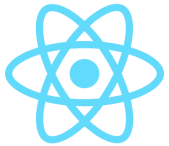
Introduction to ReactJS and beyond

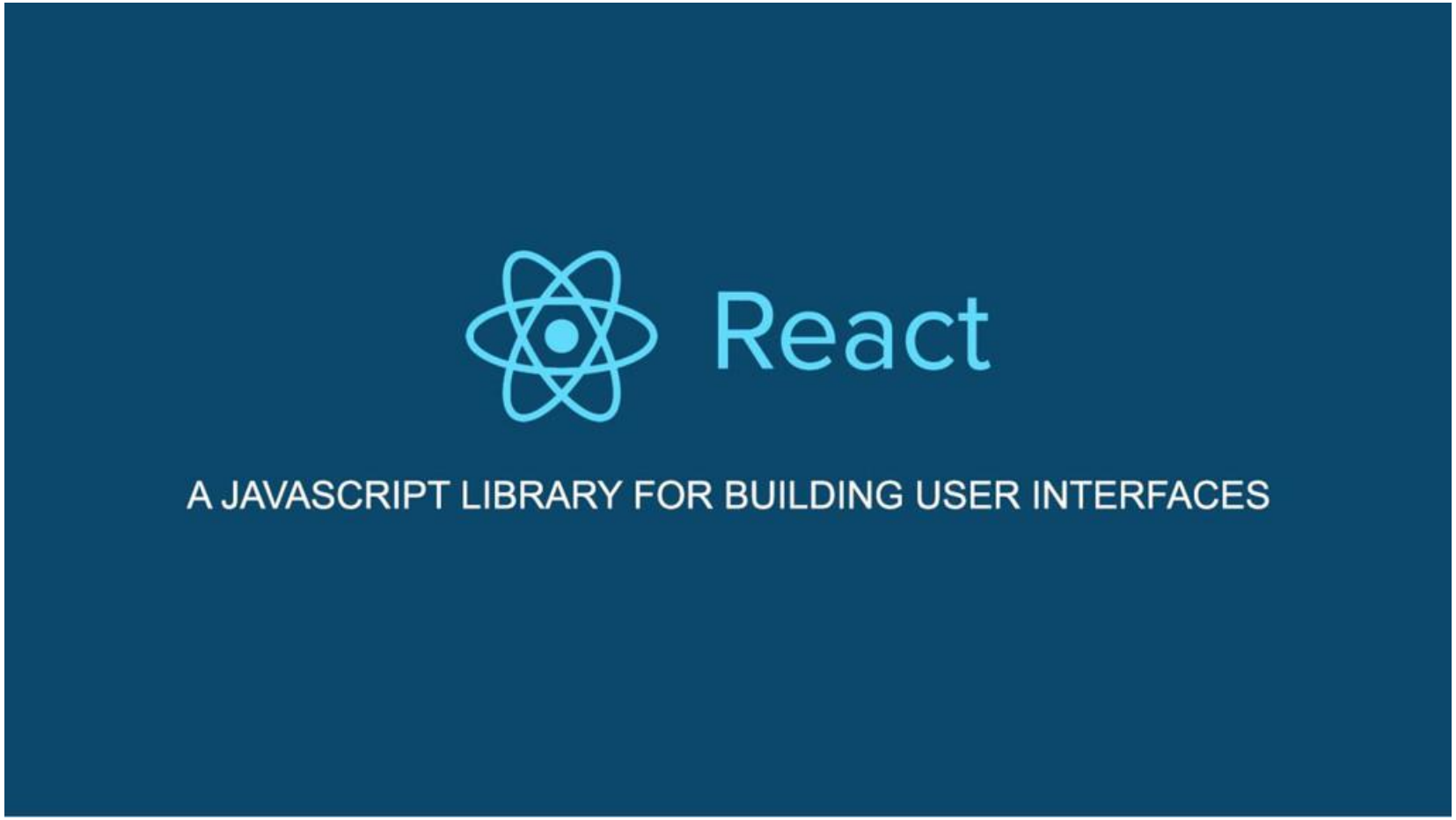
Dam Minh Tien

Outline

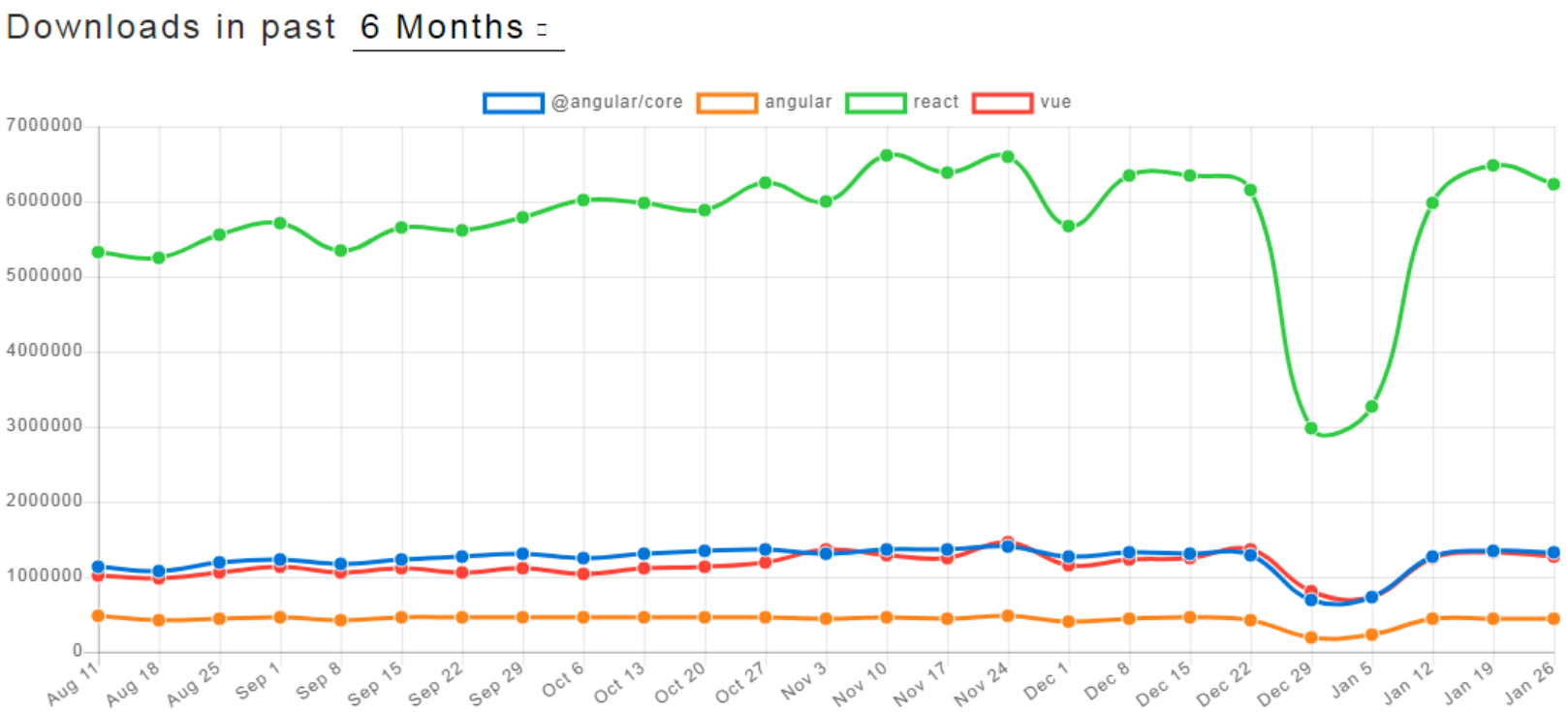
1. What is React
2. Why use React
3. How React works
4. React Hook, Redux, patterns, convention
5. Conclusion



What is React: official definition



What is React: npm trend



Why use React: history

2010 - The first signs of React

•Facebook introduced xhp into its php stack and open sourced it. Xhp allowed creating composite components.

2011 - An early prototype of React

•Jordan Walke created FaxJS, the early prototype of React - shipped a search element on Facebook.

2012 - Something new had started at Facebook

•Facebook Ads became hard to manage, so Facebook needed to come up with a good solution for it. Jordan Walke worked on the prototype and created React.

2013 - The year of the Big Launch

•May 29-31: JS ConfUS. **React gets open sourced**.

June 2: React (by Facebook) is available on JSFiddle

2014 - The year of Expansion

•Early 2014: #reactjsworldtour conferences started, to build community and to 'turn haters into advocates'.

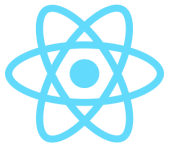
•Jan 2: React Developer Tools becomes an extension of the Chrome Developer Tools.

2015 - React is Stable

•January: Netflix likes React, Airbnb uses React

•June 2: Redux was released by Dan Abramov and Andrew Clark.

2016 - 2020 React gets mainstream



Why use React: traditional JS

Quora

Search for questions, people, and topics

HTML | JavaScript (programming language) | Web Development | Computer Programming

How do you write HTML in a JavaScript file?

7 Answers

Dennis Isaac, Developing awesome websites since 2009

Answered Jul 29 2015 · Author has **82** answers and **163.6k** answer views

Originally Answered: how do you write html in javascript file?

There are several methods to do it. It's upto you to choose the best suitable method for the situation.

Method 1 - The write() method writes HTML expressions or JavaScript code to a document. This is probably the easiest way to add HTML code through JavaScript file
Example :

```
document.write("<h1>Hello World!</h1><p>Have a nice day!</p>");
```

Method 2 - The appendChild() method appends a node as the last child of a node. You have to create the HTML as a node object before adding
Example :

```
var para = document.createElement("P");   // Create a  <p> node
var t =  document.createTextNode("This is a paragraph.");
para.appendChild(t);        // Append the text to <p>
document.getElementById("myDIV").appendChild(para);
```

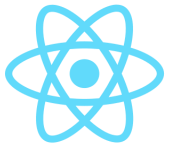
???

How React works: JavaScript eXtension

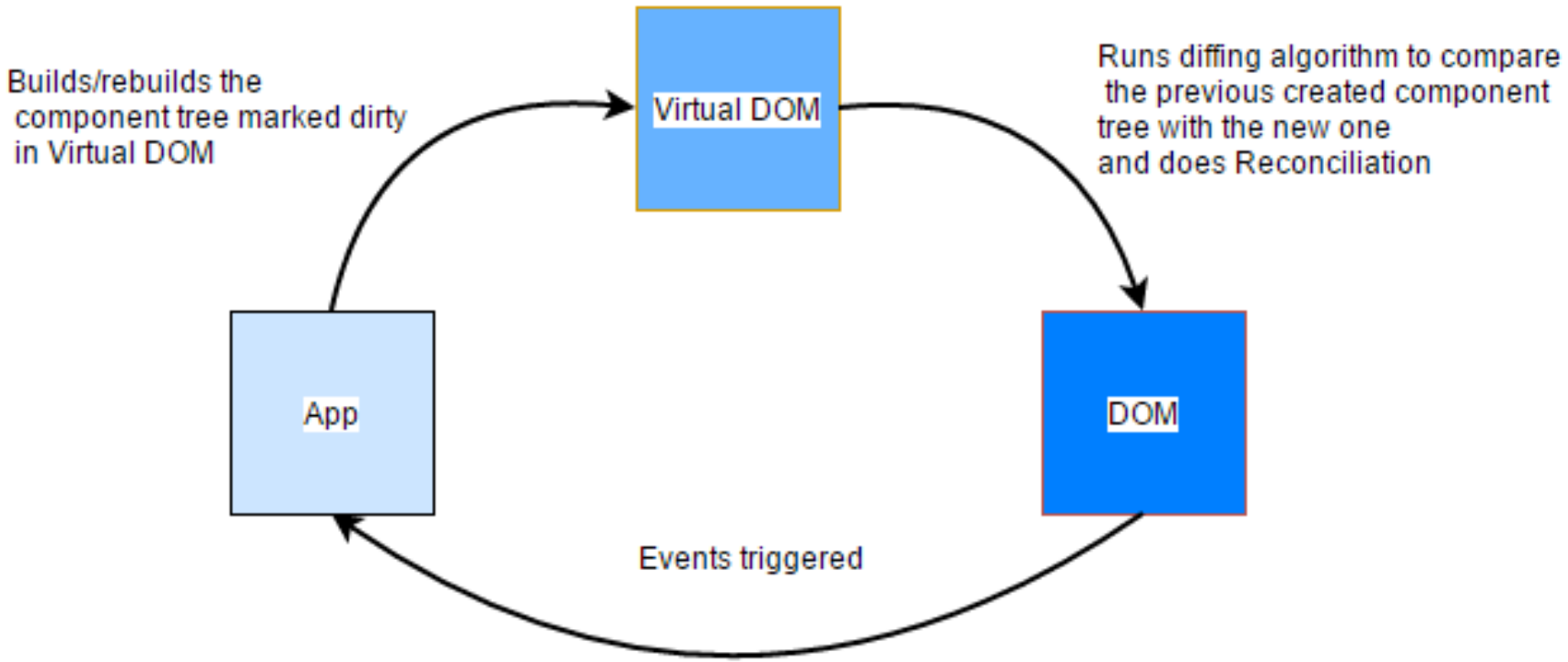
Instead of artificially separating *technologies* by putting markup and logic in separate files, React separates *concerns* with loosely coupled units called “components” that contain both

```
const element = <><h1>Hello World</h1><p>Have a nice day</p></>
```

This funny tag syntax is neither a string nor HTML.
It is called JSX, and it is a syntax extension to JavaScript. We recommend using it with React to describe what the UI should look like. JSX may remind you of a template language, but it comes with the full power of JavaScript.



How React works: virtual DOM

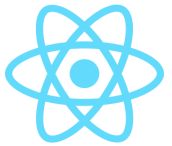


How React works: everything is component

Benefits:

- Composable
- Reusable
- Maintainable
- Testable

```
class C4IGreeting extends React.Component {
  render() {
    return <h1>Hello, {this.props.name}</h1>;
  }
}
```



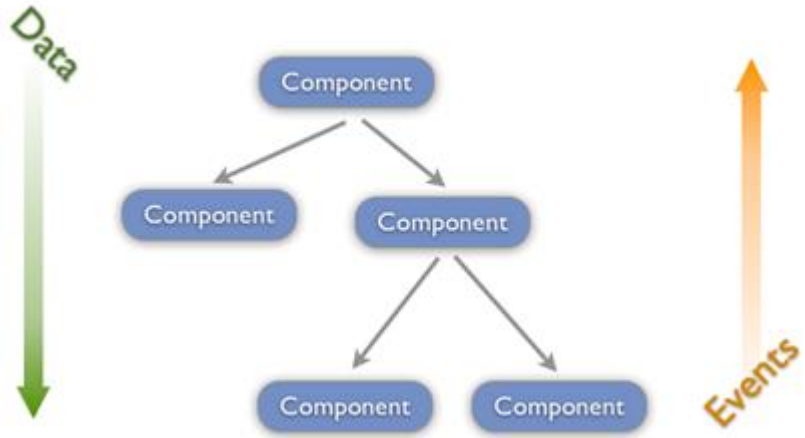
How React works: single source of Truth

Traditional data flows:

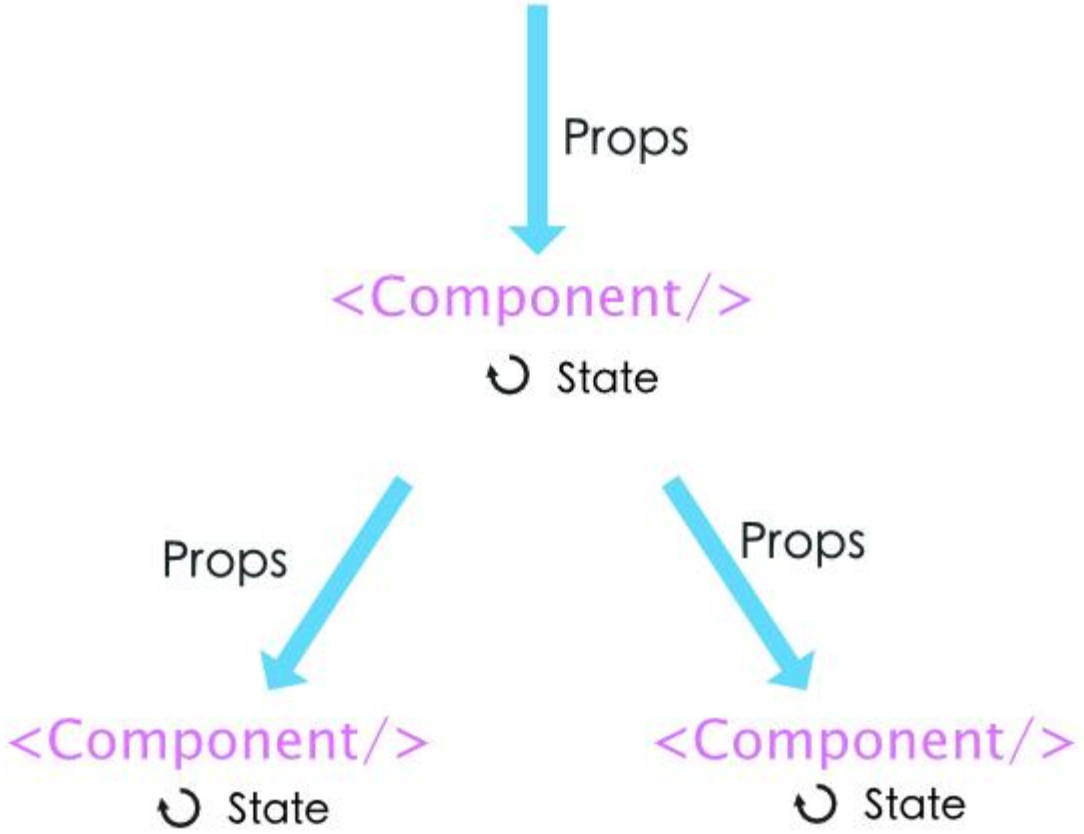
No framework: any components can communicate with any other components

AngularJS: two-way data binding

ReactJS: one-way data binding



How React works: state & prop



Prop vs State

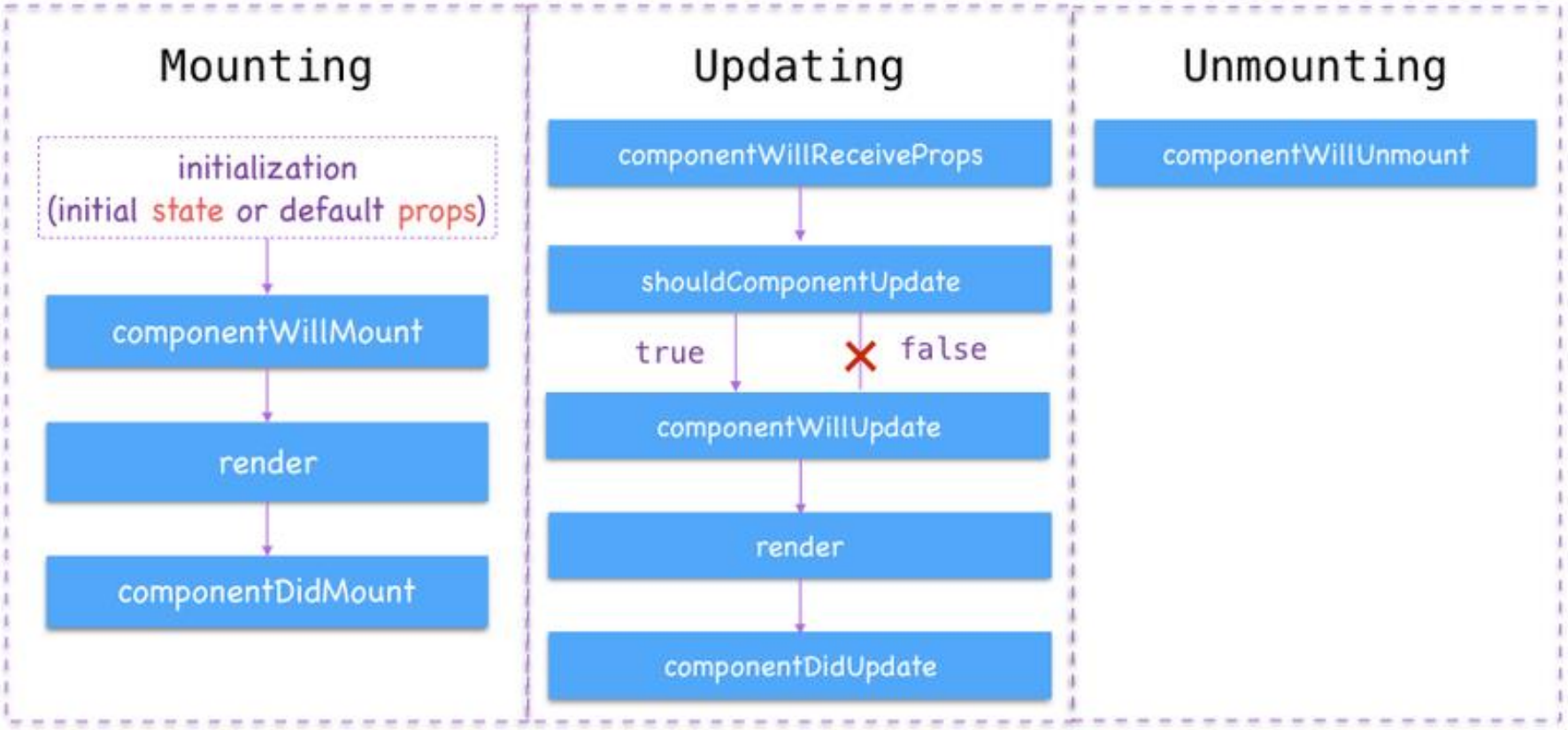
Prop

- External data
- Can’t change it

State

- Internal data
- Can change it
- Private for component

How React works: life cycle

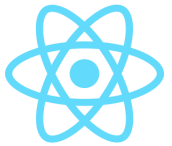


Furthermore: react hook

Hooks are a new addition in React 16.8. They let you use state and other React features without writing a class.

```jsx
import React from 'react';

class Example extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
        count: 0
    };
  }

  setCount(count) {
    this.setState({
      count,
    })
  }

  render() {
    return (
      <div>
        <p>You clicked {count} times</p>
        <button onClick={() => setCount(count + 1)}>
          Click me
        </button>
      </div>
    );
  }
}
```

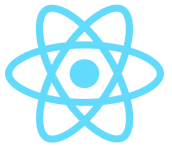
```jsx
import React, { useState } from 'react';

function Example() {
  // Declare a new state variable, which we'll call "count"
  const [count, setCount] = useState(0);

  return (
    <div>
      <p>You clicked {count} times</p>
      <button onClick={() => setCount(count + 1)}>
        Click me
      </button>
    </div>
  );
}
```



Furthermore: react hook - useEffect

If you’re familiar with React class lifecycle methods, you can think of useEffect Hook as componentDidMount, componentDidUpdate, and componentWillUnmount combined.

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

function Example() {
  const [count, setCount] = useState(0);

  // Similar to componentDidMount and componentDidUpdate:
  useEffect(() => {
    // Update the document title using the browser API
    document.title = `You clicked ${count} times`;
  });

  return (
    <div>
      <p>You clicked {count} times</p>
      <button onClick={() => setCount(count + 1)}>
        Click me
      </button>
    </div>
  );
}
```

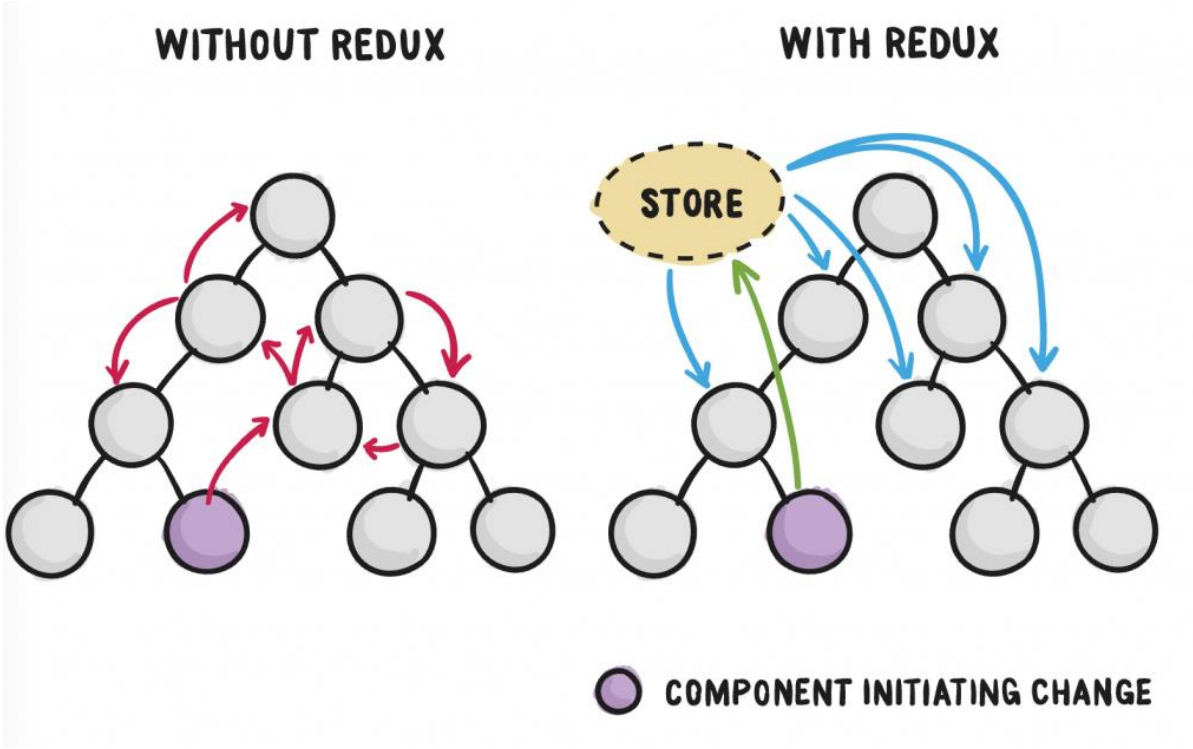


Furthermore: Redux

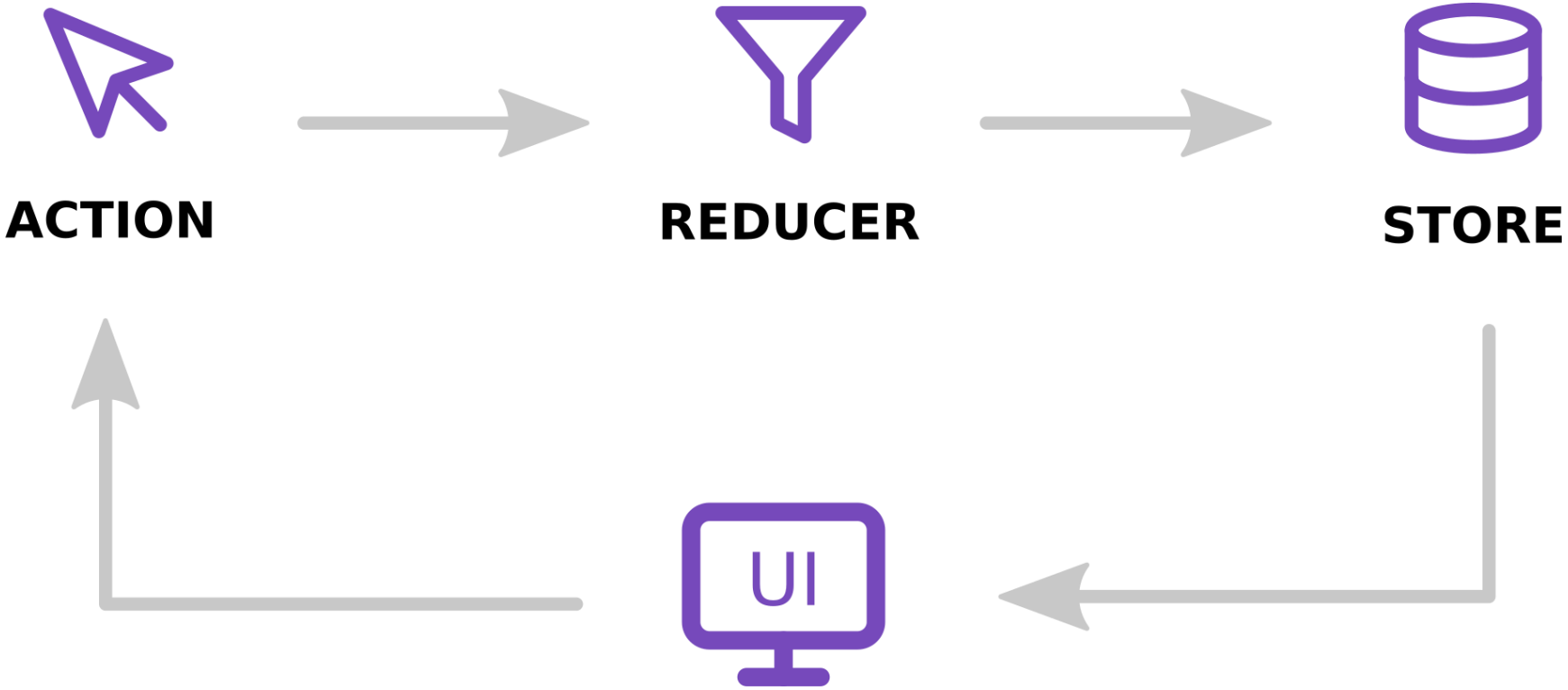
A Predictable State Container for JS Apps

Redux uses this concept of uni-directional data flow.

- The application has a *central / root* state.
- A state change triggers View updates.
- Only special functions can change the state.
- Changes are made with pure functions



Furthermore: Redux



Furthermore: patterns

React patterns: https://reactpatterns.com

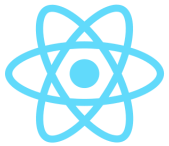
Destructuring props

Destructuring assignment is a JavaScript feature.
It was added to the language in ES2015.
So it might not look familiar.
Think of it like the opposite of literal assignment.

```
let person = { name: "chantastic" };
let { name } = person;
```

Works with Arrays too.

```
let things = ["one", "two"];
let [first, second] = things;
```



Furthermore: patterns

React patterns: https://reactpatterns.com

Conditional rendering

You can't use if/else statements inside a component declarations.

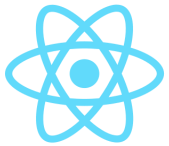
So conditional (ternary) operator and short-circuit evaluation are your friends.

if

```
{
  condition && <span>Rendered when `truthy`</span>;
}
```

if-else

```
{
  condition ? (
    <span>Rendered when `truthy`</span>
  ) : (
    <span>Rendered when `falsy`</span>
  );
}
```

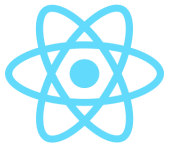


Furthermore: patterns

React patterns: https://reactpatterns.com
Array as children
Providing an array as children is a very common.
It's how lists are drawn in React.

We use map() to create an array of React Elements for every value in the array.

```
<ul>
  {["first", "second"].map(item => (
    <li>{item}</li>
  ))}
</ul>
```



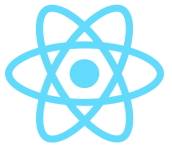
Furthermore: convention

Follow Airbnb: https://github.com/airbnb/javascript/tree/master/react
Some keywords:

Naming

- Extensions: Use .jsx extension for React components. eslint: react/jsx-filename-extension
- Filename: Use PascalCase for filenames. E.g., ReservationCard.jsx.
- Reference Naming: Use PascalCase for React components and camelCase for their instances. eslint: react/jsx-pascal-case

```
// bad
import reservationCard from './ReservationCard';

// good
import ReservationCard from './ReservationCard';

// bad
const ReservationItem = <ReservationCard />;

// good
const reservationItem = <ReservationCard />;
```



Furthermore: convention

Follow Airbnb: https://github.com/airbnb/javascript/tree/master/react
Some keywords:

Tags

- Always self-close tags that have no children. eslint: react/self-closing-comp

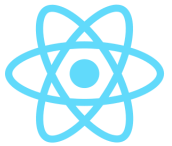
Furthermore: convention

Follow Airbnb: https://github.com/airbnb/javascript/tree/master/react
Some keywords:

Methods

- Use arrow functions to close over local variables. It is handy when you need to pass additional data to an event handler. Although, make sure they do not massively hurt performance, in particular when passed to custom components that might be PureComponents, because they will trigger a possibly needless rerender every time.

```
function ItemList(props) {
  return (
    <ul>
      {props.items.map((item, index) => (
        <Item
          key={item.key}
          onClick={(event) => { doSomethingWith(event, item.name, index); }}
        />
      ))}
    </ul>
  );
}
```



Conclusion: resources

React Official Page: https://reactjs.org/

Learn React & get Certification: https://www.freecodecamp.org/

React Pattern: https://reactpatterns.com

Awesome React Resource: https://github.com/enaqx/awesome-react

Front-end Developer Roadmap: https://roadmap.sh/frontend

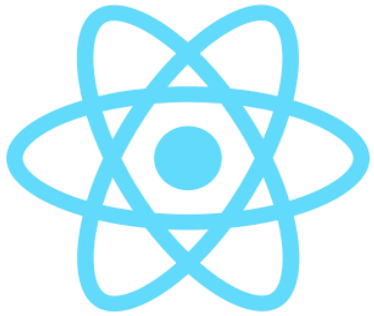
React Developer Roadmap: https://github.com/adam-golab/react-developer-roadmap

Conclusion: editor & tools

Recommend Editor: VSCode (or SublimeText)

Extension:

- Prettier
- Eslint
- Gitlens
- Path intellisense
- Auto rename tag
- ES7 React/Redux snippets



Chrome Extension:

- React Developer Tool
- Redux Developer Tool

Thank for your attention!