Web Application Programming

Week 4: Client-Side Scripting with JavaScript(DOM manipulation and event handling Form validation and user input handling)

By Elubu Joseph - MSc.IS

Lecturer

Department of Information Technology

Kumi University

Email: josebulinda@gmail.com or jose@kumiuniversity.ac.ug

Agenda

1. Summary of previous lecture
2. JavaScript in HTML
3. Event handling
4. DOM manipulation
5. Form validation and user input handling

Summary of previous lecture

1. JavaScript fundamentals: variables(looked declarations of varriables; Automatically, Using var, Using let, Using const and initialization of the same), data types(**Primitive Data Types:** Number, String, Boolean, Undefined, Null, Symbol (ES6) and BigInt (ES11), and **Reference Data Types:** Objects, Date, Arrays, functions etc), operators(Arithmetic, Assignment, Comparison, Logical Operators etc.), conditional statements, loops etc. and

JavaScript in HTML

The JavaScript code can be inserted in HTML file by using the HTML **<script>** tag. When an HTML document is loaded with the **<script>** tag in the web browser, the browser processes the content enclosed inside the script tag as JavaScript code.

The script tag can contain either scripting statements or refer to an external JavaScript file. The script tag provides a src attribute that allows us to add a reference to an external script file.

JavaScript is the default scripting language for most of the browsers.

Script tag Attributes and their Uses

Following is the basic syntax of using a <script> tag:

```
<script src="JS_FILE_NAME_OR_URL" type="..."></script>
```

we can also use the **<script>** tag to directly add the JavaScript code too, like this:

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

// JS code here

</script>
```

There are five attributes of script tag which are listed below in the table:

Attributes	Values	Description
async	•true •false	It specifies whether the script should be executed asynchronously or not.
type	•text/ECMAScript •text/javascript •application/ECMAScript •application/javascript •text/VBScript	It specifies the Multipurpose Internet Mail Extensions (MIME) type of script. Generally, **text/javascript** value is used.
charset	charset	It specifies the type of character encoding used in the script
defer	•true •false	It specifies whether the browser continues parsing the web page or not.
src	URL	It specifies the uniform source locator(URL) of a file that contains the script.

(Studytonight.com. n.d.).

JavaScript in HTML Webpage

Now that we know about the **<script>** tag which is used to include JavaScript code in a webpage, let's see the various ways you can use script tag inside an HTML web page to add JavaScript code in the following ways:

1. In the HEAD element (<head>...</head>)
2. In the BODY element (<body>...</body>)
3. To include an External JavaScript File

Its time to cover all of this one by one along with code examples to help you understand them.t ways in which we can do so.

JavaScript in HTML Webpage

1. Script tag with JavaScript Code in <head> Element

Let's put the script tag inside the HTML head element. The script placed inside the head element is loaded with the webpage and gets executed if any defined event occurs.

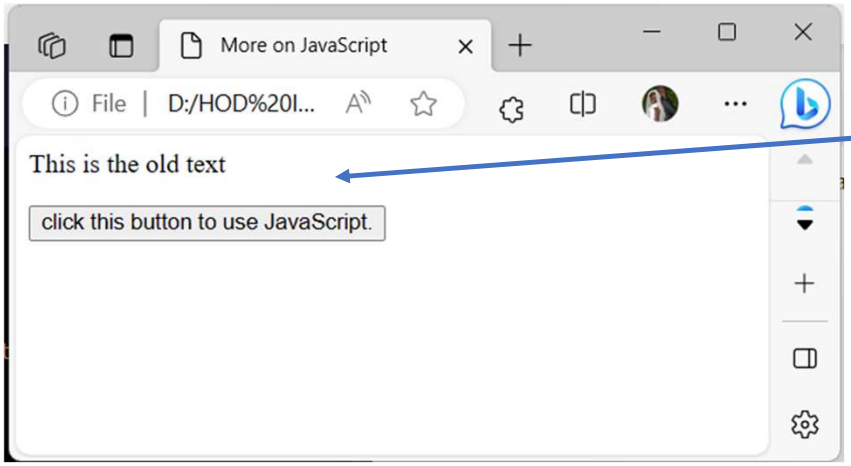
The code given below shows how to use the <script> tag inside the <head> element of an HTML document to add some JavaScript code.

JavaScript in HTML Webpage

1. Script tag with JavaScript Code in <head> Element+

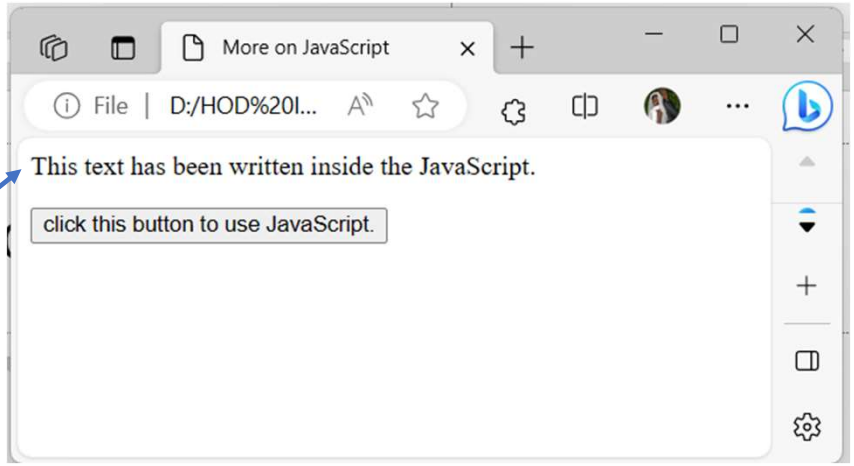
```
<html> <head>
<title>my first JavaScript</title>
      <script>
      function text() {
            document.getElementById("script").innerHTML= "This text has
            been written inside the JavaScript.";
      }
      </script>
</head>
<body>
      <p id="script">this is the old text</p>
      <button type="button" onclick=" text() ">click this button to use
      JavaScript.
      </button>
</body>
</html>
```

JavaScript in HTML Webpage code output



Output before clicking the button

On clicking the button the output changes based on the code in the body



JavaScript in HTML Webpage

2. Script tag with JavaScript Code in <body> Element

You can place a script tag inside the body element of an HTML document too. The script tag inside the body element runs when a web page starts loading in a web browser.

Below is the code to show you how to place a <script> element inside the <body> element of an HTML document:

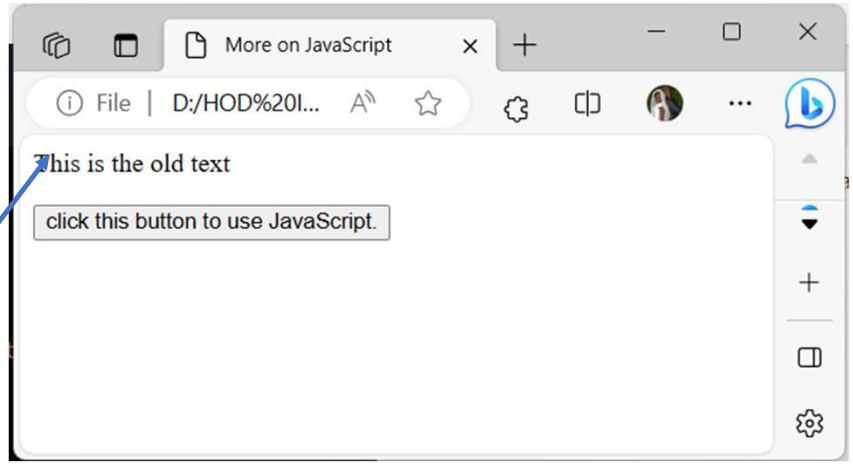
JavaScript in HTML Webpage

2. Script tag with JavaScript Code in <body> Element

```
<html> <head>
<title>my first JavaScript</title>
      </head>
<body>
      <script>
      function text() {
            document.getElementById("script").innerHTML= "This text has
            been written inside JavaScript within html body.";
      }
      </script>

      <p id="script">this is the old text</p>
      <button type="button" onclick=" text() ">click this button to use
      JavaScript.
      </button>
</body>
</html>
```

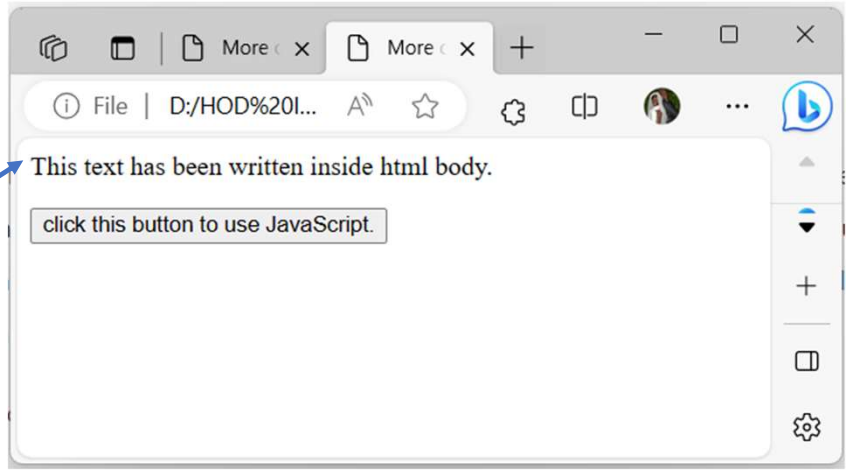
JavaScript in HTML Webpage

2. Script tag with JavaScript Code in <body> Element+output



Output before clicking the button

On clicking the button the output changes based on the code in the body



JavaScript in HTML Webpage

3. JavaScript Code in an External File

Whenever we put lengthy JavaScript code in an HTML document, it affects the readability and maintainability of the HTML document.

In addition, sometimes there is a need to use the same JavaScript code in several web pages. In such cases, we can store the JavaScript code in an external file and save that file with the **.js** extension.

All JavaScript code files should have an extension **.JS** and nothing else.

To link the external file, we can provide its location (URL) in the **src** attribute of the script tag.

JavaScript in HTML Webpage

3. JavaScript Code in an External File

Including External JavaScript Syntax:

This is the way by which we can add an external JavaScript file to our HTML file:

```
<script src= "FileName.js" type="text/javascript"></script>
```

Note! The type attribute is optional in the code example above.

JavaScript in HTML Webpage

3. JavaScript Code in an External File

The JavaScript code is stored in a file with name FileName.js

```
function kumu() {
      document.getElementById("script").innerHTML = "This text has
been written inside the JavaScript. File called 'FileName.js.";
}
```

In the code above, we have defined a simple function called kumu in JavaScript, we will learn about JavaScript functions later.

JavaScript in HTML Webpage

3. JavaScript Code in an External File FileName.js

Linking JavaScript file with html file.

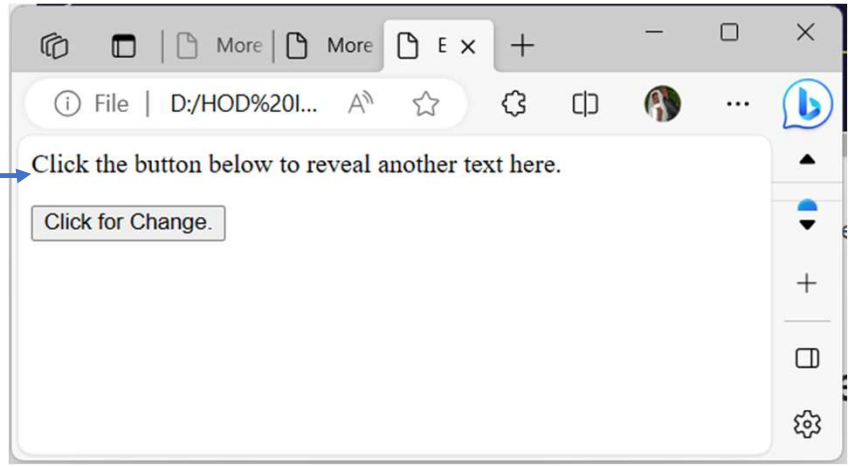
The code given below shows you how to link an external JavaScript file with an HTML document.

The JavaScript code stored in a file with name FileName.js

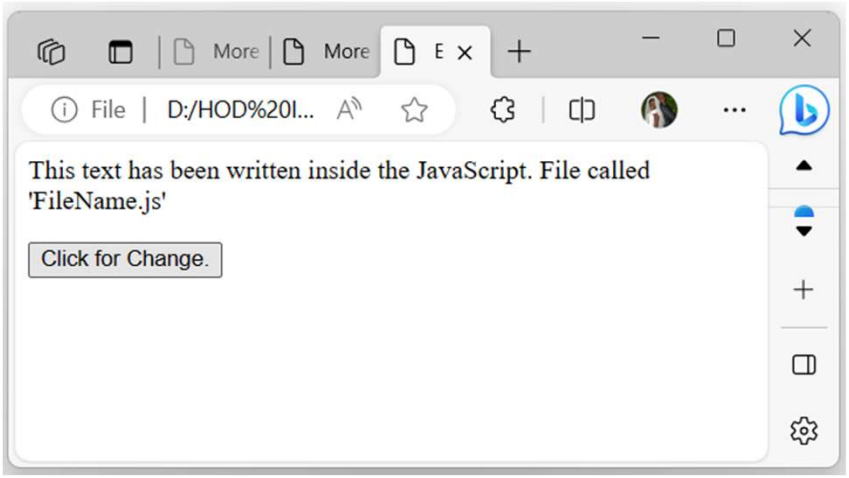
```
<html> <head> <title>Embending JavaScript file into html</title>
       <script src="FileName.js"></script>
      </head>
<body>

      <p id="script">this is the old text</p>
      <button type="button" onclick=" kumu() ">Click for Change.</button>
</body> </html>
```

JavaScript in HTML Webpage

3. JavaScript Code in an External File FileName.js



Output Before clicking the button



Output After clicking the button

JavaScript in HTML Webpage

3. JavaScript Code in an External File

Advantages of External JavaScript File:

Using an external JavaScript file has its own merits.

1. It separates the HTML and JavaScript code and makes the code look clean and easy to understand.
2. External JavaScript code can be reused in multiple HTML webpages.
3. External JavaScript code can be cached in the browser. Once cached the browser will not load the JavaScript file again and again and will use the cached version of it. This will make your webpage loading fast.

JavaScript in HTML Webpage

3. JavaScript Code in an External File

Best Practice while linking JavaScript external file into HTML Page

In large projects, JavaScript code can be huge and there can be multiple external JavaScript files included in each HTML page using multiple **<script> tags**. Yes, we can use multiple **<script> tags** to include as many external JavaScript files as we want.

For example, if we have 3 JavaScript file, with names,

1. **form.js**,
2. **tables.js** and
3. **chairs.js** and

we have to include all of these in our HTML page. We will use **3 <script> tags** in this case,

JavaScript in HTML Webpage

3. JavaScript Code in an External File

```
<script src="form.js" type="text/javascript"></script>
<script src="tables.js" type="text/javascript"></script>
<script src= "chairs.js" type="text/javascript"></script>
```

Now the question is, where should we put the above code in our HTML page. Should we put it inside the HEAD section of HTML code, or should we put it in the BODY section of our HTML page?

Well, **if we put it in the HEAD section**, then when our webpage will load, **all the JavaScript files will be loaded first which can slow down our webpage loading**, which is not good.

So, we should load the external JavaScript files used in a webpage, at last, **means either just before the closing </body> tag** or **after the closing </body> tag**, so that first our complete webpage loads and then the external JavaScript files are loaded. This way, even if we have large JavaScript files, our webpage will not slow down because of it.

JavaScript in HTML Webpage

3. JavaScript Code in Multiple External Files

```
<html> <head> <title>Embending JavaScript file into html</title>
            </head>
<body>
      <p id="script">this is the old text</p>
      <button type="button" onclick=" kumu() ">Click for Change.</button>
</body>
<script src="FileName.js"></script>
<script src="form.js"></script>
<script src="tables.js"></script>
<script src="chairs.js"></script>

</html>
```

Note that JavaScript files have been added below the closing <body> tag.

Assignment

1. Create the above three JavaScript files and link them to an html file of your choice.

JavaScript Event Handling

JavaScript Event Handling

In JavaScript, events refer to the actions that are detected by a web browser whenever any user action on the browser screen is realized.

So everything, starting from the movement of the mouse, keyboard click, hover over any particular HTML element, form submission, click on any input field, selection of a value from a dropdown, and everything else you do on a webpage, the browser generates an event for it which can be handled using JavaScript.

JavaScript enables us to write scripts to be executed when any of these events occur.

JavaScript Event Handling+

mouseEvents.html file

For example, the onclick event is detected by the browser whenever you click the mouse button and using JavaScript we can perform any action on mouse click like **we can set a counter** and keep a track of mouse clicks to see how many times the user used the mouse click.

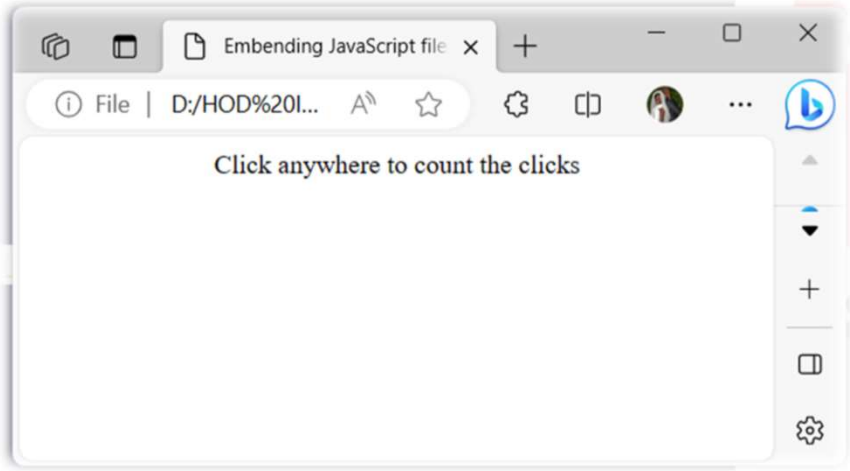
In the **live example** below, we have implemented a simple **mouse click counter**, click on the Output area to see the number of clicks getting updated as we have captured the mouse click event on the complete body of the HTML page for the HTML File named **mouseEvents.html**

JavaScript Event Handling++

```
<html> <head>
<title>Embending JavaScript file into html</title>
<script>
      var counter = 0;
      function countClicks() {
      counter =   counter + 1;
      document.getElementById("num_of_clicks").innerHTML =  "Number of
Mouse clicks: " + counter;
      }
</script> </head>
<body onclick="countClicks()">
<!-- HTML page body -->
<p id="num_of_clicks"></p>
 <p align = 'center'>Click anywhere to count the clicks</p>
</body>
</html>
```

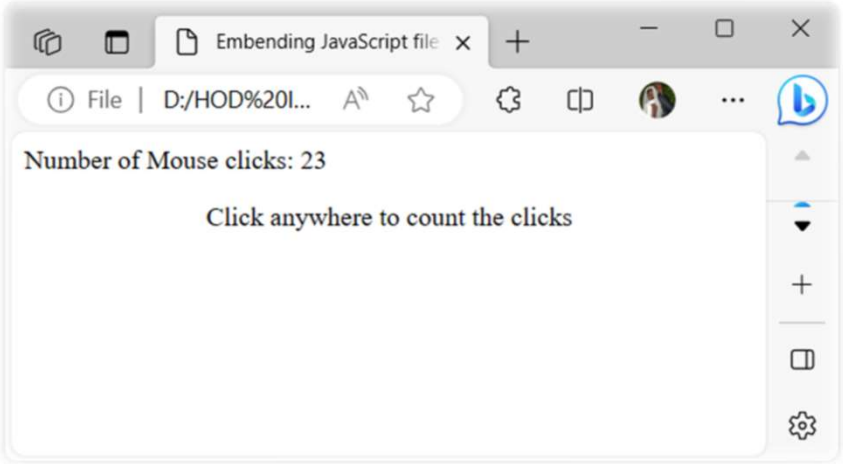
JavaScript Event Handling+

mouseEvents.html file -Output



Output before a click

Output After a mouse click



In the code above the function that is used to perform an action upon capturing the mouse click event is known as an **Event handler**, which handles a particular event when the event is triggered.

JavaScript Event Handling Syntax

Following is the syntax to add a handler for any particular event in any HTML element.

```
<ABC_ELEMENT onXYZ="EVENT_HANDLER_CODE"></ABC_ELEMENT>
```

In the above code, we have added an event handler to capture the **XYZ** event on **ABC_ELEMENT** of the webpage.

So whenever the **XYZ** event will occur with respect to the **ABC_ELEMENT** our event handler code will be executed.

Let's take a few examples covering a few different events so that you can understand how this works.

JavaScript Event Handling

JavaScript onchange Event Example

This event is created when an HTML element like a select, or a radio button, or a checkbox, etc. changes. You can define an event handler to perform some action when this event occurs. In the code example below, we have created a dropdown using the **<select>** tag with **<option>** tags used inside it to add options to the dropdown.

Now once we add the **onchange** event handling on this element, whenever use selects any value, our event handler function will be executed.

JavaScript Event Handling

JavaScript onchange Event Example+

```
<!doctype html> <head> <title>JavaScript onchange Event</title> </head>
<body>
	<select id="dropdown">
	<option value="1">1</option>
	<option value="2">2</option>
	<option value="3">3</option>
	</select>
	<script> /* JS comes here */
	function eventHandler() {
		// do something
		alert("Value selected");
	}
	// setup event handler on HTML element
	document.getElementById("dropdown").onchange = eventHandler;
	</script>
</body> </html>
```

JavaScript Event Handling

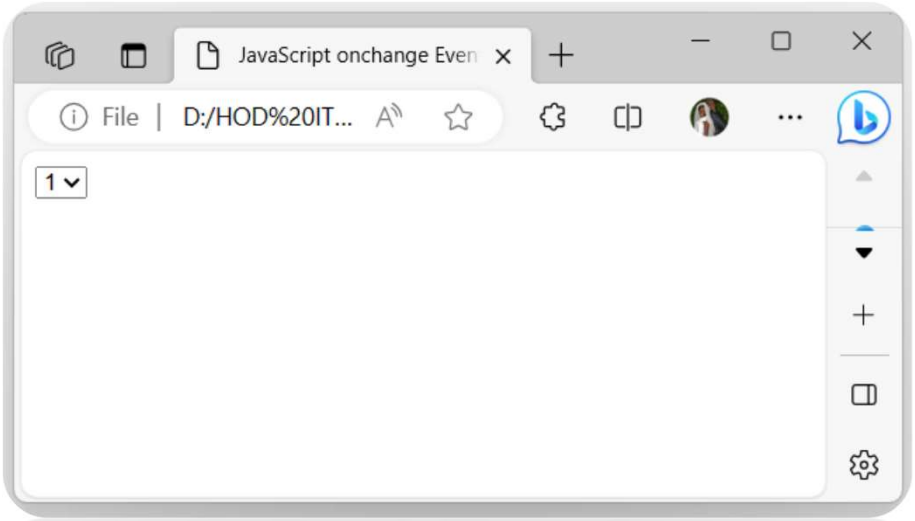
JavaScript onchange Event Example+

MultipleJS_Files.html | mouseEvents.html | onChangeEvent.html

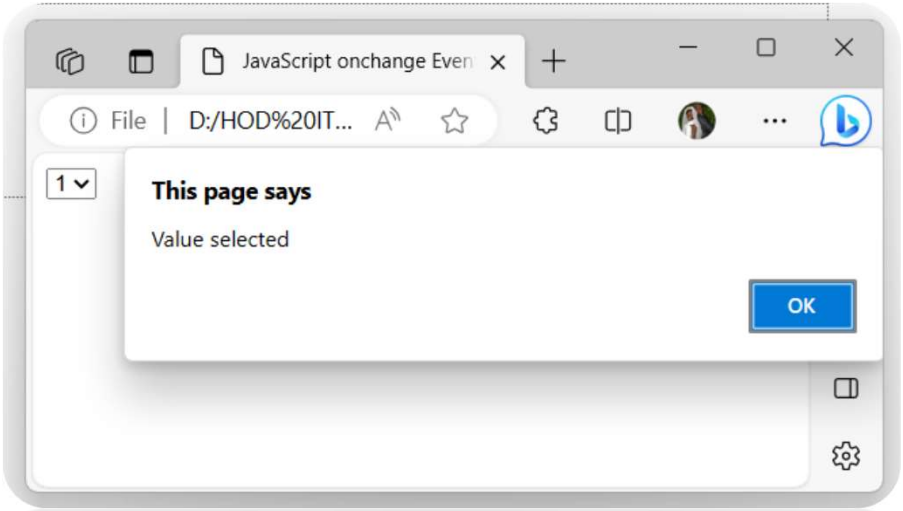
```
<!doctype html>
    <head>
        <title>JavaScript onchange Event</title>
    </head>
    <body>
        <select id="dropdown">
            <option value="1">1</option>
            <option value="2">2</option>
            <option value="3">3</option>
        </select>
        <script>
            /* JS comes here */
            function eventHandler() {
                // do something
                alert("Value selected");
            }
            // setup event handler on HTML element
            document.getElementById("dropdown").onchange = eventHandler;
        </script>
    </body>
</html>
```

JavaScript Event Handling

JavaScript **onchange** Event Example+ onChange.html file



Output before selection of an item



Output after selection of an item

JavaScript Event Handling

JavaScript onClick Event Example

This event occurs when a user clicks on a button or clicks any HTML element on which we have set the event handler.

In the **example below**, we have set a simple **onClick** handler in the <button> HTML tag.

JavaScript Event Handling

JavaScript onClick Event Example+Code

```
<!doctype html> <html><head>
      <title>JavaScript onClick Example</title>
       <script>
           // event handler function
           function doSomething()
           {
               alert("Event triggered");
           }
       </script>

</head>
<body>
      <p>This button has an Event Handler...</p>
      <!-- onClick attribute in HTML -->
      <button id="btn" onClick="doSomething()">Do Something!</button>

</body> </html>
```

JavaScript Event Handling

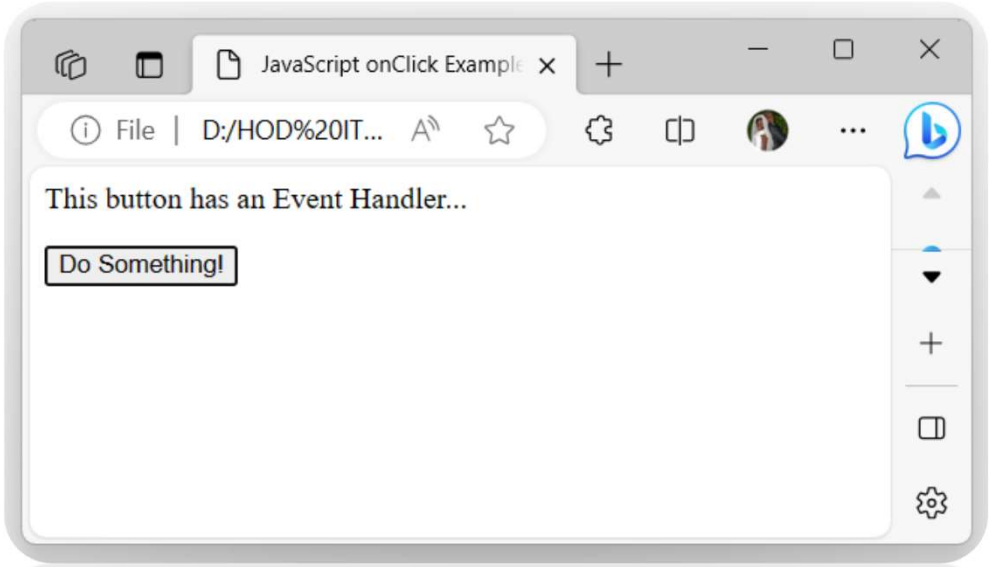
JavaScript onClick Event Example+Code

MultipleJS_Files.html | mouseEvents.html | onChangeEvent.html | onClickEvent.html | onblurEvent.html | emailVAlidator.html | Sample.html

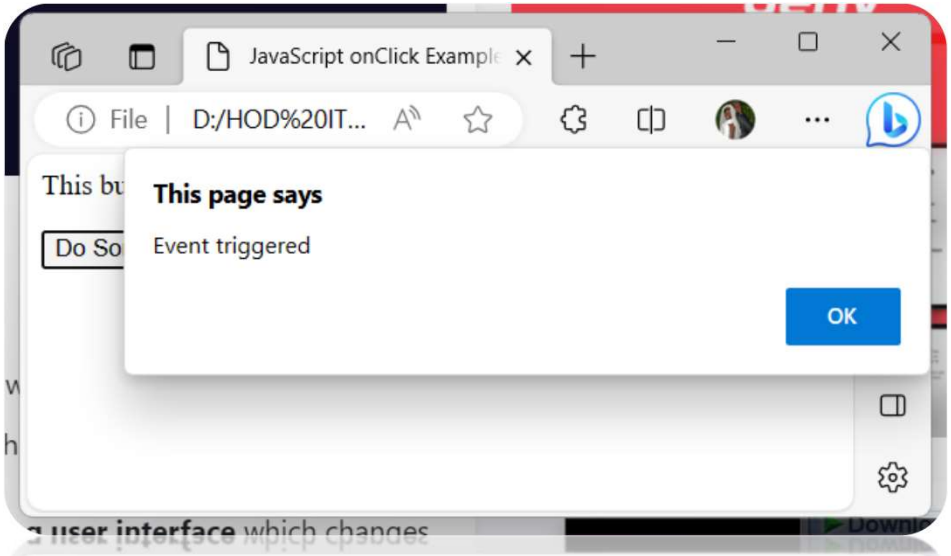
```
1  <html>
2      <head>
3          <title>JavaScript onClick Example</title>
4          <script>
5              // event handler function
6              function doSomething()
7              {
8                  alert("Event triggered");
9              }
10         </script>
11     </head>
12     <body>
13         <p>This button has an Event Handler...</p>
14         <!-- onClick attribute in HTML -->
15         <button id="btn" onClick="doSomething()">Do Something!</button>
16     </body>
17 </html>
```

JavaScript Event Handling

JavaScript onClick Event Example+Output



Output before button click



When the button is clicked, the event is triggered. Click ok to close

JavaScript Event Handling

JavaScript onMouseOver Event Example+

You can make your webpage more interactive by using the **onmouseover** event. When we set this event handling on any HTML element, whenever the user takes the mouse cursor over that HTML element, then the event handler is triggered.

We can use this event to change **color**, **size** of **text**, or in general **create an interesting user interface** which changes on mouse hover.

JavaScript Event Handling

JavaScript onMouseOver Event Example

```
<!doctype html> <html><head>
     <title>JavaScript onmouseover Event</title>
     <script>
     function over() {
          document.write("Mouse Over");
     }
     </script>

</head>
<body>
      <h2 onmouseover="over()">Hover mouse here!</h2>
</body>
</html>
```

JavaScript Event Handling

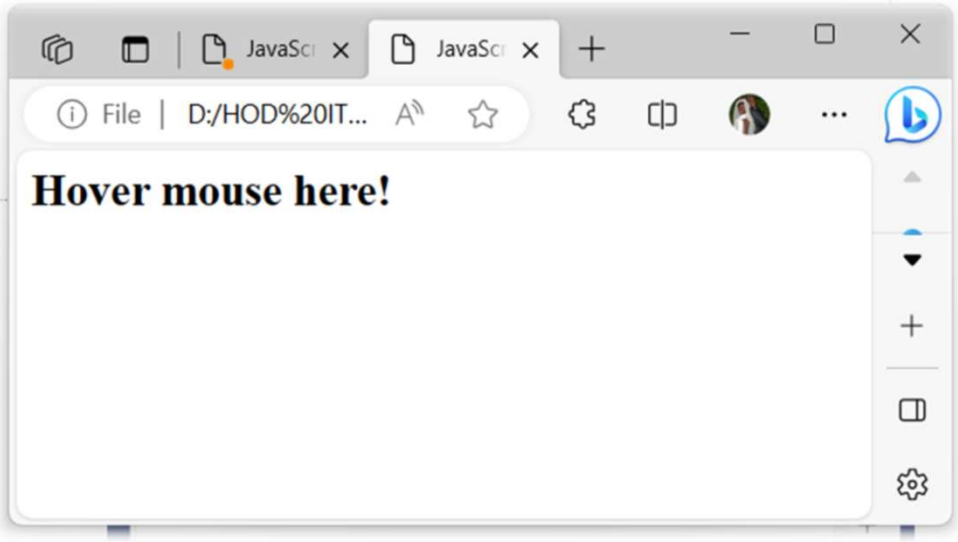
JavaScript **onMouseOver** Event Example+Code

MultipleJS_Files.html | mouseEvents.html | onChangeEvent.html | onClickEvent.html | onMouseOver.html

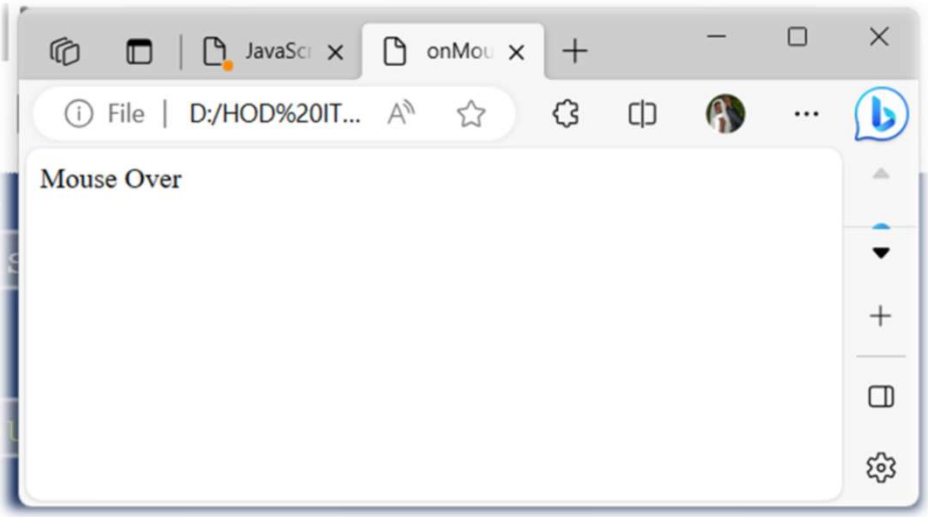
```
1  <!doctype html> <html><head>
2      <head>
3          <title>JavaScript onmouseover Event</title>
4          <script>
5              function over() {
6                  document.write("Mouse Over");
7              }
8          </script>
9      </head>
10     <body>
11         <h2 onmouseover="over()">Hover mouse here!</h2>
12     </body>
13 </html>
```

JavaScript Event Handling

JavaScript **onMouseOver** Event Example+Output



Display before Mouse Hover



Output after Mouse hover

JavaScript Event Handling

JavaScript **onblur** Event Example+Output

The "**onblur**" event handler attribute is utilized in HTML and JavaScript to designate a JavaScript function that should run when an HTML element loses focus. This "blur" event is triggered when a user clicks outside of an input field or moves away from it, causing the element to no longer be the active focus.

Here's a typical usage of the "**onblur**" attribute:

HTML Attribute: You can include the "onblur" attribute within an HTML element, commonly applied to form elements like input fields or text areas. This attribute allows you to specify a JavaScript function that will be executed when the element loses focus.

JavaScript Event Handling

JavaScript **onblur** Event Example+code

```
<!doctype html> <html><head>
      <title>JavaScript onblur Event</title>
      <script>
      function wait() {
            document.getElementById("result").innerHTML = "Please enter a
value...";

      }
      </script>

</head>
<body>
      <p>Click in the Input Field and then click outside.</p>
        <input type="text" onblur="wait()" placeholder="Your Email..." />
        <div id="result"></div>
</body>
</html>
```

JavaScript Event Handling

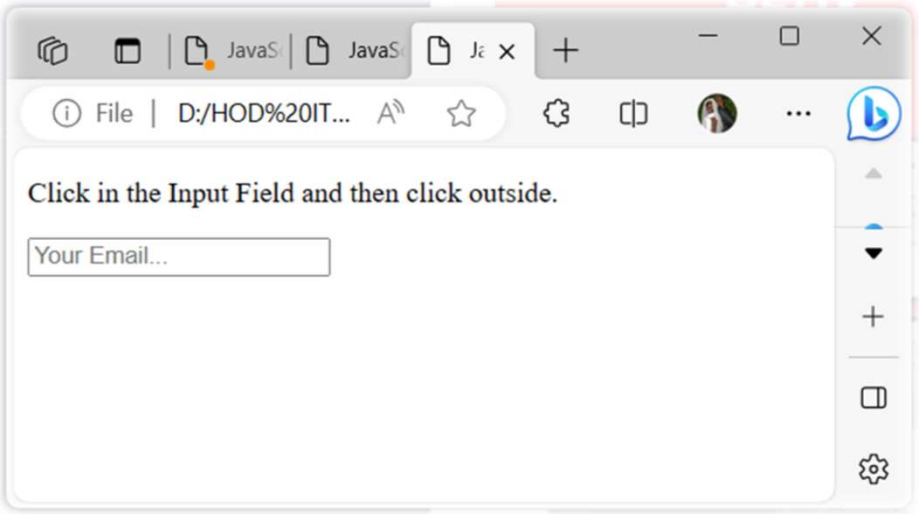
JavaScript **onblur** Event Example+code

MultipleJS_Files.html | mouseEvents.html | onChangeEvent.html | onClickEvent.html | onblurEvent.html

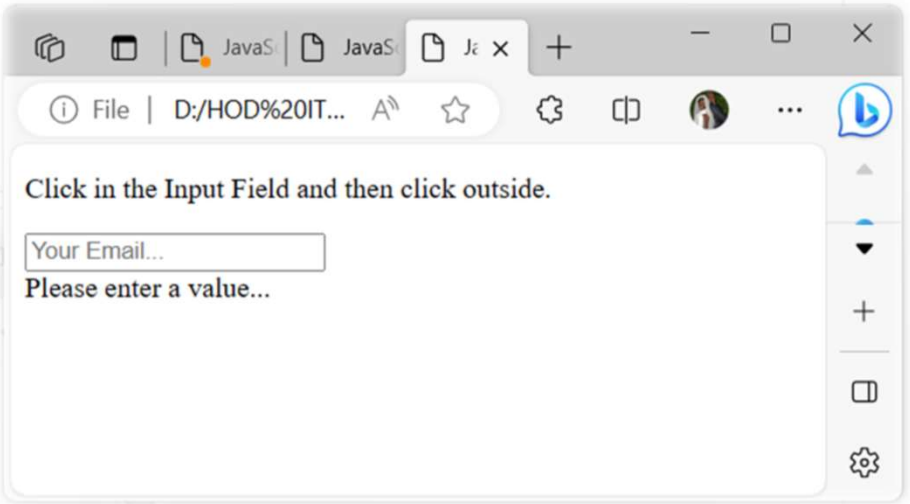
```
1  <!doctype html> <html><head>
2      <head>
3          <title>JavaScript onblur Event</title>
4          <script>
5              function wait()
6              {
7                  document.getElementById("result").innerHTML = "Please enter a value...";
8              }
9          </script>
10     </head>
11     <body>
12         <p>Click in the Input Field and then click outside.</p>
13         <input type="text" onblur="wait()" placeholder="Your Email..." />
14         <div id="result"></div>
15     </body>
16 </html>
```

JavaScript Event Handling

JavaScript **onblur** Event Example+Output



Output before clicking email text box



Output after clicking email text box and clicking outside the text box

NB: Should the user try to click the email text box and click anywhere else without entering the email, the message is displayed asking the user to enter the email.

JavaScript events used with HTML forms

Below we have listed down all the events which can be used with HTML forms.

Event	Description
onsubmit	triggers on submission of a form
onselect	triggers on selecting an element
oninvalid	triggers, when an element is, is in invalid
oninput	triggers when input is provided for an element
onforminput	triggers when input is provided on a form

 (https://www.studytonight.com/javascript/include-javascript-in-html)

JavaScript events used with HTML forms

Below we have listed down all the events which can be used with HTML forms.

Event	Description
onforminput	triggers when input is provided on a form
onformchange	triggers when a form changes
onfocus	triggers when a window get focus
oncontextmenu	triggers when the context menu is used
onchange	triggers when an element changes
onblur	triggers when a window loses focus.

 (https://www.studytonight.com/javascript/include-javascript-in-html)

JavaScript Keyboard events

Following are events available for Keyboard key press.

Event	Description
onkeydown	triggers on pressing a key
on keypress	triggers when the key is pressed
onkeyup	triggers on releasing a key

(https://www.studytonight.com/javascript/include-javascript-in-html)

JavaScript **Mouse** events used in HTML

We have already covered an example for **onmouseover** event above, these are other mouse events available. Try using these too and see what they can do.

Event	Description
onclick	triggers on clicking the mouse button
ondblclick	triggers on double-clicking the mouse button
ondrag	triggers on dragging an element
ondrop	triggers on dropping the dragged element
onscroll	triggers on scrolling the scroll bar of an element

 (https://www.studytonight.com/javascript/include-javascript-in-html)

JavaScript **Mouse** events used in HTML

We have already covered an example for **onmouseover** event above, these are other mouse events available. Try using these too and see what they can do.

Event	Description
onmouseup	triggers on releasing the mouse button
onmousewheel	triggers on rotating the mouse wheel
ondragleave	triggers on leaving the valid target while dragging an element
onmousedown	triggers on pressing the mouse button
onmousemove	triggers on moving the mouse pointer

JavaScript Events of Browser

Following are the browser events which you can use in your JavaScript code.

Events	Description
onblur	triggers when a window loses focus
onerror	triggers when an error occurs
onfocus	triggers when a window get focus
onload	triggers at the time of loading a document
onmessage	triggers when the postMessage() method sends a message to the current document

JavaScript Events of Browser+

Following are the browser events which you can use in your JavaScript code.

Events	Description
onmessage	triggers when the postMessage() method sends a message to the current document
onafterpoint	triggers after printing a document
onbeforepoint	triggers before printing a document
onhaschange	triggers at the time of changing a document
onredo	triggers at the time of performing redo action in a document
ononline	triggers when a document gets online

(https://www.studytonight.com/javascript/include-javascript-in-html)

JavaScript Events for HTML Media Elements

Below are the events which can be used to manage various media types in an HTML webpage like video pause, video play etc.

Events	Description
onwaiting	triggers when the media file has stopped but is expected to resume later
onpause	triggers on pausing the media file
onplay	triggers when a media file is going to play
onseeking	triggers when the seeking process begins
onabort	triggers on aborting a process

(https://www.studytonight.com/javascript/include-javascript-in-html)

JavaScript Events for HTML Media Elements+

Below are the events which can be used to manage various media types in an HTML webpage like video pause, video play etc.

Events	Description
onemptied	triggers when a media element becomes empty due to some errors
onended	triggers when the media file ends
onloadeddata	triggers on loading the media file
ontimeupdate	triggers on changing the playing position of the media file
onvolumechange	triggers on changing the volume

(https://www.studytonight.com/javascript/include-javascript-in-html)

Using JavaScript Events in Web Development

Events play a crucial role in crafting dynamic and interactive web applications. The process of application development is rich with events, including those for:

1. Initializing the application
2. Responding to button clicks
3. Handling user form submissions
4. Managing textbox focus
5. Reacting to element selections, and more.

For web developers, harnessing JavaScript events is key to enhancing webpage responsiveness, implementing client-side validations, and seamlessly managing user interactions. We've provided a list of these events, giving you the opportunity to explore and understand their functionality. Feel free to experiment with these events in the live examples above to see how they function in action.

The JavaScript HTML DOM

The JavaScript HTML DOM (Document Object Model) is a programming interface provided by browsers that allows JavaScript to interact with and manipulate HTML documents.

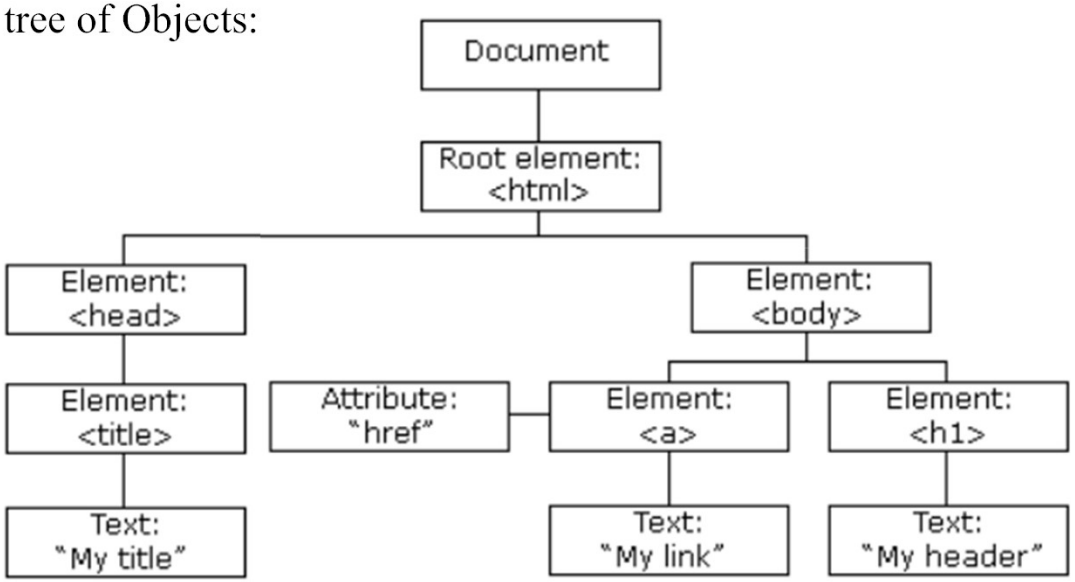
It represents the web page as a structured tree of objects, where each element, attribute, and text node in the HTML document is treated as an object.

Developers can use the DOM to access and modify the content, structure, and styles of a web page dynamically, making it a fundamental part of web development.

The JavaScript HTML DOM

With the HTML DOM, JavaScript can access and change all the elements of an HTML document. When a web page is loaded, the browser creates a Document Object Model of the page.

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



(https://www.w3schools.com/js/js_htmldom.asp)

JavaScript - HTML DOM Methods

JavaScript HTML DOM methods are functions provided by the Document Object Model (DOM) API that allow you to interact with and manipulate HTML documents in web development.

These methods provide a way to access and **modify** the **structure**, **content**, and **behavior** of HTML elements on a web page.

Here are some commonly used JavaScript HTML DOM methods:

1. **getElementById(id):** Retrieves an element with a specific ID.

```
var element = document.getElementById("myId");
```

JavaScript - HTML DOM Methods

2. getElementsByClassName(className): Returns a collection of elements with a specific class name.

```
var elements = document.getElementsByClassName("myClass");
```

3. getElementsByTagName(tagName): Returns a collection of elements with a specific HTML tag.

```
var elements = document.getElementsByTagName("p");
```

JavaScript - HTML DOM Methods

4. querySelector(selector): Retrieves the first element that matches the specified CSS selector.

```
var element = document.querySelector(".myClass");
```

5. querySelectorAll(selector): Retrieves all elements that match the specified CSS selector.

```
var elements = document.querySelectorAll("p.myClass");
```

JavaScript - HTML DOM Methods

6. createElement(tagName): Creates a new HTML element.

```
var newElement = document.createElement("div");
```

7. appendChild(element): Appends an element as a child of another element.

```
parentElement.appendChild(newElement);
```

8. removeChild(element):Removes a child element from its parent.

```
parentElement.removeChild(childElement);
```

JavaScript - HTML DOM Methods

9. **setAttribute(attribute, value):** Sets the value of an attribute for an element.

```
element.setAttribute("class", "newClass");
```

10. **getAttribute(attribute):** Retrieves the value of an attribute for an element.

```
var value = element.getAttribute("src");
```

11. **addEventListener(event, function):** Attaches an event listener to an element.

```
element.addEventListener("click", function() {
  // Handle click event
});
```

JavaScript - HTML DOM Methods

12. **style.property = value:** Changes the CSS style of an element.

```
element.style.color = "red";
```

13. **innerHTML:** Gets or sets the HTML content of an element.

```
element.innerHTML = "New content";
```

14. **innerText:** Gets or sets the text content of an element.

```
element.innerText = "New text";
```

JavaScript - HTML DOM Methods

15. **classList:** Provides methods to manipulate an element's classes.

```
element.classList.add("newClass");
element.classList.remove("oldClass");
element.classList.toggle("active");
```

These methods are essential for creating dynamic, interactive web pages and web applications by enabling you to manipulate the DOM structure and content using JavaScript. They form the foundation for many web development tasks, from simple animations to complex web applications.

JavaScript HTML DOM Document

The HTML DOM document object is the owner of all other objects in your web page.

The 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.
- Below are some examples of how you can use the document object to access and manipulate HTML.

JavaScript HTML DOM Document
Finding HTML Elements

You can find HTML Elements using the following methodes

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

JavaScript HTML DOM Document

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

JavaScript HTML DOM Document
Finding HTML Objects

The first HTML DOM Level 1 (1998), defined 11 HTML objects, object collections, and properties. These are still valid in HTML5.

Later, in HTML DOM Level 3, more objects, collections, and properties were added.

Property	Description	DOM
document.anchors	Returns all <a> elements that have a name attribute	1
document.applets	Deprecated	1
document.baseURI	Returns the absolute base URI of the document	3
document.body	Returns the <body> element	1
document.cookie	Returns the document's cookie	1
document.doctype	Returns the document's doctype	3

JavaScript HTML DOM Document
Finding HTML Objects

Property	Description	DOM
document.documentElement	Returns the <html> element	3
document.documentMode	Returns the mode used by the browser	3
document.documentURI	Returns the URI of the document	3
document.domain	Returns the domain name of the document server	1
document.domConfig	Obsolete.	3
document.embeds	Returns all <embed> elements	3

JavaScript HTML DOM Document
Finding HTML Objects+

Property	Description	DOM
document.forms	Returns all <form> elements	1
document.head	Returns the <head> element	3
document.images	Returns all <img> elements	1
document.implementation	Returns the DOM implementation	3
document.inputEncoding	Returns the document's encoding (character set)	3
document.lastModified	Returns the date and time the document was updated	3

JavaScript HTML DOM Document
Finding HTML Objects++

Property	Description	DOM
document.links	Returns all <area> and <a> elements that have a href attribute	1
document.readyState	Returns the (loading) status of the document	3
document.referrer	Returns the URI of the referrer (the linking document)	1
document.scripts	Returns all <script> elements	3
document.strictErrorChecking	Returns if error checking is enforced	3
document.title	Returns the <title> element	1
document.URL	Returns the complete URL of the document	1

Form validation and user input handling

Form validation and user input handling

Form validation and user input handling are critical aspects of web development, ensuring that data submitted by users through forms is accurate, complete, and secure.

JavaScript plays a central role in implementing client-side form validation and handling user inputs.

Here's an overview of how to perform form validation and handle user input using JavaScript:

Form validation and user input handling

Here's an overview of how to perform form validation and handle user input using JavaScript:

1. HTML Form Setup:
2. JavaScript Validation Functions
3. Event Handling
4. Providing User Feedback
5. Server-Side Validation

Form validation and user input handling

How to perform form validation and handle user input using JavaScript

1. HTML Form Setup:

Start by creating an HTML form that collects user input. Define form fields, such as text inputs, checkboxes, radio buttons, and select boxes, and include appropriate form elements like labels and submit buttons.

Form validation and user input handling+ 1 Create HTML Form Setup

MultipleJS_Files.html | mouseEvents.html | onChangeEvent.html | formUserValidate.html | onblurEvent.html | emailVAlidator.html | Samp

```
1  <html>
2      <head>
3          <title>Form & User input Validation</title>
4          <script>
5
6          </script>
7      </head>
8      <body>
9          <form id="myForm">
10           <label for="name">Name:</label>
11           <input type="text" id="name" name="name" required>
12           <br>
13           <label for="email">Email:</label>
14           <input type="email" id="email" name="email" required>
15           <br>
16           <input type="submit" value="Submit">
17         </form>
18
19     </body>
20 </html>
```

Form validation and user input handling+ 2. JavaScript Validation Functions

```
4       <script>
5           function validateForm() {
6               var name = document.getElementById("name").value;
7               var email = document.getElementById("email").value;
8
9               if (name === "") {
10                  alert("Name must be filled out");
11              return false;
12              }
13
14              if (!isValidEmail(email)) {
15                  alert("Invalid email address");
16                  return false;
17              }
18
19              return true;
20              }
21
22          function isValidEmail(email) {
23              var emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
24          return emailRegex.test(email);
25          }
26          </script>
```

Write JavaScript functions to validate the form inputs. Common validation checks include checking for empty fields, verifying email formats, ensuring password complexity, and validating numeric inputs.

Form validation and user input handling+ 3. Event Handling

Attach an event listener to the form's submit button or the submit event of the form. This listener will invoke the validation function before allowing the form to be submitted.

```
26          //Attach event listener to the form
27          var form = document.getElementById("myForm");
28              form.addEventListener("submit", function(event) {
29              if (!validateForm()) {
30          event.preventDefault(); // Prevent form submission if validation fails
31          }
32          });
```

Form validation and user input handling+ 4 Providing User Feedback:

```
34          //Validate the form
35          function validateForm() {
36            var name = document.getElementById("name").value;
37            var email = document.getElementById("email").value;
38            var nameError = document.getElementById("name-error");
39            var emailError = document.getElementById("email-error");
40
41            nameError.textContent = "";
42            emailError.textContent = "";
43
44            if (name === "") {
45              nameError.textContent = "Name must be filled out";
46              return false;
47            }
48
49            if (!isValidEmail(email)) {
50              emailError.textContent = "Invalid email address";
51              return false;
52            }
53
54            return true;
55          }
```

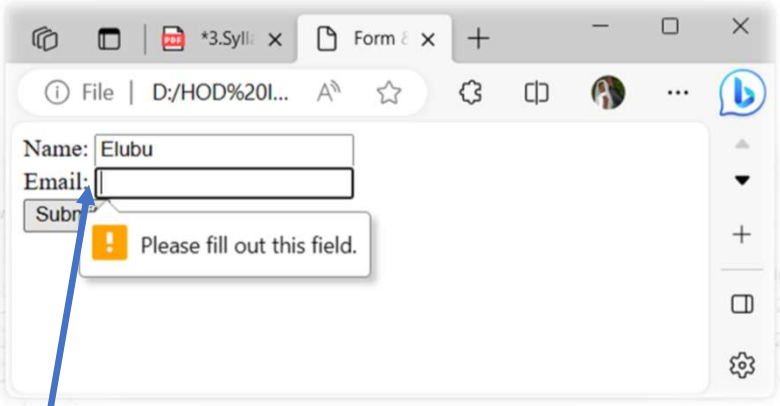
When validation fails, provide feedback to the user by displaying error messages near the problematic fields or using alert() boxes. Use CSS to style error messages and make them visually distinct.

Form validation and user input handling+ 5. Server-Side Validation:

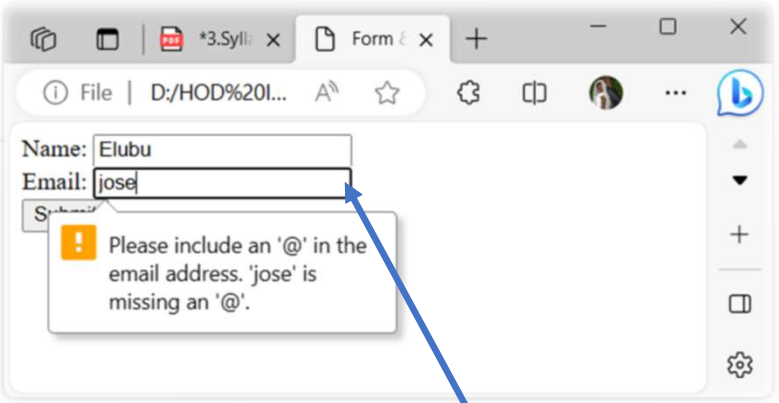
Client-side validation should be considered as a user experience improvement, but it should not replace server-side validation. Always validate user inputs on the server to ensure security and data integrity.

By implementing these steps, you can create client-side form validation and user input handling in your web applications to enhance user experience and ensure the accuracy and completeness of data submitted through forms.

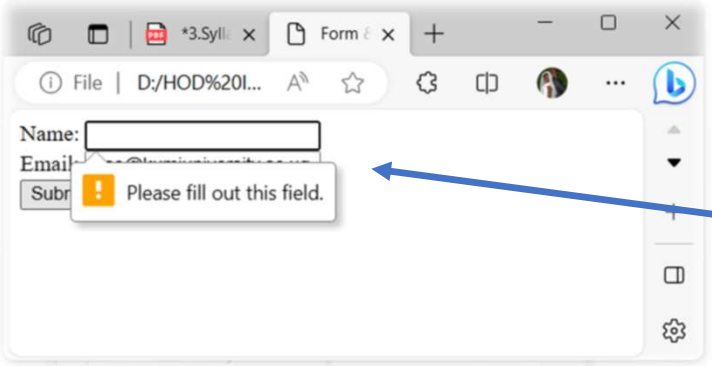
Form validation and user input handling+ formUserValidate.html file Output



Email field empty



Invalid email entered



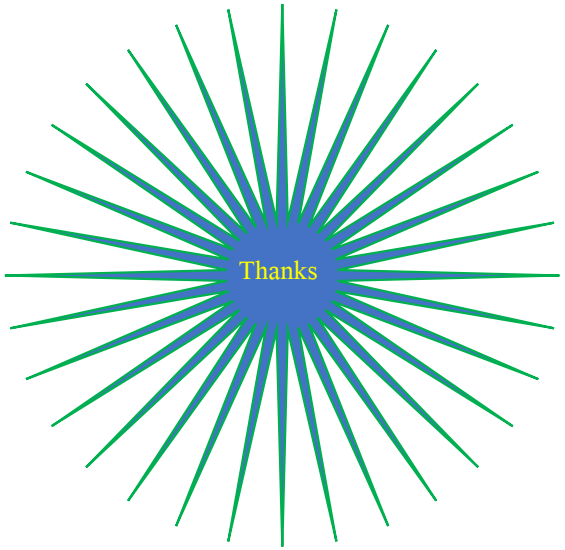
Name field empty

Summary

Summary

1. Summary of previous lecture
2. JavaScript in HTML(script tag attribute and their uses)
3. Event handling(Definition, mouse events, Keyboard event, JavaScript HTML Media Elements, Created operational events etc)
4. DOM manipulation(Definition, Structure, methods, DOM document objects, Finding HTML Objects,)

Thank you for Listening



References

Include JavaScript in HTML. Studytonight.com. (n.d.). https://www.studytonight.com/javascript/include-javascript-in-html

ECMA-262. Ecma International. International , E. (2023, June 30). https://www.ecma-international.org/publications-and-standards/standards/ecma-262/

Web app development in 2023: Everything you need to know. Brewster, C. (2023). Trio. https://www.trio.dev/front-end/resources/web-app-development

JavaScript Data Types. OpenAI Knowledge Base. OpenAI. (2021, September 22)..