🌍 1. What is the DOM?

The Document Object Model (DOM) is a programming interface for HTML and XML documents.
It represents a web page as a hierarchical tree of objects that can be accessed and modified with JavaScript.

When a browser loads a web page, it:

1. **Parses the HTML.**
2. **Builds a DOM Tree.**
3. **Exposes it to JavaScript via the `document` object.**

💡 In essence:

The DOM connects HTML (structure) + CSS (style) + JavaScript (logic).

 2. DOM Tree Representation

Example HTML:

```
<!DOCTYPE html>
<html>
 <head>
  <title>My Page</title>
 </head>
 <body>
  <h1 id="main">Hello, DOM!</h1>
  <p class="intro">Learning the Document Object Model.</p>
 </body>
</html>
```

DOM Tree:

```
Document
 └── html
     ├── head
     │    └── title
     │         └── #text: "My Page"
     └── body
          ├── h1#main
          │    └── #text: "Hello, DOM!"
          └── p.intro
               └── #text: "Learning the Document Object Model."
```

✅ Every tag, text, attribute, and comment is a node in the DOM tree.

🧱 3. Types of DOM Nodes

Node Type	Example	Description
Document	document	The root node of the DOM tree
Element	<p>, , <body>	Represents HTML tags
Text	"Hello"	Represents text content inside an element
Attribute	id="main"	Represents element attributes
Comment	<!-- comment -->	Represents comments in HTML

🔍 4. Accessing the DOM

DOM can be accessed using the global object `document`.

By ID:

```
document.getElementById("main");
```

By Class:

```
document.getElementsByClassName("intro");
```

By Tag:

```
document.getElementsByTagName("p");
```

By CSS Selector (Modern, Recommended):

```
document.querySelector(".intro");     // first match
document.querySelectorAll("p");       // all matches
```

🧬 5. DOM Relationships

Example:

```
<div id="parent">
  <h2>Child 1</h2>
  <p>Child 2</p>
</div>
```

Relationships:

- `div` → **Parent Node**
- `h2`, `p` → **Child Nodes**
- `h2` ↔ `p` → **Sibling Nodes**

Access in JavaScript:

```
let div = document.getElementById("parent");
console.log(div.children);         // [h2, p]
console.log(div.firstElementChild); // h2
console.log(div.lastElementChild);  // p
console.log(div.parentNode);       // body
```

✍ 6. Manipulating DOM Elements

Change Text:

```
document.getElementById("main").textContent = "Welcome Rudraksh!";
```

Change HTML:

```
document.querySelector(".intro").innerHTML = "<b>Learning DOM</b> dynamically!";
```

Change Attributes:

```
document.querySelector("img").setAttribute("src", "image.png");
```

Change CSS Styles:

```
document.querySelector("h1").style.color = "blue";
document.querySelector("h1").style.fontSize = "32px";
```

Add / Remove / Toggle Classes:

```
let box = document.querySelector("#box");
box.classList.add("active");
box.classList.remove("hidden");
box.classList.toggle("highlight");
```

⚡ 7. Creating, Appending, Removing Elements

```
// Create
let newDiv = document.createElement("div");
newDiv.textContent = "This div was created dynamically.";

// Append
document.body.appendChild(newDiv);

// Remove
document.querySelector("p").remove();
```

Insert before a specific node:

```
let list = document.querySelector("ul");
let newItem = document.createElement("li");
newItem.textContent = "New Item";
list.insertBefore(newItem, list.firstChild);
```

🎨 8. DOM Events (Interactivity)

Event Syntax:

```
element.addEventListener(eventType, callbackFunction);
```

Example:

```
<button id="btn">Click Me!</button>
<p id="msg"></p>

<script>
document.getElementById("btn").addEventListener("click", function() {
  document.getElementById("msg").textContent = "Button clicked!";
});
</script>
```

Common Event Types:

Event	Description
click	When element is clicked
mouseover / mouseout	Mouse enter/leave
keydown / keyup	Key pressed/released
scroll	On page scroll
load	Page fully loaded
submit	Form submission
change	Input value changed

---

9. Traversing the DOM

Navigate the DOM Tree:

```
let body = document.body;

body.firstElementChild;     // first child
body.lastElementChild;      // last child
body.children;              // HTMLCollection of children
body.parentNode;            // parent (document)
body.nextElementSibling;    // next sibling
```

---

10. Working with Forms

```
<form id="myForm">
  <input type="text" id="name" placeholder="Enter your name">
  <button type="submit">Submit</button>
</form>

<script>
document.getElementById("myForm").addEventListener("submit", (e) => {
  e.preventDefault(); // prevent reload
  let name = document.getElementById("name").value;
  alert("Hello, " + name);
});
</script>
```

---

11. Expanded DOM Cheatsheet (Comprehensive)

Property / Method	Description	Example
.innerHTML	Gets/sets HTML inside element	div.innerHTML = "Hello";
.textContent	Gets/sets only text	p.textContent = "Text";
.innerText	Gets/sets visible text (excludes hidden)	element.innerText;
.outerHTML	Includes the element itself	div.outerHTML;
.getAttribute()	Gets attribute value	img.getAttribute("src");
.setAttribute()	Sets attribute value	img.setAttribute("alt", "photo");
.removeAttribute()	Removes an attribute	a.removeAttribute("target");
.hasAttribute()	Checks attribute presence	div.hasAttribute("id");
.classList.add()	Add a class	el.classList.add("active");
.classList.remove()	Remove class	el.classList.remove("hide");
.classList.toggle()	Toggle class	el.classList.toggle("on");

Property / Method	Description	Example
.classList.contains()	Check if class exists	el.classList.contains("show");
.style.property	Modify CSS	div.style.background = "red";
.appendChild()	Append element	parent.appendChild(child);
.removeChild()	Remove element	parent.removeChild(child);
.replaceChild()	Replace element	parent.replaceChild(newNode, oldNode);
.createElement()	Create a new element	document.createElement("div");
.createTextNode()	Create text node	document.createTextNode("Hi");
.querySelector()	First match	document.querySelector(".box");
.querySelectorAll()	All matches	document.querySelectorAll("p");
.addEventListener()	Attach event	btn.addEventListener("click", fn);
.removeEventListener()	Remove event	btn.removeEventListener("click", fn);
.parentNode	Parent node	child.parentNode;
.children	Child elements	div.children;
.firstElementChild	First child	div.firstElementChild;
.lastElementChild	Last child	div.lastElementChild;
.nextElementSibling	Next sibling	div.nextElementSibling;
.previousElementSibling	Previous sibling	div.previousElementSibling;
.contains()	Checks if a node is inside	parent.contains(child);
.cloneNode(true)	Duplicates a node	node.cloneNode(true);
.insertBefore()	Insert before another	parent.insertBefore(newNode, refNode);
.nodeName	Returns tag name	element.nodeName;
.nodeType	Node type (1, 3, etc.)	element.nodeType;
.nodeValue	Value for text/attr node	textNode.nodeValue;
.getBoundingClientRect()	Element's size/position	div.getBoundingClientRect();
.scrollIntoView()	Scrolls to element	element.scrollIntoView();
.focus()	Focus on input	input.focus();
.blur()	Removes focus	input.blur();
.dataset	Access data-* attributes	div.dataset.user;
.matches()	Matches selector	el.matches(".active");

12. DOM Events in Detail

Event Object Properties

Property	Description
event.type	Type of event (click, keydown)
event.target	Element that triggered the event
event.preventDefault()	Stops default behavior
event.stopPropagation()	Stops bubbling
event.key	Key pressed on keyboard event
event.clientX, event.clientY	Mouse coordinates

---

13. Real DOM vs Virtual DOM

Feature	Real DOM	Virtual DOM
Definition	Browser's live DOM tree	In-memory representation
Speed	Slower (re-renders entire tree)	Faster (diff & patch)
Used In	Vanilla JS, jQuery	React, Vue
Efficiency	Low	High
Updates	Direct	Batched

---

14. Browser Object Model (BOM) vs DOM

Feature	DOM	BOM
Definition	Represents HTML structure	Represents browser environment
Access Object	document	window
Example	document.body	window.location.href

---

15. Example: Interactive Web Page

```
<!DOCTYPE html>
<html>
<head>
  <title>Interactive DOM</title>
  <style>
    .highlight { color: red; font-weight: bold; }
  </style>
</head>
<body>
  <h1 id="title">Welcome</h1>
  <button id="btn">Change Text</button>
  <p class="desc">Click the button to see the change.</p>

  <script>
    const title = document.getElementById("title");
    const desc = document.querySelector(".desc");
    const btn = document.getElementById("btn");

    btn.addEventListener("click", () => {
      title.textContent = "Hello, Rudraksh!";
      desc.classList.toggle("highlight");
    });
  </script>
</body>
</html>
```

---

16. Real-World Applications of DOM

- Form validation
- Dynamic web content
- Popups, modals, and dropdowns
- Animations and transitions
- Live data dashboards
- Browser-based games
- SPA (Single Page Application) frameworks

---

17. Visualization of DOM Tree

```
Document
└── html
    ├── head
    │   └── title → "Page Title"
    └── body
        ├── h1 → "Heading"
        ├── p → "Paragraph"
        └── button → "Click Me"
```

---

18. Best Practices for DOM Manipulation

✅ Use querySelector() and querySelectorAll() for flexibility.
✅ Always wait for DOMContentLoaded before manipulating DOM.
✅ Avoid too many reflows (combine multiple changes).
✅ Use classList instead of direct style.
✅ Cache DOM selections in variables.
✅ Use event delegation for many similar elements.