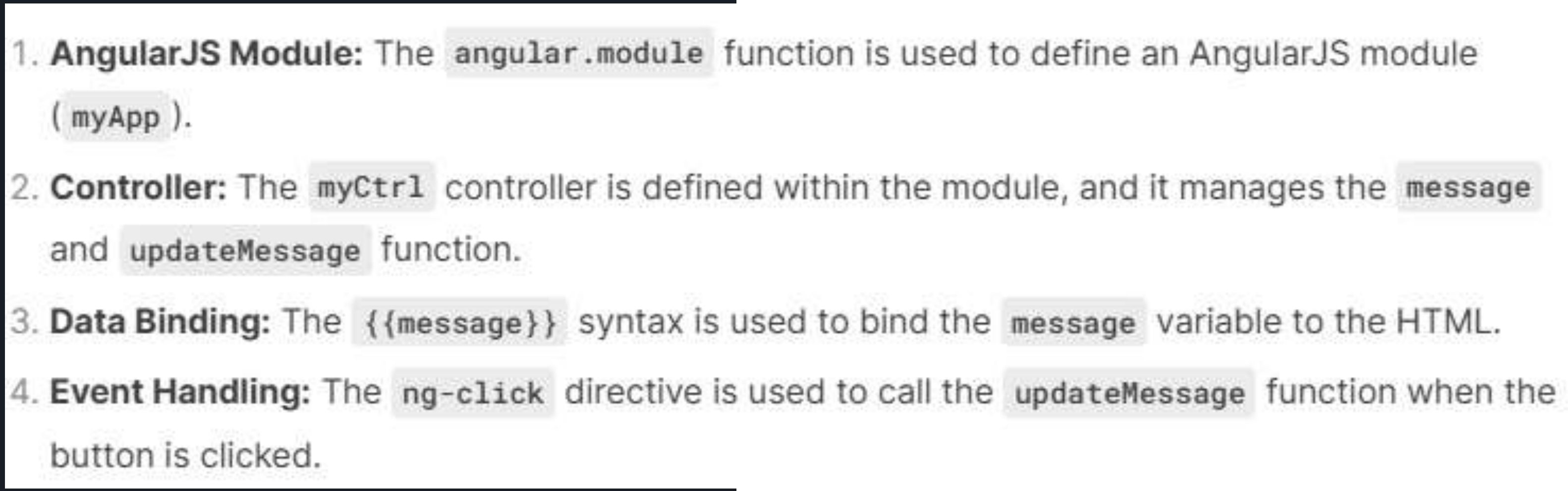
Sample Output

1. **Initial Output:**

```
Hello, World!
[Click Me Button]
```

2. **After Clicking the Button:**

```
Hello, AngularJS!
[Click Me Button]
```



1. **AngularJS Module:** The `angular.module` function is used to define an AngularJS module (`myApp`).
2. **Controller:** The `myCtrl` controller is defined within the module, and it manages the `message` and `updateMessage` function.
3. **Data Binding:** The `{{message}}` syntax is used to bind the `message` variable to the HTML.
4. **Event Handling:** The `ng-click` directive is used to call the `updateMessage` function when the button is clicked.

Major Differences Between AngularJS and React

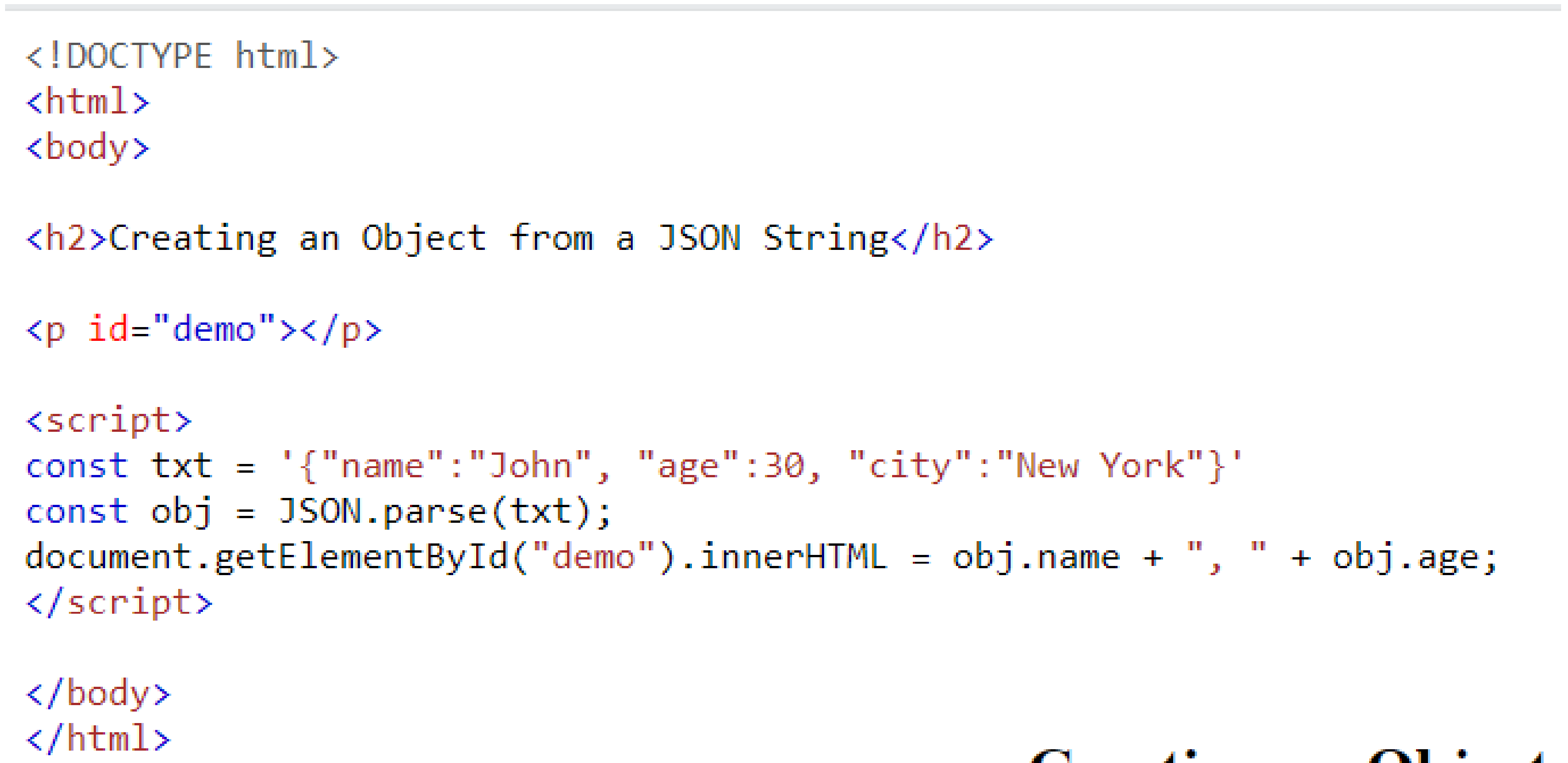
Feature	AngularJS	React
Type	Full-fledged MVC framework	JavaScript library for building UIs
Architecture	Model-View-Controller (MVC)	Component-based
Data Binding	Two-way data binding	One-way data binding
DOM Manipulation	Real DOM	Virtual DOM
Learning Curve	Steeper due to complex concepts	Easier due to simpler concepts
Performance	Slower for large applications	Faster due to virtual DOM
Ecosystem	Built-in features (e.g., routing, HTTP)	Requires additional libraries
Language	JavaScript (with Angular-specific syntax)	JavaScript (with JSX syntax)
Community	Large, but declining (Angular 1.x)	Very large and active
Use Case	Enterprise-level applications	Single-page applications, reusable UIs

JSON

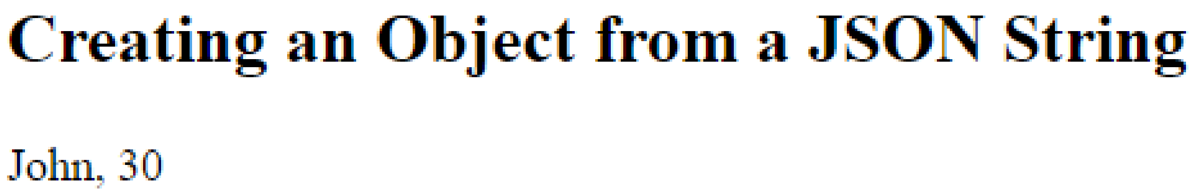
- JSON stands for **J**ava**S**cript **O**bject **N**otation
- JSON is a lightweight data-interchange format
- JSON is plain text written in JavaScript object notation
- JSON is used to send data between computers
- JSON is language independent *****

- The JSON format is syntactically similar to the code for creating JavaScript objects. Because of this, a JavaScript program can easily convert JSON data into JavaScript objects.
- Since the format is text only, JSON data can easily be sent between computers, and used by any programming language.

```
'{"name":"John", "age":30, "car":null}'
```



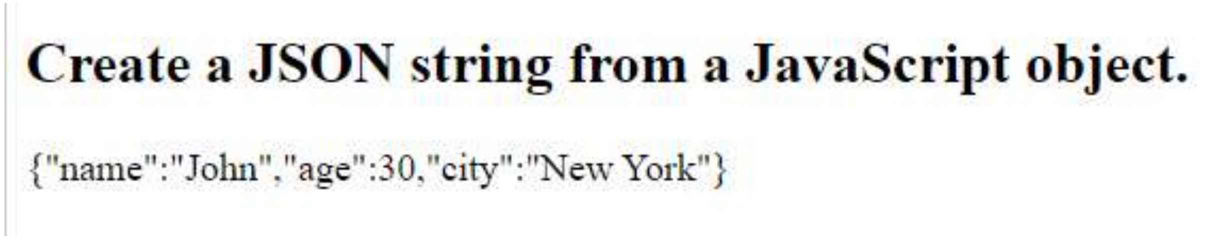
```html
<!DOCTYPE html>
<html>
<body>

<h2>Creating an Object from a JSON String</h2>

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

<script>
const txt = '{"name":"John", "age":30, "city":"New York"}'
const obj = JSON.parse(txt);
document.getElementById("demo").innerHTML = obj.name + ", " + obj.age;
</script>

</body>
</html>
```



Creating an Object from a JSON String

John, 30

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

<h2>Create a JSON string from a JavaScript object.</h2>
<p id="demo"></p>

<script>
const obj = {name: "John", age: 30, city: "New York"};
const myJSON = JSON.stringify(obj);
document.getElementById("demo").innerHTML = myJSON;
</script>

</body>
</html>
```



Create a JSON string from a JavaScript object.

```
{"name":"John","age":30,"city":"New York"}
```

Lab exercises

28. Write an example to set cookie and display that cookie.
29. Write an example of AJAX
30. Write an example of jQuery
31. Write a program to convert JSON into JavaScript Object