Front-End Development

React & Redux

#5 Components direct access

Purwadhika
Startup and Coding School

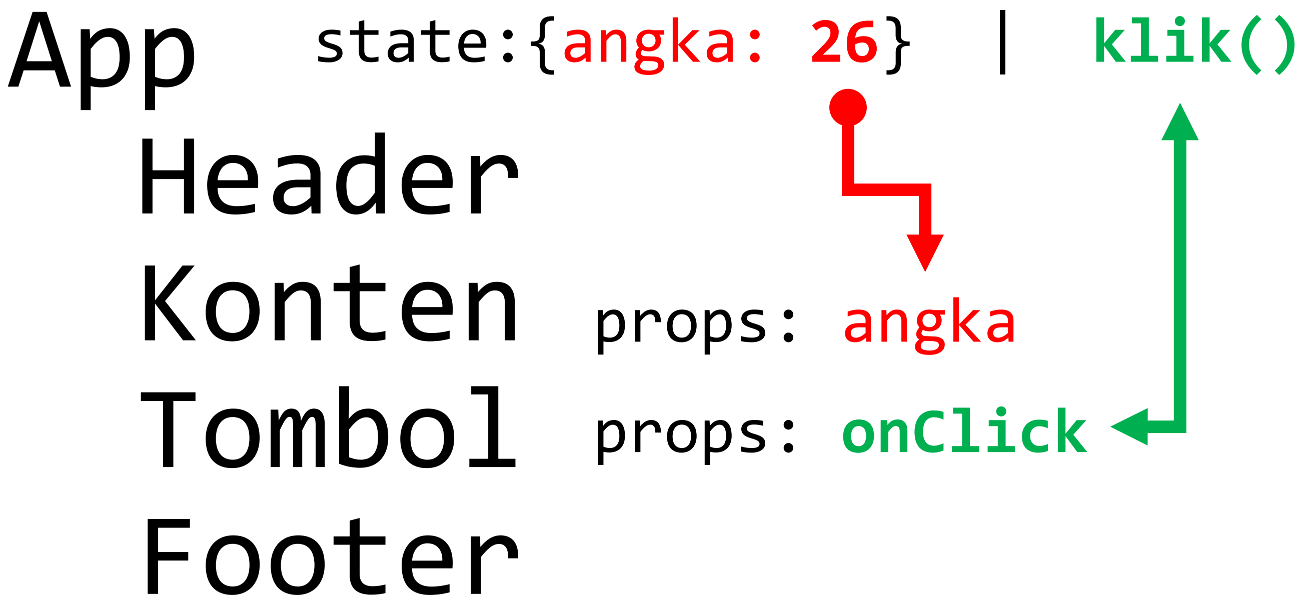
Redux

- Redux is a predictable state container for any JavaScript apps. It gives every components *direct access* to the data they need.
- Redux evolves the ideas of Flux, but avoids its complexity by taking cues from Elm.
- Installation
  (for React project, use 2 standard packages)

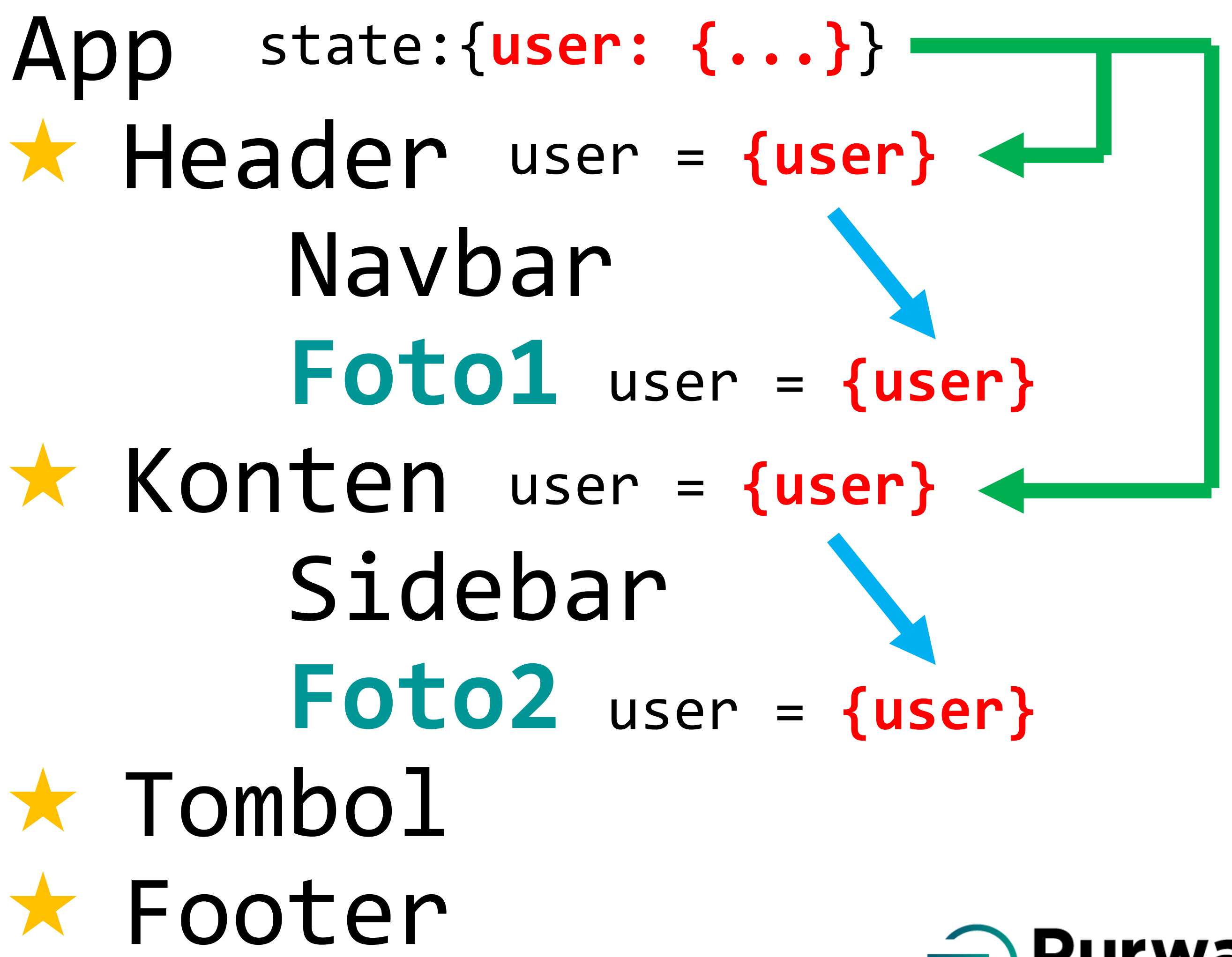
```
$ npm install redux react-redux --save
```

Why Should We Use Redux?

- In React (one way data flow), data is passed down the component tree via props & through a callback function to come back up the tree.



Why Should We Use Redux?



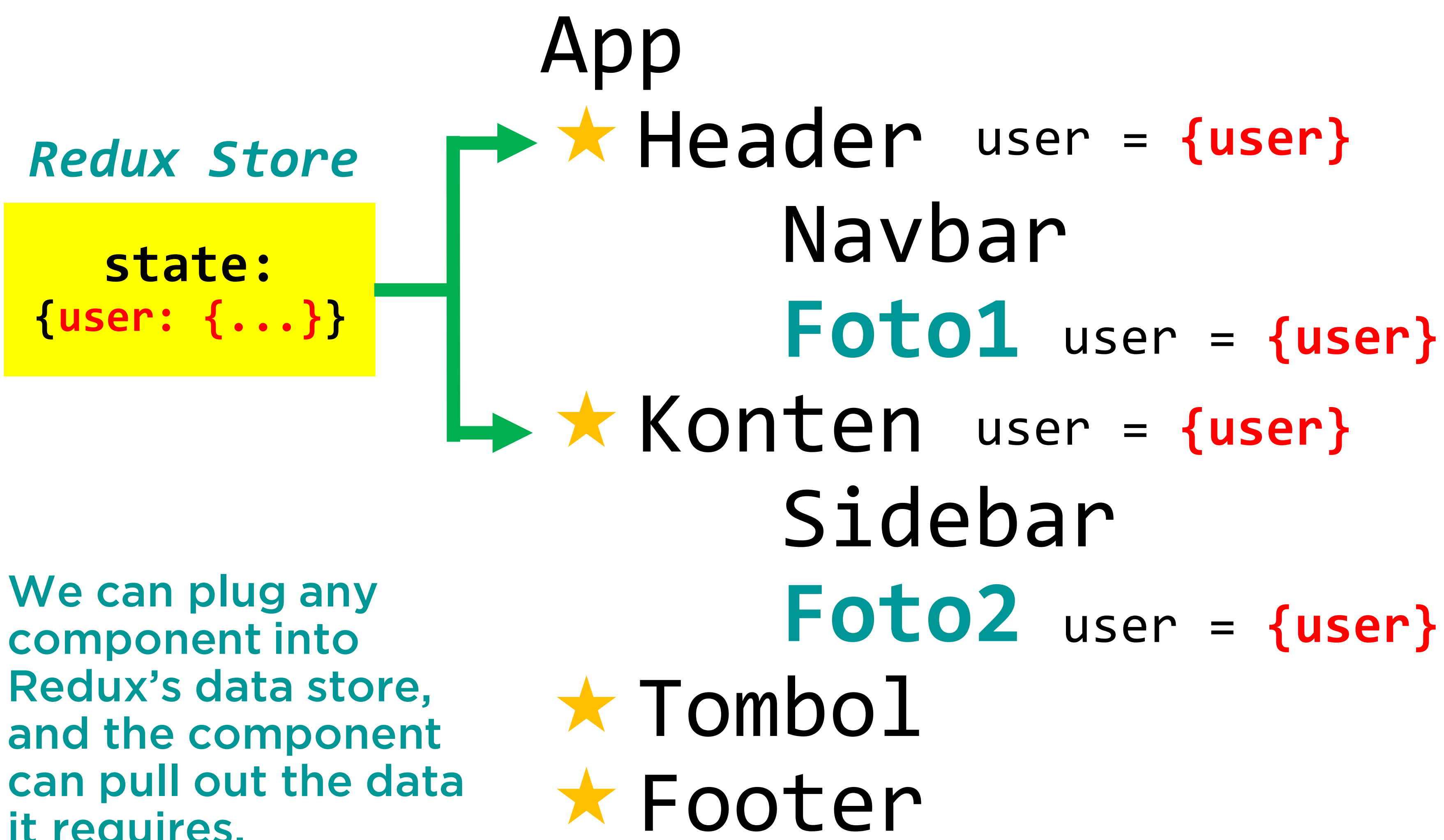
```
App  state:{user: {...}}
★ Header  user = {user}
      Navbar
      Foto1  user = {user}
★ Konten  user = {user}
      Sidebar
      Foto2  user = {user}
★ Tombol
★ Footer
```

Purwadhika
Startup and Coding School

Why Should We Use Redux?

- Getting data down like on the previous slide, is quite wasting time. More than that, it’s not a good software design.
- Intermediate components in the chain (on this case: Header & Konten) must accept & pass along props that they don’t care about.
- It would be nice if the components that didn’t need the data didn’t have to see it at all.
- This is the problem that Redux solves. It gives components direct access to the data needed.

Why Should We Use Redux?

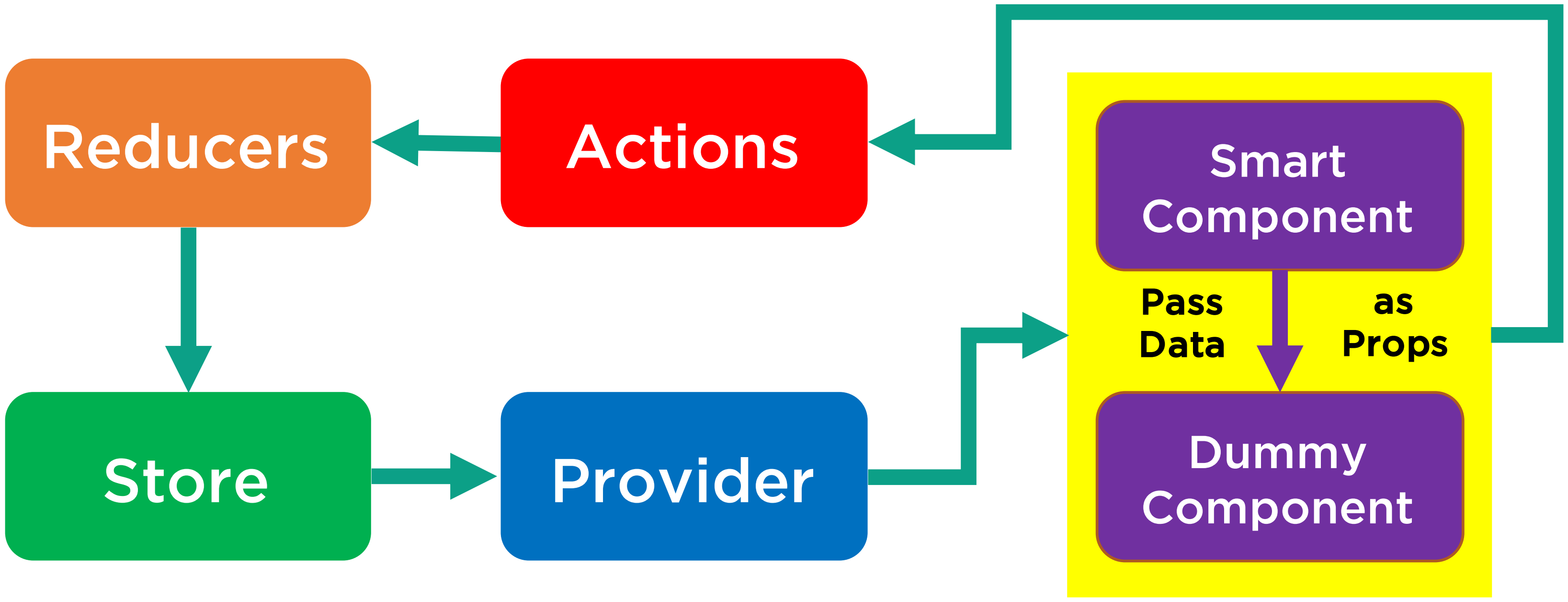


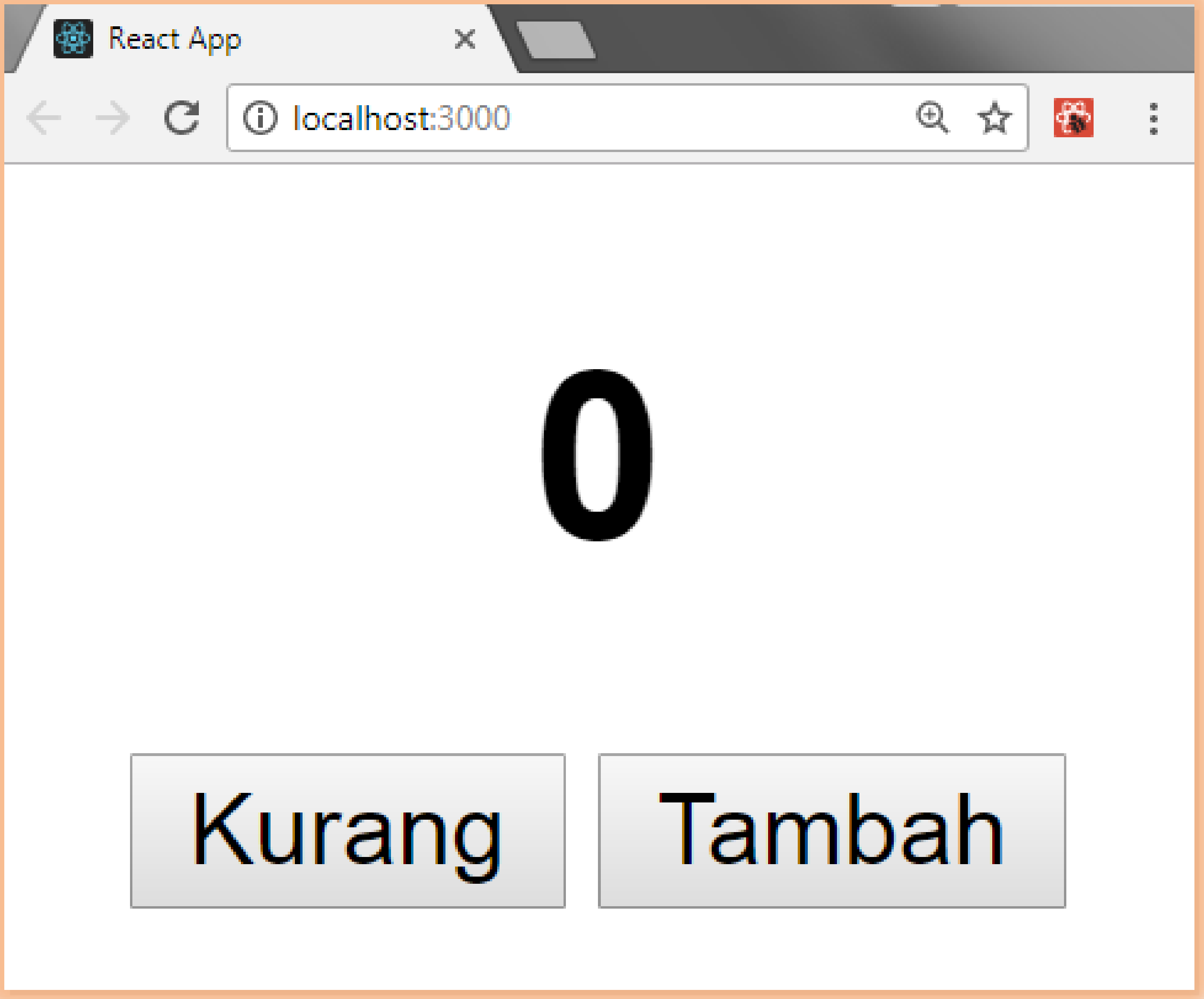
We can plug any component into Redux's data store, and the component can pull out the data it requires.

Basic Schema

- Install

```
$ npm install redux react-redux --save
```





Create without Redux!

Purwadhika
Startup and Coding School

src/App.js
#1 Basic Template

```
import React, { Component } from 'react';

class App extends Component {
state = { count: 0 }

increment = () => {
this.setState({
count: this.state.count + 1
});
}

decrement = () => {
this.setState({
count: this.state.count - 1
});
}
```

Purwadhika
Startup and Coding School

src/App.js
#1 Basic Template

```
render(){
return (
<div>
<center>
<h1>{this.state.count}</h1>
<div>
<button onClick = {this.decrement}>Kurang
</button>
<span> </span>
<button onClick = {this.increment}>Tambah
</button>
</div>
</center>
</div>
);
}
}

export default App;
```

Purwadhika
Startup and Coding School

src/index.js

#2 Basic template

```
import React from 'react';
import ReactDOM from 'react-dom';

import App from './App';

ReactDOM.render(<App />,
document.getElementById('root'));
```

Purwadhika
Startup and Coding School

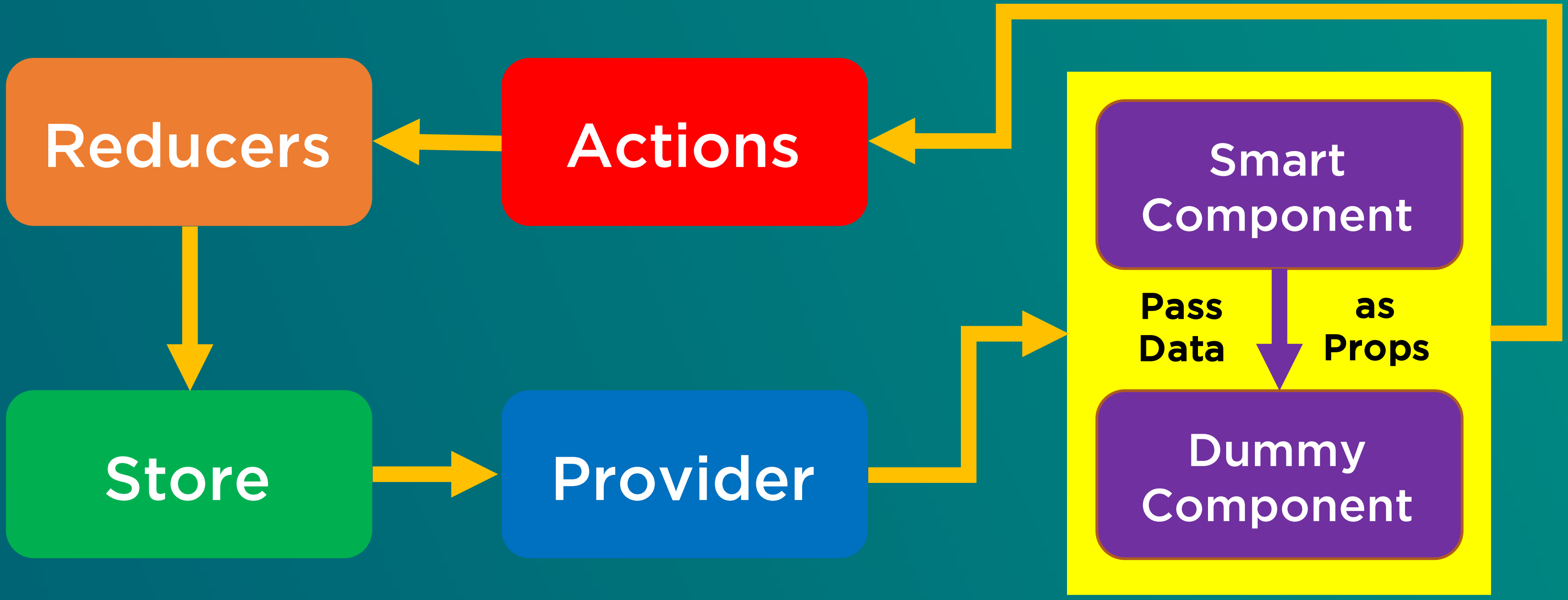
How to Use Redux On React Project

“let’s code step by step”

Purwadhika
Startup and Coding School

Remember this schema!

```
$ npm install redux react-redux --save
```



Purwadhika
Startup and Coding School

src/App.js

#1a Import Connect Redux
#1b Delete setState on Func

```
import React, { Component } from 'react';
import { connect } from 'react-redux';

class App extends Component {
state = { count: 0 }

increment = () => {
// fill in later
}

decrement = () => {
// fill in later
}
```

Purwadhika
Startup and Coding School

src/App.js

#1c Insert this.props
#1d Connect to Redux

```
render(){
return (
 <div>
   <center>
     <h1>{this.props.count}</h1>
     <div>
       <button onClick = {this.decrement}>Kurang</button>
       <span> </span>
       <button onClick = {this.increment}>Tambah</button>
     </div>
  </center>
 </div>
);
}
}

function mapStateToProps(state){
  return {
    count: state.count
  };
}

export default connect(mapStateToProps)(App)
```

src/index.js

#2 Provide a Store

```
import React from 'react';
import ReactDOM from 'react-dom';
import App from './App';

import { Provider } from 'react-redux';

ReactDOM.render(<Provider>
<App />
</Provider>,
document.getElementById('root'));
```

Purwadhika
Startup and Coding School

src/index.js
#3 Create a Store

```
import React from 'react';
import ReactDOM from 'react-dom';
import App from './App';

import { Provider } from 'react-redux';

import { createStore } from 'redux';

const store = createStore();

ReactDOM.render(
<Provider store={store}>
<App />
</Provider>, document.getElementById('root'));
```

Purwadhika
Startup and Coding School

src/index.js

#4 Create a Reducer

```
import React from 'react';
import ReactDOM from 'react-dom';
import App from './App';

import { Provider } from 'react-redux';
import { createStore } from 'redux';

function reducer(){
  return {
    count: 42
  };
}

const store = createStore(reducer);

ReactDOM.render(
<Provider store={store}>
<App />
</Provider>, document.getElementById('root'));
```

src/index.js

#5 State and Action Param

```
import React from 'react';
import ReactDOM from 'react-dom';
import App from './App';

import { Provider } from 'react-redux';
import { createStore } from 'redux';

const initialState = {
  count: 0
};

function reducer(state=initialState, action){
  return state;
}

const store = createStore(reducer);

ReactDOM.render(
<Provider store={store}>
<App />
</Provider>, document.getElementById('root'));
```

src/index.js

#6 Insert Action

```
. . . . . .

function reducer(state=initialState, action){
  switch(action.type){
    case 'INCREMENT':
      return {
      count: state.count + 1
    };
    case 'DECREMENT':
      return {
      count: state.count - 1
    };
    default:
      return state;
    }
  }

. . . . . .
```

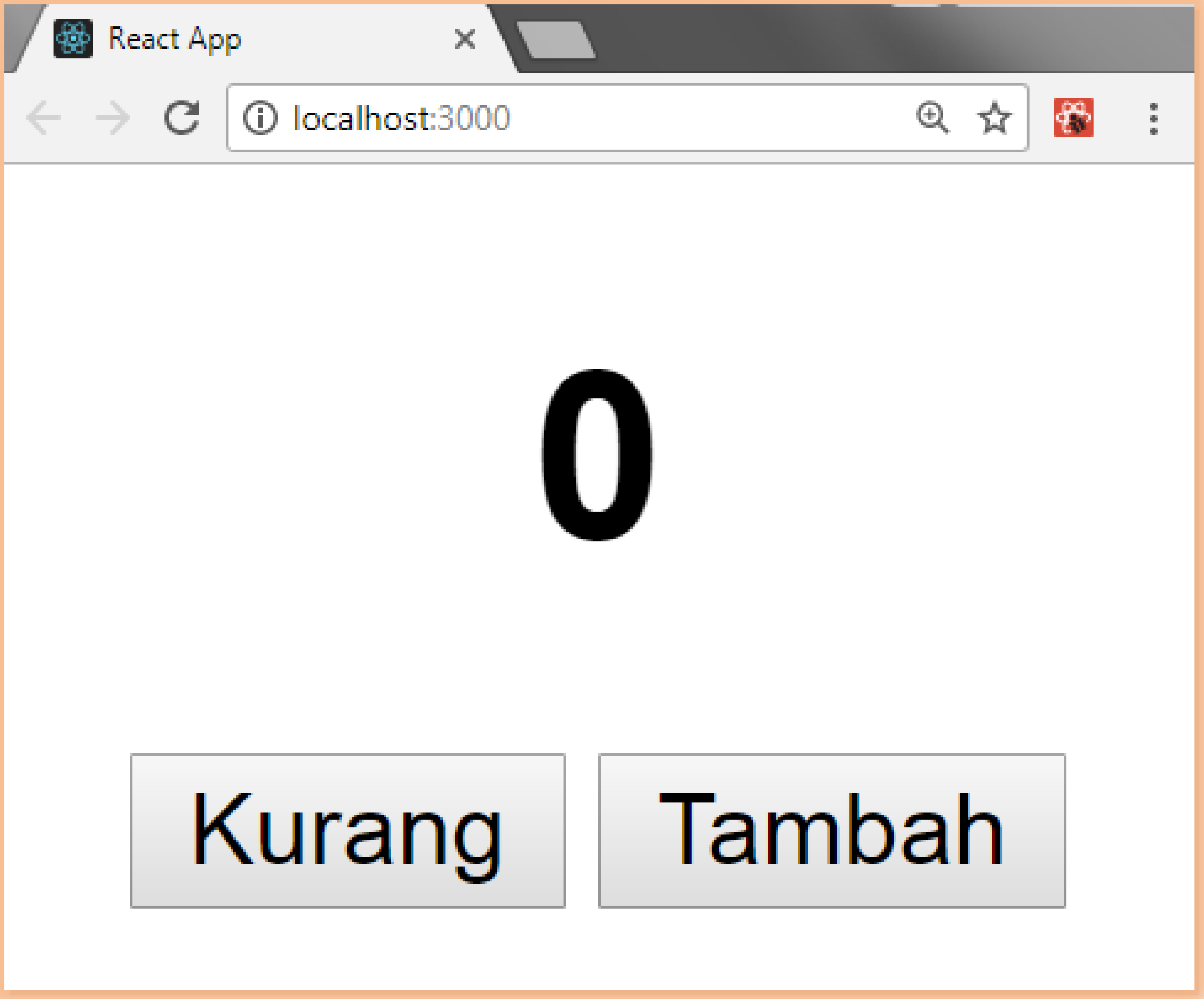
Purwadhika
Startup and Coding School

src/App.js

#7 Dispatch Action

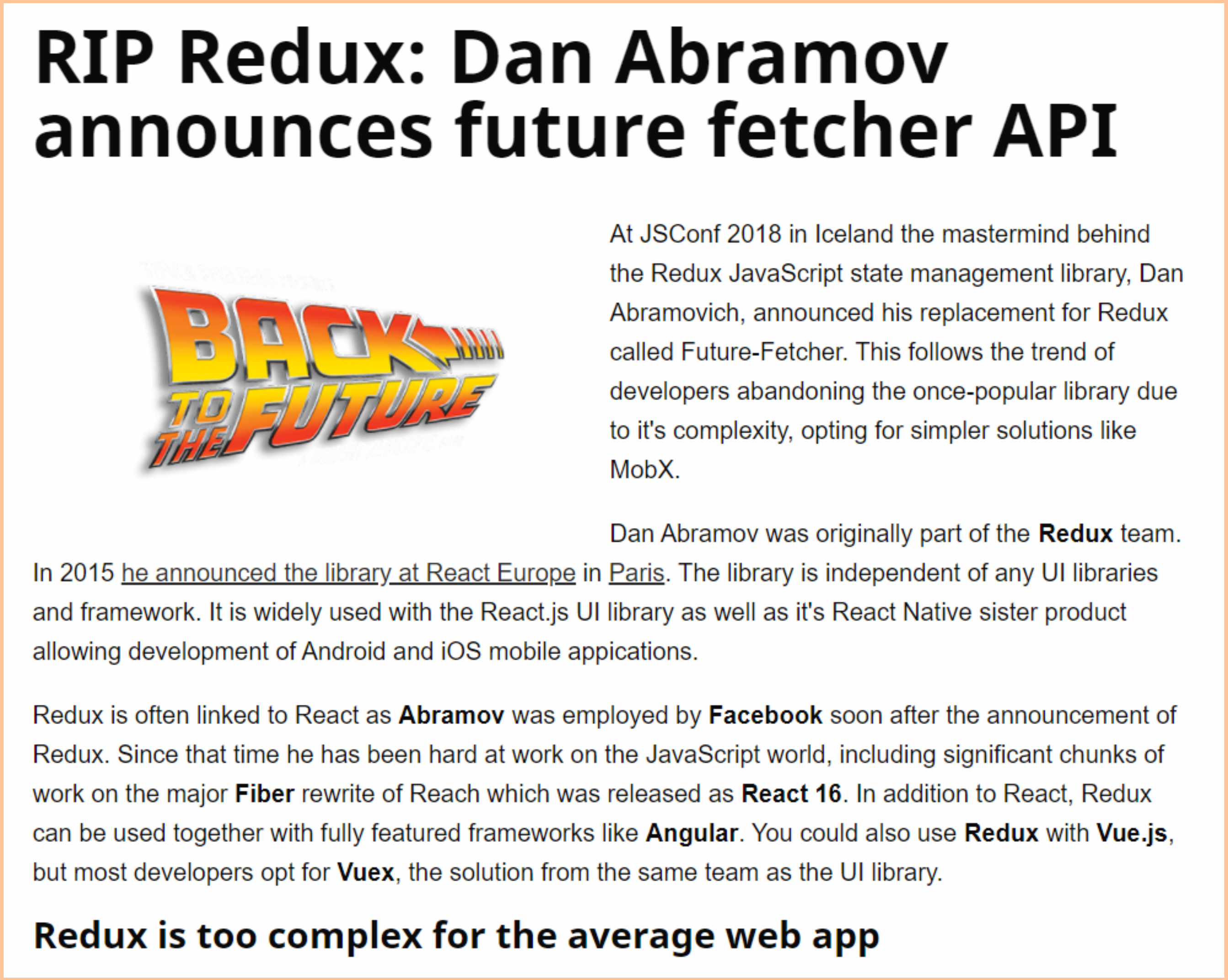
```
. . . . . .

class App extends Component {
state = { count: 0 }

increment = () => {
this.props.dispatch({type:'INCREMENT'});
}

decrement = () => {
this.props.dispatch({type:'DECREMENT'});
}

render(){

. . . . . .
```

Purwadhika
Startup and Coding School



Redux’s in da house!

Purwadhika
Startup and Coding School



RIP Redux: Dan Abramov announces future fetcher API

At JSConf 2018 in Iceland the mastermind behind the Redux JavaScript state management library, Dan Abramovich, announced his replacement for Redux called Future-Fetcher. This follows the trend of developers abandoning the once-popular library due to it's complexity, opting for simpler solutions like MobX.

Dan Abramov was originally part of the **Redux** team. In 2015 he announced the library at React Europe in Paris. The library is independent of any UI libraries and framework. It is widely used with the React.js UI library as well as it's React Native sister product allowing development of Android and iOS mobile appications.

Redux is often linked to React as **Abramov** was employed by **Facebook** soon after the announcement of Redux. Since that time he has been hard at work on the JavaScript world, including significant chunks of work on the major **Fiber** rewrite of Reach which was released as **React 16**. In addition to React, Redux can be used together with fully featured frameworks like **Angular**. You could also use **Redux** with **Vue.js**, but most developers opt for **Vuex**, the solution from the same team as the UI library.

Redux is too complex for the average web app

https://react-etc.net/entry/rip-redux-dan-abramov-announces-future-fetcher

Purwadhika
Startup and Coding School

Front-End Development

React & Redux

#5 Components direct access

Purwadhika
Startup and Coding School