DOM in JS

give complete notes on dom in js

Here's a complete set of notes on the **Document Object Model (DOM) in JavaScript**:

Dom Js Notes

Document Object Model (DOM) in JavaScript

Introduction

The Document Object Model (DOM) is a programming interface that allows JavaScript to interact with and manipulate HTML and XML documents. The DOM represents a document as a tree-like structure where each node is an object that corresponds to a part of the document.

DOM Tree Structure

The DOM represents an HTML document as a tree of nodes, with different types of nodes:

1. **Document Node** - Represents the entire HTML document.
2. **Element Node** - Represents an HTML element (e.g., `<div>`, `<p>`).
3. **Attribute Node** - Represents attributes of an element.
4. **Text Node** - Represents text inside an element.
5. **Comment Node** - Represents comments in the HTML.

Example:

```html
<!DOCTYPE html>
<html>
<head>
    <title>DOM Example</title>
</head>
<body>
```

```
    <p id="demo">Hello, DOM!</p>
</body>
</html>
```

This structure can be visualized as:

```
Document
├── html
│   ├── head
│   │   ├── title
│   │   │   └── "DOM Example"
│   ├── body
│   │   ├── p (id="demo")
│   │   │   └── "Hello, DOM!"
```

Accessing DOM Elements

JavaScript provides several methods to select elements from the DOM:

1. By ID

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

2. By Class Name

```
document.getElementsByClassName("classname");
```

3. By Tag Name

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

4. By Query Selector

```
document.querySelector("#demo"); // Selects the first match

document.querySelectorAll(".classname"); // Selects all matching el
```

Manipulating DOM Elements

1. Changing Content

```
document.getElementById("demo").innerHTML = "New Content";
```

2. Changing Attributes

```
document.getElementById("demo").setAttribute("class", "new-class");
```

3. Changing Styles

```
document.getElementById("demo").style.color = "red";
```

4. Adding Elements

```
let para = document.createElement("p");
para.innerText = "New paragraph";
document.body.appendChild(para);
```

5. Removing Elements

```
let element = document.getElementById("demo");
element.parentNode.removeChild(element);
```

Event Handling in DOM

Events allow JavaScript to respond to user interactions.

1. Adding an Event Listener

```
document.getElementById("demo").addEventListener("click", function(
    alert("Paragraph clicked!");
});
```

2. Removing an Event Listener

```
function showAlert() {
    alert("Clicked!");
}
let element = document.getElementById("demo");
element.addEventListener("click", showAlert);
element.removeEventListener("click", showAlert);
```

3. Common DOM Events

- `click` - When an element is clicked.
- `mouseover` - When mouse hovers over an element.
- `mouseout` - When mouse leaves an element.
- `keydown` - When a key is pressed.
- `load` - When the page finishes loading.

DOM Traversal

You can navigate the DOM tree using:

- `parentNode` - Access parent node.
- `childNodes` - Get all child nodes.
- `firstChild` / `lastChild` - Get the first/last child node.
- `nextSibling` / `previousSibling` - Get next/previous sibling.

Example:

```
let parent = document.getElementById("demo").parentNode;
console.log(parent.nodeName);
```

Conclusion

The DOM is a powerful interface for interacting with web pages dynamically. Understanding how to select, manipulate, and respond to changes in the DOM is crucial for front-end development.

Here are the complete notes on the DOM in JavaScript. Let me know if you need any modifications or additional details! 🚀