Comprehensive Guide to DOM in JavaScript

1. What is DOM?

->DOM (Document Object Model) is a programming interface for web documents.
->It represents the structure of a web page as a tree of nodes.
->JavaScript uses the DOM to interact with and manipulate the content, structure, and styles of a webpage.
->Every element in an HTML document becomes a node in this tree.

Example:

```
<!DOCTYPE html>
<html>
  <head>
    <title>DOM Example</title>
  </head>
  <body>
    <h1 id="title">Hello, DOM!</h1>
    <p class="description">Welcome to the Document Object Model.</p>
  </body>
</html>
```

DOM Tree Representation:

```
- Document
  └── html
      ├── head
      │   └── title
      └── body
          ├── h1 (id="title")
          └── p (class="description")
```

-> The entire HTML is a **document**.

-> Tags like <head>, <body>, <h1>, and <p> are **nodes**.

-> Attributes (id, class) and text content (Hello, DOM!) are also part of the DOM.

2. How Does DOM Work?

-> When a browser loads a webpage, it goes through three steps:

1. HTML Parsing: Converts HTML code into a structured DOM tree.
2. CSS Parsing: Builds a CSSOM (CSS Object Model) to apply styles.
3. JavaScript Execution: Uses the DOM API to manipulate the elements.

->Flowchart:

HTML Code → DOM Tree
CSS Code → CSSOM
DOM + CSSOM → Rendered Page

Note :**-> JavaScript can dynamically update the DOM, and changes will reflect on the browser without reloading the page.**

3. What is the DOM Tree?

The DOM represents an HTML document as a hierarchical tree.

There are 4 main types of nodes:

1. Document Node → The root (represents the entire HTML document)
2. Element Node → HTML tags (like <div>, <p>, etc.)
3. Attribute Node → Represents attributes (class, id)
4. Text Node → Actual text inside tags

Example:
<p class="text">Hello, World!</p>

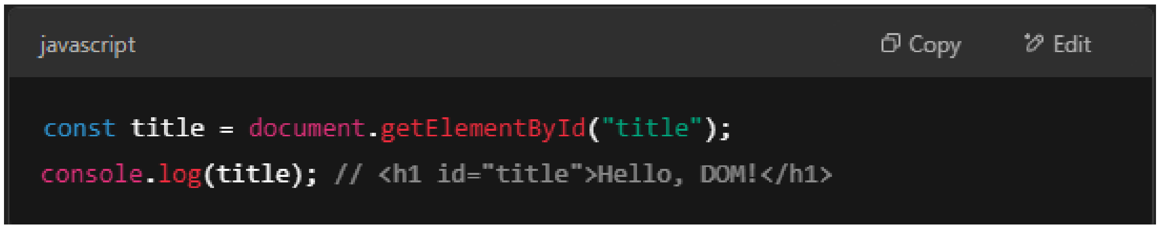
- Element Node: <p>
- Attribute Node: class="text"
- Text Node: "Hello, World!"

4. How to Select Elements from the DOM

JavaScript provides various ways to select elements from the DOM using the **document** object.

1. Select by ID (getElementById)

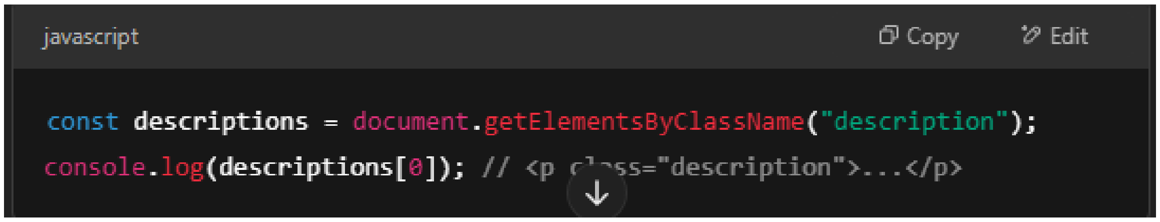
- **Returns a single element.**



```javascript
const title = document.getElementById("title");
console.log(title); // <h1 id="title">Hello, DOM!</h1>
```

2. Select by Class (getElementsByClassName)

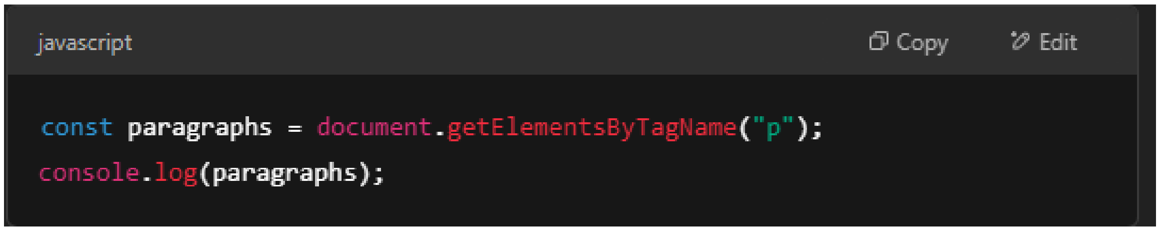
- **Returns a live HTMLCollection.**



```javascript
const descriptions = document.getElementsByClassName("description");
console.log(descriptions[0]); // <p class="description">...</p>
```

3. Select by Tag Name (getElementsByTagName)

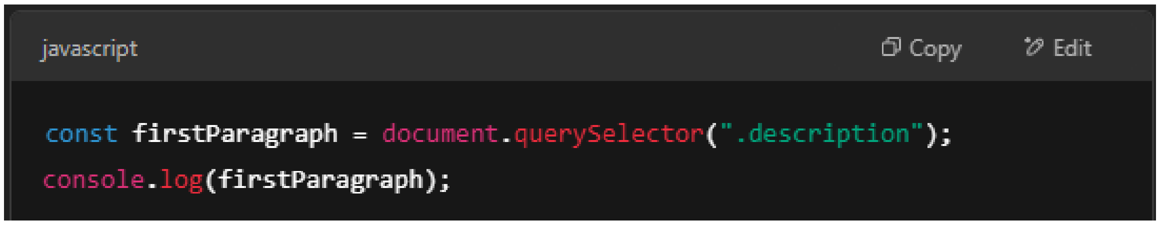
- **Selects elements using the tag name and returns an HTMLCollection.**



```javascript
const paragraphs = document.getElementsByTagName("p");
console.log(paragraphs);
```

4. Select Using querySelector()

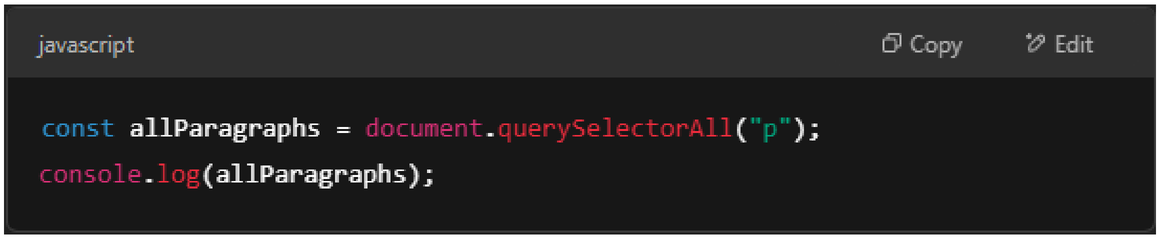
- **Returns the first matching element using CSS selectors.**



```javascript
const firstParagraph = document.querySelector(".description");
console.log(firstParagraph);
```

5. Select Using querySelectorAll()

- **Returns all matching elements as a NodeList.**



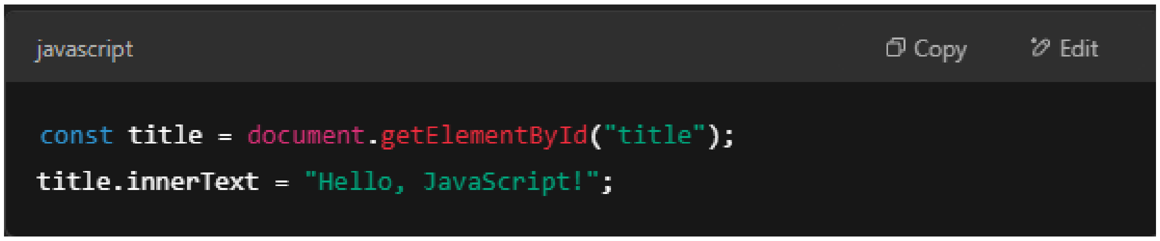
```javascript
const allParagraphs = document.querySelectorAll("p");
console.log(allParagraphs);
```

5. How to Manipulate Elements

Once you've selected elements, you can modify them using various properties and methods.

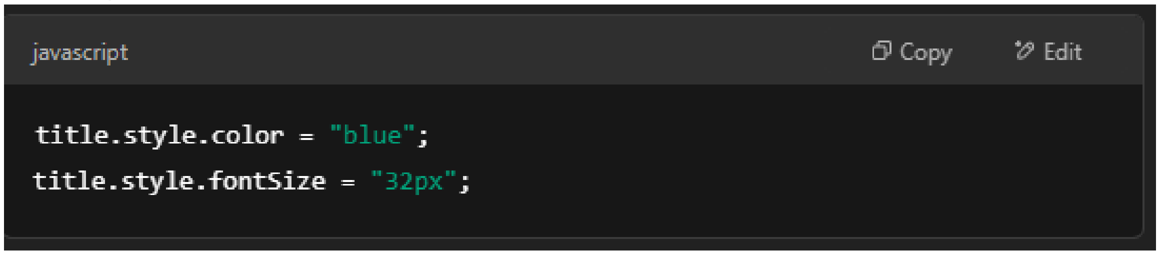
1. **Change Text**

->Use **innerText**, **textContent**, or **innerHTML**.



```javascript
const title = document.getElementById("title");
title.innerText = "Hello, JavaScript!";
```

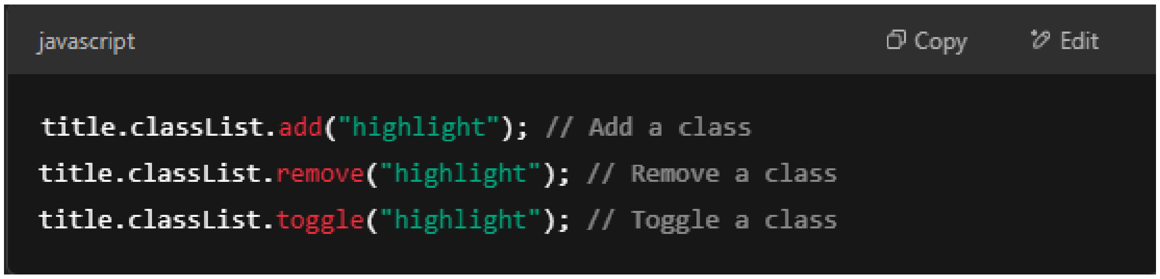
2. Change Styles



```javascript
title.style.color = "blue";
title.style.fontSize = "32px";
```

3. Add or Remove Classes

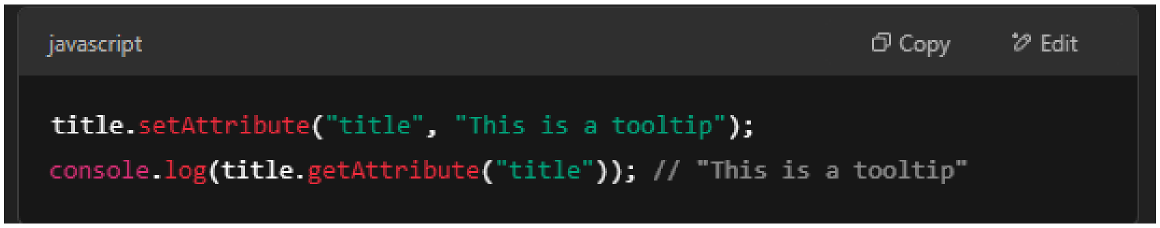
- Use classList.



```javascript
title.classList.add("highlight"); // Add a class
title.classList.remove("highlight"); // Remove a class
title.classList.toggle("highlight"); // Toggle a class
```

4. Change Attributes

Use setAttribute() and getAttribute().

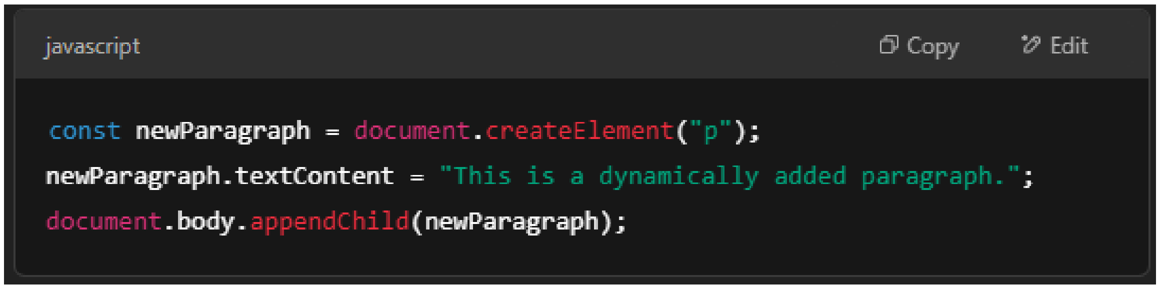


```javascript
title.setAttribute("title", "This is a tooltip");
console.log(title.getAttribute("title")); // "This is a tooltip"
```

6. Creating and Inserting Elements

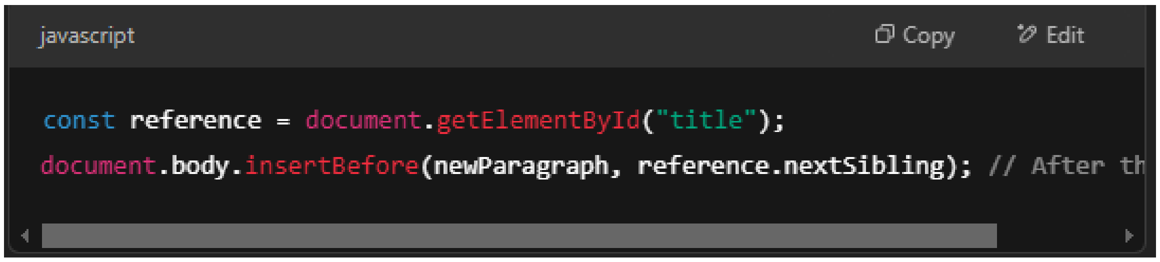
You can create new elements using **document.createElement()** and add them to the DOM using **appendChild(), insertBefore(),** or **insertAdjacentHTML()**.

Example:



```javascript
const newParagraph = document.createElement("p");
newParagraph.textContent = "This is a dynamically added paragraph.";
document.body.appendChild(newParagraph);
```

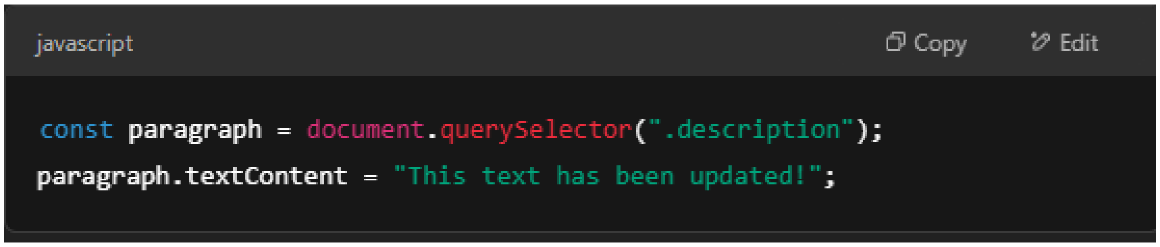
Insert Before or After



```javascript
const reference = document.getElementById("title");
document.body.insertBefore(newParagraph, reference.nextSibling); // After th
```

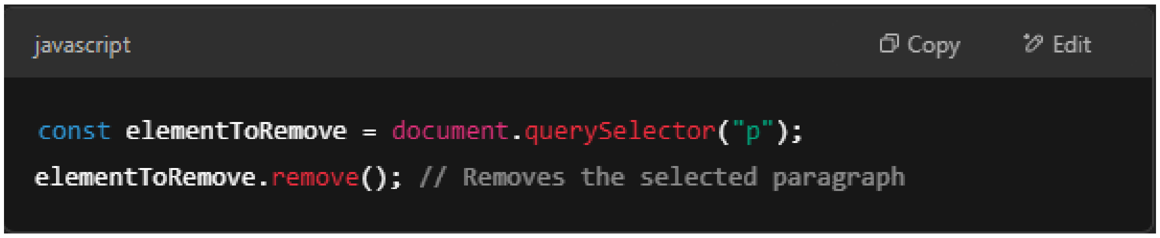
7. Editing and Removing Elements

Editing Content:



```javascript
const paragraph = document.querySelector(".description");
paragraph.textContent = "This text has been updated!";
```

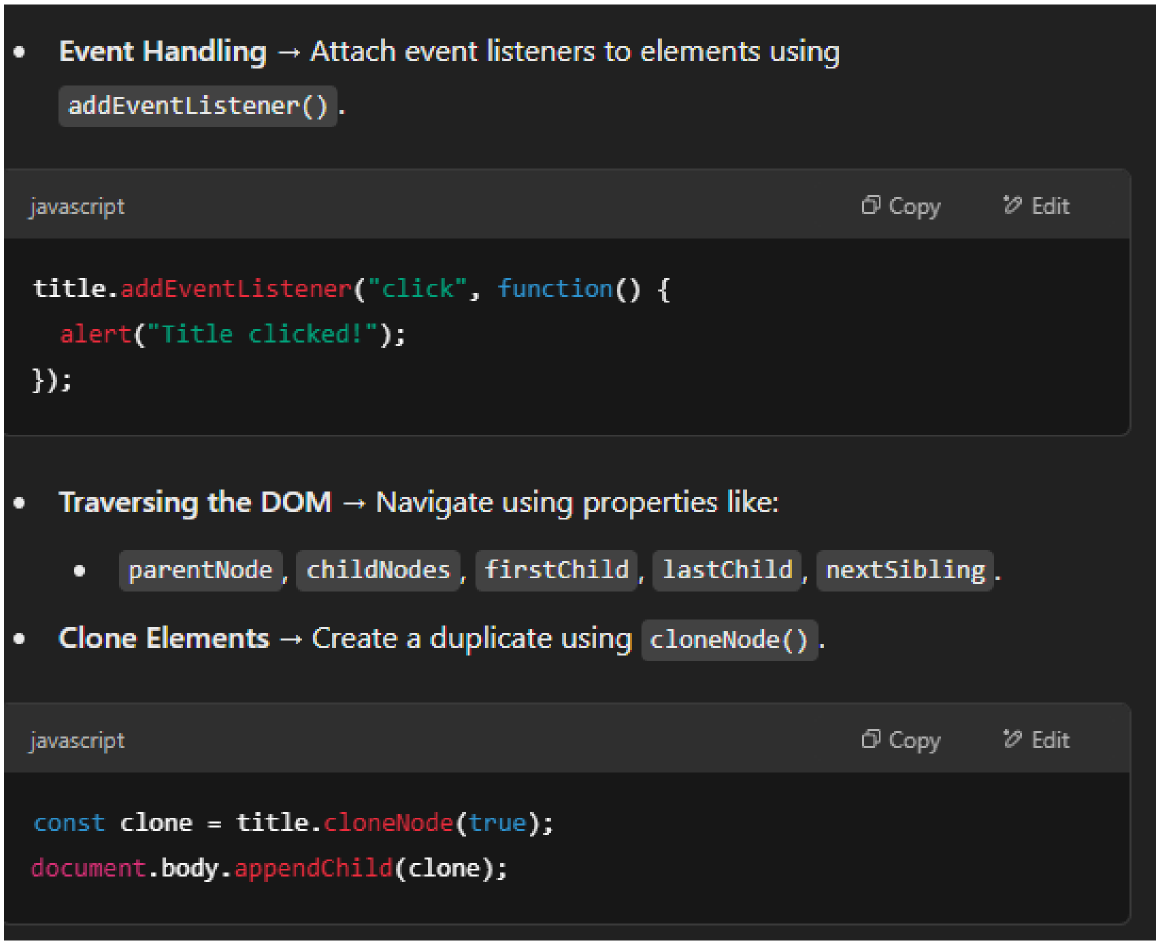
Removing Elements:



```javascript
const elementToRemove = document.querySelector("p");
elementToRemove.remove(); // Removes the selected paragraph
```

8. Additional DOM Functionalities

JavaScript offers additional functionalities like Event Handling, Traversing the DOM, and Cloning Elements.



- **Event Handling** → Attach event listeners to elements using `addEventListener()`.

```javascript
title.addEventListener("click", function() {
  alert("Title clicked!");
});
```

- **Traversing the DOM** → Navigate using properties like:
  - `parentNode`, `childNodes`, `firstChild`, `lastChild`, `nextSibling`.
- **Clone Elements** → Create a duplicate using `cloneNode()`.

```javascript
const clone = title.cloneNode(true);
document.body.appendChild(clone);
```