Lessons Learned Building React Applications

Adhithi Ravichandran

Software Consultant, Author, Speaker

@AdhithiRavi www.adhithiravichandran.com

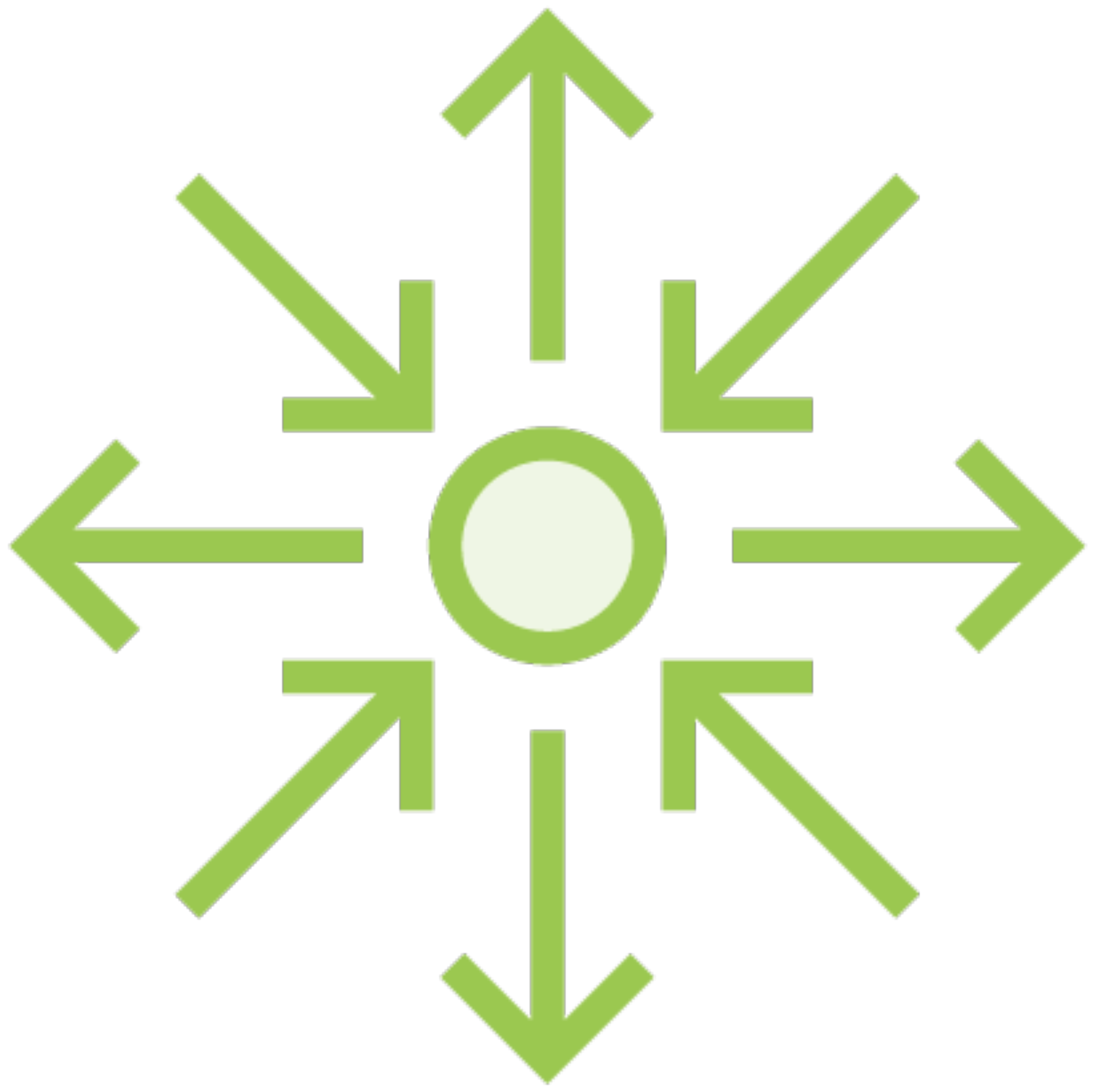
React

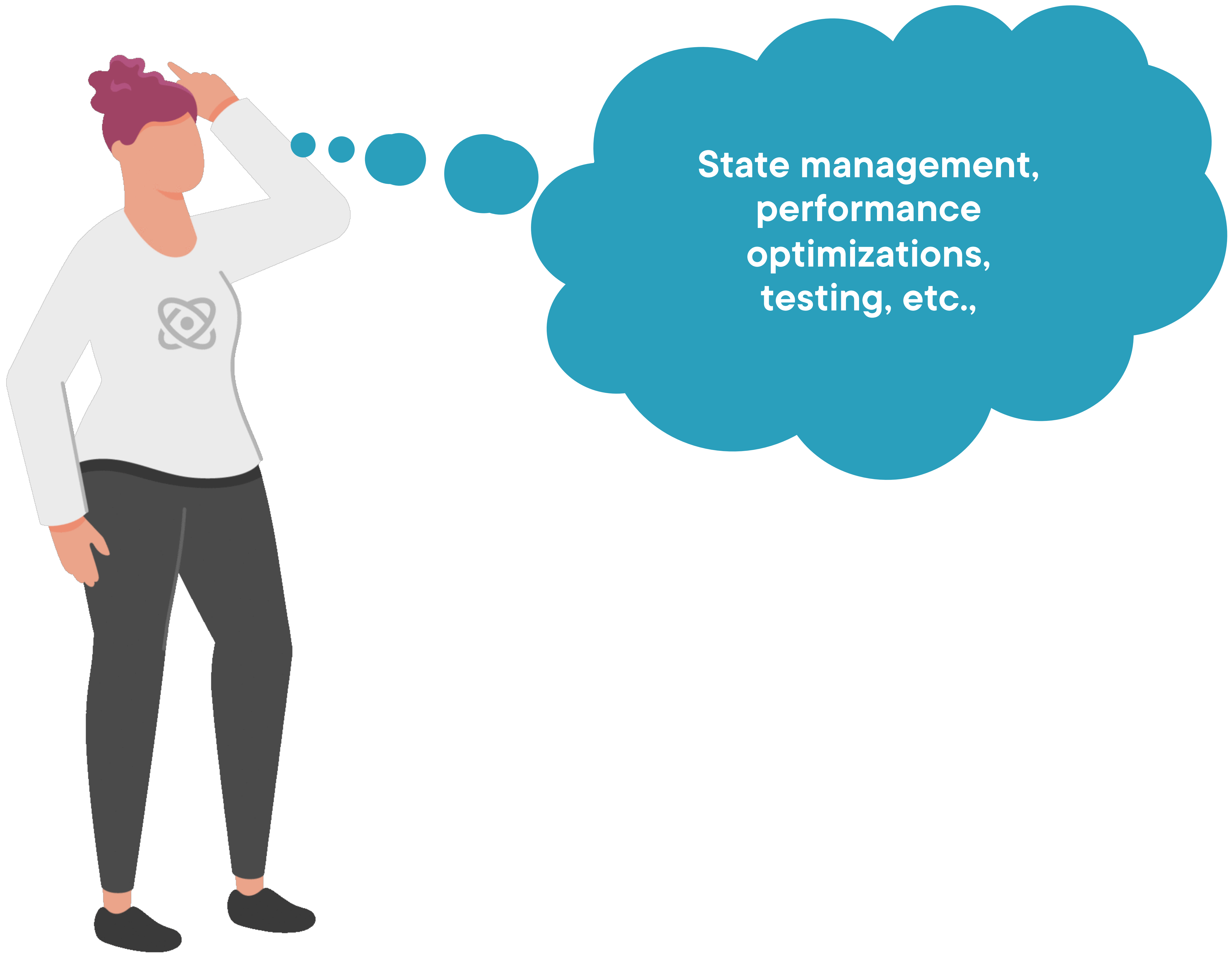
React is a JavaScript library for building rich user interfaces

Flexibility

Offers lots of flexibility

Freedom to choose state management, navigation, server-side rendering and more

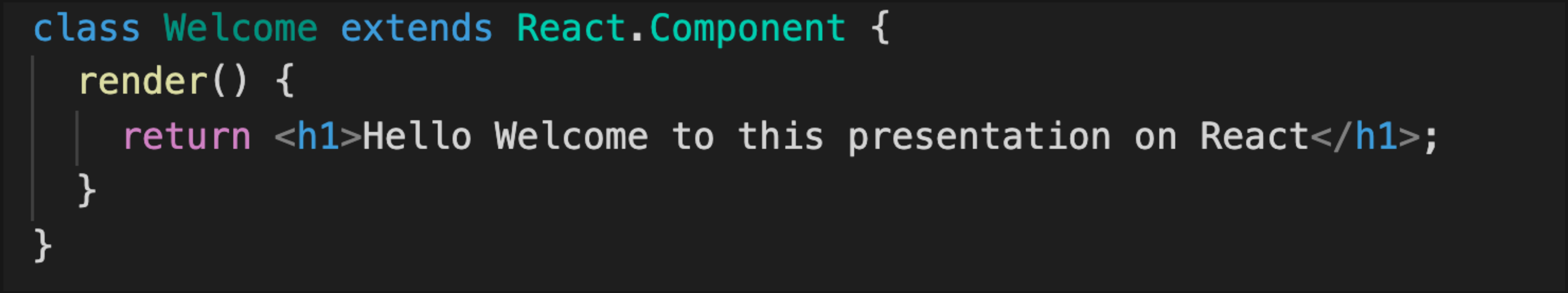



State management, performance optimizations, testing, etc.,

Lesson 1: Use Functional Components

Class components are a story of the past

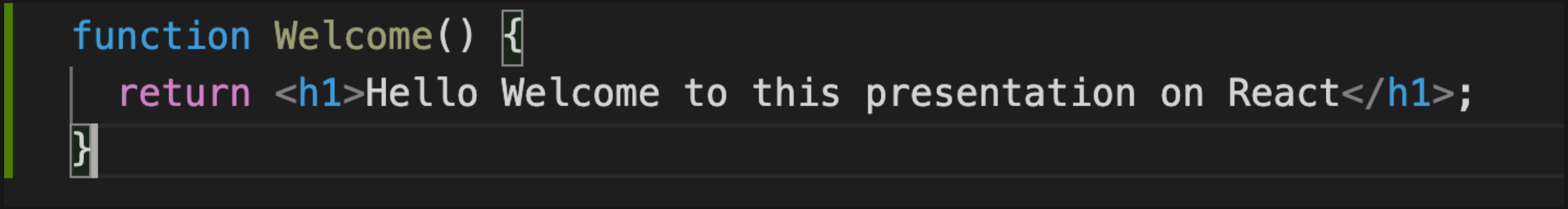
Class components are confusing to people and machines



```
class Welcome extends React.Component {
  render() {
    return <h1>Hello Welcome to this presentation on React</h1>;
  }
}
```

React Class Component

Define a component as an ES6 class



```
function Welcome() {
  return <h1>Hello Welcome to this presentation on React</h1>;
}
```

React Functional Component

A functional component is a JavaScript/ES6 function that returns a React element (JSX).

Write React Functional Components

Simpler to learn and write

Less code

Easier to maintain and test

Hooks

Hook is a special function that let’s you hook into React features, within functional components!

Hooks

- useState
- useEffect
- useContext
- useReducer
- useMemo
- useRef
- useCallback
- useLayoutEffect

Functional Component

```jsx
import React, { useState } from "react";

function YesNoButtonComponent() {
  const [button, setButton] = useState("");

  const onButtonPress = (buttonName) => {
    setButton(buttonName);
    console.log({ button });
  };

  return (
    <div>
      <button onClick={() => onButtonPress("Yes")}>Yes</button>
      <button onClick={() => onButtonPress("No")}>No</button>
    </div>
  );
}

export default YesNoButtonComponent;
```

Functional Component with Arrow Function

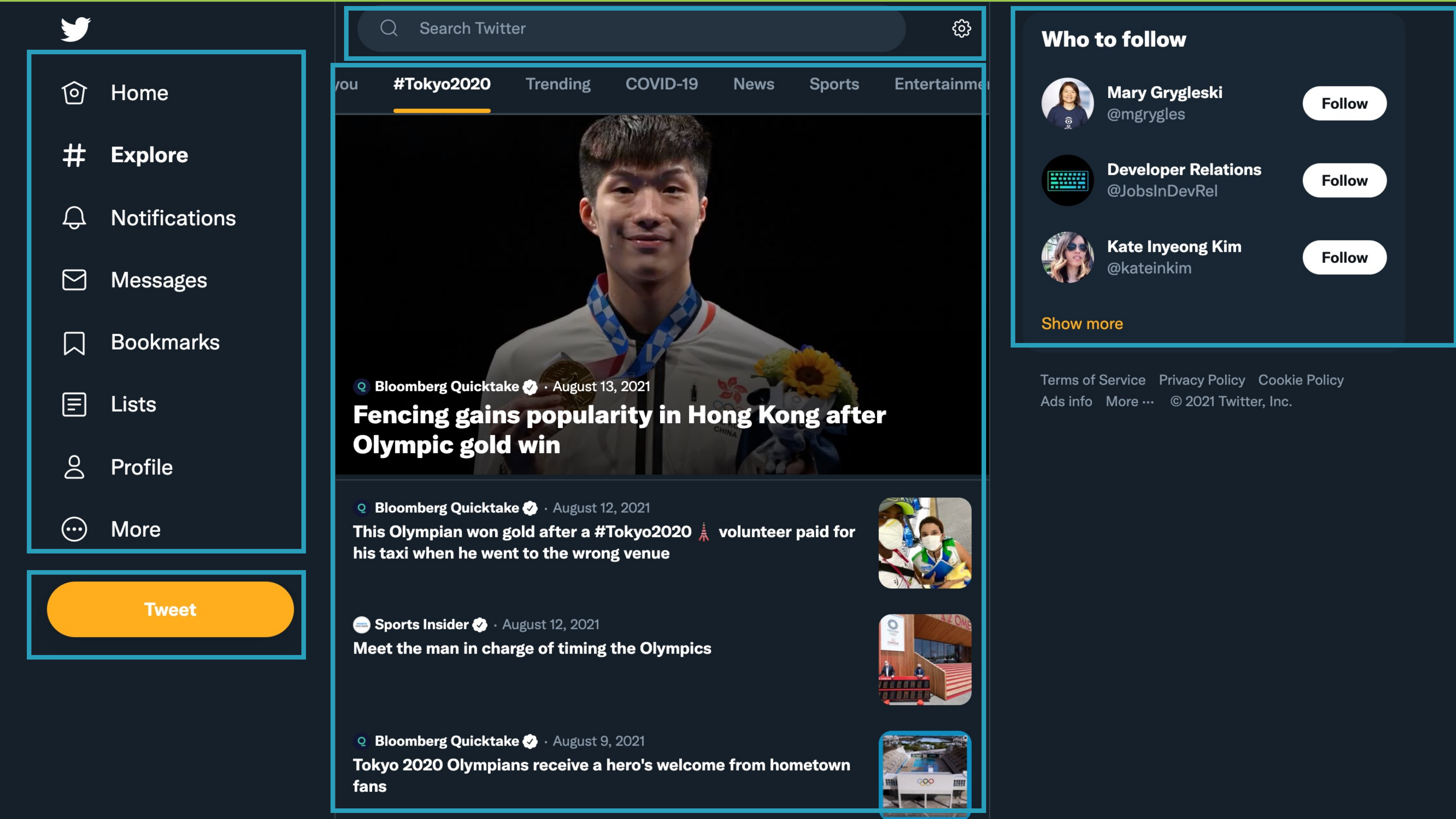
```
import React, { useState } from "react";

const YesNoButtonComponent = () => {
  const [button, setButton] = useState("");

  const onButtonPress = (buttonName) => {
    setButton(buttonName);
    console.log({ button });
  };

  return (
    <div>
      <button onClick={() => onButtonPress("Yes")}>Yes</button>
      <button onClick={() => onButtonPress("No")}>No</button>
    </div>
  );
};

export default YesNoButtonComponent;
```

Functional components are the future of React!

Lesson 2: Break down your components – when needed!

When do I break down components?

- **Managing state is a nightmare**
- **Performance concerns with re-rendering of application**
- **Code readability takes a hit**
- **Working with multiple developers on the codebase becomes challenging**
- **Testing code is harder**



Bloomberg Quicktake · August 13, 2021
Fencing gains popularity in Hong Kong after Olympic gold win

Bloomberg Quicktake · August 12, 2021
This Olympian won gold after a #Tokyo2020 🗼 volunteer paid for his taxi when he went to the wrong venue

Sports Insider · August 12, 2021
Meet the man in charge of timing the Olympics

Bloomberg Quicktake · August 9, 2021
Tokyo 2020 Olympians receive a hero's welcome from hometown fans

Who to follow

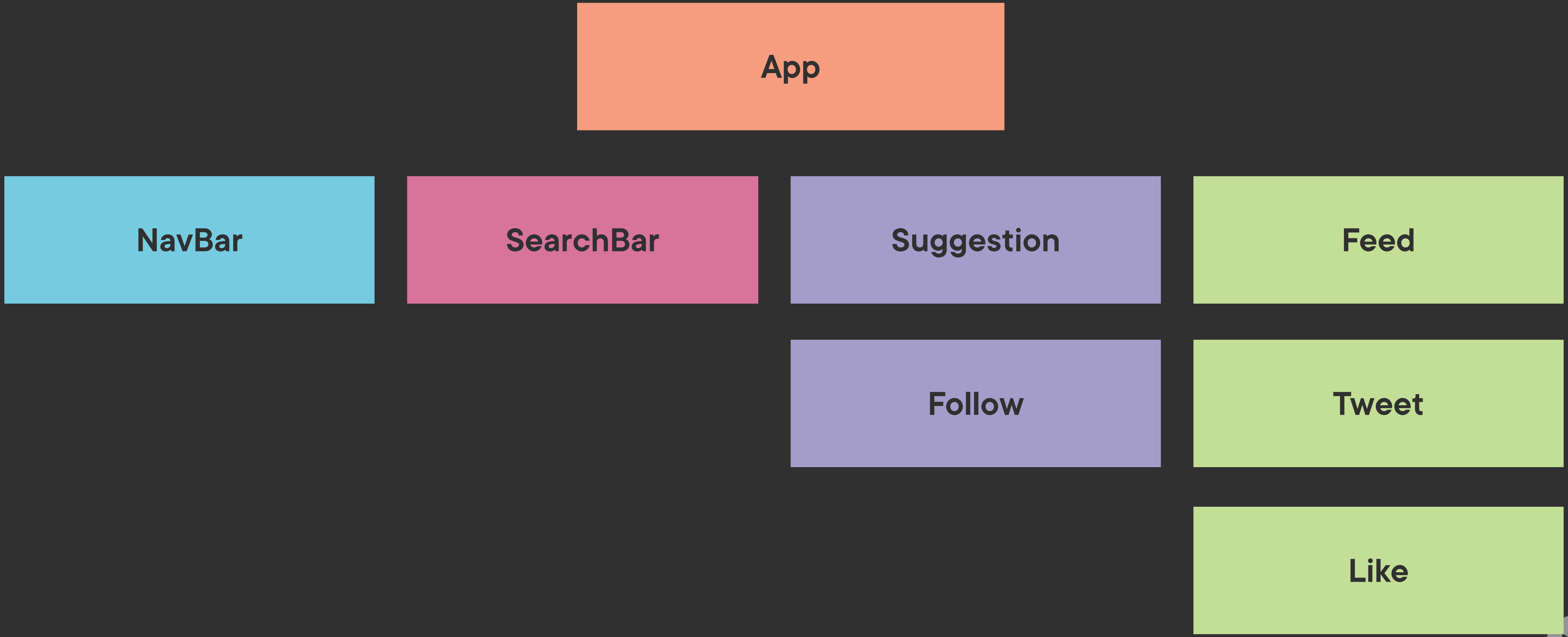
Mary Grygleski
@mgrygles

Developer Relations
@JobsInDevRel

Kate Inyeong Kim
@kateinkim

Terms of Service Privacy Policy Cookie Policy
Ads info More ··· © 2021 Twitter, Inc.

Components Hierarchy



Each component is a building block that is a reusable piece of UI. Putting them together results in a complete application!

Lesson 3: TypeScript is a life saver!

Why TypeScript?

Catch problems early on

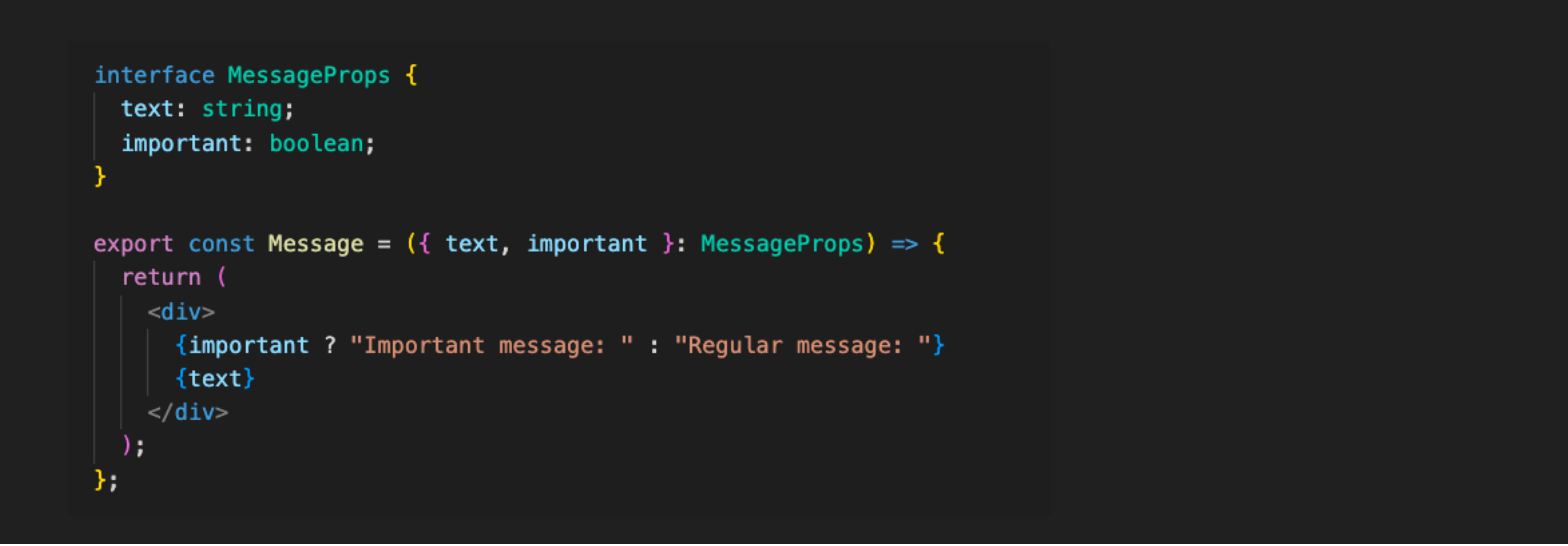
Intellisense is accurate

Easier to refactor code

Readable code

Easier to maintain and test

High quality documentation (TSDoc)



```
interface MessageProps {
  text: string;
  important: boolean;
}

export const Message = ({ text, important }: MessageProps) => {
  return (
    <div>
      {important ? "Important message: " : "Regular message: "}
      {text}
    </div>
  );
};
```

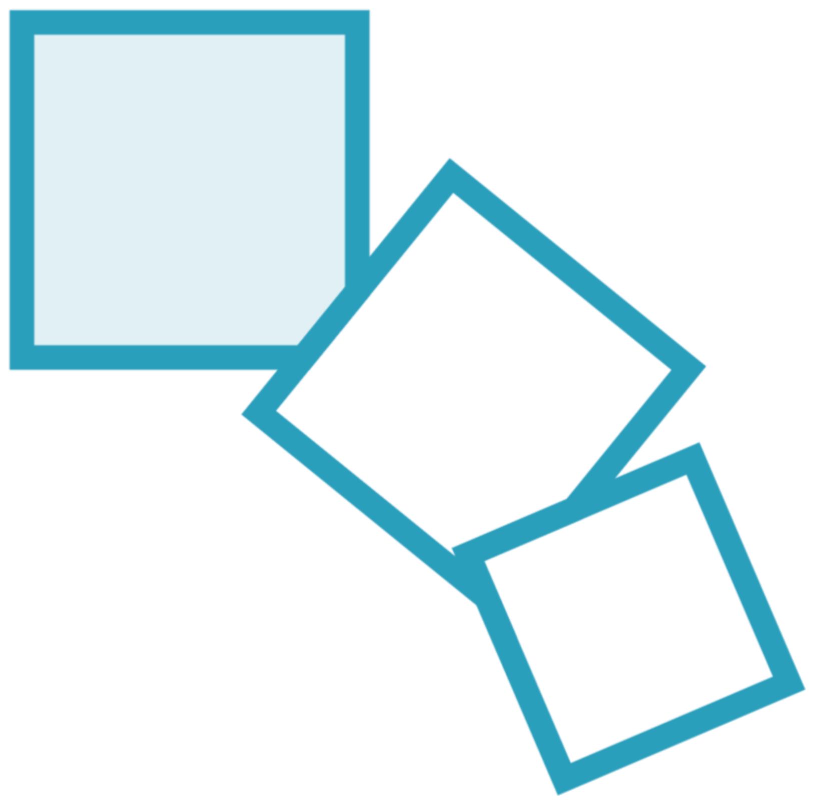
Component in TypeScript

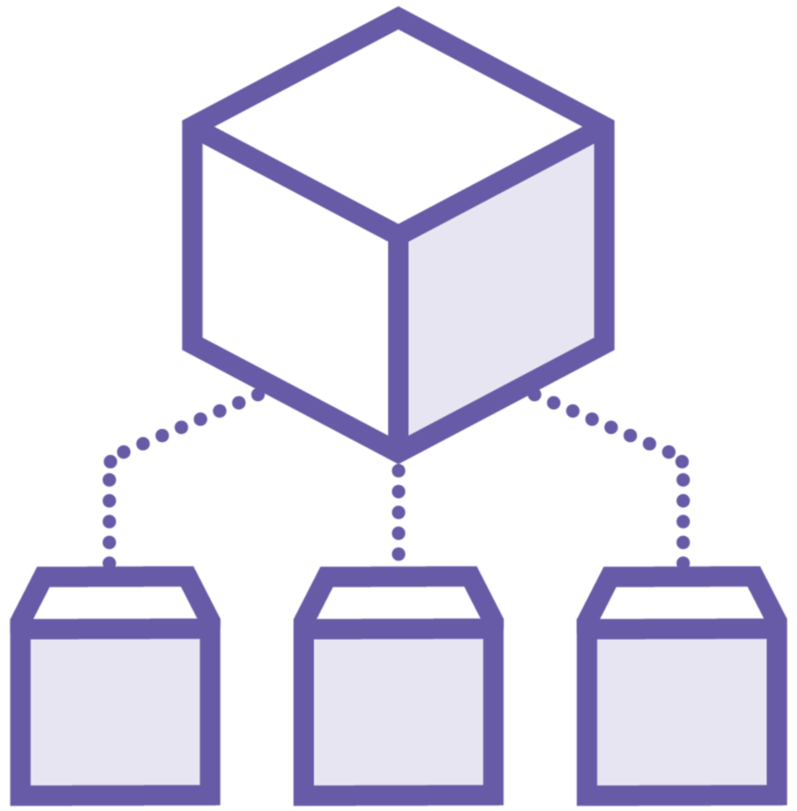
Interface defines the *props* that are accepted by the component using an object type

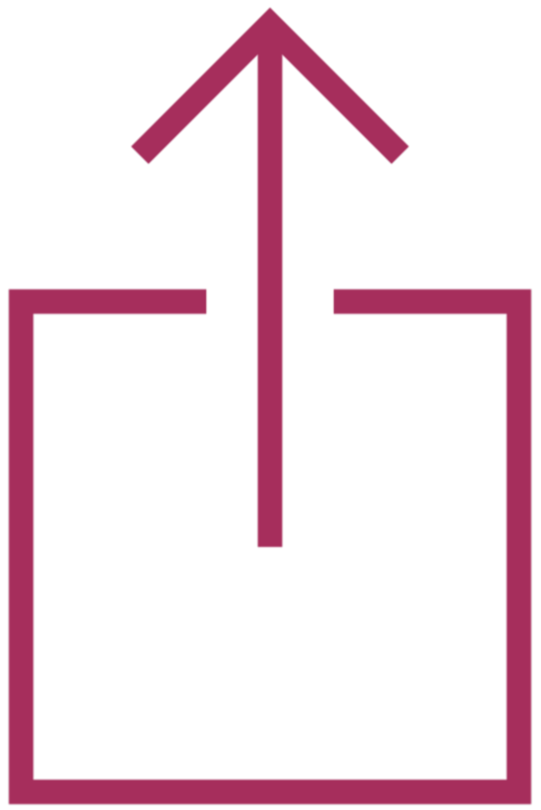
***MessageProps* is the interface that describes the *props* the component accepts**

Lesson 4: Start with local state!

State Management

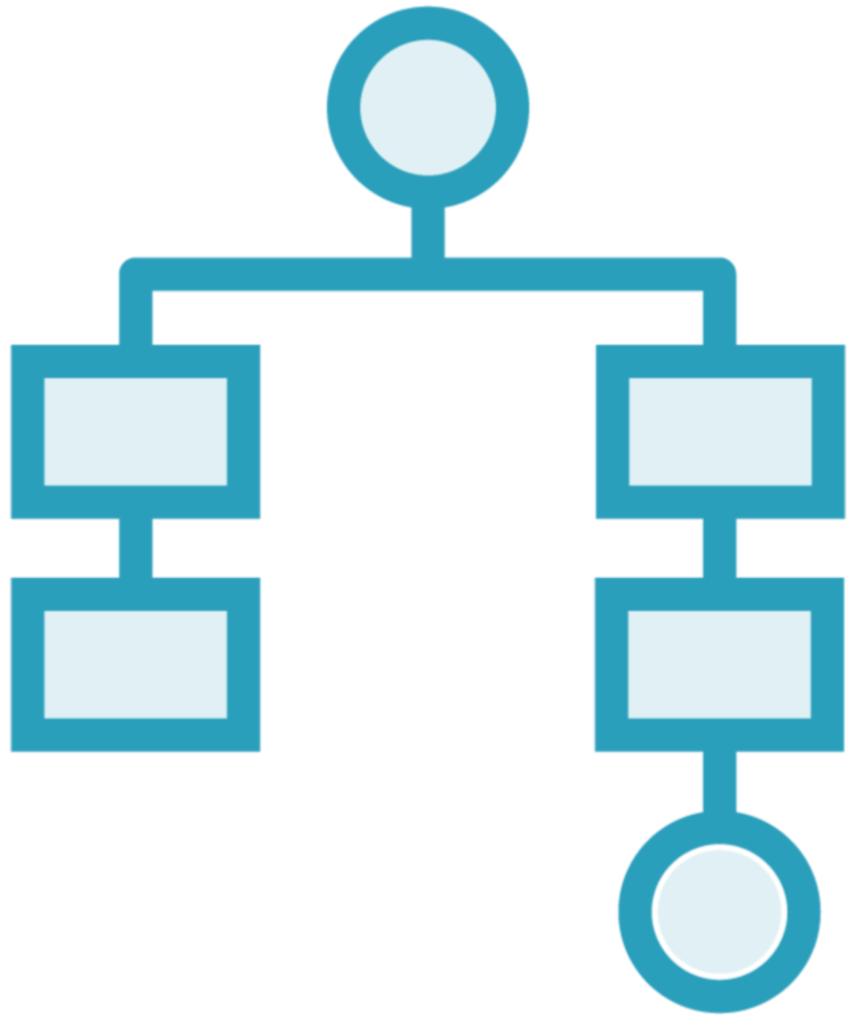

Start with local state first


Pass down state via *props* if child component needs it


Lift state up if non-child component needs the data

Next choice: Context or external state management

Context



Context is designed to share data that is global for a tree of React components.

Avoid passing props through multiple layers of components.

Examples: Theme, demographic information, language, etc..

Lesson 5: Understand when React renders

What triggers a React render?

State changes

Prop changes

Parent renders

Context changes

React Memo

If your component renders the same result, given the same *props* – Use React.memo.

React will skip rendering the component and reuse latest rendered result.

Only for performance optimizations, do not rely on it to prevent renders.

Lesson 6: Test, Test and Test!

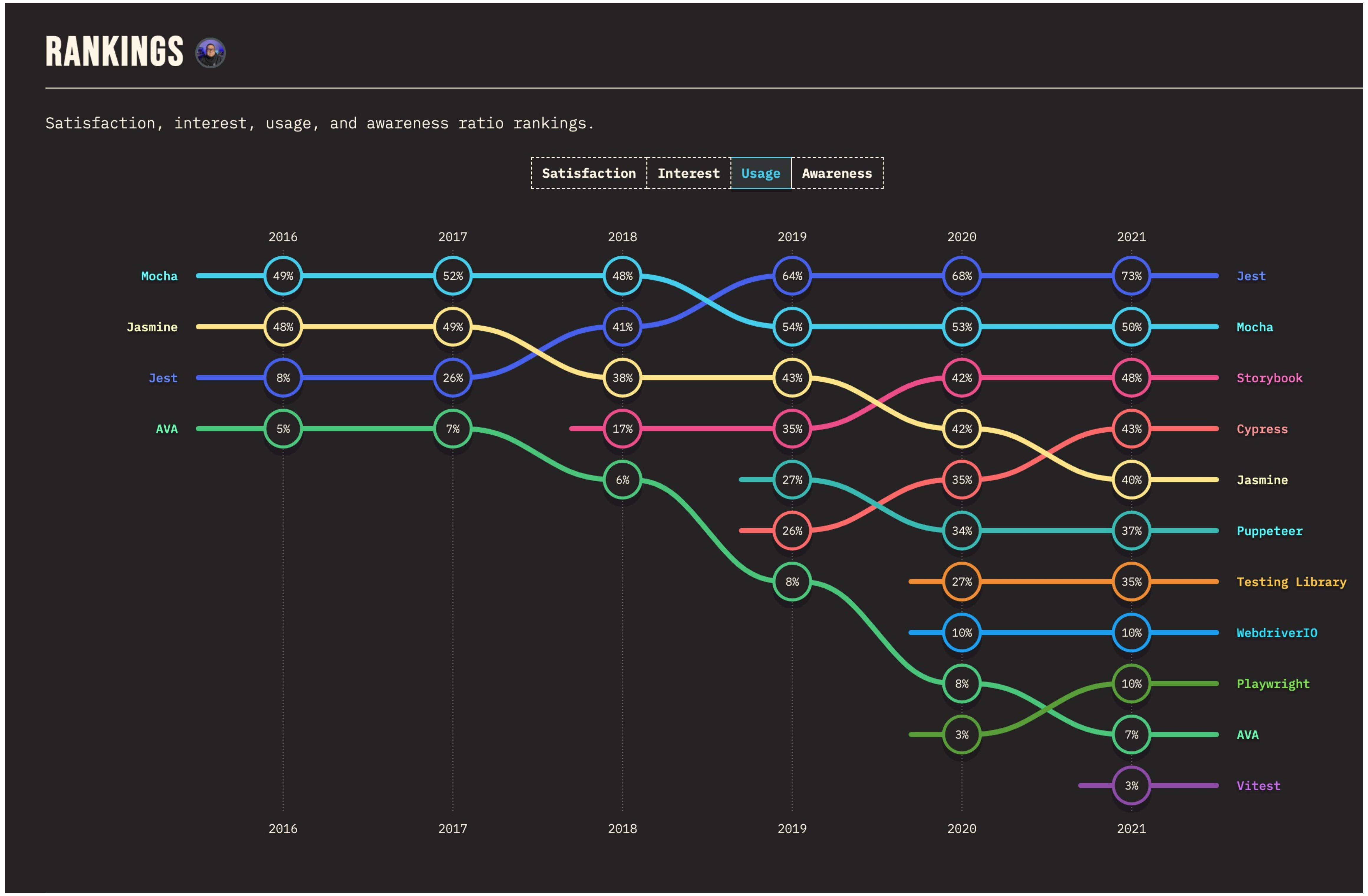
Types of Testing

Unit testing - Jest

Component testing – React testing library

Automated End-to-end tests - Cypress

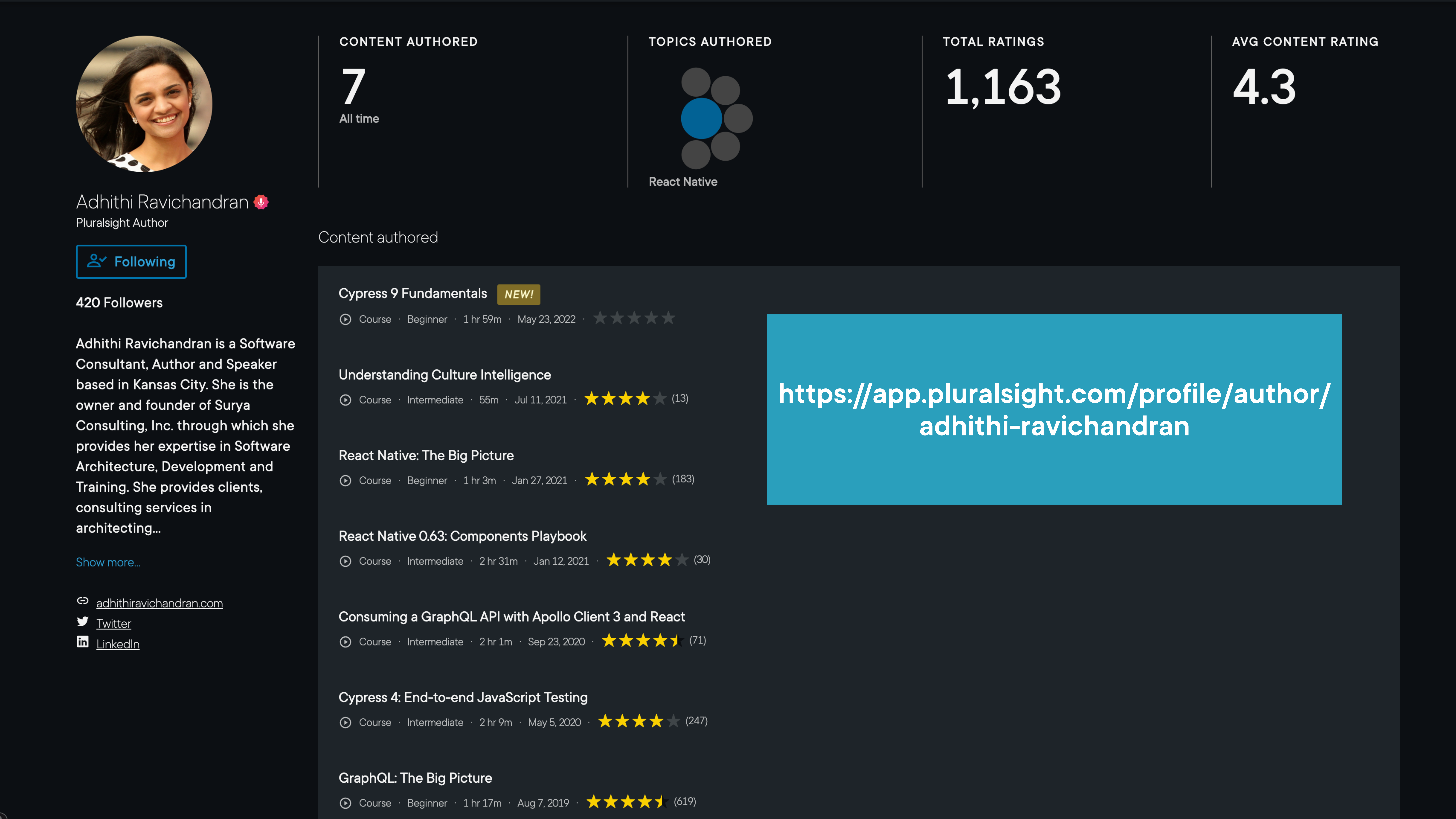
State of JS Survey 2021



RANKINGS

Satisfaction, interest, usage, and awareness ratio rankings.

Thank You!



Adhithi Ravichandran
Pluralsight Author

Following

420 Followers

Adhithi Ravichandran is a Software Consultant, Author and Speaker based in Kansas City. She is the owner and founder of Surya Consulting, Inc. through which she provides her expertise in Software Architecture, Development and Training. She provides clients, consulting services in architecting...

Show more...

adhithiravichandran.com
Twitter
LinkedIn

CONTENT AUTHORED
7
All time

TOPICS AUTHORED
React Native

TOTAL RATINGS
1,163

AVG CONTENT RATING
4.3

Content authored

Cypress 9 Fundamentals *NEW!*
Course · Beginner · 1 hr 59m · May 23, 2022 · ☆☆☆☆☆

Understanding Culture Intelligence
Course · Intermediate · 55m · Jul 11, 2021 · ★★★★☆ (13)

React Native: The Big Picture
Course · Beginner · 1 hr 3m · Jan 27, 2021 · ★★★★☆ (183)

React Native 0.63: Components Playbook
Course · Intermediate · 2 hr 31m · Jan 12, 2021 · ★★★★☆ (30)

Consuming a GraphQL API with Apollo Client 3 and React
Course · Intermediate · 2 hr 1m · Sep 23, 2020 · ★★★★½ (71)

Cypress 4: End-to-end JavaScript Testing
Course · Intermediate · 2 hr 9m · May 5, 2020 · ★★★★☆ (247)

GraphQL: The Big Picture
Course · Beginner · 1 hr 17m · Aug 7, 2019 · ★★★★½ (619)

https://app.pluralsight.com/profile/author/adhithi-ravichandran

[*@AdhithiRavi*](https://twitter.com/AdhithiRavi)