Changing the Document Tree

In this lesson we will learn how to change the content of elements in a document through simple coding examples.

WE'LL COVER THE FOLLOWING

It is great that you can traverse through the document tree and access its content. However, it is just a single step toward creating interactive pages.

The more exciting thing is to change the structure and content of the tree, and it opens totally new horizons to vivify your web pages.

The DOM contains about a dozen useful operations that allow you to change the document tree. Among the others, you can insert new elements, remove existing ones, and replace the content and attributes of existing elements.

Changing the content of elements

You already used the textContent, `innerHTML` properties, and the `getAttributes()` function to query the content of the document tree. The very same properties can be used to set the content of elements and `setAttribute()`, the pair of `getAttribute()`, is the one you can use to set or change the value of an element's attribute.

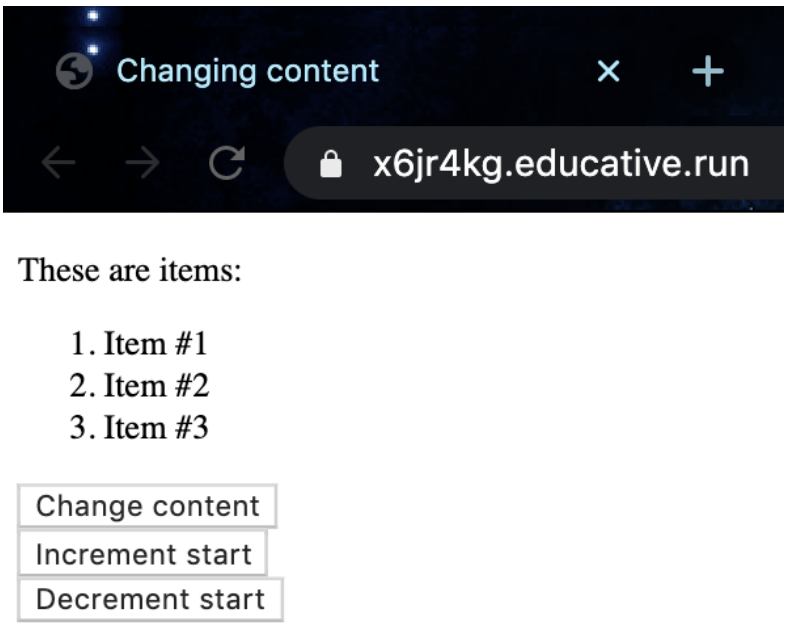
Listing 6-7 shows an example with three short JavaScript methods, each demonstrating a way to change the document tree content.

Listing 6-7: Changing document tree content

```
<!DOCTYPE html>
<html>
<head>
  <title>Changing content</title>
</head>
<body>
  <p id="para">These are items:</p>
  <ol id="list" start="1">
    <li id="item1">Item #1</li>
    <li id="item2">Item #2</li>
    <li id="item3">Item #3</li>
  </ol>
  <button onclick="changeContent()">
    Change content
  </button>
  <br />
  <button onclick="incrementStart()">
    Increment start
  </button>
  <br />
  <button onclick="decrementStart()">
    Decrement start
  </button>
  <script>
    function changeContent() {
      var para = document.getElementById('para');
      var item = document.getElementById('item1');

      para.innerHTML = 'These are '
        + '<strong>new </strong> items:';

      item.textContent = "First item";
      item = item.nextElementSibling;
      item.textContent = "Second item";
      item = item.nextElementSibling;
      item.textContent = "Third item";
    }

    function incrementStart() {
      var list = document.getElementById('list');
      var start = parseInt(list.getAttribute("start"));
      start++;
      list.setAttribute("start", start);
    }

    function decrementStart() {
      var list = document.getElementById('list');
      list.start--;
    }
  </script>
</body>
</html>
```

When the page is displayed, it shows the original content (image below). The three buttons in the page are assigned to the `changeContent()`, `incrementStart()` and `decrementStart()` methods, respectively.

`incrementStart()`, and `decrementStart()` methods, respectively.



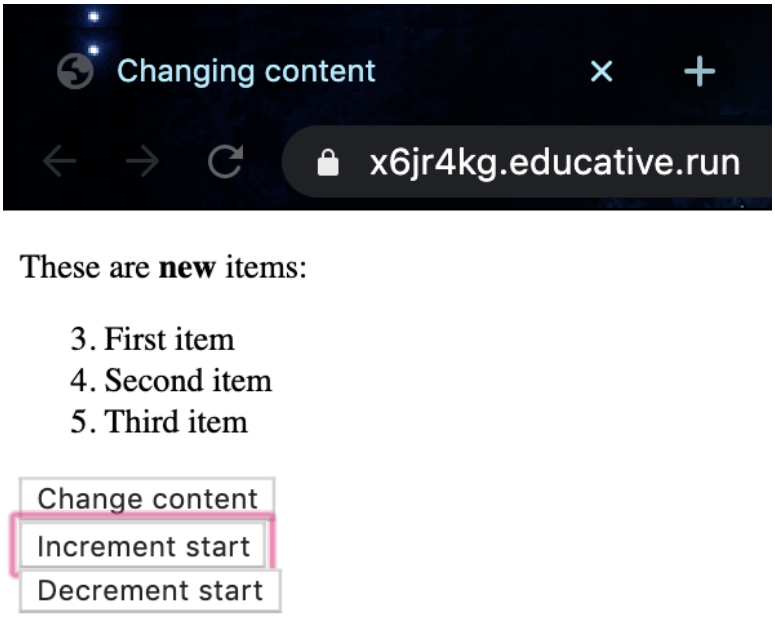
The original content of the page defined in Listing 6-7

The `changeContent()` method uses the innerHTML property to change the text assigned to the `<p>` tag at the top of the screen. It uses the `textContent` property to change the textual content of each of the list items and uses the `nextElementSibling` property to navigate from one list item to the subsequent one.

The `incrementStart()` method leverages `getAttribute()` to obtain the current content of the `<ol>` tag's start attribute. It is a string, so the method uses the `parseInt()` method to convert it to an integer number, then increments its values and stores this value back to start with `setAttribute()`.

Instead of reading, incrementing, and writing back the start attribute's value, you can simply change the start property of the list object as it represents the start attribute of `<ol>`, as implemented by the `decrementStart()` method.

The effect of these methods is shown in the image below:



The original content of the document tree has been changed

In the *next lesson*, we will learn how to add new child elements to the document.