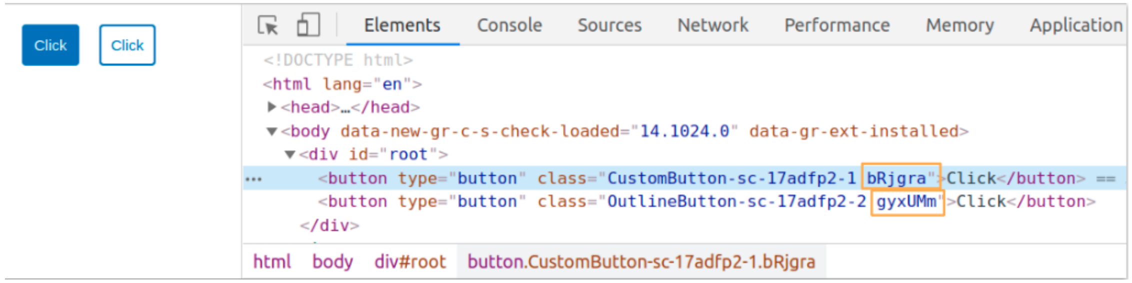
3.Advantages of Styled Components

3.1 Unique Class Names

Styled Components generate a **unique** class name(s) for your styles.

Example:



```
<!DOCTYPE html>
<html lang="en">
 ▶<head>…</head>
 ▼<body data-new-gr-c-s-check-loaded="14.1024.0" data-gr-ext-installed>
   ▼<div id="root">
...     <button type="button" class="CustomButton-sc-17adfp2-1 bRjgra">Click</button> ==
        <button type="button" class="OutlineButton-sc-17adfp2-2 gyxUMm">Click</button>
     </div>
```

html body div#root button.CustomButton-sc-17adfp2-1.bRjgra

3.2 Elimination of dead styles

Styled components remove unused styles, even if they're declared in your code.

3.3 Dynamic Styling

- Let's assume we have two types of buttons on our page, one with a **white** background, and the other **navy blue**.
- We do not have to create two styled-components for them. We can adapt their styling based on their props.

File:src/styledComponents.js

JSX

```jsx
import styled from "styled-components";

export const CustomButton = styled.button`
  padding: 10px;
  margin-right: 20px;
  font-size: 15px;
  color: ${(props) => (props.outline ? "#0070c1" : "#ffffff")};
  border-radius: 4px;
  border: 2px solid #0070c1;
  background-color: ${(props) => (props.outline ? "#ffffff" : "#0070c1")};
```

```
`;
```

File:src/App.js

```jsx
import "./App.css";
import { CustomButton } from "./styledComponents";

const App = () => (
  <>
    <CustomButton type="button" outline={false}>Click</CustomButton>
    <CustomButton type="button" outline={true}>Click</CustomButton>
  </>
);

export default App;
```

3.4 Can differentiate between the types of props they receive

File:src/App.js

```jsx
import "./App.css";
import { CustomButton } from "./styledComponents";

const App = () => (
  <>
    <CustomButton outline={false}>Click</CustomButton>
    <CustomButton outline={true}>Click</CustomButton>
  </>
);

export default App;
```

You'll notice, though, that we haven't given our button a type in the above code snippet. Let's do that:

File:src/App.js

JSX

```jsx
import "./App.css";
import { CustomButton } from "./styledComponents";

const App = () => (
  <>
    <CustomButton type="button" outline={false}>Click</CustomButton>
    <CustomButton type="submit" onClick={() => alert("clicked")} outline={true} >Click</CustomButton>
  </>;
);

export default App;
```

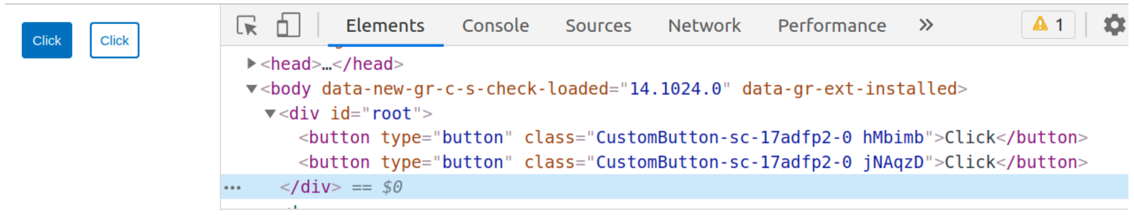
Collapse

Styled components can differentiate between the types of props they receive. They know that type is an HTML attribute, so they render

`<button type="button">Click</button>` , while using the **outline** prop in their own processing. Notice how we attached an event handler, too?

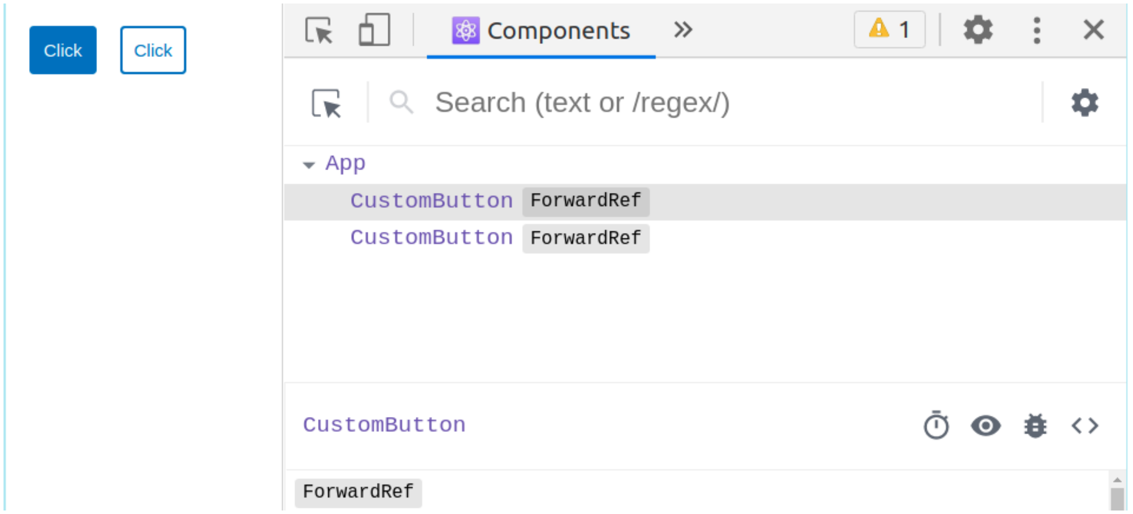
3.5 Easier Debugging

- babel-plugin-styled-components plugin adds support for a nicer debugging experience.
- This option enhances the attached CSS class name on each component with richer output to help identify your components in the DOM without React DevTools. In your page source you'll see: `<button type="button" class="CustomButton-asdfxxx asdfxx" />` instead of just `<button type="button" class="asdfxxx" />` .



```html
▶<head>…</head>
▼<body data-new-gr-c-s-check-loaded="14.1024.0" data-gr-ext-installed>
  ▼<div id="root">
      <button type="button" class="CustomButton-sc-17adfp2-0 hMbimb">Click</button>
      <button type="button" class="CustomButton-sc-17adfp2-0 jNAqzD">Click</button>
    </div> == $0
```

- It also allows you to see the component's displayName in React DevTools. For example, consider writing a styled component that renders a button element, called **CustomButton**. It will normally show up in DevTools as styled.button, but with the displayName option enabled, it has the name you gave it **CustomButton**.



App
CustomButton ForwardRef
CustomButton ForwardRef

CustomButton
ForwardRef

- This makes it easier to find your components and to figure out where they live in your app.
- For more info, you can go through this link here.

Note

- In order to configure create-react-app with the babel-plugin-styled-components, we have to use craco - Create React App Configuration Override is an easy and comprehensible configuration layer for create-react-app.
- We'll need to install craco with npm, and then create a craco.config.js at the root of our application, with the content:

JSX

```jsx
module.exports = {
  babel: {
    plugins: [
      [
        "babel-plugin-styled-components",
        {
          fileName: false,
        },
      ],
    ],
  },
};
```

Collapse

It is already configured.

3.6 Global Styling

- We will write **global** styles in the `styledComponents.js` file. Inside the `GlobalStyle` variable is where we define all global styles.
- We will import `GlobalStyle` component and place it at the top of React tree. In many react applications, that's typically the `App.js` file.
- `createGlobalStyle` is a helper function to generate a special StyledComponent that handles global styles.
- Normally, styled-components are automatically scoped to a local CSS class and therefore isolated from other components. In the case of **createGlobalStyle**, this limitation is removed.

Example:

File: src/styledComponents.js

JSX

```jsx
import { createGlobalStyle } from "styled-components";

export const GlobalStyle = createGlobalStyle`
 body {
  margin: 0;
  font-family: -apple-system, BlinkMacSystemFont, "Segoe UI", "Roboto", "Oxygen",
    "Ubuntu", "Cantarell", "Fira Sans", "Droid Sans", "Helvetica Neue",
    sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}
`;
```

Collapse

File: src/App.js

JSX

```jsx
import { CustomButton } from "./styledComponents";
import { GlobalStyle } from "./styledComponents";
```

```

const App = () => (
  <>
    <GlobalStyle />
    <CustomButton type="button">Click</CustomButton>
    <CustomButton type="button" outline>Click</CustomButton>
  </>
);

export default App;
```

- To know more, you can go through this link.