5. The "as" prop

- You can pass the **as** prop to your styled component with the value of your preferred element.
- If you want to keep all the styling you've applied to a component but just switch out what's being ultimately rendered (be it a different HTML tag or a different custom component), you can use the **"as"** prop to do this at runtime.

Example:

File:src/App.js

JSX

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

const App = () => (
  <>
    <CustomButton outline={false}>Click</CustomButton>
    <CustomButton as="a" href="#" outline={true}>Click</CustomButton>
  </>
);

export default App;
```

Collapse

File:src/styledComponents.js

JSX

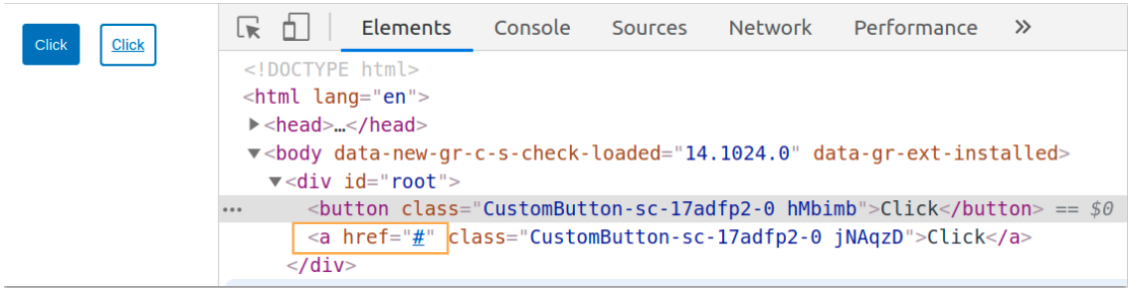
```jsx
import styled from "styled-components";

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")};
```

```
11  `;
```

Collapse

- Consider the above code snippet, here the **OutlineButton** styled component is rendered as a button element, but you would prefer an **anchor** to a button for **OutlineButton**, you can pass the `as` a prop to your styled component with the value of your preferred element.



- This sort of thing is very useful in use cases like a navigation bar where some of the items should be links and some just buttons, but all be styled the same way.