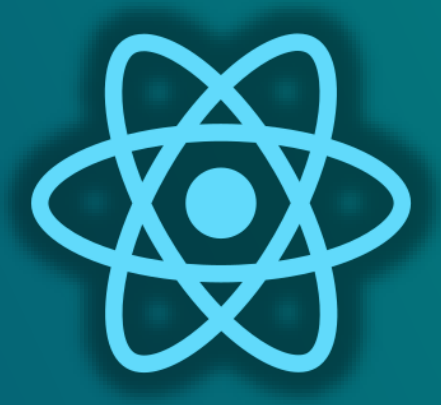
Front-End Development



React & Redux

Components direct access

Purwadhika
Startup and Coding School

Redux

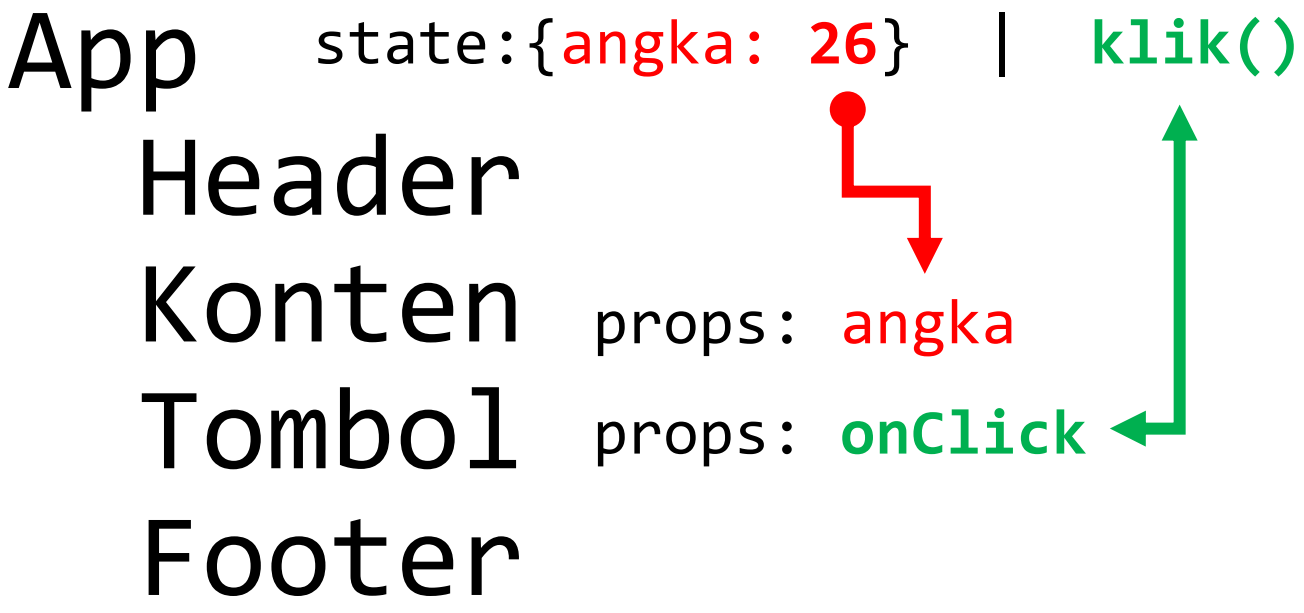
- Redux is a predictable state container for any JavaScript apps. It gives every components *direct access* to the data they need.
- Redux evolves the ideas of Flux, but avoids its complexity by taking cues from Elm.
- Installation
  (for React project, use 2 standard packages)

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

Purwadhika
Startup and Coding School

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.

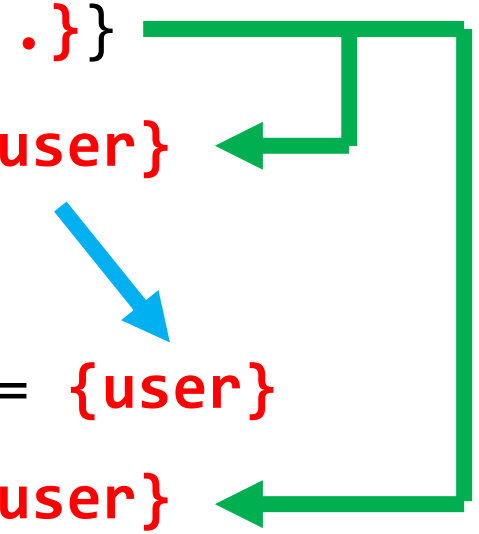


Purwadhika
Startup and Coding School

Why Should We Use Redux?

App state:{user: {...}}

- ★ Header user = {user}
  - Navbar
  - Foto1 user = {user}
- ★ Konten user = {user}
  - Sidebar
  - Foto2 user = {user}
- ★ Tombol
- ★ Footer



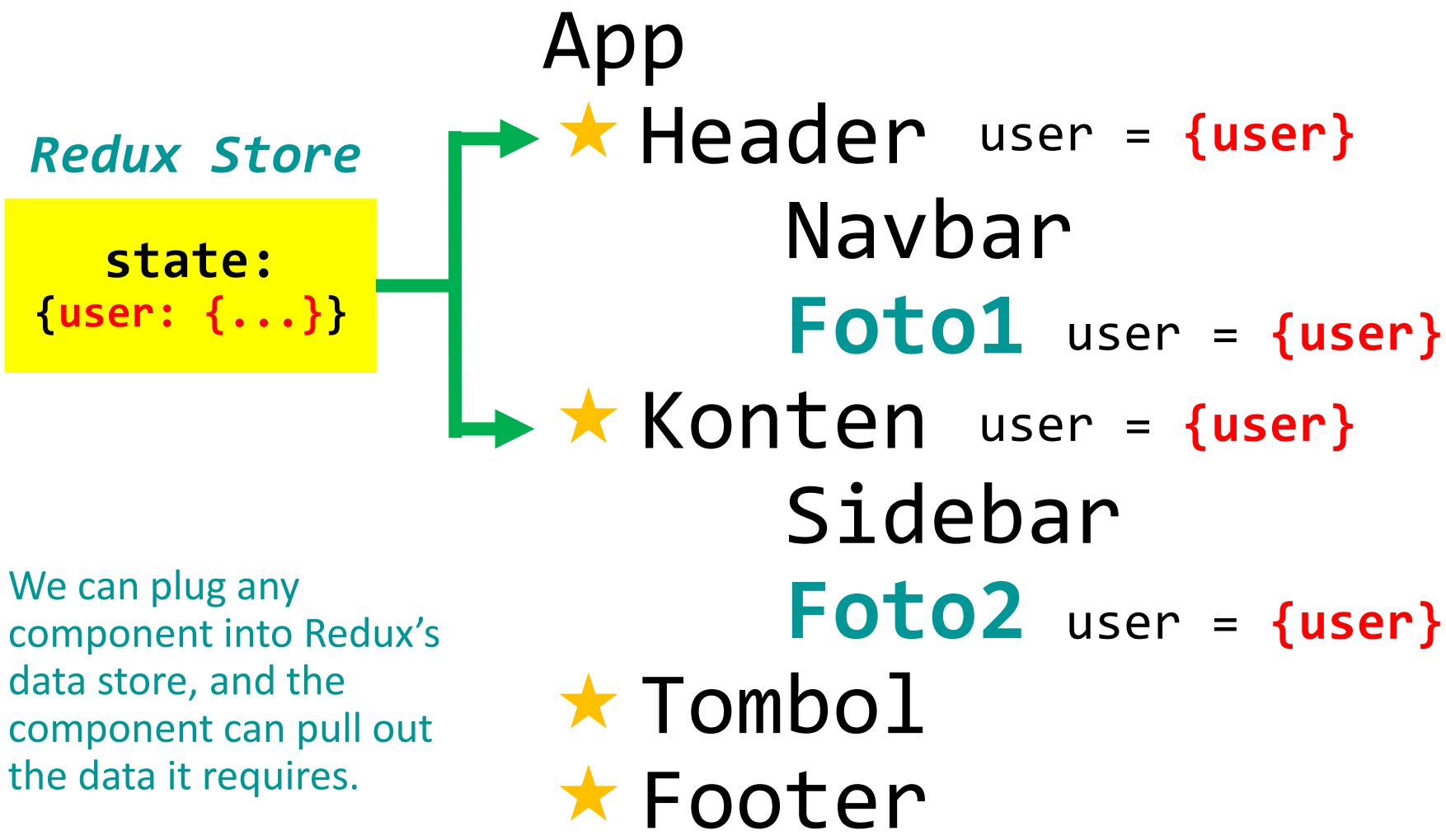
Purwadhika
Startup and Coding School

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.

Purwadhika
Startup and Coding School

Why Should We Use Redux?



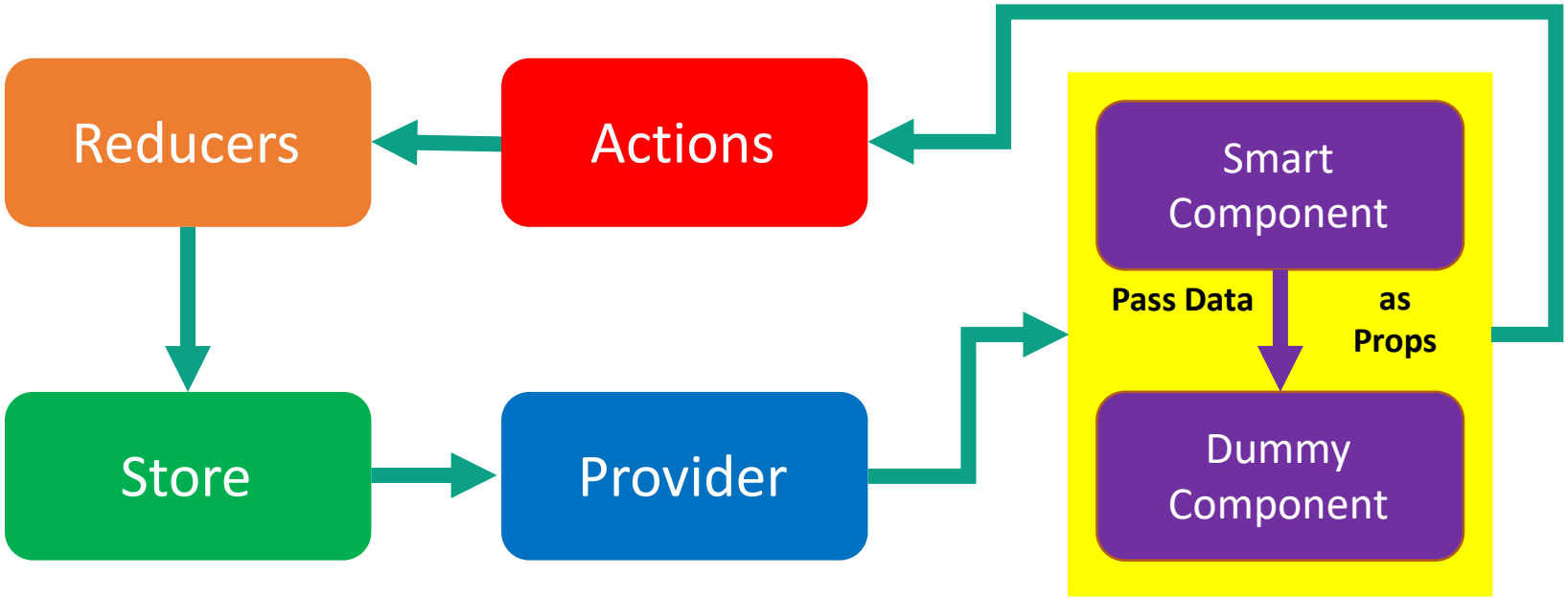
We can plug any component into Redux's data store, and the component can pull out the data it requires.

Purwadhika
Startup and Coding School

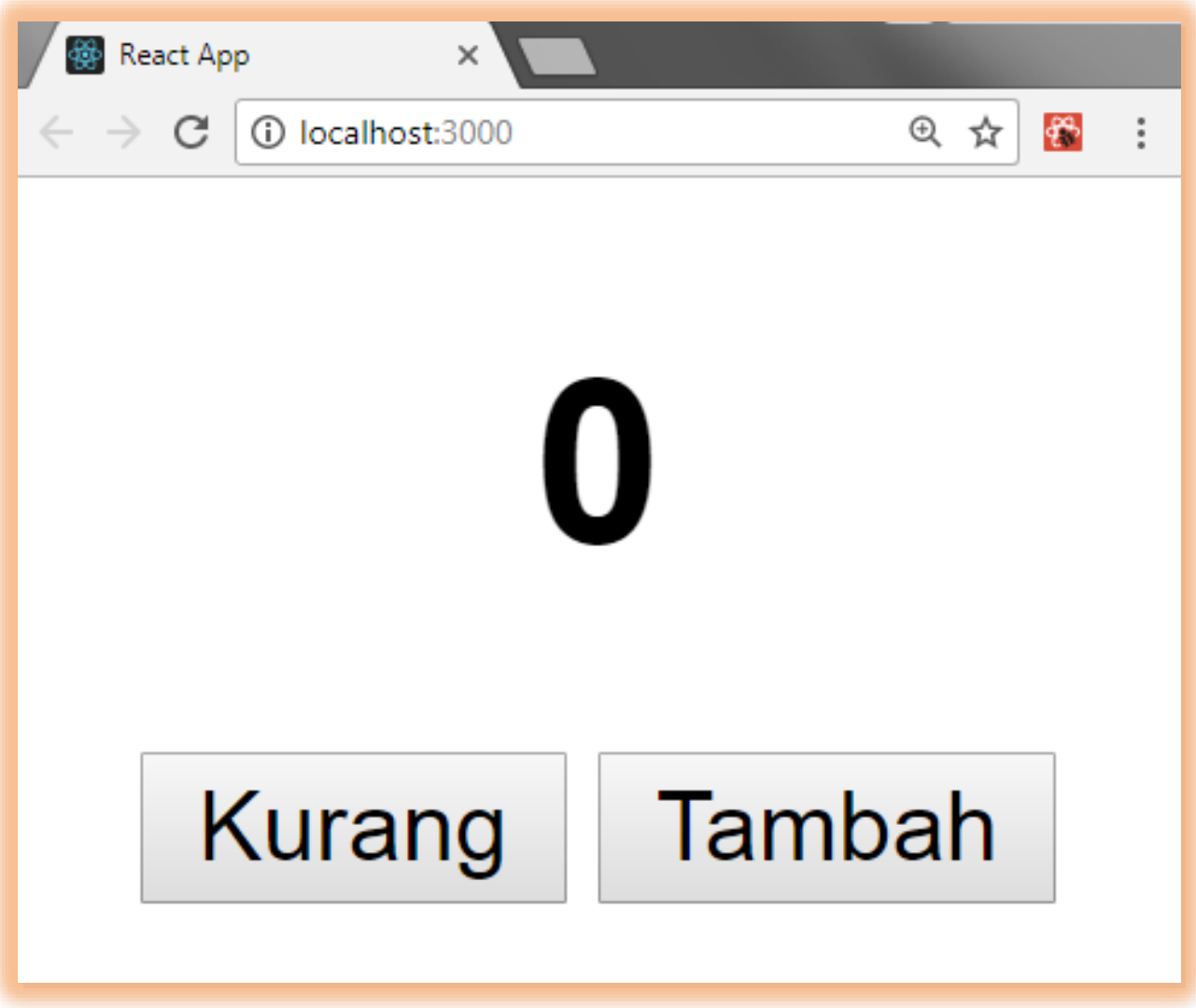
Basic Schema

- Install

  $ npm install redux react-redux --save



Purwadhika
Startup and Coding School



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

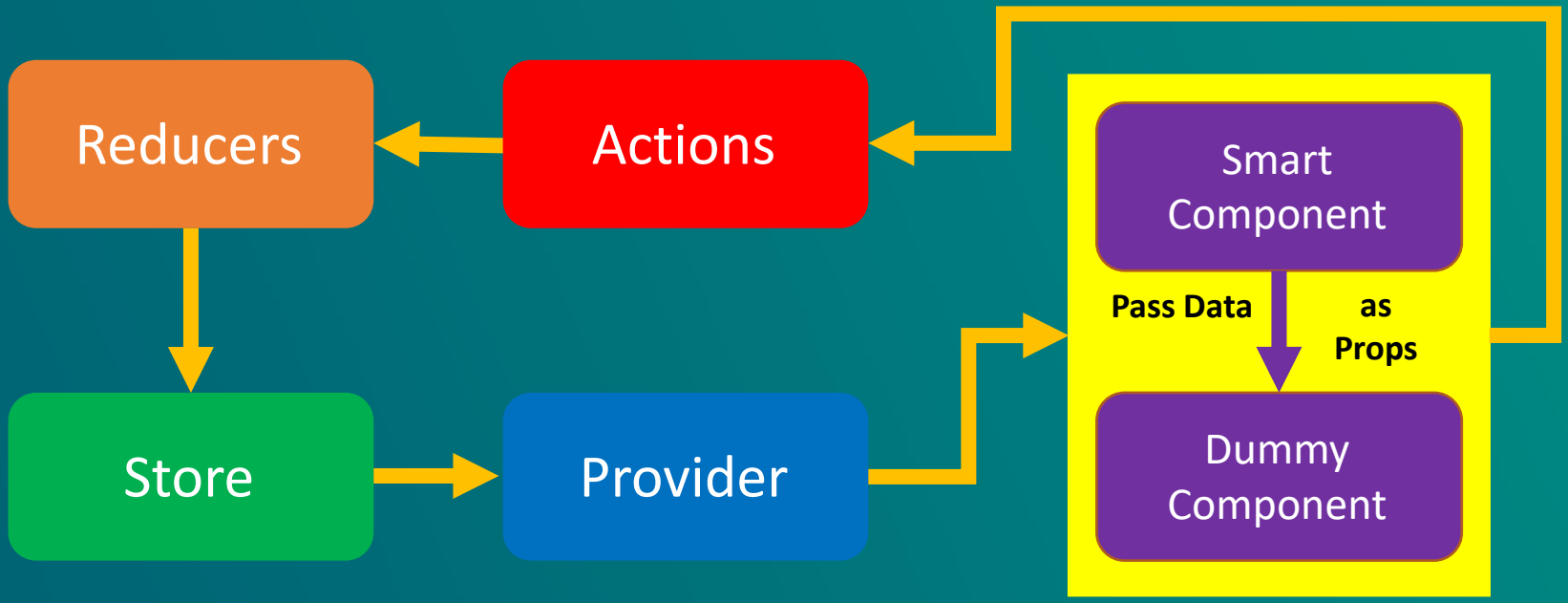
How to Use Redux On React Project

“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;
    }
  }

. . . . . .
```

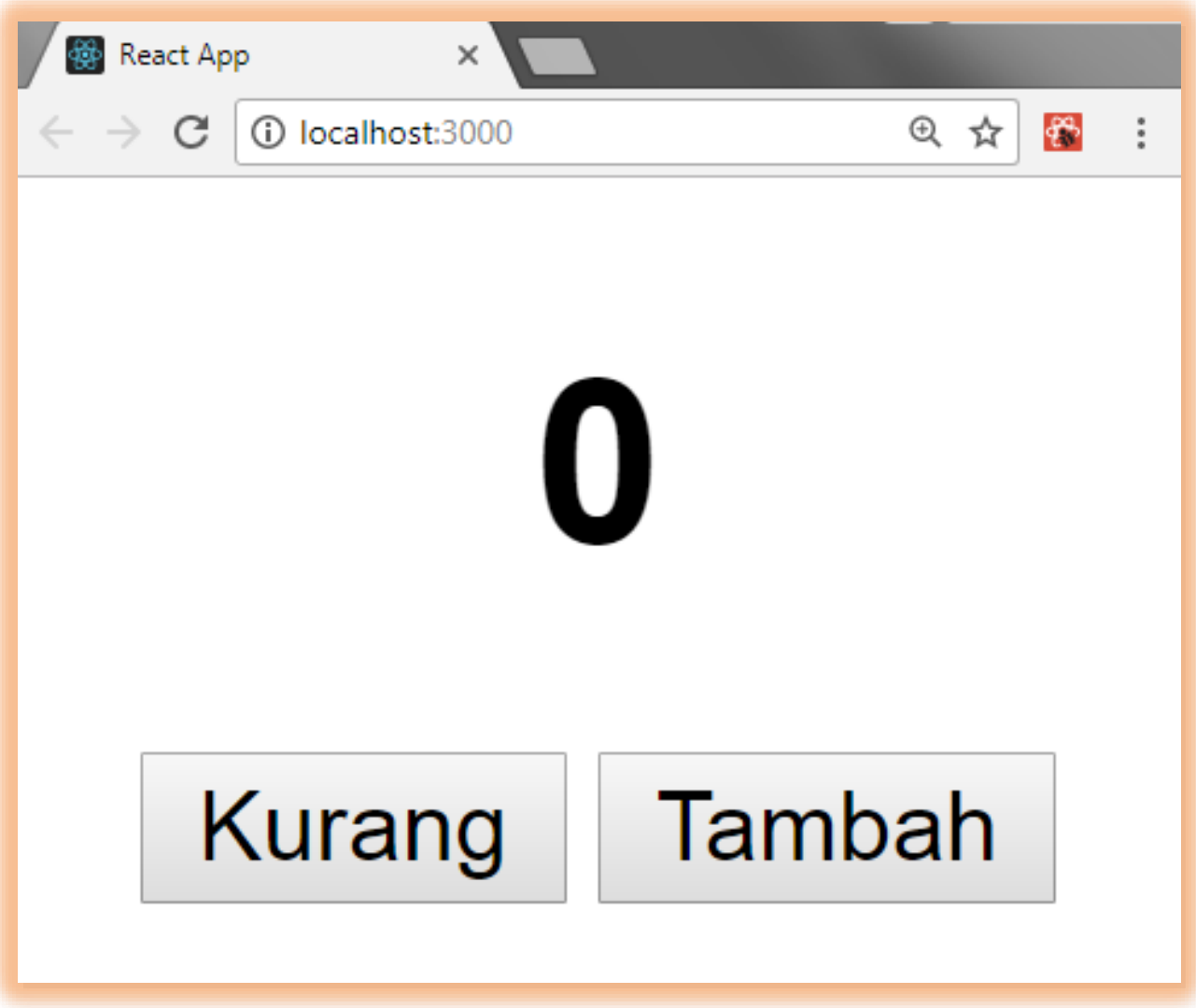
Purwadhika
Startup and Coding School

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(){

. . . . . .
```

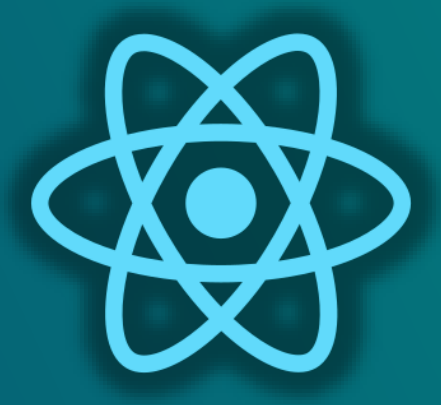
Purwadhika
Startup and Coding School



Redux's in da house!

Purwadhika
Startup and Coding School

Front-End Development



React & Redux

Components direct access

Purwadhika
Startup and Coding School