Why CSS-in-JS?

Learn what CSS-in-JS is and why one might use it over CSS in a stylesheet.

Introduction

This article will explore CSS-in-JS as a concept and evaluate the pros and cons of using CSS-in-JS. This article will not prescribe any specific library or syntax for using CSS-in-JS, as examples of both will be provided in the tutorial later in this course.

CSS-in-CSS

By now, we should know the traditional way of styling a web page by using a stylesheet in CSS. In CSS, selectors are used to select the elements we want to style. Inside the selector, rules will be established by assigning values to different properties. This might look something like this:

```
p {
    color: pink;
    background: blue;
}
```

CSS-in-JS

In contrast to traditional CSS stylesheets, CSS-in-JS is a technique that uses JavaScript to create, add, and manage styles. Styles that are created using CSS-in-JS are written in a Javascript file and using Javascript Syntax.

It’s also important to note that knowing CSS-in-JS is not a replacement for understanding CSS in a stylesheet. CSS-in-JS relies on preexisting knowledge of CSS. To use CSS-in-JS, we must first understand how CSS styles are applied to DOM elements, how styles are inherited, what the different properties of different elements are, etc.

There are many libraries that enable us to implement CSS-in-JS. For some of these libraries, when the JavaScript styles are parsed, CSS is generated and attached to the DOM. Here’s an example of a styled component implementation of the CSS above:

```
import styled from 'styled-components';

const pText = styled.p`
  color: pink;
  background: blue;
```

```
`;

<pText>This is my text</pText>
```

These styled components can be defined in the same file or in another file. Since they're React components, they can be exported or used in JSX.

Why should I use CSS-in-JS?

With CSS-in-JS, the build system will allow us as developers to not be concerned with certain CSS-specific practices. Some of these include:

- We get to use JavaScript syntax to write our styling, so we don't need to be consistently switching syntaxes.
- Automatic inlining of critical-path CSS
- No class names and IDs collisions, every class name and ID is guarenteed to be unique
- Refactoring CSS that's in CSS-in-JS is easier and safer since our styles are now represented as an abstract syntax tree. In normal CSS, it's hard to be sure that changing a selector will not have unintended changes.
- Automatic vendor prefixing
- Finally, it reduces the amount of code and the file size as the compiled CSS only includes necessary declarations.

In the right context, CSS-in-JS can allow us to make styling our components an easy and concise process.

Why shouldn't I use CSS-in-JS?

While CSS-in-JS can be very useful, it is not ideal for every situation. It's usage is also somewhat controversial to some web developers. Here are some notable cons of CSS-in-JS:

- Many people find CSS easier to understand as one place in which all of our styling occurs in a (somewhat) easy-to-read format.
- CSS-in-JS styles are less easily shared with other web apps.
- Depending on the implementation, using CSS-in-JS will add a runtime performance cost because Javascript has to be downloaded, parsed, and executed before styling/layout can happen.

All of this is to say that using CSS-in-JS in our applications is not necessary. It is, however, a popular tool used often in professional web development.

CSS-in-JS Libraries

While there are many CSS-in-JS libraries that go about CSS-in-JS in similar ways, the two we wanted to highlight are Styled Components and Emotion, which are the most popular two. We will be exploring Emotion in the tutorial later in this course. This will allow us to see how to use CSS-in-JS in a more hands-on, real context.

Summary

In this article, we learned about CSS-in-JS, the concept of styling our web app in JavaScript, and when and why it should be done. Consider what we've just learned when starting our next project and click next to get started with using Emotion!