Adding Adjacent Elements

In this lesson, we will add adjacent elements in the document tree.

WE'LL COVER THE FOLLOWING

To extend the document tree, it is not enough to add child nodes to HTML elements. You also need to add sibling elements and text to existing HTML tags. Unfortunately, the HTML DOM standard does not contain any method that can be used to add sibling nodes directly before or after a certain document tree node.

A few browser vendors extend the standard DOM API with extra operations.

For example, Google Chrome, Opera, Safari, and Internet Explorer provide three additional methods, `insertAdjacentText()`, `insertAdjacentElement()`, and `insertAdjacentHTML()` which can be used for this task.

Firefox does not implement these operations. In this section, you will learn how to use them.

Listing 6-9 demonstrates using `insertAdjacentText()`. This code wraps every link in the page with square brackets, adding a `"["` character before, and another `"]"` character after the link with `insertAdjacentText()`.

Listing 6-9: Using `insertAdjacentText()`

```
<!DOCTYPE html>
<html>
<head>
  <title>Add new elements</title>
  <base href="http://www.w3schools.com" />
</head>
<body onload="decorateLinks3()">
  <p>
```

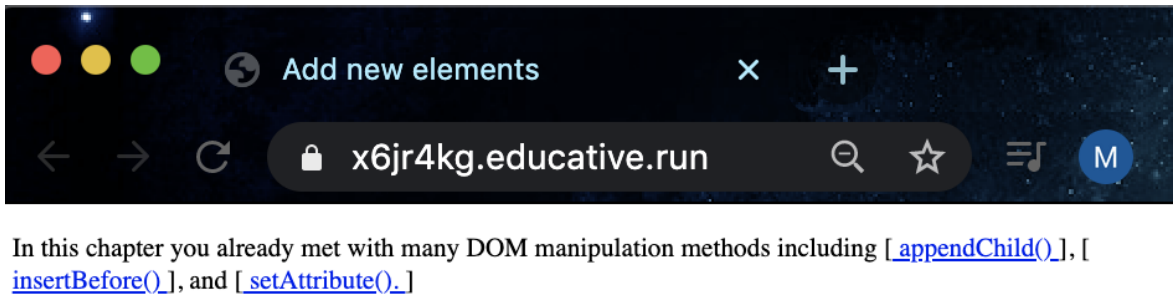
```
    In this chapter you already met with many DOM
    manipulation methods including
    <a href="/jsref/met_node_appendchild.asp">
      appendChild()
    </a>,
    <a href="/jsref/met_node_insertbefore.asp">
      insertBefore()
    </a>, and
    <a href="/jsref/met_element_setattribute.asp">
      setAttribute().
    </a>
  </p>
  <script>
    function decorateLinks() {
      for (i = 0; i < document.links.length; i++) {
        var link = document.links[i];
        link.insertAdjacentText('beforeBegin', '[');
        link.insertAdjacentText('afterEnd', ']');
      }
    }

    function decorateLinks2() {
      for (i = 0; i < document.links.length; i++) {
        var link = document.links[i];
        var openSpan = document.createElement('span');
        openSpan.textContent = '[';
        var closeSpan = document.createElement('span');
        closeSpan.textContent = ']';
        link.insertAdjacentElement('beforeBegin', openSpan);
        link.insertAdjacentElement('afterEnd', closeSpan);
      }
    }

    function decorateLinks3() {
      for (i = 0; i < document.links.length; i++) {
        var link = document.links[i];
        link.insertAdjacentHTML('beforeBegin',
          '<span>[</span>');
        link.insertAdjacentHTML('afterEnd',
          '<span>]</span>');
      }
    }

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

As the `onload` attribute of `<body>` shows, the `decorateLinks()` method is invoked as soon as the page has been loaded. It iterates through all links declared in the document (using the `document.links` property), and calls the `insertAdjacentText()` method twice on the link element.

The first argument of this method is a string that specifies where to insert the text. The listing uses the `“beforeBegin”` and `“afterEnd”` values that instruct the method to add the text before the opening tag and after the closing tag of the corresponding node.

Besides these values, you can use `“beforeEnd”` and `“afterBegin”` to add the text before the closing tag or after the opening tag.

When you display the page, you can immediately observe the result of this short code snippet, as shown below:



The code in Listing 6-9 wraps links into square brackets

If you want to apply your own style settings on square brackets, you’d better wrap them into a `<span>` tag. You can immediately try this approach with the `decorateLinks2()` method:

```
function decorateLinks2() {
      for (i = 0; i < document.links.length; i++) {
        var link = document.links[i];
        var openSpan = document.createElement('span');
        openSpan.textContent = '[';
        var closeSpan = document.createElement('span');
        closeSpan.textContent = ']';
        link.insertAdjacentElement('beforeBegin', openSpan);
        link.insertAdjacentElement('afterEnd', closeSpan);
      }
    }
```

As you can see in the highlighted part of this code, it creates an `openSpan` and a `closeSpan` object to represent two `<span>` elements and uses the textContent to set them to the appropriate square bracket.

Instead of utilizing the `insertAdjacentText()` method, this code leverages `insertAdjacentElement()`. The first method argument specifies the insert position, just like `insertAdjacentText()`, and in the second argument you pass the object representing the element to insert.

To try `decorateLinks2()`, do not forget to change the `onload` attribute of `<body>`:

```
<body onload="decorateLinks2()">
```

You can implement adding `<span>` tags easier with the help of `insertAdjacentHTML()`, as shown in this code snippet:

```
function decorateLinks3() {
     for (i = 0; i < document.links.length; i++) {
       var link = document.links[i];
       link.insertAdjacentHTML('beforeBegin',
         '<span>[</span>');
       link.insertAdjacentHTML('afterEnd',
         '<span>]</span>');
     }
   }
```

The `insertAdjacentHTML()` method has the same semantics as the other two `insertAdjacent...()` methods, but here you have to pass an HTML markup string in the second argument.

To try `decorateLinks3()`, do not forget to alter the `onload` attribute of `<body>`:

```
<body onload="decorateLinks3()">
```

In the next lesson we will tackle removing and replacing elements in a document tree.