DOM Manipulation Methods

In this reading, you will learn about a Document Object Model (DOM) manipulation method known as querySelectorAll.

querySelectorAll

querySelectorAll is a method in JavaScript that selects multiple HTML elements within the DOM based on CSS-like selectors. It returns a collection (a non-live NodeList) of elements that match the specified selector. You can use it to select elements by class, ID, or tag name.

Here are examples of how to use querySelectorAll for class, ID, and tag selections with console.log and explanations of their syntax:

1. Selecting by Class:

HTML code

```
<html>
<head>
    <title>querySelectorAll Example</title>
</head>
<body>
    <p class="highlighted">This is a highlighted paragraph.</p>
    <p class="highlighted">This is another highlighted paragraph.</p>
    <p>This is a regular paragraph.</p>
</body>
</html>
```

JavaScript Code

```
const elementsByClass = document.querySelectorAll('.highlighted');
// Log the selected elements to the console
console.log(elementsByClass);
```

Output

```
NodeList [ <p.highlighted>, <p.highlighted> ]
```

Explanation:

- `document.querySelectorAll('.highlighted')` selects all elements with the class "highlighted" within the document.
- The elementsByClass collection stores the selected elements, which form a NodeList.
- `console.log(elementsByClass);` logs the selected elements to the console.
- Output Explanation: The elementsByClass NodeList contains two <p> elements with the class "highlighted." The console.log statement displays the NodeList with these two elements.

2. Selecting by ID:

HTML Code

```
<!DOCTYPE html>
<html>
<head>
    <title>querySelectorAll Example</title>
</head>
<body>
    <p id="my-paragraph">This is a paragraph with an ID.</p>
    <p>This is another paragraph.</p>
</body>
</html>
```

JavaScript code

```
// Select the element with the ID "my-paragraph" using querySelectorAll
const elementByID = document.querySelectorAll('#my-paragraph');
// Log the selected element to the console
console.log(elementByID);
```

Output

```
NodeList [ <p#my-paragraph> ]
```

Explanation:

- `document.querySelectorAll('#my-paragraph')` selects the element with the ID "my-paragraph" within the document. Even though querySelectorAll is used, it still returns a collection, but in this case, it contains only one element (if the ID is unique).
- The elementByID collection stores the selected elements, which form a NodeList.
- `console.log(elementByID);` logs the selected element to the console.
- Output Explanation: The elementByID NodeList contains the <p> element with the ID "my-paragraph." Even though it's a single element, it's still represented as a NodeList. The console.log statement displays the NodeList with this element.

3. Selecting by Tag Name:

HTML code

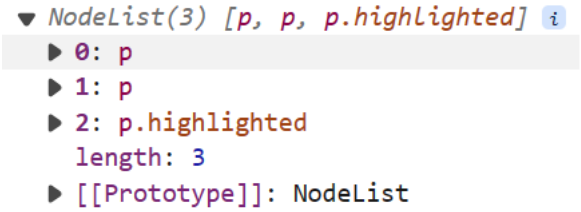
```
<!DOCTYPE html>
<html>
<head>
    <title>querySelectorAll Example</title>
</head>
<body>
    <p>This is a paragraph.</p>
    <p>This is another paragraph.</p>
    <p class="highlighted">This is a highlighted paragraph.</p>
```

```
</body>
</html>
```

JavaScript Code

```
// Select all <p> elements using querySelectorAll
const elementsByTag = document.querySelectorAll('p');
// Log the selected elements to the console
console.log(elementsByTag);
```

Output



Explanation:

- `document.querySelectorAll('p')` selects all <p> elements within the document.
- The selected elements are stored in the elementsByTag collection, which is a NodeList.
- `console.log(elementsByTag);` logs the selected elements to the console.
- Output Explanation: The elementsByTag NodeList contains all three <p> elements in the document. The console.log statement displays the NodeList with these three elements.

ClassList

The classList property is a useful feature that allows you to manipulate classes on HTML elements easily. Let's dive into an overview of the classList property and its methods.

The classList Property in JavaScript

In the DOM, the classList property is associated with an HTML element and provides a collection of methods for working with the element's classes.

Accessing classList

You can access the classList property of an element using JavaScript like this:

```
const element = document.getElementById('myElement');
const classes = element.classList;
```

Common Methods of classList

1. `add(class1, class2, ...)`

   This method adds one or more classes to the element.

   ```
   element.classList.add('newClass');
   ```

2. `remove(class1, class2, ...)`

   Removes one or more classes from the element.

   ```
   element.classList.remove('oldClass');
   ```

3. `toggle(class, force)`

   Toggles a class. If the class exists, it is removed; otherwise, it is added. If the second parameter is true, the class is added; if false, the class is removed.

   ```
   element.classList.toggle('active');
   ```

4. `contains(class)`

   Checks if a class is present on the element. Returns true if the class exists; otherwise, it is false.

   ```
   if (element.classList.contains('special')) {
     // Do something special
   }
   ```

5. `replace(oldClass, newClass)`

   Replaces a class with another class.

   ```
   element.classList.replace('oldClass', 'newClass');
   ```

6. `item(index)`

   Returns the class name at the specified index.

   ```
   const firstClass = element.classList.item(0);
   ```

7. `toString()`

   Returns a string representing the element's classes.

```
const classString = element.classList.toString();
```

Example

```
    <!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>classList Example</title>
    <style>
        .highlight {
            color: red;
            font-weight: bold;
        }
        .italic {
            font-style: italic;
        }
        .underline {
            text-decoration: underline;
        }
        .strike {
            text-decoration: line-through;
        }
    </style>
</head>
<body>

    <p id="myParagraph" class="highlight">This is a paragraph.</p>
    <button onclick="performClassListOperations()">Perform Operations</button>

    <script>
        function performClassListOperations() {
            const paragraph = document.getElementById('myParagraph');

            // Adding a class
            paragraph.classList.add('italic');

            // Removing a class
            paragraph.classList.remove('highlight');

            // Toggling a class
            paragraph.classList.toggle('underline', true);

            // Checking if a class exists
            const hasItalicClass = paragraph.classList.contains('italic');
            console.log(`Has italic class: ${hasItalicClass}`);

            // Replacing a class after a delay (for demonstration)
            setTimeout(() => {
                paragraph.classList.replace('underline', 'strike');

                // Accessing classes as a string
                const classString = paragraph.classList.toString();
                console.log(`Current classes: ${classString}`);
            }, 2000); // Delay for 2 seconds
        }
    </script>

</body>
</html>
```

Let's break it down step by step:

HTML Structure

- `<!DOCTYPE html>` Specifies the HTML version being used.
- `<html lang="en">` Declares the document language as English.
- The <head> section contains metadata like character encoding, viewport settings, and the title of the document.
- Inside the <head>, there's a <style> block defining style tag to apply internal css.

Body Content

- `<p id="myParagraph" class="highlight">This is a paragraph.</p>` This HTML paragraph (<p>) element has an ID of "myParagraph" and a class of "highlight." It's the element on which we'll perform classList operations.
- `<button onclick="performClassListOperations()">Perform Operations</button>` This button, when clicked, triggers the performClassListOperations() function.

JavaScript Section

JavaScript Function `performClassListOperations()`

This function gets executed when a button, probably named "Perform Operations," is clicked in the HTML. Here's a detailed breakdown of each step within the function:

1. Getting the Paragraph Element

```
const paragraph = document.getElementById('myParagraph');
```

- const paragraph: Declares a variable named paragraph.
- `document.getElementById('myParagraph')` Retrieves the HTML element with the ID "myParagraph" and assigns it to the paragraph variable.

2. Adding a Class

```
paragraph.classList.add('italic');
```

- `paragraph.classList.add('italic')` Adds the class "italic" to the paragraph element's class list.

3. Removing a Class

```
paragraph.classList.remove('highlight');
```

- `paragraph.classList.remove('highlight')` Removes the class "highlight" from the paragraph element's class list.

4. Toggling a Class

```
paragraph.classList.toggle('underline', true);
```

- `paragraph.classList.toggle('underline,' true)` Toggles the class "underline" on the paragraph element. In this case, it explicitly adds the class "underline" because the second parameter is true.

5. Checking if a Class Exists

```
const hasItalicClass = paragraph.classList.contains('italic');
console.log(`Has italic class: ${hasItalicClass}`);
```

- `paragraph.classList.contains('italic')` Checks if the class "italic" exists in the paragraph element's class list.
- The result (true or false) is stored in the variable hasItalicClass and logged to the console.

6. Replacing a Class with a Delay

```
setTimeout(() => {
  paragraph.classList.replace('underline', 'strike');

  // Accessing classes as a string
  const classString = paragraph.classList.toString();
  console.log(`Current classes: ${classString}`);
}, 2000); // Delay for 2 seconds
```

- `setTimeout(() => { ... }, 2000)` Delays the execution of the inner code by 2000 milliseconds (2 seconds).
- Inside the timeout function:
  - `paragraph.classList.replace('underline,' 'strike')` Replaces the class "underline" with "strike" in the paragraph element's class list.
  - `const classString = paragraph.classList.toString()` Retrieves the updated classes as a string.
  - `console.log(Current classes: ${classString})` Logs the current classes of the paragraph element to the console.

Summary

This reading demonstrates how to dynamically manipulate classes of an HTML element using JavaScript's classList property. It showcases adding, removing, toggling, checking existence, and replacing classes, offering a practical example of class manipulation within a web page.