Styling Elements

You can also add style properties to existing elements on a web page. Let's find out how to style the elements using JavaScript!

WE'LL COVER THE FOLLOWING

JavaScript not only lets you interact with your web page structure, but it also lets you change the style of elements. It's time to learn how. Here is the example HTML content used in the next paragraphs.

HTML

```
<html>
 <head>
 </head>
 <body>

        <p>First</p>
        <p style="color: green;">Second</p>
        <p id="para">Third</p>

 </body>
</html>
```

And here is the associated CSS *stylesheet*. The rules in a stylesheet determine the appearance of elements on a page. Here, the one element we're adjusting via CSS is the element with the para ID. Its text will be blue and in italics.

Output

Output

HTML

CSS (SCSS)

```
#para {
    font-style: italic;
    color: blue;
}
```

Style Property

DOM elements are equipped with a property called `style`, which returns an object representing the HTML element's `style` attribute. This object's properties match up to its CSS properties. By defining these properties with JavaScript, you're actually modifying the element's style. The code below selects the page's first paragraph and modifies its text color and margins.

Output

JavaScript

HTML

CSS (SCSS)

```
const paragraphElement = document.querySelector("p");
paragraphElement.style.color = "red";
paragraphElement.style.margin = "50px";
```

Compound CSS Properties

Some CSS properties have compound names, meaning they're composed of two words (like background-color). To interact with these properties via JavaScript, you have to ditch the hyphen and capitalize the first letter of following words. This example modifies the same paragraph element's fontfamily and background-color properties.

Output

JavaScript

HTML

CSS (SCSS)

```
// ...
paragraphElement.style.fontFamily = "Arial";
paragraphElement.style.backgroundColor = "black";
```

This naming convention, already encountered in previous chapters, is called *camelCase*. You can see CSS properties and their JavaScript properties on the Mozilla Developer Network.

Limits of the `style` Property

Let’s try to display the text color of each of our example paragraphs.

Output

JavaScript

HTML

CSS (SCSS)

```
const paragraphElements = document.getElementsByTagName("p");
console.log(paragraphElements[0].style.color); // "red"
console.log(paragraphElements[1].style.color); // "green"
console.log(paragraphElements[2].style.color); // Show an empty string
```

Console

Clear

Why is the color of the third paragraph (blue) not showing? Because the `style` property used in this code only represents the `style` attribute of the element. Using this property, you cannot access style declarations defined elsewhere, for example in a CSS stylesheet. This explains why the third paragraph's style, defined externally, is not shown here.

Access Element Styles

A better solution for accessing element styles is to use a function called `getComputedStyle()`. This function takes a DOM node as a parameter and returns an object that represents the element's style. You can then see the different CSS properties of the object. The following example will show the style properties of the third paragraph:

Output

JavaScript

HTML

CSS (SCSS)

```
const paragraphStyle = getComputedStyle(document.getElementById("para"));
console.log(paragraphStyle.fontStyle); // "italic"
console.log(paragraphStyle.color);     // color blue in RGB values
```

Console

Clear

The blue color is represented as 3 color values: red, green, and blue

(RGB). For each of these primary colors, values will always be between or equal to 0 and 255.