CS213: Software Systems Laboratory

Autumn 2023-24

Koteswararao Kondepu

k.kondepu@iitdh.ac.in

Recap

- CSS Bootstrap: Forms
  - Form controls
  - Checks
  - Range
  - Floating labels
  - Validation

{property}{sides}-{size}

<input>s and ***<textarea>***

```
<div class="form-check">
```

form-check-input

```
type="range" class="form-range"
<div class="form-floating mb-3">
id="floatingSelect"
<div class="valid-feedback">
```

- CSS Bootstrap: Components
  - Alerts
  - Dropdown
  - Badge
  - Collapse
  - List group

```
<div class="alert alert-primary" role="alert">
<div class="dropdown">
class="badge text-bg-secondary"
class="btn-group" role="group
class="collapse"
class="list-group"
```

https://getbootstrap.com/docs/5.3/getting-started/introduction/

Outline

- Introduction Java Script (JS)
- JavaScript Linking
- JavaScript Document Object Model (DOM)
- JavaScript DOM Tree
- JavaScript DOM Nodes/Objects

What is JavaScript

- A lightweight programming language (“scripting language”)
  - Created in 1995 by Brandon Eich of Netscape/Mozilla
  - Originally called "LiveScript" to match Netscape branding
  - Renamed to JavaScript to capitalize on the popularity of Java
  - Submitted as a standard to ECMA in 1997 as "ECMAScript
  - Used to make web pages interactive
  - Insert dynamic text into HTML (e.g., user name)
  - React to events (e.g., page load user click)
  - NOT related to Java other than by name and some syntactic similarities

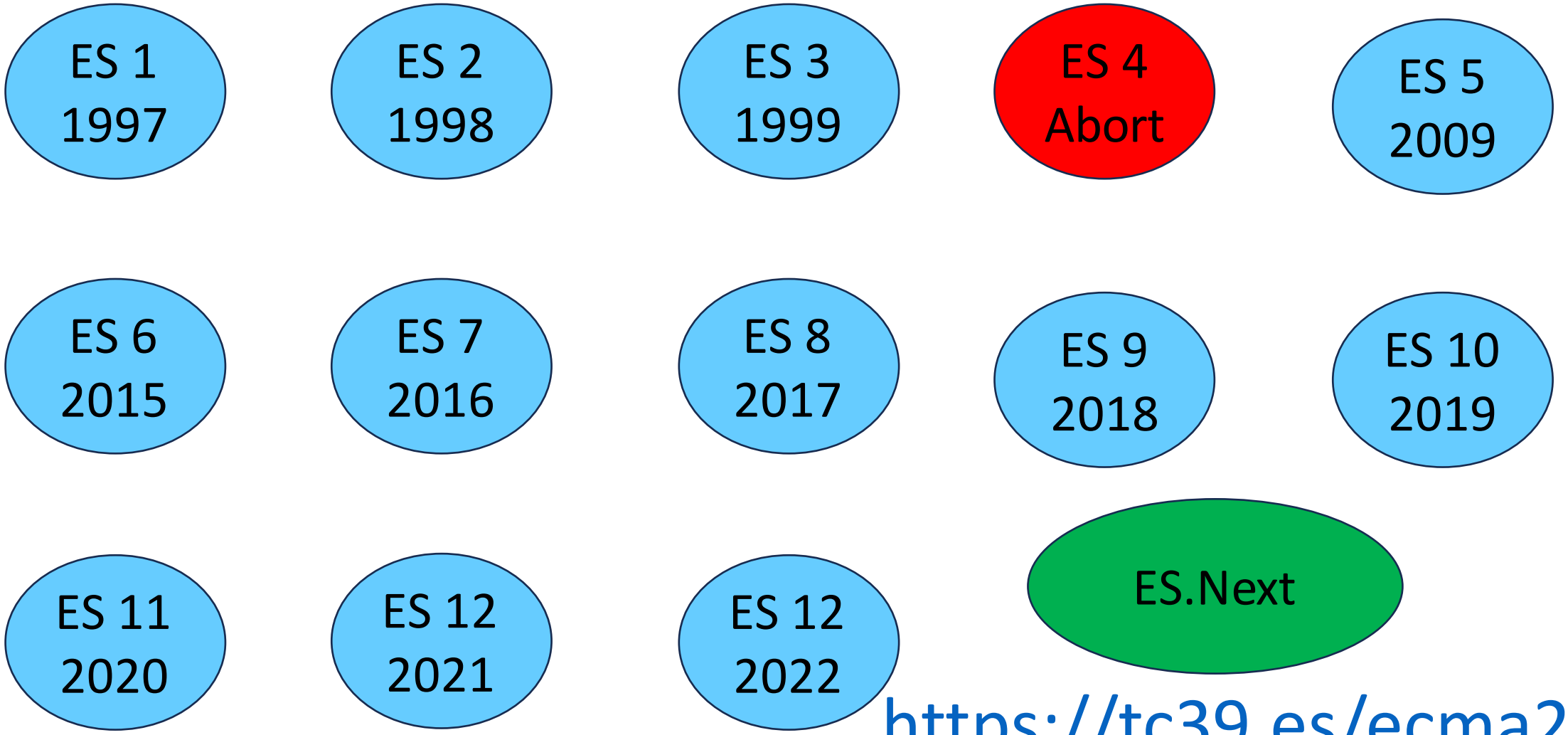
What is JavaScript (1)

- Possibly the most used programming language today (!!)
  - mostly used → client-side web page scripting, but increasingly used to build server apps, other programs → node.js
  - current standardized version: ECMAScript 5 (2009)
  - JS in browser works with "DOM" (Document Object Model)

- Client-side scripting (JavaScript) benefits:
  - **usability**: can modify a page without having to post back to the server
  - **efficiency**: can make small, quick changes to page without waiting for server
  - **event-driven**: can respond to user actions like clicks and key presses

JavaScript Versions

ES: ECMAScript



https://tc39.es/ecma262/

Overview

HTML

- Creates the content

CSS

- Change the appearance/styling

JavaScript

- Interactive/dynamic content

Document Object Model (DOM)

- **DOM**
  - A standard defined by the World Wide Web Consortium (W3C) for accessing documents.
  - A platform and language-neutral interface that allows programs and scripts to *dynamically access and update the content, structure, and style of a document*.
  - **It is one of the most unique and useful tools of JavaScript**

- It has all the power that needs to create a dynamic HTML
  - Creates new HTML events on the page.
  - Removes the existing HTML elements and attributes.
  - Changes all the HTML elements on the page.

Document Object Model (DOM)

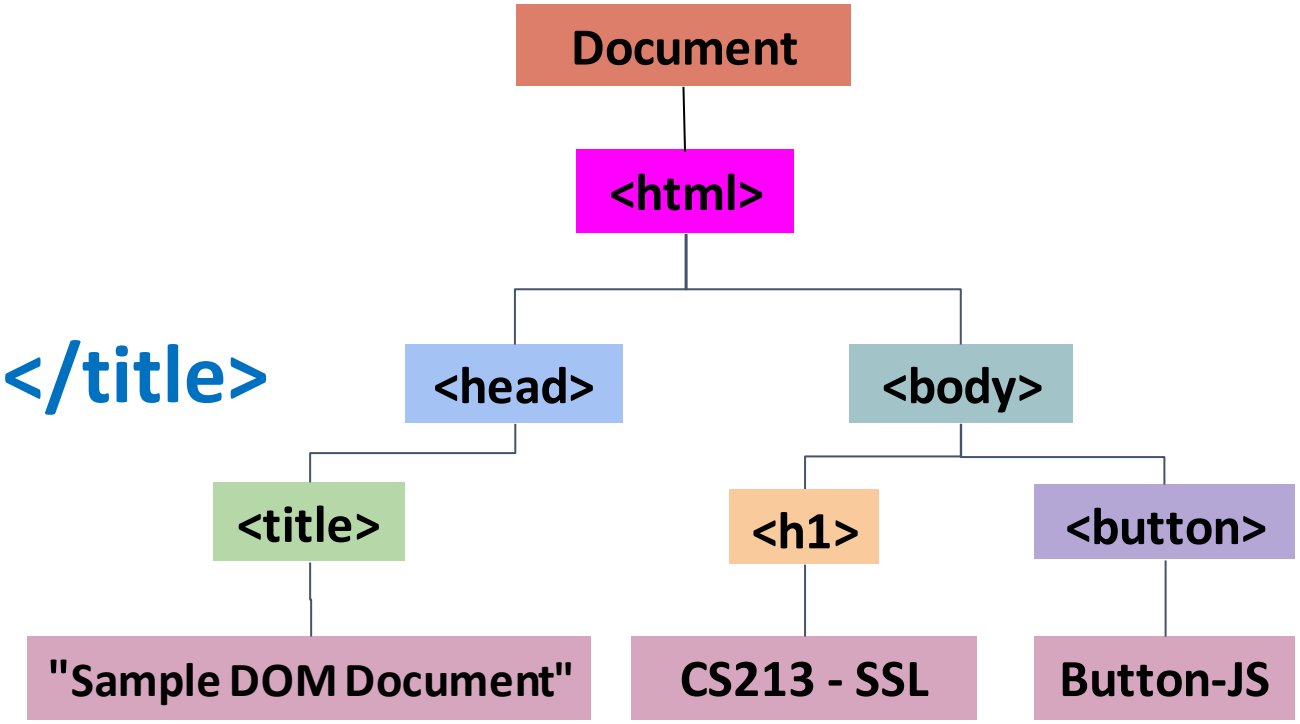
- DOM Programming Interface
  - HTML elements → Objects
  - Changing the content of an HTML element → Property
  - Adding or deleting an HTML element → Method
  - Provides representation of the HTML hierarchical data.
  - Examples:

    Properties: document.alinkColor, document.URL, document.forms[ ], document.links[ ], document.anchors[ ], …

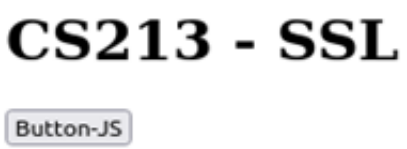
    Methods: document.write(document.referrer) – These change the content of the page!

DOM: Tree Example

```
<!DOCTYPE html>
<html>
 <head>
  <title>Sample DOM Document</title>
 </head>
 <body>
  <h1>CS213 - SSL</h1>
  <button>Button-JS</body>
</html>
```



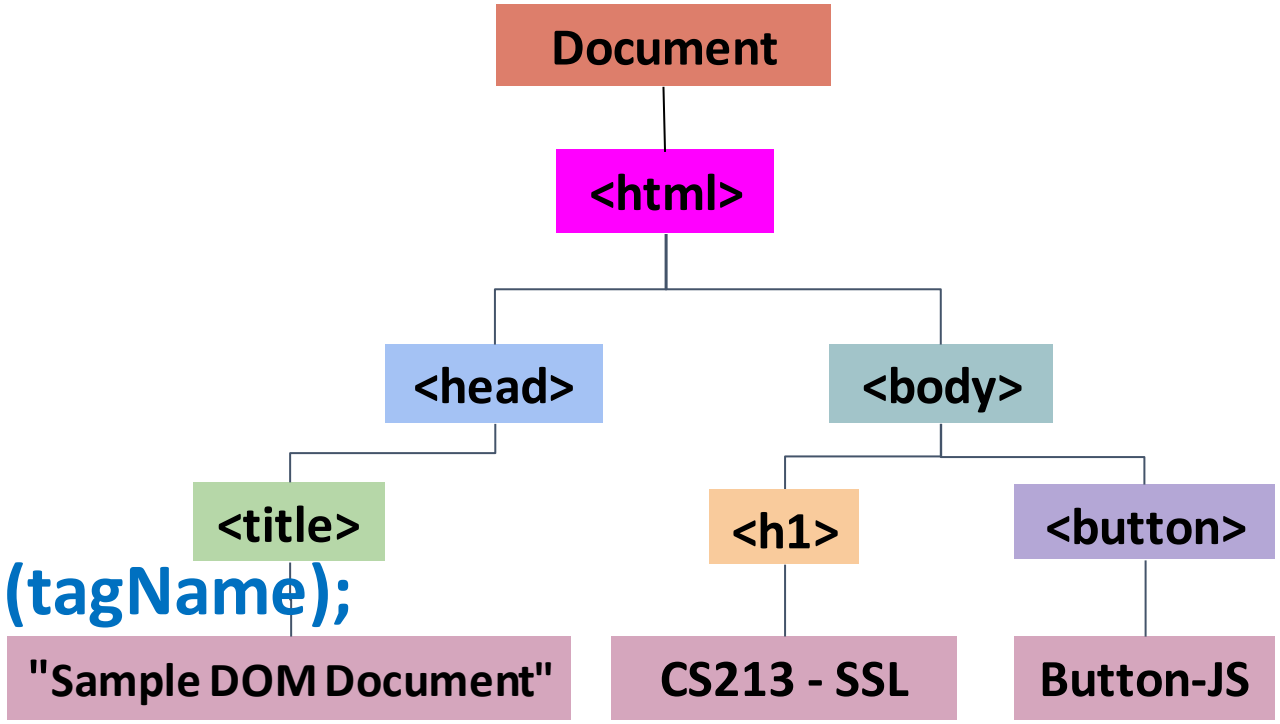
HTML DOM Tree



- The DOM Tree represents HTML document as nodes.
- Each node is referred to as an **Object**

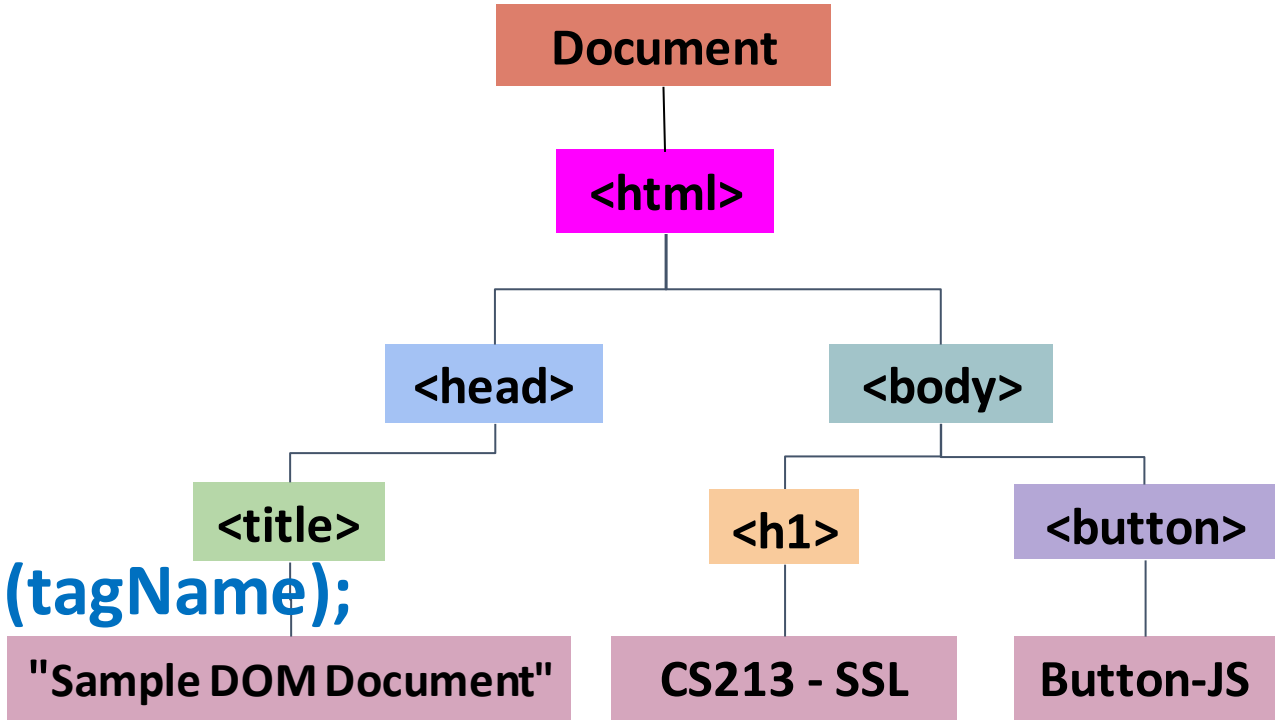
DOM: Nodes/Objects

- DOM Nodes Accessing Methods
  - Finding DOM nodes by id:
    document.getElementById(id);
  - Finding DOM nodes by tag name:
    document.getElementsByTagName(tagName);
  - Finding DOM nodes by class name:
    document.getElementsByClassName(className);
  - Finding DOM nodes by queryselector:
    document.querySelector(cssQuery);
    document.querySelectorAll(cssQuery);



DOM: Nodes/Objects

- DOM Nodes Accessing Methods
  - Finding DOM nodes by id:
    document.getElementById(id);
  - Finding DOM nodes by tag name:
    document.getElementsByTagName(tagName);
  - Finding DOM nodes by class name:
    document.getElementsByClassName(className);
  - Finding DOM nodes by queryselector:
    document.querySelector(cssQuery);
    document.querySelectorAll(cssQuery);



Linking to JavaScript

```
<!DOCTYPE html>
<html>
<head>
    <title>
        Welcome to CS213
    </title>
     <script>
        function myFun() {
            document.getElementById("demo")
                .innerHTML = "Welcome to CS213 from PC!";
        }
    </script>
</head>
 <body>
   <h2 id="demo" style="color:blue;">
        Welcome to CS213
   </h2>
   <button type="button" onclick="myFun()"> Click Here </button>
</body>
</html>
```

Linking to JavaScript (1)

```
<!DOCTYPE html>
<html>
<head>
    <title>
        Welcome to CS213
    </title>
</head>
 <body>
   <h2 id="demo" style="color:blue;">
        Welcome to CS213
   </h2>
   <button type="button" onclick="myFun()"> Click Here </button>
   <script>
     function myFun() {
       document.getElementById("demo")
         .innerHTML = "Welcome to CS213 from PC!"
     }
    </script>
</body>
</html>
```

Linking to JavaScript (2)

```
<!DOCTYPE html>
<html>
<head>
    <title>
        Welcome to CS213
    </title>
</head>
 <body>
   <h2 id="demo" style="color:blue;">
        Welcome to CS213
   </h2>
   <button type="button" onclick="myFun()"> Click Here </button>

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

</body>
</html>
```

myscript.js

```
function myFun() {
   document.getElementById("demo")
    .innerHTML = "Welcome to CS213
from PC!";
    }
```

DOM Nodes/Objects: Example#1

getElementById(id);

```
<html>
    <head> <title> welcome to CS213 </title> </head>
<body>
<p id="demo"></p>
<script>
document.getElementById("demo").innerHTML = "Welcome to CS213
Lecture";
</script>
</body>
</html>
```

Welcome to CS213 Lecture

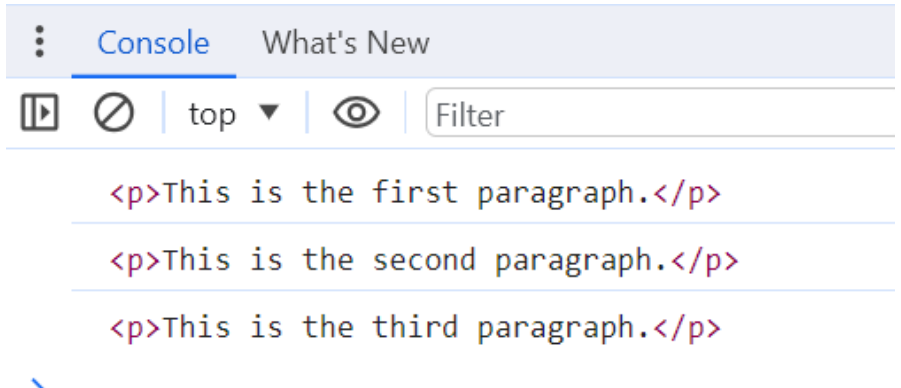
Note: **getElementById → a method**,

innerHTML → a property.

DOM Nodes/Objects: Example#2

```
<!DOCTYPE html>
<html>
<body>
    <p>This is the first paragraph.</p>
    <p>This is the second paragraph.</p>
    <p>This is the third paragraph.</p>
    <script>
        // Get all <p> elements
        var elements = document.getElementsByTagName('p');
        // Loop through the elements and print them to the console
        for (var i = 0; i < elements.length; i++) {
            console.log(elements[i]); // This prints the elements in
browser console
        }
    </script>
</body>
</html>
```

getElementsByTagName(tagName);



This is the first paragraph.

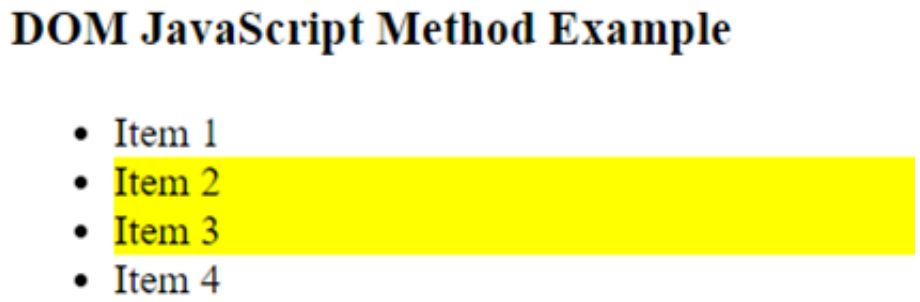
This is the second paragraph.

This is the third paragraph.

DOM Nodes/Objects: Example#3

```
<!DOCTYPE html>
<html>
<body>
    <ul>
        <li>Item 1</li>
        <li class="highlight">Item 2</li>
        <li class="highlight">Item 3</li>
        <li>Item 4</li>
    </ul>
    <script>
        // Get all elements with class "highlight"
        var elements = document.getElementsByClassName('highlight');
        // Apply the style to each element
        for (var i = 0; i < elements.length; i++) {
            elements[i].style.background = "yellow";
        }
    </script>
</body>
</html>
```

getElementsByTagName(tagName);

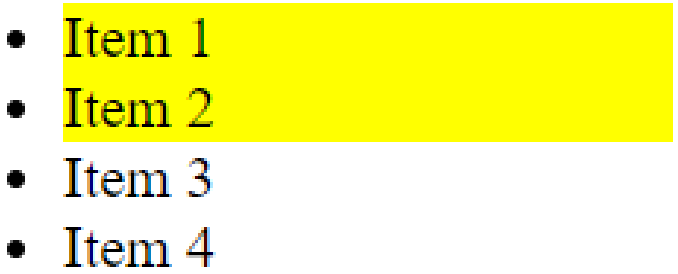


Note: The **getElementsByClassName()** method in Javascript returns an object containing all the elements. They can be accessed using an index, and it starts from 0-indexing.

DOM Nodes/Objects: Example#4

querySelectorAll();

```
<!DOCTYPE html>
<html>
<body>
    <ul>
        <li class="highlight">Item 1</li>
        <li class="highlight">Item 2</li>
        <li>Item 3</li>
        <li>Item 4</li>
    </ul>
    <script>
        // Get all list items with class "highlight"
        var elements = document.querySelectorAll('ul > li.highlight');
        // Apply a background color to each element with the class
"highlight"
        elements.forEach(function(element) {
            element.style.backgroundColor = 'yellow';
        });
    </script>
</body>
</html>
```

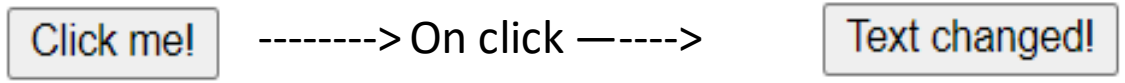


Note: Adding a specific style to each element with a given classname using a query selector

DOM Events: Example#1

- Allows JavaScript to react to HTML events like:
  - When the mouse moves over an element
  - When an HTML form is submitted
  - When an input field is changed

```
<!DOCTYPE html>
<html>
<body>
<button onclick="changeText(this)">Click me!</button>
<script>
function changeText(button) {
  button.innerHTML = "Text changed!";
}
</script>
</body>
</html>
```



DOM Events: Example#2

```
<!DOCTYPE html>
<html>
<body>
    <div onmouseover="mOver(this)"
    onmouseout="mOut(this)"
    style="background-color:blue; color: white;">
    Welcome
    </div>
<script>
    function mOver(obj) {
      obj.innerHTML = "to CS 213"
      obj.style.backgroundColor = "#27ae60";
    }
    function mOut(obj) {
      obj.innerHTML = "Welcome to "
      obj.style.backgroundColor = "#3498db";
    }
</script>
</body>
</html>
```

DOM Forms: Example#1

```
<!DOCTYPE html>
<html>
<body>
    <form id="myForm">
        <label for="name">Name:</label>
        <input type="text" id="name" name="name" required><br>
        <label for="email">Email:</label>
        <input type="email" id="email" name="email" required><br>
        <input type="submit" value="Submit">
    </form>
    <p id="output"></p>
    <script>
        const form = document.getElementById("myForm");
        const output = document.getElementById("output");
        form.addEventListener("submit", function(event) {
            event.preventDefault(); // Prevent the form from submitting normally
            // Retrieve form data
            const formData = new FormData(form);
            const name = formData.get("name");
            const email = formData.get("email");
            // Display form data in the output paragraph
            output.innerHTML = `<strong>Form Data:</strong><br>Name: ${name}<br>Email: ${email}`;
        });
    </script>
</body> </html>
```

Note: event.preventDefault() prevents the browser from doing its default action (like submitting a form or following a link) so that you can write custom code to handle the action in a way that suits your needs.

DOM Forms: Example#2

```
<script>
    function validateForm() {
        const name = document.getElementById("name").value;
        const email = document.getElementById("email").value;
        const error = document.getElementById("error");
        const emailPattern = /^[a-zA-Z0-9._-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,4}$/;
        if (name.trim() === "") {
            error.textContent = "Name is required.";
            return false; // Prevent form submission
        } else if (!email.match(emailPattern)) {
            error.textContent = "Invalid email address.";
            return false; // Prevent form submission
        } else {
            error.textContent = ""; // Clear any previous error message
            return true; // Allow form submission
        }
    }
</script>
```

Note: We use the regular expression to match if the entered mail is valid or not

thank you!

email:
k.kondepu@iitdh.ac.in