JavaScript after() Method

Summary: in this tutorial, you'll learn how to use the JavaScript `after()` method to insert a node after an element.

Introduction to the JavaScript after() method

The `after()` is a method of the `Element` type. The `element.after()` method allows you to insert one or more nodes after the `element` .

Here's the syntax of the `after()` method:

```
Element.after(node)
```

In this syntax, the `after()` method inserts the node after the Element in the DOM tree.

For example, suppose you have a `<h1>` element and you want to insert a `<p>` element after it, you can use the `after()` method like this:

```
h1.after(p)
```

To insert multiple nodes after an element, you pass the nodes to the `after()` method as follows:

```
Element.after(node1, node2, ... nodeN)
```

The `after()` method also accepts one or more strings. In this case, the `after()` method treats the strings as `Text` nodes:

```
Element.after(str1, str2, ... strN)
```

The `after()` method returns `undefined`. If a node cannot be inserted, it'll throw a `HierarchyRequestError` exception.

JavaScript after() examples

Let's take some examples of using the JavaScript `after()` method.

1) Using JavaScript after() to insert a node after an element

The following example uses the `after()` method to insert a paragraph after a `<h1>` element:

```
<!DOCTYPE html>
<html lang="en">

    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>JavaScript DOM - after()</title>
    </head>

    <body>
        <h1>JavaScript DOM - after()</h1>

        <script>
            const h1 = document.querySelector('h1');

            // create a new paragraph element
            const p = document.createElement('p');
            p.innerText = 'This is JavaScript DOM after() method demo';

            // insert the paragraph after the heading
            h1.after(p);
        </script>
    </body>

</html>
```

How it works.

First, get the heading element using the `querySelector()` method:

```
const h1 = document.querySelector('h1');
```

Second, create a new paragraph element and set its `innerText` :

```
const p = document.createElement('p');
p.innerText = 'This is JavaScript DOM after() method demo';
```

Third, insert the `<p>` element after the `<h1>` element:

```
h1.after(p);
```

2) Using JavaScript after() to insert multiple nodes after an element

The following example uses the `after()` method to insert multiple nodes after an element:

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>JavaScript DOM - after()</title>
    </head>
    <body>
        <ul>
            <li>Angular</li>
            <li>Vue</li>
        </ul>
        <script>
            const list = document.querySelector('ul');

            const libs = ['React', 'Meteor', 'Polymer'];
            const items = libs.map((lib) => {
                const item = document.createElement('li');
                item.innerText = lib;
                return item;
```

```
            });

            list.lastChild.after(...items);

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

How it works:

First, select the ul element using the `querySelector()` method:

```
const list = document.querySelector('ul');
```

Second, define an array of strings. In practice, you may get it from an API call.

```
const libs = ['React', 'Meteor', 'Polymer'];
```

Third, transform the array of strings into an array of li elements using the `map()` method:

```
const items = libs.map((lib) => {
  const item = document.createElement('li');
  item.innerText = lib;
  return item;
});
```

Finally, insert the list item elements after the last child of the ul element:

```
list.lastChild.after(...items);
```

Note that the `...items` uses the spread operator to spread out the element of the `items` array.

The ul element will look like the following:

```
<ul>
    <li>Angular</li>
```

```
    <li>Vue</li>
    <li>React</li>
    <li>Meteor</li>
    <li>Polymer</li>
</ul>
```

The last three items (React, Meteor, and Polymer) were inserted after the item `Vue`, which was the last child of the `<ul>` element.

3) Using JavaScript after() to insert strings

When you use strings in the `after()` method, it will treat them as `Text` nodes. For example:

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>JavaScript DOM - after()</title>
        <style>
            button {
                padding: 0.75em 1em;
                background-color: #F7DF1E;
                color: #000;
                cursor: pointer;
                border-radius: 50vw;
            }
        </style>
    </head>
    <body>
        <button>Donate Here</button>
        <script>
            const button = document.querySelector('button');
            button.firstChild.after(' ❤️');

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

Summary

- Use the `element.after()` method to insert one or more nodes after the element.

Quiz