Below is a **clean, exam-ready, deeply explained, and internally consistent version** of **HTML DOM (Document Object Model)**.
I've rewritten your content to be **conceptually strong**, **internals-focused**, and **easy to revise**, without changing meaning.

🌐 HTML DOM (Document Object Model)

The **HTML DOM** allows JavaScript to **access, modify, create, and delete** HTML elements, attributes, styles, and events dynamically.

In simple terms:

> **HTML builds structure**
> **CSS styles it**
> **JavaScript + DOM makes it interactive**

🧠 Big Picture (Before Going Deep)

- Every web page runs **inside a browser window**
- That browser window is represented by a **Window object**
- Inside that window, the loaded HTML becomes a **Document object**
- The **DOM** is the structured representation of that document

So JavaScript **never directly touches HTML**.
Instead, it talks to the **DOM**, and the DOM updates the page.

📌 What is DOM?

DOM stands for **Document Object Model**.

It is:

- A **programming interface**
- A **tree-based representation** of a document
- A **W3C standard**

Key Definition (Exam Friendly)

> The DOM is a programming interface that represents an HTML or XML document as a tree of objects, allowing programs to dynamically access and manipulate structure, content, and style.

🌳 DOM as a Tree Structure

The DOM represents a web page as a **tree of nodes**.

- Each HTML element → **Node**

- Text inside elements → **Text Node**
- Attributes → **Attribute Nodes**
- Entire document → **Document Node**

Example HTML:

```
html

<html>
 <body>
  <h1>Hello</h1>
 </body>
</html>
```

DOM Tree (Conceptually):

```
css

Document
└── html
    └── body
        └── h1
            └── "Hello"
```

📌 This tree structure is **why DOM traversal and manipulation is possible**

📄 What is HTML DOM?

Problem:

JavaScript **cannot directly interact with raw HTML**

Solution:

When a page loads, the browser:

1. Parses HTML
2. Converts it into a **DOM tree**
3. Exposes that tree through the **Document object**

This DOM is called the **HTML DOM**

🪟 Window Object (Top of Hierarchy)

The **Window object** represents the browser window or tab.

- It is the **global object** in browsers

- All global variables and functions become properties of `window`
- It sits at the **top of the DOM hierarchy**

```javascript
window
└── document
```

Examples:

```js
window.alert("Hello");
alert("Hello"); // same thing
```

📄 Document Object

The **Document object** represents the loaded HTML page.

Using `document`, JavaScript can:

- Access elements
- Modify content
- Change styles
- Add/remove elements
- Attach events

Example:

```js
document.getElementById("title").innerHTML = "New Title";
```

📌 Every `<iframe>` has its **own document and DOM**

🗒️ Form Object

When HTML contains:

```html
<form> ... </form>
```

The browser creates a **Form object**

- It represents the entire form

- It contains all form controls inside it

🧩 Form Control Elements

Inside a form:

- `<input>`
- `<textarea>`
- `<select>`
- `<button>`

Each becomes an **object** accessible through DOM.

Example:

```
js

document.forms[0].elements[0].value
```

🏗️ DOM Object Hierarchy (Simplified)

```
css

Window
└── Document
    └── HTML
        └── Body
            ├── Form
            │   └── Input
            └── Div
```

This hierarchy is **critical for event bubbling, capturing, and traversal**

📚 Types of DOM

1️⃣ Legacy DOM

- Oldest DOM model
- Limited access (forms, images, links)
- Still supported for backward compatibility

2️⃣ W3C DOM (Modern & Standard)

Defined by **W3C**, supported by all modern browsers.

Provides:

- Full access to document

- Standardized methods and properties

W3C DOM Types

DOM Type	Description
Core DOM	Works with all document types
HTML DOM	Specifically for HTML documents
XML DOM	Specifically for XML documents

❓ Why is DOM Required?

Without DOM:

- JavaScript cannot modify HTML
- Web pages would be static

With DOM, JavaScript can:

✅ Access elements
✅ Modify content
✅ Change CSS
✅ Add / remove elements
✅ Update attributes
✅ Add animations
✅ Attach event handlers
✅ Create custom events

Example:

```
js

document.createElement("div");
document.body.appendChild(div);
```

🔌 DOM Interfaces (Very Important)

DOM interfaces are **objects provided by the browser** that allow interaction with the DOM.

Common DOM Interfaces

Interface	Purpose
Document	Represents entire HTML document

Interface	Purpose
Element	Represents a single HTML element
Node	Base interface for all DOM nodes
NodeList	Collection of nodes (live/static)
HTMLCollection	Live collection of HTML elements
DOMTokenList	Used for classList operations
Attr	Represents an attribute node

Example: DOMTokenList

```js
element.classList.add("active");
element.classList.remove("hidden");
```

Example: NodeList

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

🎯 Final Summary (Revision Ready)

- DOM is a **tree-based representation** of a web page
- JavaScript interacts with HTML **only through DOM**
- `window` → top-level object
- `document` → represents HTML page
- DOM enables **dynamic, interactive web applications**
- W3C DOM is the **modern standard**

If you want, next we can cover:

- DOM Nodes (types + internals)
- DOM Traversal (parent/child/sibling)
- DOM Manipulation methods
- DOM vs BOM
- Interview MCQs on DOM
- DOM performance best practices

Just say **next** 🚀