Day 22 - Manipulating the DOM

Creating an Element

To create an element, we use a tag name as a string. We use the method document.createElement(). This method takes an HTML element tag name as a string parameter.

```
// syntax
document.createElement('tagName')
```

```
<!DOCTYPE html>
<html>

<head>
    <title>Document Object Model:30 Days Of JavaScript</title>
</head>

<body>

    <script>
        let title = document.createElement('h1')
        title.className = 'title'
        title.style.fontSize = '24px'
        title.textContent = 'Creating HTML element DOM Day 2'

        console.log(title)
    </script>
</body>

</html>
```

Creating elements

To create multiple elements, we can use a loop, which allows us to create as many HTML elements as we want. After creating an element, we can assign values to the different properties of the HTML object.

```
<!DOCTYPE html>
<html>
```

```
<head>
    <title>Document Object Model: 30 Days of JavaScript</title>
</head>

<body>

    <script>
        let title = document.createElement('h1')
        title.className = 'title'
        title.style.fontSize = '24px'
        title.textContent = 'Creating HTML elememt DOM Day 2'

        let subheading
        for(let i = 0; i < 3; i++) {
            subheading = document.createElement('h2')
            subheading.className = 'subheading'
            subheading.style.fontSize = '16px'
            subheading.textContent = i
            console.log(subheading)
        }
    </script>
</body>

</html>
```

Appending child to a parent element

To see a created element on the HTML document, we can append it to the parent as a child element. We access the HTML document body using document.body, which supports the appendChild() method.

```
<!DOCTYPE html>
<html>

<head>
    <title>Document Object Model:30 Days Of JavaScript</title>
</head>

<body>

    <script>
        // creating multiple elements and appending to parent element
        let title
        for (let i = 0; i < 3; i++) {
            title = document.createElement('h1')
            title.className = 'title'
            title.style.fontSize = '24px'
            title.textContent = i
            document.body.appendChild(title)
```

```
        }
    </script>
</body>
</html>
```

Removing a child element from a parent node

After creating HTML, we may want to remove an element or elements using the removeChild() method.

```
<!DOCTYPE html>
<html>

<head>
    <title>Document Object Model:30 Days Of JavaScript</title>
</head>

<body>
    <h1>Removing child Node</h1>
    <h2>Asabeneh Yetayeh challenges in 2020</h1>
    <ul>
        <li>30DaysOfPython Challenge Done</li>
        <li>30DaysOfJavaScript Challenge Done</li>
        <li>30DaysOfReact Challenge Coming</li>
        <li>30DaysOfFullStack Challenge Coming</li>
        <li>30DaysOfDataAnalysis Challenge Coming</li>
        <li>30DaysOfReactNative Challenge Coming</li>
        <li>30DaysOfMachineLearning Challenge Coming</li>
    </ul>

    <script>
        const ul = document.querySelector('ul')
        const lists = document.querySelectorAll('li')
        for (const list of lists) {
            ul.removeChild(list)

        }
    </script>
</body>

</html>
```

There is a better way to eliminate all inner HTML elements or children of a parent element using the *innerHTML* property.

```
<!DOCTYPE html>
<html>

<head>
    <title>Document Object Model:30 Days Of JavaScript</title>
</head>

<body>
    <h1>Removing child Node</h1>
    <h2>Asabeneh Yetayeh challenges in 2020</h1>
    <ul>
        <li>30DaysOfPython Challenge Done</li>
        <li>30DaysOfJavaScript Challenge Done</li>
        <li>30DaysOfReact Challenge Coming</li>
        <li>30DaysOfFullStack Challenge Coming</li>
        <li>30DaysOfDataAnalysis Challenge Coming</li>
        <li>30DaysOfReactNative Challenge Coming</li>
        <li>30DaysOfMachineLearning Challenge Coming</li>
    </ul>

    <script>
        const ul = document.querySelector('ul')
        ul.innerHTML = ''
    </script>
</body>

</html>
```