Evolution Engineering

React Patterns

Bootcamp [TypeScript]

Higher-Order Components

ℹ️ A higher-order component (HOC) is an advanced technique in React for reusing component logic.

💡 Basically, it's a function that takes a component and returns it with additional data, behavior or even additional JSX.

```
// HOC JavaScript example

function withHOC(WrappedComponent) {
    const HOC = () => <WrappedComponent />;

    return HOC;
};
```

https://reactjs.org/docs/higher-order-components.html

Higher-Order Components

➡️ Conventions:

- Pass Unrelated Props Through to the Wrapped Component
- Maximizing Composability
- Wrap the Display Name for Easy Debugging (see more about displayName)

⚠️ Caveats:

- Don't Use HOCs Inside the render Method
- Static Methods Must Be Copied Over
- Refs Aren't Passed Through

https://reactjs.org/docs/higher-order-components.html

Higher-Order Components

- Basic patterns:
  - **Enhancers** - wrap a component with additional functionality.
  - **Injectors** - inject props into a component.

```
import React from "react";

export interface WithEnhancerHOCProps {
    hasIdea?: boolean;
}

export function withEnhancerHOC<T extends WithEnhancerHOCProps>(WrappedComponent:
React.ComponentType<T>) {
    const displayName = WrappedComponent.displayName || WrappedComponent.name || "Component";
    const ComponentWithEnhancerHOC = ({ hasIdea = false, ...props }: T & WithEnhancerHOCProps) => (
        <>
            <span className="Idea" role="img" aria-label="idea">
                {hasIdea ? "💡" : "Hmm..."}
            </span>
            <WrappedComponent {...(props as T)} />
        </>
    );

    ComponentWithEnhancerHOC.displayName = `withEnhancerHOC(${displayName})`;
```

https://codesandbox.io/s/higher-order-components-50wxj?file=/src/App.tsx

https://reactjs.org/docs/higher-order-components.html

Render Props

ℹ️ The term “render prop” refers to a simple technique for sharing code between React components using a prop whose value is a function.

```
export default function App() {
  return (
    <div className="App">
        <ComponentWithRenderProp render={prop => (
            <h1>Wow, {prop}</h1>
        )}/>
    </div>
  );
}
```

via function as **render** prop

```
export default function App() {
  return (
    <div className="App">
        <ComponentWithRenderProp>
            {prop => <h1>Wow, {prop}</h1>}
        </ComponentWithRenderProp>
    </div>
  );
}
```

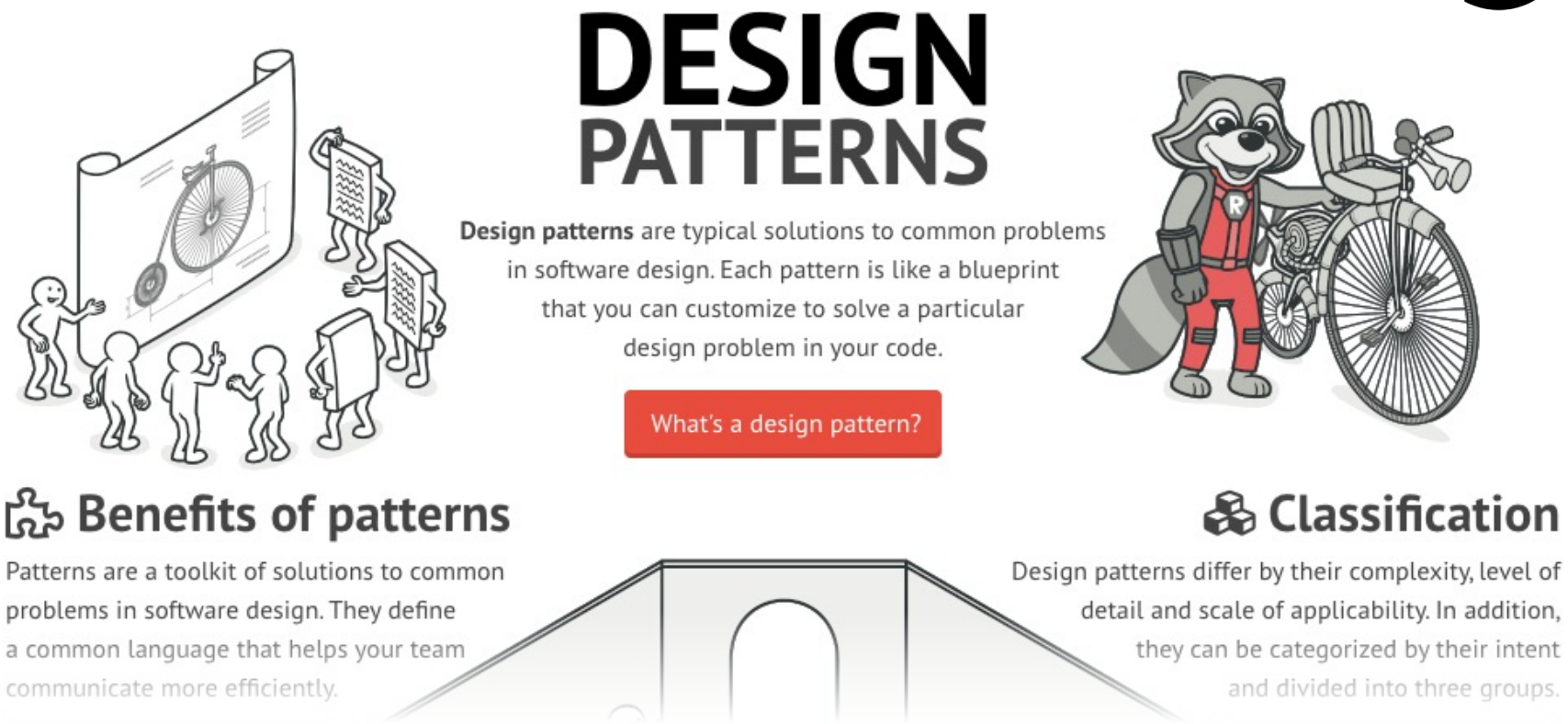
via function as **children**

https://reactjs.org/docs/render-props.html

Code Quality Principles

ℹ️ **KISS** - Keep it simple, stupid
DRY - Don’t Repeat Yourself
YAGNI - You ain't gonna need it
SOLID - Single responsibility principle
Open/closed principle
Liskov substitution principle
Interface segregation principle
Dependency inversion principle

Design Patterns



https://refactoring.guru/design-patterns

Additional Materials

1. Do React Hooks Replace Higher Order Components (HOCs)?: https://medium.com/javascript-scene/do-react-hooks-replace-higher-order-components-hocs-7ae4a08b7b58
2. React Patterns https://reactpatterns.com/
3. The Principles of Functional Programming: https://www.freecodecamp.org/news/the-principles-of-functional-programming/
4. Professor Frisby's Mostly Adequate Guide to Functional Programming: https://github.com/MostlyAdequate/mostly-adequate-guide
5. Courses recommended by reactjs.org: https://reactjs.org/community/courses.html

Thank you!

React Patterns ✅