Styles can be implement in the following ways

* Css style sheets
* Inline style
* Css in js
* CSS Modules( in react)
* Style-components(in react)

**CSS style sheets**

Css style sheets is nothing but language that describe the style of an Html document. In general it use selector to identify the elements. Css is store as text with “.css”. And it can be import in script tag.

*CSS style sheet in React*

In General, react CSS file can import in “jsx” file. It will load by css loader to html page. You can use sass in instead of css file by using sass-loader.

**Pros:**

1. Css is universal and it won’t depends any frontend language. That means once CSS Sheets created, it can be use any webproject by simply import. Example bootstrap stylesheet can be used React,vue and static webpage.
2. **Caching & Performance :** standard css files are easy for browser to optimize for, And caching files locally for repeated visits so giving performance wins.
3. **Quickly Iterate A New Design :**

* Can create entire new design digging through potentially hundreds of components
* You can also split your stylesheets into specific files to create some modularity

1. **Frameworks**

* Lot of frameworks are available for css (bootstrap,materialcss..etc).
* For the new developer css sheet is very easy to work.

**Cons:**

1. Reability: for fullscale project readability is much more difficult.
2. **Legacy CSS Can Live On For Years**: Identification unused css will be difficult and may live deprecated styles after so long years
3. **No True Dynamic** **Styling** : if you want conditionally change style of an element, it will js.
4. **Maintaining Consistency** : for maintain consistency you need to follow some best practices to create class name and var name

**Inline style**

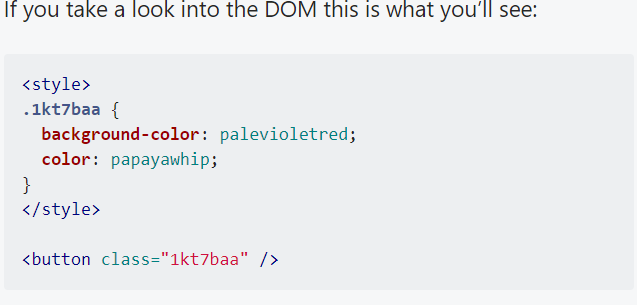
* Basically inline styles will use only for override some property.
* For developing styles css sheet is better than inline style. And inline style should not use much case and should use only for override css properties.

**CSS in js**

“Css in JS “ is CSS property definition should in with in js file. This approach are like inline style, but then they take those styles and inject an actual string of CSS in a <style> tag into the DOM.

In React does not support natively for css in js. So you can use **Aphrodite** npm library.



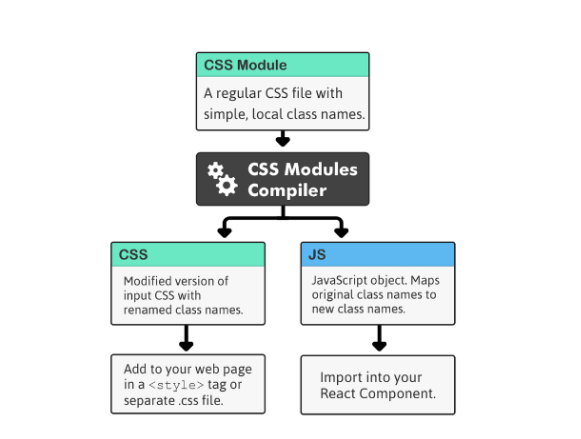


Diff **inline vs css-in-js:**

* Inline style does not support pseudocode selector like :disabled, :before, :nth-child. Meanwhile in css-in-js, can support all css features.

**CSS modules:**

A CSS Module is a CSS file in which all class names and animation names are scoped locally- by default. Take a look.

****

So Every component own css file which is scope only with in component this magic will happen only building time. After building the component new classname will create and it will inject in dom

For example

**Before build**

/\* Thumbnail.css \*/

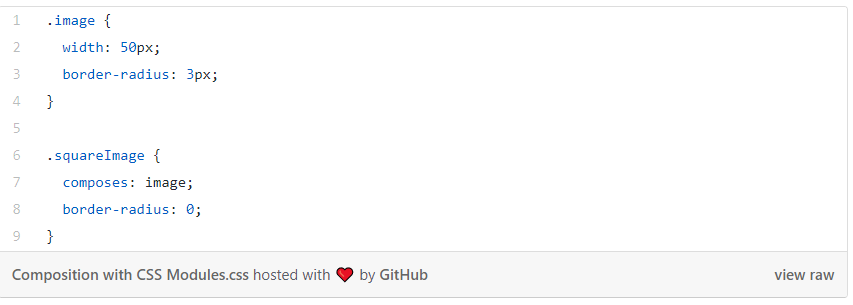
|  |
| --- |
| .image { |
|  | border-radius: 3px; |
|  | } |

|  |
| --- |
| import styles from './Thumbnail.css'; |
| export default ()=> { return (<img className={styles.image}/>) }  **After Build**   |  | | --- | | /\* Rendered DOM \*/ |   <img class="Thumbnail\_\_image\_\_\_1DA66"/>  /\* Processed Thumbnail.css \*/   |  | | --- | | .Thumbnail\_\_image\_\_\_1DA66 { border-radius: 3px;  } | |  |

|  |  |
| --- | --- |
|  |  |

Props:

1. Maintain Namespacing will unnecessary because of local scope
2. And Build tool will eliminate unused css when build css. So project css code will be optimized.
3. **Composition –** it will allow compose css selector like object.assign method.

****

1. **Dependencies**

Can Import any dependency modules by using import like below structure

.header {

composes: clearfix from './utils.css';

}

1. **Conditional styling**

Changing styles based on state and other conditions is another good use-case for composition.

/\* Button.css \*/

.button {

background-color: black;

}

.disabledButton {

composes: button;

opacity: .5;

}

/\* Button.jsx \*/

import styles from './Button.css';

static propTypes = {

isDisabled: React.PropTypes.bool

};

render() {

var className = this.props.isDisabled ? styles.disabledButton : styles.button;

return (<button className={className}/>);

}

**Styled-component**

One of the solution for css-in-js is styled-components.

const Button = styled.button`

  background: ${props => props.bg ? props.bg : "#0069D9"};

  color: ${props => props.color ? props.color : "white"};

  font-size: 1em;

  margin: 1em;

  padding: 0.25em 1em;

  border: 2px solid ${props => props.bg ? props.bg : "#0069D9"};

  border-radius: 3px;

  width: ${props => props.block ? '98%' : ""};

  cursor: pointer;

`;

const LargeButton = styled(Button)`

    font-size : 1.5em;

`;

const SmallButon = styled(Button)`

    font-size : 0.7em;

`;

Pros:

Above code style is written in styled-components. So custom button component is created with style and that some style will change based on props. For example if you give background color as props then it will override default color.

**Styled HOD:**

It will add extra css.

**Conclusion**

Before going to create style we have to decide css-in-js or normal css.

For taking css-in-js for the project, **Aphrodite,** **Radium, and styled component** are option to make style.

For taking css with global stylesheets and stylesheets are need to use another fron-end framework, in that case normal CSS are option

For taking css with with local scope styles, for this case Css Modules are for better for react js.