JS

JavaScript Class

#JavaScript Notes

String

JS

String

- JavaScript uses string data type to represent textual data. It is marked by enclosing in quotes or double quotes.
- Each element in the String occupies a position index. The first element is at index 0, the next at index 1, and so on.
- The length of a String is the number of elements in it.

- 'This is a string'
- "Another string here"

Escape sequences

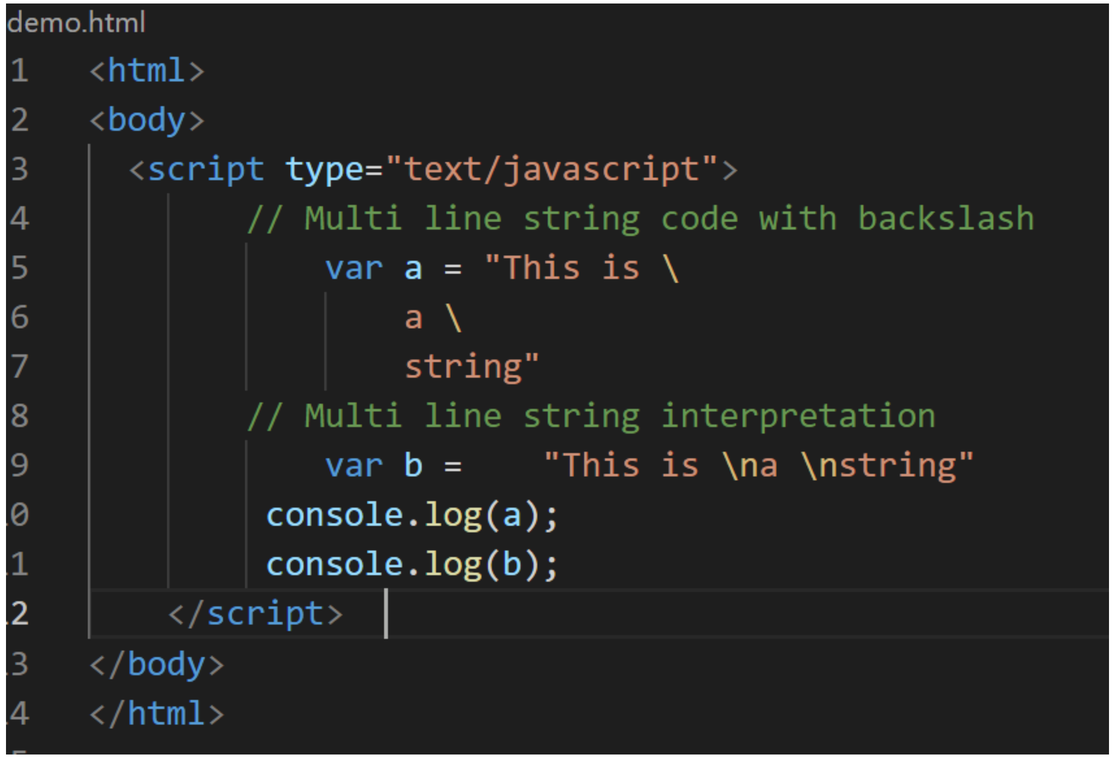
- Sometimes you need to put quotation mark inside a string. JavaScript provides escape sequences in the form of backslash \ to handle this kind of operation

- 'I\'m a Software Developer and I work with \'JavaScript\'“

- Output
  - // I'm a Software Developer and I work with 'JavaScript'

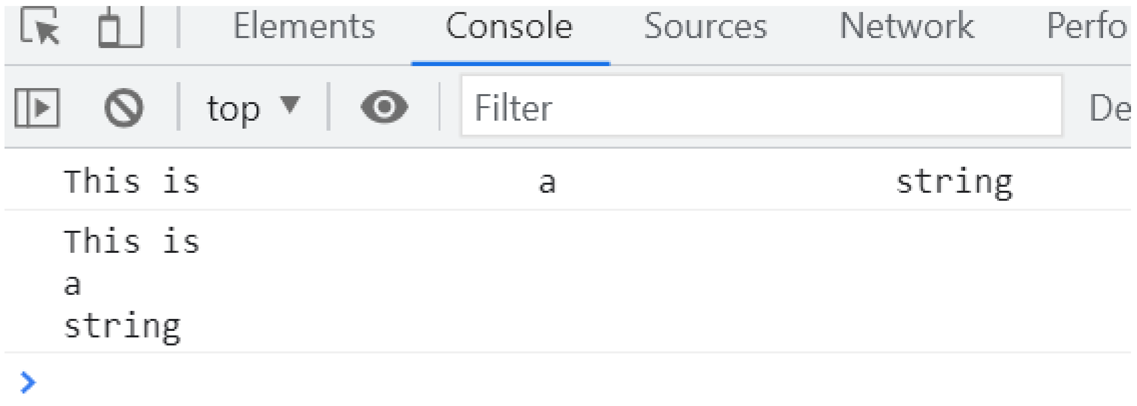
Escape Sequence

- Other forms of backslash escape sequence includes: ' single quote " double quote \ backslash \n new line \t tab \b backspace You might use them when you deal with string data types



demo.html

```html
<html>
<body>
  <script type="text/javascript">
        // Multi line string code with backslash
            var a = "This is \
                a \
                string"
        // Multi line string interpretation
            var b =    "This is \na \nstring"
         console.log(a);
         console.log(b);
    </script>
</body>
</html>
```



```
This is                  a                    string
This is
a
string
```

String template literal

- Template literals are string literals that allows developer to embed expressions in a string. They are marked by the enclosing back-tick mark ``` that surrounds them.

```
// template literals can create multi line string interpretation without \n
 `This is
 a
 string`
```

- Template literals can use variables or expressions as part of the string. This is done by using the ${} to enclose the expression.

```
let name = "Akash";

 `Your name is ${name}.`

 // normal string equivalent
 'Your name is ' + name + ' I Love JS.'

 // Using expressions
 `The result of 10 + 19 is ${10+19}`
```

== and === operator

- Both double equals == and triple equals === operator is used for comparing between two values on which the operator is used on.
- The difference between the two operators is that the double equals == will compare the values loosely, meaning that it will try to convert values with different types before comparing them.
- The triple equals === won’t convert values of different types. It will simply return false when comparing values of different types.
- Comparing two different values between the number value 0 and boolean value false:

Example

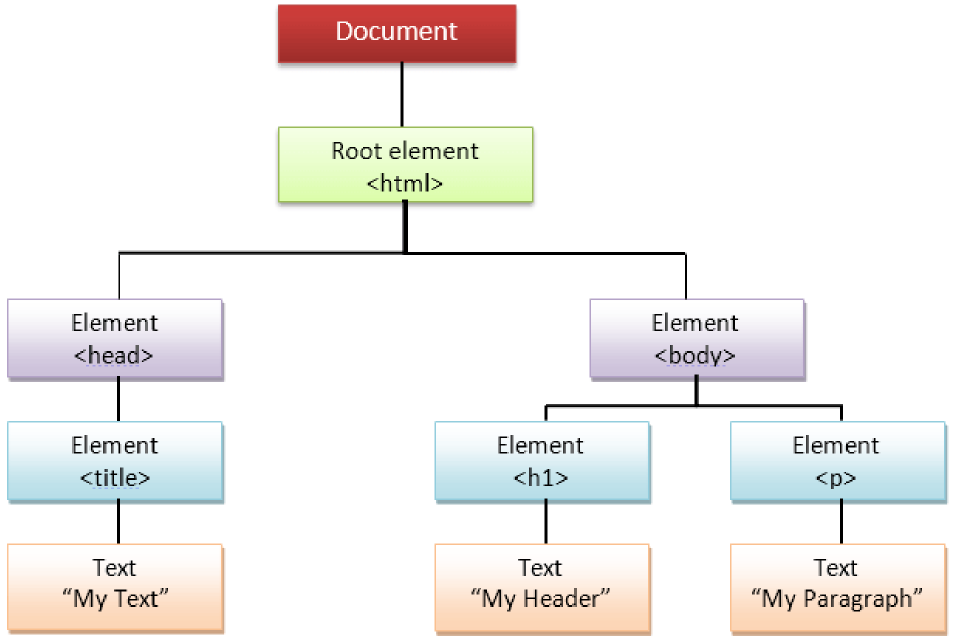
- console.log(0 == false); // true
- console.log(0 === false); // false

- console.log(0 === 0); // true

JS Dom

What is DOM in JavaScript?

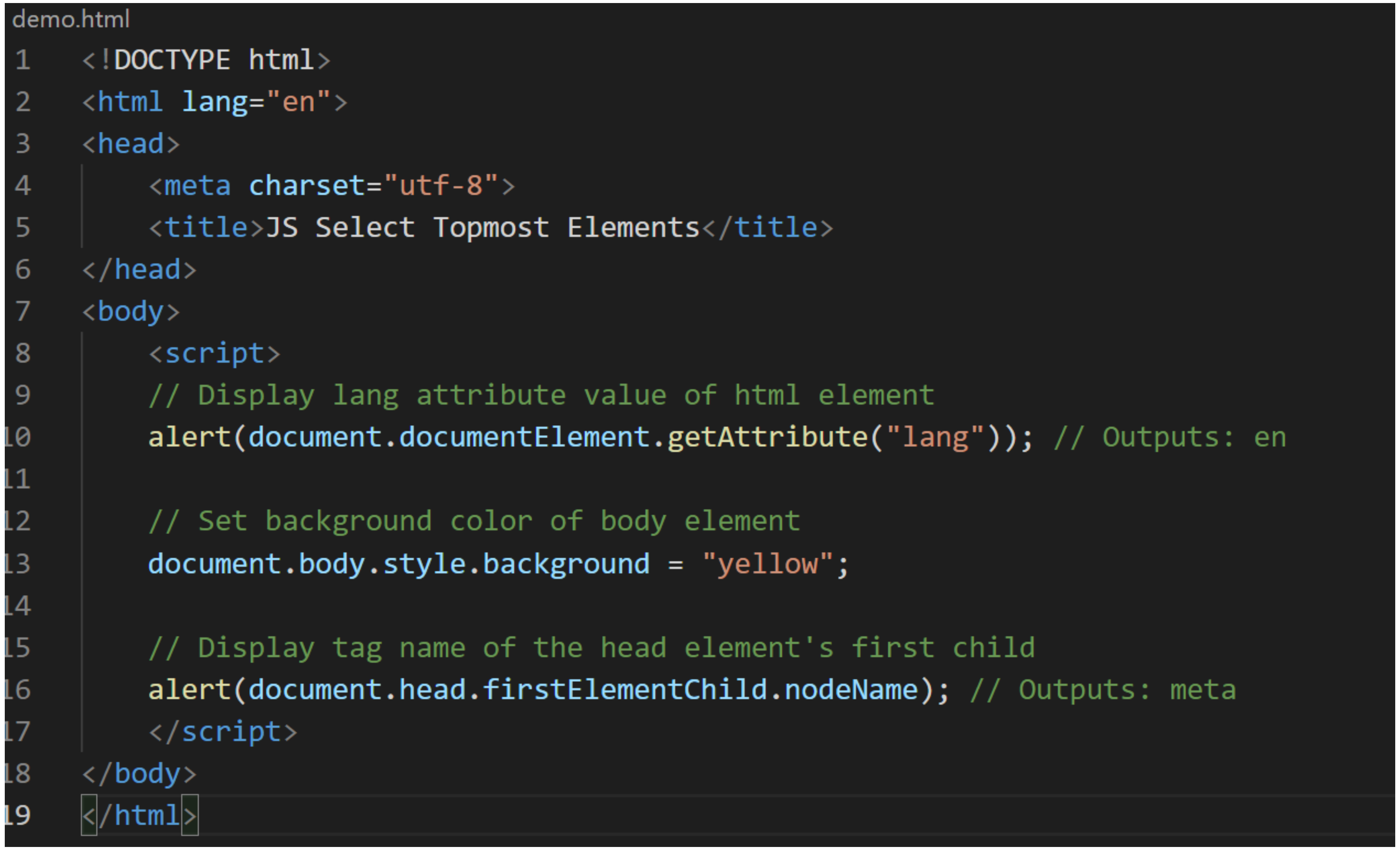
- JavaScript can access all the elements in a webpage making use of Document Object Model (DOM). In fact, the web browser creates a DOM of the webpage when the page is loaded. The DOM model is created as a tree of objects like this:



Selecting the Topmost Elements

- The topmost elements in an HTML document are available directly as document properties. For example, the <html> element can be accessed with document.documentElement property,
- whereas the <head> element can be accessed with document.head property, and the <body> element can be accessed with document.body property.

Example



demo.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JS Select Topmost Elements</title>
</head>
<body>
    <script>
    // Display lang attribute value of html element
    alert(document.documentElement.getAttribute("lang")); // Outputs: en

    // Set background color of body element
    document.body.style.background = "yellow";

    // Display tag name of the head element's first child
    alert(document.head.firstElementChild.nodeName); // Outputs: meta
    </script>
</body>
</html>
```

How to use DOM and Events

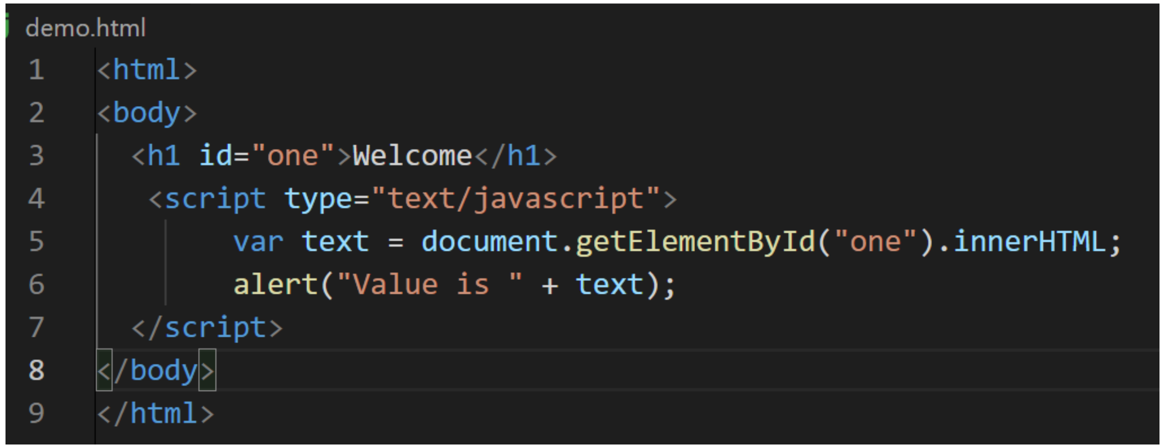
- Using DOM, JavaScript can perform multiple tasks.
- It can create new elements and attributes, change the existing elements and attributes and even remove existing elements and attributes.
- JavaScript can also react to existing events and create new events in the page.

Method

- getElementById()
- getElementsByName()
- getElementsByClassName()
- getElementsByTagName()

getElementById()

- To access elements and attributes whose id is set.
- The id is case-sensitive. For example, the 'root' and 'Root' are totally different id.
- id is unique within a document. However, HTML is a forgiving language. If a document has more than one element with the same id, the getElementById() method returns the first one it encounters.

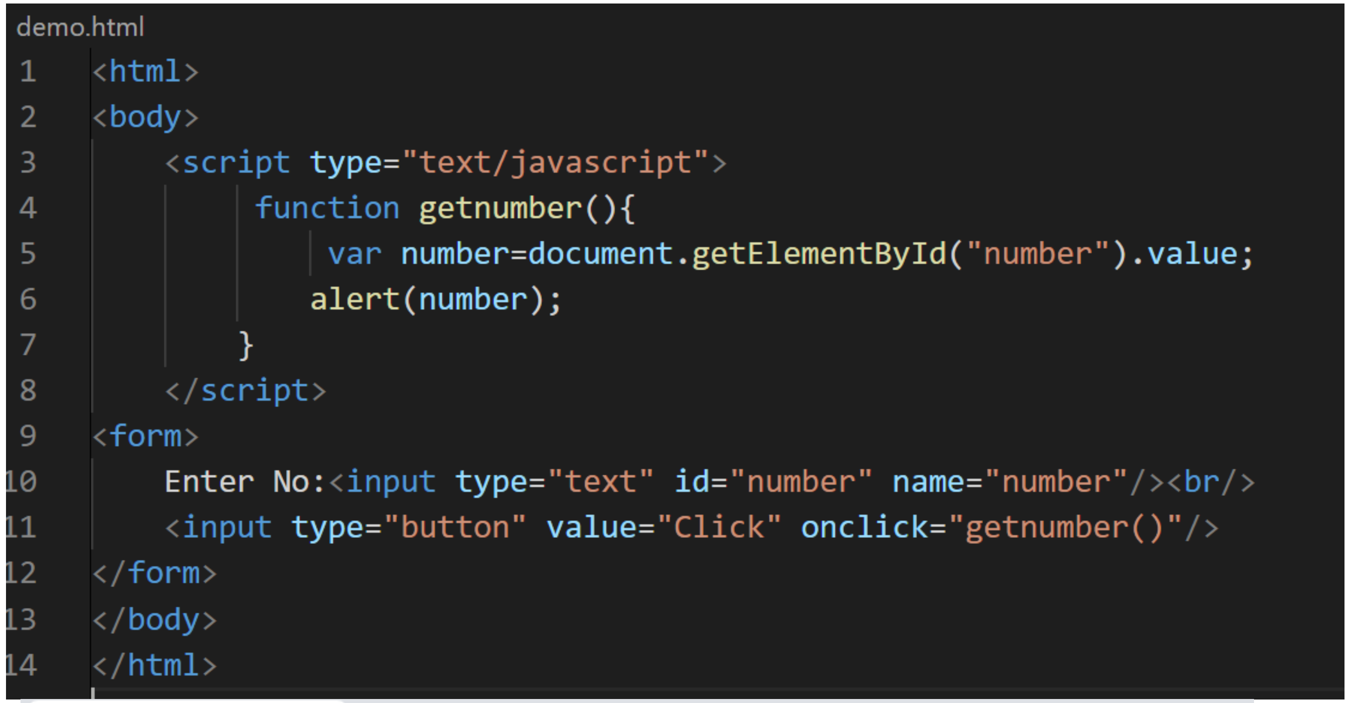


demo.html

```
<html>
<body>
  <h1 id="one">Welcome</h1>
   <script type="text/javascript">
        var text = document.getElementById("one").innerHTML;
        alert("Value is " + text);
  </script>
</body>
</html>
```

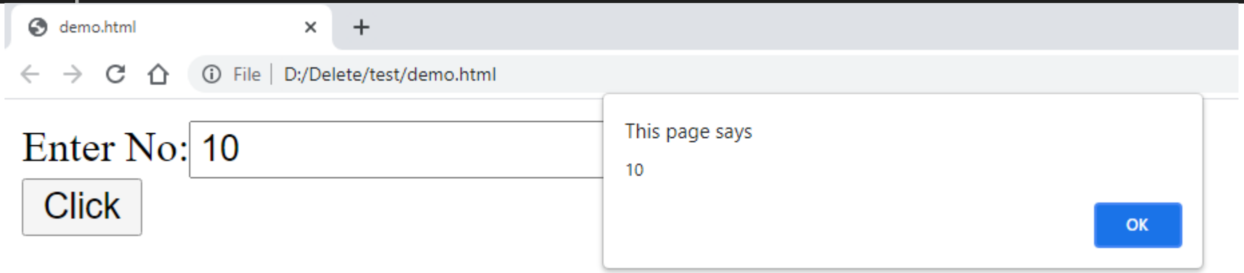
```
<html>
<body>
  <h1 id="one">Welcome</h1>
   <script type="text/javascript">
        var text = document.getElementById("one").innerHTML;
        alert("Value is " + text);
  </script>
</body>
</html>
```

Get Value of Form



demo.html

```
<html>
<body>
    <script type="text/javascript">
        function getnumber(){
            var number=document.getElementById("number").value;
            alert(number);
        }
    </script>
<form>
    Enter No:<input type="text" id="number" name="number"/><br/>
    <input type="button" value="Click" onclick="getnumber()"/>
</form>
</body>
</html>
```

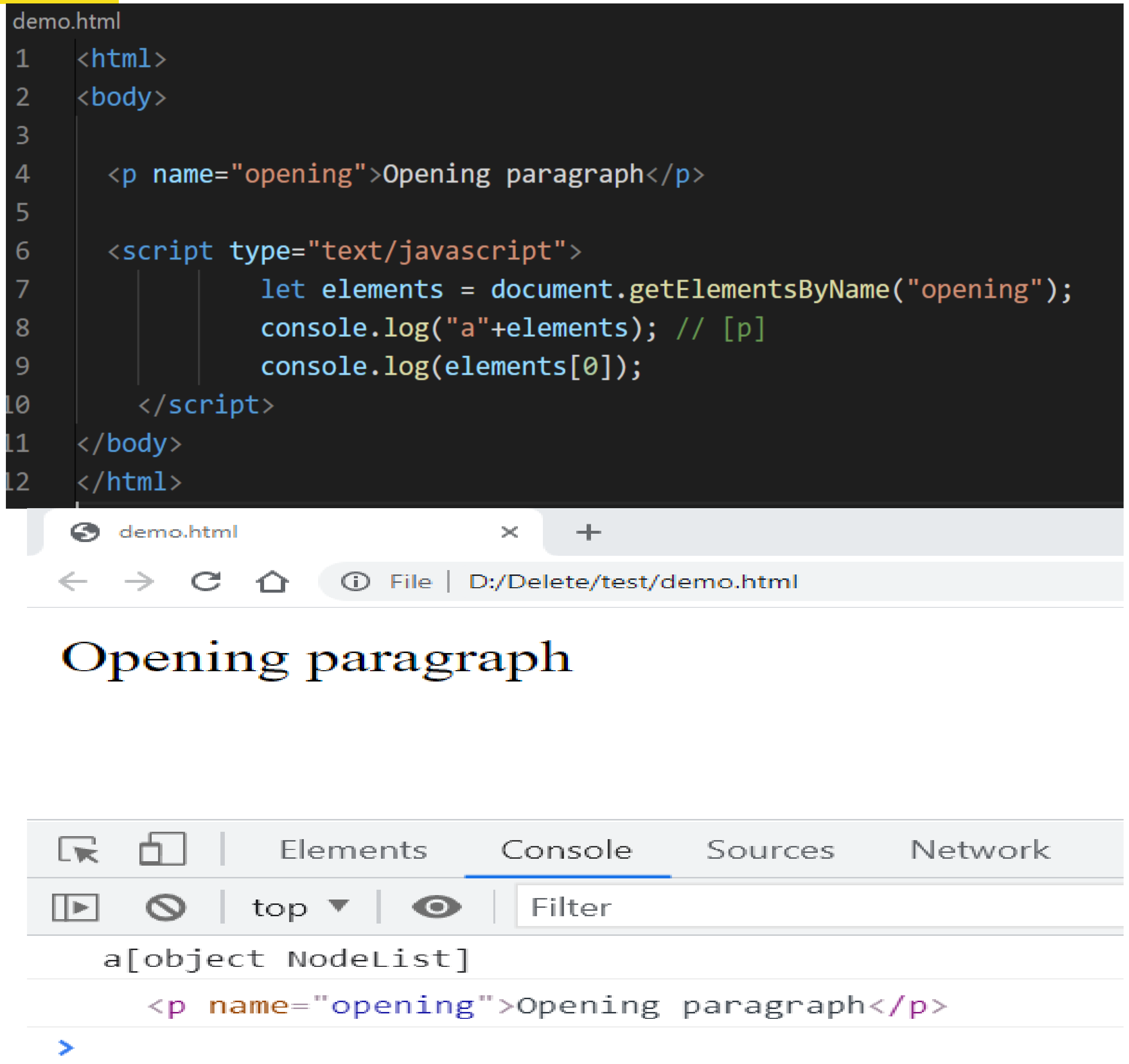


```
<html>
<body>
   <script type="text/javascript">
       function getnumber(){
          var number=document.getElementById("number").value;
         alert(number);
       }
   </script>
<form>
   Enter No:<input type="text" id="number" name="number"/><br/>
   <input type="button" value="Click" onclick="getnumber()"/>
</form>
</body>
</html>
```

getElementsByName()

- The **document.getElementsByName()** method returns all the element of specified name.
- The getElementsByName() method returns an array-like object called HTMLCollection which stores all elements that matches the value passed as the method’s argument.

Example



demo.html

```
<html>
<body>

  <p name="opening">Opening paragraph</p>

  <script type="text/javascript">
          let elements = document.getElementsByName("opening");
          console.log("a"+elements); // [p]
          console.log(elements[0]);
    </script>
</body>
</html>
```

demo.html

File | D:/Delete/test/demo.html

Opening paragraph

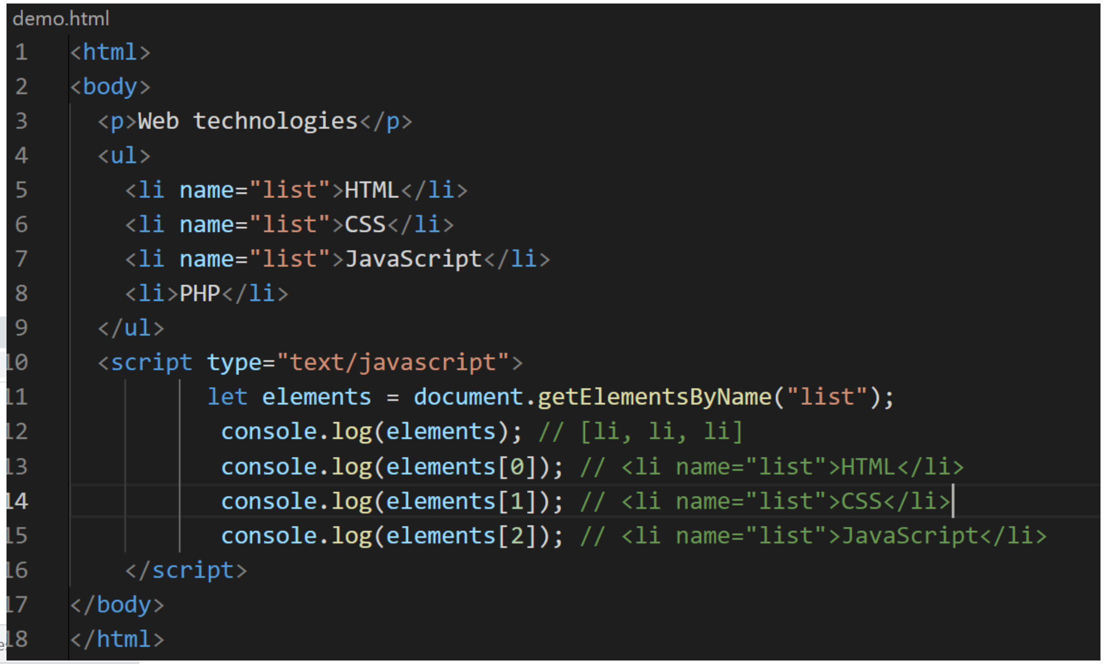
Elements Console Sources Network

top Filter

```
a[object NodeList]
  <p name="opening">Opening paragraph</p>
>
```

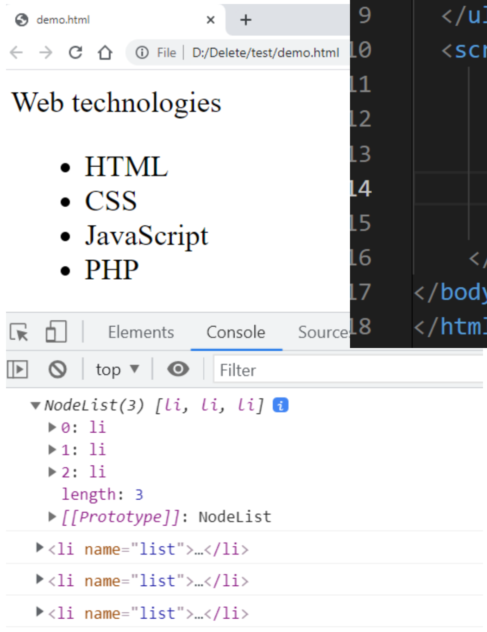
```
<html>
<body>

  <p name="opening">Opening paragraph</p>

  <script type="text/javascript">
            let elements = document.getElements
ByName("opening");
            console.log("a"+elements); // [p]
            console.log(elements[0]);
    </script>
</body>
</html>
```

Example



demo.html

```html
<html>
<body>
  <p>Web technologies</p>
  <ul>
    <li name="list">HTML</li>
    <li name="list">CSS</li>
    <li name="list">JavaScript</li>
    <li>PHP</li>
  </ul>
  <script type="text/javascript">
          let elements = document.getElementsByName("list");
           console.log(elements); // [li, li, li]
           console.log(elements[0]); // <li name="list">HTML</li>
           console.log(elements[1]); // <li name="list">CSS</li>
           console.log(elements[2]); // <li name="list">JavaScript</li>
    </script>
</body>
</html>
```



Web technologies

- HTML
- CSS
- JavaScript
- PHP

```
▾NodeList(3) [li, li, li]
  ▸0: li
  ▸1: li
  ▸2: li
   length: 3
  ▸[[Prototype]]: NodeList
▸<li name="list">…</li>
▸<li name="list">…</li>
▸<li name="list">…</li>
```

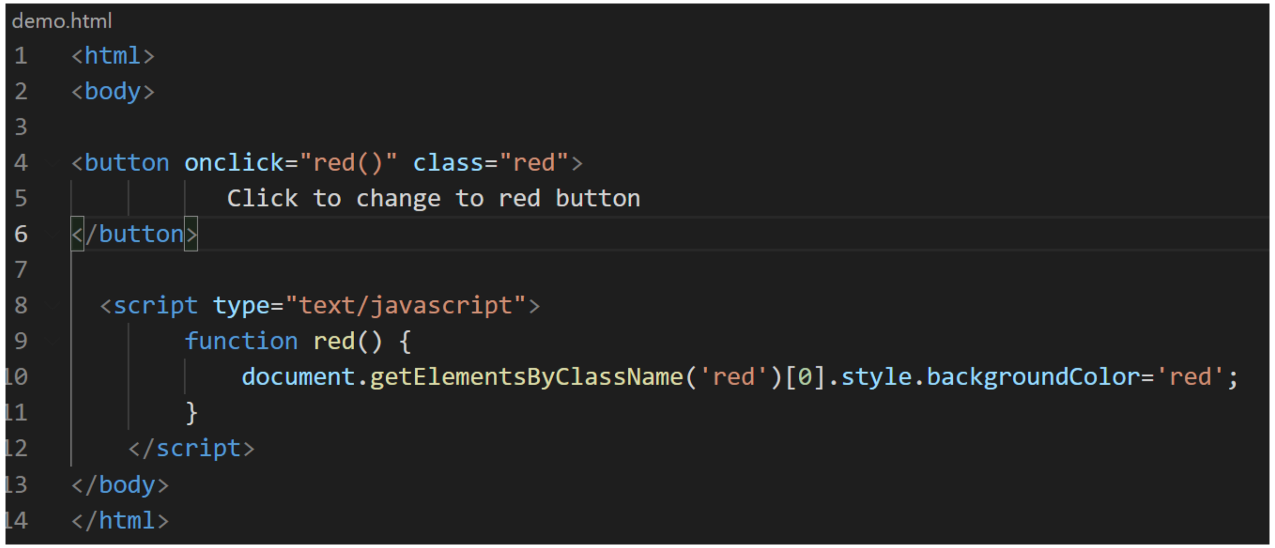
```html
<html>
<body>
 <p>Web technologies</p>
 <ul>
  <li name="list">HTML</li>
  <li name="list">CSS</li>
  <li name="list">JavaScript</li>
  <li>PHP</li>
 </ul>
 <script type="text/javascript">
        let elements = document.getElementsByName("list");
         console.log(elements); // [li, li, li]
         console.log(elements[0]); // <li name="list">HTML</li>
         console.log(elements[1]); // <li name="list">CSS</li>
         console.log(elements[2]); // <li name="list">JavaScript</li>
  </script>
</body>
</html>
```

getElementsByClassName()

- getElementsByClassName() method returns an object containing all the elements with the specified class names in the document as objects.
- Each element in the returned object can be accessed by its index.
- This method can be called upon any individual element to search for its descendant elements with the specified class names.

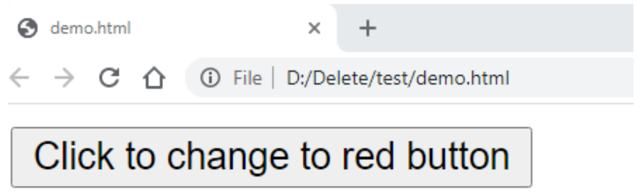
```
document.getElementsByClassName(classnames);
```

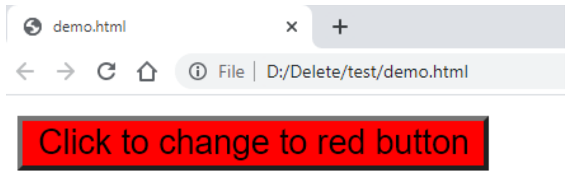
Example

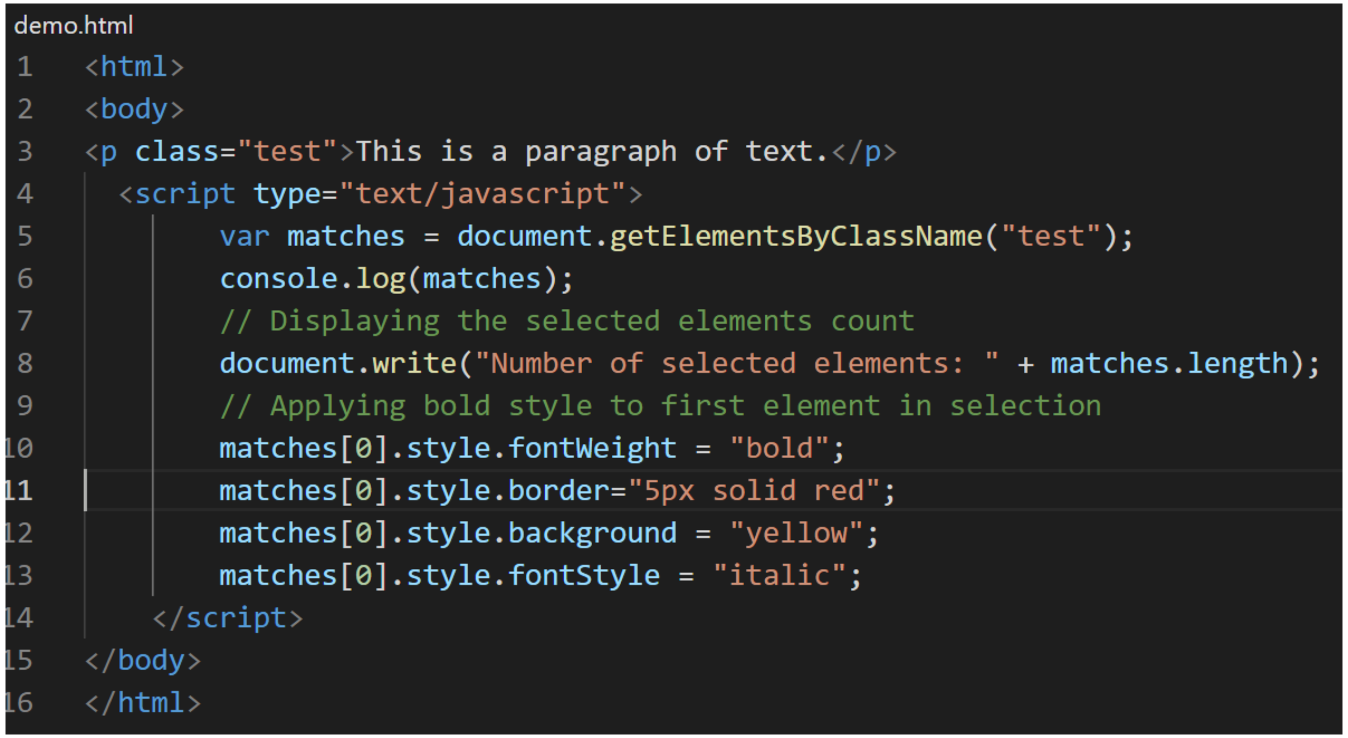

demo.html

```html
<html>
<body>

<button onclick="red()" class="red">
        Click to change to red button
</button>

  <script type="text/javascript">
        function red() {
            document.getElementsByClassName('red')[0].style.backgroundColor='red';
        }
    </script>
</body>
</html>
```

```html
<html>
<body>

<button onclick="red()" class="red">
           Click to change to red button
</button>

  <script type="text/javascript">
        function red() {
            document.getElementsByClassName('red')[0].
style.backgroundColor='red';
        }
    </script>
</body>
</html>
```

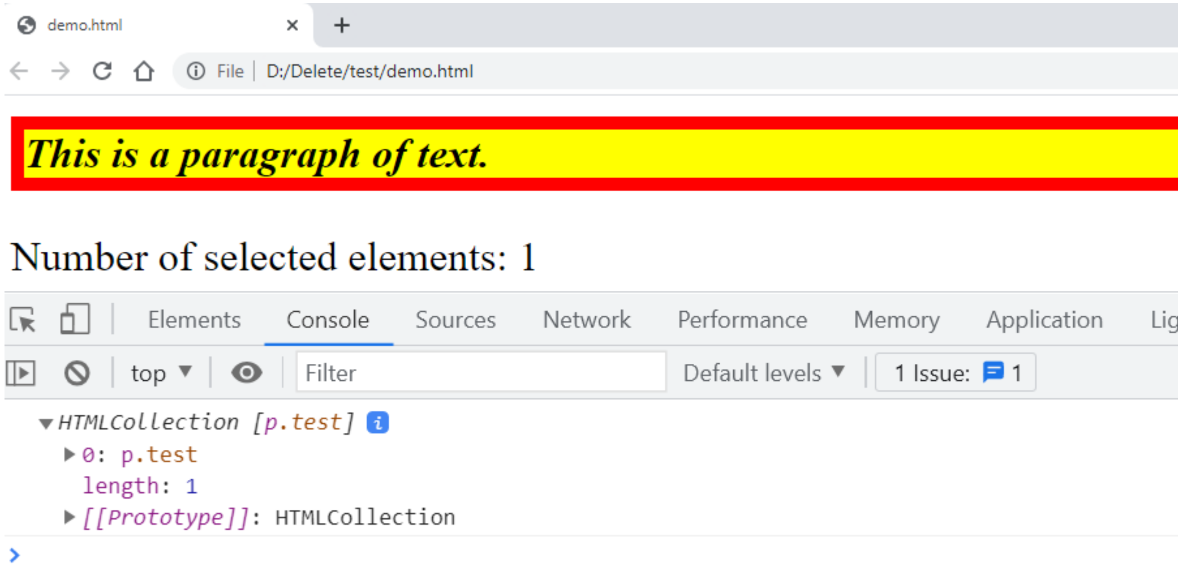






demo.html

```
<html>
<body>
<p class="test">This is a paragraph of text.</p>
  <script type="text/javascript">
      var matches = document.getElementsByClassName("test");
      console.log(matches);
      // Displaying the selected elements count
      document.write("Number of selected elements: " + matches.length);
      // Applying bold style to first element in selection
      matches[0].style.fontWeight = "bold";
      matches[0].style.border="5px solid red";
      matches[0].style.background = "yellow";
      matches[0].style.fontStyle = "italic";
    </script>
</body>
</html>
```



This is a paragraph of text.

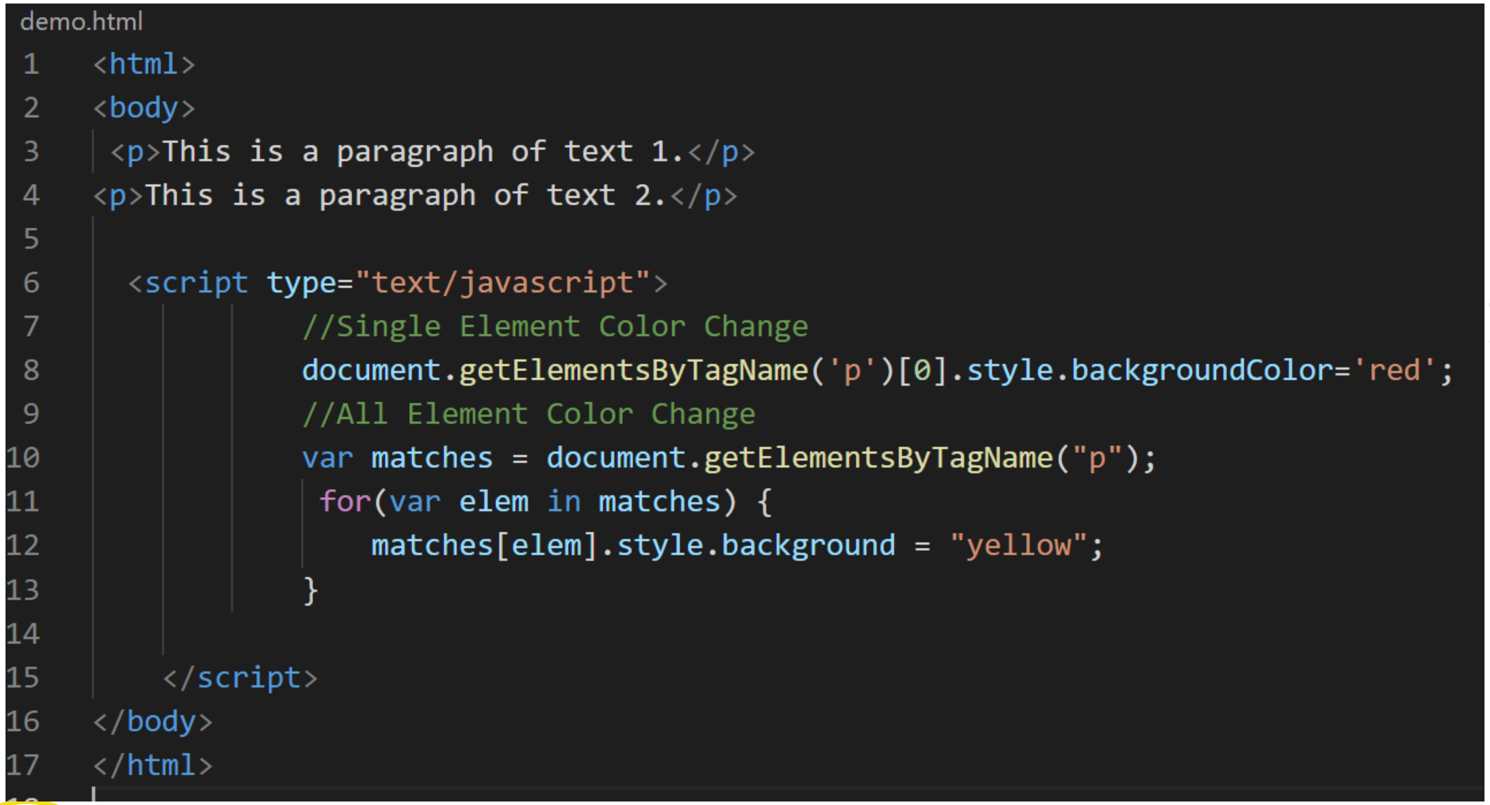
Number of selected elements: 1

```
▼HTMLCollection [p.test]
  ▶0: p.test
   length: 1
  ▶[[Prototype]]: HTMLCollection
>
```

```
<html>
<body>
<p class="test">This is a paragraph of text.</p>
  <script type="text/javascript">
        var matches = document.getElementsByClassName("test");
        console.log(matches);
        // Displaying the selected elements count
        document.write("Number of selected elements: " + matches.length);
        // Applying bold style to first element in selection
        matches[0].style.fontWeight = "bold";
        matches[0].style.border="5px solid red";
        matches[0].style.background = "yellow";
        matches[0].style.fontStyle = "italic";
    </script>
</body>
</html>
```

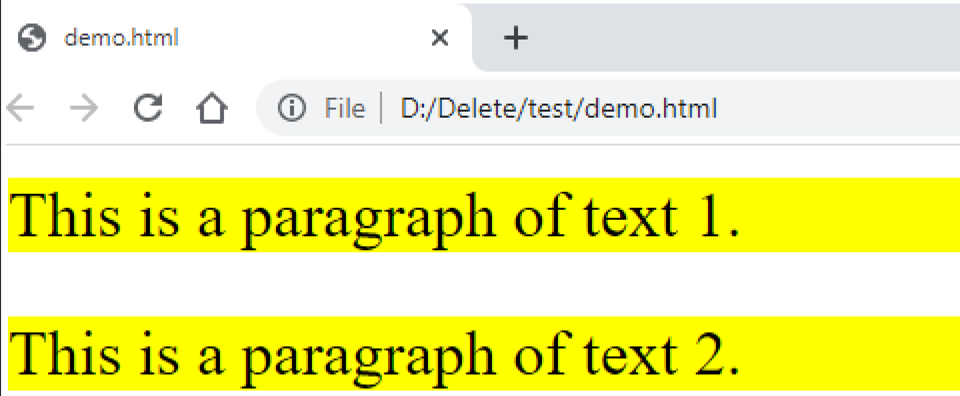
getElementsByTagName()

- You can also select HTML elements by tag name using the getElementsByTagName() method. This method also returns an array-like object of all child elements with the given tag name.



demo.html

```
<html>
<body>
 <p>This is a paragraph of text 1.</p>
<p>This is a paragraph of text 2.</p>

  <script type="text/javascript">
          //Single Element Color Change
          document.getElementsByTagName('p')[0].style.backgroundColor='red';
          //All Element Color Change
          var matches = document.getElementsByTagName("p");
           for(var elem in matches) {
              matches[elem].style.background = "yellow";
          }

    </script>
</body>
</html>
```



demo.html

File | D:/Delete/test/demo.html

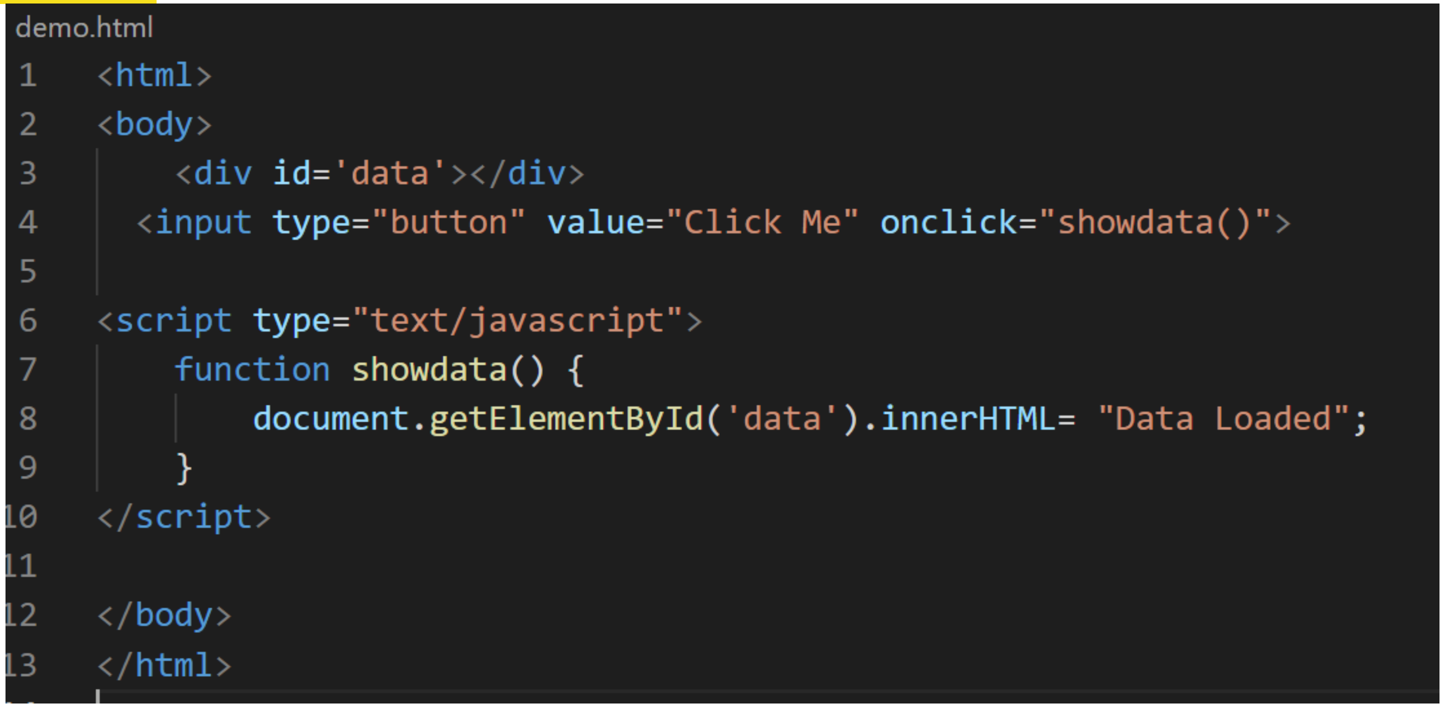
This is a paragraph of text 1.

This is a paragraph of text 2.

innerHTML

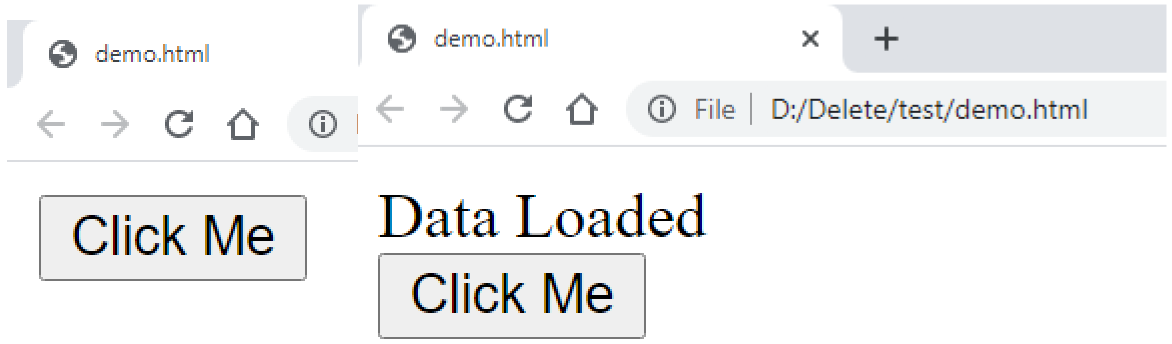
- The **innerHTML** property can be used to write the dynamic html on the html document.
- It is used mostly in the web pages to generate the dynamic html such as registration form, comment form, links etc.
-

Load Data



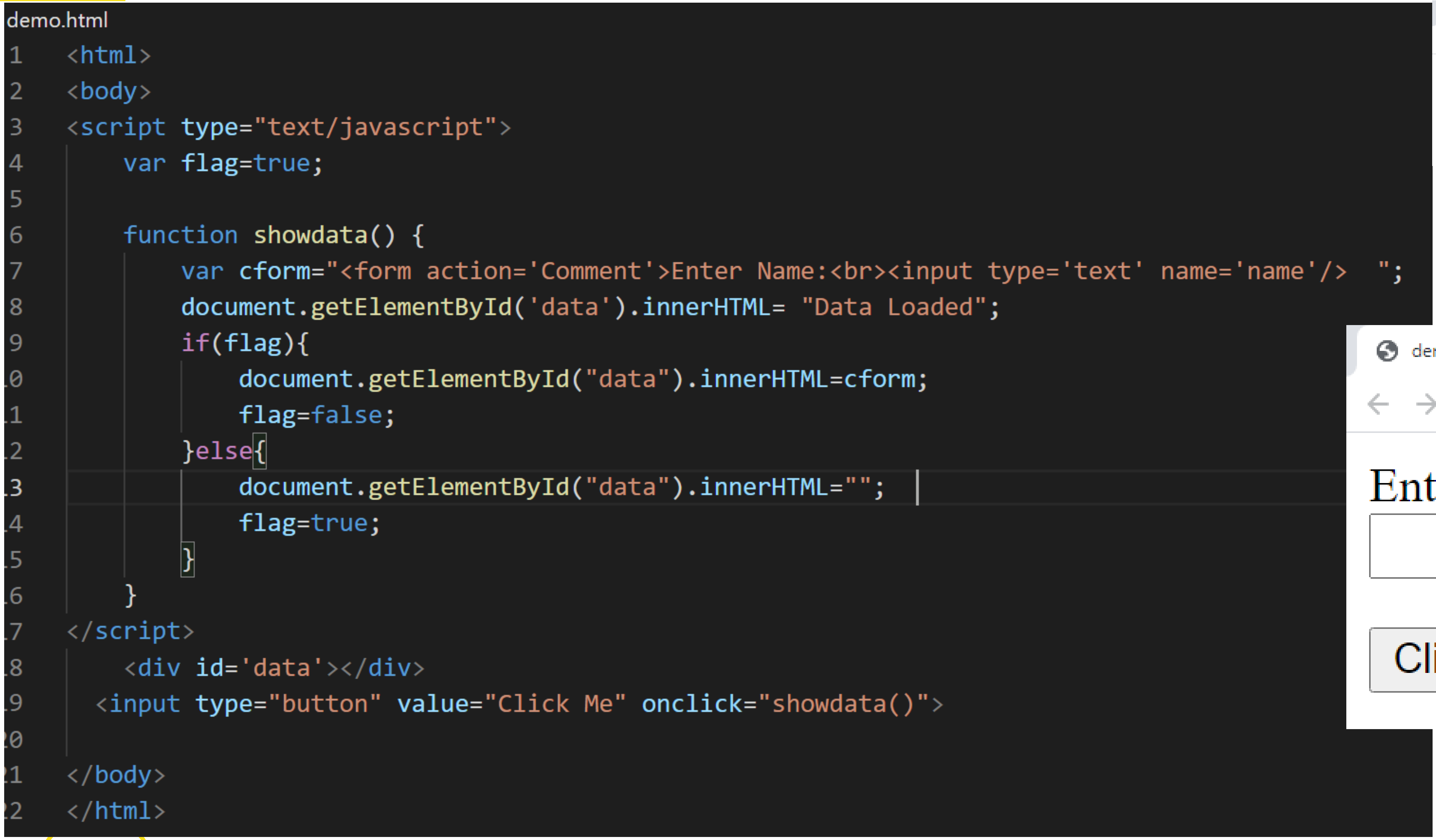
demo.html

```
<html>
<body>
    <div id='data'></div>
  <input type="button" value="Click Me" onclick="showdata()">

<script type="text/javascript">
    function showdata() {
        document.getElementById('data').innerHTML= "Data Loaded";
    }
</script>

</body>
</html>
```



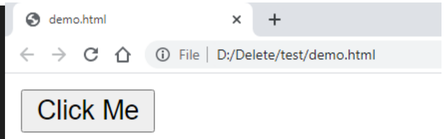
```
<html>
<body>
    <div id='data'></div>
  <input type="button" value="Click Me" onclick="showdata()">

<script type="text/javascript">
    function showdata() {
        document.getElementById('data').innerHTML= "Data Loaded";
    }
</script>

</body>
</html>
```

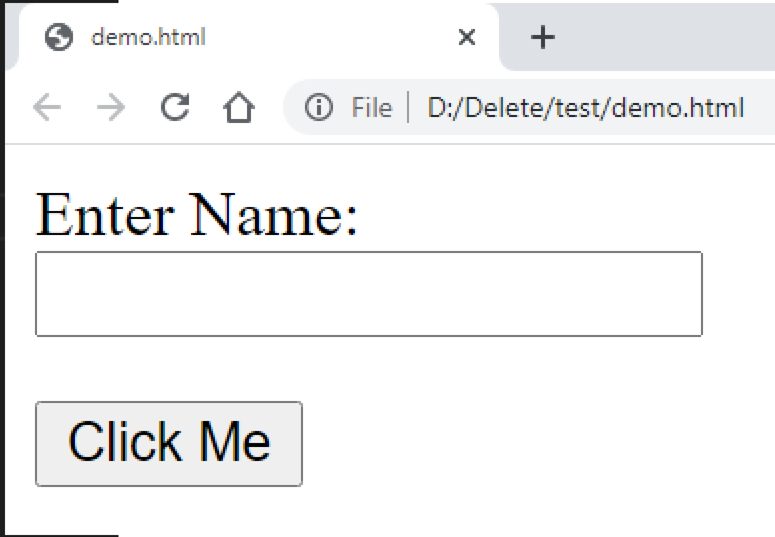
Show Hide Form



demo.html

```
<html>
<body>
<script type="text/javascript">
    var flag=true;

    function showdata() {
        var cform="<form action='Comment'>Enter Name:<br><input type='text' name='name'/>  ";
        document.getElementById('data').innerHTML= "Data Loaded";
        if(flag){
            document.getElementById("data").innerHTML=cform;
            flag=false;
        }else{
            document.getElementById("data").innerHTML="";
            flag=true;
        }
    }
</script>
    <div id='data'></div>
  <input type="button" value="Click Me" onclick="showdata()">

</body>
</html>
```

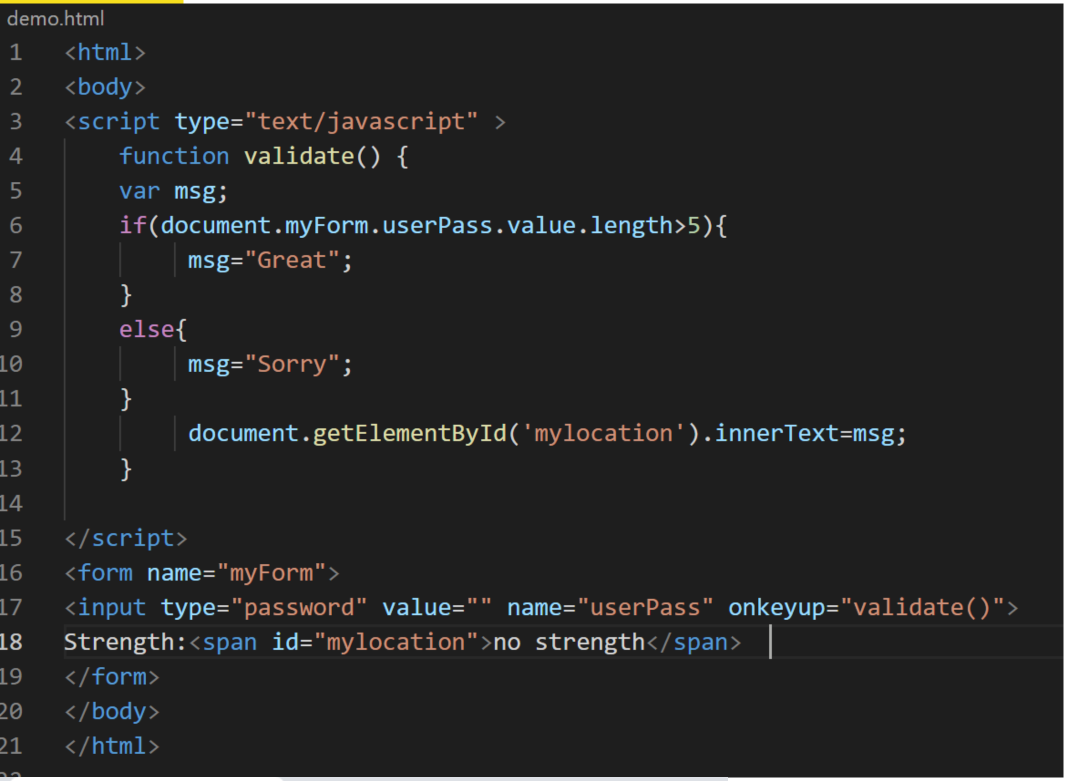




innerText

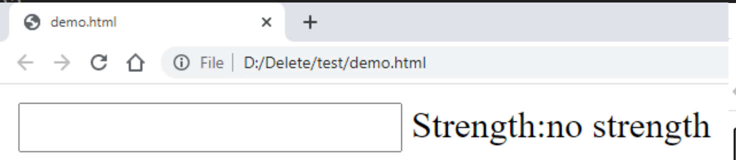
- The **innerText** property can be used to write the dynamic text on the html document. Here, text will not be interpreted as html text but a normal text.
- It is used mostly in the web pages to generate the dynamic content such as writing the validation message, password strength etc.

Example


demo.html

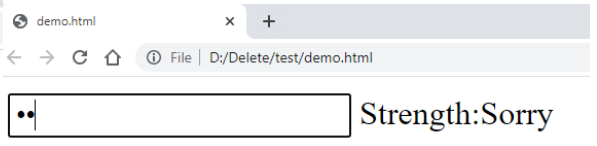
```
<html>
<body>
<script type="text/javascript" >
    function validate() {
    var msg;
    if(document.myForm.userPass.value.length>5){
         msg="Great";
    }
    else{
         msg="Sorry";
    }
         document.getElementById('mylocation').innerText=msg;
    }

</script>
<form name="myForm">
<input type="password" value="" name="userPass" onkeyup="validate()">
Strength:<span id="mylocation">no strength</span>
</form>
</body>
</html>
```

```
<html>
<body>
<script type="text/javascript" >
    function validate() {
    var msg;
    if(document.myForm.userPass.value.length>5){
         msg="Great";
    }
    else{
         msg="Sorry";
    }
         document.getElementById('mylocation').innerText=msg;
    }

</script>
<form name="myForm">
<input type="password" value="" name="userPass" onkeyup="validate()">
Strength:<span id="mylocation">no strength</span>
</form>
</body>
</html>
```


demo.html

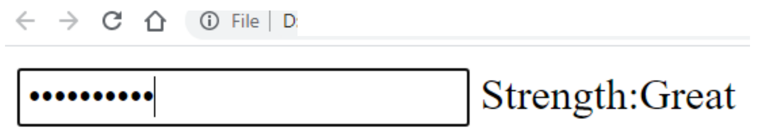
File | D:/Delete/test/demo.html

Strength:no strength


demo.html

File | D:/Delete/test/demo.html

•• Strength:Sorry


File | D:

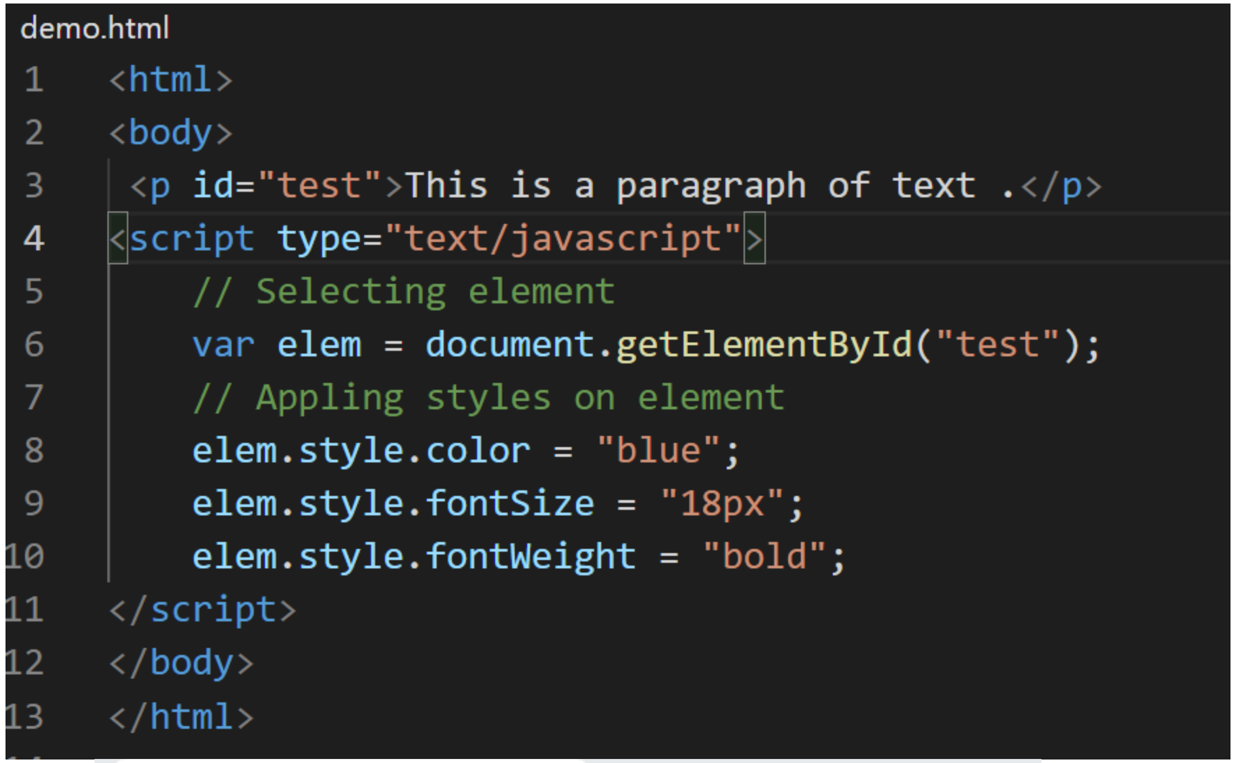
•••••••••• Strength:Great

JavaScript DOM Styling

Styling DOM Elements in JavaScript

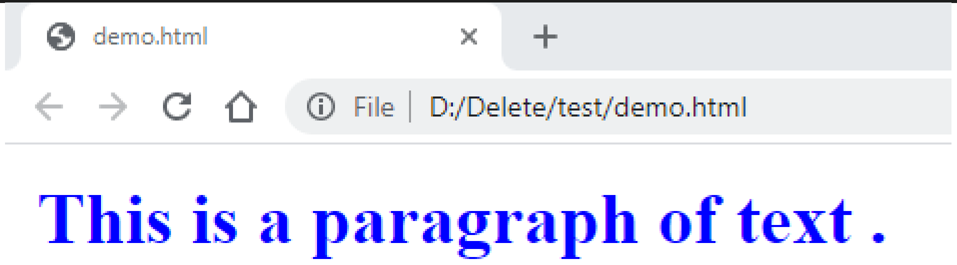
- You can also apply style on HTML elements to change the visual presentation of HTML documents dynamically using JavaScript.
- You can set almost all the styles for the elements like, fonts, colors, margins, borders, background images, text alignment, width and height, position, and so on.

Example



demo.html

```
<html>
<body>
 <p id="test">This is a paragraph of text .</p>
<script type="text/javascript">
    // Selecting element
    var elem = document.getElementById("test");
    // Appling styles on element
    elem.style.color = "blue";
    elem.style.fontSize = "18px";
    elem.style.fontWeight = "bold";
</script>
</body>
</html>
```



This is a paragraph of text .

```
<html>
<body>
 <p id="test">This is a paragraph of text .</p>
<script type="text/javascript">
    // Selecting element
    var elem = document.getElementById("test");
    // Appling styles on element
    elem.style.color = "blue";
    elem.style.fontSize = "18px";
    elem.style.fontWeight = "bold";
</script>
</body>
</html>
```

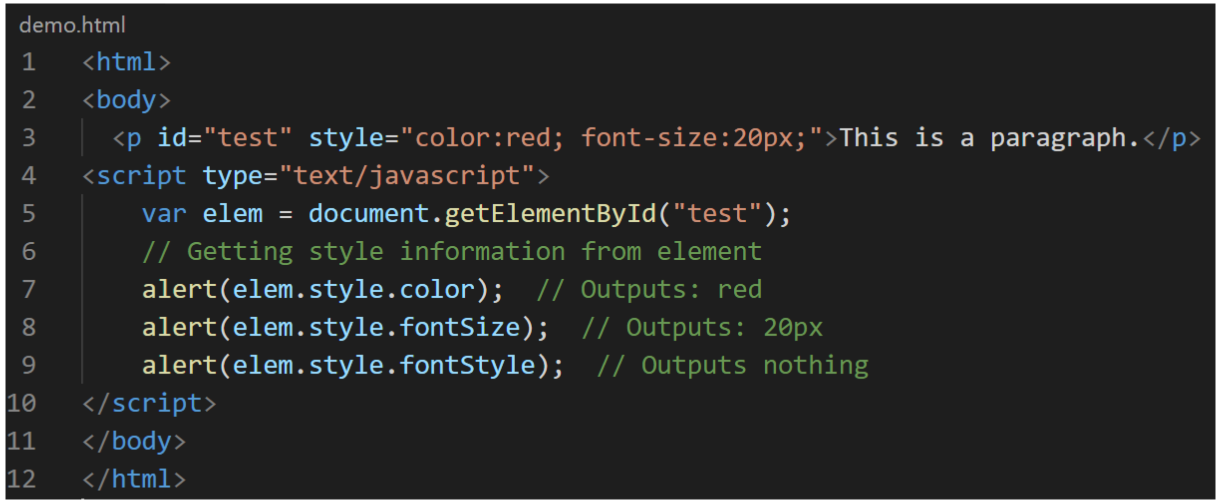
Naming Conventions of CSS Properties in JavaScript

- Many CSS properties, such as font-size, background-image, text-decoration, etc. contain hyphens (-) in their names. Since, in JavaScript hyphen is a reserved operator and it is interpreted as a minus sign, so it is not possible to write an expression, like: elem.style.font-size

- Therefore, in JavaScript, the CSS property names that contain one or more hyphens are converted to intercapitalized style word. It is done by removing the hyphens and capitalizing the letter immediately following each hyphen, thus the CSS property font-size becomes the DOM property fontSize, border-left-style becomes borderLeftStyle, and so on.

Getting Style Information from Elements

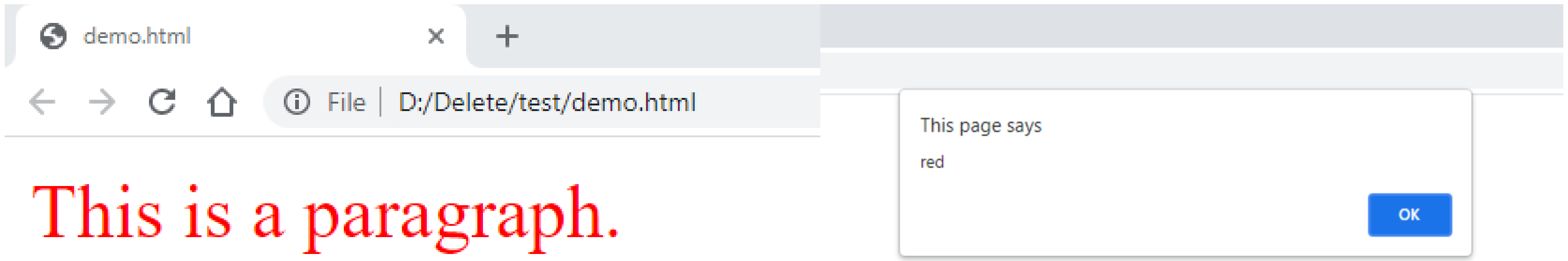
- you get the styles applied on the HTML elements using the style property.



demo.html

```
<html>
<body>
  <p id="test" style="color:red; font-size:20px;">This is a paragraph.</p>
<script type="text/javascript">
    var elem = document.getElementById("test");
    // Getting style information from element
    alert(elem.style.color);  // Outputs: red
    alert(elem.style.fontSize);  // Outputs: 20px
    alert(elem.style.fontStyle);  // Outputs nothing
</script>
</body>
</html>
```

```
<html>
<body>
  <p id="test" style="color:red; font-size:20px;">This is a paragraph.</p>
<script type="text/javascript">
    var elem = document.getElementById("test");
    // Getting style information from element
    alert(elem.style.color);  // Outputs: red
    alert(elem.style.fontSize);  // Outputs: 20px
    alert(elem.style.fontStyle);  // Outputs nothing
</script>
</body>
</html>
```



demo.html

File | D:/Delete/test/demo.html

This is a paragraph.

This page says

red

Objects

JavaScript Objects

- A JavaScript object is just a collection of named values. These named values are usually referred to as properties of the object.
- An object is similar to an array, but the difference is that you define the keys yourself, such as name, age, gender, and so on.

Creating Objects

- An object can be created with curly brackets {} with an optional list of properties.
- A property is a "key: value" pair, where the key (or property name) is always a string, and value (or property value) can be any data type, like strings, numbers, Booleans or complex data type like arrays, functions, and other objects.
- Additionally, properties with functions as their values are often called methods to distinguish them from other properties.

Example

```
var person = {
    name: "Akash",
    age: 30,
    gender: "Male",
    displayName: function() {
        alert(this.name);
    }
};
```

Accessing Object's Properties

- To access or get the value of a property, you can use the dot (.), or square bracket ([]) notation.
- The dot notation is easier to read and write, but it cannot always be used. If the name of the property is not valid (i.e. if it contains spaces or special characters), you cannot use the dot notation; you'll have to use bracket notation

```
var book = {
    "name": "Harry Potter and the Goblet of Fire",
    "author": "J. K. Rowling",
    "year": 2000
};

// Dot notation
document.write(book.author);  // Prints: J. K. Rowling

// Bracket notation
document.write(book["year"]); // Prints: 2000
```

Looping Through Object's Properties

- You can iterate through the key-value pairs of an object using the for...in loop. This loop is specially optimized for iterating over object's properties.

```
 <script>
 var person = {
   name: "Akash",
   age: 30,
   gender: "Male"
};

// Iterating over object properties
for(var i in person) {
   document.write(person[i] + "<br>"); // Prints: name, age and gender
}
   </script>
```

Setting Object's Properties

- you can set the new properties or update the existing one using the dot (.) or bracket ([]) notation.

```
<script>
    var person = {
      name: "Akash",
      age: 30,
      gender: "Male"
    };
    //Add New Value
    person.country = "India";
    // Iterating over object properties
    for(var i in person) {
      document.write(person[i] + "<br>"); // Prints: name, age and gender Country
    }
  </script>
```

Deleting Object's Properties

- The delete operator can be used to completely remove properties from an object. Deleting is the only way to actually remove a property from an object.
- Setting the property to undefined or null only changes the value of the property. It does not remove property from the object.

```
<script>
      var person = {
         name: "Akash",
         age: 30,
         gender: "Male"
      };
      //Delete Value
      delete person.age;
      // Iterating over object properties
      for(var i in person) {
         document.write(person[i] + "<br>"); // Prints: name and gender
      }
   </script>
```

JSON

JavaScript JSON Parsing

- JSON stands for **J**ava**S**cript **O**bject **N**otation. JSON is extremely lightweight data-interchange format for data exchange between server and client which is quick and easy to parse and generate.
- Like XML, JSON is also a text-based format that's easy to write and easy to understand for both humans and computers, but unlike XML, JSON data structures occupy less bandwidth than their XML versions. JSON is based on two basic structures:

- Object: This is defined as an unordered collection of key/value pairs (i.e. key:value). Each object begins with a left curly bracket { and ends with a right curly bracket }. Multiple key/value pairs are separated by a comma ,.
- Array: This is defined as an ordered list of values. An array begins with a left bracket [and ends with a right bracket]. Values are separated by a comma ,.

Object

```
{
    "book": {
        "name": "Harry Potter and the Goblet of Fire",
        "author": "J. K. Rowling",
        "year": 2000,
        "genre": "Fantasy Fiction",
        "bestseller": true
    }
}
```

Array

```
{
    "fruits": [
        "Apple",
        "Banana",
        "Strawberry",
        "Mango"
    ]
}
```

Parsing JSON Data in JavaScript

- In JavaScript, you can easily parse JSON data received from the web server using the JSON.parse() method.
- This method parses a JSON string and constructs the JavaScript value or object described by the string. If the given string is not valid JSON, you will get a syntax error.

Example

```
// Store JSON data in a JS variable
var json = '{"name": “Akash", "age": 30, "country": “India"}';

// Converting JSON-encoded string to JS object
var obj = JSON.parse(json);

// Accessing individual value from JS object
alert(obj.name); // Outputs: Akash
alert(obj.age); // Outputs: 30
alert(obj.country); // Outputs: India
```

Encoding Data as JSON in JavaScript

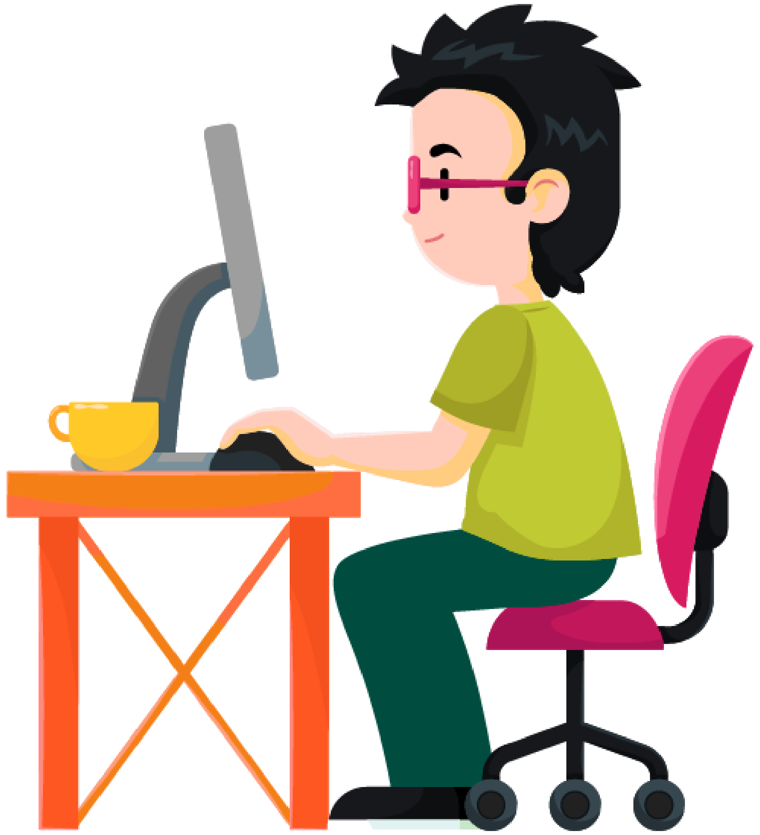
- Sometimes JavaScript object or value from your code need to be transferred to the server during an Ajax communication.
- JavaScript provides JSON.stringify() method for this purpose which converts a JavaScript value to a JSON string

```
// Sample JS object
var obj = {"name": "Akash", "age": 30, "country": "India"};

// Converting JS object to JSON string
var json = JSON.stringify(obj);
alert(json);
```

Stringify a JavaScript Array

```
// Sample JS array
var arr = ["Apple", "Banana", "Mango", "Orange",
"Papaya"];

// Converting JS array to JSON string
var json = JSON.stringify(arr);
alert(json);
```

Get Exclusive Video Tutorials



www.aptutorials.com

https://www.youtube.com/user/Akashtips

Get More Details

www.akashsir.com

If You Liked It !

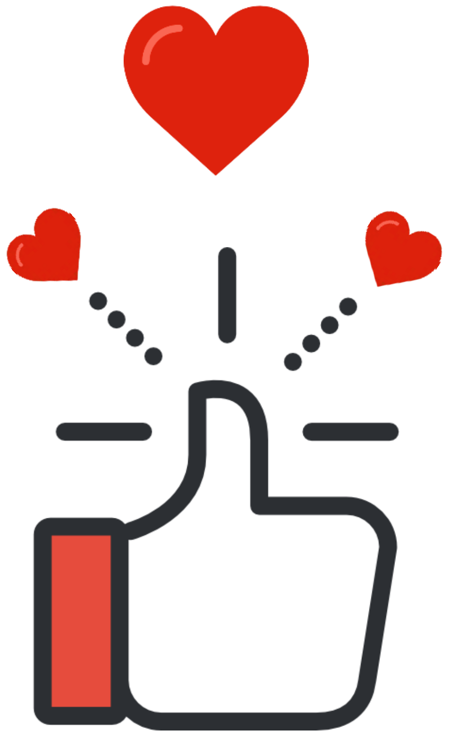
Rating Us Now

Just Dial

https://www.justdial.com/Ahmedabad/Akash-Technolabs-Navrangpura-Bus-Stop-Navrangpura/079PXX79-XX79-170615221520-S5C4_BZDET

Sulekha

https://www.sulekha.com/akash-technolabs-navrangpura-ahmedabad-contact-address/ahmedabad



Connect With Me

Akash Padhiyar
#AkashSir

www.akashsir.com
www.akashtechnolabs.com
www.akashpadhiyar.com
www.aptutorials.com

# Social Info

Akash.padhiyar

Akashpadhiyar

Akash_padhiyar

+91 99786-21654

#Akashpadhiyar
#aptutorials