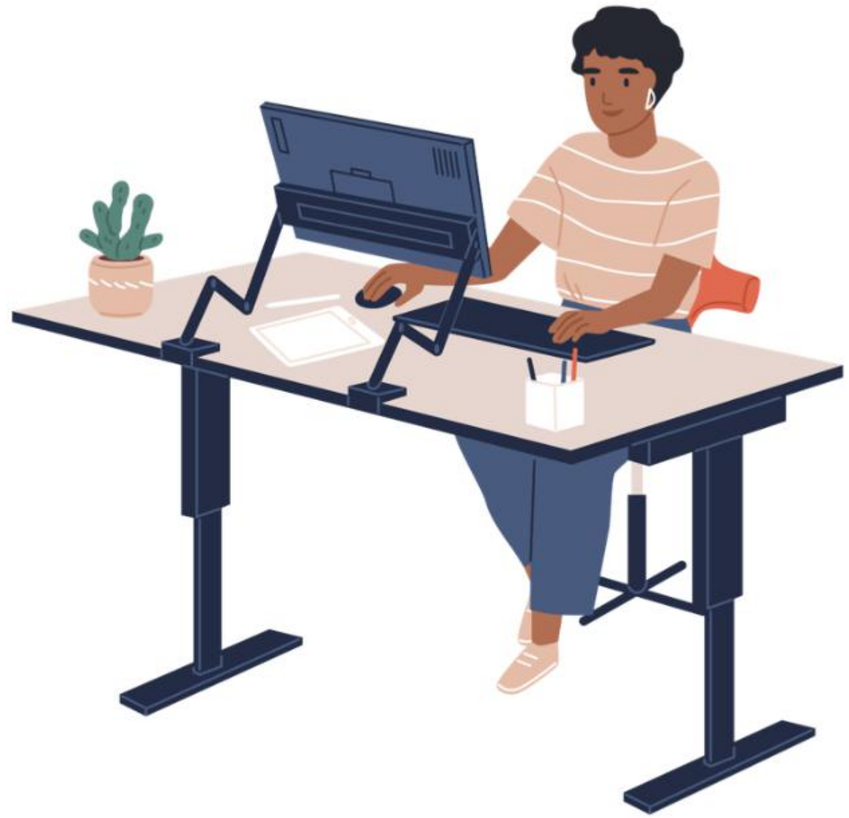
Slide Note

Learning Consolidation

Develop Interactive Web Pages Using DOM and DOM Events



Menu

00:01:00:00:26

01/ 26

Slide Note

In this sprint, you learned to:

- Explore Document Object Model (DOM) elements
- Manipulate DOM to add dynamic effects
- Work with DOM object properties to manipulate element styles
- Handle events by an action when a user interacts with the program
- Perform form validations using HTML5 built-in functionality

Slide Note

Event is an action that occurs as a result of the user or another source, such as a mouse click.

What Is DOM (Document Object Model)?

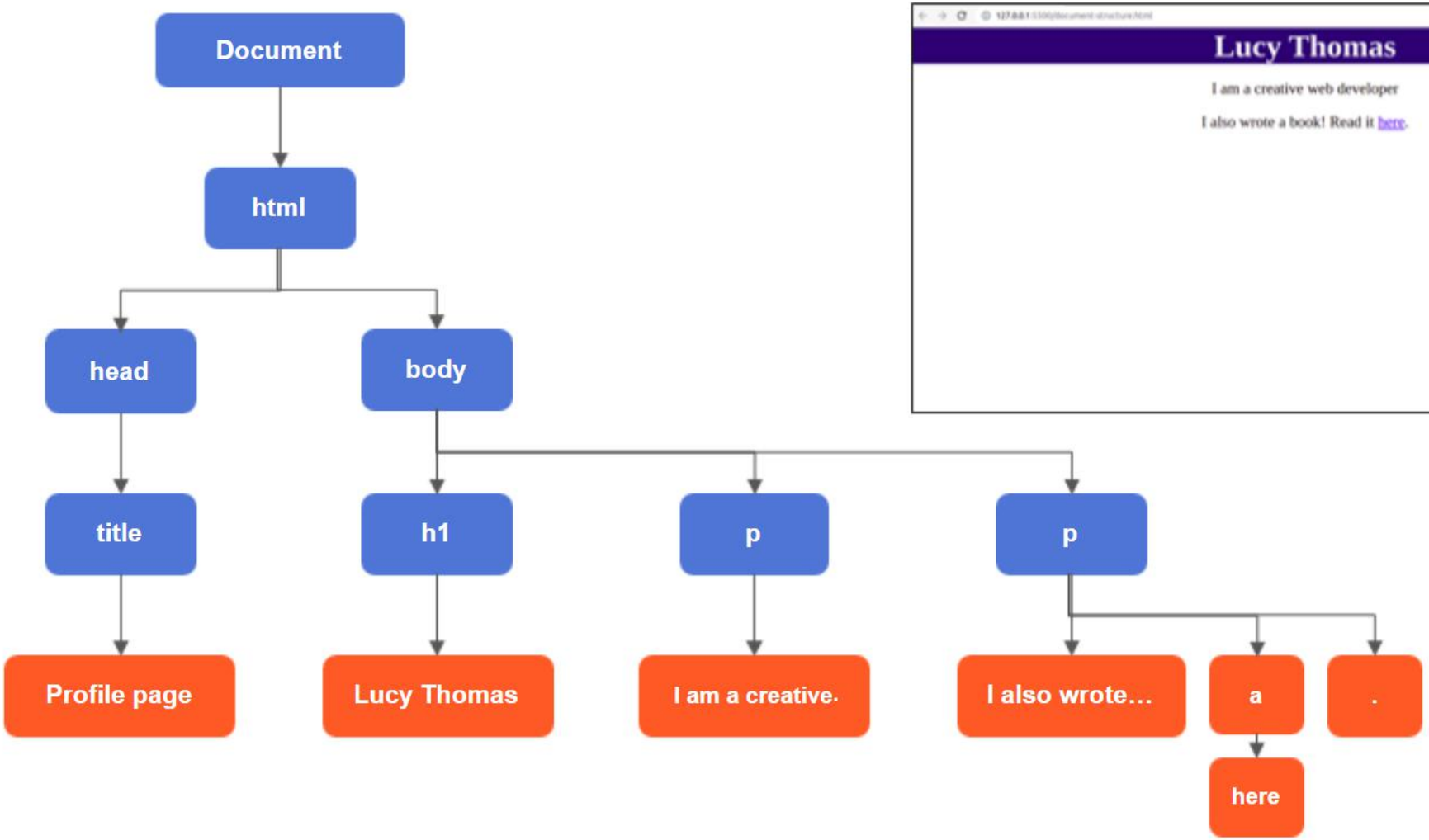
- DOM is a standard for modelling web documents as objects that manipulates them using a standard programming interface.
- DOM defines:
  - The HTML elements as objects.
  - The properties and methods to access all HTML elements.
  - The events of all HTML elements.
- DOM tells you how to get, change, add or delete HTML elements dynamically which, makes the web page interactive.

Note: An event is an action that occurs because of the user or another source, such as a mouse click.

Slide Note

Document Structure

DOM Tree



Web Page Preview

Slide Note

Relation between HTML page and how browser reads, how it containerize and how it displays. How the JS code is running?

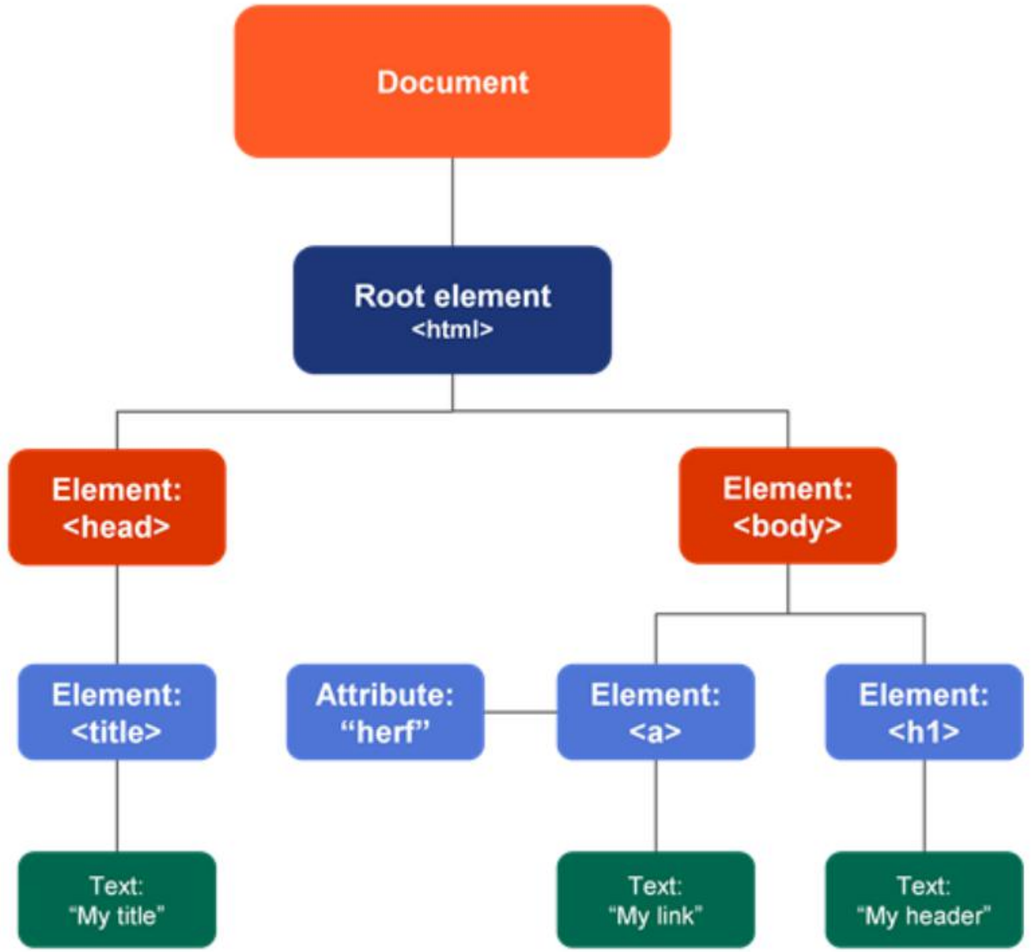
JavaScript HTML Document Object

- When an HTML document is loaded into a web browser, it becomes a **document object** that serves as an entry point into the web page's content, which is the DOM tree.
- The HTML DOM document object is the owner of all other objects in your web page.
- Any element in the HTML page can be accessed with the document object.
- Some examples where a document object is used to access and manipulate HTML are given below:
  - Finding HTML elements
  - Changing HTML elements
  - Adding and deleting elements
  - Adding or modifying the style properties of existing elements

Slide Note

Traversing the DOM Tree

- The DOM tree consists of a tree of objects called nodes.
- The entire document is a document node.
- There are three main type of nodes:
  - Element Node: Any HTML element in the DOM.
  - Text Node: Any lone text outside of a document.
  - Comment Node: An HTML comment.
- You can traverse the node tree using the node relationships with the HTML DOM.



Slide Note

• When a W3C event listener's event occurs and it calls its associated function, it also passes a single argument to the function which is a reference to the event object.

• The event object contains a number of properties that describe the event that occurred.

• The table lists few of the most commonly used event object properties.

• *Note: Execute the following code in the subsequent demostration.*

• *If we want to determine the type of event that occurred, such as a click, we would have to write the following code.*

```
function myEvent(e)
{
var evtType = e.type alert(evtType) // displays the event type
}
```

Finding DOM Elements

Method	Description
getElementsByTagName( )	Returns all elements matching the specified tag name
getElementsByClassName( )	Returns all elements matching the specified class name
getElementById( )	Returns the first matched element, ignoring the remaining
querySelector( )	Returns the first element that matches the specified CSS selector
querySelectorAll( )	Returns all the elements that match the specified CSS selector

Slide Note

Manipulate DOM Elements

- Change HTML content dynamically:
  - The content of an HTML element can be modified using various properties.
  - `innerHTML, innerText, textContents`
- Add a new HTML element to the DOM:
  - To add a new element to the HTML DOM, the element node should be created before appending it to the existing element.
    - `createElement(), createTextNode(), appendChild(), replaceChild()`
- Delete or replace an existing element from the DOM:
  - `remove(), removeChild(), replaceChild(), replaceWith()`

Slide Note

Creating, Removing and Updating DOM

Method	Description
document.createElement( )	Creates a new element
document.createTextNode( )	Helps in creating a text node
node.appendChild(node1)	Appends a new element at the end of the existing node
node.removeChild(node1)	Helps to remove a child node using a reference to the parent node
childNode.remove( )	Helps to remove a node based only on a reference to itself

Slide Note

Basic Differences Between DOM Properties

- innerHTML returns all text, including html tags, that is contained by an element.
- innerHTML can lead to script injections which can be malicious code.
- append() accepts node objects and DOMStrings.
- append() allows you to add multiple items.
- createElement() provides better performance than innerHTML

- innerText returns all text contained by an element and all its child elements.
- innerText is safe when the input to innerText is provided by the user.
- appendChild() accepts only node objects.
- appendChild() allows only a single item.
- innerHTML is less efficient as it causes bowsers to parse and recreate all DOM nodes inside that element.

Slide Note

Get Attributes of HTML Elements

- In HTML, tags may have attributes that contain additional information about HTML elements as name/value pairs.
- When the browser parses the HTML to create DOM objects for tags, it recognizes the standard attributes and creates DOM properties from them.
- The `Element.attributes` property returns a collection of attributes in an element as a list of built-in `Attr` objects.
- As shown in the code, attribute names and values can be obtained using the `for...of` loop.

```
<html>
  <body>
    <img id="mg1" src="bird.jpg"
alt="bird"/>
    <script>
     for(let attr of mg1.attributes){
        console.log(`${attr.name} =
${attr.value}`);
      }
    </script>
  </body>
</html>
```

```
id = mg1
src = bird.jpg
alt = bird
```

Slide Note

Modify Attributes of HTML Elements

In JavaScript, there are four methods, that can be used to work with element attributes:

Method	Description	Example
hasAttribute()	Returns a true or false Boolean	element.hasAttribute('src');
getAttribute()	Returns the value of a specified attribute or null	element.getAttribute('src');
setAttribute()	Adds or updates value of a specified attribute	element.setAttribute('src', 'img.jpg');
removeAttribute()	Removes an attribute from an element	element.removeAttribute('src');

```
// Assign image element
const img = document.querySelector('img');

img.hasAttribute('src'); //returns true
img.getAttribute('src');  //returns "img.png"
img.removeAttribute('src');//remove the src attribute and
value
```

Slide Note

JavaScript Events

- Events are actions or occurrences that happen in the system you are programming, which the system tells you about so your code can react to them.
- In the case of the Web, events are fired inside the browser window and tend to be attached to a specific item that resides in it.
- Here are examples of different types of events:
  - A web page finishes loading.
  - The user resizes or closes the browser window.
  - The user selects a certain element or hovers the cursor over a certain element.
  - The user chooses a key on the keyboard.
  - A form is submitted.
  - An error occurs.
- Programmers can create *event handler* code that will run when an event fires, allowing web pages to respond appropriately to change.

Slide Note

• In the case of the Web, events are fired inside the browser window, and tend to be attached to a specific item that resides in it.

• This might be a single element, set of elements, the HTML document loaded in the current tab, or the entire browser window. There are many different types of events that can occur.

Handling Events

- Event handling refers to an action being performed when an event occurs.
- An event handler has to be registered with the event to listen to the event.
- Registering events can be done using:
  - onEvent property
  - addEventListener( ) function
- When an event fires, the registered event handler is called with the event object.
- The event object contains a number of properties that describe the event that occurred.
- Events can be unregistered when an element should stop listening to the events.
- removeEventListener( ) helps to unregister an event.

Slide Note

• For example, here is a link that cannot be followed when clicked.

Event.preventDefault()

- Most events have a default action associated with them.
- For most of the events, the JavaScript event handlers are called before the default behavior takes place.
- If the handler does not want this normal behavior to happen, since it is already taken care of, it can call the preventDefault method.

```
<a href="https://developer.mozilla.org/" >MDN</a>
<script>
  let link = document.querySelector("a");
  link.addEventListener("click", event => {
    console.log("Nope.");
    event.preventDefault();
  });
</script>
```

Slide Note

1. The details where the server is not required can be validated at client side such as:
   1. Checking for mandatory inputs.
   2. Checking for lengths of input data.
   3. Checking for password strength.
   4. Checking for patterns as in case of emails, zip code, phone #, credit card #, social security #, etc.
2. The server's involvement would be required when the check is made against the existing data.
   1. Checking for availability of a user ID.
   2. Authenticating user with the help of credentials provided.
   3. Authorizing user to check for the access privileges.
   4. Checking for amount in user's account while accepting payment request.

Client-Side vs. Server-Side Validation

Client-Side Validation

- Immediate validation
- Not all validations need server support
- E.g., Required field validation, Length validation, Pattern validation, Compare Password validation.

Server-Side Validation

- Makes network calls
- Increases server workloads and latency
- E.g., Credit card validity check, User validity check (these checks need to verify authenticity of these data).

Slide Note

HTML5 Built-in Form Validation

- Built-in form validation can be performed using validation **attributes** on form elements.
- Some of the common attributes used with form elements are shown below.

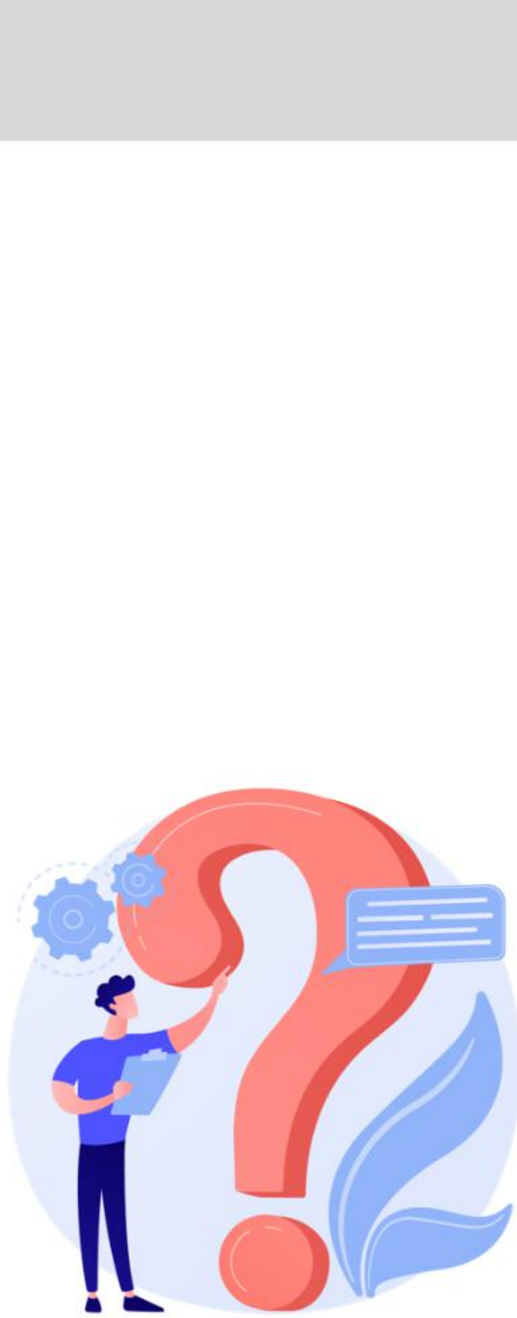
Attributes	Description
max	Specifies the maximum value of an input element
min	Specifies the minimum value of an input element
pattern	Specifies the value pattern of an input element
required	Specifies that the input field requires an element
disabled	Specifies that the input element should be disabled
type	Specifies the type of an input element
maxlength	Specifies the maximum number of characters allowed in the input field
multiple	Specifies that the user is allowed to enter more than one value in an input field

Slide Note

Self-Check

What is the differentiate between:

`<input type=“submit”>` and `<button type=“submit”><button>`



Slide Note

Self-Check: Solution

What is the difference between:

`<input type=“submit”>` and `<button type=“submit”><button>`

<input type=“submit”>

- Creates a button type element with caption Submit
- When clicked, submits form data
- Cannot add any HTLM content for the caption

<button type=“submit”></button>

- Creates a button type element without any caption
- When clicked, submits form data
- Can add any HTML content for the caption

<button type=“submit>Save</button>

Slide Note

Self-Check

What are the different ways an invalid zip code can be prevented from getting stored?

Slide Note

Self-Check: Solution

What are the different ways an invalid zip code can be prevented from getting stored?

- **Check if the zip code is left blank.**
- **If not left blank, check if it is of a valid pattern.**
- **On a device, if the location is turned on, the user's zip code can automatically be fetched, then the user does not have to input it.**

Slide Note

Self-Check

When should validation be done for the following checks (client-side or server-side)?

- Checking whether username exists.
- Checking if zip code is of valid format.
- Checking strength of password.
- Checking for valid onward and return journey dates.
- Checking whether credit is not blocked.

Slide Note

Self-Check: Solution

When should validation be done for the following checks (client-side or server-side)?

- Checking whether username exists – **server-side validation.**
- Checking if zip code is of valid format – **client-side validation.**
- Checking strength of password – **client-side validation.**
- Checking for valid onward and return journey dates – **client-side validation.**
- Checking whether credit card # is not blocked – **server-side validation.**

Slide Note

Self-Check

Is the comparison of Password and Confirm Password fields a client-side or server-side validation?

Slide Note

Self-Check: Solution

Is the comparison of Password and Confirm Password fields a client-side or server-side validation?

Client-side

Both the values are available at client-end and hence server's involvement is not required.