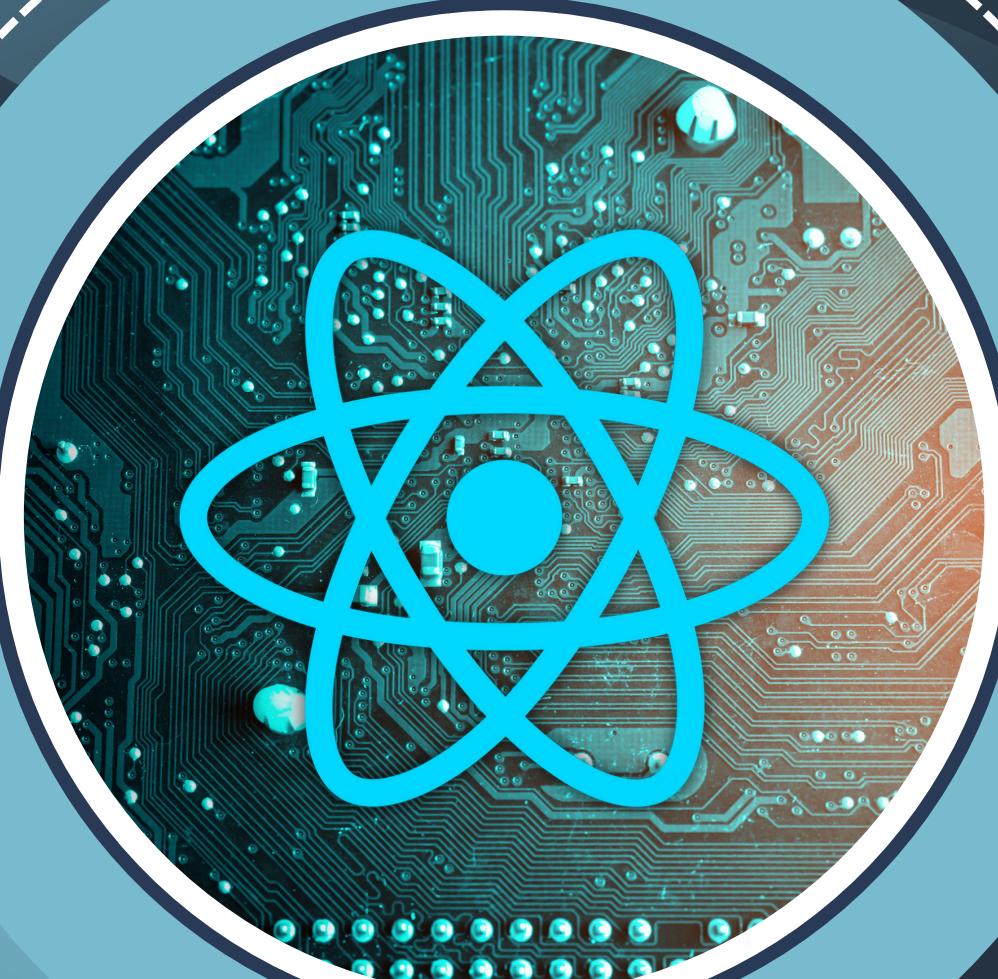


Decorator

دیزاین پترن هادر React

Structural Patterns

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



Structural Patterns

Decorator

الگوی Decorator برای افزودن قابلیت ها یا رفتارهای اضافی به کامپوننت ها، بدون تغییر در کد اصلی آن ها، استفاده می شود. در React، پیاده سازی این الگو معمولاً از طریق Higher-Order Components (HOCs) یا Render Props انجام می شود.

فرض کنید می خواهیم برای تست و دییاگ Props یک یا چند کامپوننت را در Console نمایش دهیم و هر موقع که لازم بود این امکان را با یک تغییر کوچک غیر فعال کنیم. برای انجام این کار یک Decorator به نام `withLogger` می سازیم.

بخش Button

ابتدا Props و Button را به ساده ترین شکل ممکن به صورت زیر پیاده سازی می کنیم.
یک Decorator به نام withLogger را مشاهده می کنید که در ادامه آن را می سازیم.



Button

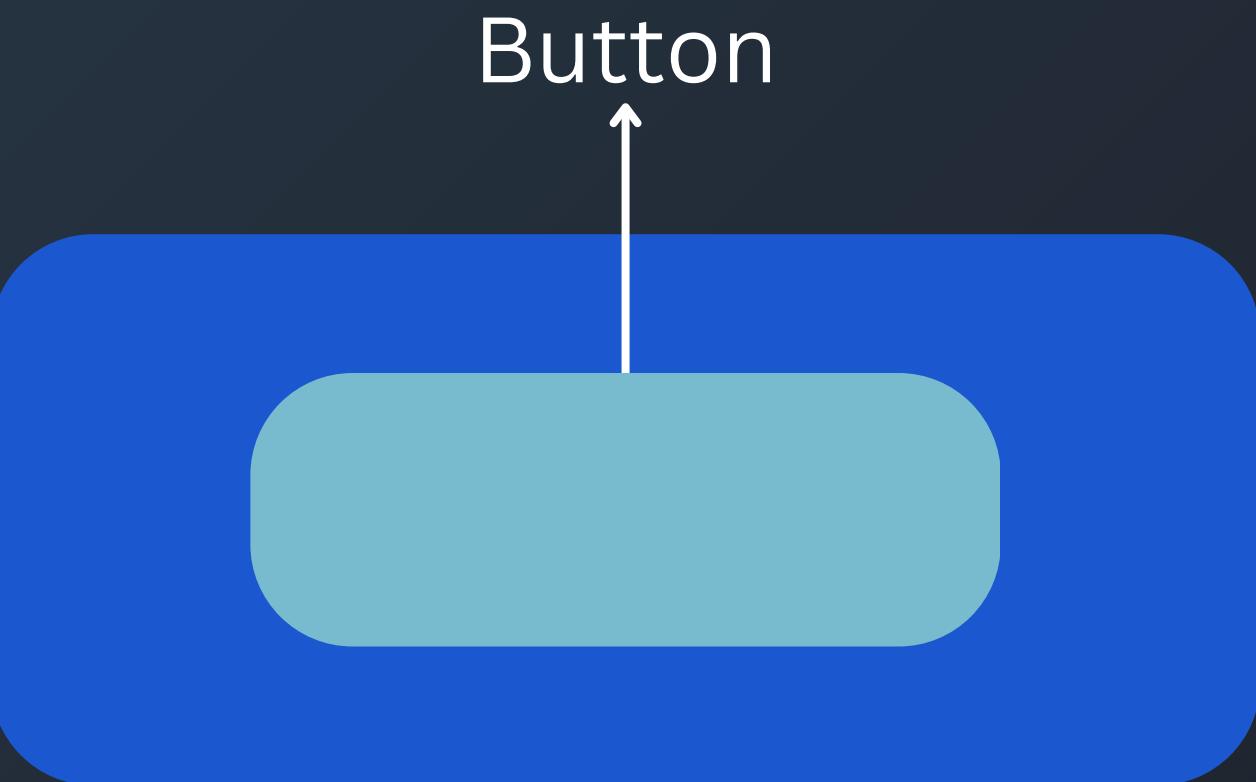
```
import withLogger from "./withLogger";

interface ButtonProps {
  label: string;
}

const Button: React.FC<ButtonProps> = ({ label }) => {
  return <button>{label}</button>;
};

const ButtonWithLogger = withLogger(Button);

export default ButtonWithLogger;
```



Button
ButtonWithLogger

بخش withLogger

در این بخش را به شکل زیر می سازیم، توجه کنید که استفاده از Hook های useEffect و useRef صرفا جهت اینکه برای هر Component فقط یکبار Log نمایش داده شود استفاده شده اند و در منطق Decorator هیچ تاثیری ندارند.



withLogger

```
import React, { useEffect, useRef } from 'react';

const withLogger = <P extends object>(WrappedComponent: React.ComponentType<P>) => {
  return (props: P) => {
    const log = useRef(false);

    useEffect(() => {
      if (!log.current) {
        console.log('Props:', props);
        log.current = true;
      }
    }, []);

    return <WrappedComponent {...props} />;
};

export default withLogger;
```

```
const withLogger = <P extends object>(WrappedComponent: React.ComponentType<P>) => {
  return (props: P) => {
    console.log('Props:', props);
    return <WrappedComponent {...props} />;
  };
};

export default withLogger;
```

در حالات زیر می توانید کد این بخش را به این شکل کوتاه شده بنویسید.

- برای هر Component بیش از یکبار Log نمایش داده شود.
- بخش StrictMode را غیر فعال کرده باشید.

طرز استفاده از Button

در پایان به شکل زیر از Button که ساختیم، استفاده می کنیم.



DecoratorDesignPattern



```
import Button from './pattern/Button';

const DecoratorDesignPattern = () => {
  return (
    <div>
      <Button label={'Click Me'} />
    </div>
  );
};

export default DecoratorDesignPattern;
```

استفاده از Render Props

مثال اصلی را با استفاده از HOC پیاده سازی کردیم اما در این بخش بصورت خلاصه و همان کار را پیاده سازی کرده ایم.



Example With Render Props

```
● ● ●

type WithLoggerProps<P> = {
  children: (props: P) => React.ReactNode;
} & P;

const WithLogger = <P extends object>({ children, ...props }: WithLoggerProps<P>) => {
  console.log('Props:', props);
  return <>{children(props as P)}</>;
};

type ButtonProps = {
  label: string;
};

const Button: React.FC<ButtonProps> = ({ label }) => {
  return <button>{label}</button>;
};

const Usage: React.FC = () => {
  return (
    <WithLogger<ButtonProps> label="Click Me">
      {(props) => <Button {...props} />}
    </WithLogger>
  );
};

export default Usage;
```