upGrad

#RahoAmbitious

Full Stack Software Development

upGrad

Course: Advanced Front-End Development Using React

Lecture on: Styling React Components

Instructor: Parth DalalZ

TODAY'S AGENDA

- 'Styling' in React components
- Different ways to style components
- Inline styling
- Normal CSS
- CSS in JS libraries
- CSS modules
- SASS, SCSS and LESS
- Stylable
- What to use when

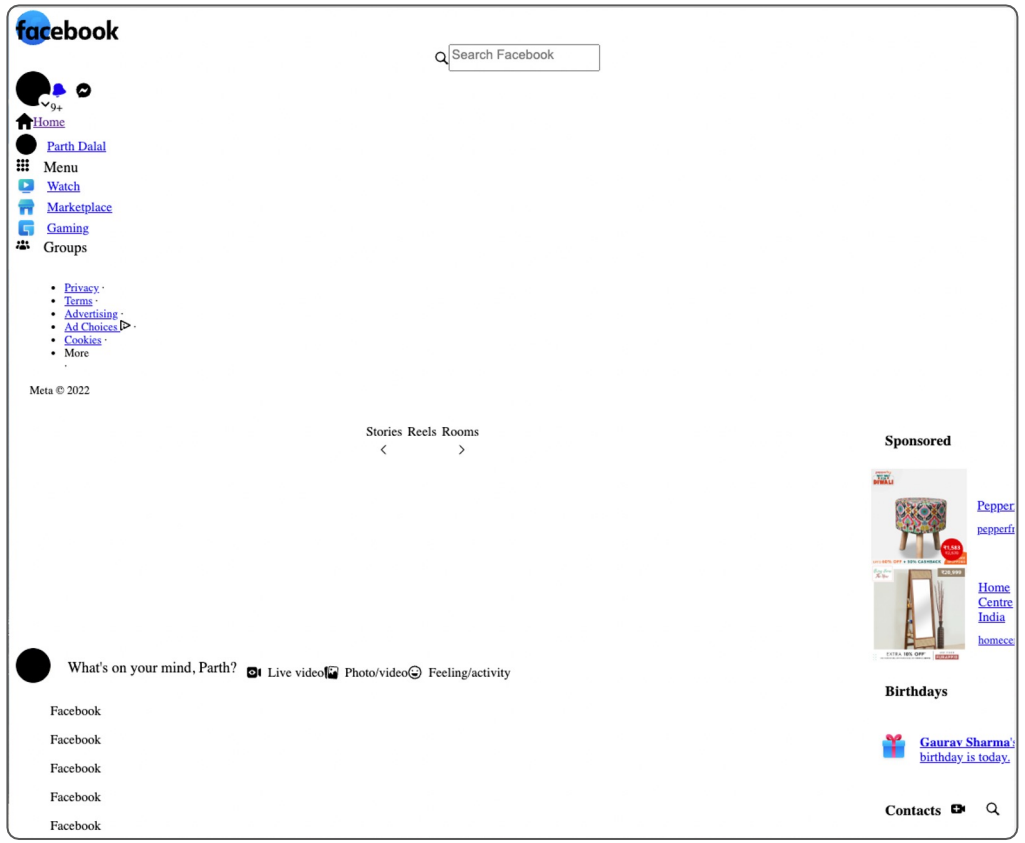
Styling in React Components

REACT COMPONENTS

- As you have seen, some websites look great and contain animations
- Many other websites look simple and subtle
- All the websites that you see uses CSS to provide intuitive look and the same is also in Reactjs.
- It is used to give a better appearance to DOM elements. This is analogous to how painting the walls of your house makes them look better

REACT COMPONENTS

This is how Facebook would look if styling were not applied



STYLING IN REACT

Ways of Styling

React offers styling in two ways: **inline** and **external**

- Let's use an example that shows two approaches to using inline styling to style the header component inside a phone directory app to help you understand inline styling:
  - Directly writing the styles alongside JSX
  - Moving all the styles to a constant object

STYLING IN REACT

Inline Styling: Approach 1

```
function App() {
 //...
    return (
    <div style={{ backgroundColor: "#44014C", width: "300px", minHeight:
"200px"}}>
      <h2 style={{ padding: "10px 20px", textAlign: "center", color:
"white"}}>ToDo</h2>
      <div>
        <Input onChange={this.handleChange} />
        <p>{this.state.error}</p>
      </div>
    </div>
  );
 }
```

In the above example, we have added inline style to the HTML element using a style attribute. Here, we used {{}} braces to incorporate the styles

STYLING IN REACT

Inline Styling – Approach 2: Moving Styles to a Constant Object

```
const divStyle = {
backgroundColor: "#44014C",
 width: "300px",
minHeight: "200px"};

<div style={divStyle}>
</div>
```

To move all the *styles* to a constant object, follow these steps:

- Move the JavaScript object out of the style property of the element
- Assign this object to a variable
- Assign this variable back to the style property of the element

Note: This approach makes the code more readable when you have a lot of styles to be applied to an element or component

STYLING IN REACT

Inline Styling – Approach 2: Moving Style as a Constant Object

- The property names look like CSS property names, but they are not exactly like them. These names are actually JavaScript identifiers; they can be considered the keys (or properties) of a JavaScript object

```
<div style={{textTransform: 'uppercase'}}>
Hello Div
</div>
```

This is why textTransform is written in camelCase in JSX unlike text-transform in CSS

- When a component or element is styled by moving out styles to a constant object, only one pair of curly braces is used in the style property

STYLING IN REACT

Characteristics of External Styling

- Write all the styles in an external stylesheet
- This is similar to writing external CSS
- Give your element or component a selector, id or class. Skip this in case you want to use a tag selector
- Create a .css file and write your CSS code in it based on the element's selector
- Import this stylesheet into the file where the component or element is defined on which you want to apply the given style

STYLING IN REACT

Characteristics of External Styling

```css
.header {
    text-align: center;
    padding: 20px;
    background: #000;
    color: #fff;
    text-transform: uppercase;
}
```

Import the file in which the CSS code is written, and the CSS will be applied automatically (similar to how it happens with external stylesheets)

Poll 6

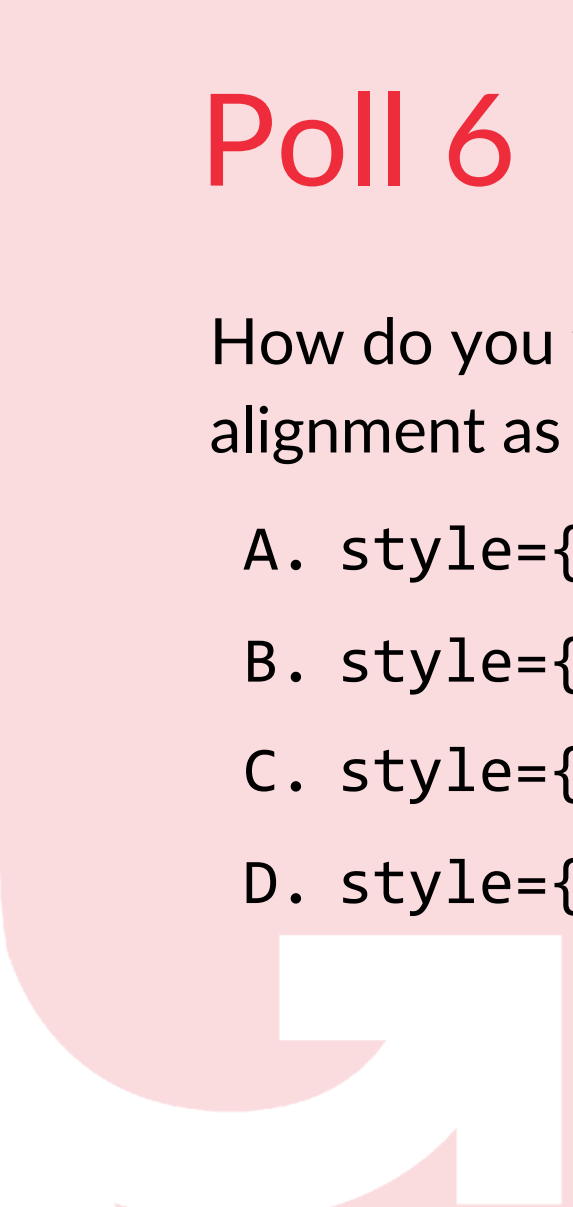
How do you write an inline style directly alongside JSX to specify the text alignment as ‘center’ and the color as ‘blue’?

A. `style={{textAlign: 'center', color: 'blue'}}`

B. `style={text-align: 'center', color: 'blue'}`

C. `style={{text-align: center, color: blue}}`

D. `style={{textAlign: center, color: blue}}`



Poll 6

How do you write an inline style directly alongside JSX to specify the text alignment as ‘center’ and the color as ‘blue’?

A. `style={{textAlign: 'center', color: 'blue'}}`

B. `style={text-align: 'center', color: 'blue'}`

C. `style={{text-align: center, color: blue}}`

D. `style={{textAlign: center, color: blue}}`

STYLING IN REACT

CSS in JS

- Similar to how HTML can be written in JS using JSX, CSS can be written using JS and is known as CSS-in-JS
- These styles are also scoped to individual components
- You do not need to write a CSS file separately for this
- Changing one component's styles will not impact the styles of the rest of your application
- Also known as 'tagged template literal', CSS-in-JS often makes use of a special type of this JavaScript function. We can still write plain CSS style rules directly in our JS

STYLING IN REACT

CSS in JS: Example

Example here

STYLING IN REACT

CSS in JS

Some important points to note are as follows:

- You can write normal CSS styles and include nested styles and pseudo-classes (such as hover)
- You can associate styles with any valid HTML element, such as the button element above (see styled.button).
- You can create new components with these associated styles. See how Button is used in the example
- You can pass the styles as props and can create dynamic style on the fly

STYLING IN REACT

CSS in JS: Pros and Cons

Pros

- CSS-in-JS is predictable – styles are scoped to individual components
- Since our CSS is now JS, we can export, reuse and even extend our styles through props
- CSS-in-JS libraries ensure there are no styling conflicts by generating unique classnames for your written styles
- There is no need to focus on naming conventions for your classes; just write styles

Cons

- Unlike plain CSS, you will need to install one or more third-party JavaScript libraries, which will add weight to your built project

STYLING IN REACT

CSS Modules

CSS modules are another alternative to styling React components:

- You can write normal CSS or SASS both while using CSS modules, and no additional library is required to get started
- Every CSS module file must have the name ‘module’ in it and end in the appropriate extension (if we are using CSS or SASS/SCSS)
- CSS modules are written just like normal CSS but are imported and used as if they were created as objects (inline styles)
- The advantage of using CSS modules is that it helps avoid the problem of class conflicts with normal CSS. The properties that we are referencing turn into unique classnames that cannot conflict with one another when our project is built

STYLING IN REACT

CSS Modules: Example

Example here

STYLING IN REACT

SASS and SCSS

SASS stands for syntactically awesome style sheets and has the following characteristics:

- SASS provides additional features to CSS, such as variables, extending styles and nesting
- Every CSS module file must have the name ‘module’ in it and end in the appropriate extension (if we are using CSS or SASS/SCSS)
- It can be written in two file formats: .scss & .sass
- SCSS are written in a similar syntax to normal CSS; however, SASS styles do not require us to use open and closing brackets while writing style rules
- As SASS is not written in plain CSS, it must be converted into it. If we are using React, we must use a library called node-sass and to install that you can use below command:

```
npm install node-sass
```

STYLING IN REACT

SASS and SCSS: Example

SCSS:

```scss
nav {
  ul {
    margin: 0;
    padding: 0;
    list-style: none;
  }

  li { display: inline-block; }

  a {
    display: block;
    padding: 6px 12px;
    text-decoration: none;
  }


 }
```

SASS:

```sass
nav
  ul
    margin: 0
    padding: 0
    list-style: none

  li
    display: inline-block

  a
    display: block
    padding: 6px 12px


       text-decoration:

  none
```

STYLING IN REACT

SASS and SCSS

SASS has the following benefits:

- **Variables:** You can use dynamic values by writing variables, just like in JavaScript, by declaring them with a $ at the beginning
- Two variables can be used in multiple rules: $font-stack and $text-color
- **Extending/Inheritance:** You can add onto style rules by extending them. To extend rules, you must create your own selector, which can be reused like a variable. The names of rules that you want to extend start with %
- The variable %font-basic is inherited by the rules body and .testimonial-name
- **Nesting:** Instead of writing multiple rules that begin with the same selector, you can nest them

STYLING IN REACT

LESS

LESS stands for syntactically leaner style sheets. It has the following characteristics:

- LESS just looks like CSS, with some additions to the CSS language
- It provides additional features such as variables, mixins, nesting and @ rules

```
@width: 10px;

@height: @width + 10px;

#header {

  width: @width;

  height: @height;

}
```

```
.bordered {
  border-top: dotted 1px black;
  border-bottom: solid 2px black;
}

#menu a {
  color: #111;
  .bordered();
}

.post a {
  color: red;
  .bordered();
}
```

STYLING IN REACT

LESS

LESS in React

- However, there is no such easy solution to handle compiling LESS stylesheets. This is problematic because you have to manually configure webpack to compile LESS stylesheets

Setting Up LESS Watcher Config

- The less-watcher-compiler package takes a .json file for configuration. Create a .json file called less-watcher.config.json in the base directory. It will watch for all the directories which needs to be compiled and will generate css files to the given output directories

Setting Up Project Scripts

- We want to run the LESS compiler along with the start script. Let's install the **concurrently package** as a dev dependency so we can run our compile script together with react-scripts

The setup script looks similar to:

```
"scripts": {
    "start": "concurrently --kill-others \"less-watch-compiler --config less-watcher.config.json\" \"react-scripts start\"",
}
```

STYLING IN REACT

react-styleable

Styleable is the CSS superset that extends CSS.

- This extends CSS so it is easier to use in a component ecosystem, but without losing any of the declarative, familiar, static and fast aspects of CSS
- Each component exposes a Style API that maps its internal parts and states so you can reuse components across teams without sacrificing stylability or scalability
- It provides the ability to see errors at build time or even while working in your IDE. Wave goodbye to silent run-time breakage misery
- It performs build-time transpilation and requires only a minimal runtime. It uses custom states and properties for dynamic interactions
- You can easily create complex designs using CSS or JavaScript and reuse them across projects
- You can use our language-intelligence IDE extension for a better development experience for completions, definitions, hinting, diagnostics and more

STYLING IN REACT

react-styleable

Styleable is a CSS preprocessor that enables you to write reusable, highly-performant and styled components

You can install Styleable through the following steps:

- To begin writing your own project, you can *create a stylable app* from their boilerplate
- To work with an existing project, install one of our *available integrations*
  These integrations include NextJS, TypeScript and Webpack
- There are two dependencies: `stylable and @stylable/webpack-plugin`
- The difficulty level is relatively high

STYLING IN REACT

When to Use What

Using these different ways of styling can be ones decision depending on the application structure, complexity and so on:

- If you know CSS, you probably know SASS. SASS comes with two different syntaxes: SASS itself and SCSS, which is used more. SCSS syntax is CSS compatible, so you just have to rename your .css file to .scss,and you can start availing of the advantages of SCSS directly
- Nesting in CSS (or SCSS) is one of the best things that SASS offers
- A comparison between SCSS and CSS shows that SCSS has more features. When it comes to composing, mixin and include can be used in SCSS, as they work quite well
- If you are not the biggest fan of CSS-in-JS, then Stylable might be for you

KEY TAKEAWAYS

- There are various ways to style react application which includes inline CSS, normal CSS, CSS modules, SASS and SCSS, LESS, Styleable.
- How these packages are used and their configurations.
- Examples on styling a react components

THE FOLLOWING TASKS ARE TO BE COMPLETED AFTER TODAY'S SESSION

MCQs
Coding Questions

upGrad

#RahoAmbitious

Thank You!