What is the DOM (Document Object Model) ?

The DOM, or Document Object Model, is a programming interface for web documents. It represents the structure of a document as a tree of objects, where each object corresponds to a part of the document, such as elements, attributes, and text. The DOM provides a way for programs to manipulate the structure, style, and content of web documents dynamically.

Key points about the DOM:

- **Tree Structure:** The DOM represents an HTML or XML document as a tree structure, with each node in the tree corresponding to an object in the document.
- **Objects:** Each element, attribute, and piece of text in the document is represented by a specific object in the DOM. These objects can be manipulated using programming languages like JavaScript.
- **Dynamic Interaction:** The DOM enables dynamic interaction with web pages. JavaScript can be used to access, modify, and update the content and structure of a document in real-time, allowing for interactive and responsive user interfaces.
- **Platform-Neutral:** The DOM is platform-neutral, meaning that it provides a standardized way to access and manipulate document content regardless of the underlying operating system or browser.
- **Event Handling:** The DOM allows the registration of event handlers, enabling developers to respond to user actions (such as clicks or key presses) and update the document accordingly.

What Does the HTML DOM Look Like?

Imagine your webpage as a tree

- The document is the root.
- HTML tags like <html>, <head>, and <body> are branches.
- Attributes, text, and other elements are the leaves.

Why is DOM Required?

The DOM is essential because

- **Dynamic Content Updates:** Without reloading the page, the DOM allows content updates (e.g., form validation, AJAX responses).
- **User Interaction:** It makes your webpage interactive (e.g., responding to button clicks, form submissions).
- **Flexibility:** Developers can add, modify, or remove elements and styles in real-time.

- **Cross-Platform Compatibility:** It provides a standard way for scripts to interact with web documents, ensuring browser compatibility.

How the DOM Works?

The DOM connects your webpage to JavaScript, allowing you to:

- Access elements (like finding an <h1> tag).
- Modify content (like changing the text of a <p> tag).
- React to events (like a button click).
- Create or remove elements dynamically.

Properties of the DOM

- **Node-Based:** Everything in the DOM is represented as a node (e.g., element nodes, text nodes, attribute nodes).
- **Hierarchical**: The DOM has a parent-child relationship, forming a tree structure.
- **Live:** Changes made to the DOM using JavaScript are immediately reflected on the web page.
- **Platform-Independent:** It works across different platforms, browsers, and programming languages.

Commonly Used DOM Methods

Methods	Description
getElementById(id)	Selects an element by its ID.
getElementsByClassName(class)	Selects all elements with a given class.
querySelector(selector)	Selects the first matching element.
querySelectorAll(selector)	Selects all matching elements.
createElement(tag)	Creates a new HTML element.

Methods	Description
appendChild(node)	Adds a child node to an element.
remove()	Removes an element from the DOM.
addEventListener(event, fn)	Attaches an event handler to an element.

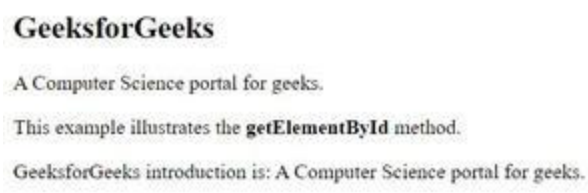
Example: In this example, We use HTML element id to find the DOM HTML element.

HTML

```
<html>

<body>
    <h2>GeeksforGeeks</h2>

    <!-- Finding the HTML Elements by their Id in DOM -->
    <p id="intro">
        A Computer Science portal for geeks.
    </p>
    <p>
        This example illustrates the <b>getElementById</b> method.
    </p>
    <p id="demo"></p>
    <script>
        const element = document.getElementById("intro");
        document.getElementById("demo").innerHTML =
            "GeeksforGeeks introduction is: " + element.innerHTML;
```

```
  </script>
</body>


</html>
```

Output:



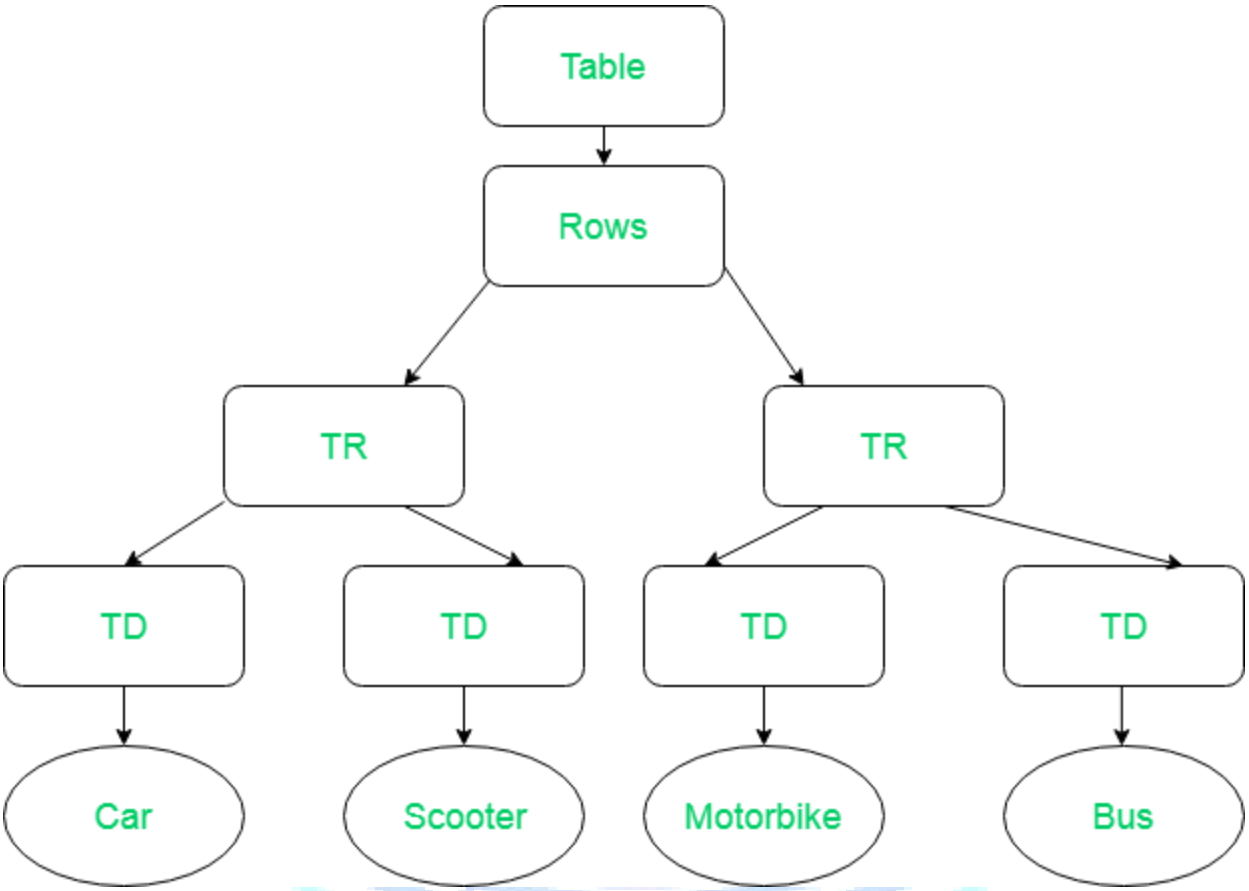
Example : This example describes the representation of the HTML elements in the tree structure.

html

```
1<table>
2
```

1. <ROWS>
2. 3
3. <tr>
4. 4
5. <td>Car</td>
6. 5
7. <td>Scooter</td>
8. 6
9. </tr>
10. 7
11. <tr>
12. 8
13. <td>MotorBike</td>
14. 9
15. <td>Bus</td>
16. 10
17. </tr>
18. 11
19. </ROWS>

20. 12
21. </table>



What DOM is not?

- It is not a **binary description** where it does not define any binary source code in its interfaces.
- It is not used to describe **objects in XML or HTML** whereas the DOM describes XML and HTML documents as objects.
- It is not represented by a **set of data structures**; it is an interface that specifies object representation.
- It does not show the **criticality of objects** in documents i.e it doesn't have information about which object in the document is appropriate to the context and which is not.

Levels of DOM

DOM consisted of multiple levels, each representing different aspect of the document.

- **Level 0:** Provides a low-level set of interfaces.
- **Level 1:** DOM level 1 can be described in two parts: **CORE** and **HTML**.
    - **CORE** provides low-level interfaces that can be used to represent any structured document.
    - **HTML** provides high-level interfaces that can be used to represent HTML documents.
- **Level 2:** consists of six specifications: **CORE2**, **VIEWS**, **EVENTS**, **STYLE**, **TRAVERSAL**, and **RANGE**.
    - **CORE2**: extends the functionality of CORE specified by DOM level 1.
    - **VIEWS**: views allows programs to dynamically access and manipulate the content of the document.
    - **EVENTS**: Events are scripts that are either executed by the browser when the user reacts to the web page.
    - **STYLE**: allows programs to dynamically access and manipulate the content of style sheets.
    - **TRAVERSAL**: This allows programs to dynamically traverse the document.
    - **RANGE**: This allows programs to dynamically identify a range of content in the document.
- **Level 3:** consists of five different specifications: **CORE3**, **LOAD and SAVE**, **VALIDATION**, **EVENTS**, and **XPATH**.
    - **CORE3**: extends the functionality of CORE specified by DOM level 2.
    - **LOAD and SAVE**: This allows the program to dynamically load the content of the XML document into the DOM document and save the DOM Document into an XML document by serialization.
    - **VALIDATION**: This allows the program to dynamically update the content and structure of the document while ensuring the document remains valid.
    - **EVENTS**: extends the functionality of Events specified by DOM Level 2.
    - **XPATH**: XPATH is a path language that can be used to access the DOM tree.

The Document Object Model (DOM) organizes a web document as a tree of nodes, making it easy to navigate and manipulate using programming languages like JavaScript.

Key Node Types

- **Document Node**: The root of the tree, representing the whole document.
- **Element Nodes**: Represent HTML or XML elements and can contain other element nodes, text nodes, and attributes.
- **Text Nodes**: Contain text content of elements and are always leaf nodes, meaning they cannot have child nodes.
- **Attribute Nodes**: Associated with element nodes but not considered children; they define characteristics such as id or class.
- **Comment Nodes**: Represent comments within the HTML or XML files, visible in the DOM tree but not rendered in the user interface.

Tree Structure

The DOM tree's hierarchical structure includes parent, child, and sibling relationships:

- **Parent Nodes**: Nodes with child nodes under them in the tree.
- **Child Nodes**: Nodes directly nested within another node. For example, an <li> element inside an <ul>.
- **Sibling Nodes**: Nodes that share the same parent. For instance, two elements within the same <body> element are siblings.

This structured approach allows developers to efficiently target and manipulate elements, enabling dynamic updates and interactions on web pages.

Differences Between HTML DOM and DOM

Although "HTML DOM" and "DOM" are often used interchangeably, they serve distinct purposes in web development.

While both provide ways to interact with a document, the HTML DOM is specifically designed for HTML documents, offering specialized tools and methods, whereas the DOM provides a more generalized structure for interacting with any type of document, such as XML or HTML.

HTML DOM

The **HTML DOM** is a specialized version of the DOM tailored for HTML documents. It provides specific methods and properties like getElementById() and innerHTML for interacting with HTML elements and events.

DOM

The **DOM** (Document Object Model) is a broader interface applicable to various document types, including HTML and XML. It offers a generic way to access and manipulate document content but lacks HTML-specific features.

Key Differences

- The HTML DOM is a subset of the DOM, designed specifically for HTML documents.
- DOM is generic, while HTML DOM adds methods tailored to HTML manipulation.

Fundamental Data Types

The DOM uses various data types to represent different parts of a document:

- **Document**: Represents the entire HTML or XML document.
- **Element**: Represents an element in the document (e.g., , <p>).
- **Attribute**: Represents an attribute of an element (e.g., class, id).
- **Text**: Represents the text content of an element.
- **Node**: Represents any single node in the DOM tree (elements, attributes, text, etc.).

DOM Interfaces

The DOM API provides various interfaces to interact with and manipulate the DOM:

- **Document**: The entry point to the DOM, representing the entire HTML or XML document.
- **Element**: Represents an element in the DOM and provides methods to manipulate it (e.g., getElementById, getElementsByClassName).
- **Node**: Represents a single node in the DOM tree and provides methods for interacting with nodes (e.g., appendChild, removeChild).
- **Event**: Represents events that can occur in the DOM, such as clicks, key presses, and more.