

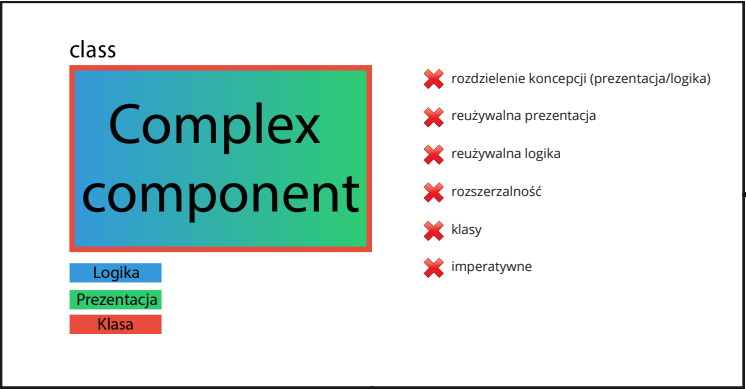
before Hooks

after Hooks

Complex Component

```
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
    );
  }
}
```

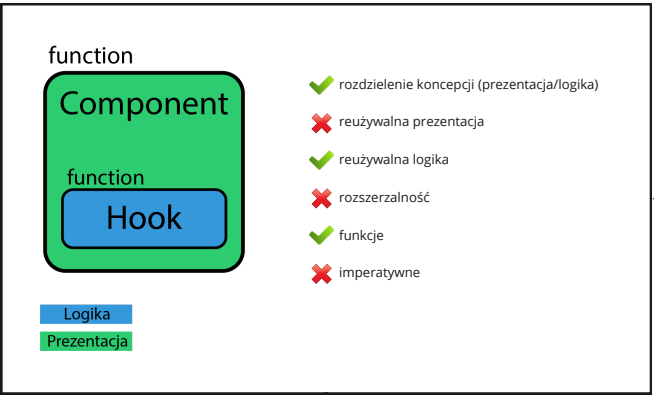


Hook

```
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
  )
}
```

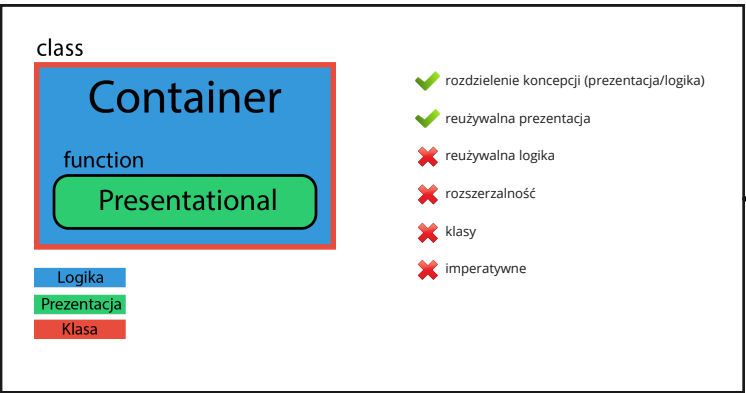


Container and Presentational Component

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

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

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



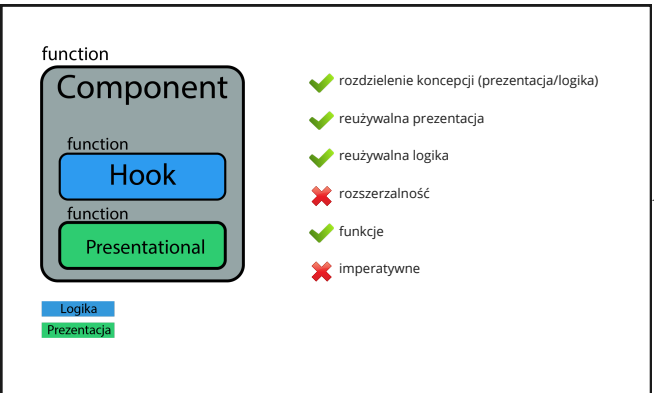
Hook and Container and Presentational...

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

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

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

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



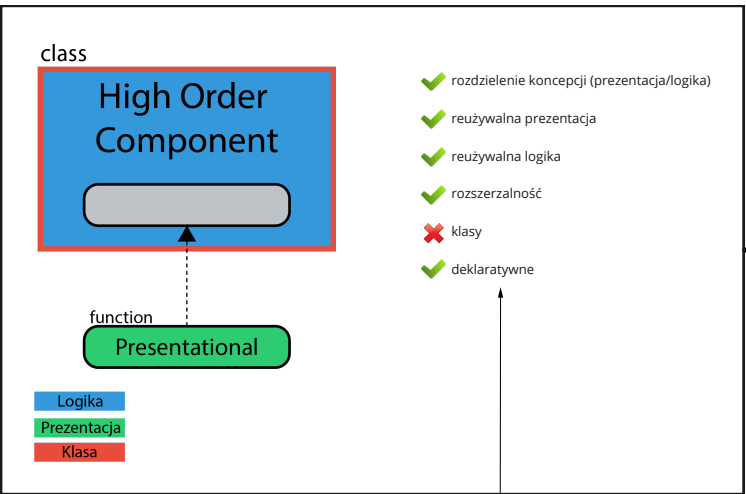
Higher Order Components (HOC)

```
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 = withFetchUsersHOC(UsersPresentation)
```



Hook & Higher Order Component (HOC)

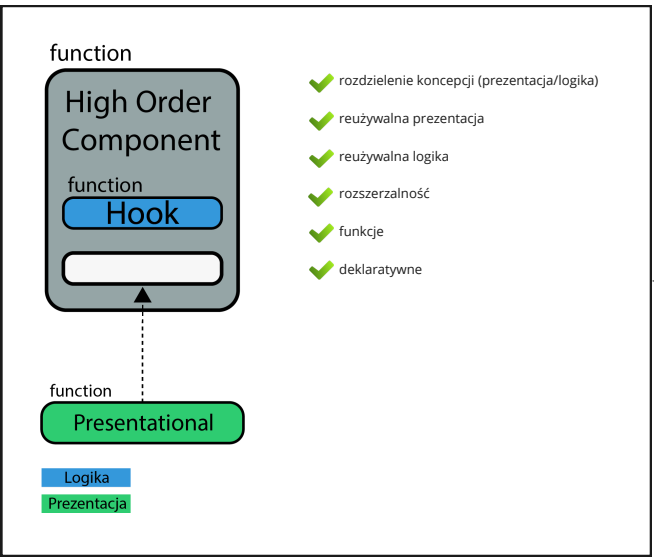
```
// how?
const useFetchUsers = () => {
  // fetch users and complex Logic
  return users
}

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

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

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

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



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