Session 22

Mohamed Emary

June 10, 2024

1 innerHTML and innerText

1.1 innerHTML

- `innerHTML` returns the HTML content of an element.
- When assigning a value with HTML tags to `innerHTML`, the browser will render the HTML tags as HTML elements.
- When printing the value of `innerHTML` of an element that contains HTML tags, the browser will show the HTML tags in the output.

Example:

HTML:

```
<p id="example">My <strong>example</strong> paragraph</p>
```

JavaScript:

```
var example = document.getElementById('example');
console.log(example.innerHTML); // My <strong>example</strong> paragraph

example.innerHTML = 'My <strong>new</strong> paragraph';
console.log(example.innerHTML); // My <strong>new</strong> paragraph
```

1.2 innerText

- `innerText` returns the text content of an element.
- When assigning a value with HTML tags to `innerText`, the browser will render the HTML tags as plain text.
- When printing the value of `innerText` of an element that contains HTML tags, the browser will show the HTML tags in the output.

Example:

HTML:

```
<p id="example">My <strong>example</strong> paragraph</p>
```

JavaScript:

```
var example = document.getElementById('example');
console.log(example.innerText); // My example paragraph

example.innerText = 'My <strong>new</strong> paragraph';
console.log(example.innerText); // My <strong>new</strong> paragraph
```

Here the `<strong>new</strong>` appear in the web page as it is because `innerText` does not render HTML tags.

2 Creating Elements

To create an element, you can use the `document.createElement()` method. This method creates a new element with the specified tag name.

Example:

```
var newElement = document.createElement('div');
```

To set the attributes of the new element, you can use the `setAttribute()` method or the `.` notation.

Example:

```
// Using the setAttribute() method
newElement.setAttribute('id', 'new-element');
newElement.setAttribute('class', 'new-class');

// Or using the . notation
newElement.id = 'new-element';
newElement.className = 'new-class';
```

2.1 Appending & Prepending Elements (Child)

To append an element inside another element in the DOM, you can use the `append()` method, and to prepend an element, you can use the `prepend()` method.

Example:

HTML:

```
<div id="parent" style="background-color: gold">
  <p>First paragraph</p>
  <p>Second paragraph</p>
</div>
```

JavaScript:

```
var parent = document.getElementById('parent');
var newElement = document.createElement('p');
newElement.innerText = 'New paragraph';

```

```
// Append the new element inside the parent element
parent.append(newElement);
```

Now the result will look like this:

Using `append`

Figure 1: Using `append`

2.2 Add Element Before or After Another (Sibling)

To add an element before or after another element in the DOM, you can use the `before()` and `after()` methods.

Example:

HTML:

```
<div id="parent" style="background-color: gold">
  <p>First paragraph</p>
  <p>Second paragraph</p>
</div>
```

JavaScript:

```
var parent = document.getElementById('parent');
var newElement = document.createElement('p');
newElement.innerText = 'New paragraph';

// Add the new element after the second paragraph
parent.after(newElement);
```

Now the result will look like this:

Using `after`

Figure 2: Using `after`

Note

You can only send elements as arguments to the `append()`, `prepend()`, `before()`, and `after()` methods. If you send HTML tag or text, it will be treated as a string and not as an element.

2.3 Traversing the DOM

Traversing the DOM which is a way to move around the DOM tree and select elements based on their relationship to other elements.

Some useful properties and methods for traversing the DOM are:

1. `parentElement`: returns the parent **element** of an element.
2. `parentNode`: returns the parent **node** of an element.
3. `firstElementChild`: returns the first child **element** of an element.

4. `lastElementChild`: returns the last child **element** of an element.
5. `children`: returns an **HTML collection** of an element's child elements.
6. `childNodes`: returns a **NodeList** of an element's child nodes.
7. `nextElementSibling`: returns the next sibling **element** of an element.
8. `previousElementSibling`: returns the previous sibling **element** of an element.
9. `nextSibling`: returns the next sibling **node** of an element.
10. `previousSibling`: returns the previous sibling **node** of an element.

Example:

HTML:

```html
<div id="parent" style="background-color: gold">
  <p id="p1">First paragraph</p>
  <p>Second paragraph</p>
</div>
```

JavaScript:

```javascript
var parent = document.getElementById('parent');
var p1 = document.getElementById('p1');

// Get the parent element of the first paragraph
var parentElement = p1.parentElement;
console.log(parentElement.id); // parent

// Get the parent node of the first paragraph
var parentNode = p1.parentNode;
console.log(parentNode.id); // parent

// Get the first child of the parent element
var firstChild = parent.firstElementChild;
console.log(firstChild.innerText); // First paragraph

// Get the last child of the parent element
var lastChild = parent.lastElementChild;
console.log(lastChild.innerText); // Second paragraph

// Get all the child elements of the parent element
var children = parent.children;
console.log(children.length); // 2
console.log(children[1]); // <p>Second paragraph</p>

// Get all the child nodes of the parent element
var childNodes = parent.childNodes;
console.log(childNodes.length); // 3
console.log(childNodes[1]); // #text

// Get the next sibling element of the first paragraph
var nextSibling = p1.nextElementSibling;
```

```
32 console.log(nextSibling.innerText); // Second paragraph
33
34 // Get the previous sibling element of the second paragraph
35 var previousSibling = lastChild.previousElementSibling;
36 console.log(previousSibling.innerText); // First paragraph
37
38 // Get the next sibling node of the first paragraph
39 var nextNode = p1.nextSibling;
40 console.log(nextNode); // #text
41
42 // Get the previous sibling node of the second paragraph
43 var previousNode = lastChild.previousSibling;
44 console.log(previousNode); // #text
```