Below is a **clean, exam-ready, deeply explained, and structured rewrite** of your topic **“DOM Elements”**, keeping **all concepts**, but removing repetition, correcting terminology, and adding **internal understanding** so it’s easy for **viva + theory + practical use**.

📘 JavaScript DOM Elements

DOM Elements are the **HTML elements represented as objects** inside the **Document Object Model (DOM)**.
Using JavaScript, we can **access, modify, replace, remove, and attach events** to these elements.

When a web page loads:

- Browser parses HTML
- Creates a **DOM tree**
- Each HTML tag becomes a **DOM element object**
- JavaScript interacts with these objects

👉 **HTML itself is static — DOM makes it programmable.**

🧠 Internal Picture (Very Important)

```css
document
 └── html
      ├── head
      └── body
           ├── div
           ├── p
           └── button
```

Each box above is a **DOM element (object)** with:

- properties (innerHTML, style, id, classList...)
- methods (appendChild, remove, replaceChild...)

🔹 Accessing DOM Elements

JavaScript provides **multiple selector methods** to access elements.

1️⃣ Access Element by `id`

Each `id` is **unique**.

Syntax

```js
```

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

Example

```html
<div id="output"></div>

<script>
 document.getElementById("output").innerHTML =
   "Element accessed using ID";
</script>
```

📌 **Returns:** single element or `null`

📌 **Fastest & most commonly used**

2️⃣ Access Element by `name`

Used mainly with **forms**.

Syntax

```js
document.getElementsByName("name");
```

📌 **Returns:** NodeList (array-like)

Example

```html
<div name="text">Hello World!</div>

<script>
 const elems = document.getElementsByName("text");
 console.log(elems[0].innerHTML);
</script>
```

3️⃣ Access Element by `className`

One element can have **multiple classes**.

Syntax

```js
document.getElementsByClassName("className");
```

📌 **Returns:** live HTMLCollection

Example

```html
<div class="fruit">Apple</div>

<script>
  const items = document.getElementsByClassName("fruit");
  console.log(items[0].innerHTML);
</script>
```

4️⃣ Access Element by `tagName`

Selects elements by HTML tag.

Syntax

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

📌 **Returns:** HTMLCollection

Example

```html
<p>First</p>
<p>Second</p>

<script>
  const p = document.getElementsByTagName("p");
  console.log(p[0].innerHTML);
</script>
```

⚠️ Summary of Access Methods

Method	Returns	Notes
getElementById	Element	Unique

Method	Returns	Notes
getElementsByName	NodeList	Mostly forms
getElementsByClassName	HTMLCollection	Live
getElementsByTagName	HTMLCollection	Live

🔹 Modifying DOM Elements

Once accessed, elements can be **modified dynamically**.

✏️ Modifying Text Content

`textContent`

- Changes **only text**
- Ignores HTML tags
- Safe from XSS

```js
element.textContent = "New text";
```

Example

```html
<p id="msg">Hello</p>

<script>
  document.getElementById("msg").textContent = "Updated text";
</script>
```

`innerHTML` (⚠️ Powerful but risky)

- Inserts HTML
- Replaces existing content

```js
element.innerHTML = "<b>Hello</b>";
```

🔄 Replacing Child Elements

`replaceChild(newNode, oldNode)`

Syntax

```js
parent.replaceChild(newNode, oldNode);
```

Example

```html
<div id="box">
  <p>Old Text</p>
</div>

<script>
  const parent = document.getElementById("box");
  const newText = document.createTextNode("Hello World");
  parent.replaceChild(newText, parent.childNodes[1]);
</script>
```

📌 Works at **node level** (text, comment, element)

🔹 Adding Events to DOM Elements

Events allow **interaction**.

`addEventListener()`

Syntax

```js
element.addEventListener(event, callback);
```

Example

```html
<div id="box" style="background:red;color:white">Click Me</div>
<p id="output"></p>

<script>
  document.getElementById("box")
    .addEventListener("click", function () {
      document.getElementById("output").innerHTML =
        "Div clicked!";
    });
</script>
```

✔️ Multiple events allowed

✔️ Preferred over inline events

🔹 Important DOM Element Methods (Grouped)

🔧 Attribute Methods

- `setAttribute()`
- `getAttribute()`
- `removeAttribute()`
- `hasAttribute()`

🧱 Structure Manipulation

- `appendChild()`
- `removeChild()`
- `replaceChild()`
- `insertBefore()`
- `cloneNode()`
- `remove()`

🎯 Selector Helpers

- `querySelector()`
- `querySelectorAll()`
- `closest()`
- `matches()`
- `contains()`

📐 Position & Size

- `getBoundingClientRect()`
- `scrollIntoView()`

🎧 Event Control

- `addEventListener()`
- `removeEventListener()`
- `click()`
- `focus()`
- `blur()`

🔹 Important DOM Element Properties (Grouped)

📄 Content

- `innerHTML`
- `innerText`
- `textContent`
- `outerHTML`
- `outerText`

🎨 Style & Class

- `style`
- `className`
- `classList`

👨‍👩‍👧‍👦 Relationships

- `parentNode`
- `parentElement`
- `children`
- `childNodes`
- `firstChild`
- `firstElementChild`
- `lastChild`
- `nextSibling`

📏 Layout & Size

- `offsetWidth`
- `offsetHeight`
- `offsetTop`
- `offsetLeft`
- `clientWidth`
- `clientHeight`
- `scrollTop`

🆔 Identity

- `id`
- `tagName`
- `nodeName`
- `nodeType`
- `nodeValue`

🧠 Internal Difference (Exam Favorite)

Property	Includes
childNodes	elements + text + comments
children	only elements
innerText	visible text
textContent	all text
innerHTML	text + HTML

📌 One-Line Exam Summary

> **DOM Elements are HTML elements represented as objects in the DOM, allowing JavaScript to dynamically access, modify, replace, and control web page content and behavior.**

If you want next, I strongly recommend:

- 🔹 **DOM Traversal (parent/child/sibling) – deep**
- 🔹 **querySelector vs getElementById**
- 🔹 **innerHTML vs textContent vs innerText**
- 🔹 **DOM node types (1, 3, 8)**
- 🔹 **Exam MCQs + viva questions**

Just say **next** 👍