

# Modify an Existing Element

Let's see how to use JavaScript to modify a web page once it's been loaded by the browser! You can thus make your content more dynamic and interactive.

## WE'LL COVER THE FOLLOWING ^

- Example Page
- HTML Content
- Text Content
- Attributes
- Classes

The DOM traversal properties studied in the previous chapter can also be used to update elements in the page.

## Example Page #

The examples in the next paragraphs use the HTML code below.

Output

HTML

```
<html>
<head>
</head>
<body>
<h3 class="beginning">Some languages</h3>
<div id="content">
  <ul id="languages">
    <li id="cpp">C++</li>
    <li id="java">Java</li>
    <li id="csharp">C#</li>
    <li id="php">PHP</li>
  </ul>
</div>
</body>
```

</html>



## HTML Content #

The `innerHTML` property can be used to change the content of an element within the DOM. For example, you can add a new language to our list with the code below. We'll access the `<ul>` tag identified by `"languages"` and then add an entry to the end of the list via an operator (`+=`) and an `<li>`.

Output

JavaScript

HTML

```
// Modifying an HTML element: adding an <li>
document.getElementById("languages").innerHTML += '<li id="c">C</li>';
```



The `innerHTML` property is often used to “empty” content. Try the following example:

Output

JavaScript

HTML

```
// Delete the HTML content of the list, replacing it with nothing
document.getElementById("languages").innerHTML = "";
```



Before moving on, remove the above line from your JavaScript program. Otherwise, you'll have no content!

When using `innerHTML`, you put HTML content into strings. To keep your code readable and avoid mistakes, you should only use `innerHTML` to make small content changes. You'll discover more versatile solutions below.

## Text Content #

Use the `textContent` property to modify the text content of a DOM element. Here is how to complete the title displayed by our page.

Output

JavaScript

HTML

```
document.getElementById("languages").innerHTML += '<li id="c">C</li>';

// Modify the title's text content
document.querySelector("h3").textContent += " for programming";
```



## Attributes #

The `setAttribute()` method sets the value of an attribute of an element. You pass the name and value of the attribute as parameters.

Output

JavaScript

HTML

```
// Define the id attribute of the first title
document.querySelector("h2").setAttribute("id", "title")
```

```
document.querySelector("h3").setAttribute("id", "title");
```



As you saw in the previous chapter, some attributes exist as properties and can be directly updated.

Output

JavaScript

HTML

```
// Define the id attribute of the first title
document.querySelector("h3").id = "title";
```



## Classes #

You can use the `classList` property to add or remove classes from a DOM element!

Output

JavaScript

HTML

```
const titleElement = document.querySelector("h3"); // Grab the first h3
titleElement.classList.remove("beginning");       // Remove the class "beginning"
titleElement.classList.add("title");              // Add a class called "title"
console.log(titleElement);
```



