Design Patterns in React

Proxy

دیزاین پترن ها در React

Structural Patterns

به زبان ساده همراه با مثال های عملی
کاربرد دیزاین پترن ها برای برنامه نویسان React

Farid Karami
www.faridkarami.com

Structural Patterns

Proxy

الگوی Proxy یک واسطه یا نماینده برای یک شیء دیگر ارائه می دهد تا دسترسی به آن را کنترل کند. از این الگو می توان برای مدیریت یا محدود کردن دسترسی به کامپوننت ها یا مدیریت درخواست های API، بارگذاری Lazy، یا کنترل دسترسی بر اساس نقش کاربر استفاده کرد.

فرض کنید می خواهیم بر اساس نقش کاربر سطح دسترسی او برای مشاهده بخش ها و کامپوننت های برنامه را مشخص نماییم. به عنوان مثال اگر کاربر ادمین نبود نتواند بعضی از کامپوننت ها را مشاهده کند.

بخش Button

ابتدا یک Component را به ساده ترین شکل ممکن می سازیم.


```tsx
const Button: React.FC = () => {
  return <button>Admin Button</button>;
};

export default Button;
```

بخش ProxyComponent

یک Proxy به شکل زیر می سازیم که چک می کند اگر نقش کاربر برابر با admin بود Component مورد نظر را برمی گرداند و در غیر این صورت پیام عدم دسترسی برمی گرداند.

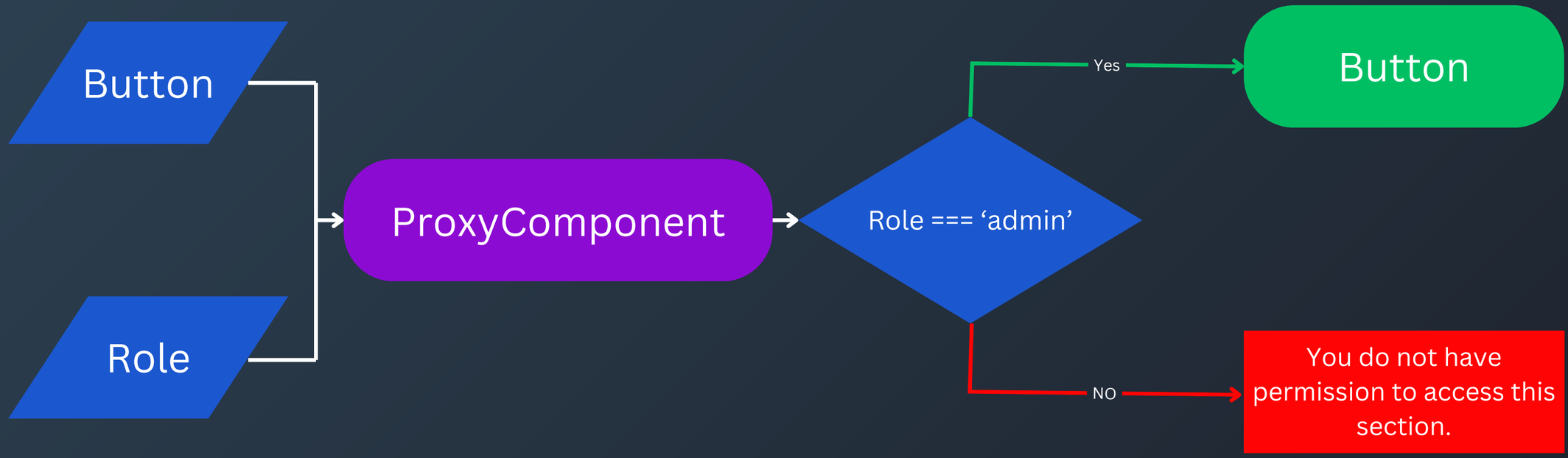

```tsx
interface ProxyProps {
  role: string;
  component: React.FC;
}

const ProxyComponent: React.FC<ProxyProps> = ({ role, component: Component }) => {
  if (role === 'admin') {
    return <Component />;
  }
  return <div>You do not have permission to access this section.</div>;
};

export default ProxyComponent;
```

نحوه کارکرد Proxy

همانطور که در فلوچارت زیر مشاهده می کنید در ProxyComponent بررسی می شود که آیا نقش کاربر admin است یا خیر و بر اساس آن تصمیم گرفته می شود که چه چیزی نمایش داده شود.



طرز استفاده از ProxyComponent

در پایان به شکل زیر از ProxyComponent استفاده می کنیم.


```jsx
import Button from "./pattern/Button";
import ProxyComponent from "./pattern/ProxyComponent";

const ProxyDesignPattern = () => {
  const userRole = 'admin'; // or user

  return (
    <div>
      <ProxyComponent role={userRole} component={Button} />
    </div>
  );
};

export default ProxyDesignPattern;
```