Accessing Element Content

In this lesson, we'll learn three simple ways to access node content.
Let's begin!

WE'LL COVER THE FOLLOWING

To implement useful functionality, it is not enough to navigate to a document tree node. You often need to query the content of a specific node, or a set of nodes.

Three simple ways to access node content

The DOM provides three simple ways to access node content.

Method 1

The first and most obvious way is to use the HTML element and attribute navigation methods to access child elements and attributes. Sooner or later you reach a node that does not have any child.

Method 2

The second way is to use the `textContent` property of the node you have grasped. It retrieves the concatenated text within the element, excluding all other nodes.

Method 3

The third and most frequently used way is to obtain the value of the

The third and most frequently used way is to obtain the value of the `innerHTML` property, which retrieves the textual representation of the HTML markup embedded within the element.

Listing 6-5 demonstrates using the `textContent` and `innerHTML` properties.

Listing 6-5: Using the `textContent` and `innerHTML` properties.

```
<!DOCTYPE html>
<html>
<head>
  <title>DOM Tree</title>
  <script>
    function logContent() {
      console.log('paragraph content:');
      var para = document
        .getElementById('para');
      console.log('text:' +
        '"' + para.textContent + '"');
      console.log('HTML:' + para.innerHTML);

      console.log('ol content:');
      var ol = document
        .getElementsByTagName('ol')[0];
      console.log('text:' +
        '"' + ol.textContent + '"');
      console.log('HTML:' + ol.innerHTML);
    }
  </script>
</head>
<body onload="logContent()">
  <p id="para">
    This is an
    <strong>ordered</strong> list
  </p>
  <ol start="1">
    <li id="item1">Item #1</li>
    <li id="item2">Item #2</li>
    <li id="item3">Item #3</li>
  </ol>
</body>
</html>
```

This script's code queries the text and HTML content of the `<p>` tag and the `<ol>` tag, and produces this console output:

console

```
paragraph content:
text:"
    This is an
```

```
    ordered list
  "
HTML:

    This is an
    <strong>ordered</strong> list

ol content:
text:"
    Item #1
    Item #2
    Item #3
  "
HTML:
    <li id="item1">Item #1</li>
    <li id="item2">Item #2</li>
    <li id="item3">Item #3</li>
```

This output clearly shows that the `innerHTML` property retrieves the full HTML markup of the elements while `textContent` omits all HTML elements and their attributes. The output highlights the HTML tags. If you omit them, you get exactly the output produced by `textContent`.

You can observe another important thing. The `<p>` element has a child element, `<strong>`:

```
<p id="para">
  This is an
  <strong>ordered</strong> list
</p>
```

The value of `textContent` recursively obtains text nodes within all child nodes, and this is why the output includes `“ordered”` which is not a child node of `<p>`, but the child of `<strong>` (so, actually the `“grandchild”` node of `<p>`).

In the *next lesson*, we will get a brief reference on navigation.