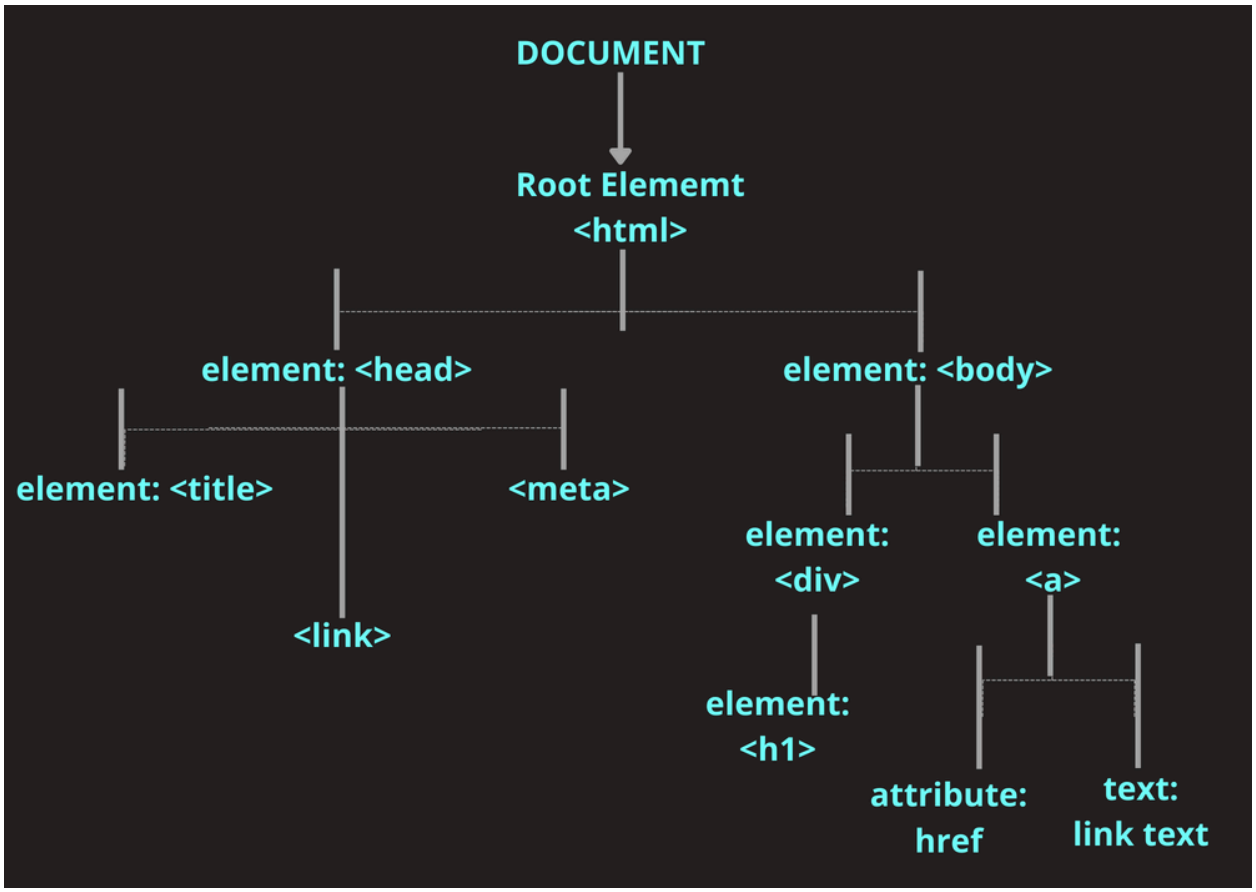
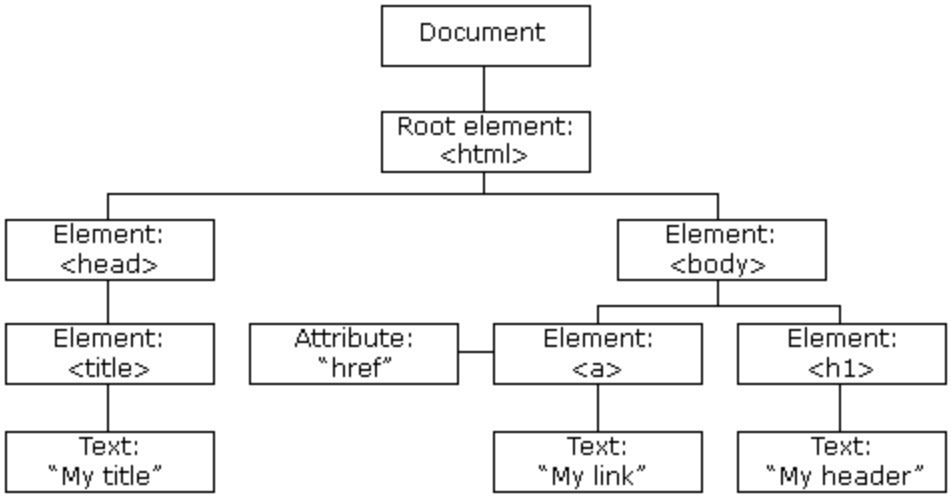
Dom Tree (Document Object Model)



With the HTML DOM, JavaScript can access and change all the elements of an HTML document.

When a web page is loaded, the browser creates a Document Object Model of the page.

The HTML DOM model is constructed as a tree of Objects:



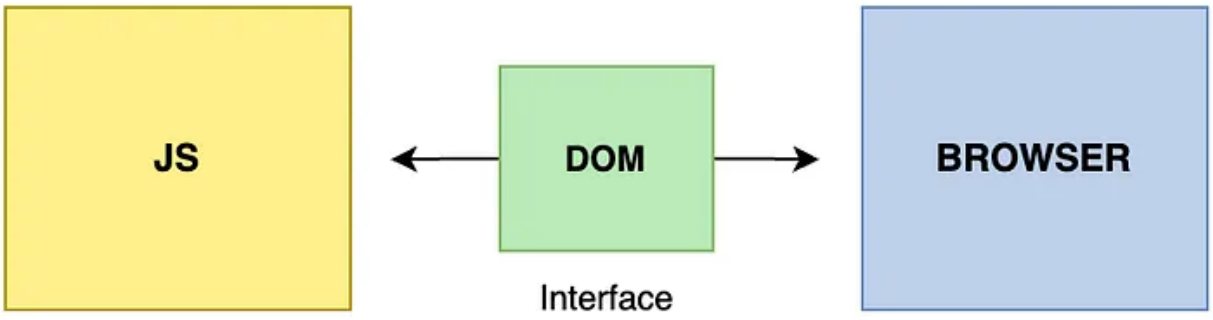
With the object model, JavaScript gets all the power it needs to create dynamic HTML:

- JavaScript can change all the HTML elements in the page
- JavaScript can change all the HTML attributes in the page
- JavaScript can change all the CSS styles in the page
- JavaScript can remove existing HTML elements and attributes
- JavaScript can add new HTML elements and attributes
- JavaScript can react to all existing HTML events in the page
- JavaScript can create new HTML events in the page

The HTML DOM is a standard object model and programming interface for HTML. It defines:

- The HTML elements as objects
- The properties of all HTML elements
- The methods to access all HTML elements
- The events for all HTML elements

In other words: The HTML DOM is a standard for how to get, change, add, or delete HTML elements.



The **Document Object Model** (DOM) is an interface between all JavaScript code and the browser, specifically between HTML documents rendered in and by the browser. Through DOM, we can make JavaScript interact with the browser by creating, modifying, and deleting elements, setting styles, classes, and attributes, and listening and responding to events. This is because a DOM tree, a tree-like structure comprising nodes, is generated from an HTML document.

Here's how it generally works:

1. **Parsing HTML:** When a web page is loaded, the browser parses the HTML document and constructs a

DOM tree based on the HTML structure. This DOM tree represents the hierarchical structure of the document, with each HTML element represented as a node in the tree.

2. **DOM API Access:** JavaScript code running in the browser can access and manipulate this DOM tree using the DOM API. The DOM API provides a set of methods and properties that allow developers to interact with the DOM elements.
3. **Element Selection:** Developers can select specific elements in the DOM using methods like `getElementById`, `getElementsByClassName`, `querySelector`, `querySelectorAll`, etc. These methods return references to DOM elements that can be manipulated further.

4. **Manipulation:** Once a DOM element is selected, developers can modify its attributes, content, styles, and structure using the DOM API. For example, you can change the text content of an element, add or remove classes, update styles, append or remove child elements, etc.
5. **Event Handling:** The DOM API also allows developers to handle user interactions and events such as clicks, keypresses, form submissions, etc. Event listeners can be attached to DOM elements to trigger specific actions or functions when events occur.
6. **Rendering Updates:** As developers make changes to the DOM structure or content using the DOM API, the browser automatically updates the rendered web page to reflect these changes. This dynamic updating process

ensures that the web page remains interactive and responsive to user actions.