before Hooks

Complex Component

```jsx
class Users extends React.Component {
    componentDidMount() {
        const users = fetchUsers()
        this.setState({
            users,
        })
        // redux, Logging, auth, etc...
    }

    render() {
        // some of Logic
        // that needs to be used in presentation
        return (
            // render users
        );
    }
}
```

class

Complex component

Logika
Prezentacja
Klasa

- ✗ rozdzielenie koncepcji (prezentacja/logika)
- ✗ reużywalna prezentacja
- ✗ reużywalna logika
- ✗ rozszerzalność
- ✗ klasy
- ✗ imperatywne

Container and Presentational Component

```jsx
const UsersPresentation = ({ users }) => {
    // render users
)

class UsersContainer extends React.Component {
    // fetch users and complex Logic

    render() {
        // complex logic
        return (
            <UsersPresentation users={this.state.users} />
        );
    }
}
```

class

Container

function

Presentational

Logika
Prezentacja
Klasa

- ✓ rozdzielenie koncepcji (prezentacja/logika)
- ✓ reużywalna prezentacja
- ✗ reużywalna logika
- ✗ rozszerzalność
- ✗ klasy
- ✗ imperatywne

Higher Order Components (HOC)

```jsx
const UsersPresentation = ({ users }) =>  {
    // render users
)

const withFetchUsersHOC = (UsersPresentation) => {
    return class UsersContainer extends React.Component {
        // fetch users and complex Logic

        render() {
            // complex logic
            return (
                <UsersPresentation users={this.state.users} />
            );
        }
    }
}

// usage
const component = withFetchedUsersHOC(UsersPresentation)
```

class

High Order Component

function

Presentational

Logika
Prezentacja
Klasa

- ✓ rozdzielenie koncepcji (prezentacja/logika)
- ✓ reużywalna prezentacja
- ✓ reużywalna logika
- ✓ rozszerzalność
- ✗ klasy
- ✓ deklaratywne

```jsx
// declarative approach

const component = compose(
    withFetchUsersHOC,
    withReduxHOC,
    withLoggingHOC,
    withAuthHOC,
)(UsersPresentation)
```

after Hooks

Hook

function

Component

function

Hook

Logika
Prezentacja

- ✓ rozdzielenie koncepcji (prezentacja/logika)
- ✗ reużywalna prezentacja
- ✓ reużywalna logika
- ✗ rozszerzalność
- ✓ funkcje
- ✗ imperatywne

```jsx
const useFetchUsers = () => {
    const [users, setUsers] = useState([])
    useEffect(() => {
        // fetch users and complex Logic
        setUsers(fetchedUsers)
    }, [])

    // complex logic

    return users
}

const Users = () => {
    const users = useFetchUsers()

    return (
        // render users
    )
}
```

Hook and Container and Presentational...

function

Component

function

Hook

function

Presentational

Logika
Prezentacja

- ✓ rozdzielenie koncepcji (prezentacja/logika)
- ✓ reużywalna prezentacja
- ✓ reużywalna logika
- ✗ rozszerzalność
- ✓ funkcje
- ✗ imperatywne

```jsx
const useFetchUsers = () => {
    // fetch users and complex Logic

    return users
}

const UsersContainer = () => {
    const users = useFetchUsers()

    return (
        <UsersPresentation users={users} />
    )
}

const UsersPresentation = () => (
    // render users
)
```

Hook & Higher Order Component (HOC)

function

High Order Component

function

Hook

function

Presentational

Logika
Prezentacja

- ✓ rozdzielenie koncepcji (prezentacja/logika)
- ✓ reużywalna prezentacja
- ✓ reużywalna logika
- ✓ rozszerzalność
- ✓ funkcje
- ✓ deklaratywne

```jsx
// how?
const useFetchUsers = () => {
    // fetch users and complex Logic

    return users
}

// what?
const withFetchUsersHOC = (Users) => {
    const users = useFetchUsers()

    return (
        <Users users={users} />
    )
}

// presentation
const UsersPresentation = () => (
    // render users
)

// usage
const component = withFetchUsersHOC(UsersPresentation)
```