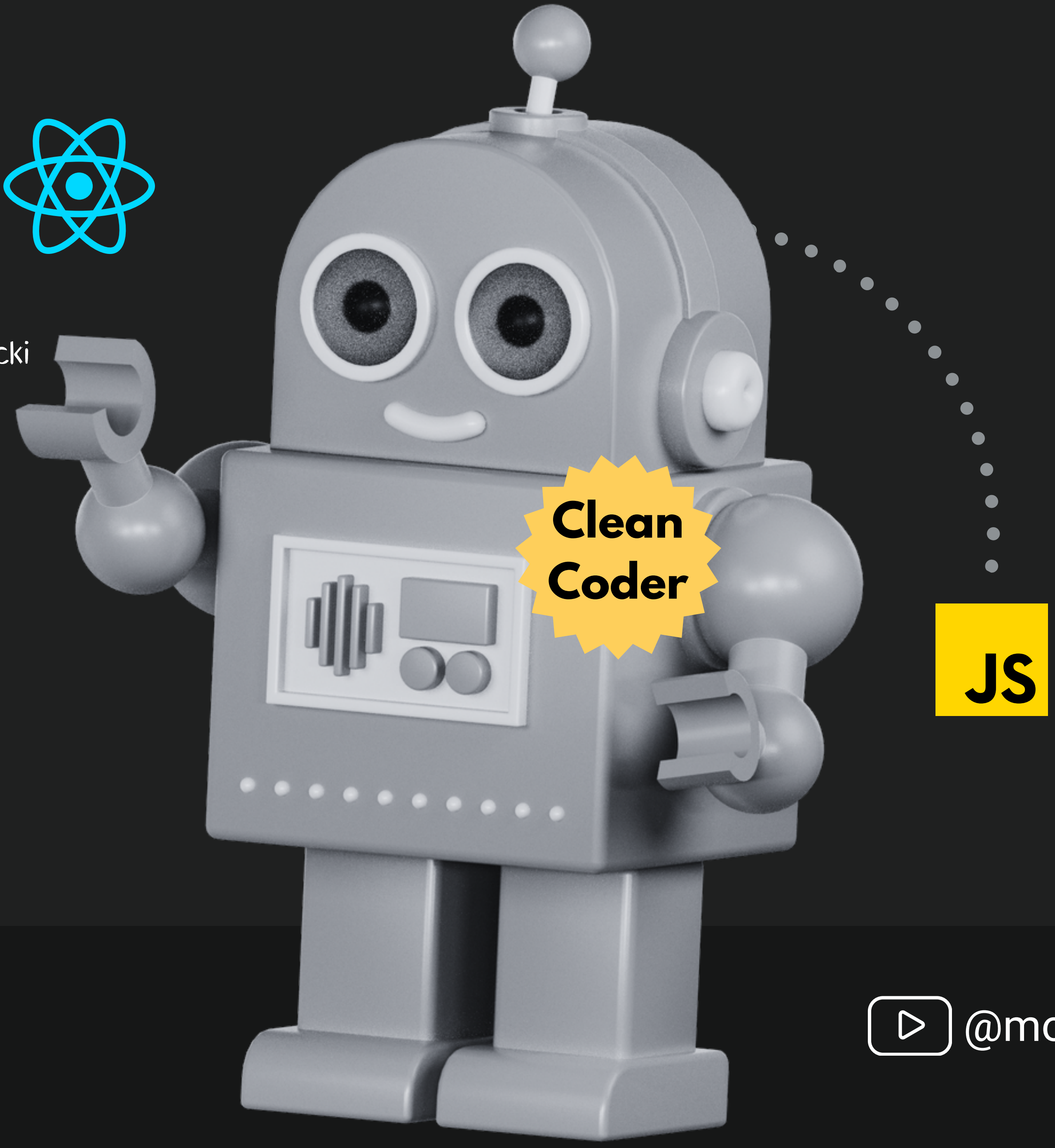
moderndev

REACT.JS

S.O.L.I.D PRINCIPLES

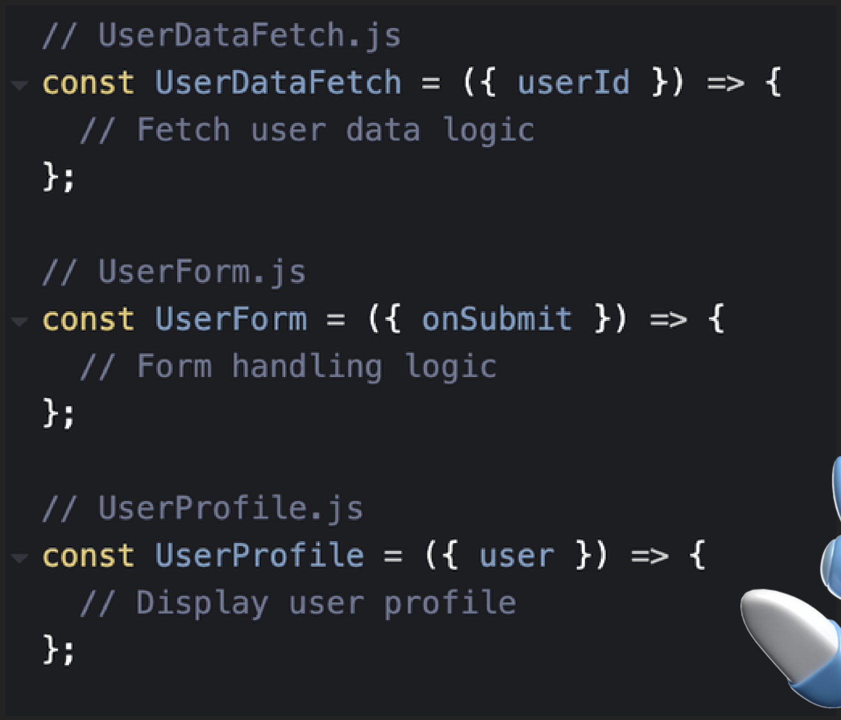


@moderndev

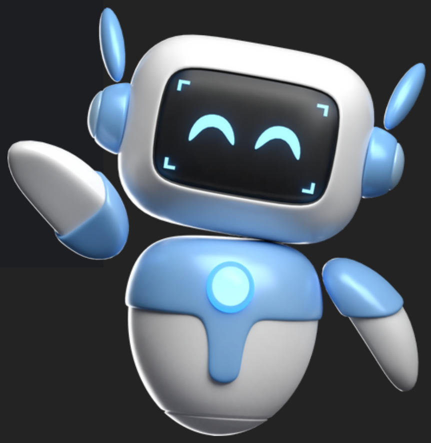
@moderndev

Single Responsibility Principle (SRP)

A component or module should have one, and only one, reason to change.



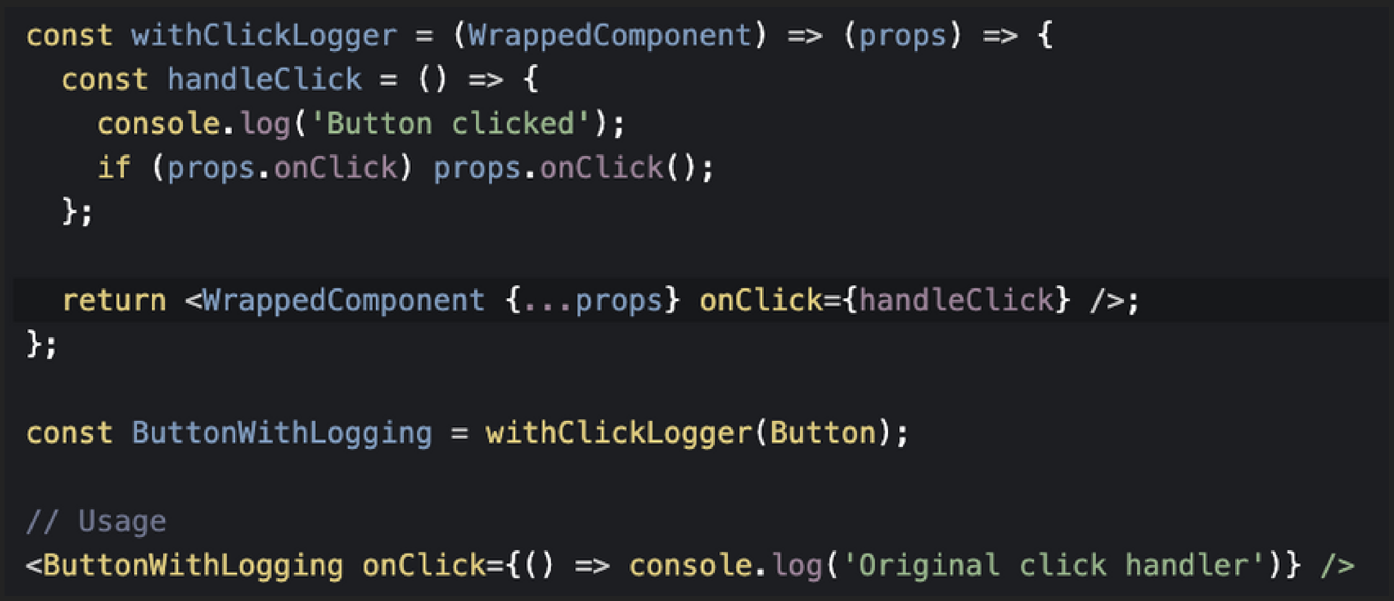
```
// UserDataFetch.js
const UserDataFetch = ({ userId }) => {
  // Fetch user data logic
};

// UserForm.js
const UserForm = ({ onSubmit }) => {
  // Form handling logic
};

// UserProfile.js
const UserProfile = ({ user }) => {
  // Display user profile
};
```



This means it should have only one job or responsibility.

Open/Closed Principle (OCP)

Software entities should be open for extension but closed for modification.

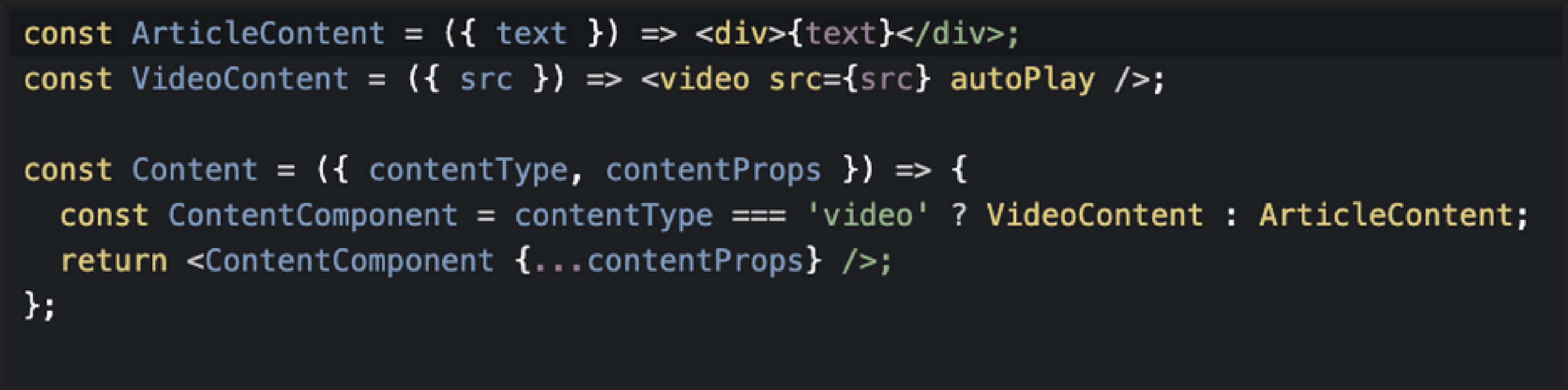


```
const withClickLogger = (WrappedComponent) => (props) => {
  const handleClick = () => {
    console.log('Button clicked');
    if (props.onClick) props.onClick();
  };

  return <WrappedComponent {...props} onClick={handleClick} />;
};

const ButtonWithLogging = withClickLogger(Button);

// Usage
<ButtonWithLogging onClick={() => console.log('Original click handler')} />
```

This means you should be able to add new functionality without changing existing code.

Liskov Substitution Principle (LSP)

Objects in a program should be replaceable with instances of their subtypes without altering the correctness of the program.



```
const ArticleContent = ({ text }) => <div>{text}</div>;
const VideoContent = ({ src }) => <video src={src} autoPlay />;

const Content = ({ contentType, contentProps }) => {
  const ContentComponent = contentType === 'video' ? VideoContent : ArticleContent;
  return <ContentComponent {...contentProps} />;
};
```

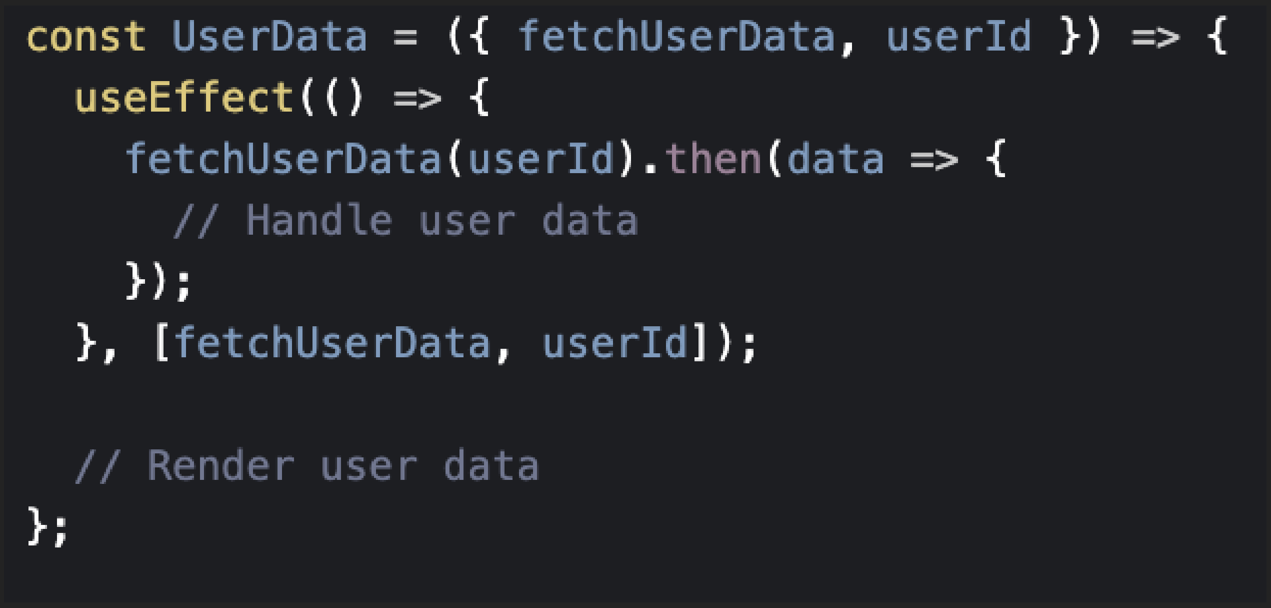
Interface Segregation Principle (ISP)

No client should be forced to depend on methods it does not use.

```
const UserProfile = ({ name, age }) => {
  // Use only name and age, ignoring other props
};

UserProfile.propTypes = {
  name: PropTypes.string.isRequired,
  age: PropTypes.number.isRequired,
};
```

Dependency Inversion Principle (DIP)

High-level modules should not depend on low-level modules. Both should depend on abstractions.



```
const UserData = ({ fetchUserData, userId }) => {
  useEffect(() => {
    fetchUserData(userId).then(data => {
      // Handle user data
    });
  }, [fetchUserData, userId]);

  // Render user data
};
```

Furthermore, abstractions should not depend on details. Details should depend on abstractions.

SOON

Do you want to build your own

AI ASSISTANT

Which covers the own custom API and AI integrations?

Make sure you follow my content and subscribe the newsletter!

Thank you!

If you want to become a modern developer, subscribe to my NEWSLETTER

newsletter.moderndev.io

Link in bio