React Hydration Error

Why This Error Occurred

While rendering your application, there was a difference between the React tree that was pre-rendered (SSR/SSG) and the React tree that rendered during the first render in the Browser. The first render is called Hydration which is a feature of React.

This can cause the React tree to be out of sync with the DOM and result in unexpected content/attributes being present.

Possible Ways to Fix It

In general this issue is caused by using a specific library or application code that is relying on something that could differ between pre-rendering and the browser. An example of this is using `window` in a component's rendering.

An example:

```
function MyComponent() {
  // This condition depends on `window`. During the first render of the browser the `color` variable will be different
  const color = typeof window !== 'undefined' ? 'red' : 'blue'
  // As color is passed as a prop there is a mismatch between what was rendered server-side vs what was rendered in the first render
  return <h1 className={`title ${color}`}>Hello World!</h1>
}
```

How to fix it:

```
// In order to prevent the first render from being different you can use `useEffect` which is only executed in the browser and is executed during hydration
import { useEffect, useState } from 'react'
function MyComponent() {
  // The default value is 'blue', it will be used during pre-rendering and the first render in the browser (hydration)
  const [color, setColor] = useState('blue')
  // During hydration `useEffect` is called. `window` is available in `useEffect`. In this case because we know we're in the browser checking for window is not needed. If you need to read something from window that is fine.
  // By calling `setColor` in `useEffect` a render is triggered after hydrating, this causes the "browser specific" value to be available. In this case 'red'.
  useEffect(() => setColor('red'), [])
  // As color is a state passed as a prop there is no mismatch between what was rendered server-side vs what was rendered in the first render. After useEffect runs the color is set to 'red'
  return <h1 className={`title ${color}`}>Hello World!</h1>
}
```

Common causes with css-in-js libraries:

- When using Styled Components / Emotion

    - When css-in-js libraries are not set up for pre-rendering (SSR/SSG) it will often lead to a hydration mismatch. In general this means the application has to follow the Next.js example for the library. For example if `pages/_document` is missing and the Babel plugin is not added.
        - Possible fix for Styled Components:
            - If you want to leverage Styled Components with SWC in Next.js 12.1+ you need to add it to your Next.js config under compiler options: https://github.com/vercel/next.js/tree/canary/examples/with-styled-components
            - If you want to use Styled Components with Babel, you need `pages/_document` and the Babel plugin: https://github.com/vercel/next.js/tree/canary/examples/with-styled-components-babel
        - Possible fix for Emotion: https://github.com/vercel/next.js/tree/canary/examples/with-emotion
- When using other css-in-js libraries
    - Similar to Styled Components / Emotion css-in-js libraries generally need configuration specified in their examples in the examples directory

Useful Links

- React Hydration Documentation
- Josh Comeau's article on React Hydration