DOM Notes in JavaScript

Basic Concepts

What is the DOM?

- The DOM represents an HTML or XML document as a tree of nodes.
- Each node represents part of the document (e.g., an element, attribute, or text).

Accessing the DOM

- Use `document` object to interact with the DOM.
- Example: Accessing the document title

  ```
  console.log(document.title); // Outputs the title of the document
  ```

Selecting Elements

- **`getElementById`**: Selects an element by its ID.

  ```
  const element = document.getElementById('myId');
  console.log(element.textContent); // Outputs the text content of the element
  ```

- **`getElementsByClassName`**: Selects elements by class name.

  ```
  const elements = document.getElementsByClassName('myClass');
  console.log(elements.length); // Outputs the number of elements with the class
  ```

- **`getElementsByTagName`**: Selects elements by tag name.

  ```
  const paragraphs = document.getElementsByTagName('p');
  console.log(paragraphs[0].textContent); // Outputs the text content of the
  first paragraph
  ```

- **`querySelector` and `querySelectorAll`**: Selects elements using CSS selectors.

  ```
  const element = document.querySelector('#myId');
  const elements = document.querySelectorAll('.myClass');
  ```

Modifying the DOM

Changing Content

- **`textContent`**: Sets or gets the text content of a node.

```
const element = document.getElementById('myId');
element.textContent = 'New Text'; // Changes the text content
```

- **`innerHTML`**: Sets or gets the HTML content of a node.

```
element.innerHTML = '<strong>New HTML</strong>'; // Changes the HTML content
```

Changing Attributes

- **`setAttribute`**: Sets the value of an attribute.

```
element.setAttribute('data-info', 'new value');
```

- **`getAttribute`**: Gets the value of an attribute.

```
console.log(element.getAttribute('data-info')); // Outputs the attribute value
```

Changing Styles

- **`style` property**: Directly modifies the inline styles of an element.

```
element.style.color = 'blue';
element.style.backgroundColor = 'yellow';
```

Creating and Removing Elements

Creating Elements

- **`createElement`**: Creates a new element.

```
const newElement = document.createElement('div');
newElement.textContent = 'Hello, World!';
```

- **`appendChild`**: Adds a new element as a child.

```
document.body.appendChild(newElement); // Adds the new element to the body
```

Removing Elements

- **`removeChild`**: Removes a child element.

```
const parent = document.getElementById('parentId');
const child = document.getElementById('childId');
parent.removeChild(child); // Removes the child element
```

- **`remove`**: Removes the element itself.

```
child.remove(); // Removes the element
```

Event Handling

Adding Event Listeners

- **`addEventListener`**: Attaches an event handler to an element.

```
const button = document.getElementById('myButton');
button.addEventListener('click', function() {
    alert('Button clicked!');
});
```

Removing Event Listeners

- **`removeEventListener`**: Detaches an event handler from an element.

```
function handleClick() {
    alert('Button clicked!');
}
button.addEventListener('click', handleClick);
button.removeEventListener('click', handleClick); // Removes the event
listener
```

Advanced Concepts

Event Delegation

- Efficiently handle events by adding a single event listener to a parent element.

```
document.body.addEventListener('click', function(event) {
    if (event.target.tagName === 'BUTTON') {
        alert('Button clicked!');
    }
});
```

Traversing the DOM

- **`parentNode`**: Accesses the parent node.

```
const parent = element.parentNode;
```

- **`childNodes`**: Accesses all child nodes.

```
const children = element.childNodes;
```

- **`nextSibling`** **and** **`previousSibling`**: Accesses sibling nodes.

```
const next = element.nextSibling;
const previous = element.previousSibling;
```

Manipulating Classes

- **`classList`**: Adds, removes, or toggles CSS classes.

```
element.classList.add('newClass');
element.classList.remove('oldClass');
element.classList.toggle('toggleClass');
```

Working with Forms

- **Accessing form elements**: Use **`document.forms`** to access form elements.

```
const form = document.forms['myForm'];
const input = form.elements['myInput'];
console.log(input.value); // Outputs the value of the input
```

Custom Data Attributes

- **`dataset`**: Accesses custom data attributes.

```
element.dataset.info = 'new value';
console.log(element.dataset.info); // Outputs the custom data attribute value
```

Mutation Observer

- **Monitoring DOM Changes**: Use **`MutationObserver`** to watch for changes in the DOM.

```
const observer = new MutationObserver(mutations => {
    mutations.forEach(mutation => {
        console.log(mutation);
    });
});

observer.observe(document.body, { childList: true, subtree: true });
```

Performance Optimization

- **Minimize Reflows and Repaints**: Batch DOM updates to reduce the number of reflows and repaints.

```
const fragment = document.createDocumentFragment();
for (let i = 0; i < 10; i++) {
    const newElement = document.createElement('div');
    newElement.textContent = `Item ${i}`;
    fragment.appendChild(newElement);
}
document.body.appendChild(fragment);
```

Best Practices

- **Minimize DOM Manipulation**: Batch DOM updates to improve performance.
- **Use Event Delegation**: Reduce the number of event listeners.
- **Avoid Inline Styles**: Use CSS classes for styling.
- **Separate Concerns**: Keep JavaScript, HTML, and CSS separate for better maintainability.

- **Debounce and Throttle**: Optimize event handling for performance-intensive tasks.

```
function debounce(func, delay) {
    let timeout;
    return function(...args) {
        clearTimeout(timeout);
        timeout = setTimeout(() => func.apply(this, args), delay);
    };
}

const debouncedHandleResize = debounce(() => {
    console.log('Window resized');
}, 200);

window.addEventListener('resize', debouncedHandleResize);
```

These notes cover the basics and advanced concepts of working with the DOM in JavaScript, providing a solid foundation for dynamic web development.