Introduction to the DOM

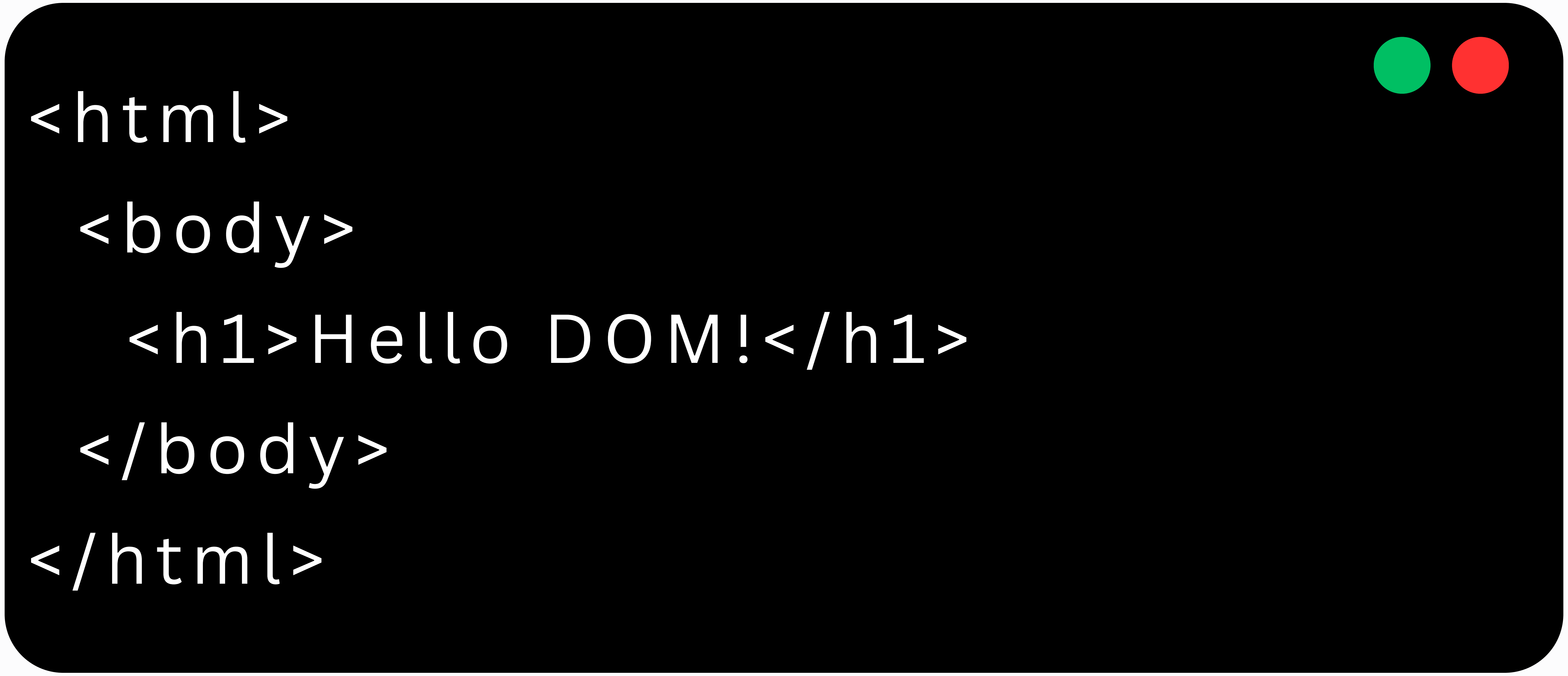
By: Gino Fernando

What is the DOM?

The Document Object Model (DOM) is a programming interface for HTML and XML documents.

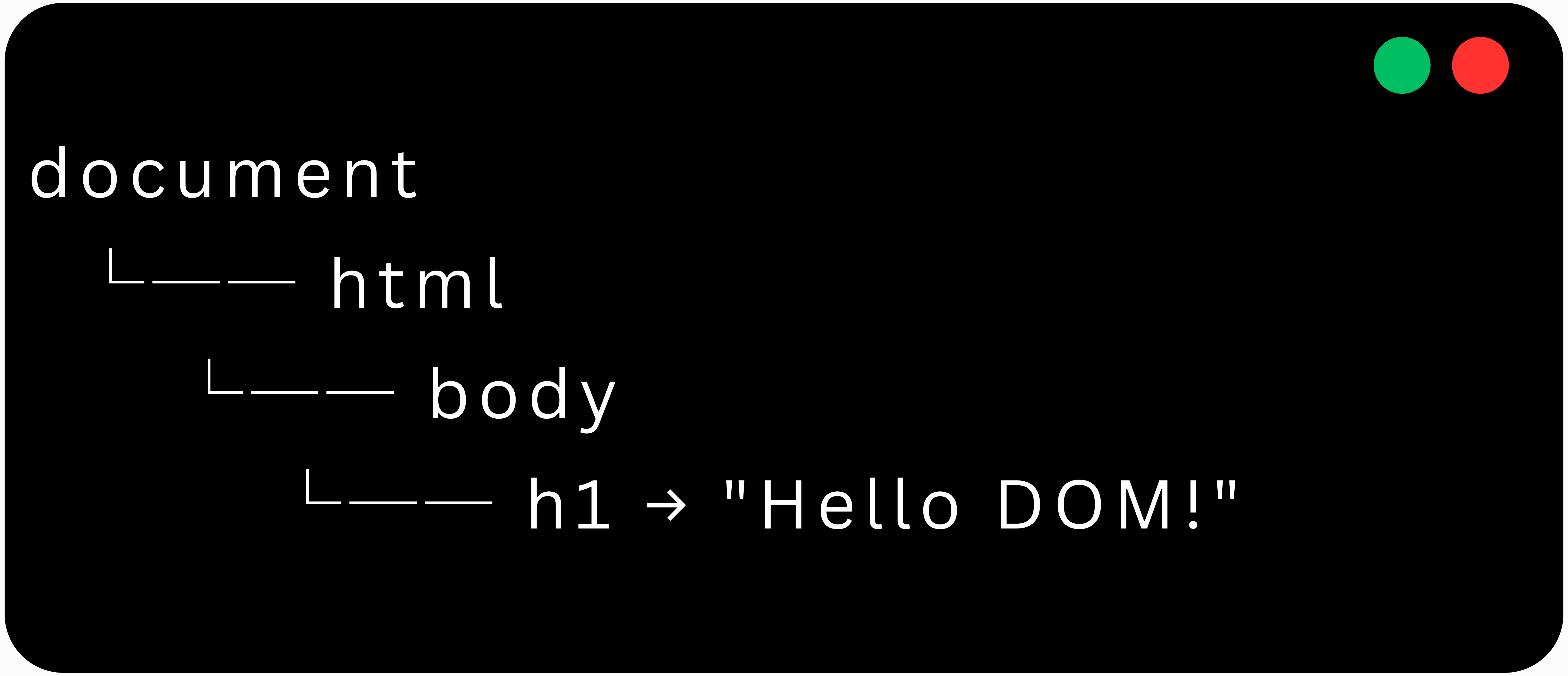
- It represents the page as a tree of nodes.
- Each element (like <h1>, <p>, <div>) is a node in the tree.
- JavaScript can use the DOM to read, modify, add, or delete elements.

Think of DOM as a bridge between HTML and JavaScript.



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

- The html element is the root node.
- body is a child of html.
- h1 is a child of body.



```
document
  └── html
        └── body
              └── h1 → "Hello DOM!"
```

- Each HTML element = a branch or a leaf
- Parent-child relationships connect them
- You can move around, edit, or grow the tree using JavaScript!

HTML, CSS, and DOM

- HTML = content (what you see)
- CSS = style (how it looks)
- DOM = bridge (lets JavaScript change things)

```
<p id="demo">I am text.</p>

<script>
  document.getElementById("demo").style.color = "red";
</script>
```

Using Browser Console

- You can explore the DOM directly:
- Open browser → Right click → Inspect.
- Go to Console tab.
- Type things like:

```
document.title    // shows page title
document.body      // shows <body> element
```

The document Object

- The document object = the whole webpage.
- It has info and tools to reach any part of the page.

```
document.title    // Title of page
document.URL       // Website link
document.body      // Everything inside <body>
```

Selecting Elements: getElementById()

- The most common way to select an element.
- You need to know the element’s id.
- Always returns one element.
- Use when you are sure the element has a unique id.

```
<p id="greet">Hello!</p>

<script>
  let el = document.getElementById("greet");
  el.textContent = "Hi there!";
</script>
```

Selecting Elements: getElementsByClassName()

- Selects all elements with the same class.
- Returns a list (like an array).

```
<p class="note">Note 1</p>
<p class="note">Note 2</p>

<script>
 let notes = document.getElementsByClassName("note");
 notes[0].style.color = "blue";   // First note
 notes[1].style.color = "green";  // Second note
</script>
```

Modern Selection: querySelector() & querySelectorAll()

- Newer and more flexible way.
- Uses CSS selectors.

```
<p class="note">First</p>
<p class="note">Second</p>

<script>
  let firstNote = document.querySelector(".note"); // First match
  firstNote.style.color = "red";

  let allNotes = document.querySelectorAll(".note"); // All matches
  allNotes[1].style.color = "blue"; // Second one
</script>
```

Node vs Element

- DOM has nodes (all things in the tree).
- Nodes include:
- Elements (<p>, <div>)
- Attributes (id, class)
- Text nodes (the text inside elements)

```
<p id="greet">Hello</p>
```

```
p (element)
 ├─── id="greet" (attribute)
 └─── "Hello" (text node)
```

Changing Element Content

- We can change text or HTML inside an element.
- innerHTML → changes HTML inside.
- textContent → changes only the text.

```
<p id="demo">Hello</p>

<script>
  document.getElementById("demo").innerHTML = "<b>Hi!</b>";
  // Now it shows: Hi! (bold)
</script>
```

Modifying Attributes

You can change attributes like src, href, alt, etc.

```
<img id="pic" src="old.png" alt="Old Image">

<script>
  let img = document.getElementById("pic");
  img.src = "new.png";        // Change image
  img.alt = "New Image";       // Change alt text
</script>
```

Changing Styles

You can change CSS styles with the style property.

```
<p id="text">Style me!</p>

<script>
 let t = document.getElementById("text");
 t.style.color = "blue";
 t.style.fontSize = "20px";
</script>
```

Changing Styles

Better Way → Use classList

```
<p id="text">Style me!</p>

<script>
 let t = document.getElementById("text");
 t.classList.add("highlight");   // Adds CSS class
 t.classList.remove("old");      // Removes CSS class
</script>
```

Adding New Elements

We can create new HTML elements using JS.

Steps:

1. createElement() → make a new element.
2. appendChild() → add it to the page.

```
<div id="box"></div>

<script>
 let newP = document.createElement("p");
 newP.textContent = "I am new here!";
 document.getElementById("box").appendChild(newP);
</script>
```

Inserting Elements

Instead of only at the end, you can insert elements in different places.

- append() → adds at the end.
- prepend() → adds at the beginning.
- insertBefore() → add before a specific element.

```
<ul id="list">
  <li>First</li>
</ul>

<script>
  let newItem = document.createElement("li");
  newItem.textContent = "Second";

  let list = document.getElementById("list");
  list.append(newItem);  // Adds "Second" after "First"
</script>
```

Removing Elements

We can delete elements from the DOM.

Two main ways:

.remove() → directly remove the element.

```
<p id="msg">Goodbye!</p>

<script>
  let el = document.getElementById("msg");
  el.remove();  // removes the <p>
</script>
```

Removing Elements

We can delete elements from the DOM.

Two main ways:

.removeChild() → remove from parent.

```
<ul id="list">
  <li>Apple</li>
  <li id="remove-me">Banana</li>
  <li>Cherry</li>
</ul>

<script>
  let parent = document.getElementById("list");
  let child = document.getElementById("remove-me");
  parent.removeChild(child);   // Removes Banana
</script>
```

Cloning Elements

We can copy an element with cloneNode().

```
<p id="item">Clone me!</p>

<script>
  let item = document.getElementById("item");
  let copy = item.cloneNode(true);   // true = deep copy (includes
children)
  document.body.append(copy);
</script>
```