**DOM and CSSOM**

The **Document Object Model (DOM**) in JavaScript is a representation of the structure and content of HTML documents. It provides a structured way to access and manipulate elements within a webpage.

Internally, the DOM is implemented as a hierarchical tree structure where each node represents an element, attribute, or text within the document. Here's an overview of the DOM's internal representation:

1. Document Node: At the top of the DOM tree is the document node, which represents the entire HTML document. This node serves as the entry point for accessing and manipulating the document's contents.

2. Element Nodes: Element nodes represent HTML elements such as `<div>`, `<p>`, `<span>`, etc. Each element node contains references to its child nodes (other elements, text nodes, etc.) as well as its parent node.

3. Text Nodes: Text nodes represent the textual content within HTML elements. For example, the text "Hello, world!" within a `<p>` element would be represented as a text node.

4. Attribute Nodes: Attribute nodes represent the attributes of HTML elements. Each element node contains a list of its attributes, and each attribute is represented as a separate node with properties such as name and value.

5. Relationships: The DOM tree structure establishes relationships between nodes. For example, a parent node can have multiple child nodes, and each child node knows its parent node. Sibling nodes share the same parent node.

6. Methods and Properties: The DOM provides a set of methods and properties that allow JavaScript to interact with and manipulate the document's elements. These include methods for querying elements (`getElementById`, `getElementsByClassName`, `querySelector`, etc.), creating new elements (`createElement`), modifying element attributes and content, and more.

7. Event Handling: The DOM also handles events such as user interactions (clicks, mouse movements, keyboard events). JavaScript can register event listeners on specific elements to respond to these events.

Understanding the internal representation of the DOM is crucial for writing efficient and maintainable JavaScript code that interacts with web documents. It helps developers navigate the DOM tree, access elements and their attributes, manipulate the document's structure and content, and handle user interactions effectively.

Bit – Character – Token - Nodes - DOM

**Structure of html**

<html>

<head>

</head>

<body>

</body>

</html>

**CSSOM**

The CSS Object Model (CSSOM) in JavaScript is a representation of the CSS styles applied to elements within a webpage. It provides a structured way to access and manipulate CSS styles programmatically.

Internally, the CSSOM is implemented as a set of objects and properties that represent the styles defined in CSS rules. Here's an overview of the CSSOM's internal representation:

1. Stylesheet Object: The top-level object in the CSSOM represents a CSS stylesheet. A stylesheet object contains references to all the CSS rules defined within it.

2. CSS Rule Objects: CSS rules define the styling properties applied to specific elements or groups of elements. Each CSS rule is represented as an object in the CSSOM. There are different types of CSS rules, such as `CSSStyleRule`, `CSSMediaRule`, `CSSKeyframesRule`, etc., depending on the type of rule defined in the CSS.

3. CSS Style Declaration: The `CSSStyleDeclaration` object represents the style properties associated with a particular element. It contains properties for each style property defined in CSS, such as `color`, `font-size`, `margin`, etc. The values of these properties can be accessed and modified programmatically.

4. Selectors and Specificity: CSS rules often include selectors that determine which elements the styles should apply to. The CSSOM maintains information about selectors and their specificity, which is used to determine which styles take precedence when multiple rules target the same element.

5. Computed Styles: In addition to the styles defined in CSS rules, the CSSOM also provides access to the computed styles of elements. Computed styles are the final styles applied to an element after considering inheritance, cascading, and browser defaults. The `getComputedStyle()` method allows JavaScript to retrieve the computed styles of an element.

6. Methods and Properties: The CSSOM provides methods and properties for querying and manipulating CSS styles. These include methods for accessing specific CSS rules (`document.styleSheets`, `insertRule()`, `deleteRule()`, etc.), modifying style properties of elements (`element.style`), and computing styles (`getComputedStyle()`).

Understanding the internal representation of the CSSOM is essential for developers who need to work with CSS styles dynamically in JavaScript. It allows them to access and modify styles programmatically, create dynamic UIs, and build interactive web applications.