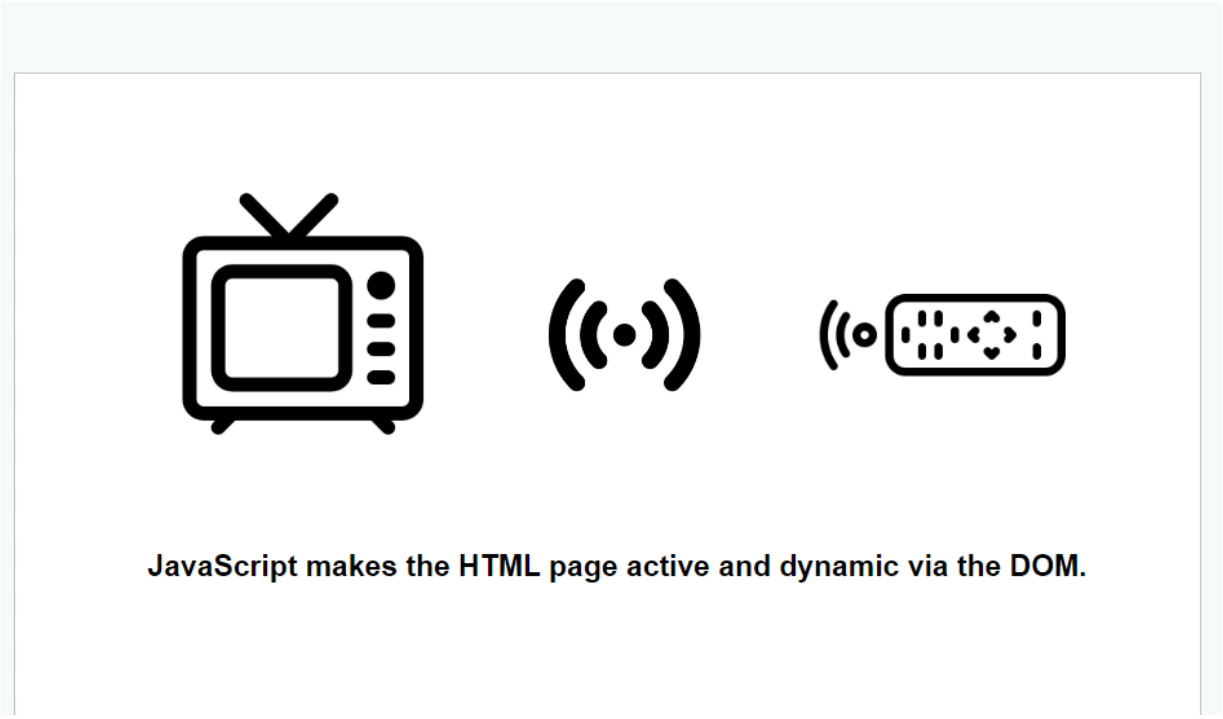
DOM

Document Object Model

A standard for how to get, change, add, or delete HTML elements.

Let take an example to understand better.

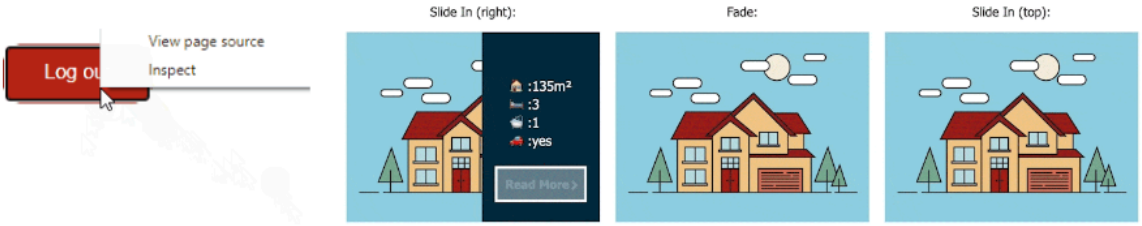


Remote is a bridge that helps us to interact with the TV.

Js doesn't doesn't do much with itself like calculation, string manipulation, to make HTML document more interactive, the script also needs to be able to access the content of the document. When the user is interacting with it. It is done by communicating with the browser using events, property, and method which is called document object model or **DOM.**

You want a button to change colours when it gets clicked or an image to appear when the mouse hovers over text.

First, you need to reference those elements from your JavaScript.



What is DOM?

- A programming interface for web documents.
- DOM is not a programming language(language independent).
- Represents the page so that programs can change the document structure, style, and content.
- The DOM is a tree-like representation of the web page that gets loaded into the browser.
- The DOM represents the document as nodes and objects.
- Without it, the JavaScript language wouldn't have any model or notion of web pages, HTML documents, SVG documents, and their component parts.
- Web API is used to build websites.

Accessing the DOM

When you create a script, whether inline in a <script> element or included in the web page, you can immediately begin using the API for the document or window objects to manipulate the document itself.

The DOM was designed to be independent of any particular programming language, making the structural representation of the document available from a single, consistent API.

```
<html lang="en">
  <head>
    <script>
      // run this function when the document is loaded
      window.onload = () => {
        // create a couple of elements in an otherwise empty
HTML page
        const heading = document.createElement("h1");
        const headingText = document.createTextNode("Big
Head!");
        heading.appendChild(headingText);
        document.body.appendChild(heading);
      };
    </script>
  </head>
  <body></body>
</html>
```

As we load the window we want to create h1 heading and append the heading in h1.

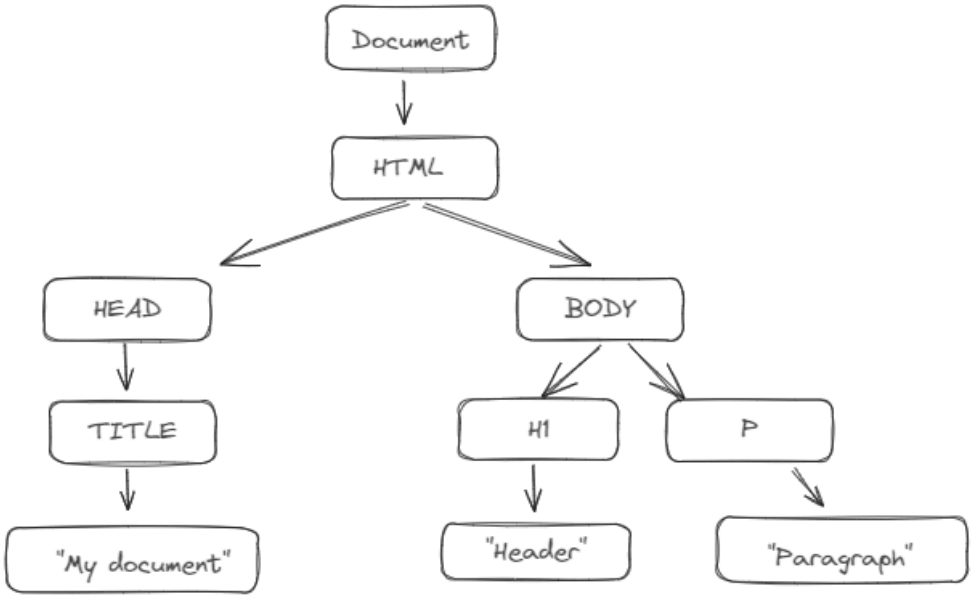
DOM represents the web page using a series of objects. The main object is the document object, which in turn houses other objects which also house their own objects, and so on.

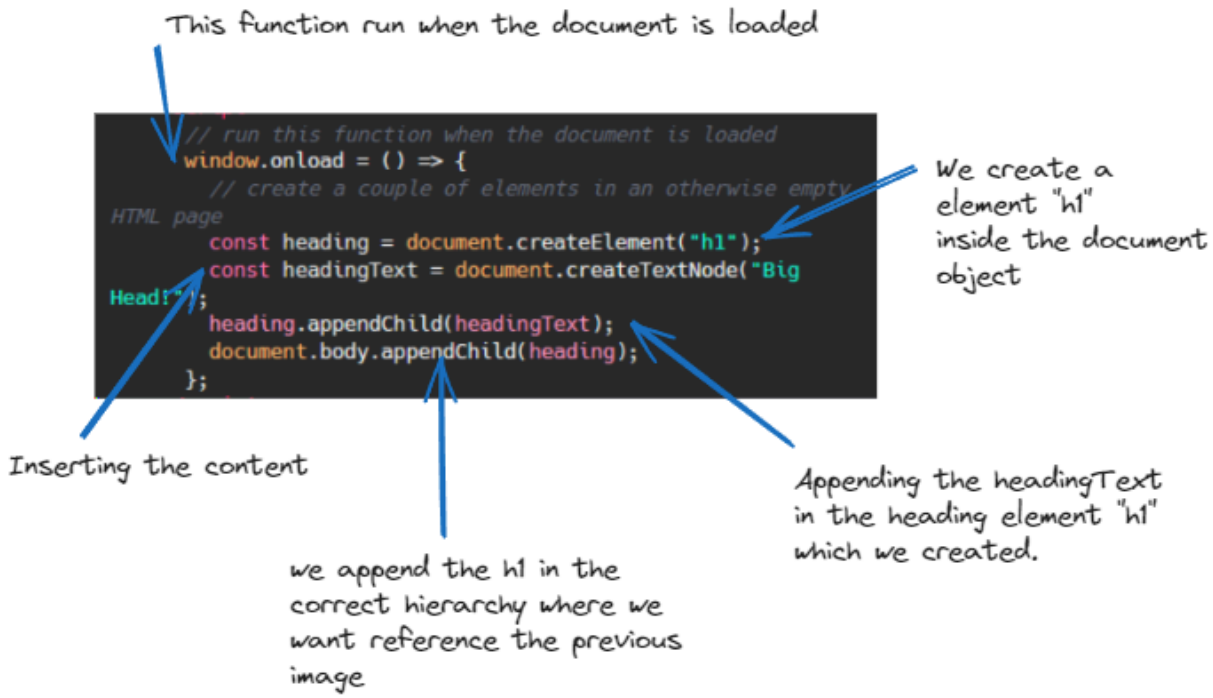
Let see an example

```html
<html lang="en">
  <head>
    <title>My Document</title>
  </head>
  <body>
    <h1>Header</h1>
    <p>Paragraph</p>
  </body>
</html>
```

DOM Tree

The DOM is a tree like representation of the web page that gets loaded into the browser.



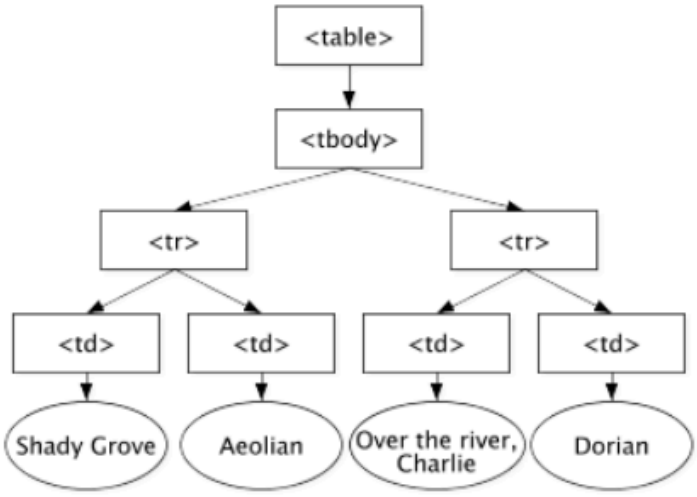


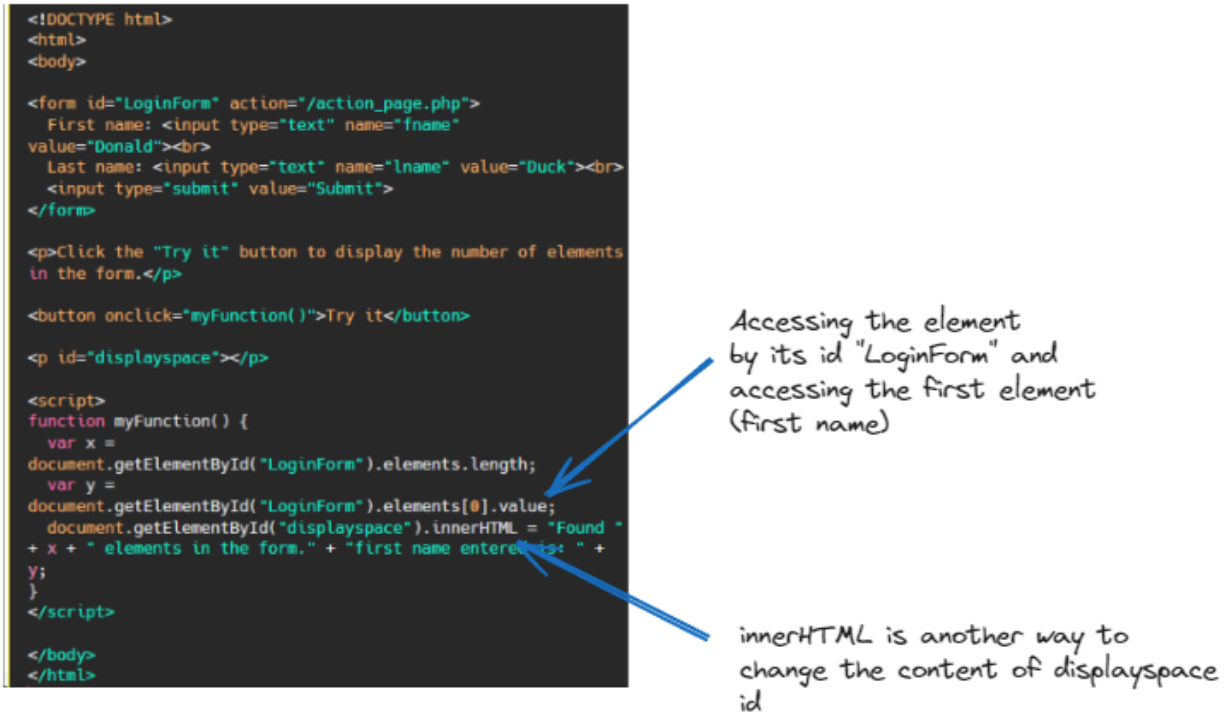
When a web browser parses an HTML document, it builds a DOM tree and then uses it to display the document.

Let's look at the different objects

The Document Object

This is the top most object in the DOM. It has properties and methods which you can use to get information about the document using a rule knows as dot notation.



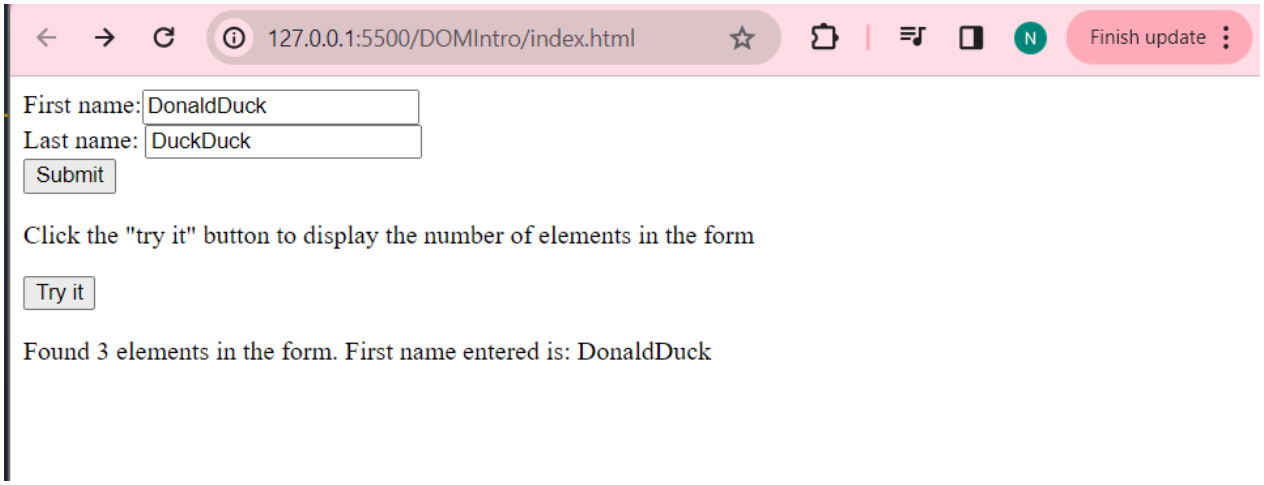


Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <form id="LoginForm" action="/action_page.php">
        First name:<input type="text" name="fname" value="DonaldDuck"><br>
        Last name: <input type="text" name="lname" value="DuckDuck"><br>
        <input type="submit" value="Submit">
    </form>

    <p>Click the "try it" button to display the number of elements in the
form</p>

    <button onclick="myFunction()">Try it</button>
    <p id="displayspace"></p>
```

```
    <script>
        function myFunction(){
            var x = document.getElementById("LoginForm").elements.length;
            var y =
document.getElementById("LoginForm").elements[0].value;
            document.getElementById("displayspace").innerHTML= "Found
"+x+" elements in the form."+" First name entered is: "+y;
        }
    </script>
</body>
</html>
```

Webpage:



We can also access lastname.

The createElement() method:

Create a specified element and insert it into the DOM:

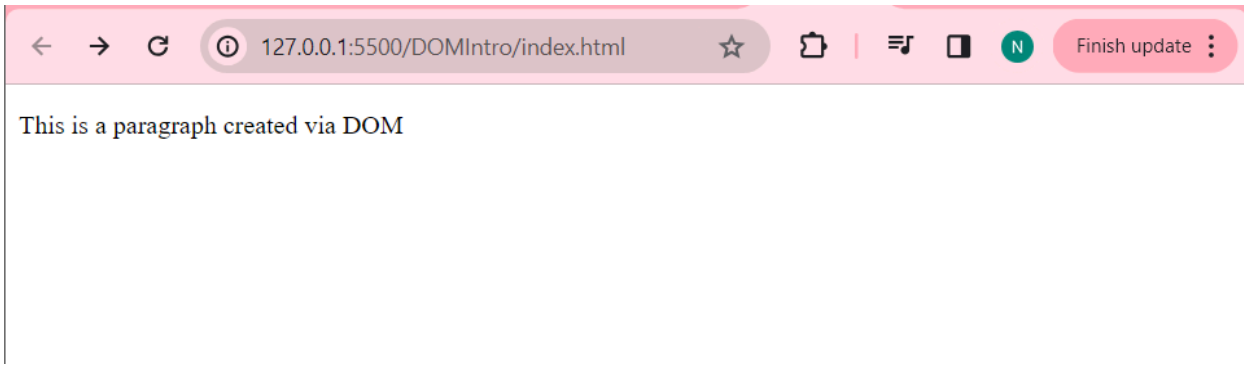
```
const para = document.createElement("p");
para.innerText = "This is a paragraph created via DOM.";
document.body.appendChild(para);
```

innerText : what info we want to display in the newly created element.

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>

    <script>
        const para = document.createElement("p")
        para.innerText = "This is a paragraph created via DOM";
        document.body.appendChild(para);
    </script>
</body>
</html>
```

Webpage:



This can be very helpful when we want to display error messages etc only when certain user action has happened.

Finding HTML Elements

If you want to manipulate HTML elements.

First we need to find the elements. Ways to do this are:

- 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.

getElementById()

Method to get an element from the document by its unique id attribute.

```
<div id="first"> Div one </div>
<p> Paragraph one </p>
<div> Div two </div>
<p> Paragraph two </p>
<div> Another div </div>

<script>
    var firstDiv = getElementById("first")
</script>
```

getElementsByTagname()

Method to access one more html elements by their tagname.

```
<div> Div one </div>
<p> Paragraph one </p>
<div> Div two </div>
<p> Paragraph Two </p>
<div> Another div </div>
```

```
    <script>
        divs = getElementByTagname("div")
    </script>
```

getElementByClassName()

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h2>Javascript HTML DOM</h2>
    <p>Finding HTML Elements by Class Name</p>
    <p class="intro">Hello World! My name is Nishant Thapa </p>
    <p class="intro">This example demonstrates the
<b>getElementsByClassName </b>Method</p>
    <p id="demo"></p>

    <script>
        const x = document.getElementsByClassName("intro");
        document.getElementById("demo").innerHTML = "The first paragraph
(Index 0) with class ='intro' is: " + x[0].innerHTML;
    </script>
</body>
</html>
```

Webpage:

There will be error if we try to access out of bound , it is better to iterate over loop or decide the range.

Finding HTML Elements by CSS Selectors: querySelectorAll()

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

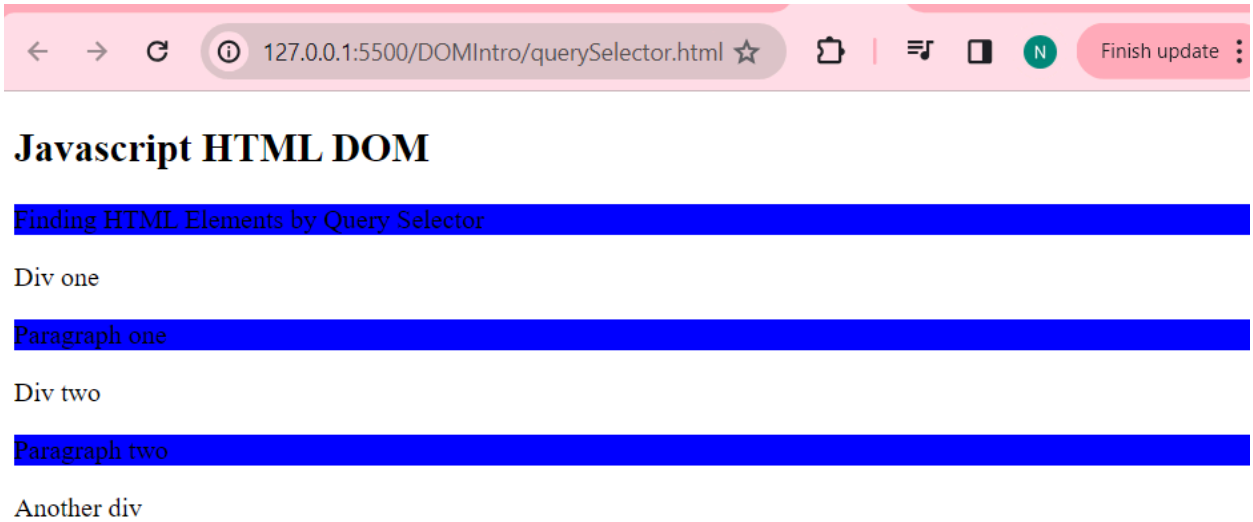
querySelectorAll() will give all the occurrences.
While the querySelector() will give only the first occurrence of the element.

Lets see an example

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h2>Javascript HTML DOM</h2>
    <p> Finding HTML Elements by Query Selector </p>

    <div> Div one </div>
    <p class="intro" > Paragraph one </p>
```

```
    <div> Div two </div>
    <p class="subheading"> Paragraph two </p>
    <div> Another div </div>

    <script>
        var paragraph = document.querySelectorAll('p');
        paragraph.forEach(paragraph => paragraph.style.backgroundColor =
"blue")
    </script>
</body>
</html>
```

Webpage:

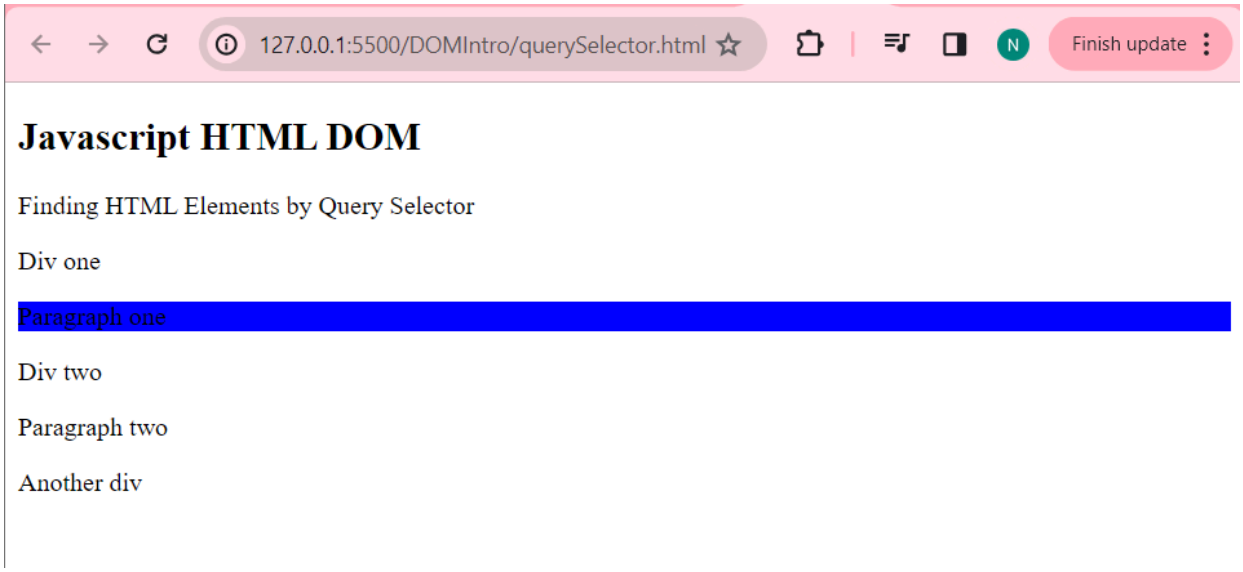


paragraph will return a node list

We can specify it further

```
    <script>
        var paragraph = document.querySelectorAll('p.intro');
        paragraph.forEach(paragraph => paragraph.style.backgroundColor =
"blue")
    </script>
```

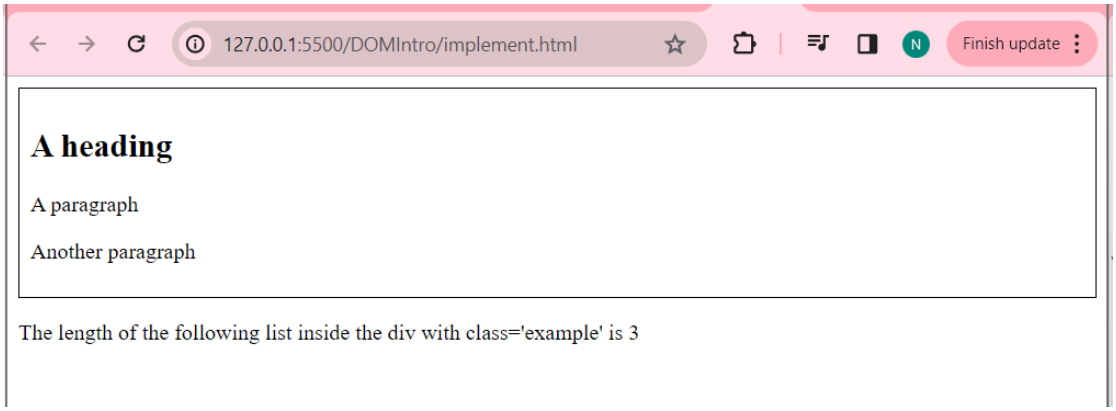
Selecting the paragraph with class as intro .



Let see another example:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <div id="myDIV" style="border:1px solid black;padding: 8px;">
        <h2 class="example">A heading</h2>
        <p class="example">A paragraph</p>
        <p class="example">Another paragraph</p>
    </div>
    <p id="demo"></p>

    <script>
        const element = document.getElementById('myDIV')
        const list = element.querySelectorAll('.example')
        document.getElementById("demo").innerHTML ="The length of the
following list inside the div with id='myDIV' is " + list.length;
    </script>
</body>
</html>
```

Output:



To read more about topic

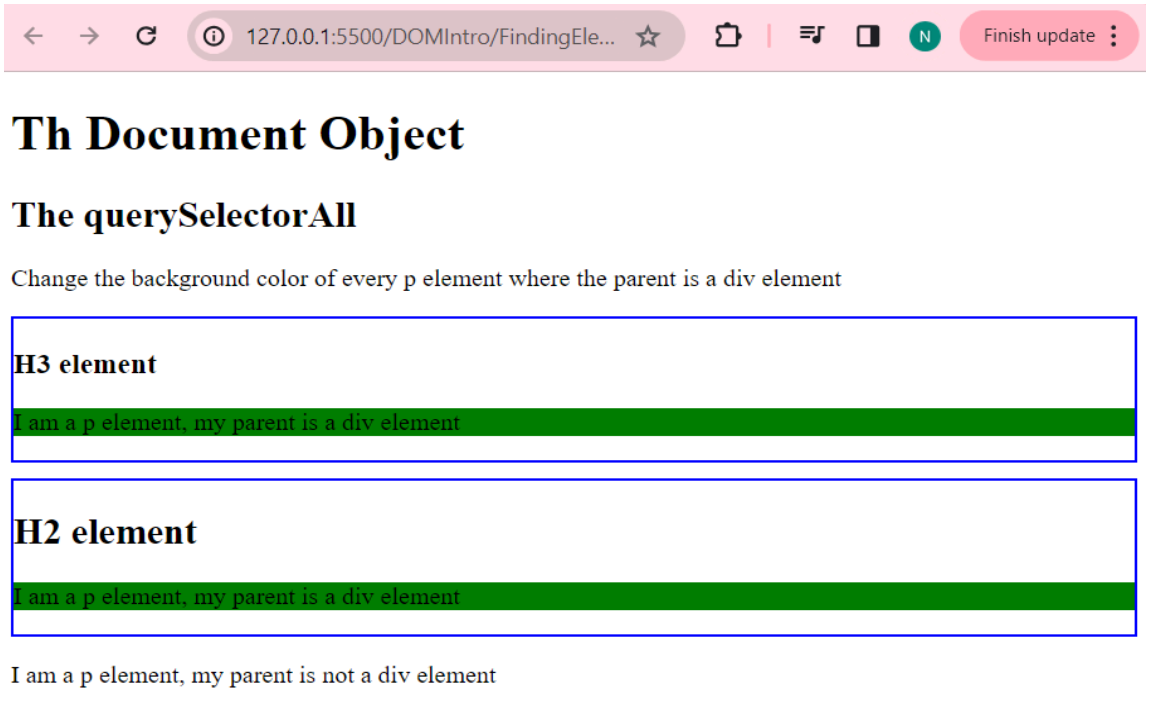
https://www.w3schools.com/cssref/css_selectors.php

Let see more examples:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <style>
        div{
            border: 2px solid blue;
            margin-bottom: 10px;
        }
    </style>
</head>
<body>
    <h1>Th Document Object</h1>
    <h2>The querySelectorAll</h2>

    <p>Change the background color of every p element where the parent is
a div element</p>
    <div>
        <h3>H3 element</h3>
        <p>I am a p element, my parent is a div element</p>
```

```
    </div>

    <div>
        <h2>H2 element</h2>
        <p>I am a p element, my parent is a div element</p>
    </div>

    <p>
        I am a p element, my parent is not a div element
    </p>

    <script>
        const nodeList = document.querySelectorAll('div > p')
        for(let i =0; i< nodeList.length; i++){
            nodeList[i].style.backgroundColor = "green";
        }
    </script>
</body>
</html>
```

Webpage:



If we want only the first occurrence we use querySelector

Finding HTML Elements by HTML Object Collection

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h2>Javascript HTML DOM</h2>
    <p>Finding HTML Elements Using <b>document.forms</b></p>

    <form id="frm1" action="/action_page.php">
        First name: <input type="text" name="fname" value="Nishu"><br>
        Last name: <input type="text" name="lname" value="Thapa"><br><br>
        <input type="submit" value="Submit">
    </form>
    <p>These are the values of each element in the form:</p>
    <p id="demo"></p>

    <script>
        const x = document.forms("frm1");
        let text = "";
        for(let i =0; i< x.length; i++){
            text += x.elements[i].value + "<br>";
        }
        document.getElementById("demo").innerHTML = text;
    </script>
    <!-- live list (updation) we will use html collection -->
</body>
</html>
```

Difference between a HTMLCollection and a NodeList

HTML collection	NodeList
getElementsByClassName() and getElementsByTagName() methods return a live HTMLCollection.	querySelectorAll() method returns a static NodeList.
A collection of document elements.	A collection of document nodes (element nodes, attribute nodes, and text nodes).
Items can be accessed by their name, id, or index number.	Items can only be accessed by their index number.
It is always a live collection. Example: If you add a element to a list in the DOM, the list in the HTMLCollection will also change.	Most often a static collection. Example: If you add a element to a list in the DOM, the list in NodeList will not change.

Main difference is Live and static collection.‘

Different object collection other than forms which can be use are

- document.anchors
- document.body
- document.documentElement
- document.embeds
- document.forms
- Document.head
- Document.images
- Document.links
- Document.scripts
- document.title

Examples

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h2>Finding HTML Elements Using document.anchors</h2>
    <a name="html">Html tutorial</a>
    <a name="css">Css tutorial</a>
    <a name="xml">XML tutorial</a>

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

    <script>
        document.getElementById("demo").innerHTML = "Number of anchors
are: " + document.anchors.length;
    </script>
</body>
</html>
```

Webpage:

Another example:

Replicating the body of existing html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
```

```
</head>
<body>
    <h2>Javascript HTML DOM</h2>
    <p>Displaying document.body</p>

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

Webpage:

Changing HTML Elements

- innerHTML property

Example: Changing .gif to .jpg

Changing the Value of an Attribute

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

<h2>JavaScript HTML DOM example</h2>
<img id="image" src="smiley.gif" width="160" height="120">

<p>The original image was a .gif, but the script changed it to
an image with .jpg extension</p>

</body>
</html>
```

JavaScript HTML DOM example

The original image was a .gif, but the script changed it to an image with .jpg extension

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

<h2>JavaScript HTML DOM example</h2>
<img id="image" src="smiley.gif" width="160" height="120">

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

<p>The original image was a .gif, but the script changed it to
an image with .jpg extension</p>

</body>
</html>
```

JavaScript HTML DOM example

The original image was a .gif, but the script changed it to an image with .jpg extension

Dynamic HTML content

document.write() : write directly to the HTML output stream.

Never use document.write() after the document is loaded. It will overwrite the document.

The setAttribute() method sets a new value to an attribute.

If the attribute does not exist, it is created first.

element.setAttribute(“style”,”background-color:red”);

vs

element.style.backgroundColor = “red”;

(Better)

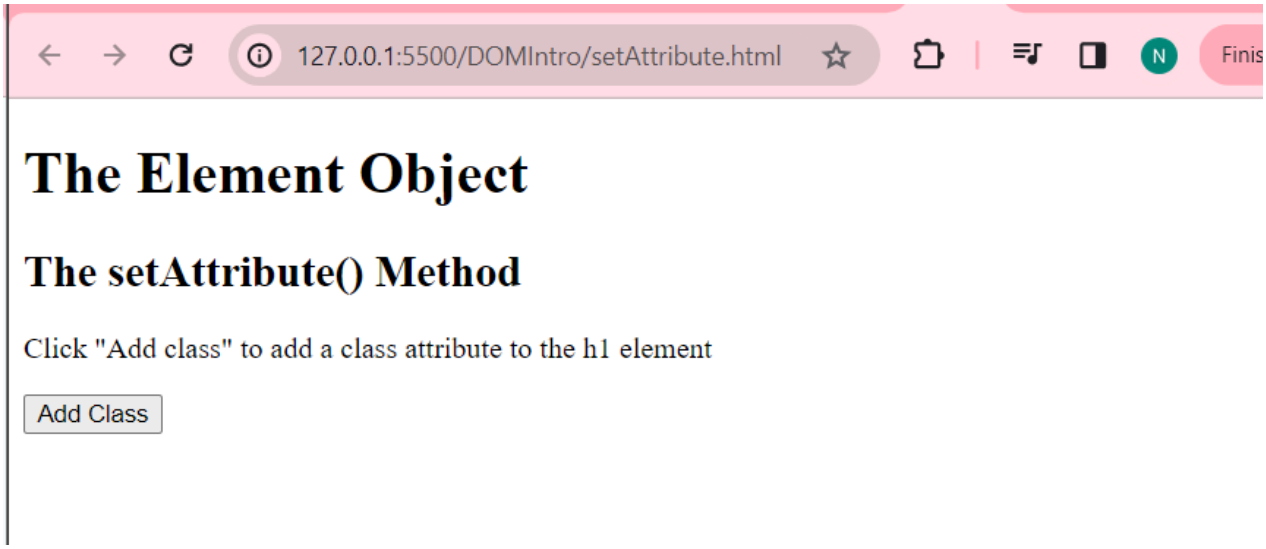
Lets see an example where we will add a class to the element

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <style>
        .democlass{
            color: red;
        }
    </style>
</head>
<body>
    <h1 id="myH1">The Element Object</h1>
    <h2>The setAttribute() Method</h2>

    <p>Click "Add class" to add a class attribute to the h1 element</p>

    <button onclick="myFunction()">Add Class</button>
    <script>
        function myFunction(){

document.getElementById("myH1").setAttribute("class","democlass");
        }
    </script>
```

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

Webpage :

Before Clicking Add Class button

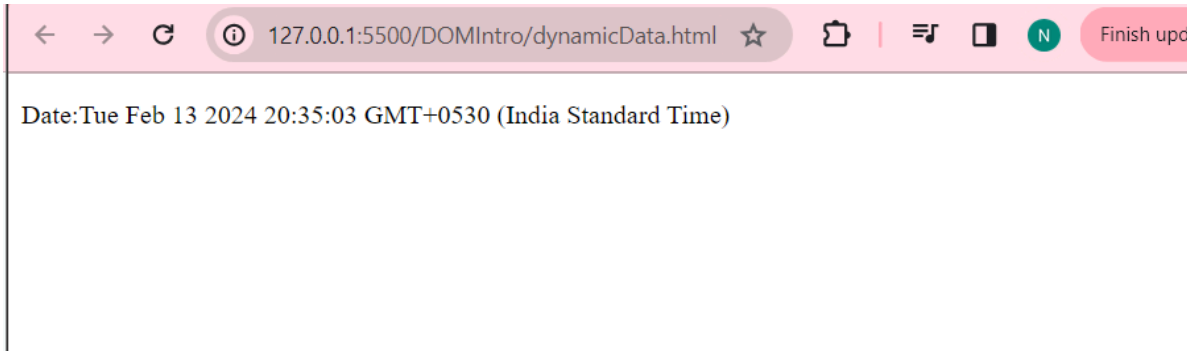


After clicking the button:

Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <p id="demo"></p>
    <script>
        document.getElementById("demo").innerHTML = "Date:"+ Date();
    </script>
</body>
</html>
```

Webpage:



Now suppose if I want to write directly to the output stream:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <p>Bla, Bla, Bla</p>
<!-- The positioning of this script tag is very important -->
    <script>
```

```
        document.write(Date());
    </script>

    <p>Bla, bla</p>
</body>
</html>
```

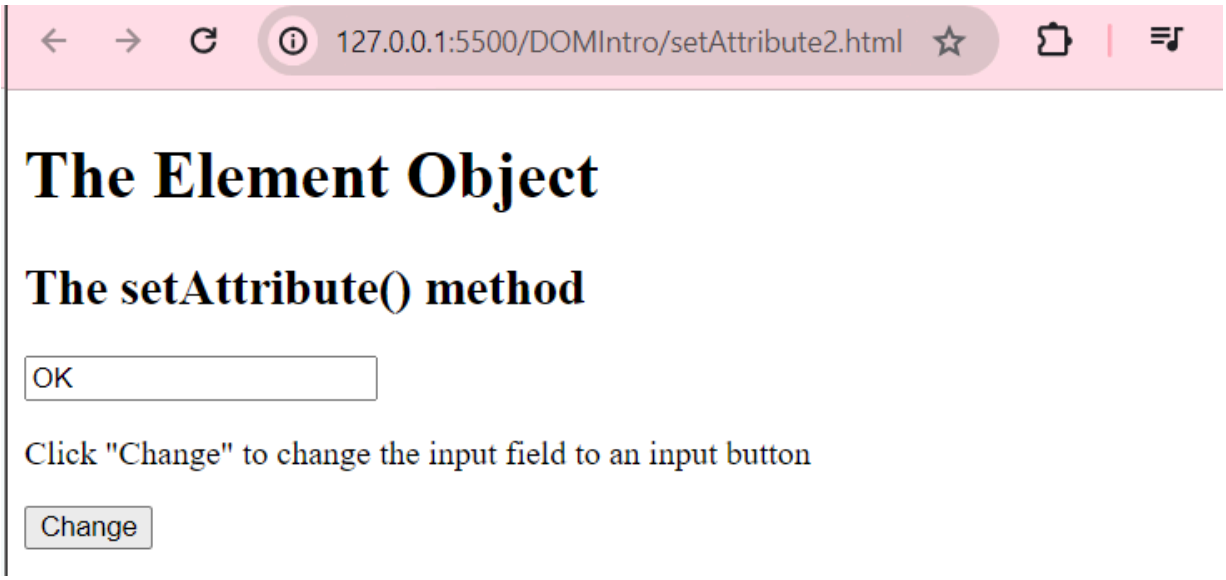
Example: Updating the type

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h1>The Element Object</h1>
    <h2>The setAttribute() method</h2>

    <input value="OK" id="myInput">
    <p>Click "Change" to change the input field to an input button</p>
    <button onclick="myFunction()">
    Change</button>

    <script>
        function myFunction(){

document.getElementById("myInput").setAttribute("type","button");
        }
```

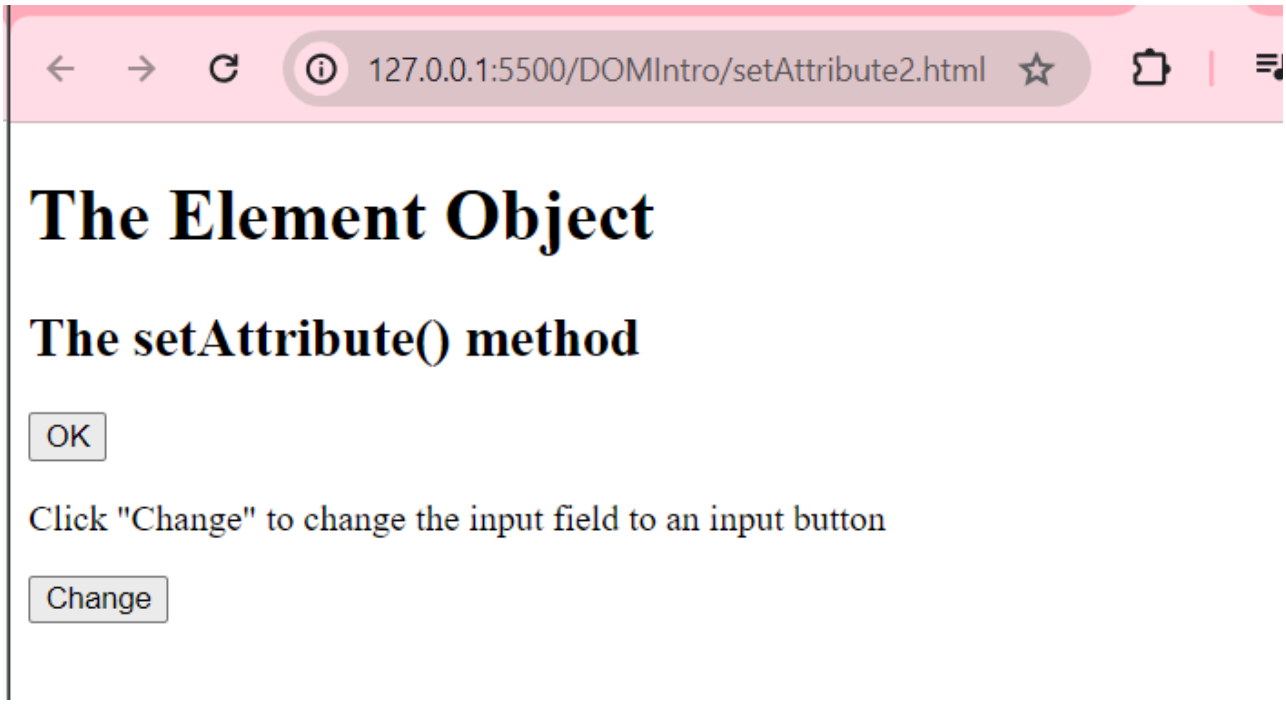
```
    </script>
</body>
</html>
```

Webpage:

Before clicking the Change button



After clicking the Change button



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
document.write(*text*)	Write into the HTML output stream

createElement()

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h1>DOM</h1>
    <h2>createElement()</h2>

    <p>Create a p element and append it to "myDiv"</p>

    <div id="myDiv" style="padding: 16px;background-color: red;">
    <h3>A DIV element</h3>
    </div>

    <script>
        const para = document.createElement("p");
        para.innerHTML = "This is a paragraph";
```

```
        //Append to another element;
        document.getElementById("myDiv").appendChild(para)
    </script>
</body>
</html>
```

Webpage:

removeElement()

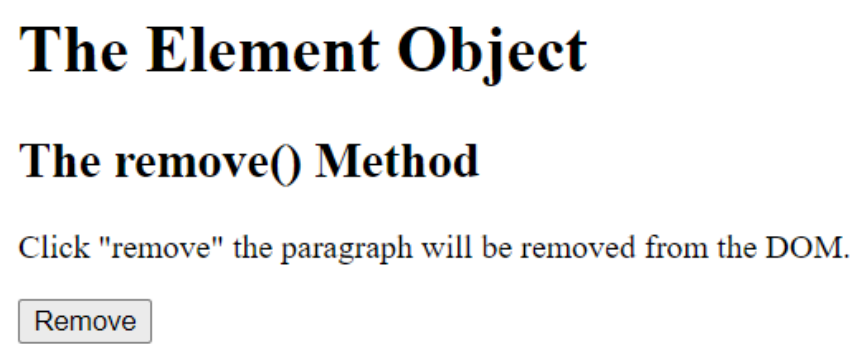
Lets see an example when we want to remove a element from our document.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h1>The Element Object</h1>
    <h2>The remove() Method</h2>
```

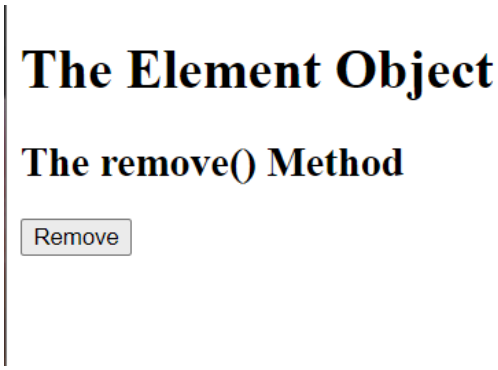
```
    <p id="demo">Click "remove" the paragraph will be removed from the
DOM.</p>
    <button onclick="myFunction()">Remove</button>

    <script>
        function myFunction(){
            const element = document.getElementById("demo");
            element.remove();
        }
    </script>
</body>
</html>
```

Webpage:

Before clicking the button



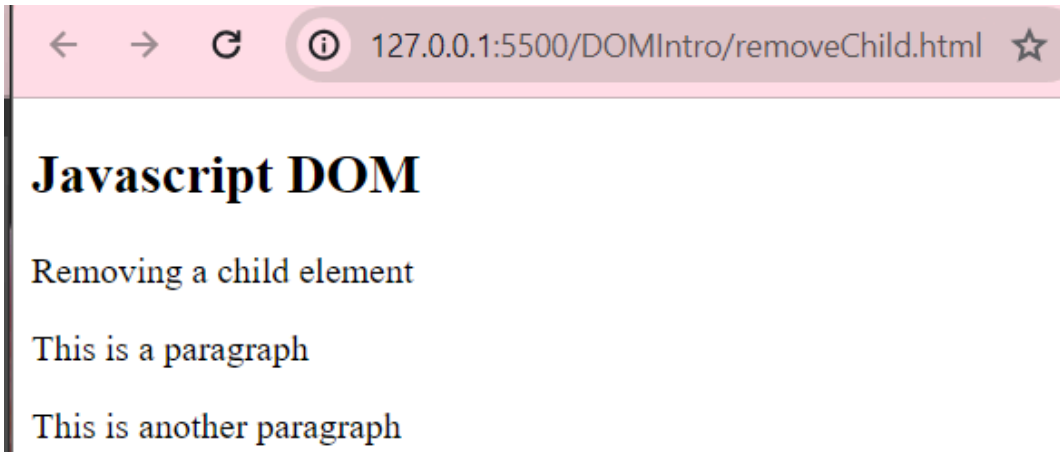
After clicking on Remove



removeChild()

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h2>Javascript DOM</h2>
    <p>Removing a child element</p>

    <div id="div1">
        <p id="p1">This is a paragraph</p>
        <p id="p2">This is another paragraph</p>
    </div>

    <script>
        const parent = document.getElementById("div1");
        const child = document.getElementById("p1");
        parent.removeChild(child);
    </script>
</body>
</html>
```

Webpage without script:



Webpage with script:

replaceChild()

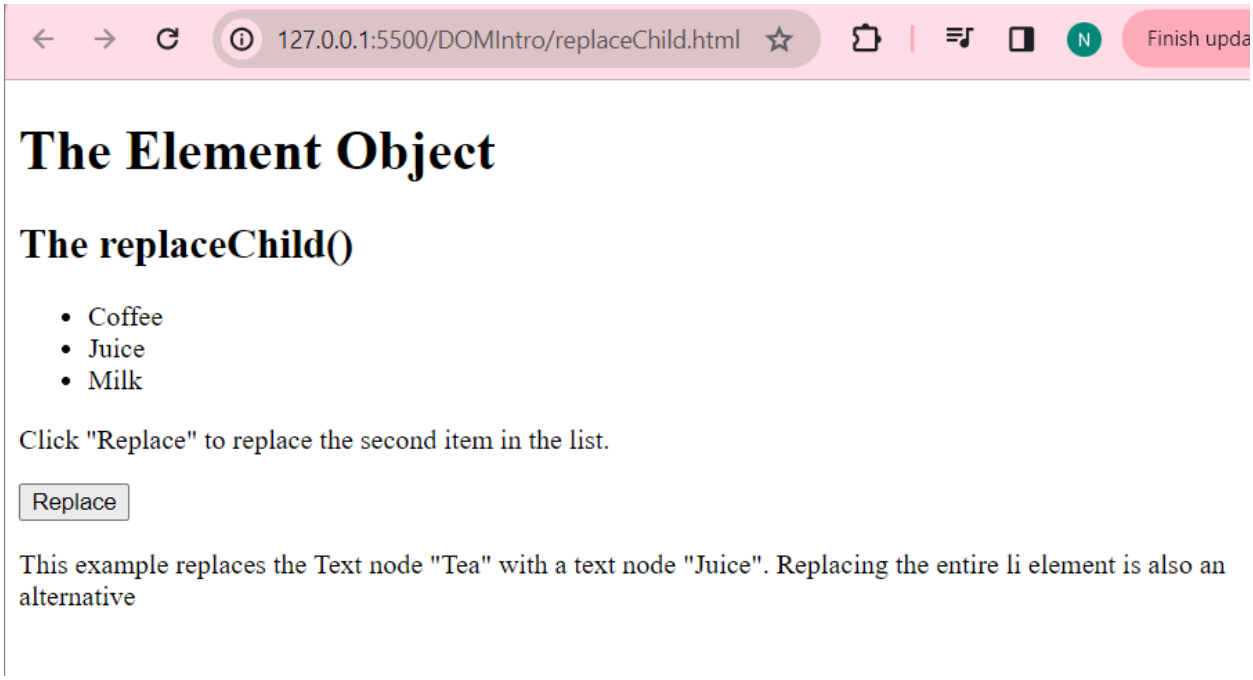
Lets see an example where we will replace a item from the list

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h1>The Element Object</h1>
    <h2>The replaceChild()</h2>

    <ul id="myList">
        <li>Coffee</li>
        <li>Tea</li>
        <li>Milk</li>
    </ul>

    <p>Click "Replace" to replace the second item in the list.</p>
    <button onclick="myFunction()">
        Replace
    </button>
    <p>This example replaces the Text node "Tea" with a text node "Juice".
Replacing the entire li element is also an alternative</p>
```

```
    <script>
        function myFunction(){
            const element = document.getElementById("myList").children[1];
            // create a new text node
            const newNode = document.createTextNode("Juice");

            // replace the text node
            element.replaceChild(newNode, element.childNodes[0]);
        }
    </script>
</body>
</html>
```

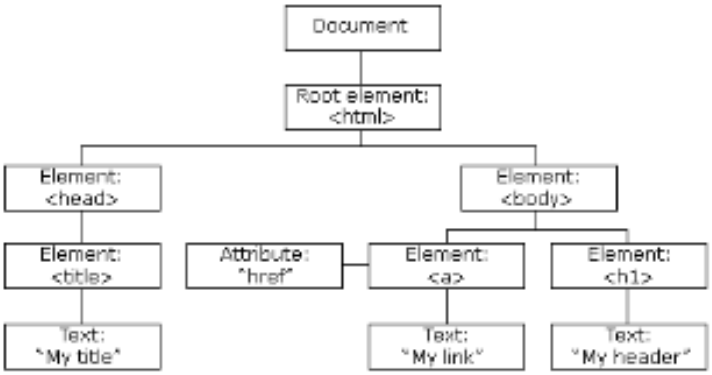
Webpage: Before Replacing

Webpage: After Replacing by clicking on Replace button

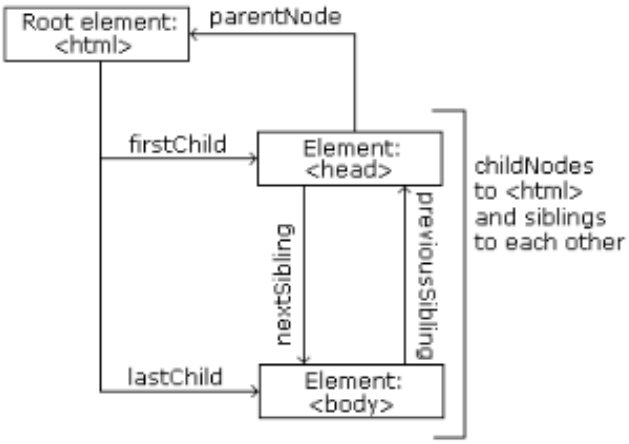


DOM Nodes

- The entire document is a document node
- Every HTML element is an element node
- The text inside HTML elements are text nodes
- Every HTML attribute is an attribute node (deprecated)
- All comments are comment nodes



```
<html>

  <head>
    <title>DOM Tutorial</title>
  </head>

  <body>
    <h1>DOM Lesson one</h1>
    <p>Hello world!</p>
  </body>

</html>
```



The node method

We can access any element by using the following properties with the node object:

- node.childNodes - accesses the child nodes of a selected parent
- node.firstChild - accesses the first child of a selected parent.
- node.lastChild - accesses the last child of a selected parent.
- node.parentNode - accesses the parent of a selected child node.
- node.nextSibling - accesses the next consecutive element (sibling) of a selected element
- node.previousSibling - accesses the previous element (sibling) of a selected element

A **common error in DOM processing** is to expect an element node to contain text.

```
<title id="demo">DOM Tutorial</title>
```

```
myTitle = document.getElementById("demo").innerHTML;
```

```
myTitle = document.getElementById("demo").firstChild.nodeValue;
```

```
myTitle = document.getElementById("demo").childNodes[0].nodeValue;
```

Title is parent for the text node DOM Tutorial

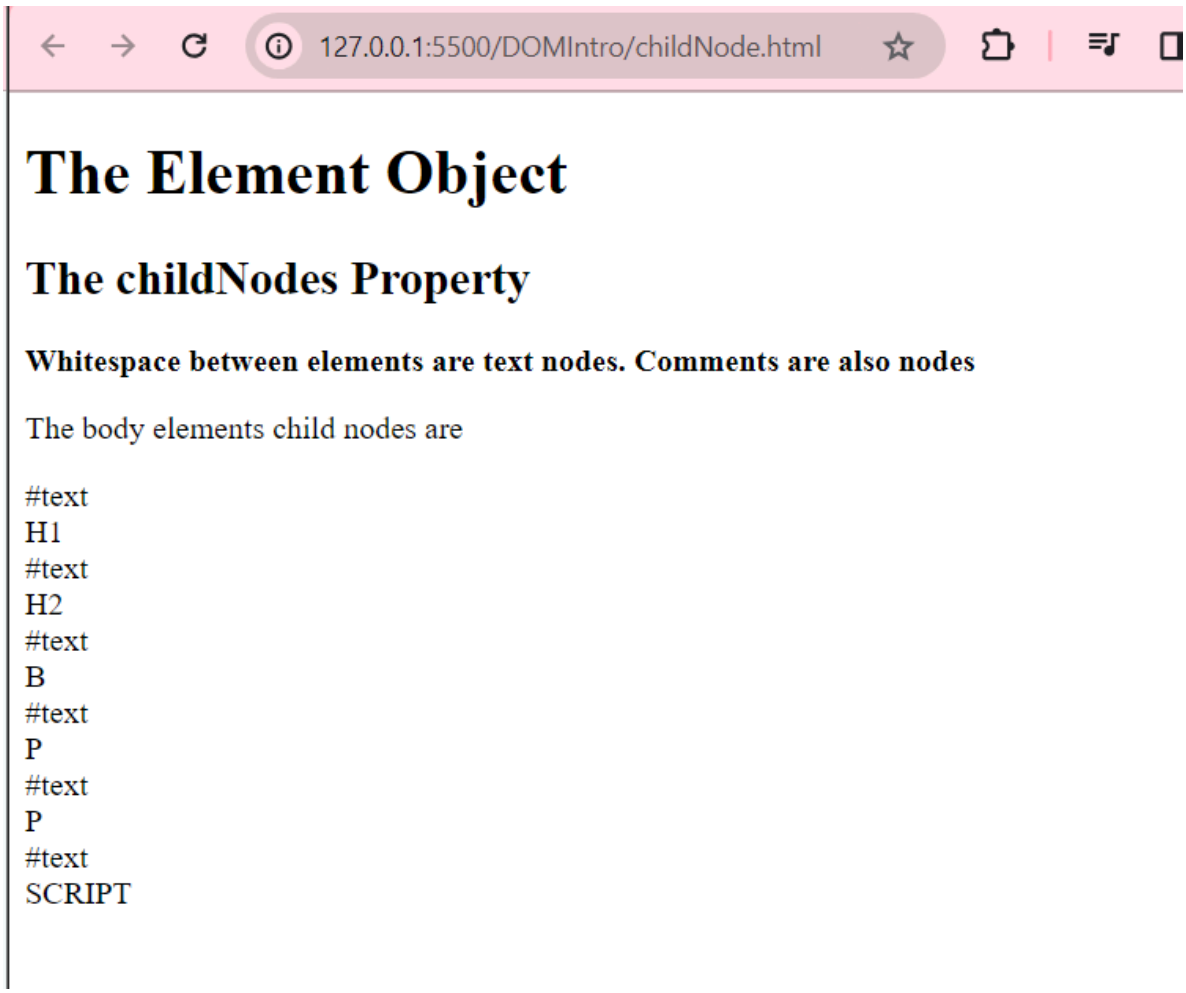
childNodes()

Lets see an example

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>

    <h1>The Element Object</h1>
    <h2>The childNodes Property</h2>
    <b>Whitespace between elements are text nodes. Comments are also
nodes</b>
    <p>The body elements child nodes are</p>
    <p id="demo"></p>

    <script>
        const nodeList = document.body.childNodes;
        let text = "";
        for(let i=0;i < nodeList.length;i++){
```

```
            text += nodeList[i].nodeName + "<br>";
        }
        document.getElementById("demo").innerHTML = text;
    </script>

</body>
</html>
```

Webpage:



Types of Nodes

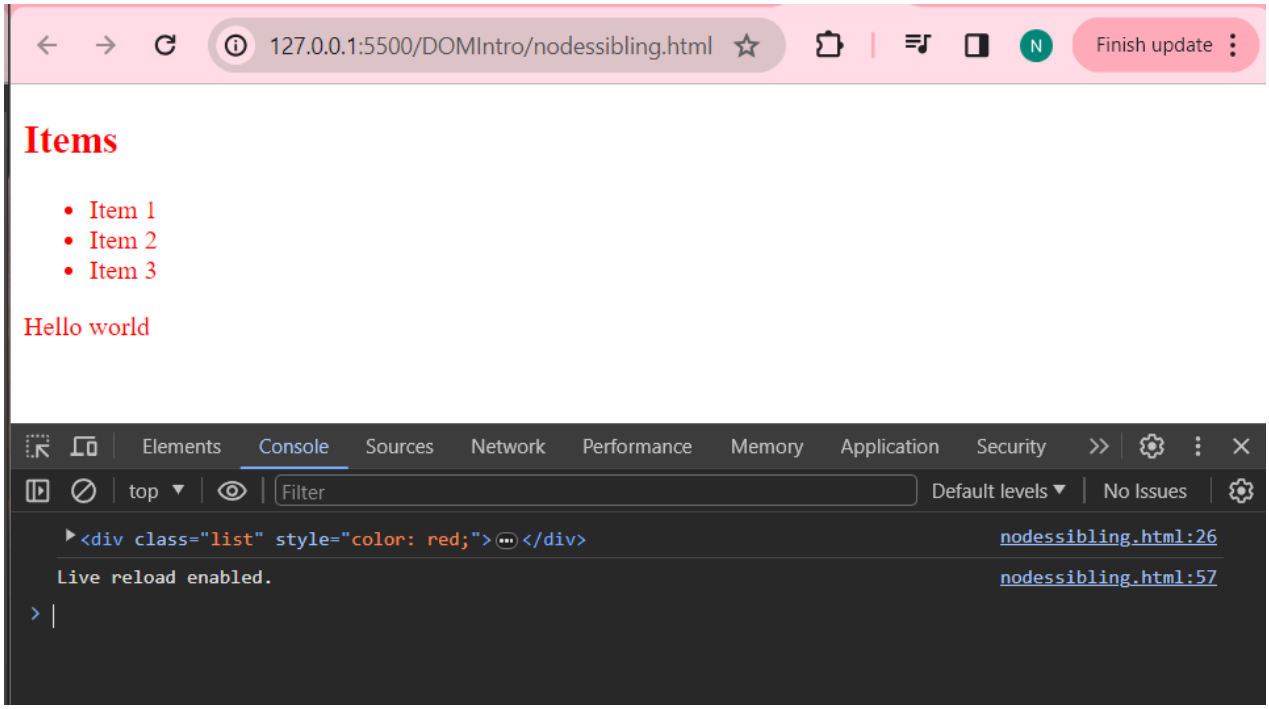
Node	Type	Example
ELEMENT_NODE	1	<h1 class="heading">W3Schools</h1>
ATTRIBUTE_NODE	2	class = "heading" (deprecated)
TEXT_NODE	3	W3Schools
COMMENT_NODE	8	<!-- This is a comment -->
DOCUMENT_NODE	9	The HTML document itself (the parent of <html>)
DOCUMENT_TYPE_NODE	10	<!Doctype html>

Lets see an example to solify our concepts:

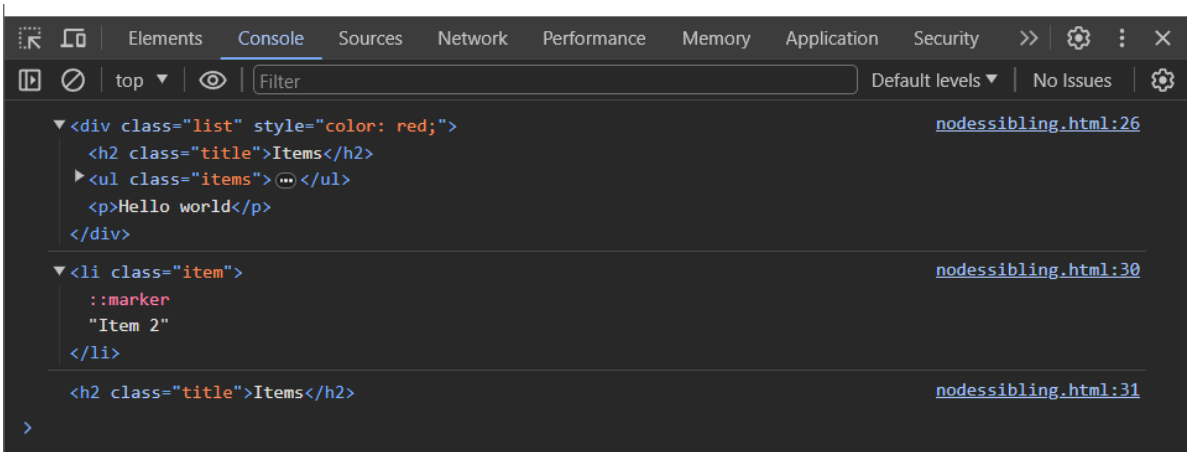
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <div class="list">
        <h2 class="title">Items</h2>
        <ul class="items">
            <li class="item">Item 1</li>
            <li class="item">Item 2</li>
            <li class="item">Item 3</li>
        </ul>
        <p>Hello world</p>
    </div>
    <!-- div is parent of h2,ul,p -->
    <!-- h2,ul,p are siblings -->
    <!-- ul is parent of li -->

    <script type="text/javascript">
        const ul = document.querySelector(".items")

        // parentNode
        console.log(ul.parentNode)
```

```
        ul.parentNode.style.color = "red";
    </script>
</body>
</html>
```

Webpage:




```
    <script type="text/javascript">
        const ul = document.querySelector(".items")

        // parentNode
        console.log(ul.parentNode)
        ul.parentNode.style.color = "red";

        // childrenNode
        console.log(ul.children[1]);
        console.log(ul.previousElementSibling)
    </script>
```



DOM Events:

HTML DOM allows Js to react to HTML events:

A Javascript can be executed when an event occurs, like when a user clicks on an HTML element.

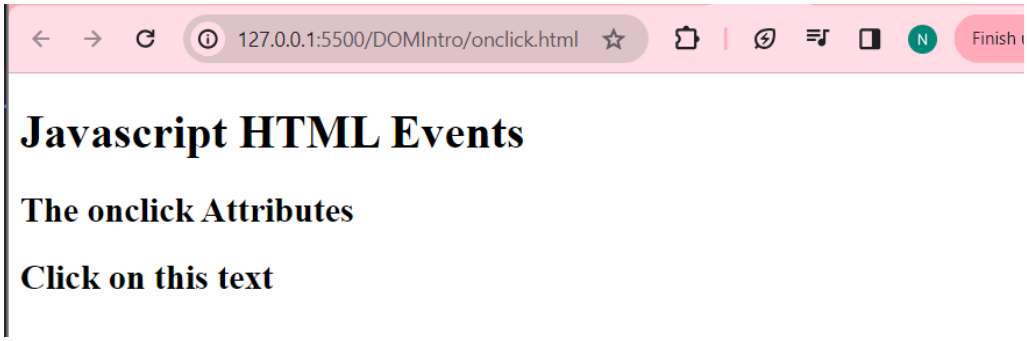
To execute code when a user clicks on an element, add Javascript code to an HTML event attribute.

onlclick = Javascript

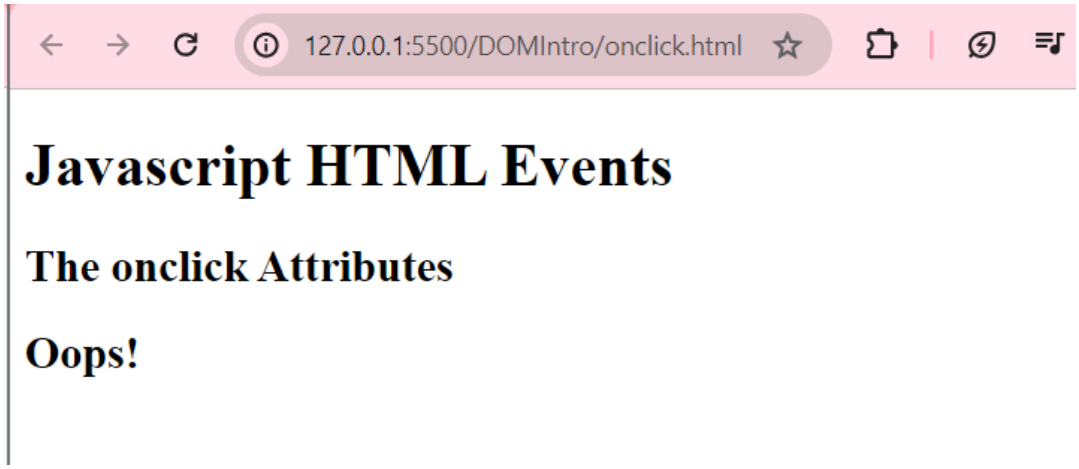
Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h1>Javascript HTML Events</h1>
    <h2>The onclick Attributes</h2>
    <h2 onclick="this.innerHTML='Oops!'">Click on this text</h2>
</body>
</html>
```

Before clicking



After onclick



Another way to do the same thing:

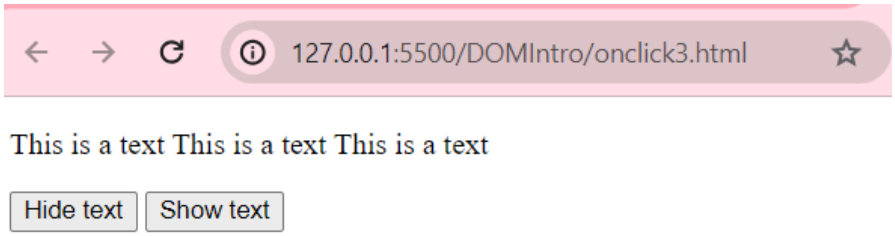
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h1>Javascript HTML Events</h1>
    <h2>The onclick Attributes</h2>

    <h2 onclick="changeText(this)">
    Click on this text
    </h2>
    <script>
```

```
        function changeText(id){
            id.innerHTML = "Ooops!"
        }
    </script>
</body>
</html>
```

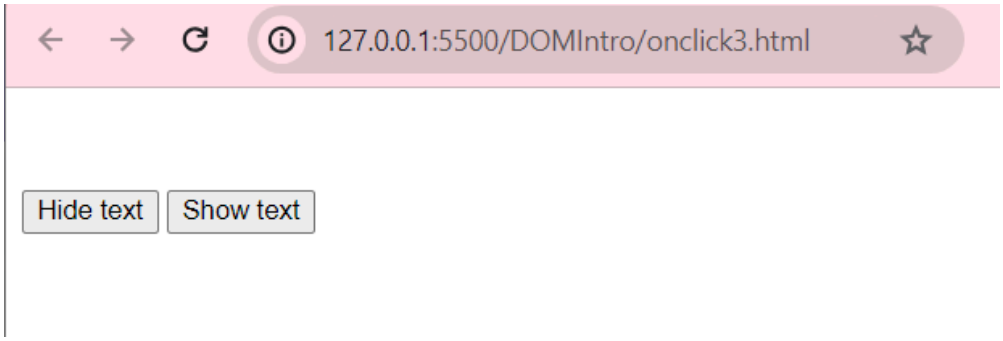
Another example

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <p id="p1">
        This is a text
        This is a text
        This is a text
    </p>
    <input type="button" value="Hide text"
onclick="document.getElementById('p1').style.visibility='hidden'">
    <input type="button" value="Show text"
onclick="document.getElementById('p1').style.visibility='visible'">
</body>
</html>
```

Webpage



Hide text button is pressed



Example: show something when we press a button.

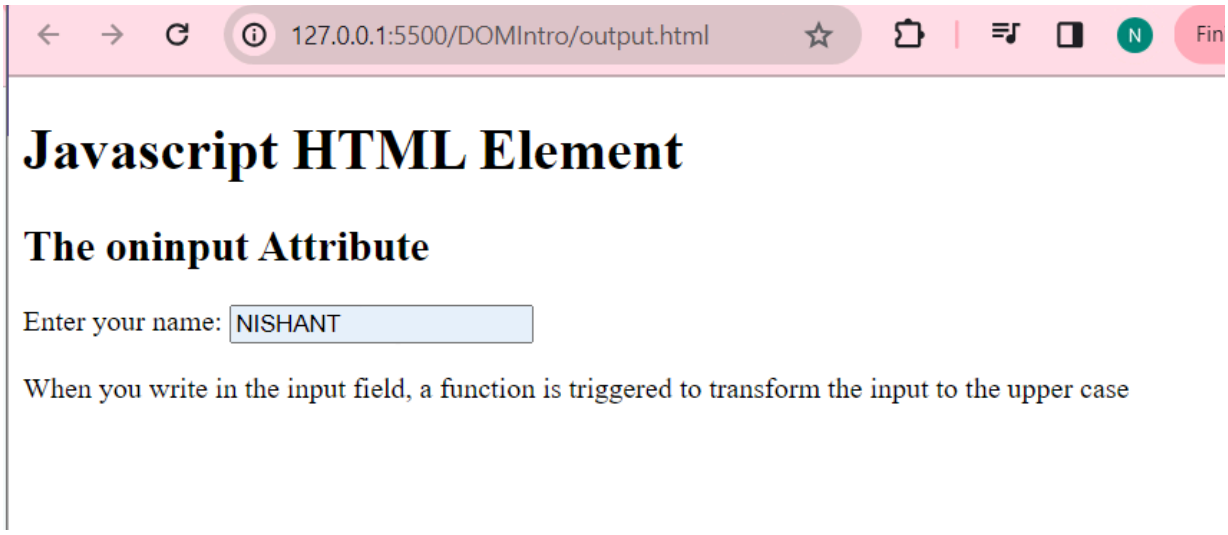
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h1>Javascript HTML Events</h1>
    <h2>The onclick Events</h2>

    <p>Click "Try it" to execute the displayDate() functions.</p>
    <button id="myBtn">Try it</button>
    <p id="demo"></p>

    <script>
        document.getElementById("myBtn").onclick = displayDate();
        function displayDate(){
            document.getElementById("demo").innerHTML = Date();
        }
    </script>
</body>
</html>
```

Another example:

Whenever i have input change it to uppercase.(Can be used in Capcha)

```
<!DOCTYPE html>
```

```html
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h1>Javascript HTML Element</h1>
    <h2>The oninput Attribute</h2>

    Enter your name: <input type="text" id="fname" oninput="upperCase()">
    <p>When you write in the input field, a function is triggered to
transform the input to the upper case</p>

    <script>
        function upperCase(){
            const x = document.getElementById("fname")
            x.value = x.value.toUpperCase();
        }
    </script>
</body>
</html>
```

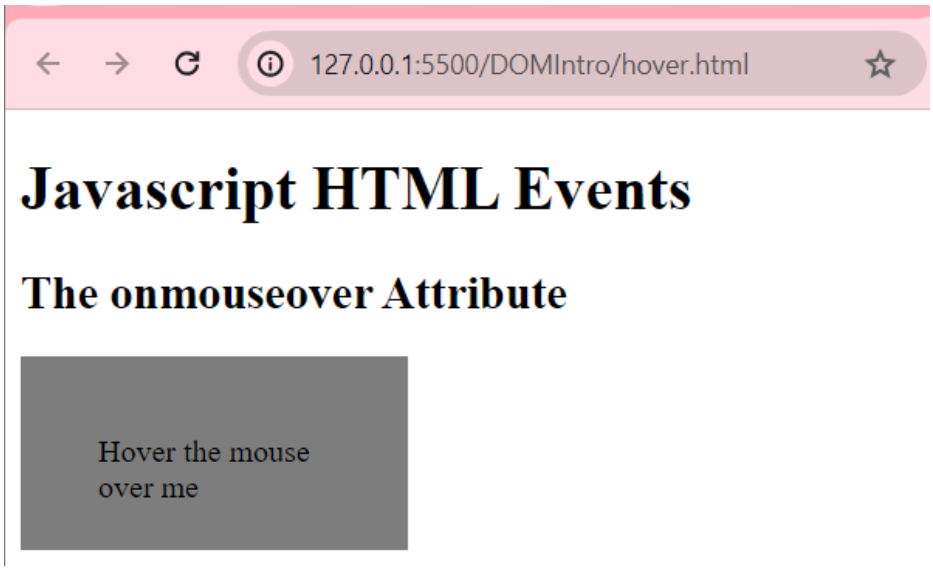


Another example

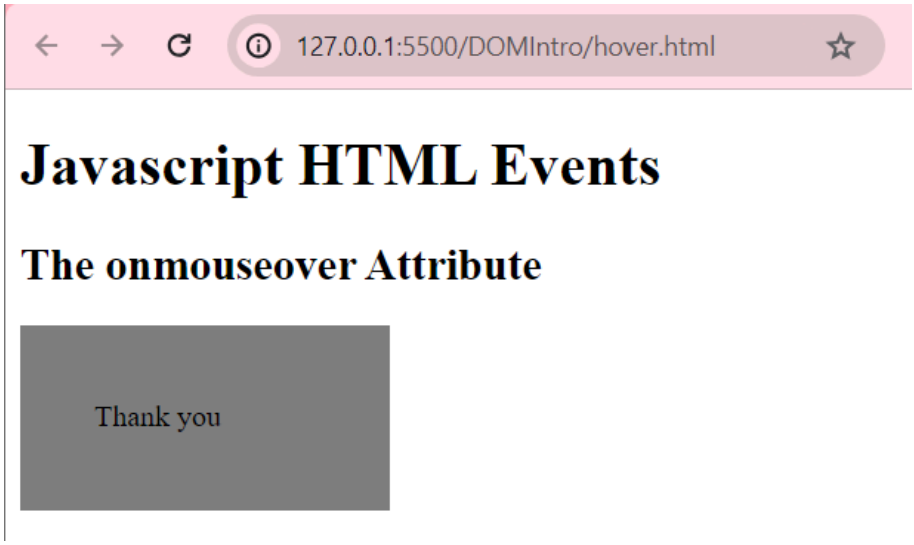
When we hover over a div things changes

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h1>Javascript HTML Events</h1>
    <h2>The onmouseover Attribute</h2>

    <div onmouseover="mOver(this)" onmouseout="mOut(this)"
style="background-color: grey;width: 120px;height: 20px;padding:
40px;">Hover the mouse over me</div>
</body>
<script>
    function mOver(obj){
        obj.innerHTML = "Thank you"
    }
    function mOut(obj){
        obj.innerHTML = "Hover the mouse over me"
    }
</script>
</html>
```

Webpage:



When we hover on the div

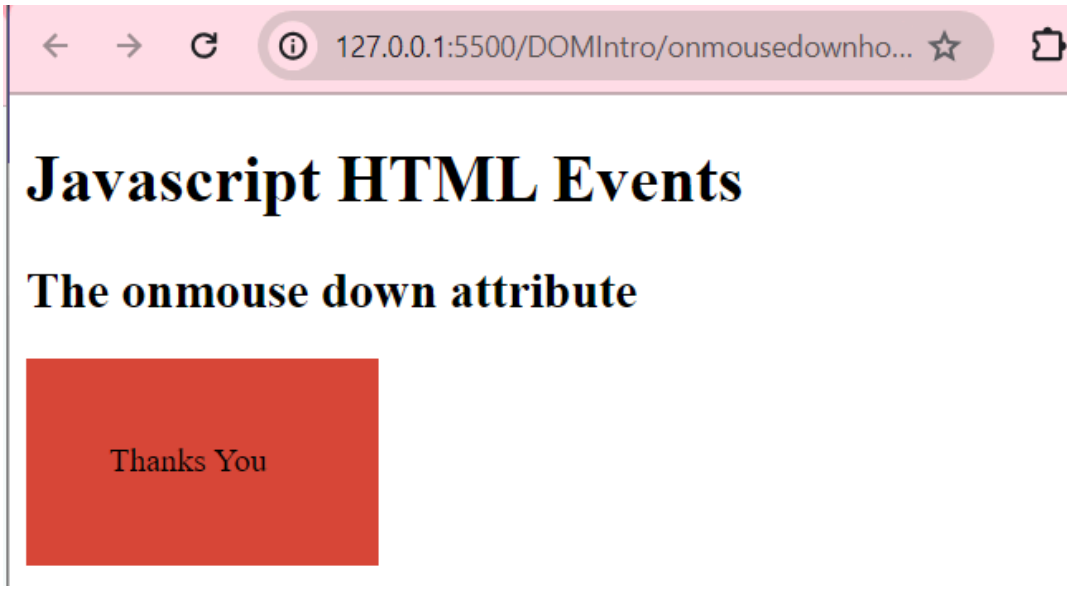


Another example: where we click on the mouse and something happens and when we release the mouse something else happens

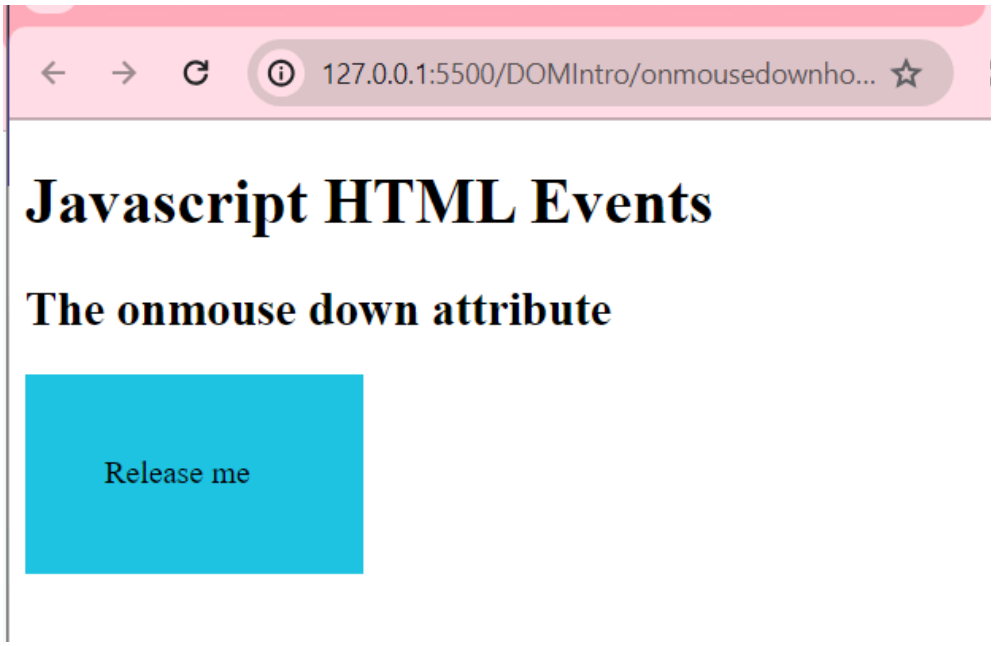
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h1>Javascript HTML Events</h1>
    <h2>The onmouse down attribute</h2>

    <div onmousedown="mDown(this)" onmouseup="mUp(this)"
style="background-color: #D94A38;width: 90px;height: 20px;padding: 40px;">
    Click Me
    </div>

    <script>
        function mDown(obj){
            obj.style.backgroundColor = "#1ec5e5";
            obj.innerHTML = "Release me";
        }
        function mUp(obj){
```

```
            obj.style.backgroundColor = "#D94A38";
            obj.innerHTML = "Thanks You";
        }
    </script>
</body>
</html>
```

Webpage:



When we clicked on the div



Onload and Onunload functions

The onload and onunload events are triggered when the user enters or leaves the page. The **onload** event can be used to check the visitor's browser type and browser version and load the proper version of the web page based on the information.

The onload and onunload events can be used to deal with cookies.

DOM EventListener

- Attaches an event handler to the specified element (without overwriting existing event handlers).
- You can add many event handlers(even of the same type) to one element.
- You can add event listeners to any DOM object not only HTML elements. I.e the window object.
- The addEventListener() method makes it easier to control how the event reacts to bubbling.(how a particular event is applied : is it from outermost element to the innermost element or vice versa.)
- When using the addEventListener() method, the JavaScript is separated from the HTML markup, for better readability and allows you to add event listeners even when you do not control the HTML markup.
- You can easily remove an event listener by using the removeEventListener() method.

Examples

Execute a function when user click on a button.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

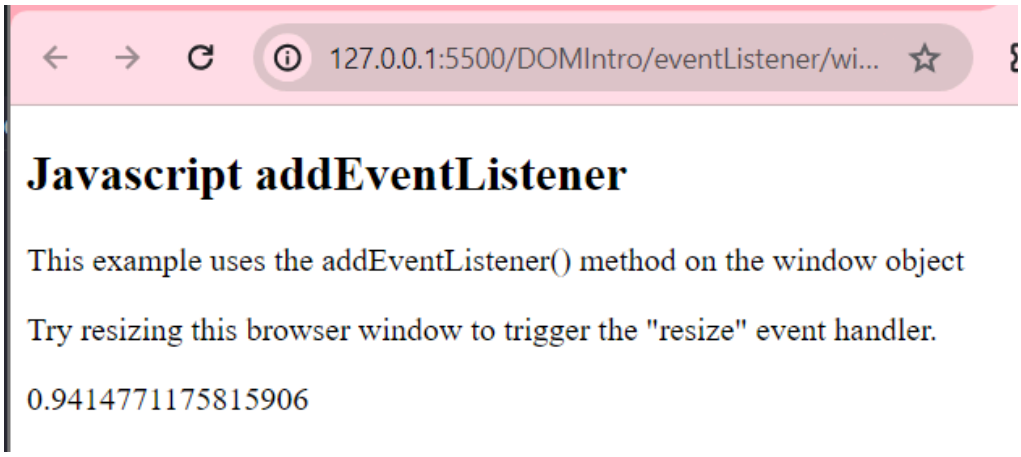
```
    <title>Document</title>
</head>
<body>
    <h2>Javascript addEventListeners</h2>
    <p>This example uses the addEventListener() method to execute a
function when a user clicks on a button. </p>
    <button id="myBtn">Try it</button>
    <script>

document.getElementById("myBtn").addEventListener("click",myFunction);
        function myFunction(){
            alert("Hello Duniya !!")
        }
    </script>
</body>
</html>
```

Windowevent

Example : Random value shown when we resize the window

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h2>Javascript addEventListener</h2>
```

```
    <p>This example uses the addEventListener() method on the window
object</p>
    <p>Try resizing this browser window to trigger the "resize" event
handler.</p>
    <p id="demo"></p>
    <script>
        window.addEventListener("resize",function(){
            document.getElementById("demo").innerHTML =Math.random();
        });
    </script>
</body>
</html>
```

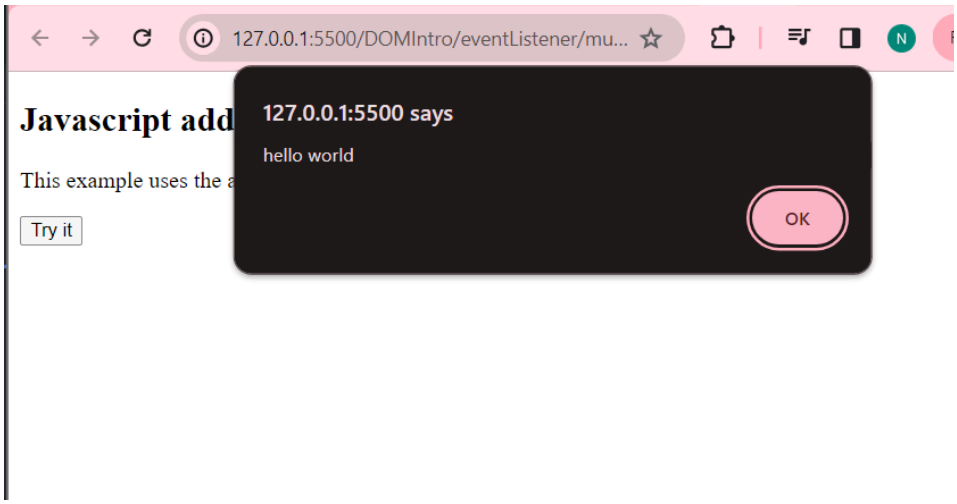


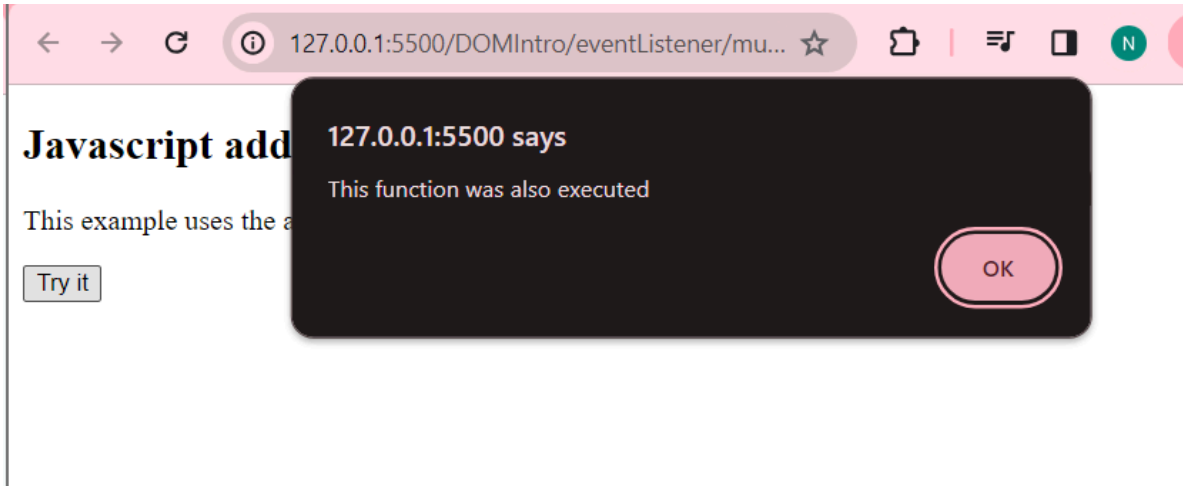
Example: Multiple event listener

We can add multiple event listener to the same element

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h2>Javascript addEventListener</h2>
    <p>This example uses the addEventListener() method to add two click
events to the same button.</p>
    <button id="myBtn">
```

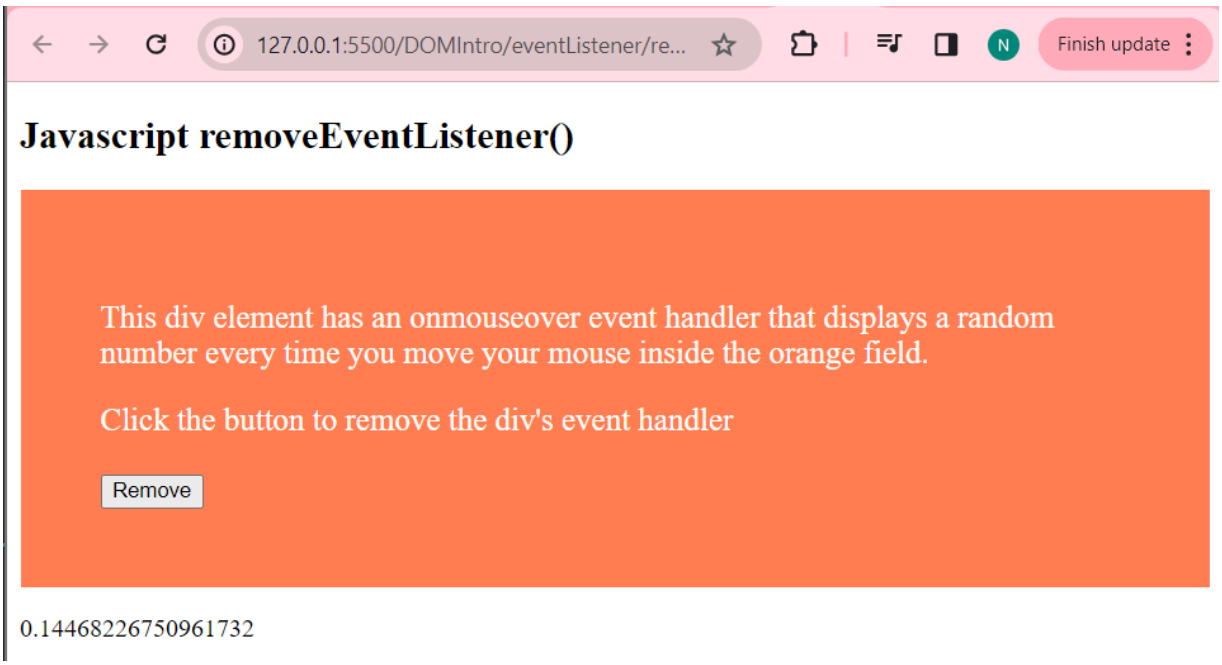
```
        Try it
    </button>

    <script>
        var x = document.getElementById("myBtn")
        x.addEventListener("click",myFunction);
        x.addEventListener("click",someOtherFunction);

        function myFunction(){
            alert("hello world");
        }
        function someOtherFunction(){
            alert("This function was also executed");
        }
    </script>
</body>
</html>
```





Example: removehandler()

In this example as we move a mouse over a div it will display a random number and the moment we click on the button to remove the div's event handler it will stop the generation and display of the new random numbers.



Event Bubbling or Event Capturing??

Two ways of event propagation in the HTML DOM - bubbling and capturing.

Event propagation is a way of defining the element order when an event occurs. If you have a <p> element inside a element and the user click on the <p> element , which element's "click" event should be handled first??

In bubbling the inner most element's event is handled first and then the outer: the <p> element's click event is handled first, then the element's click event.

In capturing the outer most element's event is handled first and then the inner: the element's click event will be handled first, then the <p> element's click event.

addEventListener(event, function, useCapture)

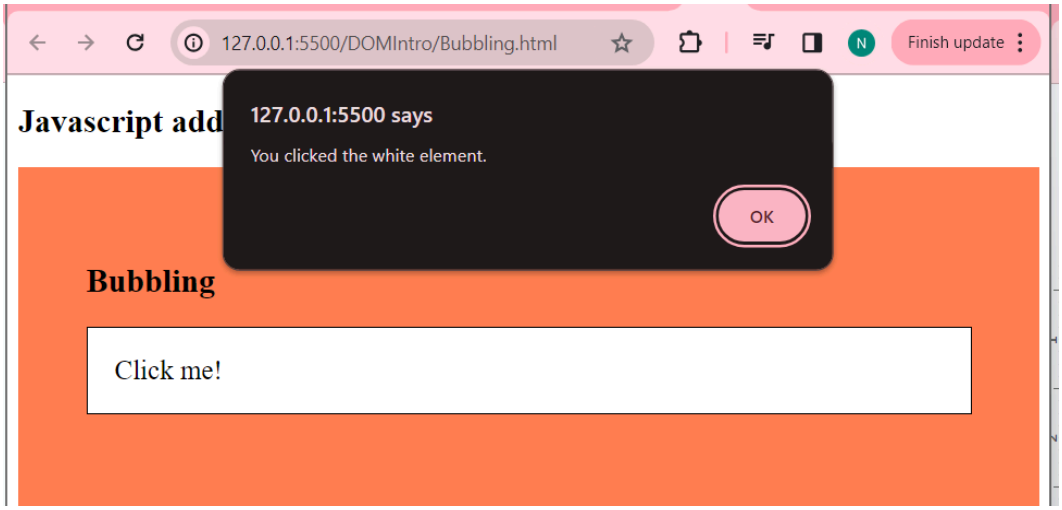
Example:

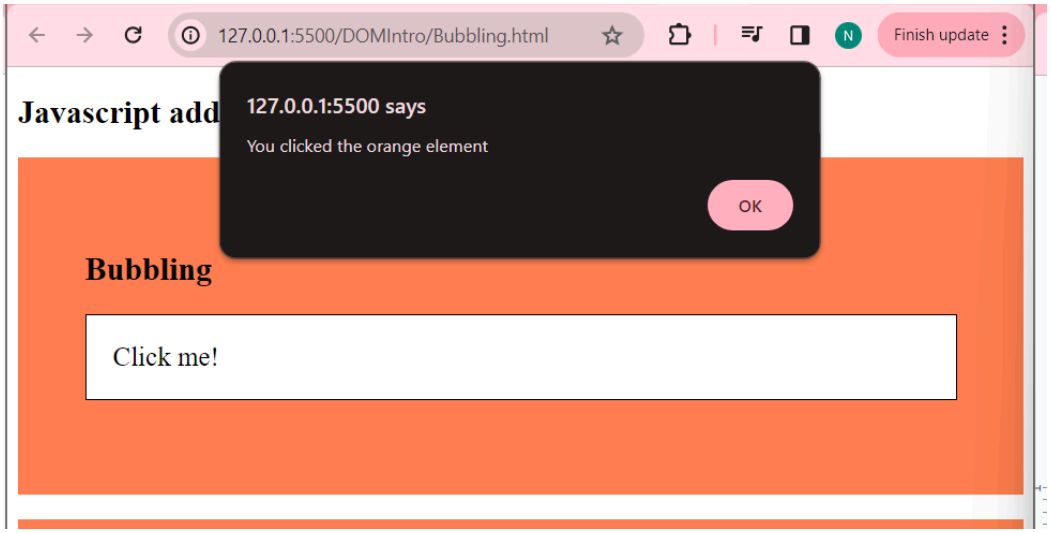
```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Document</title>
    <style>
        #myDiv1,#myDiv2{
            background-color: coral;
            padding: 50px;
        }

        #myP1,#myP2{
            background-color: white;
            font-size: 20px;
            border: 1px solid;
            padding: 20px;
        }
    </style>
    <meta content="text/html"; charset="utf-8" http-equiv="Content-Type">
</head>
<body>
```

```
    <h2>Javascript addEventListener()</h2>
    <div id="myDiv1">
        <h2>Bubbling</h2>
        <p id="myP1">Click me!</p>
    </div><br>

    <div id="myDiv2">
        <h2>Capturing</h2>
        <p id="myP2">Click me!</p>
    </div>
    <script>

document.getElementById('myP1').addEventListener("click",function(){
            alert("You clicked the white element.")
        },false)
        // default false

document.getElementById("myDiv1").addEventListener("click",function(){
            alert("You clicked the orange element");
        },false)

document.getElementById('myP2').addEventListener("click",function(){
            alert("You clicked the white element.")
        },true)

document.getElementById("myDiv2").addEventListener("click",function(){
            alert("You clicked the orange element");
        },true)
    </script>
</body>
</html>
```

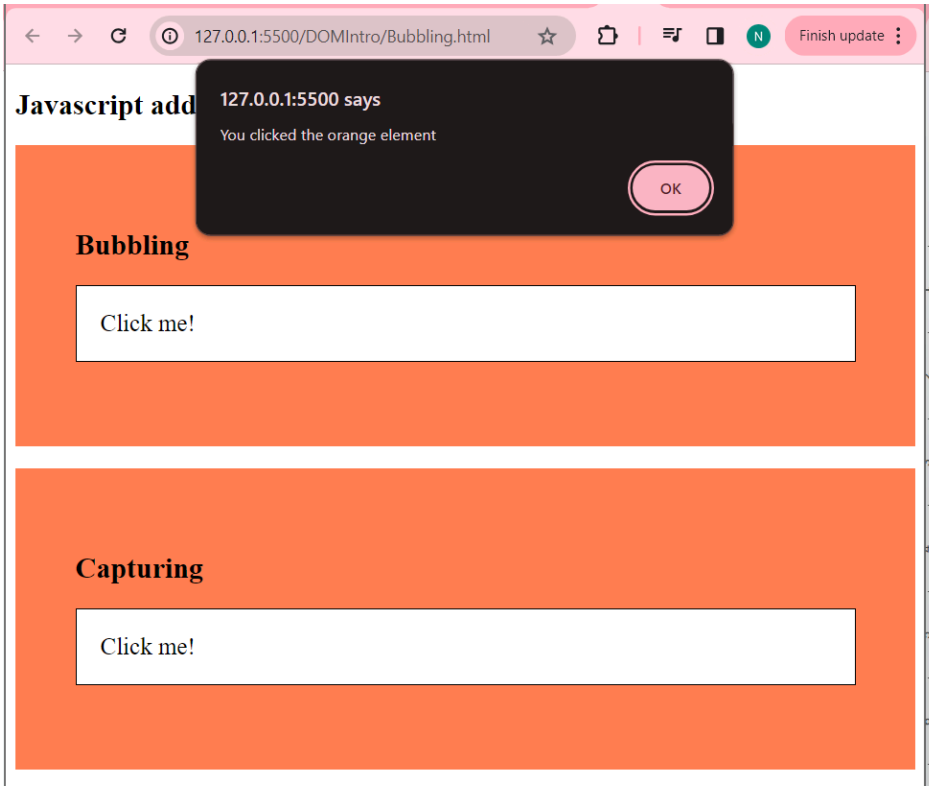
Lets see Bubbling first

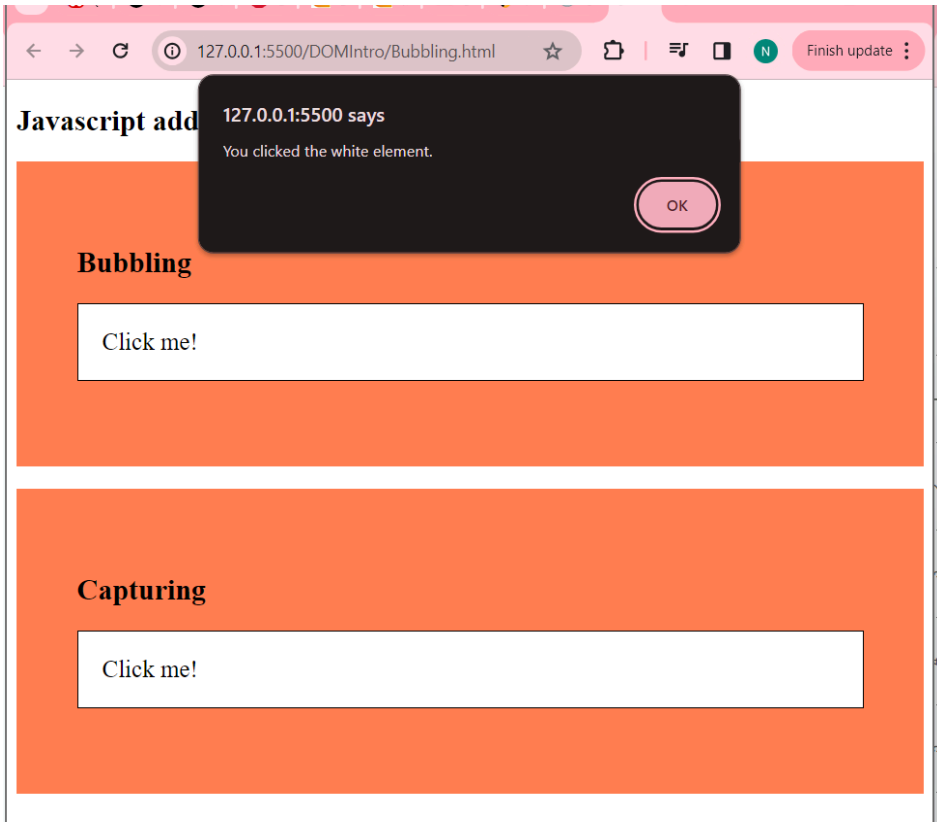
We click on the Click me ! text (white element) (p element child of div with class myDiv1)





Now lets see Capturing:


127.0.0.1:5500/DOMIntro/Bubbling.html
Finish update
Javascript add
127.0.0.1:5500 says
You clicked the orange element
OK
Bubbling
Click me!
Capturing
Click me!


127.0.0.1:5500/DOMIntro/Bubbling.html
Finish update
Javascript add
127.0.0.1:5500 says
You clicked the white element.
OK
Bubbling
Click me!
Capturing
Click me!