CoGrammar

Welcome to this session:

Task Walkthrough - 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

CoGrammar HyperionDev

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.

Learning Outcomes

- ❖ **Create and manipulate HTML elements using JavaScript**, updating the DOM in response to user input.
- ❖ **Define JavaScript functions** to add, modify, and delete items within an interactive list.
- ❖ **Use event listeners to make a webpage interactive**, responding to user actions such as clicks and keyboard input.
- ❖ **Apply CSS dynamically** to indicate changes in task status visually.

What is the purpose of document.getElementById() in JavaScript?

A. To create a new HTML element.
B. To retrieve an HTML element by its ID.
C. To add an item to an array.
D. To delete an element from the DOM.

Which of the following event listeners can detect when a button is clicked?

A. mouseenter
B. keyup
C. click
D. scroll

CoGrammar

Lecture Overview

- → Presentation of the Task
- → Introduction to DOM
- → Introduction to Event Handling
- → Task Walkthrough

CoGrammar

DOM Task

Imagine you have a busy week ahead, with several tasks to keep track of! In this task, you'll create a Task Manager that helps you manage and organize your daily to-do list using JavaScript and DOM manipulation.

You'll build functions to display the list items on the webpage, mark tasks as completed, and remove tasks when finished. This exercise will give you hands-on experience with creating interactive elements on a webpage—perfect for your future coding projects!

- Initialize an Array for Your Tasks
- Define the displayTasks Function
- Mark Tasks as Completed

Event Handling Task

Now that your basic to-do list is working, let’s make it fully interactive! In this task, you’ll add the ability to add new tasks, delete tasks, and use keyboard shortcuts to manage the list even faster. You’ll set up functions that allow users to add items by pressing Enter, delete tasks by clicking a delete button, and mark items as completed with just one click.

With these features, you’re building a tool that’s not only practical but also enhances your coding skills and understanding of how dynamic content works. 🎯

DOM...DOM...DOMMMMM

- 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.

CoGrammar

DOM...DOM...DOMMMMM

- 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).

CoGrammar

DOM...DOM...DOMMMMM

- Significance:
  - Enables developers to respond to user actions, update content dynamically, and create engaging user experiences without page reloads.
  - Forms the foundation for modern web applications, facilitating tasks such as form validation, content updates, and animation effects.
  - Empowers developers to create responsive and interactive interfaces, enhancing usability and user engagement.

CoGrammar

Don’t You...Forget About Me

- 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.

CoGrammar

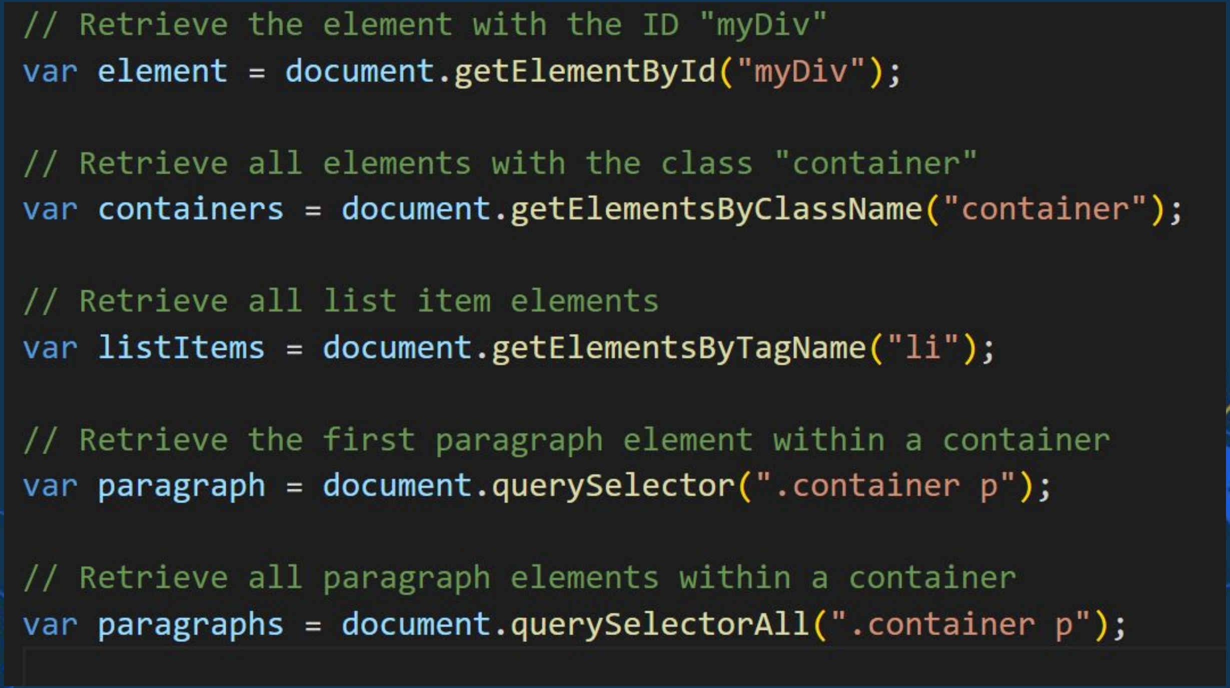
Don’t You...Forget About Me

- ❖ Offers various **properties** and **methods** for interacting with the document's structure and content.
- ❖ Serves as a fundamental component for **dynamic** web development, enabling developers to create **responsive** and **interactive** user interfaces.

CoGrammar

Moving through the madness

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

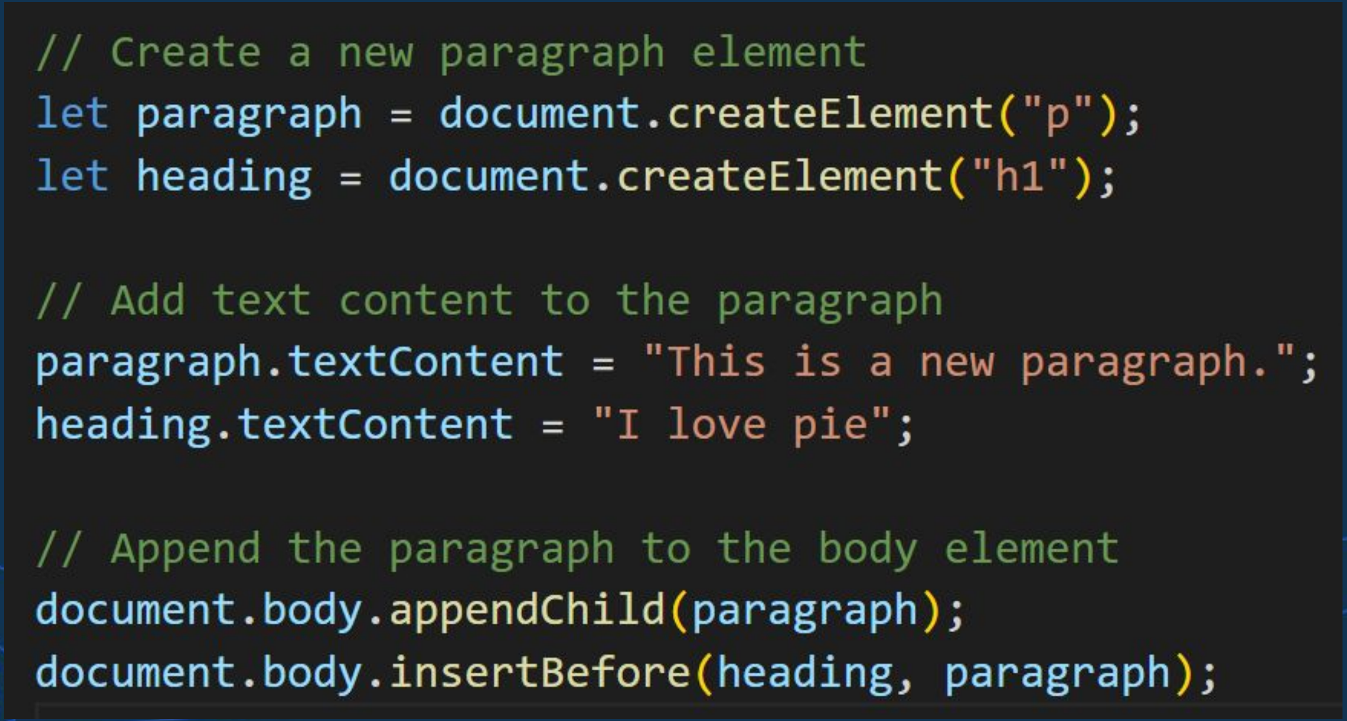


```
// Retrieve the element with the ID "myDiv"
var element = document.getElementById("myDiv");

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

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

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

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

CoGrammar

It’s Morphin’ Time

❖ Adding elements:

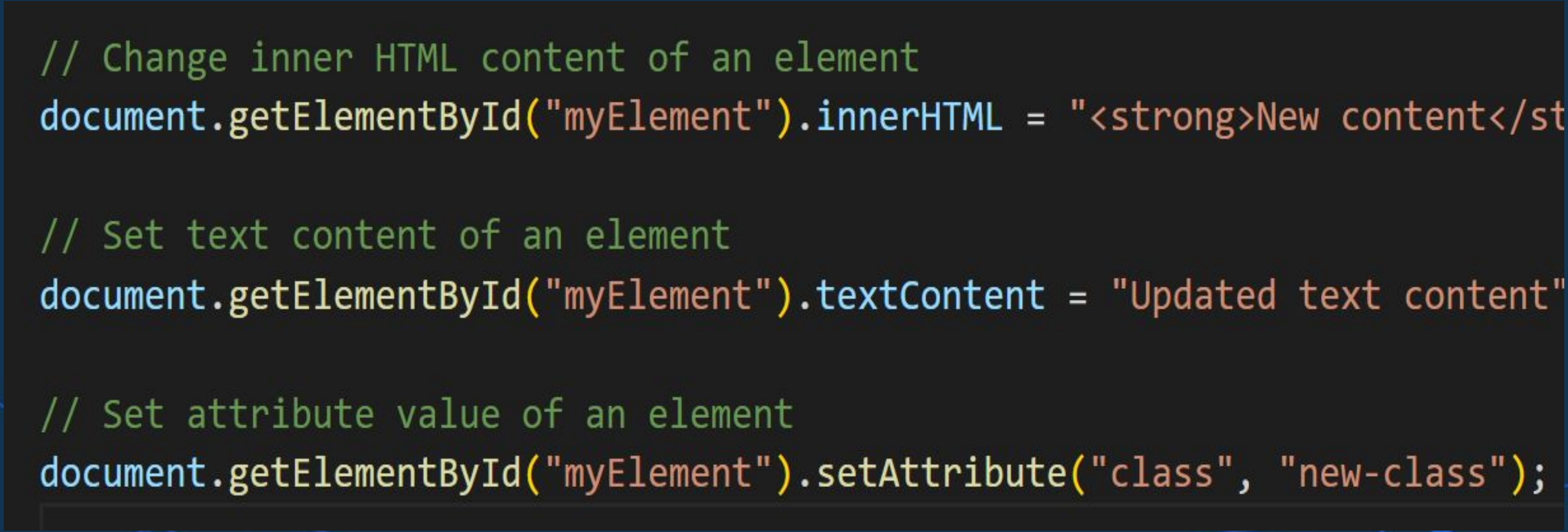


```
// Create a new paragraph element
let paragraph = document.createElement("p");
let heading = document.createElement("h1");

// Add text content to the paragraph
paragraph.textContent = "This is a new paragraph.";
heading.textContent = "I love pie";

// Append the paragraph to the body element
document.body.appendChild(paragraph);
document.body.insertBefore(heading, paragraph);
```

CoGrammar

It’s Morphin’ Time

❖ Modifying Elements:

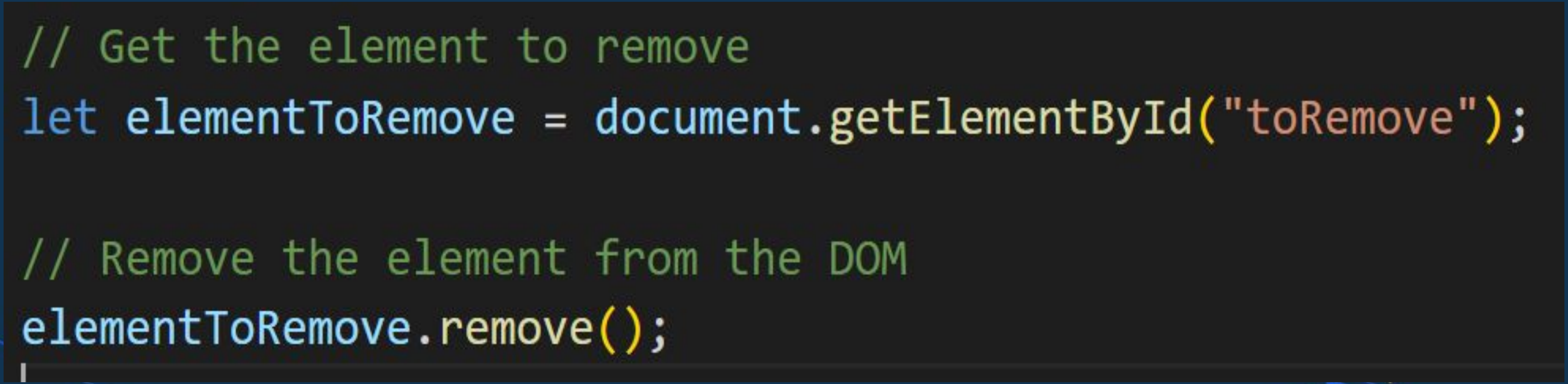


```
// Change inner HTML content of an element
document.getElementById("myElement").innerHTML = "<strong>New content</st

// 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");
```

CoGrammar

It’s Morphin’ Time

❖ Removing Elements:



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

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

CoGrammar

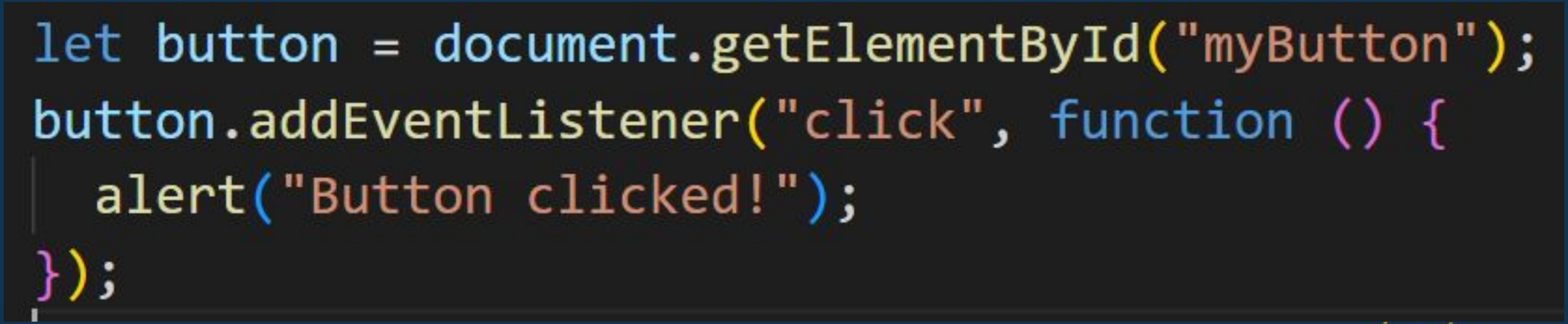
Event Handling

- ❖ Events are actions or occurrences that happen in the system you are programming.
- ❖ Event handling is the process of capturing, processing, and responding to events.
- ❖ Without event handling, applications would not be able to respond to user input or system events promptly.

CoGrammar

Event Handling in Action

❖ Event handling in JavaScript involves capturing and responding to various events that occur within a web page or application.

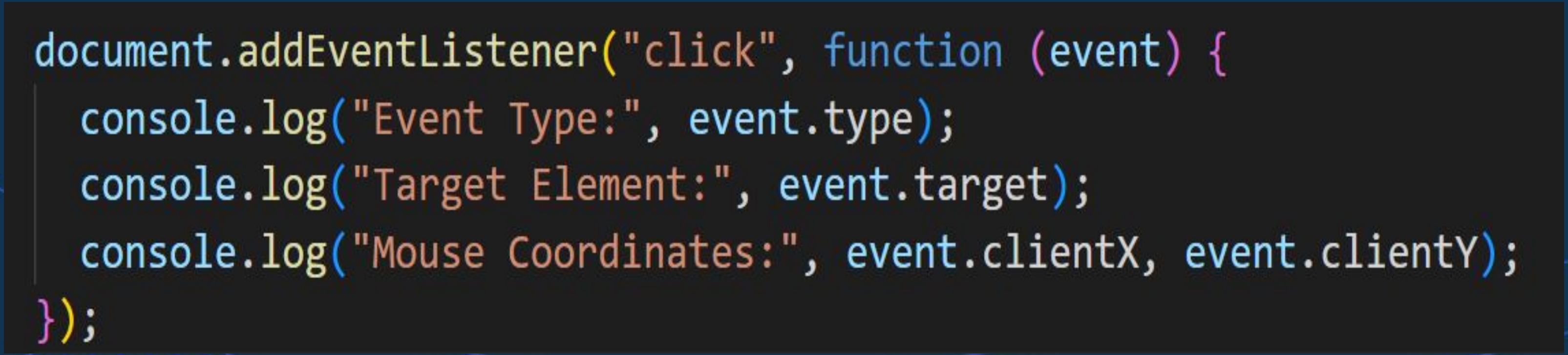


```
let button = document.getElementById("myButton");
button.addEventListener("click", function () {
  alert("Button clicked!");
});
```

CoGrammar

The event Object

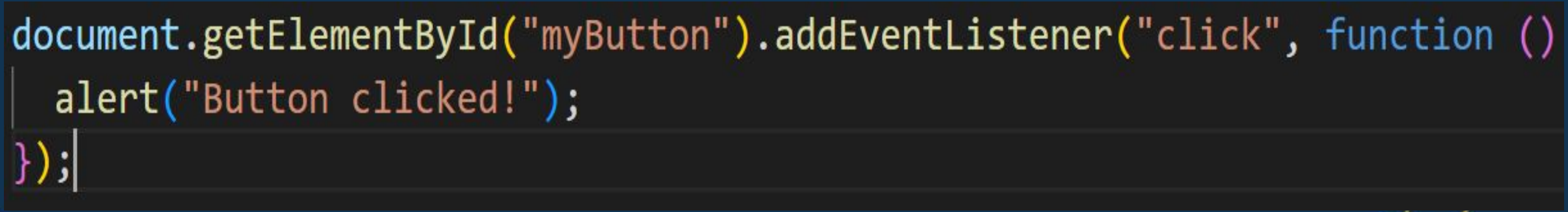
- In JavaScript, when an event occurs, an event object is automatically created by the browser.
- The event object contains information about the event that occurred, such as its type, target element, and additional event-specific data.



```
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);
});
```

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.

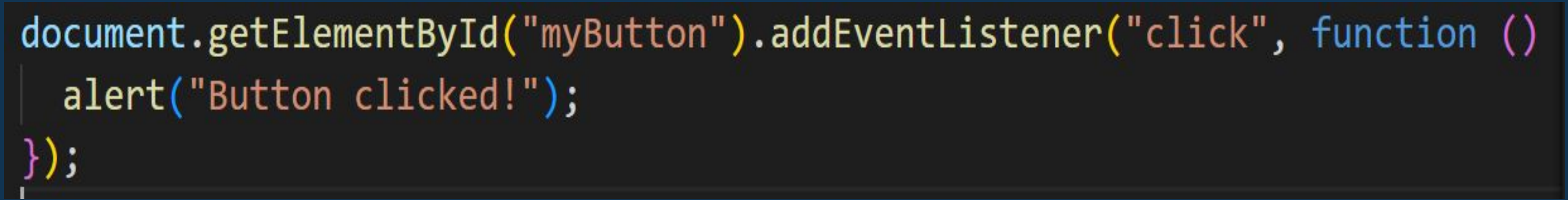


```
document.getElementById("myButton").addEventListener("click", function ()
  alert("Button clicked!");
});
```

CoGrammar

Handling UI Events

❖ Click Event: Initiates an action when a user clicks on a button, link, or any interactive element.

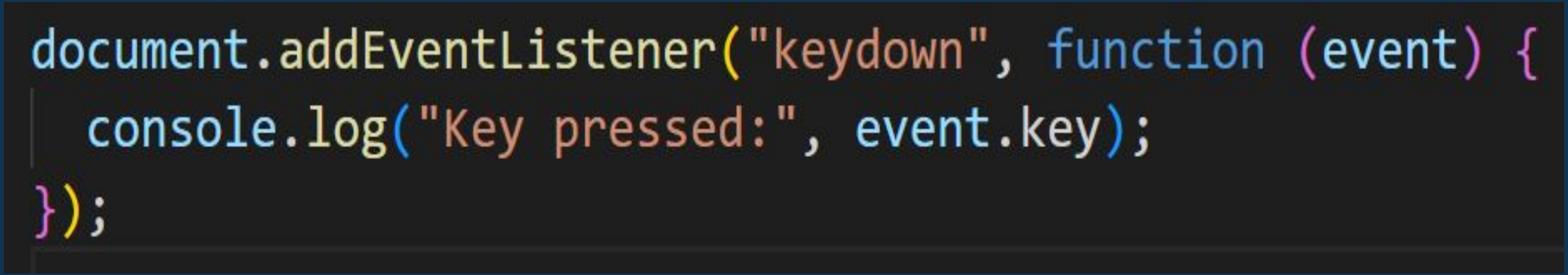


```
document.getElementById("myButton").addEventListener("click", function ()
  alert("Button clicked!");
});
```

CoGrammar

Handling UI Events

❖ Keydown Event: Captures keystrokes, allowing for keyboard-driven interactions within the application.

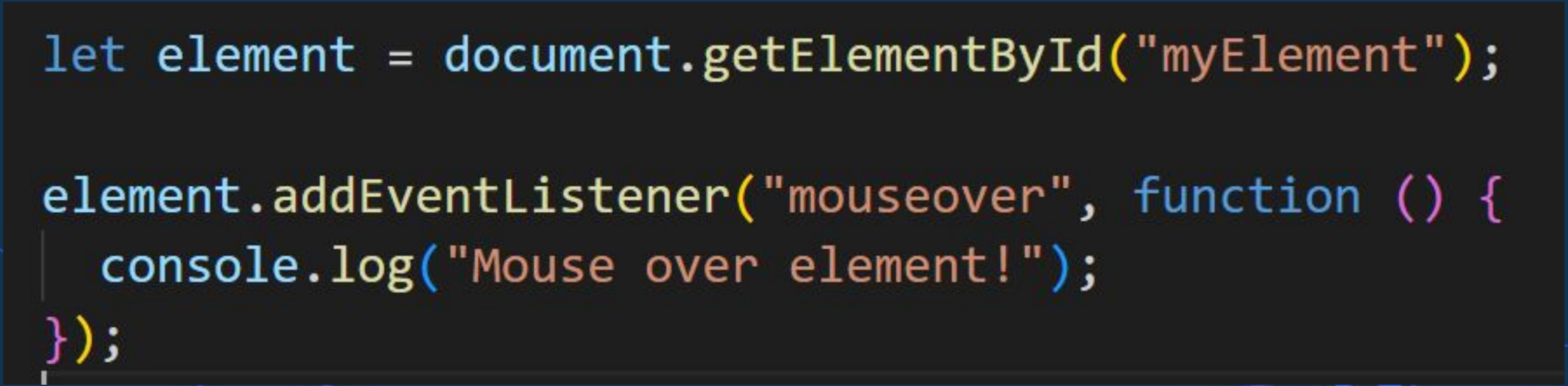


```
document.addEventListener("keydown", function (event) {
  console.log("Key pressed:", event.key);
});
```

CoGrammar

Handling UI Events

❖ Mouseover Event: Provides feedback when the mouse cursor enters a specific area or element.

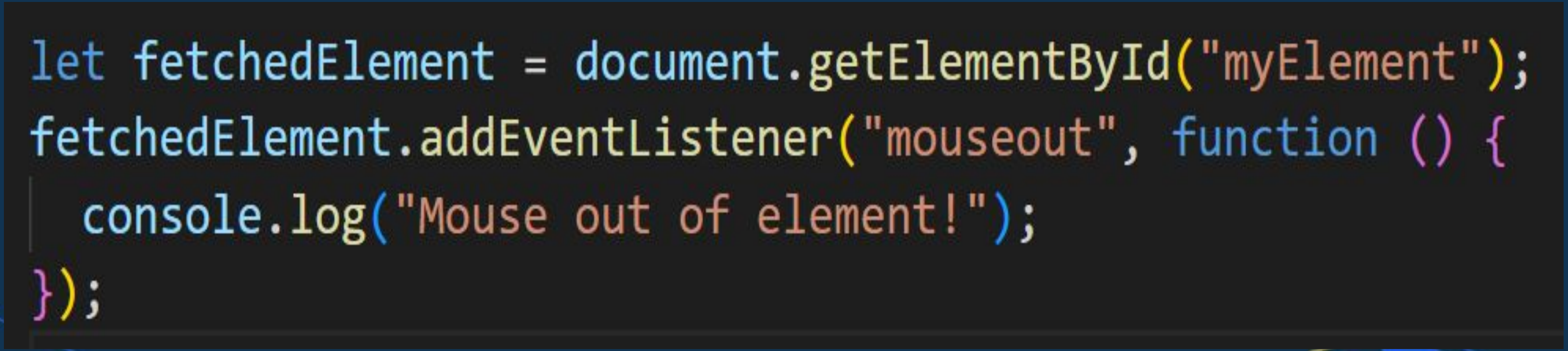


```
let element = document.getElementById("myElement");

element.addEventListener("mouseover", function () {
  console.log("Mouse over element!");
});
```

CoGrammar

Handling UI Events

❖ Mouseout Event: Triggers actions when the mouse cursor leaves a designated area or element.



```
let fetchedElement = document.getElementById("myElement");
fetchedElement.addEventListener("mouseout", function () {
  console.log("Mouse out of element!");
});
```

CoGrammar

DOM Task

Imagine you have a busy week ahead, with several tasks to keep track of! In this task, you'll create a Task Manager that helps you manage and organize your daily to-do list using JavaScript and DOM manipulation.

You'll build functions to display the list items on the webpage, mark tasks as completed, and remove tasks when finished. This exercise will give you hands-on experience with creating interactive elements on a webpage—perfect for your future coding projects!

- ❖ Initialize an Array for Your Tasks
- ❖ Define the displayTasks Function
- ❖ Mark Tasks as Completed

Event Handling Task

Now that your basic to-do list is working, let’s make it fully interactive! In this task, you’ll add the ability to add new tasks, delete tasks, and use keyboard shortcuts to manage the list even faster. You’ll set up functions that allow users to add items by pressing Enter, delete tasks by clicking a delete button, and mark items as completed with just one click.

With these features, you’re building a tool that’s not only practical but also enhances your coding skills and understanding of how dynamic content works. 🎯

What does appendChild() do in JavaScript?

A. Creates a new array element.
B. Adds a child element to the end of a parent element.
C. Deletes a parent element.
D. Changes the text of an element.

CoGrammar

What’s the purpose of using classList.toggle() in JavaScript?

A. To permanently add a new class to an element.
B. To remove an element from the DOM.
C. To add or remove a class based on its presence.
D. To create a new list in the DOM.

CoGrammar

CoGrammar

Q & A SECTION

Please use this time to ask any questions relating to the topic, should you have any.

Thank you for attending

CoGrammar

SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education