✨ LECURE 20 — ACCESSING DOM ELEMENTS

🍒 FIRST-THOUGHT PRINCIPLE (DESI STYLE)

👉 **DOM = ek joint family ka ped** 🌳

- **Parent** → Maa-Baap
- **Child** → Bachche
- **Sibling** → Bhai-Behen

📌 Bilkul ghar ke rishton jaise hi DOM me bhi **Parent → Child → Sibling** relationship hota hai.

🧠 TEXT KO SAMJHNE KA DESI TRICK

- **innerHTML** → Kapde + jewellery + makeup 👚💍
- **innerText** → Jo saamne dikh raha hai 👀
- **textContent** → Andar ka sach bhi (hidden bhi) 🧠

🏗️ EXAMPLE HTML (BASE STRUCTURE)

```
<!DOCTYPE html>
<html>
<head>
  <title>Access DOM</title>
</head>
<body>
  <h1 id="title" class="heading">Hello DOM</h1>
  <h1 class="heading">Second Heading</h1>
```

```
  <h1>Third Heading</h1>
  <p id="first">First Para</p>
  <p>Second Para</p>

  <ul id="list">
    <li>HTML</li>
    <li>CSS</li>
    <li>JavaScript</li>
  </ul>
</body>
</html>
```

🔎 1. ACCESSING BY CSS SELECTORS

🔷 (a) querySelector() — SINGLE ELEMENT

📌 Definition

👉 CSS selector ke through **pehla matching element** return karta hai.

```
const h1 = document.querySelector('#title');
console.log(h1.innerHTML);
```

📤 Output

```
Hello DOM
```

⚠ **Important**

- Chahe 100 matching ho
- **Sirf pehla hi milega**

🔷 (b) querySelectorAll() — MULTIPLE ELEMENTS

📌 Definition

👉 CSS selector se **saare matching elements** ka
👉 **static NodeList** return karta hai.

```
const allH1 = document.querySelectorAll('h1');
```

🔁 For Loop

```
for (let i = 0; i < allH1.length; i++) {

  console.log(allH1[i].innerText);

}
```

📤 Output

```
Hello DOM

Second Heading

Third Heading
```

🧺 WHAT IS NODELIST?

📌 Definition

👉 **NodeList** ek array-jaisi collection hoti hai
👉 Jo multiple DOM nodes ko store karti hai

```
NodeList(3) [

  0: <h1 id="title">Hello DOM</h1>,

  1: <h1>Second Heading</h1>,

  2: <h1>Third Heading</h1>

]
```

🔁 NODELIST ITERATION

✅ forEach (Direct Support)

```
allH1.forEach(val => console.log(val.innerText));
```

✅ for...of

```
for (let val of allH1) {
  console.log(val.innerText);
}
```

🔄 NodeList → Array Conversion (WHY?)

❌ `map`, `filter` directly nahi milte
✅ Convert karo:

```
const arr1 = Array.from(allH1);
const arr2 = [...allH1];
```

🏷️ 2. ACCESSING BY TAG NAME

📌 Definition

👉 Tag name jaise `h1`, `p`, `li`, `div`

```
let liItems = document.getElementsByTagName('li');

for (let i = 0; i < liItems.length; i++) {
  liItems[i].style.color = "blue";
}
```

📤 Output

👉 Saare `<li>` **blue color** me ho jayenge

⚠ **Return Type**

- **HTMLCollection (LIVE)**

👨‍👩‍👧 3. ACCESSING BY RELATIONSHIP

🔷 (a) Parent & Child

```
const ul = document.getElementById('list');


console.log(ul.parentNode);

console.log(ul.children);

console.log(ul.childNodes);
```

📤 Output Explanation

- `children` → sirf elements
- `childNodes` → elements + text nodes (spaces, line breaks)

🔷 (b) First & Last Child

Property	Meaning
firstChild	Pehla node (text bhi)
firstElementChild	Pehla element

lastChild	Last node
lastElementChild	Last element

```
console.log(ul.firstElementChild); // <li>HTML</li>

console.log(ul.lastElementChild);  // <li>JavaScript</li>
```

🔷 (c) Sibling Nodes

```
const p = document.getElementById('first');


console.log(p.nextSibling);          // #text

console.log(p.nextElementSibling);   // <p>Second Para</p>
```

💡 **Best Practice**
👉 **Always use `ElementSibling`**
Text nodes se bach jaoge.

📝 4. innerHTML vs innerText vs textContent

```
<p id="demo">
  Hello <b style="display:none">Hidden</b> World
</p>


demo.innerHTML;

// Hello <b>Hidden</b> World


demo.innerText;

// Hello World
```

```
demo.textContent;

// Hello Hidden World
```

✅ QUICK RECAP (TABLE)

🔷 Selectors

Method	Returns
querySelector	First match
querySelectorAll	NodeList (static)
getElementsByTagName	HTMLCollection (live)

🔷 Relationships

Property	Meaning
parentElement	Parent
children	Elements only
childNodes	All nodes
nextElementSibling	Next element

🔷 Text

Property	Kya milta hai
innerHTML	HTML + text
innerText	Visible text
textContent	All text
