Hello Redux

Dr. Jose Annunziato

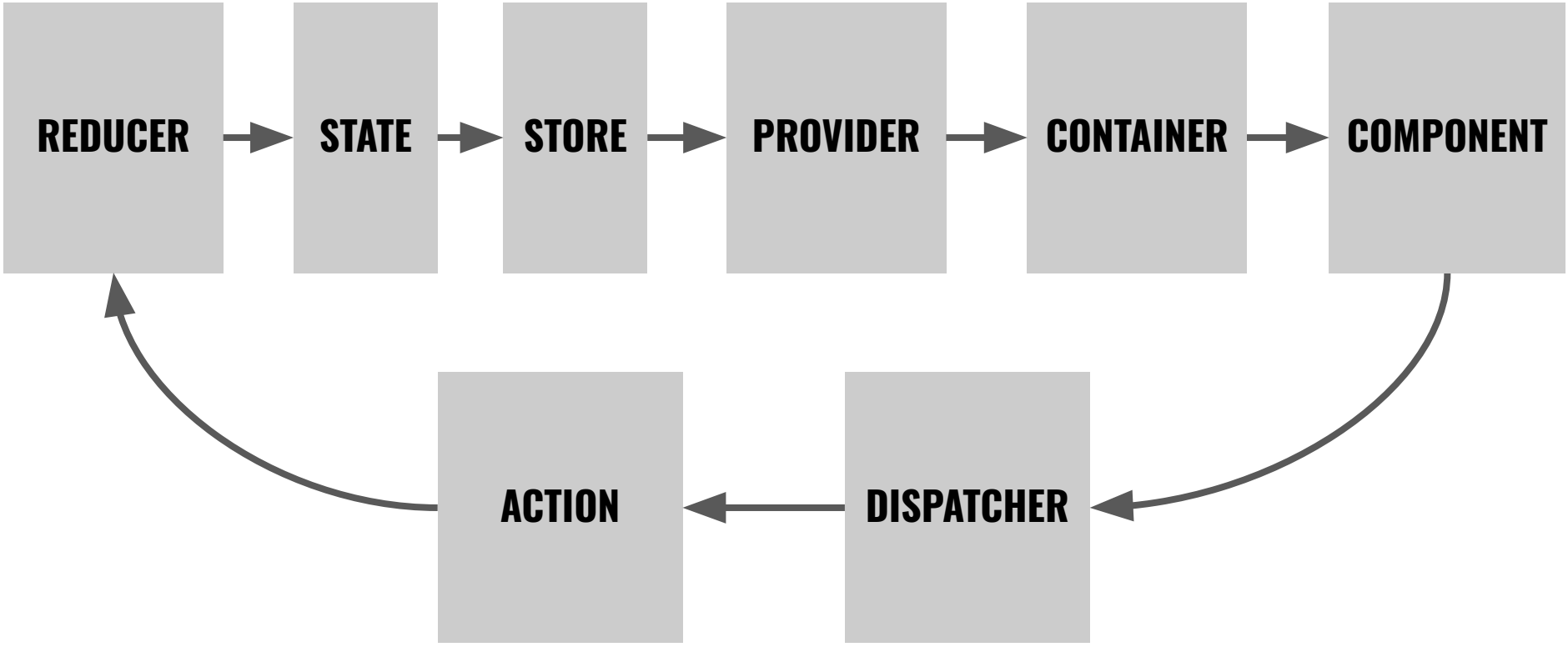
What is Redux?

Redux is a JavaScript library for managing application state

Decouples state management from rendering logic

Allows writing simpler, stateless React components: easier to test

Redux


REDUCER
STATE
STORE
PROVIDER
CONTAINER
COMPONENT
ACTION
DISPATCHER

INSTALL

Dr. Jose Annunziato

Create, configure simple React app

Use create-react-app to create react application

```
create-react-app hello-redux
```

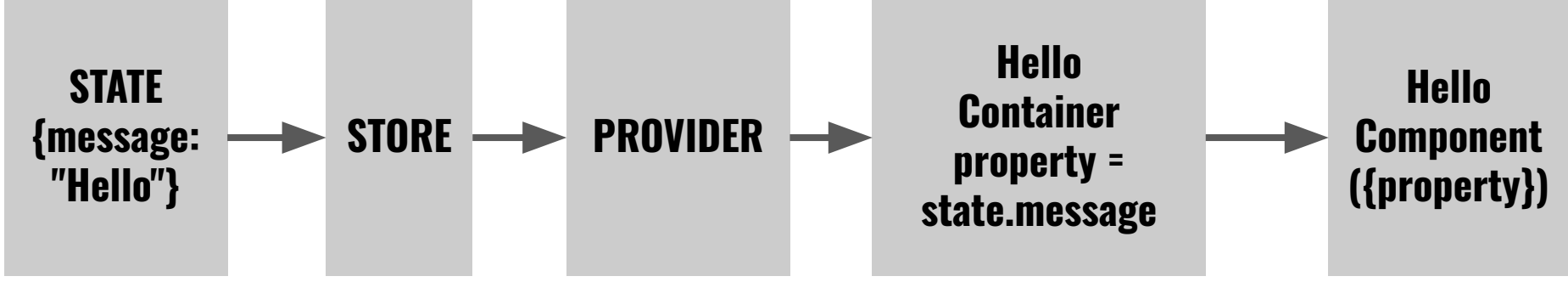
Install redux

```
cd hello-redux
npm install redux --save
npm install react-redux --save
npm start
```

HELLO 1

Dr. Jose Annunziato

Redux


STATE
{message:
"Hello"}
STORE
PROVIDER
Hello
Container
property =
state.message
Hello
Component
({property})

Import React core libraries

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

Create a store to track state

```
import React from 'react';
import ReactDOM from 'react-dom';
import {createStore} from 'redux'

const store = createStore(() => ({message: "Hello World!"}))
```

Stores keep track of state as JSON objects

State is generated by a ***reducer*** function

Components render state variables

```
import React from 'react';
import ReactDOM from 'react-dom';
import {createStore} from 'redux'

const store = createStore(() => ({message: "Hello World!"}))

const HelloComponent = ({helloProperty}) =>
  <h1>{helloProperty}</h1>
```

State variables are eventually rendered by React components

Components receive state through properties

Wrap component in container

```
import React from 'react';
import ReactDOM from 'react-dom';
import {createStore} from 'redux'
import {connect} from 'react-redux'
const store = createStore(() => ({message: "Hello World!"}))

const HelloComponent = ({helloProperty}) =>
  <h1>{helloProperty}</h1>

const HelloContainer = connect(
  state => ({helloProperty: state.message})
)(HelloComponent)
```

Containers map state variables to component properties

Provide store to container

```
import React from 'react';
import ReactDOM from 'react-dom';
import {createStore} from 'redux'
import {connect, Provider} from 'react-redux'
const store = createStore(() => ({message: "Hello World!"}))

const HelloComponent = ({helloProperty}) =>
  <h1>{helloProperty}</h1>

const HelloContainer = connect(
  state => ({helloProperty: state.message})
)(HelloComponent)

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

Providers interface Redux framework with React components

HELLO 2

Dr. Jose Annunziato

Create a simple Hello component

Create component **src/components/Hello.js**

```
import React from 'react'

const Hello = () =>
  <h1>Hello</h1>

export default Hello
```

Render simple Hello component

Render the simple **Hello** component

```
import React from 'react'
import ReactDOM from 'react-dom'
import Hello from './components/Hello'

ReactDOM.render(
  <Hello/>,
  document.getElementById("root")
)
```

STATE & REDUCERS

Dr. Jose Annunziato

Connecting state to component

Parameterize Hello component

Parameterize **src/components/Hello.js**

```
import React from 'react'

const Hello = ({message}) =>
  <h1>Hello {message}</h1>

export default Hello
```

Manage state in Redux reducer

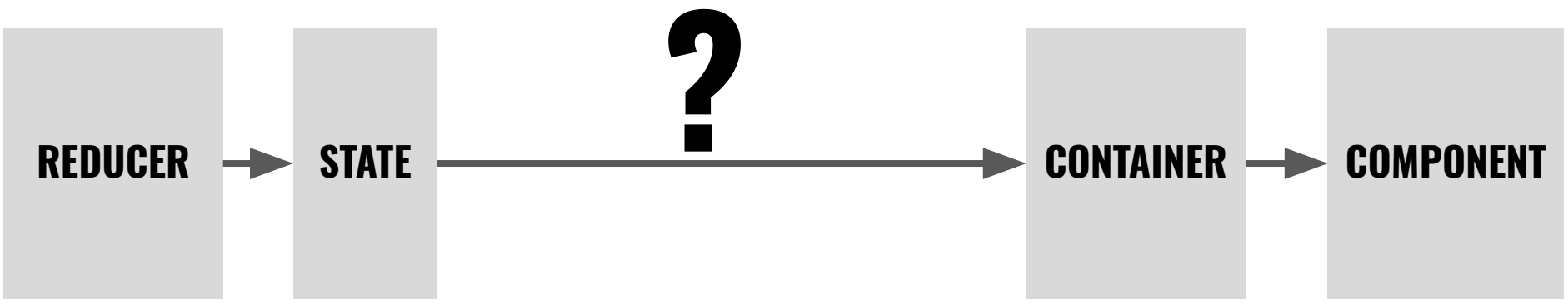
Create reducer **src/reducers/hello.js**

```
const hello = () => (
    {
        message: 'World'
    }
)
export default hello
```

CONTAINERS

Dr. Jose Annunziato

Connecting state to component



Containers map state to component

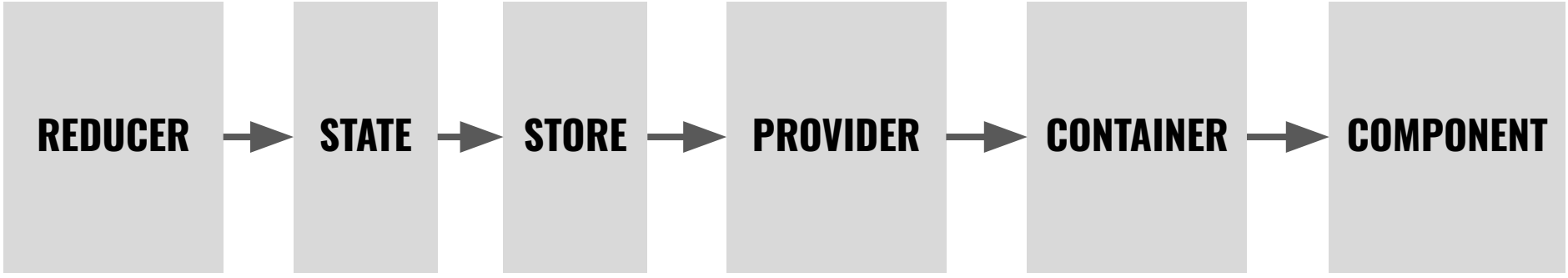
Create **src/containers/HelloContainer.js**

```
import React from 'react'
import Hello from '../components/Hello'
import {connect} from 'react-redux'
const stateToPropertyMapper = state => ({
   message: state.message
})
const HelloContainer = connect
(stateToPropertyMapper)(Hello)
export default HelloContainer
```

STORE PROVIDER

Dr. Jose Annunziato

Connecting state to component



Import reducer & container

In **src/index.js**, import reducer and container

```
import React from 'react'
import ReactDOM from 'react-dom'
~~import Hello from './components/Hello'~~
import hello from './reducers/hello'
import HelloContainer from
                    './containers/HelloContainer'
import {createStore} from 'redux'
import {Provider} from 'react-redux'
```

Create, provide store to container

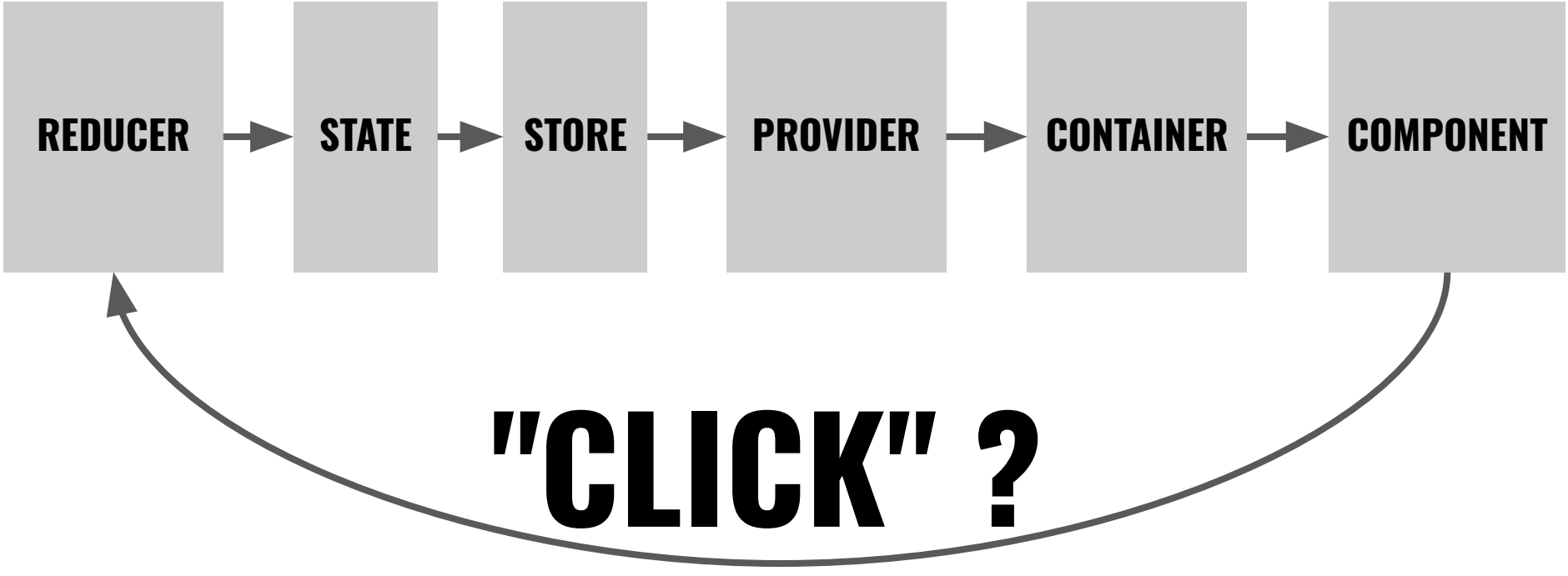
In **src/index.js** create store, provide to container

```
const store = createStore(hello)
ReactDOM.render(
  <Provider store={store}>
    <HelloContainer/>
  </Provider>,
  document.getElementById("root")
)
```

EVENTS

Dr. Jose Annunziato

Connecting state to component



Components generate events

In **src/index.js** generate events, don't handle it

```
const Hello = ({message, handleClick}) =>
  <div>
    <h1>Hello {message}</h1>
    <button onClick={() => handleClick()}>
      Click</button>
  </div>
```

Reducer generates new state

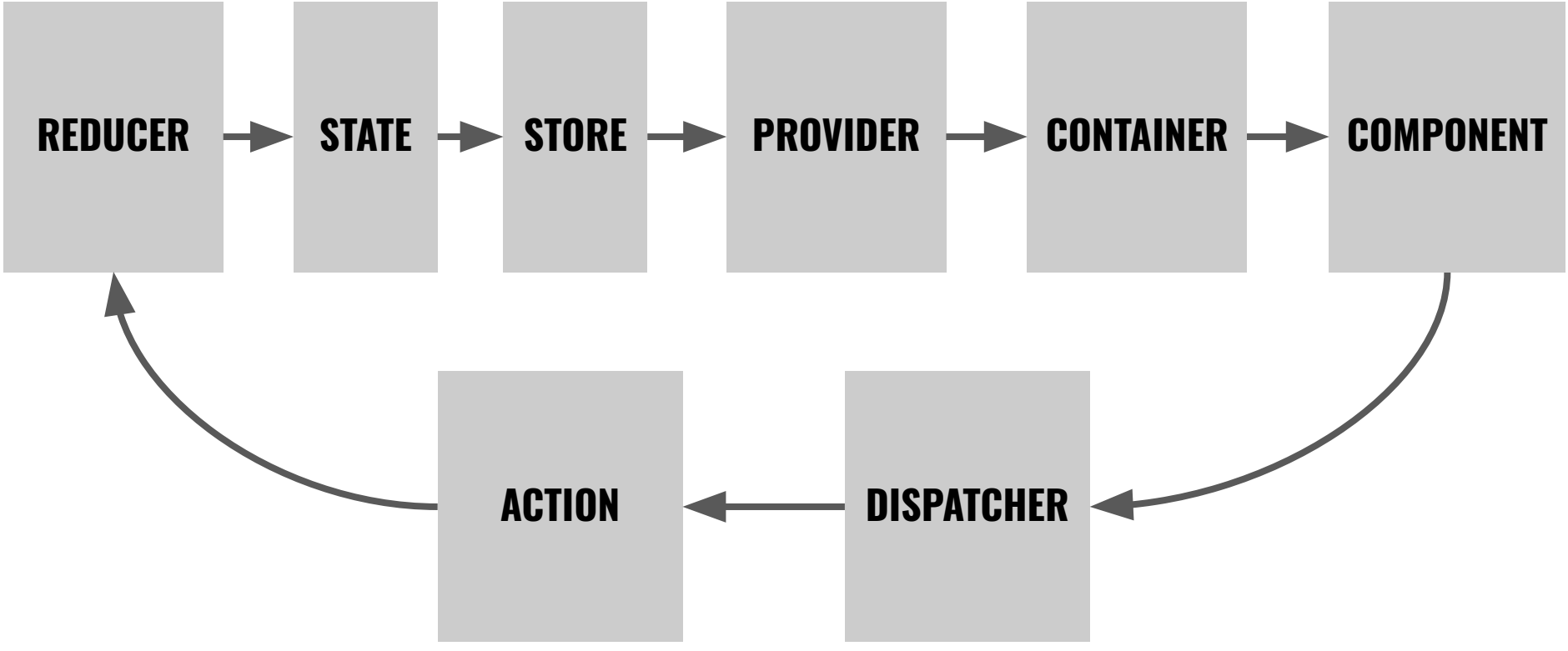
Reducer generates new state based on "action"

```
const hello = (state, action) => {
   switch (action.type) {
      case "CLICK":
         return { message: "Click" }
      default:
         return { message: "World" }
}}
export default hello
```

ACTIONS & DISPATCHERS

Dr. Jose Annunziato

Connecting state to component



Map dispatcher to properties

In **HelloContainer.js** map dispatcher to property

```
const dispatcherToPropertyMapper = dispatch => ({
handleClick: () =>
      dispatch({ type: "CLICK" })
})

const HelloContainer = connect
(stateToPropertyMapper, dispatcherToPropertyMapper)(Hello)
```

PARAMETERS

Dr. Jose Annunziato

Send arguments in components

```
import React from 'react'
const Hello = ({message, setMessage})
    => <div> <h1>Hello {message}</h1>
   <button onClick={() =>
         setMessage("Alice")}>
    Alice</button></div>
```

Map properties to dispatchers

```
const dispatcherToPropertyMapper =
    dispatch => ({
  handleClick: () =>
      dispatch({  type: "CLICK" }),
  setMessage: message =>
      dispatch({   type: "SET_MESSAGE",
                   message: message })
})
```

New states based on action params

```
const hello = (state, action) => {
   switch (action.type) {
      case "SET_MESSAGE":
         return { message: action.message }
      case "CLICK":
         return { message: "Click" }
      default:
         return { message: "World" }
}}
```