JavaScript HTML DOM

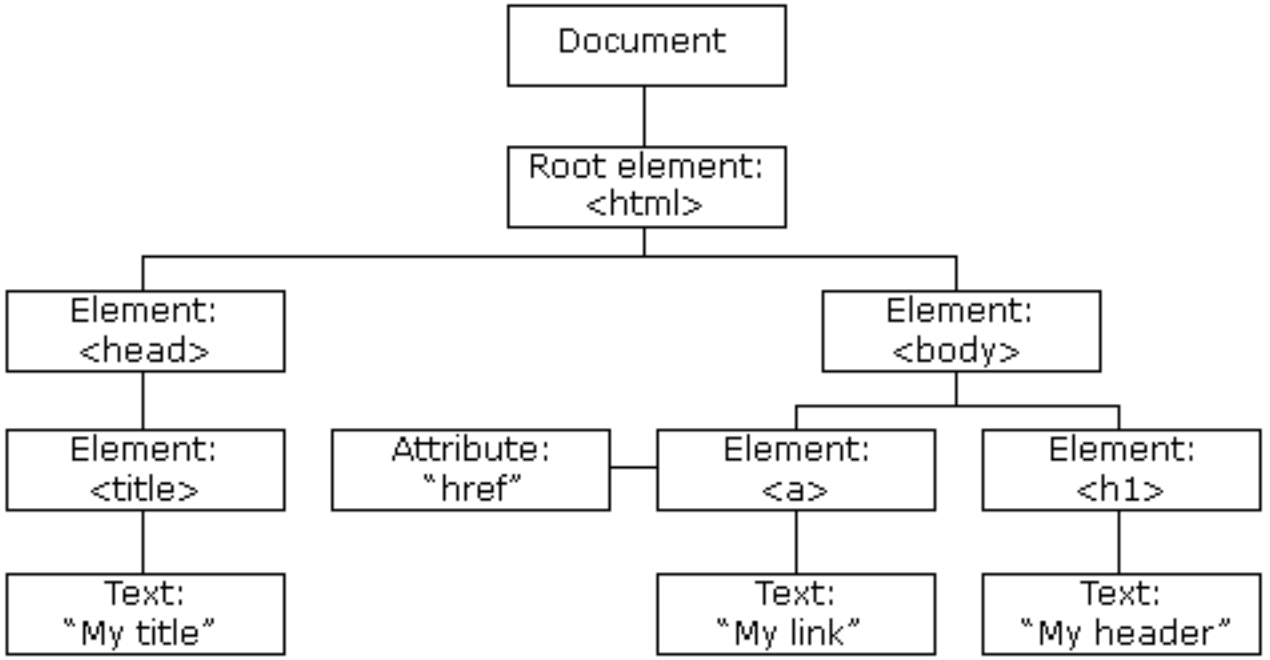
Laiba Imran

What is HTML DOM?

When a web page is loaded, the browser creates a Document Object Model of the page where each HTML element becomes an object.

With the HTML DOM, JavaScript can access and change all the elements of an HTML document.

The HTML DOM model is constructed as a tree of Objects:



What is HTML DOM?

With the object model, JavaScript gets all the power it needs to create dynamic HTML:

- JavaScript can change all the HTML elements in the page
- JavaScript can change all the HTML attributes in the page
- JavaScript can change all the CSS styles in the page
- JavaScript can remove existing HTML elements and attributes
- JavaScript can add new HTML elements and attributes
- JavaScript can react to all existing HTML events in the page
- JavaScript can create new HTML events in the page

What is DOM?

The DOM is a W3C (World Wide Web Consortium) standard.

The DOM defines a standard for accessing documents:

"The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."

HTML DOM Properties and Methods

A property is a value that you can get or set (like changing the content of an HTML element).

A method is an action you can do (like add or deleting an HTML element).

```
<html>
<body>

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

<script>
document.getElementById("demo").innerHTML = "Hello World!";
</script>

</body>
</html>
```

HTML DOM Document Object

The document object represents your web page.

If you want to access any element in an HTML page, you always start with accessing the document object.

HTML DOM Document Object

Finding HTML Elements

Method	Description
document.getElementById(*id*)	Find an element by element id
document.getElementsByTagName(*name*)	Find elements by tag name
document.getElementsByClassName(*name*)	Find elements by class name

HTML DOM Document Object

Changing HTML Elements

Property	**Description**
element.innerHTML = *new html content*	Change the inner HTML of an element
element.attribute = *new value*	Change the attribute value of an HTML element
element.style.*property* = *new style*	Change the style of an HTML element
Method	**Description**
element.setAttribute(*attribute, value*)	Change the attribute value of an HTML element

HTML DOM Document Object

Adding and Deleting Elements

Method	Description
document.createElement(*element*)	Create an HTML element
document.removeChild(*element*)	Remove an HTML element
document.appendChild(*element*)	Add an HTML element
document.replaceChild(*new, old*)	Replace an HTML element

HTML DOM Document Object

```
<div id="container"></div>

<script>
let newParagraph = document.createElement("p");

newParagraph.textContent = "This is a new paragraph";

let container = document.getElementById("container");
container.appendChild(newParagraph);

</script>
```

HTML DOM Document Object

Adding Events Handlers

Method	Description
document.getElementById(*id*).onclick = function(){*code*}	Adding event handler code to an onclick event

HTML DOM Elements

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 Element by Id

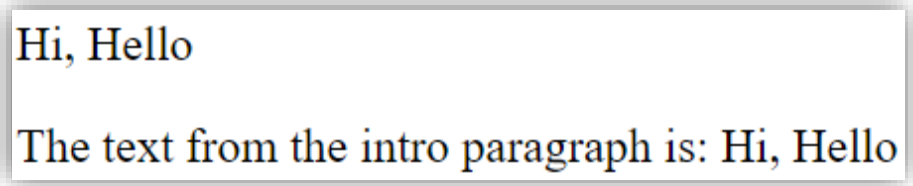
The easiest way to find an HTML element in the DOM, is by using the element id.

This example finds the element with id="intro":

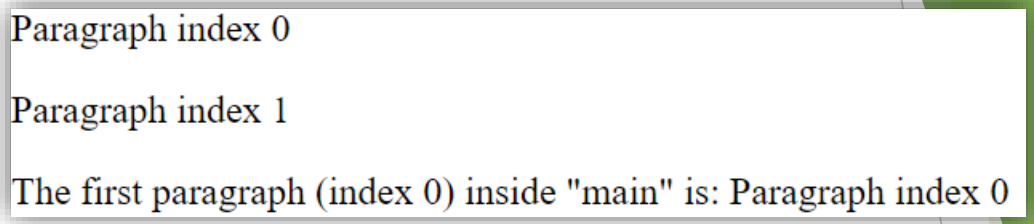
```
const element = document.getElementById("intro");
```

Finding HTML Element by Id

```
<!DOCTYPE html>
<html>
<body>
<p id="intro">Hi, Hello</p>
<p id="demo"></p>
<script>
const element = document.getElementById("intro");
document.getElementById("demo").innerHTML =
"The text from the intro paragraph is: " + element.innerHTML;
</script>
</body>
</html>
```



```
<!DOCTYPE html>
<html>
<body>
<div id="main">
<p>Paragraph index 0</p>
<p>Paragraph index 1</p>
</div>
<p id="demo"></p>
<script>
const x = document.getElementById("main");
const 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 CSS Selectors

If you want to find all HTML elements that match a specified CSS selector (id, class names, types, attributes, values of attributes, etc), use the querySelectorAll() method.

These methods allow you to search for elements within the scope of a particular DOM element, rather than the entire document.

This example returns the first matching <p> elements with class="intro".

```
const x = document.querySelector("p.intro");
```

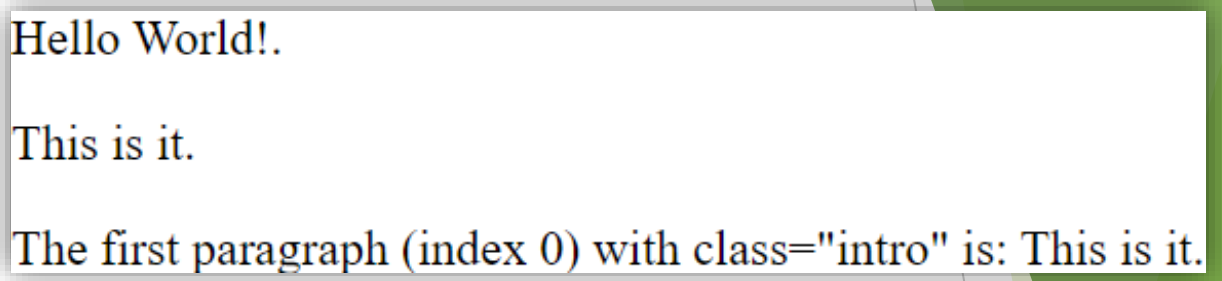
This example returns a list of all <p> elements with class="intro".

```
const x = document.querySelectorAll("p.intro");
```

Finding HTML Elements by CSS Selectors

```
const x = document.querySelector("p"); //type

const x = document.querySelector(".item"); //class

const x = document.querySelector("#main"); //id

const x = document.querySelector("[data-info]"); //attribute

const x = document.querySelector('[data-info="123"]’);  //attribute with specific value

const x = document.querySelector('div.item[data-info="123"]’);  //element with class “item” inside “main” element with having attribute value 123

const x = document.querySelector("p, .item, [data-info]"); //return the first matching element only
```

```
<!DOCTYPE html>
<html>
<body>
<p class="intro">Hello World!.</p>
<p class="intro">This is it.</p>
<p id="demo"></p>
<script>
const x = document.querySelectorAll("p.intro");
document.getElementById("demo").innerHTML =
'The first paragraph (index 0) with class="intro" is: ' + x[1].innerHTML;
</script>

</body>
</html>
```



HTML DOM - Changing HTML

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

Or

const element = document.getElementById(“id”);
element.innerHTML = New HTML;

```
<!DOCTYPE html>
<html>
<body>
<h1 id=id01>Old Heading</h1>
<script>
const element = document.getElementById(“id01”);
element.innerHTML = "New Heading";
</script>
<p>JavaScript changed "Old Heading" to "New Heading".</p>
</body>
</html>
```

Changing the Value of an Attribute

```
<!DOCTYPE html>
<html>
<body>
<img id="image" src="smiley.gif" width="160" height="120">
<script>
document.getElementById("image").src = "landscape.jpg";
</script>
<p>Smiley.gif to landscape.jpg</p>
</body>
</html>
```

Reacting to Events

A JavaScript event is an action that can be detected by JavaScript.

We say then that an event is triggered and then it can be caught by JavaScript functions, which then do something in response.

Common Event Types

- click: Triggered when an element is clicked.
- keydown: Triggered when a key is pressed on the keyboard.
- mouseover: Triggered when the mouse pointer hovers over an element.
- submit: Triggered when a form is submitted.

Reacting to Events

```
<!DOCTYPE html>
<html>
<body>
<h2>The onclick event</h2>
<p onclick="changeText(this)">Click on this text!</p>
<script>
function changeText(id) {
  id.innerHTML = "Ooops!";
}
</script>
</body>
</html>
```

Mouse Events

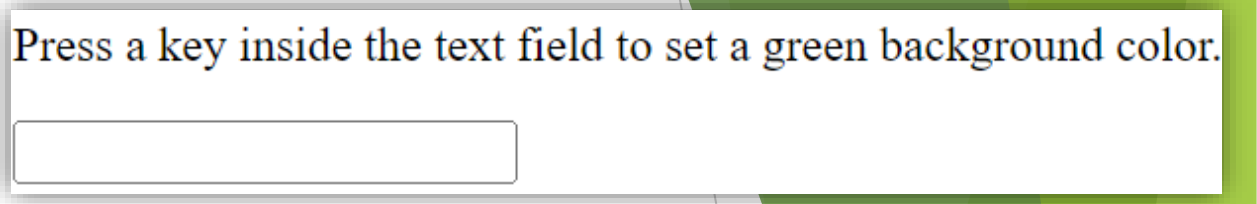
Event	Description
onclick	The mouse was clicked on an element
ondblclick	The mouse was double clicked on an element
onmousedown	The mouse was pressed down over an element
onmouseup	The mouse was released over an element
onmouseover	The mouse was moved (not clicked) over an element
onmouseout	The mouse was moved off of an element
onmousemove	The mouse was moved while over an element

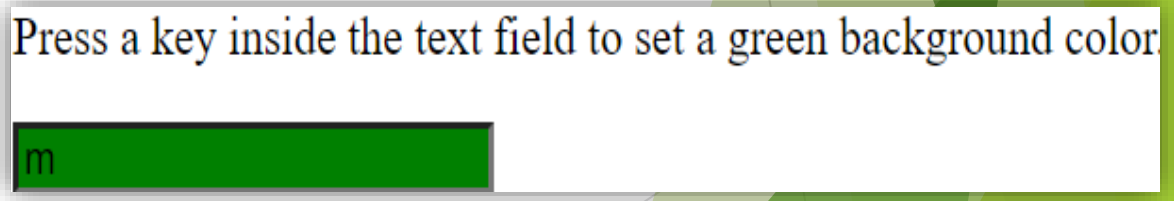
```
<!DOCTYPE html>
<html>
<body>
<h2>The onmouseover Attribute</h2>
<div onmouseover="mOver(this)" onmouseout="mOut(this)“ style="background-color:green ; color:white; width:120px;height:10px;padding:40px;">
Mouse Over Me</div>
<script>
function mOver(obj) {
  obj.innerHTML = "Thank You“ }
function mOut(obj) {
  obj.innerHTML = "Mouse Over Me“ }
</script>
</body>
</html>
```

Keyboard Events

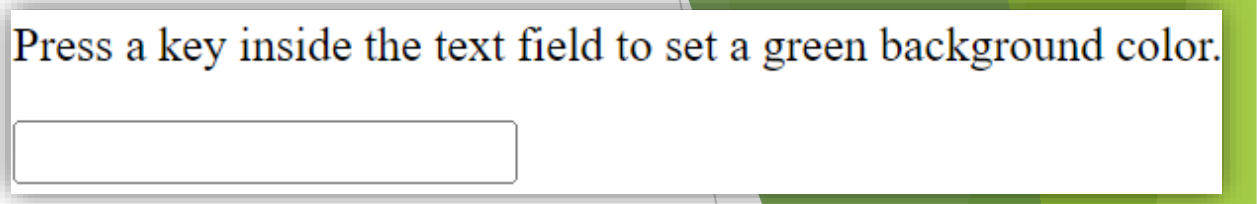
Event	Description
onkeydown	The user is pressing a key (this happens first)
onkeypress	The user presses a key
onkeyup	The user releases a key that was down (this happens last)

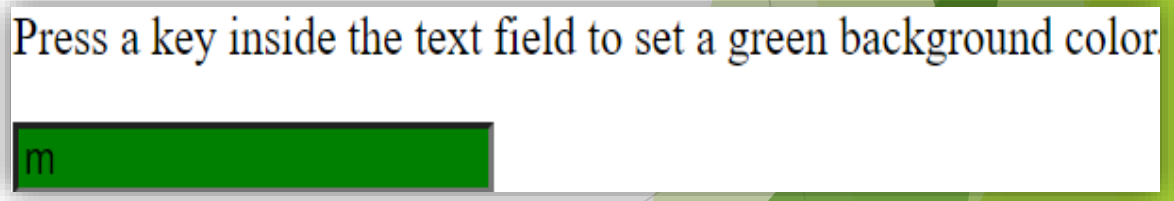
```
<!DOCTYPE html>
<html>
<body>
<p>Press a key inside the text field to set a green background color.</p>
<input type="text" id="demo" onkeydown="myFunction()">
<script>
function myFunction() {
  document.getElementById("demo").style.backgroundColor = "green";
}
</script>
</body>
</html>
```



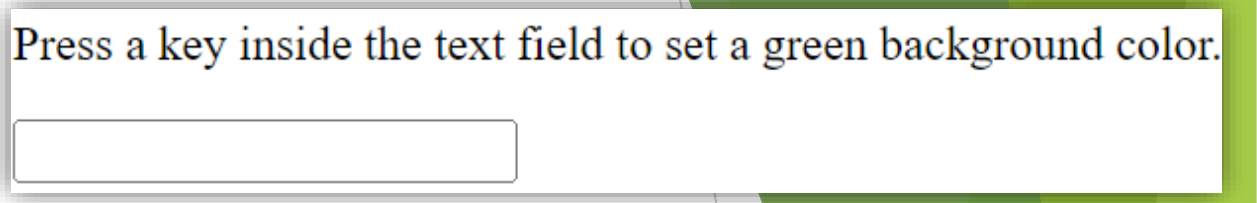


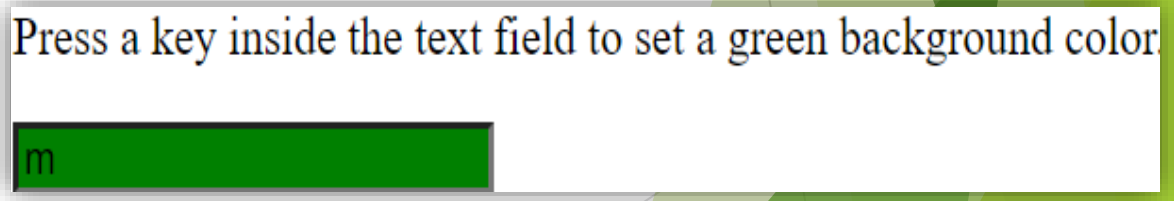
```
<!DOCTYPE html>
<html>
<body>
<p>Press a key inside the text field to set a green background color.</p>
<input type="text" id="demo">
<script>
document.getElementById("demo").onkeypress = function() {myFunction()};
function myFunction() {
  document.getElementById("demo").style.backgroundColor = “green";
}
</script>
</body>
</html>
```





```
<!DOCTYPE html>
<html>
<body>
<p>Press a key inside the text field to set a green background color.</p>
<input type="text" id="demo">
<script>
document.getElementById("demo").addEventListener("keypress", myFunction);
function myFunction() {
  document.getElementById("demo").style.backgroundColor = "green";
}
</script>
</body>
</html>
```

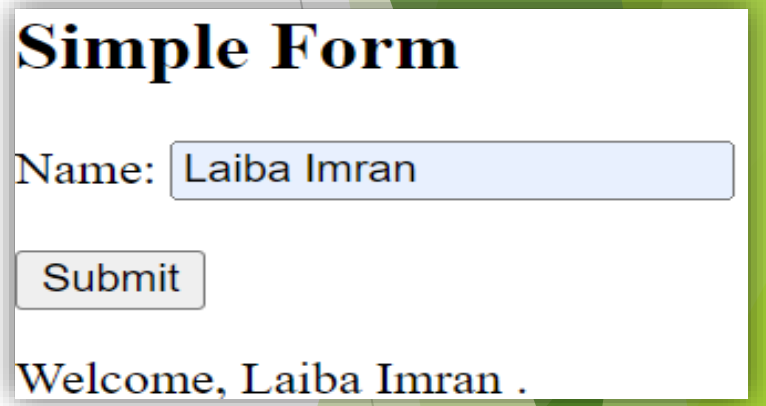




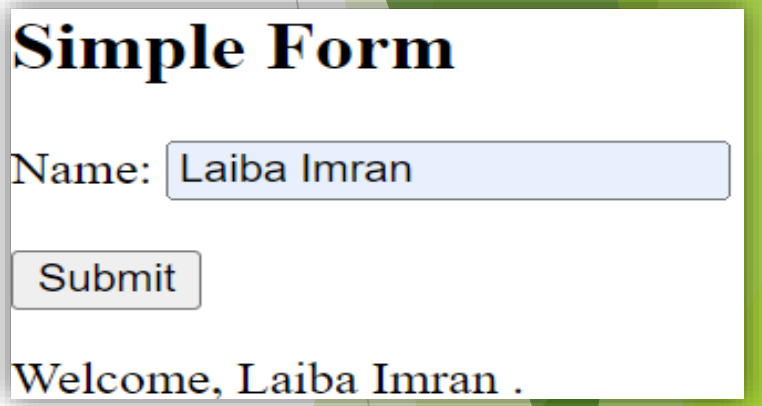
Form Events

Event	Description
onchange	Some <input>, <textarea> or <select> field had their value change. This could mean the user typed something, or selected a new choice.
onfocus	Complementing the onblur event, this is triggered when an element gets focus (the user clicks in the field or tabs to it)
onreset	HTML forms have the ability to be reset. This event is triggered when that happens.
onselect	When the users selects some text. This is often used to try and prevent copy/paste.
onsubmit	When the form is submitted this event is triggered. We can do some pre-validation when the user submits the form in JavaScript before sending the data on to the server.

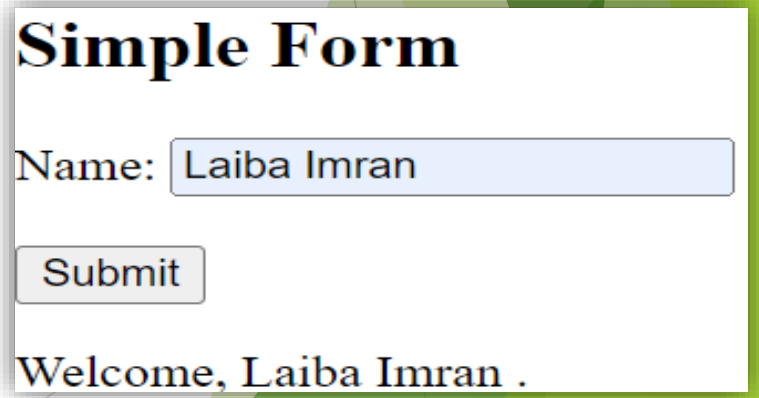
```
<h2>Simple Form</h2>
<form id="simpleForm" onsubmit="handleSubmit(event)">
  <label for="name">Name:</label>
  <input type="text" id="name" placeholder="Enter your name" required><br>
  <button type="submit">Submit</button>
</form>
<p id="message"></p>
<script>
  function handleSubmit(event) {
    event.preventDefault(); // Prevent the default form submission behavior
    let name = document.getElementById('name').value;
    document.getElementById('message').textContent = `Welcome, ${name}.`;
  }
</script>
```



```
<h2>Simple Form</h2>
<form id="simpleForm">
  <label for="name">Name:</label>
  <input type="text" id="name" placeholder="Enter your name" required><br>
  <button type="submit">Submit</button>
</form>
<p id="message"></p>
<script>
  let form = document.getElementById('simpleForm');
  let message = document.getElementById('message');
  form.addEventListener('submit', function(event) {
    event.preventDefault(); // Prevent the default form submission behavior
    let name = document.getElementById('name').value;
    message.textContent = `Welcome, ${name}.`;}
  ); </script>
```



```
<h2>Simple Form</h2>
<form id="simpleForm">
  <label for="name">Name:</label>
  <input type="text" id="name" placeholder="Enter your name"><br>
  <button type="submit">Submit</button>
</form>
<p id="message"></p>
<script>
  let form = document.getElementById('simpleForm');
  let message = document.getElementById('message');
  form.addEventListener('submit', function(event) {
    event.preventDefault();
    let name = document.getElementById('name').value;
    if (name === "") { message.textContent = "Please enter your name!";}
    else { message.textContent = `Welcome, ${name}.`;} });
</script>
```



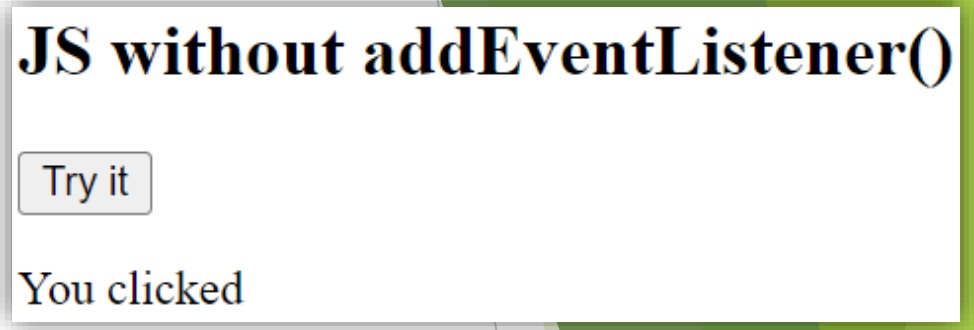
Event Listener

The addEventListener() method is used to attach an event handler to a particular element.
It does not override the existing event handlers.
It separates JS from HTML.
The addEventListener() method is an inbuilt function of JavaScript.
We can add multiple event handlers to a particular element without overwriting the existing event handlers.

```
element.addEventListener(event, function);
```

The removeEventListener() method removes event handlers that have been attached with the addEventListener() method:

```
element.removeEventListener(event, function);
```




```
<h2>JS without addEventListener()</h2>
<button id="btn">Try it</button>
<p id="demo"></p>
<script>
document.getElementById("btn").onclick = function() {changeColor()};
document.getElementById("btn").onclick = function() {addContent()};
function changeColor() {
  document.getElementById("demo").style.color = "green";
}
function addContent() {
  document.getElementById("demo").innerHTML = "You clicked";
}
</script>
```

```
<h2>JS addEventListener()</h2>
<button id="btn">Try it</button>
<p id="demo"></p>
<script>
document.getElementById("btn").addEventListener("click", changeColor);
document.getElementById("btn").addEventListener("click", addContent);
function changeColor() {
  document.getElementById("demo").style.color = "green";
}
function addContent() {
  document.getElementById("demo").innerHTML = "You clicked";
}
</script>
```