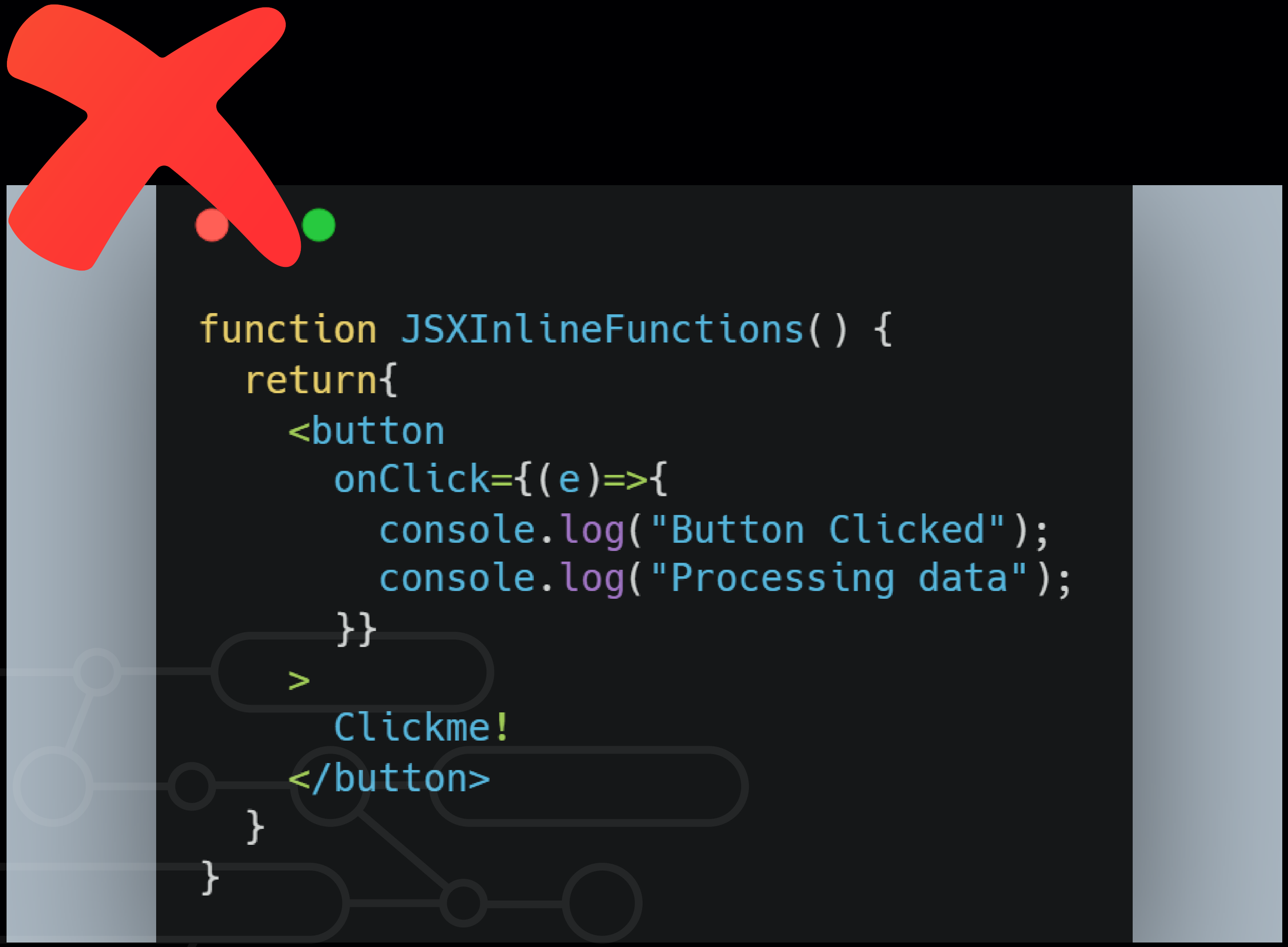
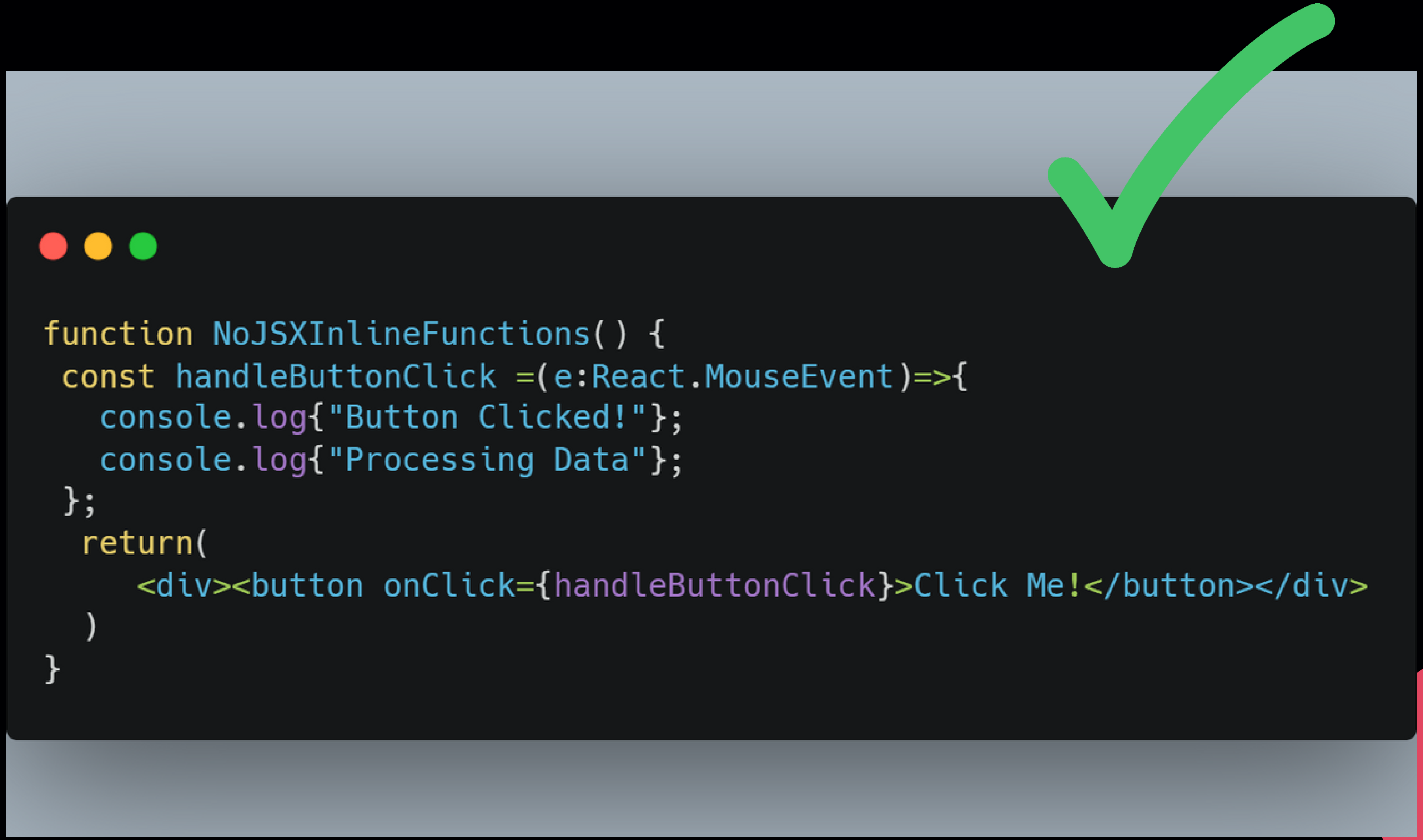
REACT BEST PRACTICES

AVOID INLINE EVENT LISTENERS



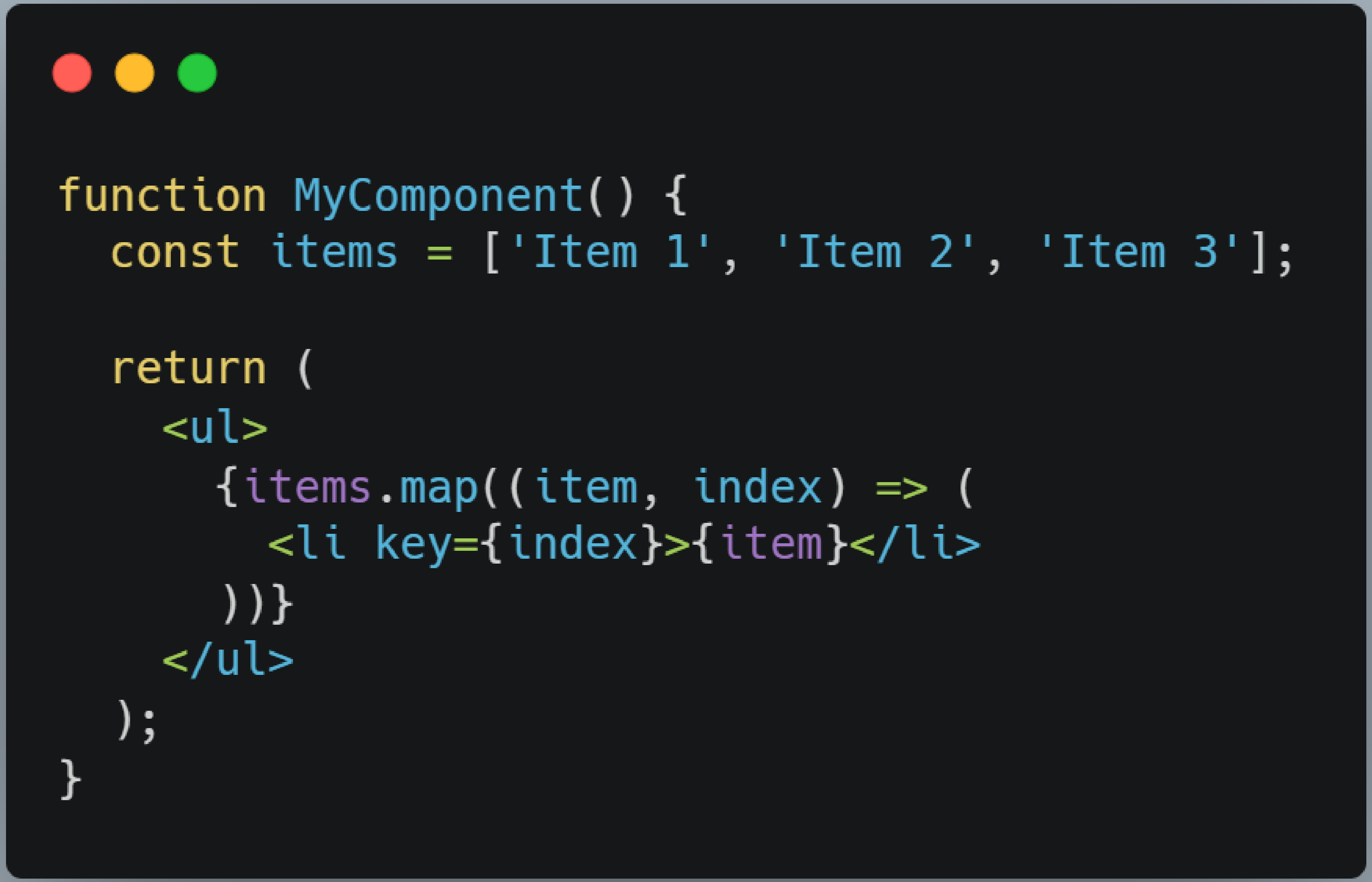
```
function JSXInlineFunctions() {
  return{
    <button
      onClick={(e)=>{
        console.log("Button Clicked");
        console.log("Processing data");
      }}
    >
      Clickme!
    </button>
  }
}
```



```
function NoJSXInlineFunctions() {
 const handleButtonClick =(e:React.MouseEvent)=>{
   console.log{"Button Clicked!"};
   console.log{"Processing Data"};
 };
  return(
     <div><button onClick={handleButtonClick}>Click Me!</button></div>
   )
}
```

USE KEYS IN LISTS

When rendering a list of items in React, it’s important to include a unique “key” prop for each item. This allows React to optimize the rendering and update only the necessary components.

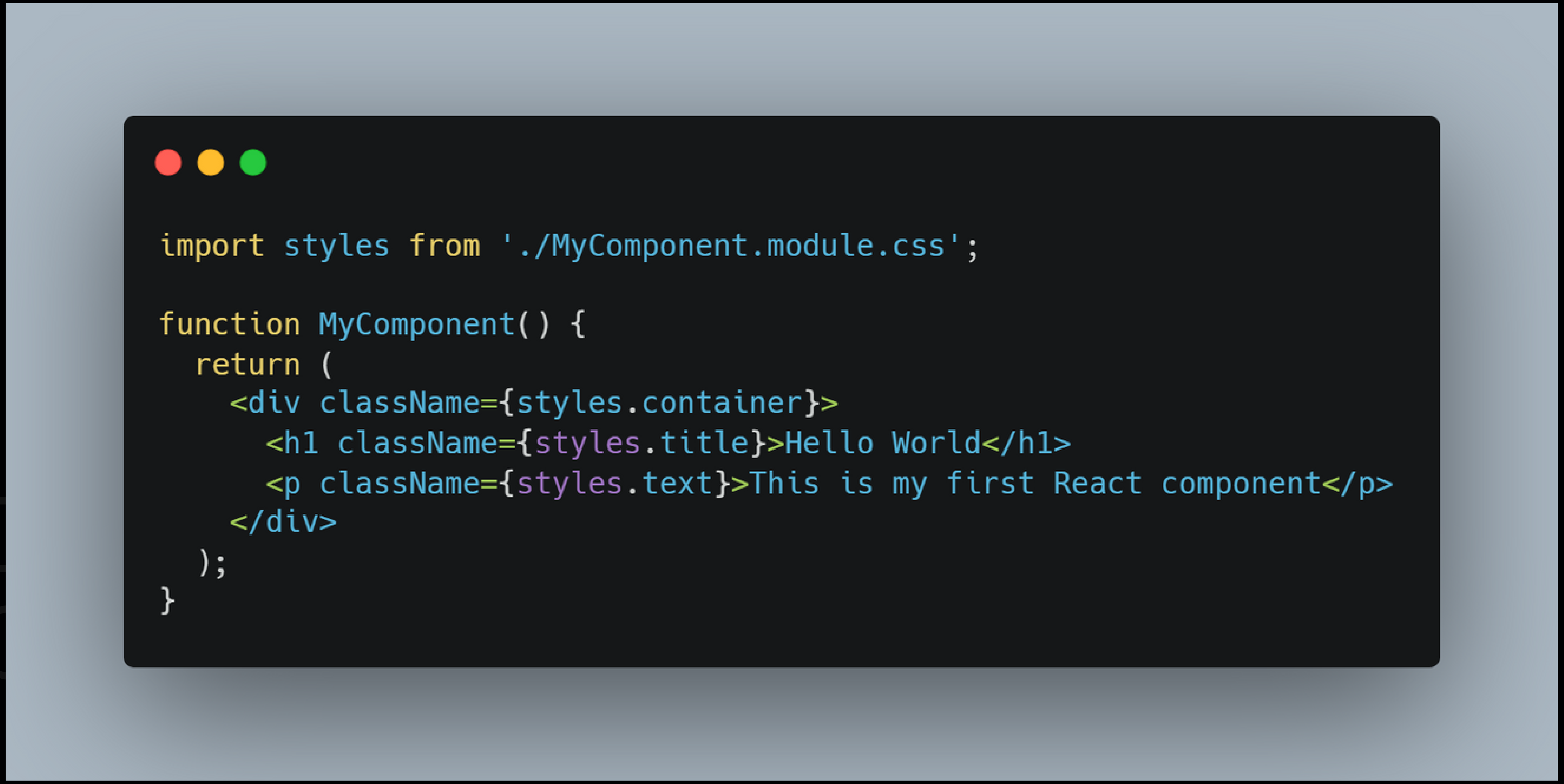


```jsx
function MyComponent() {
  const items = ['Item 1', 'Item 2', 'Item 3'];

  return (
    <ul>
      {items.map((item, index) => (
        <li key={index}>{item}</li>
      ))}
    </ul>
  );
}
```

USING MODULE.CSS

CSS Modules are a way to encapsulate CSS styles and prevent them from leaking into other components. They allow you to use class names that are unique to each component and avoid naming collisions.

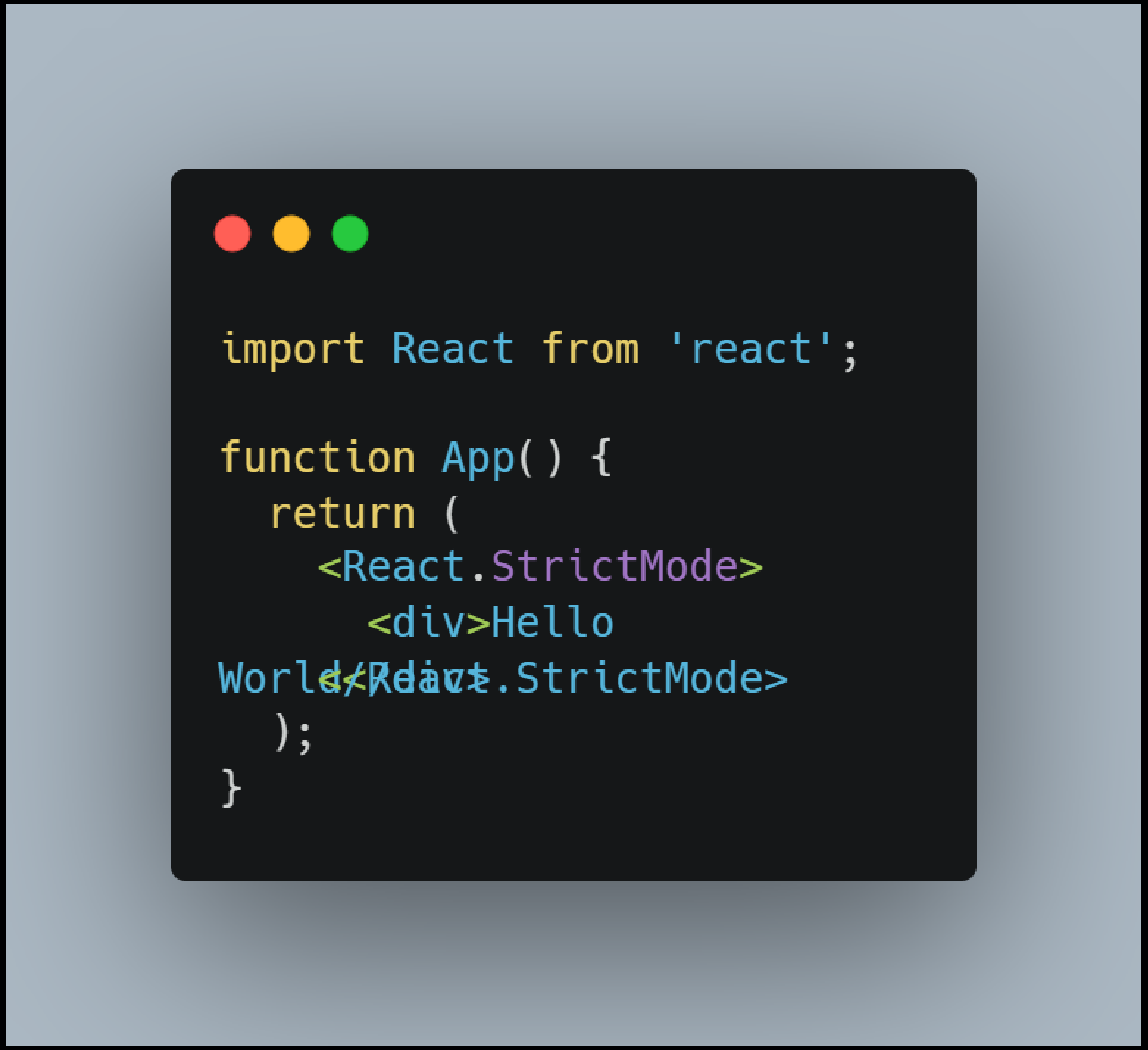


```
import styles from './MyComponent.module.css';

function MyComponent() {
  return (
    <div className={styles.container}>
      <h1 className={styles.title}>Hello World</h1>
      <p className={styles.text}>This is my first React component</p>
    </div>
  );
}
```

USE REACT.STRICTMODE FOR DEBUGGING

REACT.STRICTMODE IS A COMPONENT THAT CAN BE USED TO HIGHLIGHT POTENTIAL PROBLEMS IN YOUR CODE DURING DEVELOPMENT. IT ENABLES ADDITIONAL CHECKS AND WARNINGS IN THE CONSOLE AND HELPS YOU FIND COMMON ISSUES SUCH AS UNINTENDED SIDE EFFECTS OR DEPRECATED FEATURES.

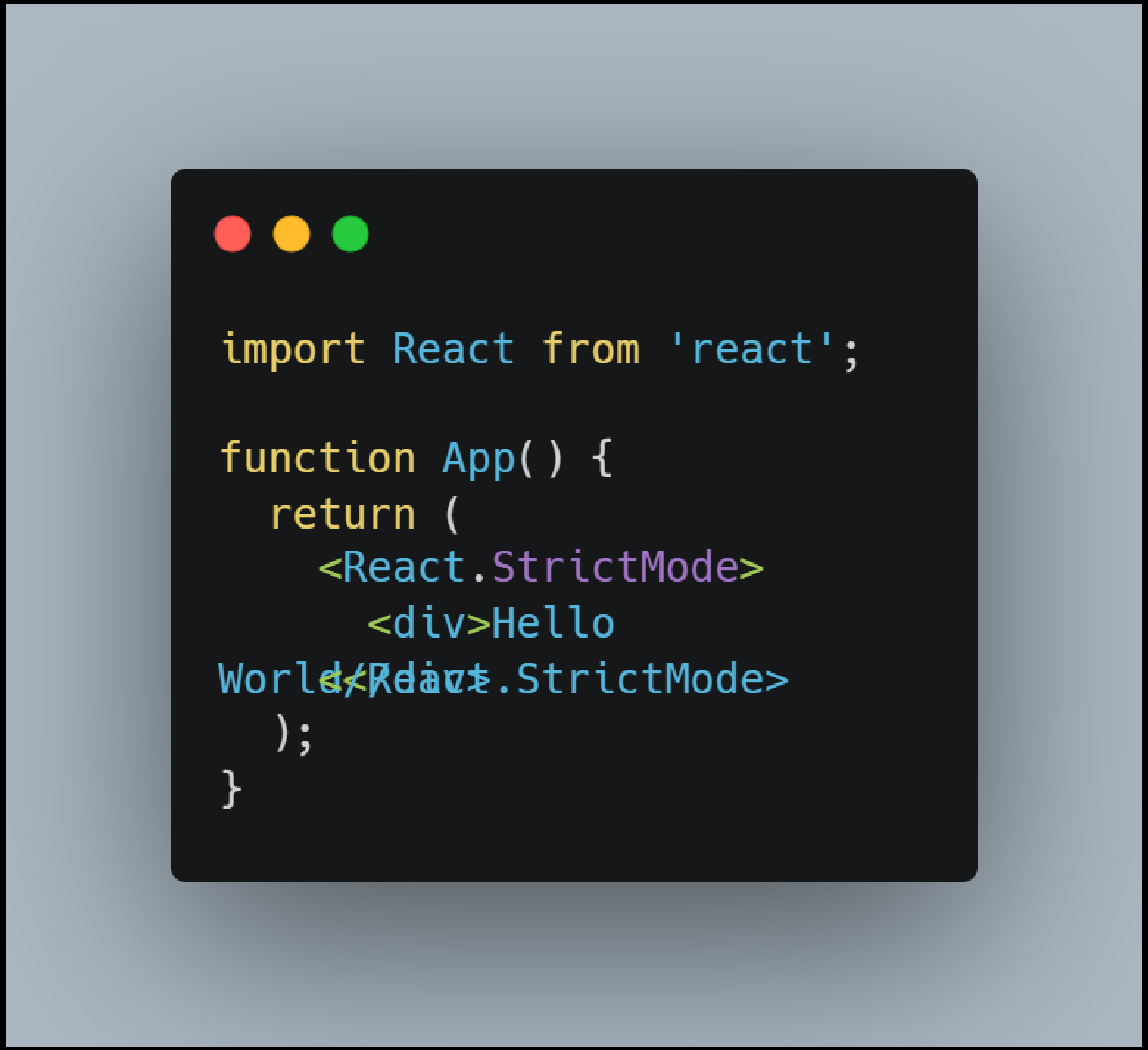


```
import React from 'react';

function App() {
  return (
    <React.StrictMode>
      <div>Hello
World</div>
    </React.StrictMode>
  );
}
```

USE REACT.STRICTMODE FOR DEBUGGING

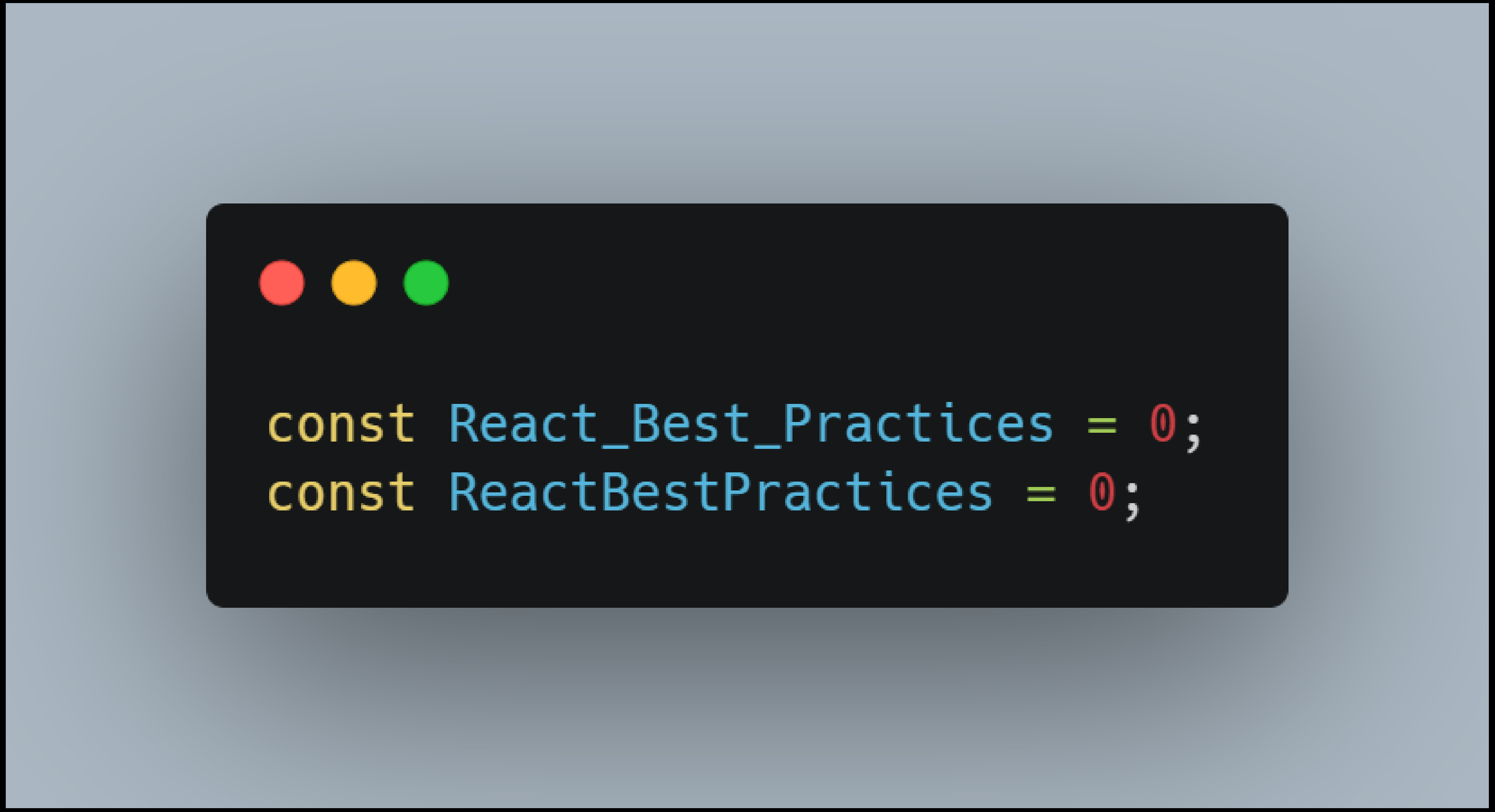
REACT.STRICTMODE IS A COMPONENT THAT CAN BE USED TO HIGHLIGHT POTENTIAL PROBLEMS IN YOUR CODE DURING DEVELOPMENT. IT ENABLES ADDITIONAL CHECKS AND WARNINGS IN THE CONSOLE AND HELPS YOU FIND COMMON ISSUES SUCH AS UNINTENDED SIDE EFFECTS OR DEPRECATED FEATURES.



```
import React from 'react';

function App() {
  return (
    <React.StrictMode>
      <div>Hello
World</div></React.StrictMode>
  );
}
```

NAMING CONVENTION



```
const React_Best_Practices = 0;
const ReactBestPractices = 0;
```

DRY YOUR REACT CODE

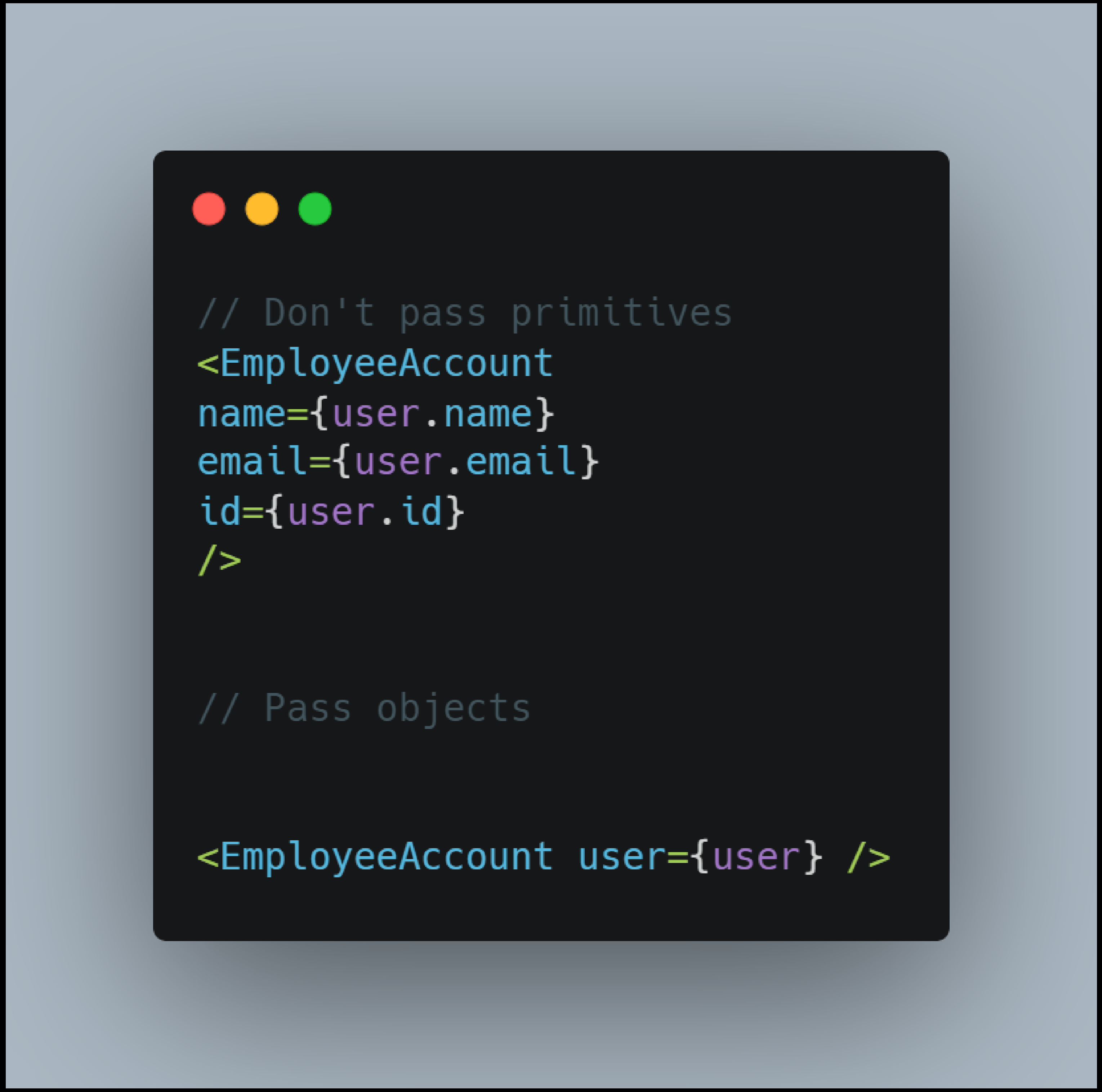
DRY STANDS FOR DON'T REPEAT YOURSELF. AS YOU ALL MAY HAVE GUESSED, HERE THIS MEANS - DO NOT REPEAT YOUR CODE. IT SIMPLY AIMS TO AVOID THE OCCURRENCE OF REPEATED CODE.
REACT BEST PRACTICES OFFER YOU TO WRITE ACCURATE AND CONCISE CODE, HENCE MAKING IT FEASIBLE AND SIMPLE.

USE REACT DEVELOPER TOOLS

REACT DEVELOPER TOOLS COME IN HANDY WHILE DEVELOPING A REACT APP. IT COMPREHENDS THE STATE, CHILDREN, PROPS, AND COMPONENT HIERARCHY. IT ALSO HELPS IN DEBUGGING THE CODE. DEVELOPERS MAY EASILY CONSTRUCT INTERACTIVE UIS WITH THE AID OF REACT DEVELOPER TOOLS.

PREFER PASSING OBJECTS

MOST PEOPLE PASS SET OF PRIMITIVE VALUES BUT TO RESTRICT THE NUMBER OF PROPS BEING PASSED, YOU MUST PREFER PASSING AN OBJECT. FOR INSTANCE, YOU CAN COMBINE THE DETAILS OF AN EMPLOYEE RATHER THAN PASSING EACH ONE INDIVIDUALLY.



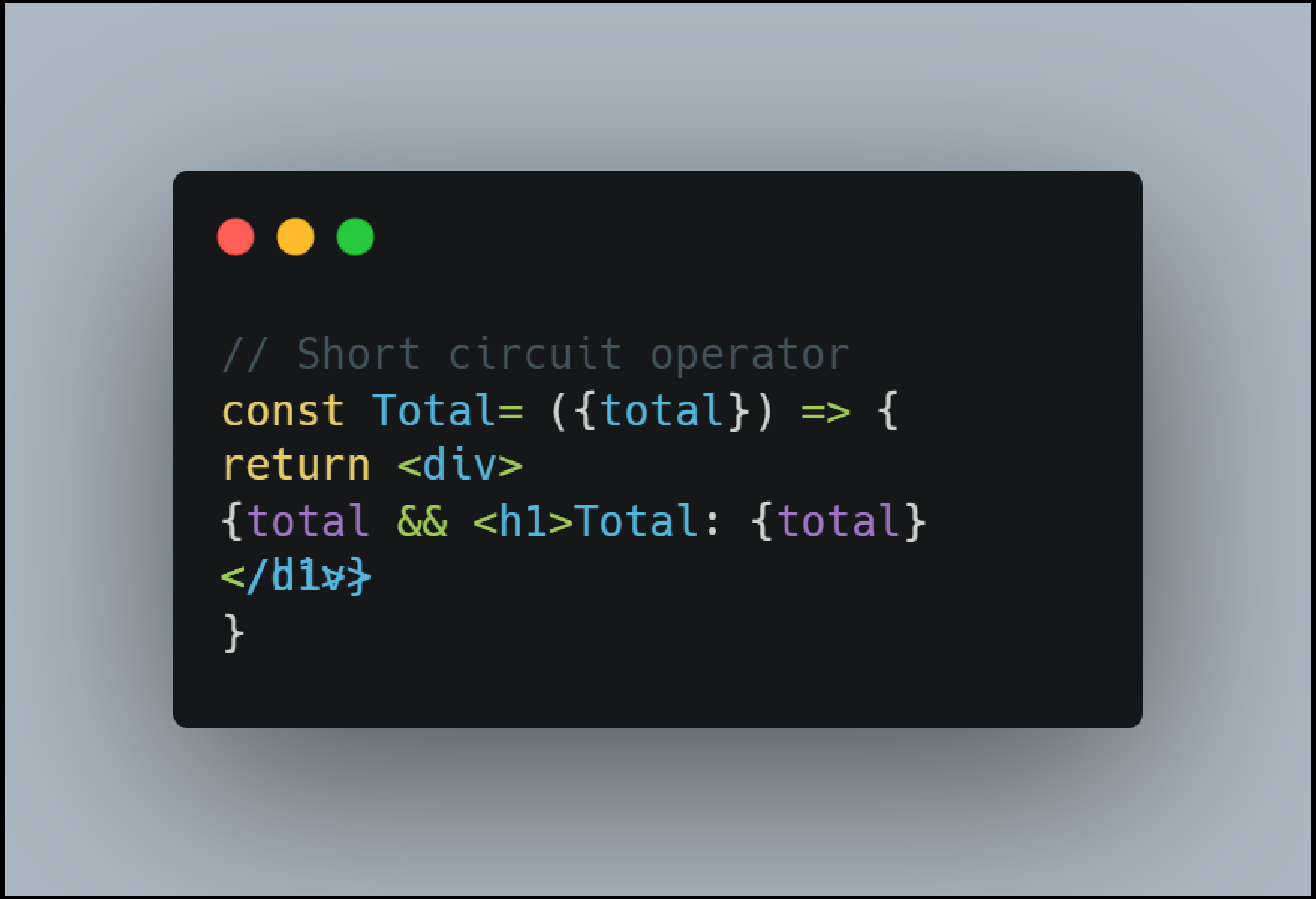
```
// Don't pass primitives
<EmployeeAccount
name={user.name}
email={user.email}
id={user.id}
/>


// Pass objects


<EmployeeAccount user={user} />
```

CONDITIONAL RENDERING PRACTICES

NOW, THERE ARE A LOT OF WAYS TO PERFORM CONDITIONAL RENDERING. HOWEVER, USING SHORT CIRCUIT OPERATORS ARE THE MOST EASY AND SIMPLE TO USE.



```
// Short circuit operator
const Total= ({total}) => {
return <div>
{total && <h1>Total: {total}
</div>}
}
```

LAZY LOADING IMPLEMENTATION

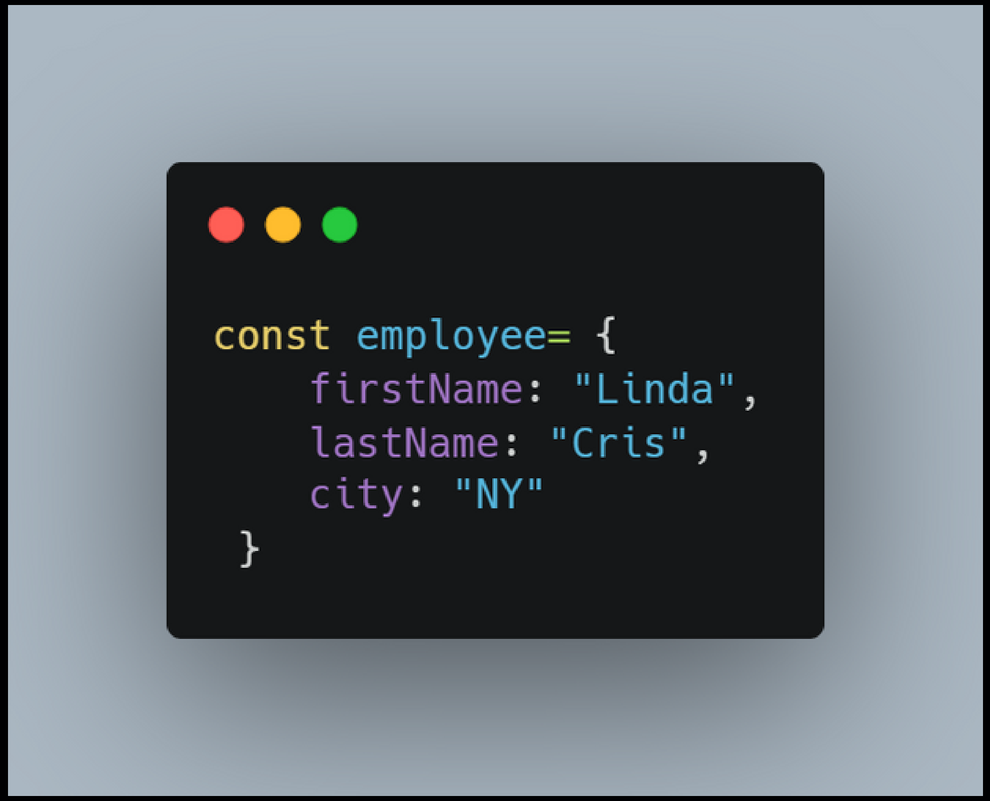
IN SIMPLE WORDS, LAZY LOADING IS A DESIGN PATTERN. YOU CAN REDUCE THE INITIAL LOAD TIME BY ALLOWING PORTIONS OF YOUR APPLICATION TO LOAD ONLY WHEN NECESSARY. FOR INSTANCE, YOU MAY INITIALLY LOAD THE PARTS AND MODULES NEEDED FOR USER REGISTRATION AND LOGIN. THE REMAINING FUNCTIONAL OR CLASS COMPONENTS CAN THEN BE LOADED BASED ON USER NAVIGATION.

HANDLING ERRORS AND DEBUGGING IN A REACTJS APPLICATION

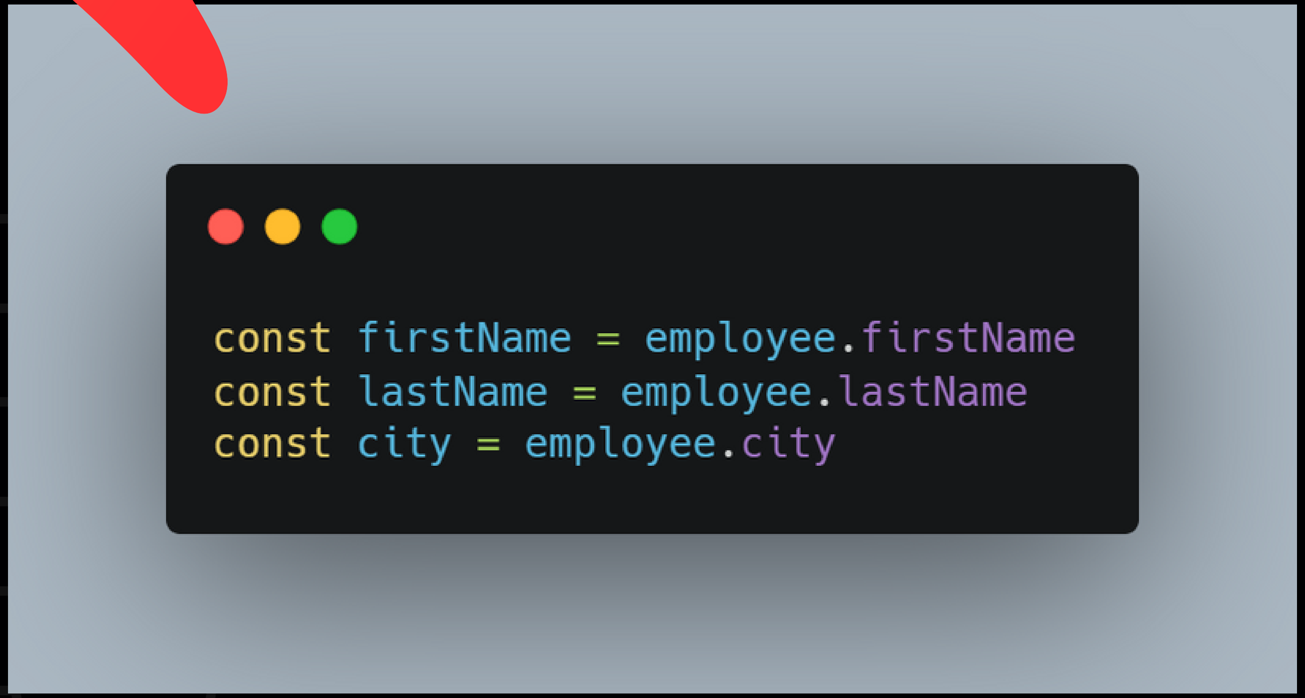
HERE ARE SOME PRACTICES THAT CAN BE USED BY DEVELOPERS FOR HANDLING AND DEBUGGING ERRORS -

- ERROR BOUNDARIES FOR CLASS COMPONENTS
- TRY-CATCH FOR CATCH BEYOND BOUNDARIES
- REACT ERROR BOUNDARY LIBRARY

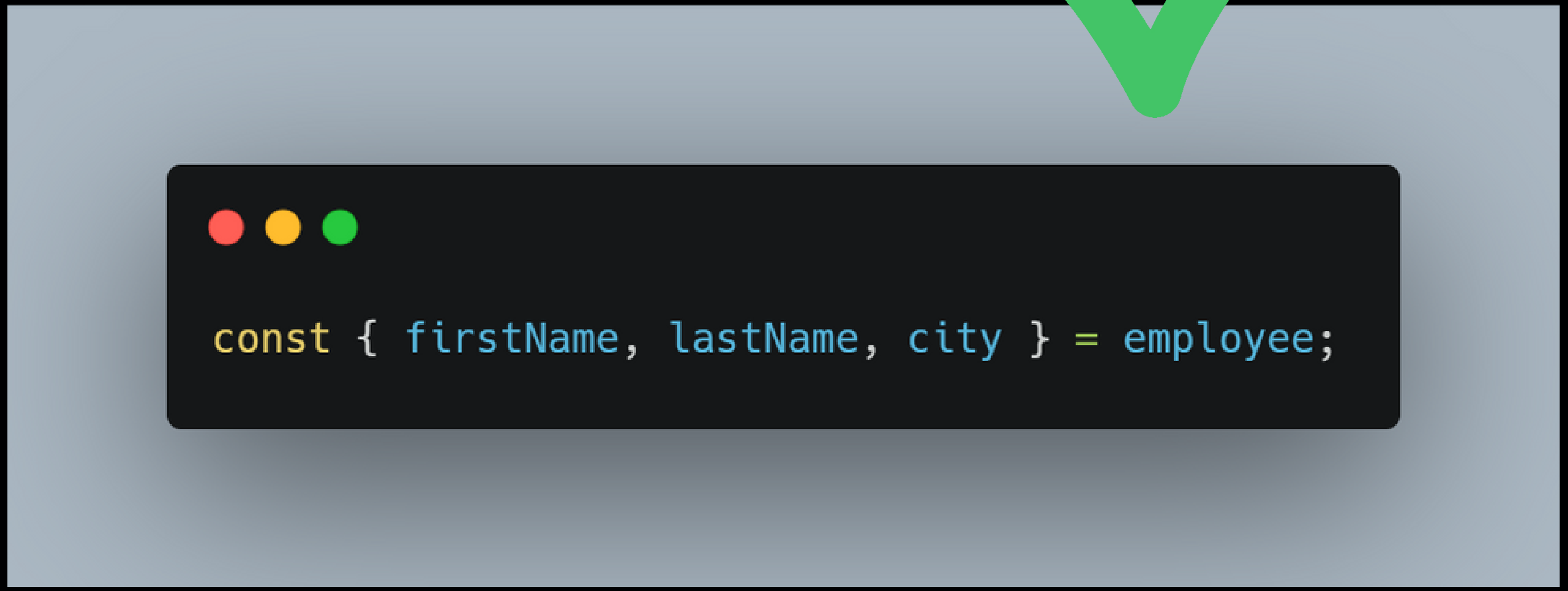
USE DESTRUCTURING TO GET PROPS



```
const employee= {
    firstName: "Linda",
    lastName: "Cris",
    city: "NY"
 }
```



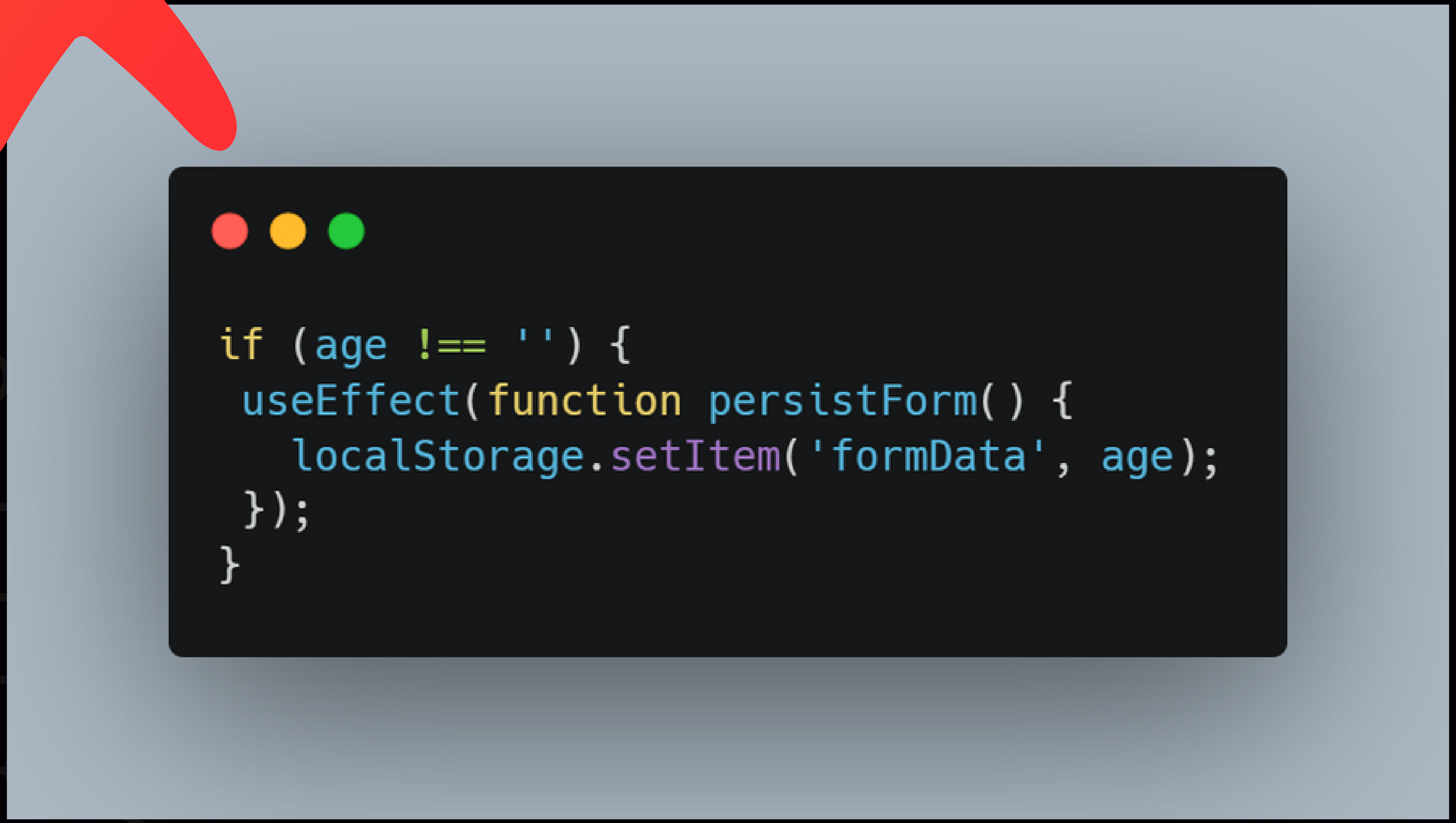
```
const firstName = employee.firstName
const lastName = employee.lastName
const city = employee.city
```



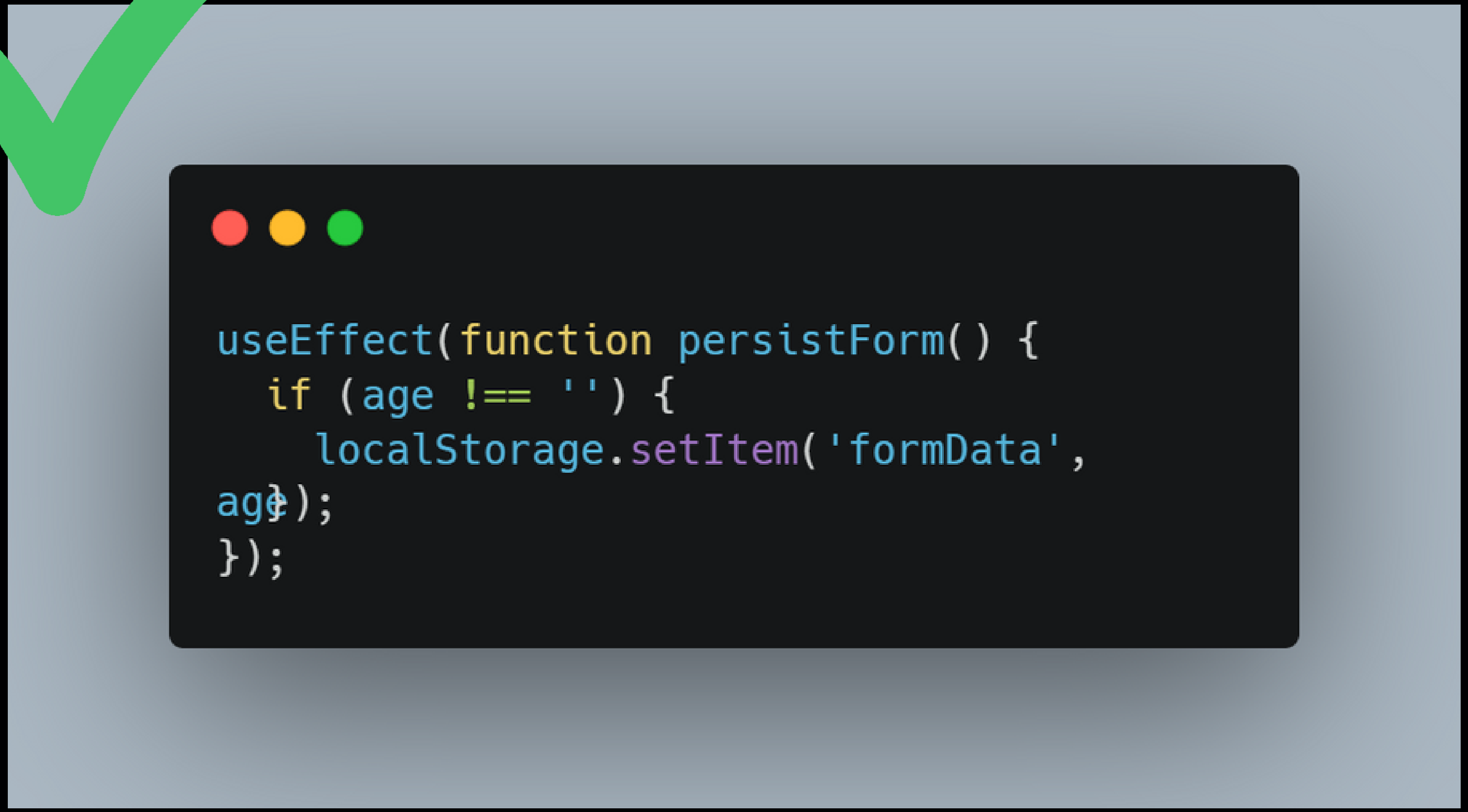
```
const { firstName, lastName, city } = employee;
```

CALL HOOKS AT THE TOP LEVEL

DO NOT CALL HOOKS WITHIN LOOPS, CONDITIONS, OR FUNCTIONS THAT ARE ALREADY NESTED. INSTEAD, MAKE SURE TO USE HOOKS AT YOUR REACT FUNCTION'S TOP LAYER.



```
if (age !== '') {
 useEffect(function persistForm() {
   localStorage.setItem('formData', age);
 });
}
```



```
useEffect(function persistForm() {
  if (age !== '') {
    localStorage.setItem('formData',
age});
});
```

ADD DEPENDENCIES WISELY

- DEPENDENCIES ARE EXTERNAL LIBRARIES OR PACKAGES THAT PROVIDE SPECIFIC FUNCTIONALITIES AND ASSIST IN IMPROVING CODE STRUCTURE, REDUCING DEVELOPMENT TIME. THEY CAN SIGNIFICANTLY STREAMLINE THE DEVELOPMENT PROCESS. BUT IF YOU ADD DEPENDENCIES TOO MANY OR OUTDATED DEPENDENCIES, YOUR APP'S PERFORMANCE MIGHT TAKE A HIT. THEREFORE, IT'S ALL ABOUT FINDING A BALANCE - USE WHAT YOU NEED TO KEEP THINGS RUNNING SMOOTHLY.

- YOU CAN USE NPM-CHECK PACKAGE TO CHECK THE HEALTH OF ALL THE PACKAGES USED IN YOUR PROJECT; IT WILL CHECK OUTDATED DEPENDENCIES AND FLAG ANY OUTDATED OR UNUSED ONES.

CONTINUOUS INTEGRATION AND DEPLOYMENT

CI/CD, OR CONTINUOUS INTEGRATION AND CONTINUOUS DEPLOYMENT, IS A SET OF PRACTICES THAT HELP DEVELOPERS DELIVER SOFTWARE MORE RELIABLY AND EFFICIENTLY. WHEN YOU MAKE A CHANGE TO THE CODEBASE, CI/CD PIPELINES RUN TESTS AND BUILD YOUR APP TO ENSURE THAT ALL CHECKS ARE GREEN BEFORE DEPLOYING THE LATEST UPDATES. THIS HELPS TO PREVENT BUGS AND ENSURE THAT YOUR APP IS ALWAYS UP-TO-DATE.

REFRENCES

- HTTPS://WWW.TATVASOFT.COM/BLOG/REACTJS-BEST-PRACTICES/
- HTTPS://WWW.BRILWORKS.COM/BLOG/REACT-NATIVE-BEST-PRACTICES/
- HTTPS://ENLEAR.ACADEMY/TYPES-OF-REACT-HOOKS-BEST-PRACTICES-45C275B55B1F
- HTTPS://WWW.YOURTEAMININDIA.COM/BLOG/REACT-BEST-PRACTICES
- HTTPS://YOUTU.BE/X6GJAURLRRY?SI=2HWSB9--U66PR8VZ
- HTTPS://YOUTU.BE/B0IZO2AHO9Y?SI=_HBD1RL7QGWSIXN1

ANY QUESTIONS YOU CAN ASK