Lecture 5.2

Manipulating The DOM

Topics

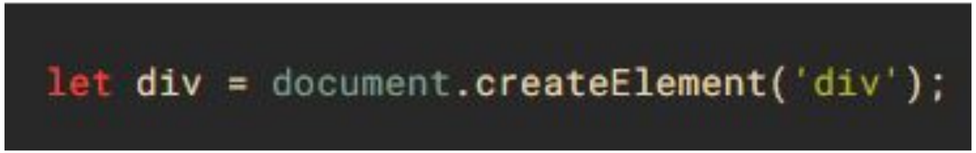
- Creating Elements
- Text Content
- Inner HTML
- Get & Set HTML Elements
- Replace & Remove Elements

Creating Elements

createElement Method

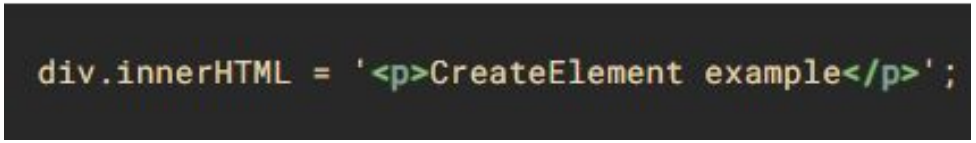
In an HTML document, the **document.createElement()** method creates the HTML element specified by tagName, or an HTMLUnknownElement if tagName isn't recognized.

The following example uses the document.createElement() to create a new <div> element:



```
let div = document.createElement('div');
```

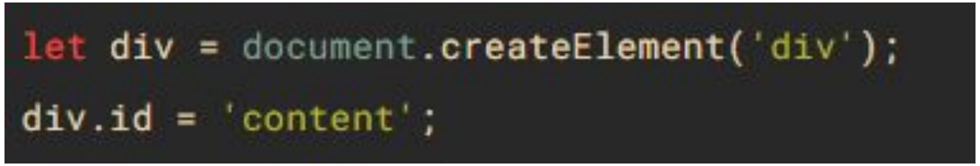
And add an HTML snippet to the div:



```
div.innerHTML = '<p>CreateElement example</p>';
```

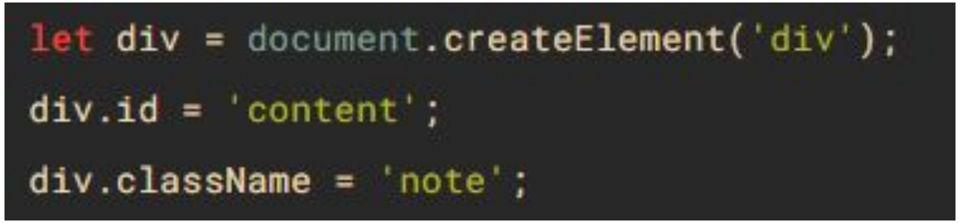
Adding id and class to the new div

If you want to add an id to a div, you set the id attribute of the element to a value, like this:



```
let div = document.createElement('div');
div.id = 'content';
```

The following example set the CSS class of a new div note:

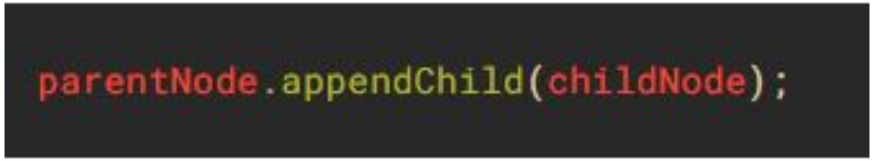


```
let div = document.createElement('div');
div.id = 'content';
div.className = 'note';
```

Append Child Elements

appendChild Method

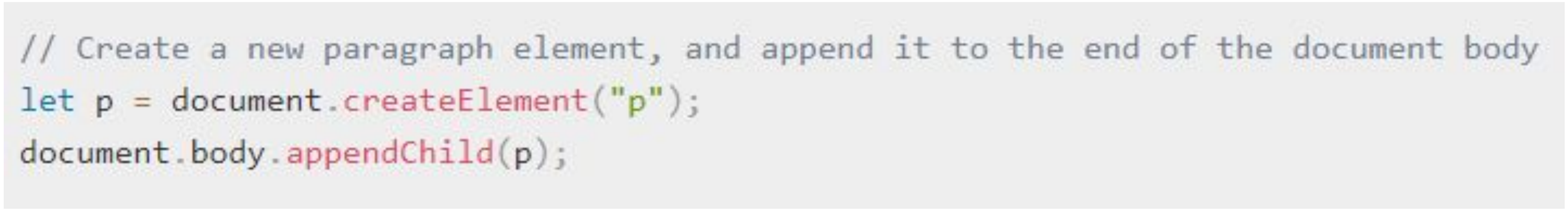
The appendChild() is a method of the Node interface. The appendChild() method allows you to add a node to the end of the list of child nodes of a specified parent node.



```
parentNode.appendChild(childNode);
```

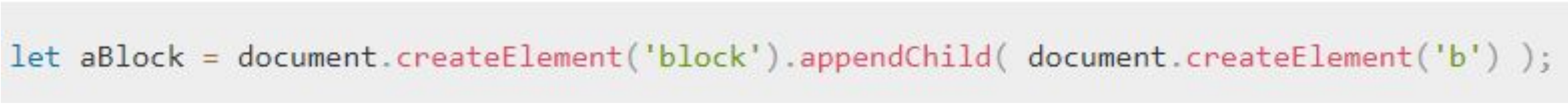
If the childNode is a reference to an existing node in the document, the appendChild() method moves the childNode from its current position to the new position.

appendChild Example

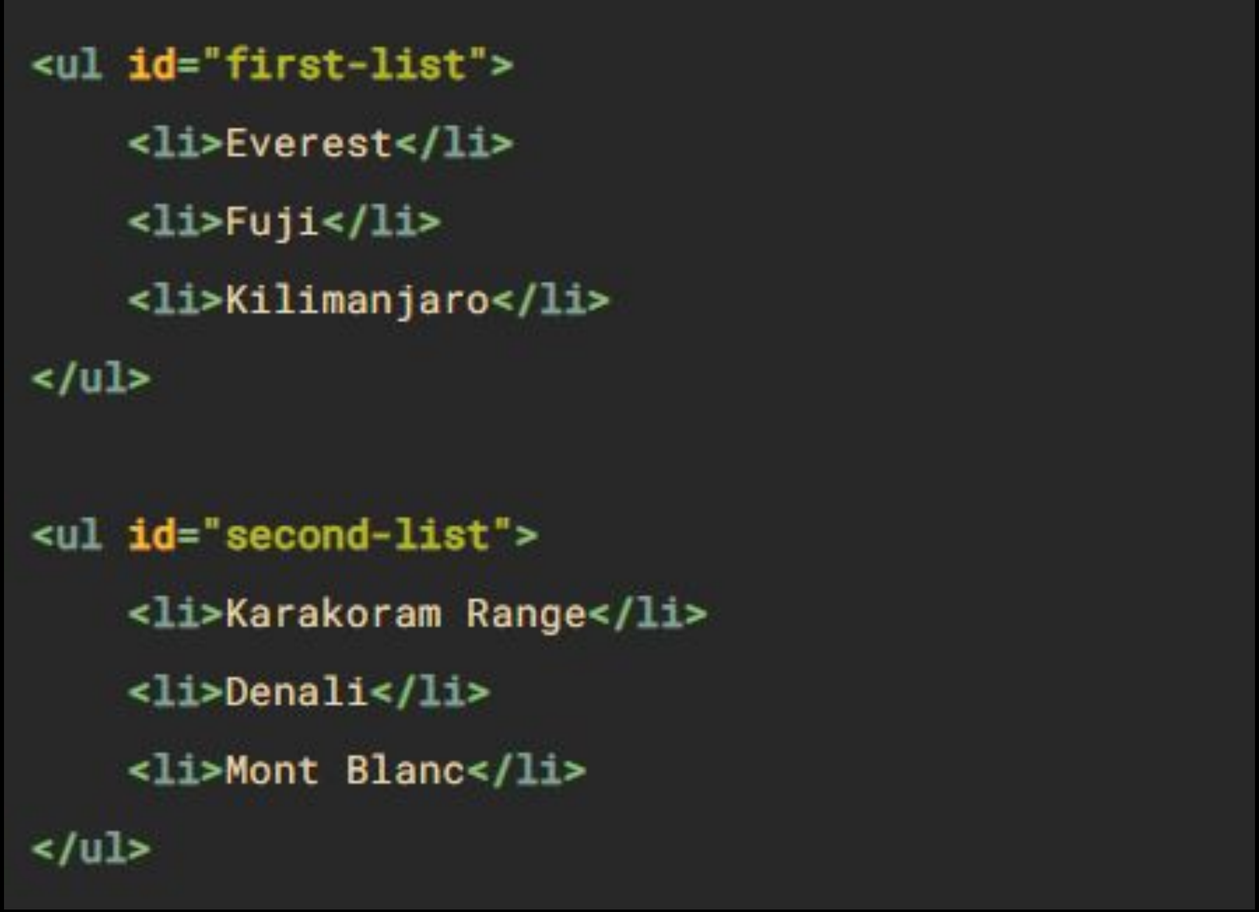


```
// Create a new paragraph element, and append it to the end of the document body
let p = document.createElement("p");
document.body.appendChild(p);
```

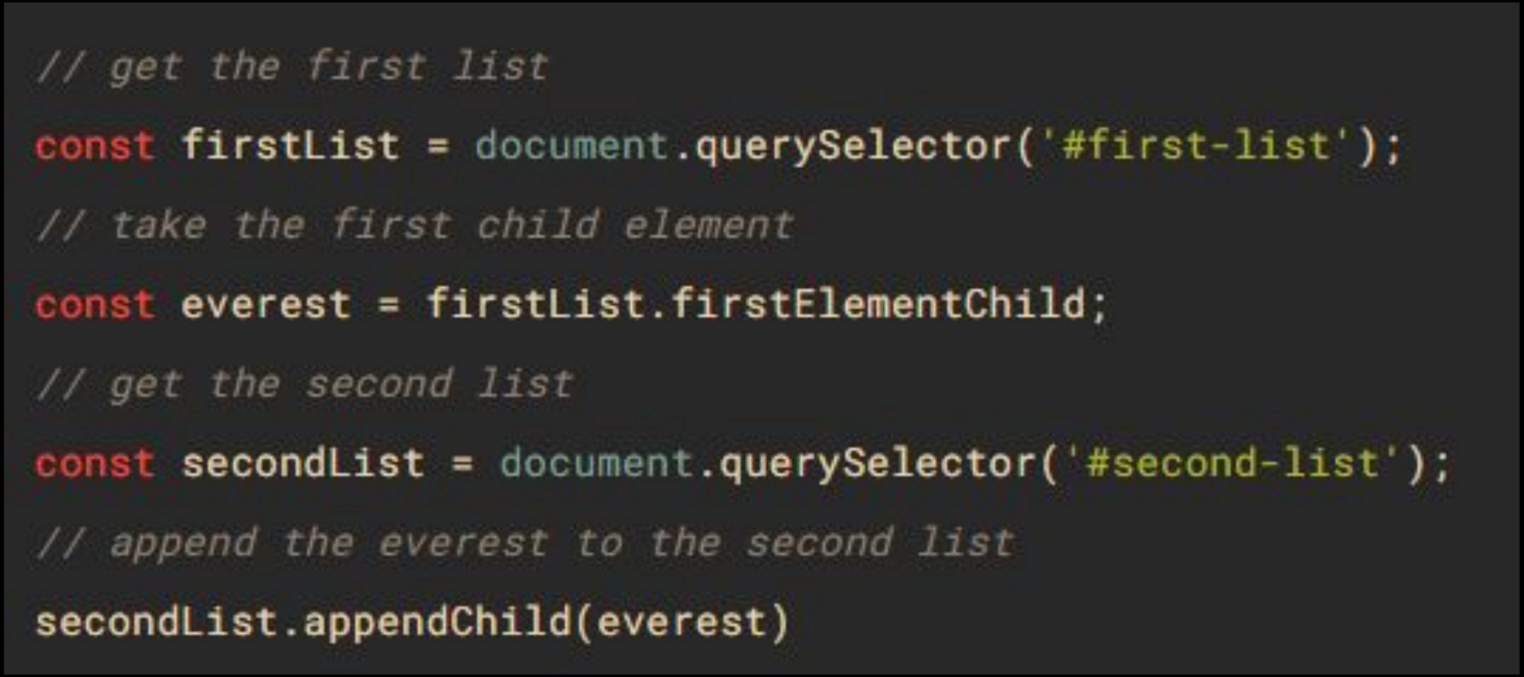
Chaining may not work as expected, due to appendChild() returning the child element:



```
let aBlock = document.createElement('block').appendChild( document.createElement('b') );
```

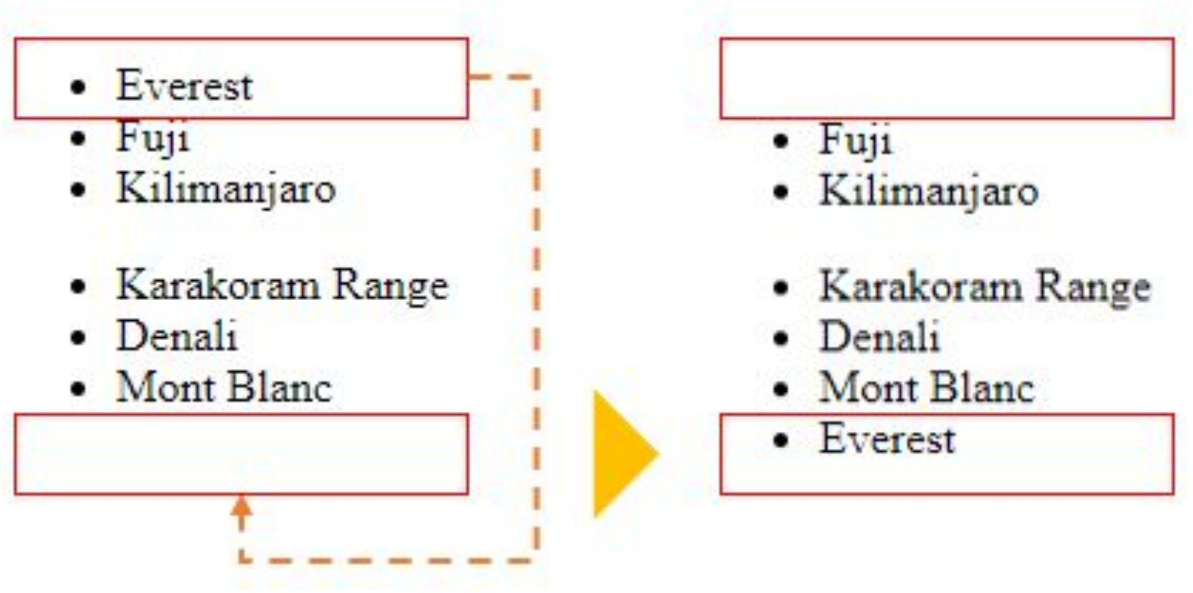

```html
<ul id="first-list">
    <li>Everest</li>
    <li>Fuji</li>
    <li>Kilimanjaro</li>
</ul>

<ul id="second-list">
    <li>Karakoram Range</li>
    <li>Denali</li>
    <li>Mont Blanc</li>
</ul>
```

Moving a node within document example



```
// get the first list
const firstList = document.querySelector('#first-list');
// take the first child element
const everest = firstList.firstElementChild;
// get the second list
const secondList = document.querySelector('#second-list');
// append the everest to the second list
secondList.appendChild(everest)
```

Use the appendChild() to move the first child element from the first list to the second list:



Use the appendChild() to move the first child element from the first list to the second list:

Text Content

Node Text Content

The **textContent** property of the Node interface represents the text content of the node and its descendants.

```html
<div id="divA">This is <span>some</span> text!</div>
```

you can use textContent to get the element's text content:

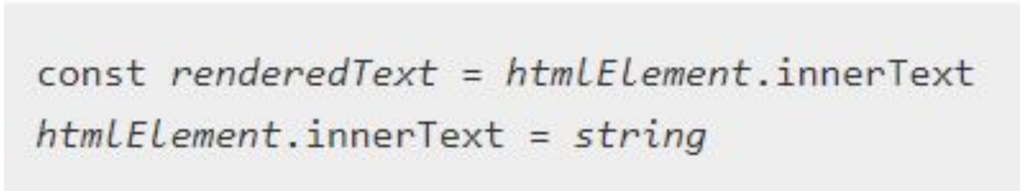
```javascript
let text = document.getElementById('divA').textContent;
// The text variable is now: 'This is some text!'
```

or set the element's text content:

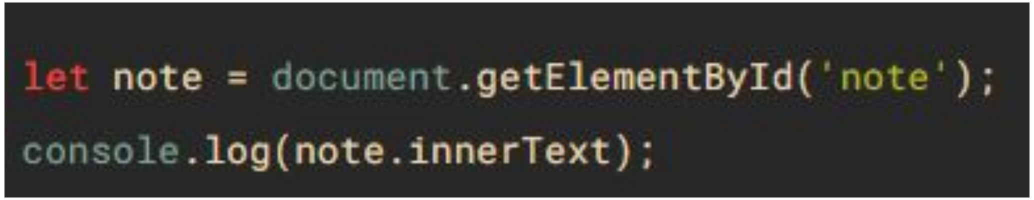
```javascript
document.getElementById('divA').textContent = 'This text is different!';
```

HTMLElement InnerText

The **innerText** property of the HTMLElement interface represents the "rendered" text content of a node and its descendants.



```
const renderedText = htmlElement.innerText
htmlElement.innerText = string
```

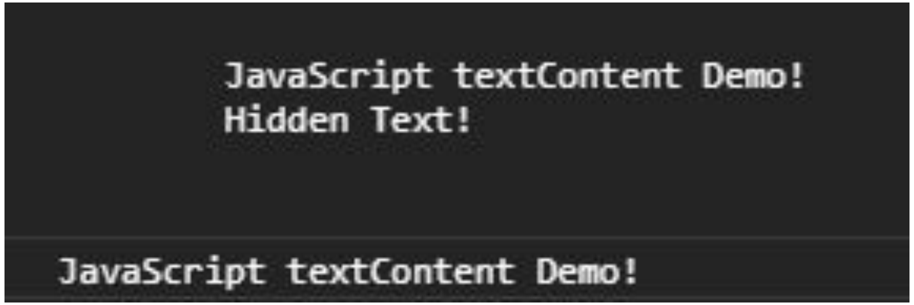


```
let note = document.getElementById('note');
console.log(note.innerText);
```

innerText vs textContent

Note: innerText is easily confused with Node.textContent, but there are important differences between the two. Basically, innerText is aware of the rendered appearance of text, while textContent is not.

The innerText returns the human-readable text that takes CSS into account.



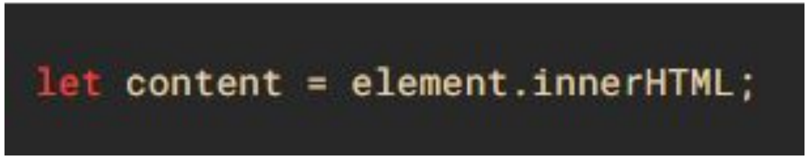
```
        JavaScript textContent Demo!
        Hidden Text!


  JavaScript textContent Demo!
```

Get & Set HTML Element

Element innerHTML

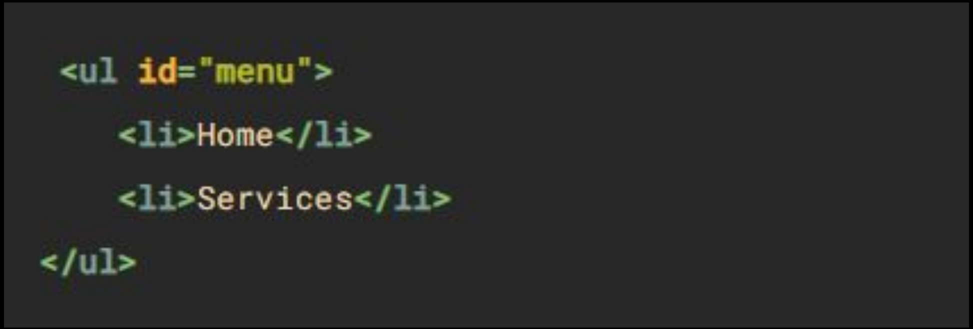
The Element property **innerHTML** gets or sets the HTML or XML markup contained within the element

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



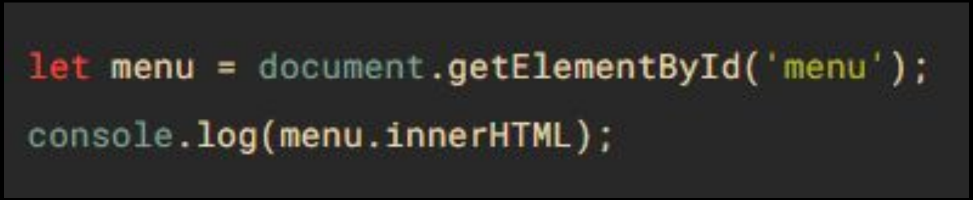
```
let content = element.innerHTML;
```

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



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

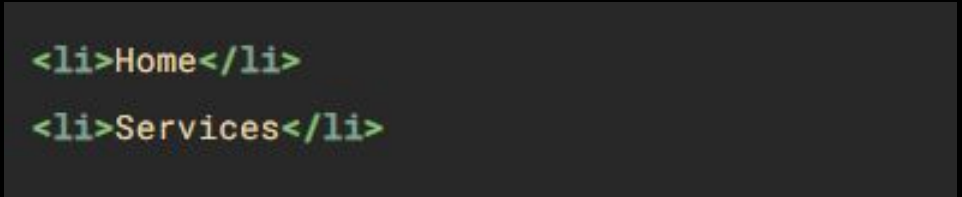
The following example uses the innerHTML property to get the content of the <ul> element:



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

How it works:

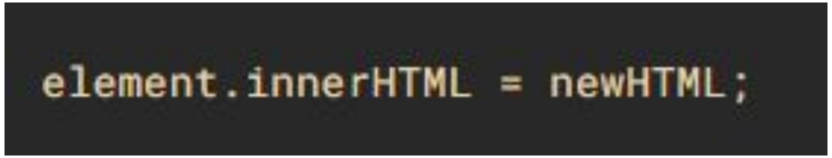
First, select the <ul> element by its id (menu) using the getElementById() method.
Then, get the HTML content of the <ul> element using the innerHTML.



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

Setting the innerHTML property of an Element

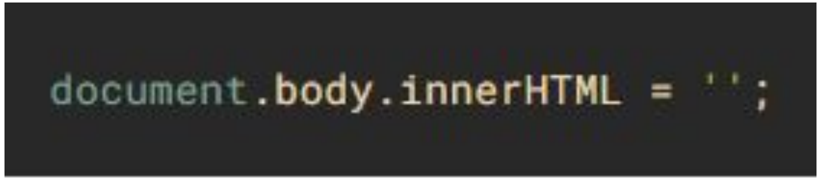
To set the value of innerHTML property, you use this syntax:



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

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

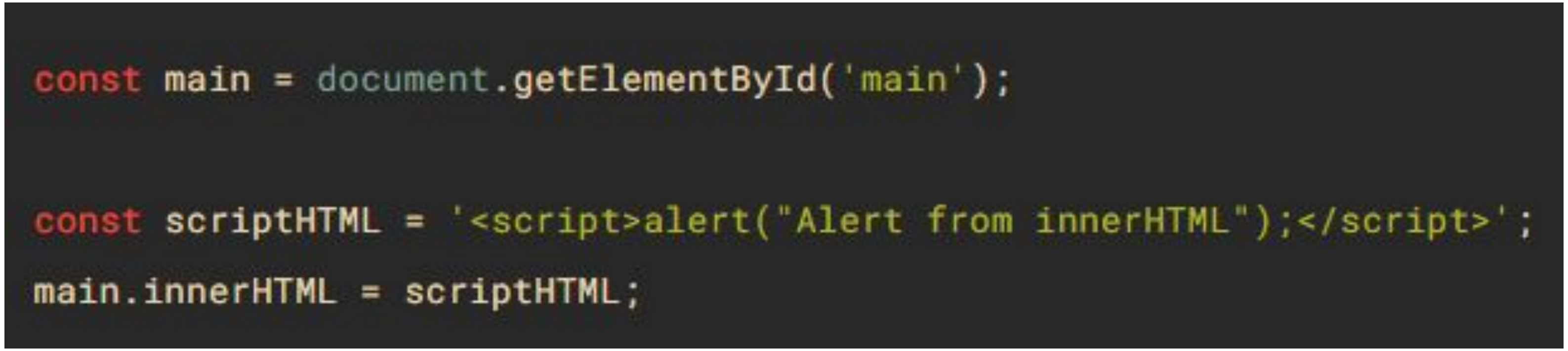
For example, you can remove the entire contents of the document by clearing contents of the document.body element:



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

Security Risk

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



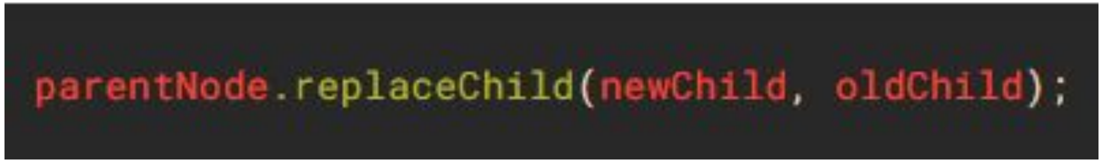
```
const main = document.getElementById('main');

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

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

Replace & Remove Element

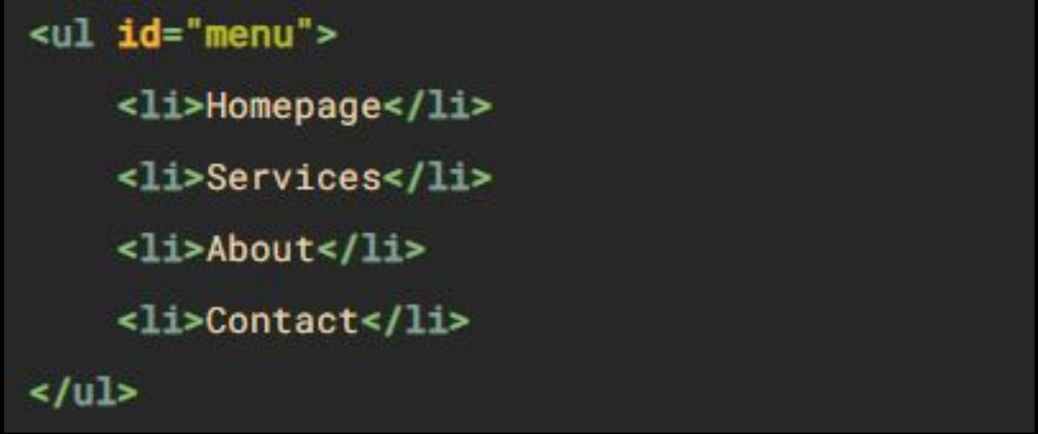
Node Replace Child

The **Node.replaceChild()** method replaces a child node within the given (parent) node.



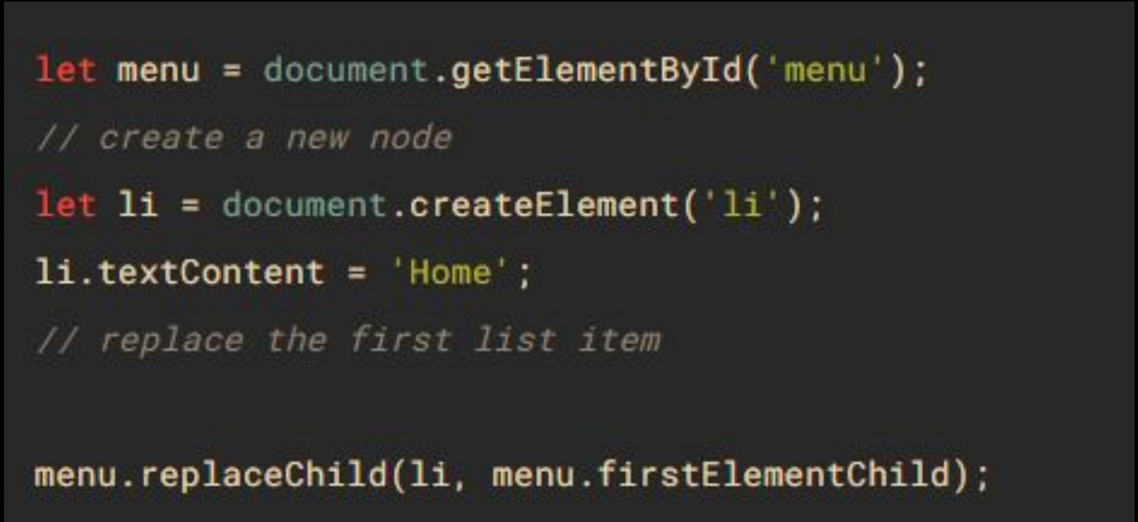
```
parentNode.replaceChild(newChild, oldChild);
```

In this method, the newChild is the new node to replace the oldChild node which is the old child node to be replaced.

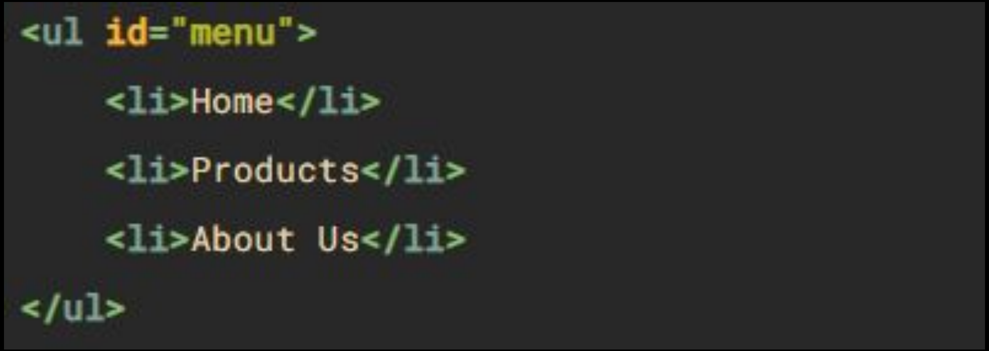


```
<ul id="menu">
    <li>Homepage</li>
    <li>Services</li>
    <li>About</li>
    <li>Contact</li>
</ul>
```

The following example creates a new list item element and replaces the first list item element in the menu by the new one:

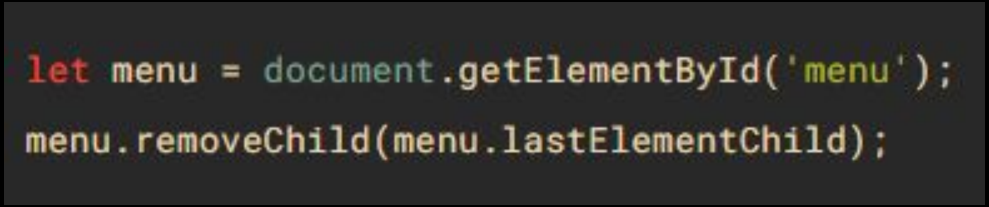


```
let menu = document.getElementById('menu');
// create a new node
let li = document.createElement('li');
li.textContent = 'Home';
// replace the first list item

menu.replaceChild(li, menu.firstElementChild);
```



```html
<ul id="menu">
    <li>Home</li>
    <li>Products</li>
    <li>About Us</li>
</ul>
```

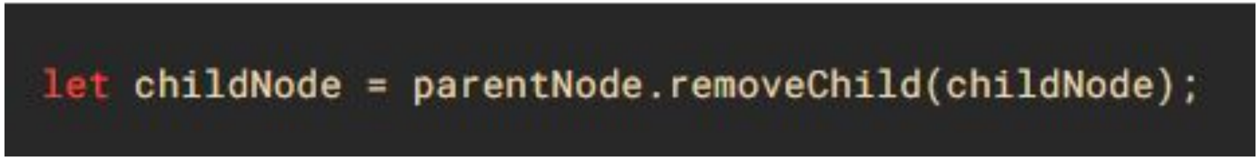
The following example uses the removeChild() to remove the last list item::



```javascript
let menu = document.getElementById('menu');
menu.removeChild(menu.lastElementChild);
```

Node Remove Child

TThe **Node.removeChild()** method removes a child node from the DOM and returns the removed node.



```
let childNode = parentNode.removeChild(childNode);
```

The childNode is the child node of the parentNode that you want to remove. If the childNode is not the child node of the parentNode, the method throws an exception.