

1

- There are many ways to handle the UI style in React
 - ▣ Use CSS files
 - ▣ Inline styles
 - ▣ CSS modules
- Developers need to give careful consideration to managing and maintaining the UI

2

Importing CSS

- In React apps the CSS code is transpiled before it's loaded into the browser.
- The import statement will not be shown in the code that's executed in the browser.

```
JS index.js
// import React from 'react'
import ReactDOM from 'react-dom/client'
import App from './App';
import './index.css';

ReactDOM.createRoot(document.getElementById('root')).render(<App />)
```

3

- Because React doesn't give developers many rules about exactly how to structure user interfaces, users are free to mix and match solutions and patterns and find out what works best.

4

Adding styling

- Add CSS styles and classes to JSX elements just as you would to regular HTML elements.
- CSS code is independent of React
- For CSS files, the styles defined in that CSS file will be applied **globally**.
- `import React from 'react';`
- `import ReactDOM from 'react-dom/client';`
- `import './index.css';`
- `import App from './App'`



5

Including CSS in a Component

- When CSS is added to a component, it cascades to the children of the component.

```

src > components > ColdDrinks.jsx > ColdDrinks
1 import './ColdDrinks.css'
2
3 export const ColdDrinks = function () {
4   return (
5     <div className="foodDiv">
6       <h3>Cold Drinks on offer</h3>
7       
8     </div>
9   )
10 }
11
  
```

6

- Within the `index.css` file or in any other CSS files that the `index.css` file imports, you can specify any styles to your HTML elements (that is, to your JSX elements in your components).
- The `index.jsx` file or any other JavaScript file (including component storage files) could also have extra CSS import statements linking to other CSS files.
- Note that CSS styles are always global.
- The styles defined in a CSS file will be applied worldwide regardless of whether you import it into `index.jsx` or a JavaScript file dedicated to a component.

7

CSS issues

- This trickling of styles from parent to child elements can become problematic.
- Prioritise the different styles.
- Can create confusion and unwanted results in components.
- Can use the one class selector (bootstrap).
- CSS is not a programming language, and it doesn't have scope.
- Many developers use a CSS preprocessor, such as SASS.

8

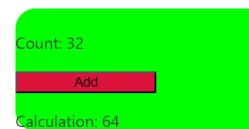
- Inline styles in REACT
- Note that the style object, is enclosed in curly brackets and written as an object literal, needs to have curly brackets surrounding it.
- However, inline styles can become difficult to maintain.

9

Inline styling

- Use CSS files to define global CSS styles and use different CSS selectors to target different JSX elements.

```
return (
  <>
    <p>Count: {count}</p>
    <button style={{ width: ' 140px', backgroundColor: 'crimson' }}
      onClick={() => setCount((c) => c + 1)}> Add</button>
    <p>Calculation: {calculation}</p>
  </>
);
```



- N.b. attribute names must use camel case

10

```
const headerStyle = {
  paddingTop: '6px', paddingRight: '14px',
  margin: '20px', borderRadius: 20,
  color: '#333', backgroundColor: 'lime'
}
```

```
function App() {
  return (
    <div style={headerStyle}>
      <Counter />
    </div>
  );
}
```

Count: 32

Add

Calculation: 64

11

Npm styled-components

□ <https://styled-components.com/docs/basics#installation>

```
PS C:\Component Dev\helloworld> npm install styled-components
```

```
added 10 packages, changed 1 package, and audited 1510 packages in 10s
```

```
242 packages are looking for funding
  run `npm fund` for details
```

```
10 vulnerabilities (3 moderate, 6 high, 1 critical)
```

```
To address issues that do not require attention, run:
  npm audit fix
```

12

Styled Components

- Styled Components is a library built in for styling React apps.
- `<StyledAddButton>add style</StyledAddButton>`

src > JS App.js > Counter

```
1 import './App.css';
2 import { useState, useEffect } from "react";
3 import styled from "styled-components";
4
```

Count: 4

add style

Add

Calculation: 8

```
const StyledAddButton = styled.button`
  float: left;
  margin: 5px 0px 0px 3px;
  align-items: center;
  padding: 6px 14px;
  border-radius: 8px;
  border: none;
  color: #fff;
  background-color: #367af6;
  cursor: pointer;
`;
```

The advantage of
Styled Components is
reuse

13

Styling using a CSS file

```
1 import './App.css';
2 import { useState, useEffect } from "react";
3 import styled from "styled-components";
4
5 function Timer() {
6
7   return (
8     <>
9       <p>Count: {count}</p>
10      <StyledAddButton>add style</StyledAddButton>
11      <button style={ { width: ' 140px', backgroundColor: 'crimson ' } }
12        onClick={() => setCount((c) => c + 1)}> Add</button>
13      <button className='highLight'>add css</button>
14      <p>Calculation: {calculation}</p>
15    </>
16  );
17 }
```

JS App.js

App.css

src > # App.css > ...

```
1 .App {
2   text-align: center;
3 }
4
5 .highLight {
6   background-color: #d1ef23;
7 }
8
```

Count: 9

add style

Add

add css

Calculation: 18

14

Conditional styling

- Derive styles or class names dynamically based on different conditions.
- `${props => props.active ? 'green' : 'red'}`

15

SASS (Syntactically Awesome Style Sheets)

- CSS preprocessor.
- Extends CSS with features like variables, nesting, and functions.
- Compiles into standard CSS for the browser.
- Makes stylesheets more organised, reusable, and easier to maintain.

16

SASS

- Why Use Sass?
 - ▣ Stylesheets are getting larger, more complex, and harder to maintain.
 - ▣ Sass lets you use features that do not exist in CSS, like variables, nested rules, mixins, imports, inheritance, built-in functions, and other stuff.
- To run `npm install node-sass`

17

Storing variables

```

□ /* define variables for the primary colours */
$primary_1: #a2b9bc;
$primary_2: #b2ad7f;
$primary_3: #878f99;

/* use the variables */
.main-header {
  background-color: $primary_1;
}

.menu-left {
  background-color: $primary_2;
}

.menu-right {
  background-color: $primary_3;
}

```

18

Sass Example

```
/* define variables for the primary colors */
$primary_1: #a2b9bc;
$primary_2: #b2ad7f;
$primary_3: #878f99;

/* use the variables */
.main-header {
  background-color: $primary_1;
}

.menu-left {
  background-color: $primary_2;
}

.menu-right {
  background-color: $primary_3;
}
```

#a2b9bc

#b2ad7f

#878f99

19

CSS MODULES

- CSS Modules give you some of the benefits of using JavaScript style objects while using standard CSS stylesheets.
- Specifically, CSS Modules solve the problem of name conflicts and scoping in CSS.
- CSS modules can be written like normal CSS files, and then imported into your components as if they were JavaScript.
- In fact, what happens during the compilation of your components is that CSS modules are converted into JavaScript objects.

20

CSS module files

- By default, the rules you create in CSS module files are scoped locally to the component you import the styles into.
- If you want to create a global rule, you can do so by prefixing the name of the class with :global, like this:

```
:global .header1 {
  font-size: 2rem;
  font-weight: bold;
}
```

21

CSS Modules

- With CSS modules, you may write locally scoped classes.
- The class name will be generated automatically using CSS modules.
- With CSS modules, global scope issues can be resolved.

```
import appStyles from "../App.module.css";
<div className="App">
  <div className={appStyles.App} >
```

- Because of the App component having unique classes, its classes will not mix up with its child components.

22

Naming CSS Module Files

- Although CSS module files resemble ordinary CSS files, when you use them inside of Create React App, their filenames must end with `.module.css` to indicate to the compiler that they need to be processed as CSS modules.
- `my-component.module.css`

23

```

8  import "../App.css";
9  // =====
10 function App() {
11   const [movies, setMovieList] = useState(initialMovies);
12   //console.log(movies);
13
14   function addMovieToList(newMovie) {
15     setMovieList((movielist) => [...movielist, newMovie]);
16   }
17
18   return (
19     <>
20     <CartProvider>
21       <Movie movies={movies} />
22       <MovieForm onAddMovie={addMovieToList} />
23       <Cart />
24     </CartProvider>
25   </>
26 );
27

```

General CSS

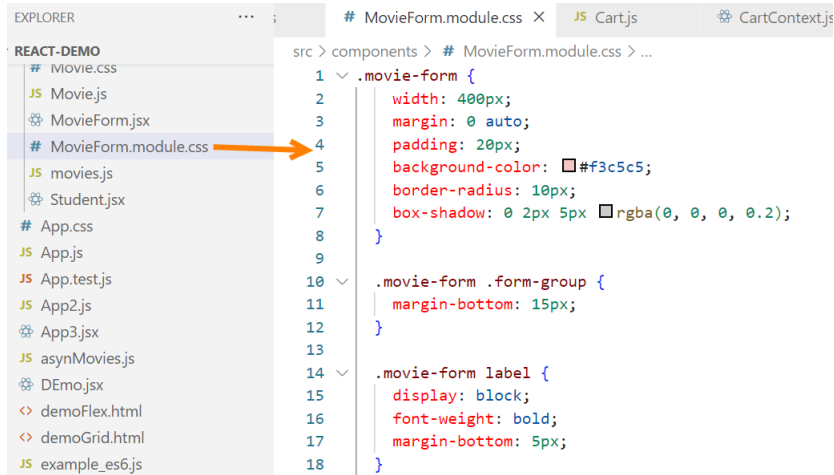
Add Movie

- Avatar Action, Adventure, Fantasy, 9.99Counter:
- I Am Legend Drama, Horror, Sci-Fi, 7.99Counter:
- 300 Action, Drama, Fantasy, 11.99Counter:

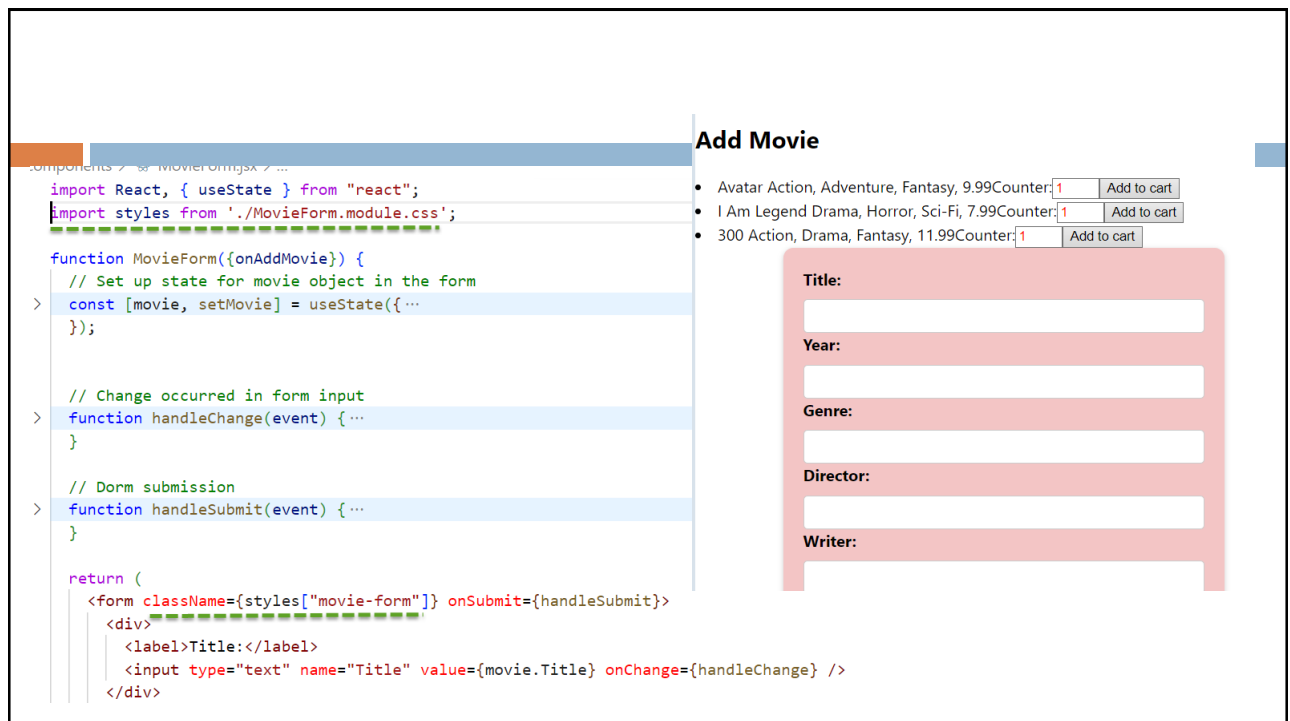
Title:
 Year:
 Genre:
 Director:
 Writer:
 Actors:
 Plot:
 Poster URL:
 Price:

Cart

24



25



26

Other styling approaches

- Material UI (MUI) is a library of advanced components
 - ▣ <https://mui.com>
- Tailwindcss
 - ▣ <https://tailwindcss.com>
- Flowbite
 - ▣ <https://flowbite.com/>

27

Tailwind

- CSS framework like bootstrap with numerous CSS classes.
- Styles elements directly in HTML or JSX using **predefined classes**

01 Create your project

Start by creating a new Vite project if you don't have one set up already. The most common approach is to use [Create Vite](#).

Terminal

```
npm create vite@latest my-project
cd my-project
```

02 Install Tailwind CSS

Install `'tailwindcss'` and `'@tailwindcss/vite'` via npm.

Terminal

```
npm install tailwindcss @tailwindcss/vite
```

- CDN `<script src="https://cdn.tailwindcss.com"></script>`

28

Install Tailwind CSS and Flowbite

- ❑ `npm install -D tailwindcss`
- ❑ `npx tailwindcss init` (creates `tailwind.config.js`)
- ❑ `npm install flowbite`

29

❑ Configure `tailwind.config.js`

```
JS tailwind.config.js > [?] <unknown> > [?] content
1  // tailwind.config.js
2  module.exports = {
3    content: [
4      "**/*.html",
5      "**/*.js",
6    ],
7    theme: {
8      extend: {},
9    },
10   plugins: [
11     require('flowbite/plugin')
12   ],
13 };
14
```

30

- Place your `styles.css` file directly in the root of your project (the same location as `index.html`).

```
# styles.css
1  @tailwind base;
2  @tailwind components;
3  @tailwind utilities;
4
```

31

Tailwind in React

- Eliminates the need for CSS files
- Responsive, fast, and maintainable
- Integrates easily with frameworks like Flowbite

32

Other UI frameworks

- Material UI
- Chakra UI
- React Bootstrap
- Semantic UI React

33

- <https://tailwindcomponents.com/cheatsheet/>
- <https://www.codeinwp.com/blog/tailwind-css-tutorial/>
- <https://tailwindcss.com/>

34

flowbite

- <https://flowbite.com/docs/getting-started/react/>
- “Coupled with Tailwind CSS and the React components from Flowbite you will be able to develop applications faster than ever based on the Flowbite React library using advanced theming functionalities and React specific components and methodologies”
- UI component library built on top of Tailwind CSS.
- Collection of prebuilt, responsive, and accessible components.

35

Using Flowbite with React

- `npm install -D tailwindcss postcss autoprefixer`
- `npx tailwindcss init -p`

36

```

{} package.json > ...
1  {
2    "name": "flowbite",
3    "version": "1.0.0",
4    "description": "",
5    "main": "index.js",
6    "scripts": {
7      "build:css": "npx tailwindcss -i ./src/styles.css -o ./dist/styles.css --watch"
8    },
9    "author": "",
10   "license": "ISC",
11   "devDependencies": {
12     "tailwindcss": "^3.4.14"
13   },
14   "dependencies": {
15     "flowbite": "^2.5.2"
16   }
17 }
18

```

npm run build:css

37

□ Then link the compiled CSS in the HTML file

```

<> index2.html > html > head > title
1  <!DOCTYPE html>
2  <html lang="en">
3    <head>
4      <meta charset="UTF-8" />
5      <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6      <title>Flowbite Form Example</title>
7      <link href="styles-built.css" rel="stylesheet" />
8
9    </head>
10   <body class="bg-pink-100 flex items-center justify-center min-h-screen">
11

```

38

Add Flowbite Components

```
<body class="bg-pink-100 flex items-center justify-center min-h-screen">

  <div class="w-full max-w-md p-8 bg-white rounded-lg shadow-lg">
    <h2 class="text-2xl font-bold mb-6 text-center">Contact Information</h2>

    <form action="#" method="POST">
      <!-- Name -->
      <div class="mb-4">
        <label for="name" class="block text-sm font-medium text-gray-700 mb-2">Name</label>
        <input
          type="text"
          id="name"
          name="name"
          required
          class="w-full px-4 py-2 border rounded-lg focus:ring-blue-500 focus:border-blue-500"
          placeholder="Enter Name"/>
      </div>
```

39

Summary

- In React you have many styling choices.
 - ▣ CSS Stylesheets
 - ▣ Inline Styles
 - ▣ CSS Modules
 - ▣ Styled-Components
 - ▣ Tailwind CSS
- Each approach has its strengths and weaknesses.

40

CSS Flexbox

- Designed for one-dimensional layouts (either a row or column).
- Container: `display: flex;` on the parent element to enable Flexbox.
- Axis:
 - ▣ Main Axis: The primary direction items are arranged in (`row` or `column`).
 - ▣ Cross Axis: Perpendicular to the main axis.

41

Container Properties

- `flex-direction:`(`row`, `row-reverse`, `column`, `column-reverse`).
- `justify-content`: **Aligns items along the main axis.**
 - ▣ `flex-start`, `flex-end`, `center`, `space-between`, `space-around`, `space-evenly`.
- `align-items`:
 - ▣ `stretch`, `flex-start`, `flex-end`, `center`, `baseline`.`flex-wrap: .`

42

Item Properties

- **flex:** A shorthand for flex-grow, flex-shrink, and flex-basis.
 - ▣ Example: `flex: 1;` (lets the item grow to fill space).
- **order:** Controls the order of items (default is 0).
- <https://css-tricks.com/snippets/css/a-guide-to-flexbox/>
- <https://granneman.com/downloads/web-dev/CSS-Layout-Flexbox.pdf>

43

CSS Grid

- Designed for two-dimensional layouts
- Key components
 - ▣ **Container:** Use `display: grid;` on the parent element to enable Grid.
 - ▣ **Grid Template:** Define rows and columns with `grid-template-columns` and `grid-template-rows`.

44

Container Properties

- `grid-template-columns / grid-template-rows`: **Defines column/row sizes.**
 - **Ex:** `grid-template-columns: 1fr 2fr;`
- `gap`: **Defines space between grid items.**
 - **Ex:** `gap: 10px;`
- `justify-items`: **Aligns items horizontally** (start, end, center, stretch).
- `align-items`: **Aligns items vertically** (start, end, center, stretch).
- <https://css-tricks.com/snippets/css/complete-guide-grid/>

45

Children recap

- **Children** : special built-in prop that represents the content nested between a component's opening and closing tags.
- Allows you to pass JSX or other components inside another component.

src > components > Demo2.jsx > Demo2

```
1 import React from 'react'
2
3 export const Demo2 = ({children}) => {
4   return <div className="border p-4 rounded-lg">{children}</div>
5 }
6
```

```
return (<>
  <Demo2>
    <h2>Demo 2</h2>
    <p>passing content as children.</p>
  </Demo2>
</>)
```

46