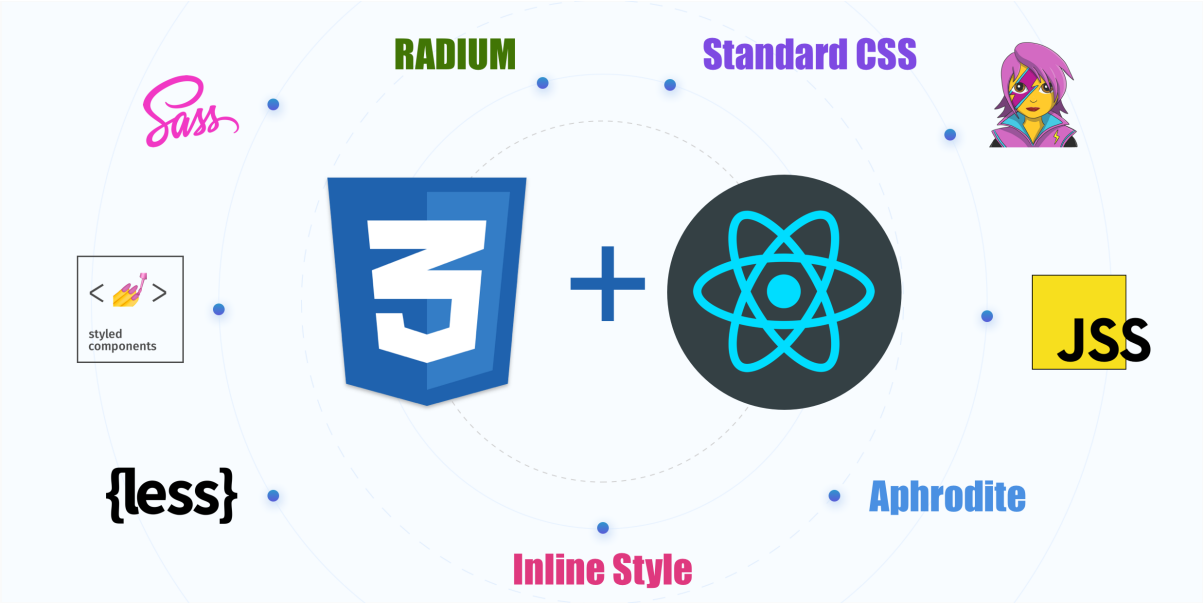
CSS in React

https://mdbootstrap.com/

https://reactstrap.github.io/

https://react-bootstrap.github.io/

https://www.npmtrends.com/react-bootstrap-vs-reactstrap



CSS-IN-CSS: CSS IN REACT

The most basic way is to just use vanilla CSS in React with CSS files. Next to each component or to each set of components, you can have a file with the .css extension. For example, the following CSS file defines a CSS class for a button:

```
.button {
  cursor: pointer;
  border: 1px solid #1a202c;
  padding: 8px;
  min-width: 64px;

  background: transparent;

  transition: all 0.1s ease-in;
}

.button:hover {
  background: #1a202c;
  color: #ffffff;
}
```

CSS-IN-CSS: CSS MODULES IN REACT

If you are using create-react-app, you can use CSS Modules right away. However, if you are using a custom React with Webpack setup, you need to configure Webpack for it.

CSS Modules can be used with vanilla CSS but also with CSS extensions like Sass. Let's see how a CSS module can be defined in a *style.module.css* (vanilla CSS) or *style.module.scss* file (Sass):

CSS-IN-JS: STYLED COMPONENTS

There is CSS setup needed for styled components, because everything comes with JavaScript. Essentially as the strategy CSS-in-JS already says, we will not need any CSS file, because all CSS is defined in JavaScript. Before you can use Styled Components you need to install them on the command line:

```
npm install styled-components
```

Styled Components takes the approach to create components just from a HTML tag and a style string. Let's see how this looks for a button element which becomes a Button component in our JavaScript file:

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

const YourButton = styled.button`
  cursor: pointer;
  border: 1px solid #1a202c;
  padding: 8px;
    min-width: 64px;

  background: transparent;

  transition: all 0.1s ease-in;

  &:hover {
    background: #1a202c;
    color: #ffffff;
  }
`;
```

The Button variable is a valid React component which can be used in JSX. Any properties like the `onClick` are passed through to the real button HTML element. In addition, a styled component already comes features (here: CSS nesting with parent selector) which we would usually gain from a CSS extension like Sass.

```
function Basket({ items, onClick }) {
  return (
    <ul>
      {items.map((item) => (
        <likey={item.id}>
          {item.name}
          <YourButton type="button"onClick={() => onClick(item)}>
            {item.isFavorite ? 'Unlike' : 'Like'}
          </YourButton>
        </li>
      ))}
    </ul>
  );
}
```

The syntax of Styled Components isn't very clear for many React beginners. Basically the `styled` object offers you a function for each HTML element (e.g. button,

ul, li). The function can be called with JavaScript template literals whereas everything you place into the template literals becomes the style of the component:

```
const UnorderedList = styled.ul`
  margin: 0;
  padding: 0;
  list-style-type: none;
`;

const ListItem = styled.li`
  display: flex;
  justify-content: space-between;
  padding: 8px 0;
`;
```

The styled components can be defined in the same file or somewhere else. After all, they are just regular React components after you have defined them, which makes them exportable or directly usable in your JSX:

```
function Basket({ items, onClick }) {
  return (
    <UnorderedList>
      {items.map((item) => (
        <ListItemkey={item.id}>
          {item.name}
          <Buttontype="button"onClick={() => onClick(item)}>
            {item.isFavorite ? 'Unlike' : 'Like'}
          </Button>
        </ListItem>
      ))}
    </UnorderedList>
  );
}
```

With a CSS-in-JS approach like Styled Components you still need to write CSS, but you write it in JavaScript. In addition, a library like Styled Components already solves many problems that we had to solve with CSS Modules (scoping) and Sass (CSS features) previously.