ual: creative computing institute

Peckham DAZ

Accessible Web Development

Session 3: DOM Manipulation with JavaScript

Date

Session Structure

- Lecture (1 hour)
- Break (15 mins)
- Labs Exercises (2 hours 30 mins)
- Debrief (15 mins)

ual: creative computing institute

DOM Manipulation

Using JavaScript to manipulate HTML & CSS

Date

In this session we'll use JavaScript to add functionality and dynamic styling to websites. We'll do this through **DOM manipulation**.

What is the DOM?

D – Document

- Represents the **entire webpage** loaded in the browser.
- The starting point where JavaScript interacts with the page.

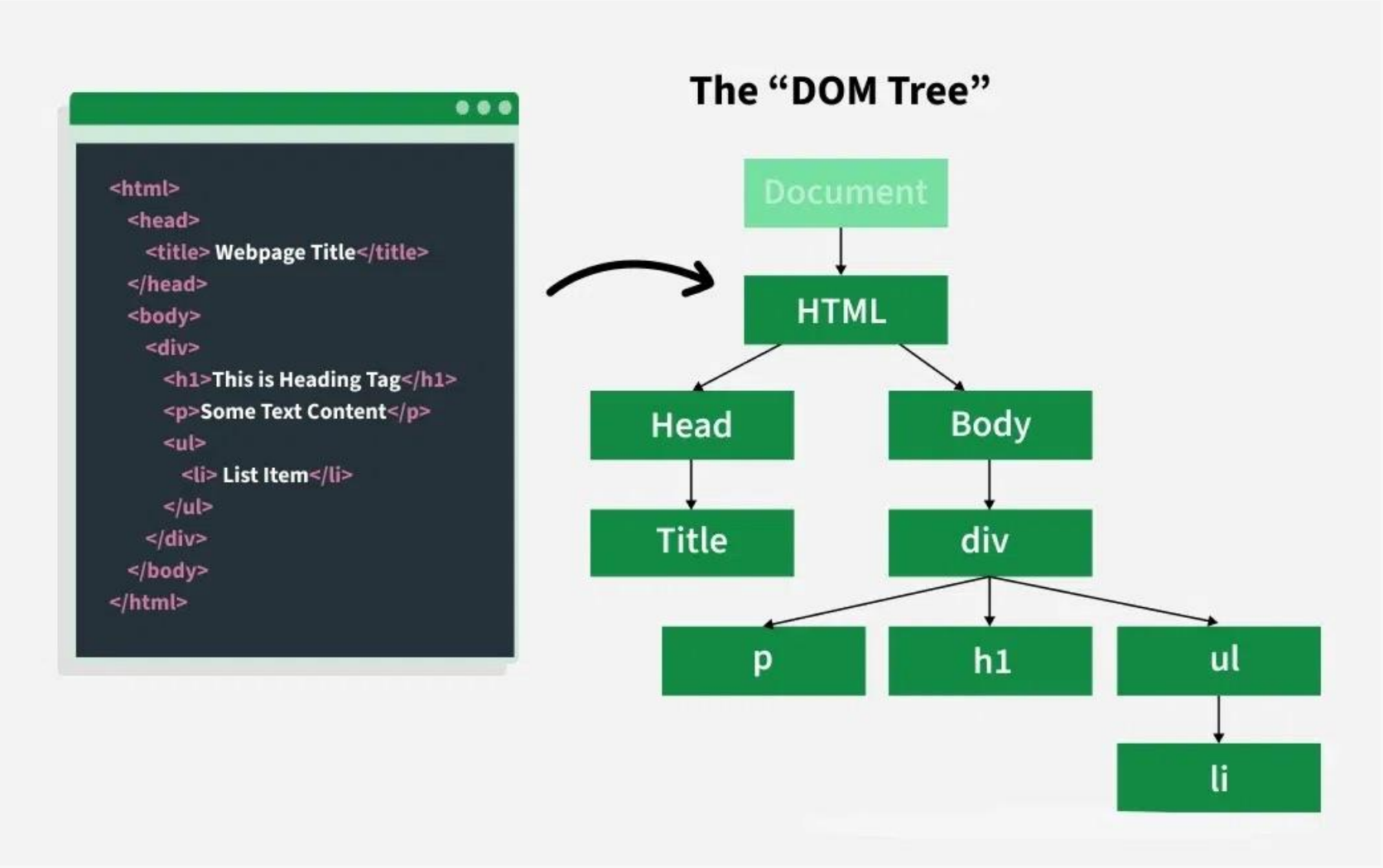
O – Object

- The webpage is structured as a **tree of objects**.
- Each HTML element is an **object** that can be modified.

M – Model

- The webpage is represented as a **hierarchical structure**.
- JavaScript can **traverse, modify, and update** this structure dynamically.

You can think of the DOM as a family tree with parents, children and sibling elements.



How to Access DOM Elements

To manipulate the DOM we need to access its HTML elements. We do this using JavaScript and the `document` object.

We can 'get' HTML elements by:

- Their ID attribute name
- Their Class attribute name
- Their element tag name
- Through CSS (query) selectors

```
// Accessing an element by its ID
const headerElement = document.getElementById('header');

// Accessing elements by class name
const paragraphs = document.getElementsByClassName('paragraph');

// Accessing elements by tag name
const images = document.getElementsByTagName('img');
```

Modify Element Content

Once we have access to an element, we can modify its content.

This is a simple but powerful way to update text.

```
// Accessing an element by its ID
const headerElement = document.getElementById('header');

// Modifying the content of an element
headerElement.innerHTML = 'New Header Text';
```

Events and Event Handling

- Events are actions or occurrences that happen in the browser, such as a user clicking a button or resizing the window.
- JavaScript allows us to handle these events and execute code in response.
- Event handling is a crucial aspect of creating interactive web pages.

How to Add Event Listeners

To respond to events, we can use event listeners. These are functions that "listen" for a specific event on a particular element. Let's consider a button click:

- Here when the button with the ID `myButton` is clicked, an alert saying `Button Clicked!` will pop up as an alert.
- Event listeners provide a way to execute custom code based on user interactions.

```
// Accessing a button element
const myButton = document.getElementById('myButton');

// Adding a click event listener
myButton.addEventListener('click', function() {
    alert('Button Clicked!');
});
```

Changing Styles Dynamically

We can use the `style` property of an element to change its appearance. Let's take an example of changing the colour of a paragraph when a button is clicked.

Here, when the button with the ID `colorButton` is clicked, the text colour of the paragraph with the ID `myParagraph` is changed to blue.

```
// Accessing a paragraph element
const myParagraph = document.getElementById('myParagraph');

// Accessing a button element
const colorButton = document.getElementById('colorButton');

// Adding a click event listener to the button
colorButton.addEventListener('click', function() {
    // Changing the color style of the paragraph
    myParagraph.style.color = 'blue';
});
```

Creating New Elements

The `createElement` method is used to create a new HTML element. Let's create a new paragraph element and append (add) it to the <body> of the document.

Here, we create a new <p> (paragraph) element, set its text content, and then append it to the body of the document.

```
// Creating a new paragraph element
const newParagraph = document.createElement('p');

// Setting the text content of the new paragraph
newParagraph.textContent = 'This is a new paragraph.';

// Appending the new paragraph to the body of the document
document.body.appendChild(newParagraph);
```

Modifying Element Attributes

We can also modify the attributes of existing elements. Let's consider changing the source of an image dynamically.

Here, we access an image element with the ID `myImage` and change its `src` attribute to `new-image.jpg`, dynamically updating the displayed image.

```
// Accessing an image element
const myImage = document.getElementById('myImage');

// Changing the source attribute of the image
myImage.src = 'new-image.jpg';
```

Toggle Element Visability

You can toggle the visibility of an element by using the `display` style property. Here we create a button that toggles the visibility of a paragraph.

Note how we toggle the paragraph CSS display style from `'none'` to `'block'` when the button is clicked.

We do this with the ternary operator, which is an `if else` statement on a single line.

```
// Accessing a button element
const toggleButton = document.getElementById('toggleButton');

// Accessing a paragraph element
const toggleParagraph = document.getElementById('toggleParagraph');

// Adding a click event listener
toggleButton.addEventListener('click', function() {
    // Toggling the visibility of the paragraph
    toggleParagraph.style.display = toggleParagraph.style.display === 'none' ? 'block' : 'none';
});
```

Let's look at some interactive
code examples

Accessibility and the DOM

As was mentioned last session, JavaScript allows us to enhance accessibility, by for example:

- Dynamically adjust font size based on user preferences.
- Automatically move focus to interactive elements (e.g., modals, notifications).
- Detect and apply high contrast mode for better readability.
- Add missing alt attributes to images dynamically.

But...

… the Document Object Model (DOM) is the interface that allows JavaScript to manipulate webpages and make them accessible.

DOM Resources

Documentation & Guides

- MDN Web Docs - Introduction to the DOM
- W3Schools - JavaScript DOM Tutorial

Interactive Learning

- FreeCodeCamp - JavaScript and the DOM
- The Odin Project - DOM Manipulation

Lab Exercises

Can be found on the Peckham DAZ github

ual: creative computing institute

Great work! ☺

@ual_cci

@ual_cci

arts.ac.uk/cci

Date