JavaScript innerHTML

Summary: in this tutorial, you will learn how to use the JavaScript `innerHTML` property of an element to get or set an HTML markup contained in the element.

The `innerHTML` is a property of the `Element` that allows you to get or set the HTML markup contained within the element:

```
element.innerHTML = 'new content';
element.innerHTML;
```

Reading the innerHTML property of an element

To get the HTML markup contained within an element, you use the following syntax:

```
let content = element.innerHTML;
```

When you read the `innerHTML` of an element, the web browser has to serialize the HTML fragment of the element's descendants.

1) Simple JavaScript innerHTML example

Suppose that you have the following markup:

```
<ul id="menu">
    <li>Home</li>
    <li>Services</li>
</ul>
```

The following example uses the `innerHTML` property to get the content of the `<ul>` element:

```
let menu = document.getElementById('menu');
console.log(menu.innerHTML);
```

How it works:

- First, select the `<ul>` element by its id (`menu`) using the `getElementById()` method.
- Then, get the HTML content of the `<ul>` element using the `innerHTML` .

Output:

```
<li>Home</li>
<li>Services</li>
```

2) Examining the current HTML source

The `innerHTML` property returns the current HTML source of the document, including all changes that have been made since the page was loaded.

See the following example:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>JavaScript innerHTML</title>
</head>
<body>
    <ul id="menu">
        <li>Home</li>
        <li>Services</li>
    </ul>
    <script>
        let menu = document.getElementById('menu');

        // create new li element
        let li = document.createElement('li');
        li.textContent = 'About Us';
        // add it to the ul element
        menu.appendChild(li);

        console.log(menu.innerHTML);
    </script>
</body>
</html>
```

Output:

```
<li>Home</li>
<li>Services</li>
<li>About Us</li>
```

How it works.

- First, get the `<ul>` element with the id `menu` using the `getElementById()` method.
- Second, create a new `<li>` element and add it to the `<ul>` element using the `createElement()` and `appendChild()` methods.
- Third, get the HTML of the `<ul>` element using the innerHTML property of the `<ul>` element. The contents of the `<ul>` element includes the initial content and the dynamic content created dynamically by JavaScript.

Setting the innerHTML property of an element

To set the value of `innerHTML` property, you use this syntax:

```
element.innerHTML = newHTML;
```

The setting will replace the existing content of an element with the new content.

For example, you can remove the entire contents of the document by clearing the contents of the `document.body` element:

```
document.body.innerHTML = '';
```

⚠️ Security Risk

HTML5 specifies that a `<script>` tag inserted with innerHTML should not execute.

Suppose you have the following `index.html` document:

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

    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>JS innerHTML example</title>
    </head>

    <body>
        <div id="main"></div>
        <script src="app.js"></script>
    </body>

</html>
```

And the `app.js` file looks like this:

```
const scriptHTML = `<script>alert("Alert from innerHTML");</script>`;
const main = document.getElementById('main');

main.innerHTML = scriptHTML;
```

In this example, the `alert()` inside the `<script>` tag will not execute.

However, if you change the source code of the `app.js` to the following:

```
const main = document.getElementById('main');

const externalHTML = `<img src='1' onerror='alert("Error loading image")'>`;
```

```
// shows the alert
main.innerHTML = externalHTML;
```

In this example, the image with the src 1 will not be loaded successfully. Therefore, the on error will execute that runs the alert().

Instead of having a simple alert(), hackers may include malicious code, and the user who opens the webpage will be vulnerable.

Therefore, you should not set the contents that you have no control over the `innerHTML` or you will face a potential security risk.

If you want to insert plain text into the document, you can use the `textContent` property instead of the `innerHTML`. The `textContent` will not be parsed as the HTML but as the raw text.

Summary

- Use `innerHTML` property of an element to get or set HTML contained within the element.
- The `innerHTML` property returns the current HTML source of the element, including any change that has been made since the page was loaded.
- Do not use `innerHTML` to set new content that you have no control over to avoid a security risk.

Quiz