



Welcome to this session: DOM Manipulation and Event Handling

The session will start shortly...

Questions? Drop them in the chat.
We'll have dedicated moderators
answering questions.



Safeguarding & Welfare

We are committed to all our students and staff feeling safe and happy; we want to make sure there is always someone you can turn to if you are worried about anything.

If you are feeling upset or unsafe, are worried about a friend, student or family member, or you feel like something isn't right, speak to our safeguarding team:



Ian Wyles
Designated Safeguarding
Lead



Simone Botes



Nurhaan Snyman



Rafiq Manan



Ronald Munodawafa



Tevin Pitts

Scan to report a
safeguarding concern



or email the Designated
Safeguarding Lead:
Ian Wyles

safeguarding@hyperiondev.com

Skills Bootcamp Cloud Web Development

- The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly. **(Fundamental British Values: Mutual Respect and Tolerance)**
- No question is daft or silly - **ask them!**
- There are **Q&A sessions** midway and at the end of the session, should you wish to ask any follow-up questions. Moderators are going to be answering questions as the session progresses as well.
- If you have any questions outside of this lecture, or that are not answered during this lecture, please do submit these for upcoming Academic Sessions. You can submit these questions here: **Questions**

Skills Bootcamp Cloud Web Development

- For all **non-academic questions**, please submit a query:
www.hyperiondev.com/support
- **Report a safeguarding incident:** **www.hyperiondev.com/safeguardreporting**
- We would love your feedback on lectures: Feedback on Lectures
- If you are hearing impaired, please kindly use your computer's function through Google chrome to enable captions.



In Web Development; what does the DOM acronym stand for?

- A. Document Oriented Model.
- B. Domain Object Model.
- C. Document Object Model.
- D. Domain Oriented Model.





What is the purpose of a method in JavaScript?

- A. To store data in an object.
- B. To perform actions or operations on data.
- C. To return a value from a function.
- D. To define variables globally.

Learning Outcomes

- Introduce the Document Object Model (DOM) and its significance in web interactions.
- Use query selectors to manipulate HTML elements through JavaScript.
- Implement event listeners to handle user interactions like clicks, mouse movements, and keyboard events.

Lecture Overview

- Integrating HTML with JavaScript
- The Document Object Model (DOM)
- Query Selectors
- DOM manipulation
- JavaScript events and event handlers.

The Document Object Model (DOM)

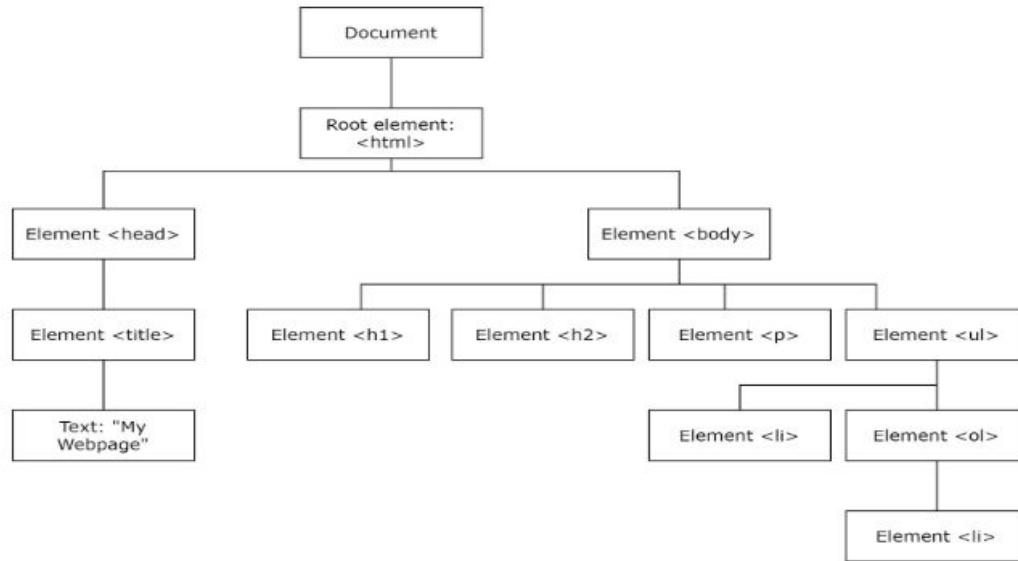
- ❖ What is the Document Object Model (DOM)?
 - **The Document Object Model (DOM)** is a programming interface for web documents.
 - It represents the **structure** of HTML documents as a hierarchical **tree** of **objects**.
 - Each **node** in the tree corresponds to a **part** of the document, such as elements, attributes, or text content.
 - The DOM **provides** a structured representation of the document, allowing **scripts** to **dynamically** access, modify, and manipulate its content and structure.

The Document Object Model Example

```
<!DOCTYPE html>
<html>
  <head>
    <title>My Webpage</title>
  </head>
  <body>
    <h1></h1>
    <h2></h2>
    <p></p>
    <ul>
      <li></li>
      <ol>
        <li></li>
      </ol>
    </ul>
  </body>
</html>
```

The Document Object Model Example

- ❖ This would be graphically represented as a tree like this:



DOM Manipulation

- ❖ What is **DOM** Manipulation?
 - DOM manipulation refers to the process of **dynamically** altering the structure, content, or style of web documents using **scripting languages** like JavaScript.
 - It allows developers to create **interactive** and **dynamic** web pages by **accessing** and **modifying** elements in the Document Object Model (DOM).

DOM Manipulation

- ❖ Why is **DOM** Manipulation significant?
 - It enables developers to respond to user actions, update content dynamically, and create engaging user experiences without page reloads.
 - It forms the foundation for modern web applications, facilitating tasks such as form validation, content updates, and animation effects.
 - It empowers developers to create responsive and interactive interfaces, enhancing usability and user engagement.

Document Object

- ❖ What is the **document** object?
 - The **document** object represents the entire HTML document as a tree structure.
 - It serves as the entry point to the web page's content, allowing **manipulation** and **interaction** with its elements.
 - The **document** object serves as the root node of the Document Object Model (DOM) tree.
 - It offers various **properties** and **methods** for interacting with the document's structure and content.
 - It serves as a fundamental component for **dynamic** web development, enabling developers to create **responsive** and **interactive** user interfaces.

DOM Traversal

- ❖ DOM **traversal** involves navigating through the DOM tree to access or manipulate elements.

```
// Retrieve the element with the ID "myDiv"
let element = document.getElementById("myDiv");

// Retrieve all elements with the CSS class "container"
let containers = document.getElementsByClassName("containers");

// Retrieve all list (li) item elements
let listItems = document.getElementsByTagName("li");

// Retrieve the first list element where the parent is an ordered list
let listOrderedParent = document.querySelector("ol > li");

// Retrieve a specific list element where the parent is an ordered list
let thirdItem = document.querySelector("ol > li:nth-child(3)");

// Retrieve the first paragraph element within a container
let paragraph = document.querySelector(".container p");

// Retrieve all paragraph elements within a container
let paragraphs = document.querySelectorAll(".container p");
```


DOM Manipulation

- ❖ Creating new elements:
 - It is possible to create entirely new elements using the createElement method in JavaScript
 - `document.createElement("p");`
- ❖ Appending to elements:
 - We can add both text and new elements to existing elements in our HTML file.
 - `document.body.appendChild(myItem);`

DOM Manipulation

❖ Modifying elements:

```
// Change inner HTML content of an element
document.getElementById("myElement").innerHTML = "<strong>New content</strong>"

// Set text content of an element
document.getElementById("myElement").textContent = "Updated text content"

// Set attribute value of an element
document.getElementById("myElement").setAttribute("class", "new-class");
```

DOM Manipulation

❖ Removing elements:

```
// Get the element to remove
let elementToRemove = document.getElementById("toRemove");

// Remove the element from the DOM
elementToRemove.remove();
```

Events

- ❖ JavaScript is often used to handle events that occur on a website.
- ❖ An **event** is an **action** that occurs that your program responds to.
- ❖ DOM **events** are either things that a user does, such as clicking a button or entering text into a text box, or actions caused by the browser, such as when a web page has been completely loaded.
- ❖ **Event handling** is the process of **capturing**, **processing**, and **responding** to events.

Inline Event Handlers

- ❖ The most common events you'll deal with when getting started are:

Event	Description
onchange	Triggered when the value of an HTML element (like an input box or dropdown menu) changes.
onclick	Happens when you click on an HTML element, such as a button or link.
onmouseover	Occurs when you move the mouse pointer over an HTML element.
onmouseout	Happens when you move the mouse pointer away from an HTML element.
onkeydown	Triggered when you press a key on the keyboard.
onload	Occurs when the entire HTML page and all its resources (like images) have finished loading.

Inline Event Handlers

- ❖ HTML attribute equivalents for event handlers exist, but using them is discouraged due to bad practice.
- ❖ Inline event handler attributes may seem convenient for quick tasks but lead to inefficiency and manageability issues.
- ❖ Mixing HTML and JavaScript makes code harder to read; separating them promotes better organization and reuse across multiple documents.
 - In files with multiple elements (e.g., 100 buttons), using inline handlers results in excessive attributes, complicating maintenance.
- ❖ Many server configurations restrict inline JavaScript for security reasons.
- ❖ Overall, HTML event handler attributes are outdated and should be avoided.

Handling UI Events

- ❖ One common approach to handle UI events is by using the **addEventListener()** method. This method allows developers to attach event listeners to DOM elements and specify callback functions to be executed when the events occur.
- ❖ The method has the following syntax:
 - `element.addEventListener(event, handler, options);`
 - **event**: A string representing the type of event to listen for (e.g., "click", "keydown").
 - **handler**: A function that is called when the event occurs.
 - **options** (optional): An object that can specify properties like *capture*, *once*, and *passive*.

The event object

- ❖ In JavaScript, when an event occurs, an **event** object is automatically created by the browser.
- ❖ The event object contains information about the event that occurred, such as its type, target element, and additional event-specific data.
- ❖ In codebases, you may see the event object argument named as one of the following:
 - event
 - evt
 - e (most commonly found)

```
document.addEventListener("click", function (event) {  
  console.log("Event Type:", event.type);  
  console.log("Target Element:", event.target);  
  console.log("Mouse Coordinates:", event.clientX, event.clientY);  
});
```



Why is the DOM important in JavaScript?

- A. It defines the structure of HTML documents.
- B. It provides a way to store data in the browser.
- C. It helps in styling web pages with CSS.
- D. It allows JavaScript to manipulate and interact with web page elements.



What is the purpose of the `addEventListener()` method in JavaScript?

- A. To remove an event handler from an element.
- B. To attach an event handler to a specified element.
- C. To create a new HTML element.
- D. To change the style of an element.



Let's take a
break



Questions and Answers



Thank you for attending



CoGrammar



Department
for Education