500Tech

Intro to React & Redux

http://500Tech.com

Boris Dinkevich

boris@500tech.com

Boris Dinkevich

CoFounder @ **500Tech**

- React evangelist
- Past Angular hippie
- CoAuthor of
The Complete Redux Book

ES2015 (ES6)

Const & Let

Strings

Arrow functions

Destructuring

A WORD ON TOOLS

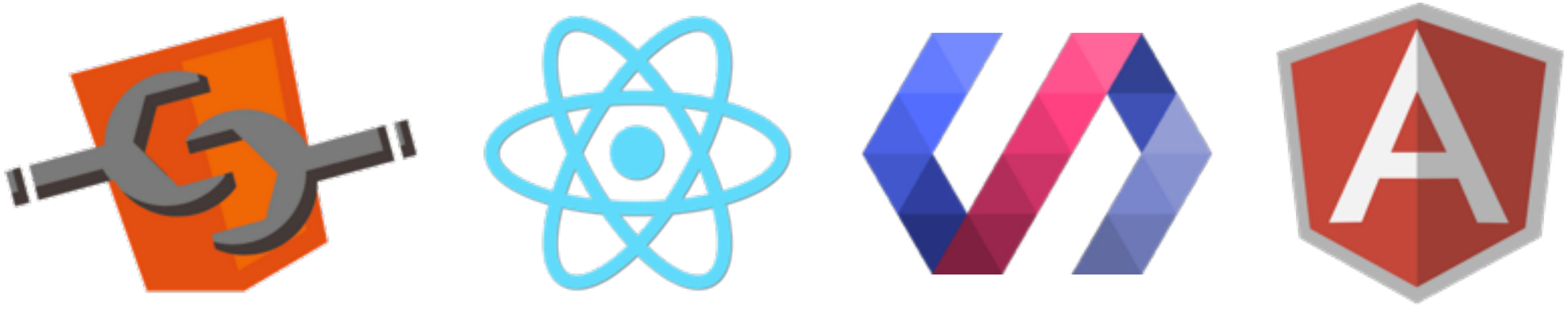
npm - package repository

babel - transpiler

webpack - build tool

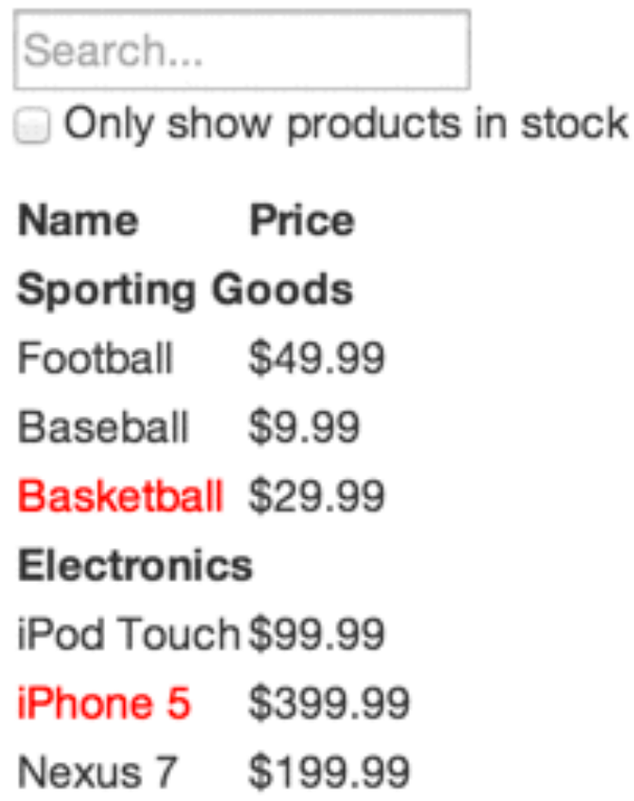
BACK TO REACT

HISTORY

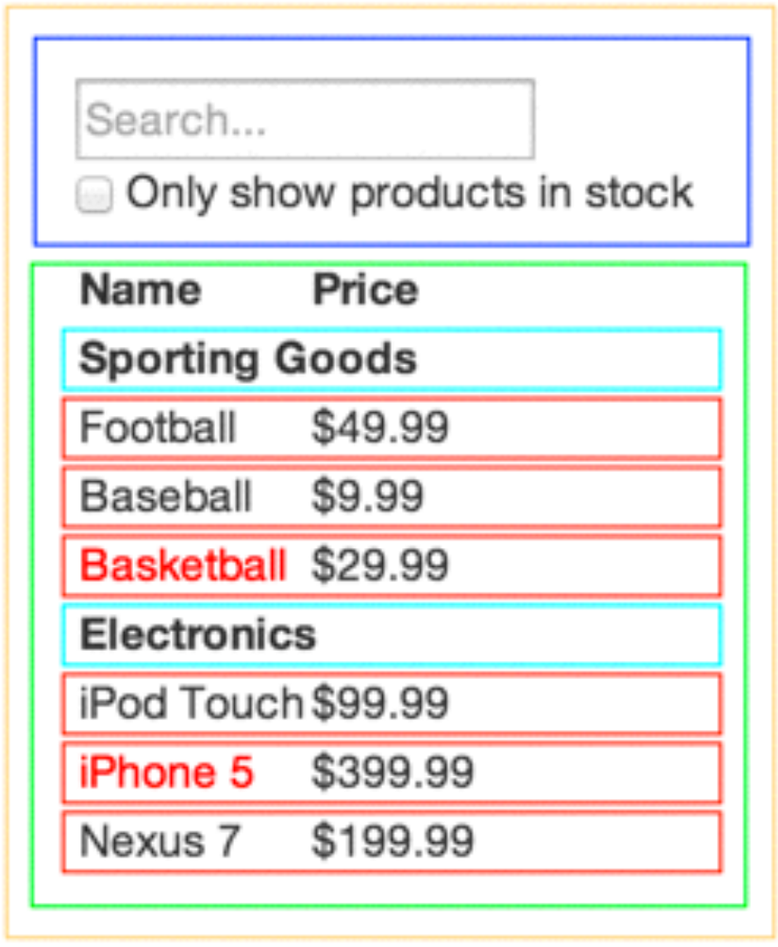


Component Driven Development

Thinking in components



Thinking in components



Lifecycle methods

Mount

componentWillMount → Angular PreLink

componentDidMount → Angular PostLink

Update

componentWillUpdate

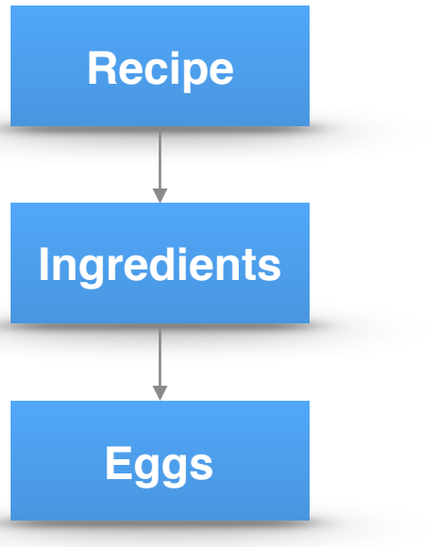
componentDidUpdate

Unmount

componentWillUnmount → $scope.$on('destroy')

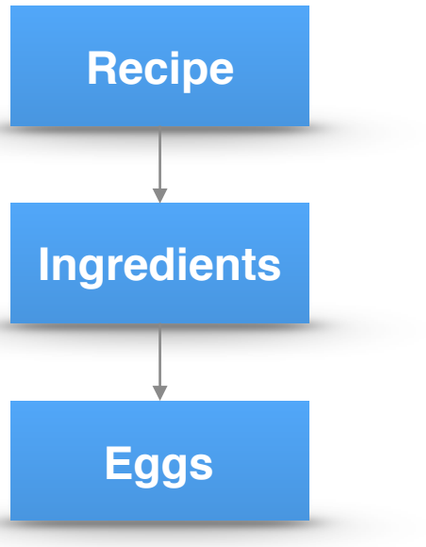
Virtual DOM

Virtual DOM

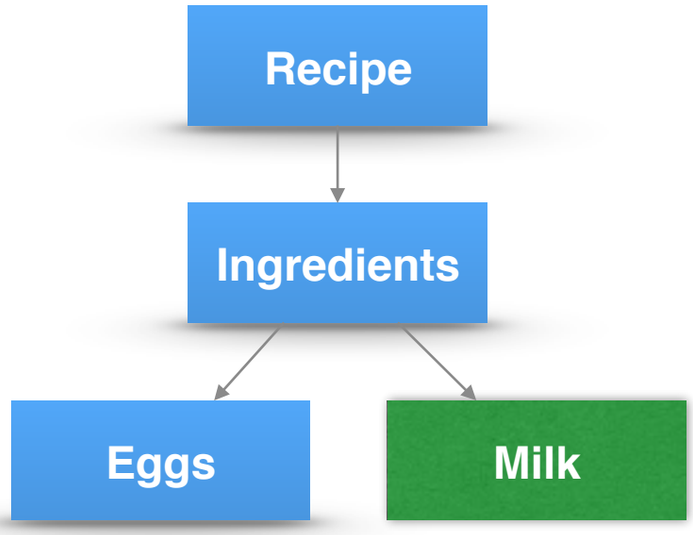


=

Real DOM

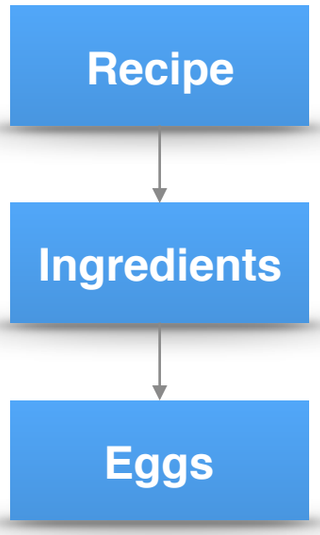


New Virtual DOM

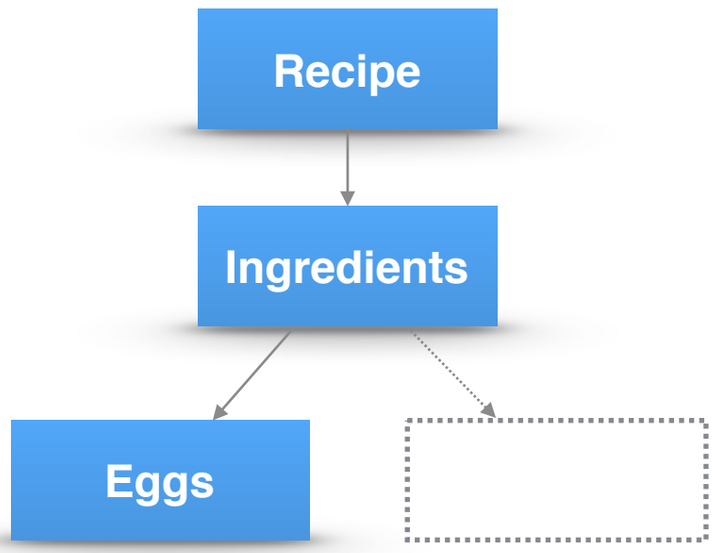


!=

Old Virtual DOM



Real DOM



JSX

Play with JSX online

https://babeljs.io/repl/

```
function App() {
  return React.createElement('div', null, [
    React.createElement('h1', null, 'I am a component!'),
    React.createElement('h2', null, 'I am a sub title')
  ]);
}
```

—

```
function App() {
  return (
    <div>
      <h1>I am a component!</h1>
      <h2>I am a sub title</h2>
    </div>
  );
}
```

PROPS

Passing Props

```
const Add = (props) => (
  <h2>Its: { props.a + props.b }</h2>
);

const App = () => (
  <Add a={ 2 } b={ 3 } />
);
```

Repeating with JavaScript (3/3)

```
const Recipes = ({ recipes }) => (
  <div>
    <h1>Recipes</h1>
    <ul>
      {
        recipes.map((recipe) => <Recipe recipe={ recipe } />)
      }
    </ul>
  </div>
);
```

CLASSES

Make <App /> into a class

```
class App extends React.Component {
  render() {
    return (
      <div>
        <Recipes recipes={ recipes }/>
      </div>
    );
  }
}
```

Component Lifecycle

```
class MyComponent extends React.Component {
  constructor() { }
  render() { }

  getInitialState() { }
  getDefaultProps() { }

  componentWillMount() { }
  componentDidMount() { }

  componentWillReceiveProps() { }
  shouldComponentUpdate() { }

  componentWillUpdate() { }
  componentDidUpdate() { }
  componentWillUnmount() { }
}
```

https://facebook.github.io/react/docs/component-specs.html

CHANGE DETECTION

Change detection

Props change

State change - setState()

Forced - forceUpdate()

Add a recipe after 1sec

```
constructor() {
  super();

  setTimeout(() => {
    console.log(`Adding new recipe`);
    recipes.push('Shakshuka');
  }, 1000);
}
```

PROP TYPES

Setting PropTypes

add-recipe.js

```
AddRecipe.propTypes = {
  addRecipe: React.PropTypes.func.isRequired
};
```

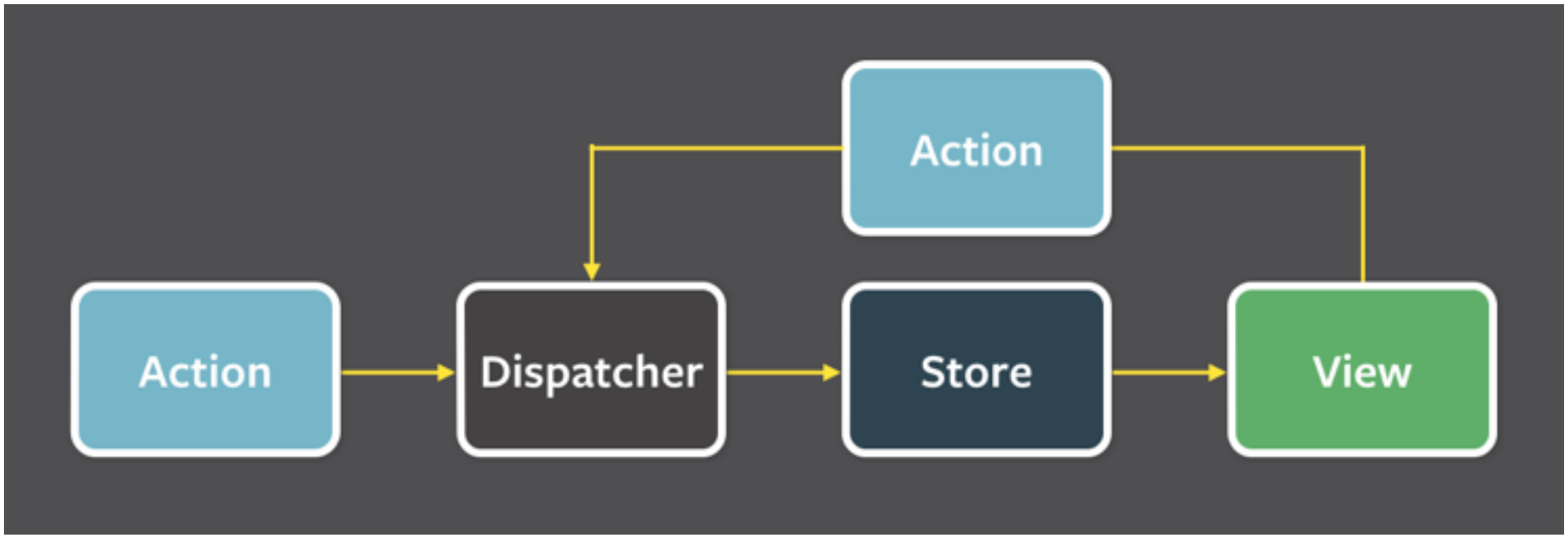
https://facebook.github.io/react/docs/reusable-components.html

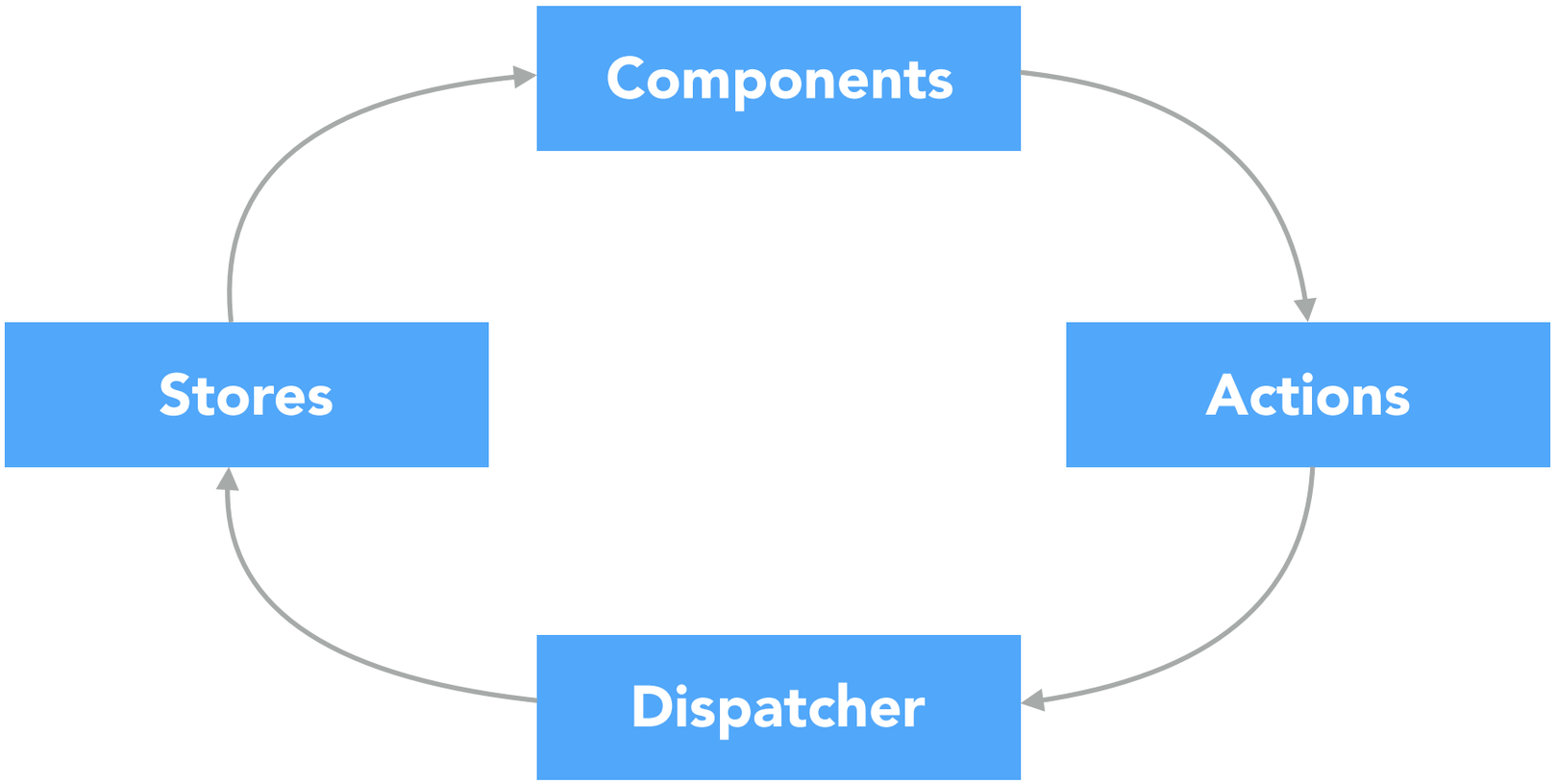
FLUX

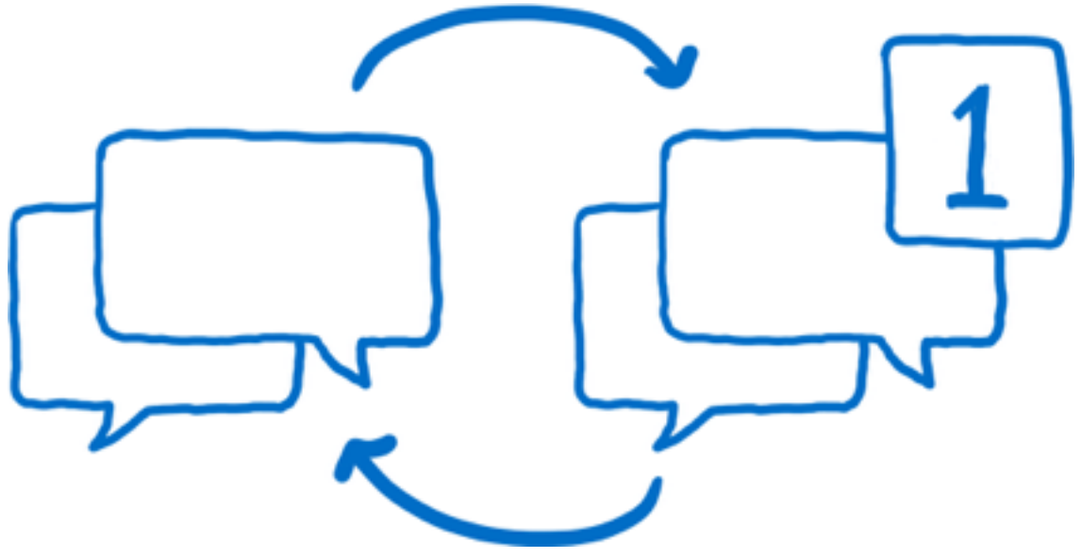
~~MVC~~

FLUX

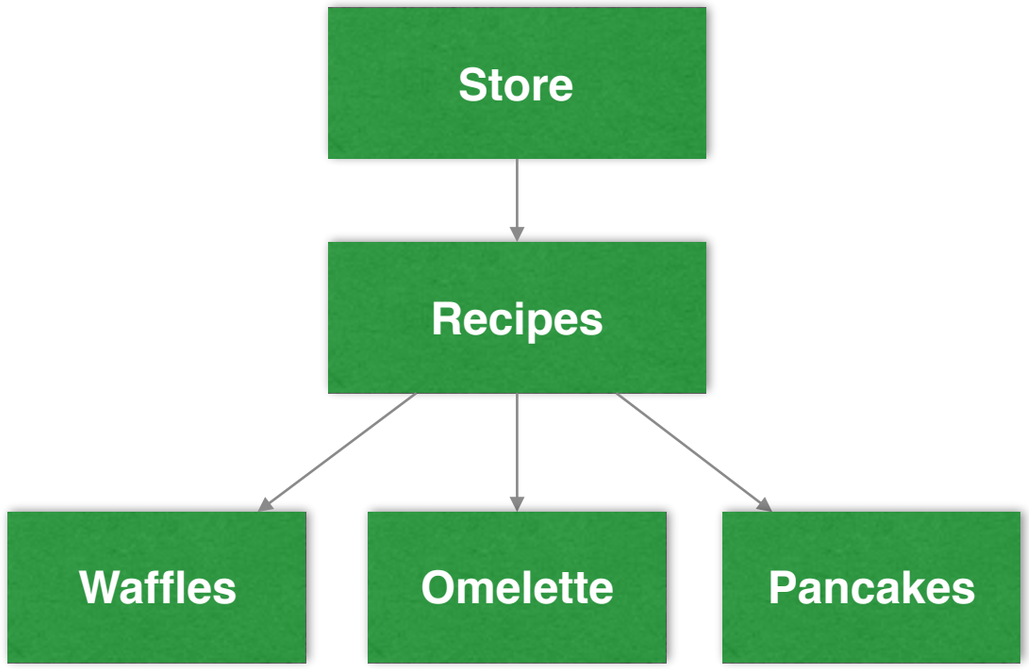
Flux

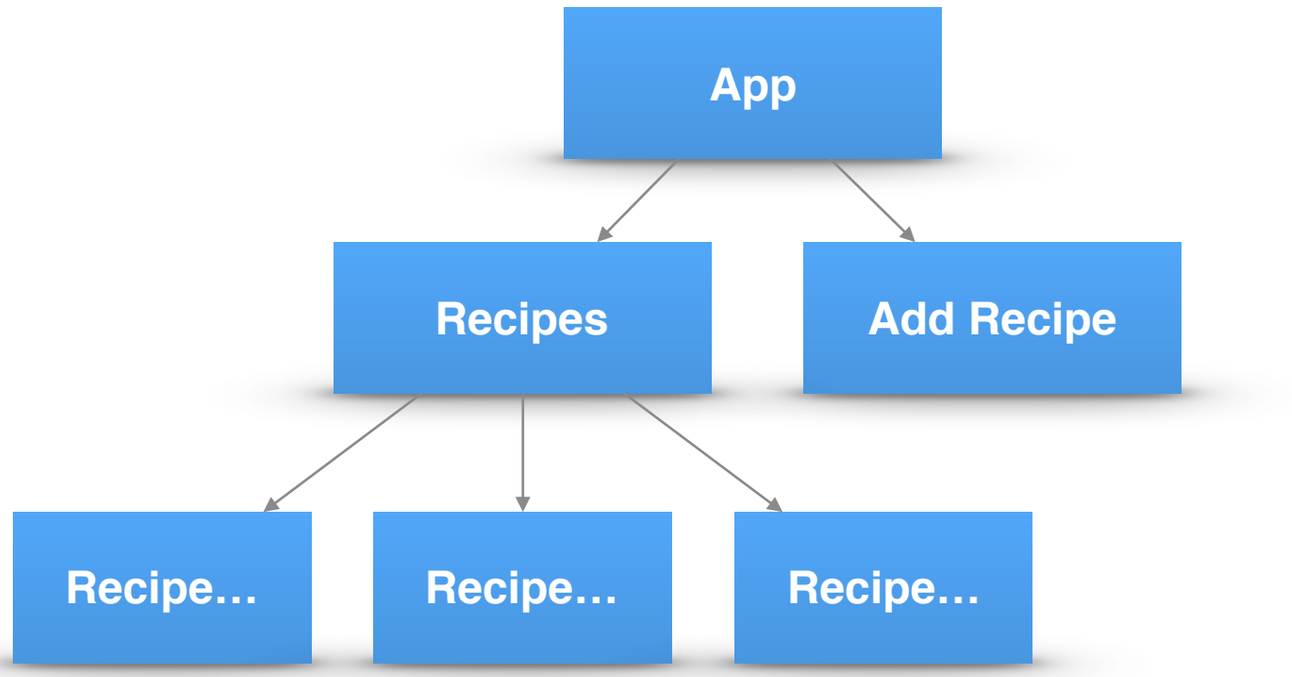



Components
Stores
Actions
Dispatcher


1

Game engine


Store
Recipes
Waffles
Omelette
Pancakes


App
Recipes
Add Recipe
Recipe…
Recipe…
Recipe…

REDUX

EVERYTHING IS AN ACTION

Click

Timeout

AJAX

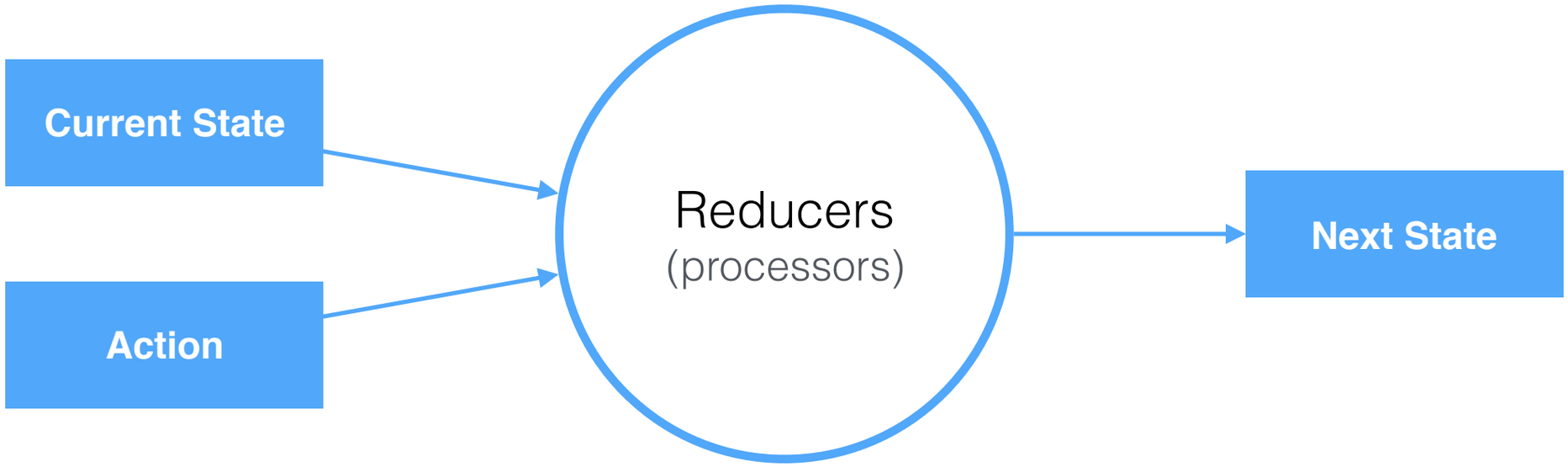
Websocket

Add Recipe

Toggle Favorite

Fetch Recipes

Start Network


Current State
Action
Reducers
(processors)
Next State

REDUCERS

Many reducers can be chained

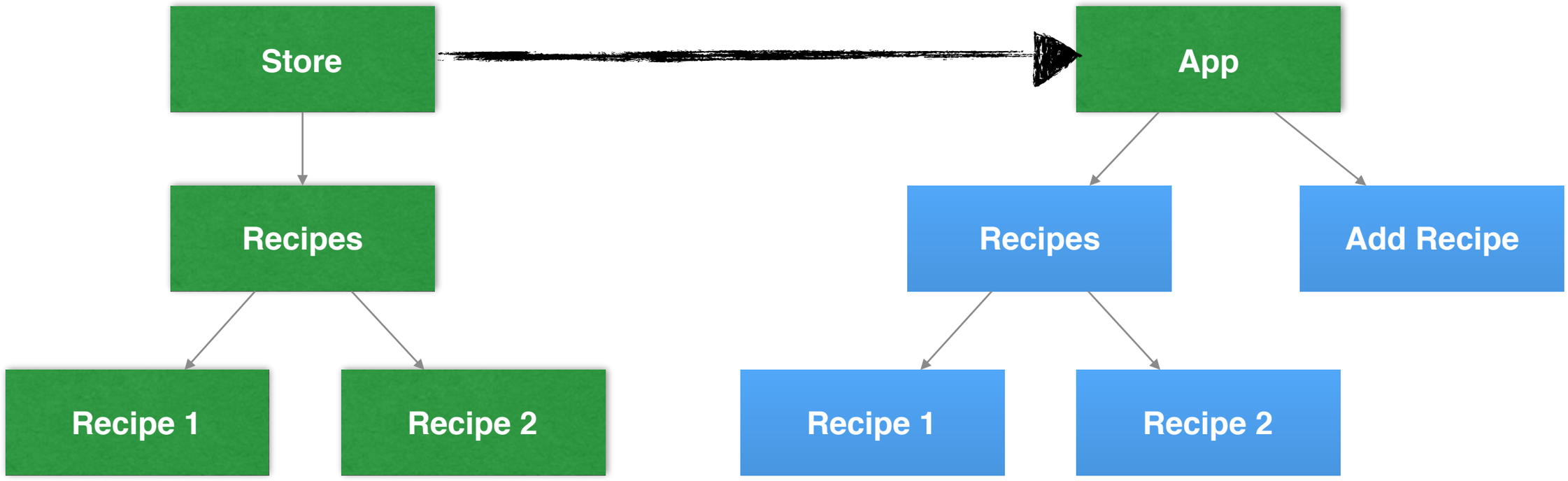
Must return a **new state** — never modify previous one

Object.assign or Immutable

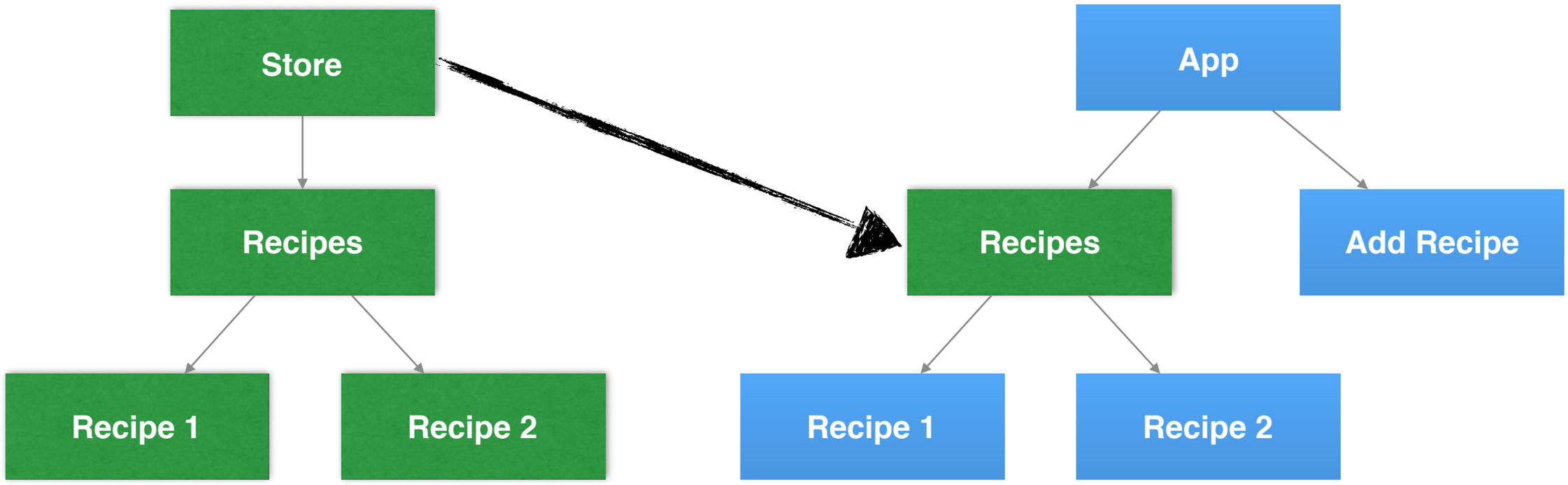
Only one store

CONNECT

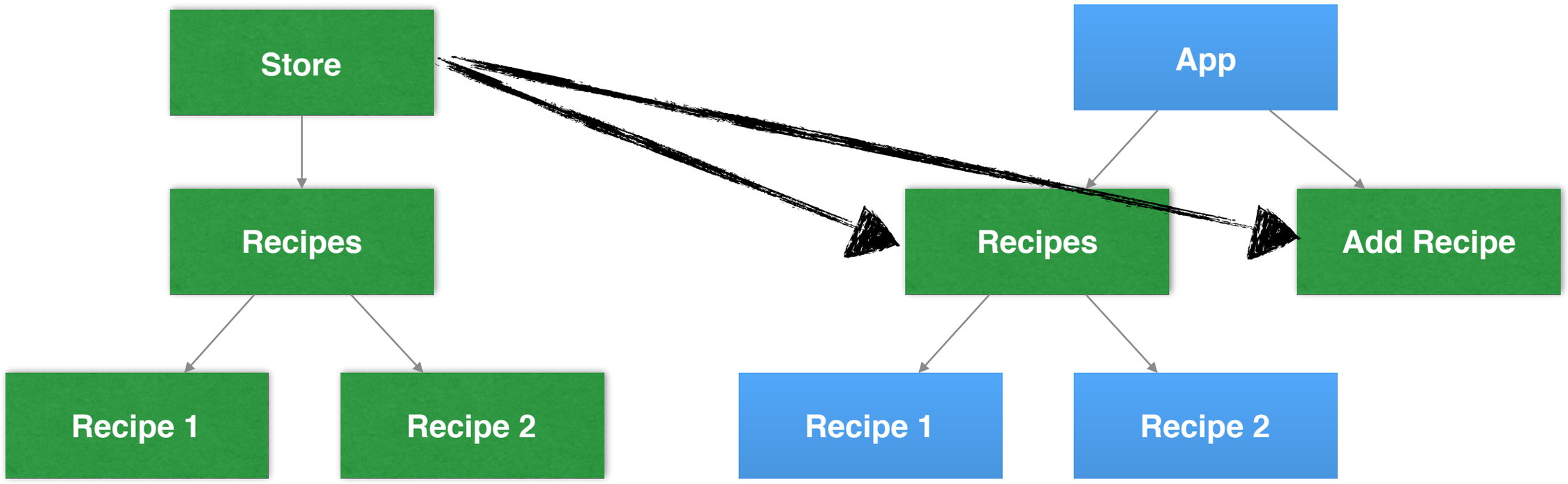
State to React



State to React



State to React



Connect

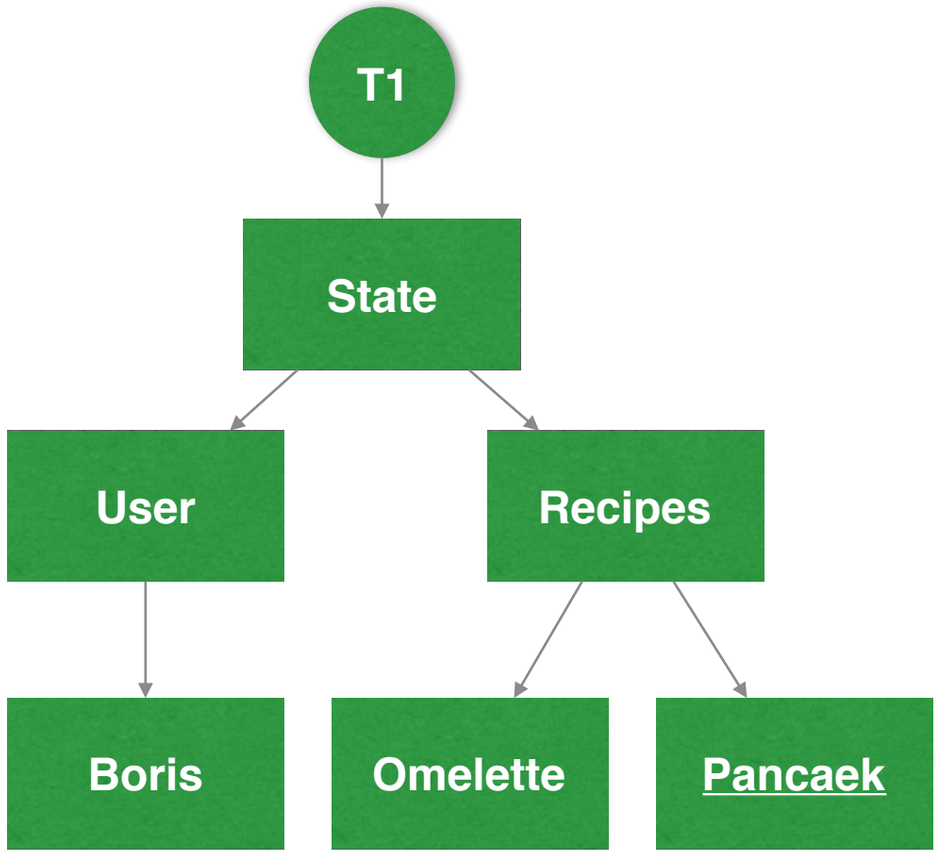
```
connect()(App);
```

Connect

```
function mapStateToProps(state) {
  return {};
}

function mapDispatchToProps(dispath) {
  return {};
}

connect(mapStateToProps, mapDispatchToProps)(App);
```

MUTATE STATE

OUR STATE



The reducer

```
action = {
  type: 'RENAME_RECIPE',
  recipeId: 871,
  name: 'Pancake'
};
```

```
const reducer = (state, action) => {
  switch (action.type) {
    case ‘RENAME_RECIPE':
      const { recipeId, newName } = action;
      state.recipes[recipeId].name = newName;
      return state;
  }
  return state;
};
```

The reducer

```
action = {
  type: 'RENAME_RECIPE',
  recipeId: 871,
  name: 'Pancake'
};
```

```
const reducer = (state, action) => {
  switch (action.type) {
    case 'RENAME_RECIPE':
      const { recipeId, newName } = action;
      state.recipes[recipeId].name = newName;
      return state;
  }
  return state;
};
```

The reducer

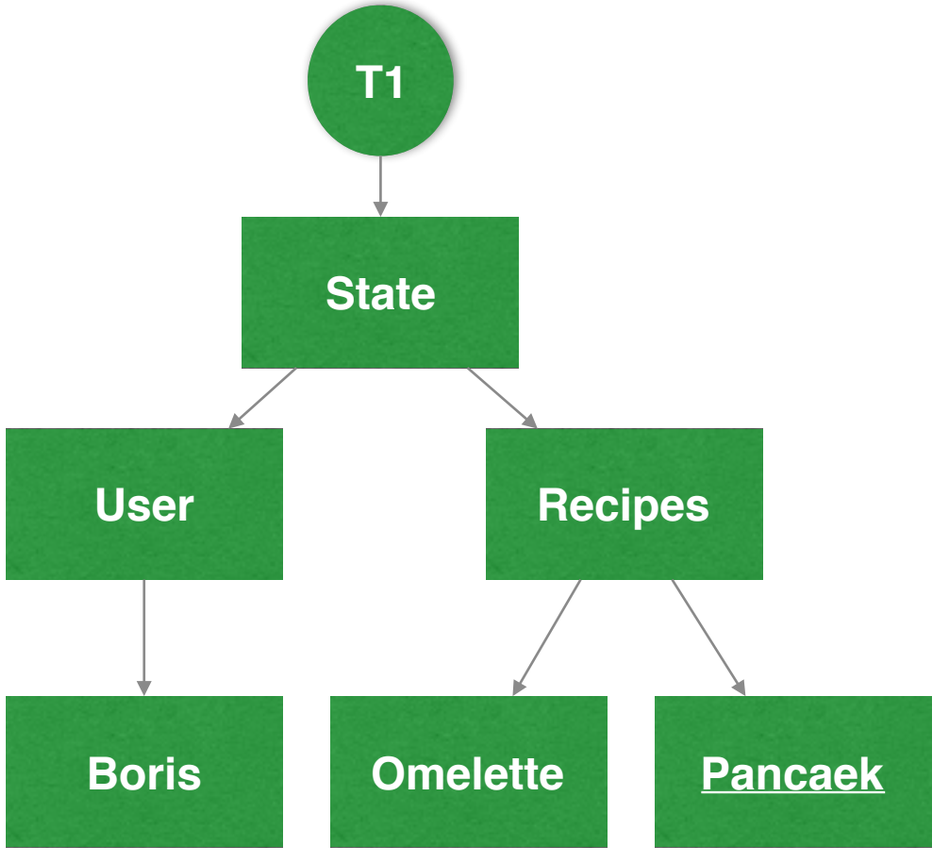
```
const reducer = (state, action) => {
  switch (action.type) {
    case 'RENAME_RECIPE':
      const recipe     = state.recipes[action.recipeId];

      const newRecipe  = Object.assign({}, recipe, {
        name: action.newName
      });

      const newRecipes = Object.assign({}, state.recipes, {
        [action.recipeId]: newRecipe
      });

      return Object.assign({}, state, {
        recipes: newRecipes
      });
  }
  return state;
};
```

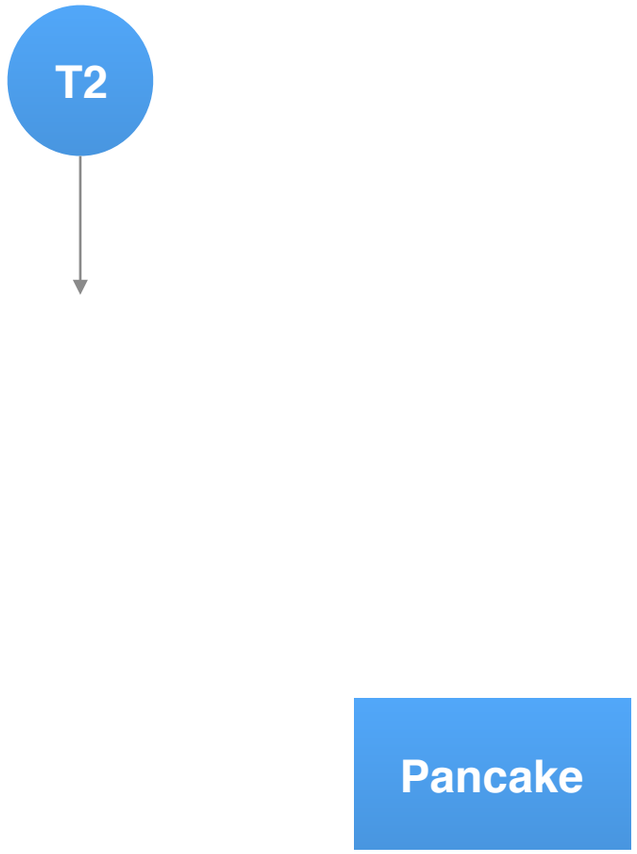
Object.assign()

```
const original = {
  name: 'Cat',
  age: 3
};

const updated = Object.assign({}, original, {
  name: 'Dog'
});
```

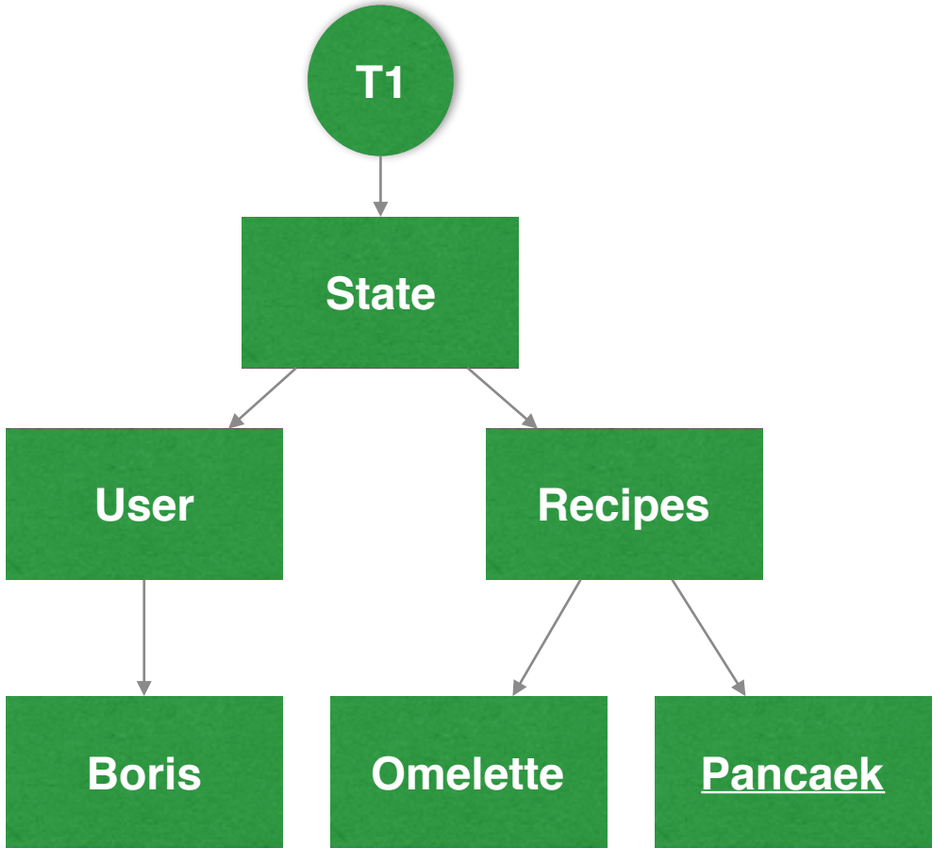
```
updated
> { name: 'Dog', age: 3 }

updated === original
> false
```

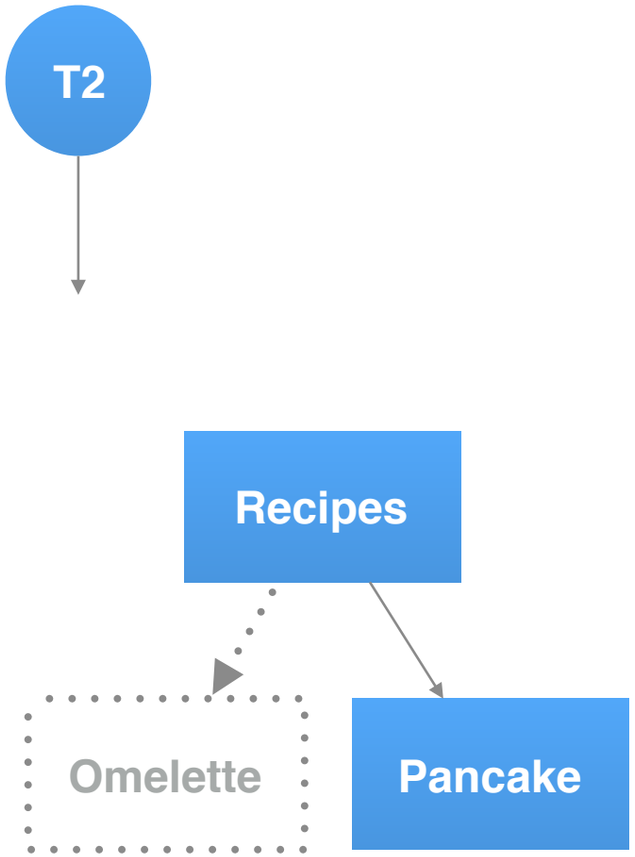
REFERENCE TREES



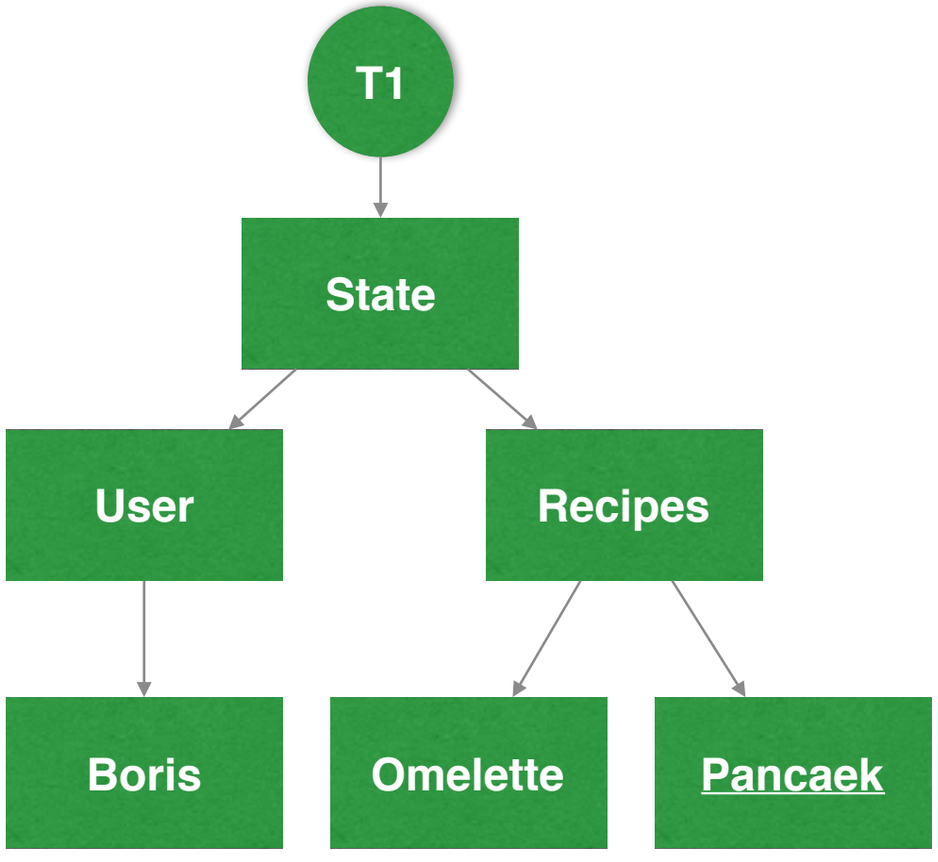


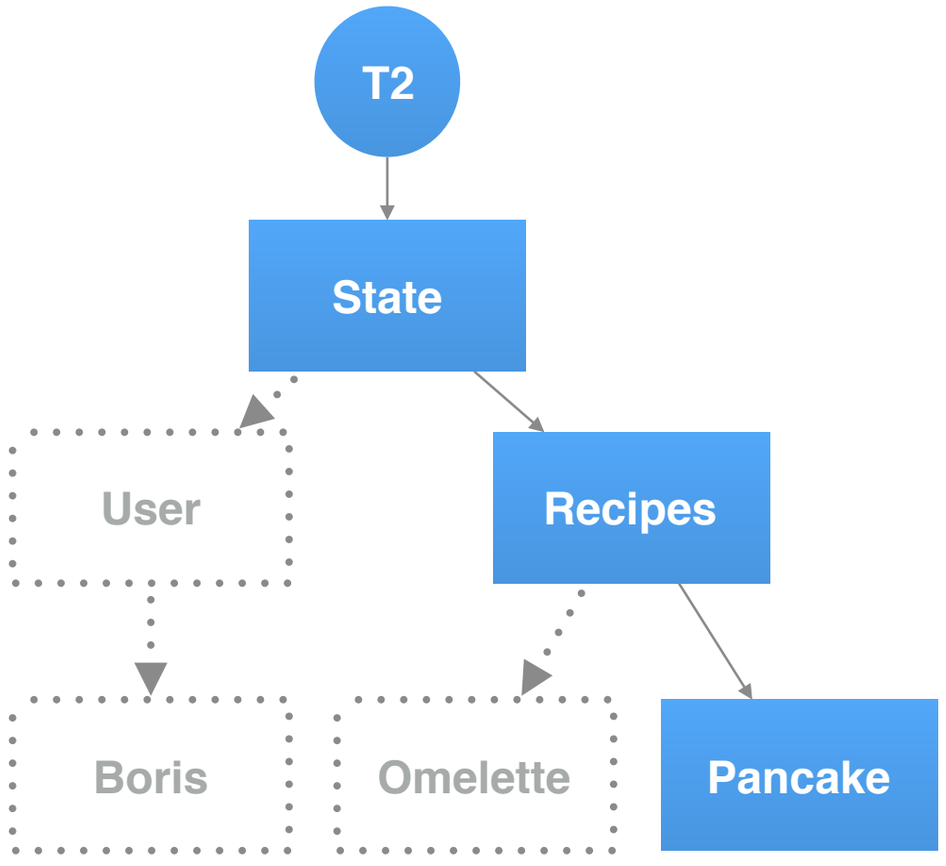
REFERENCE TREES



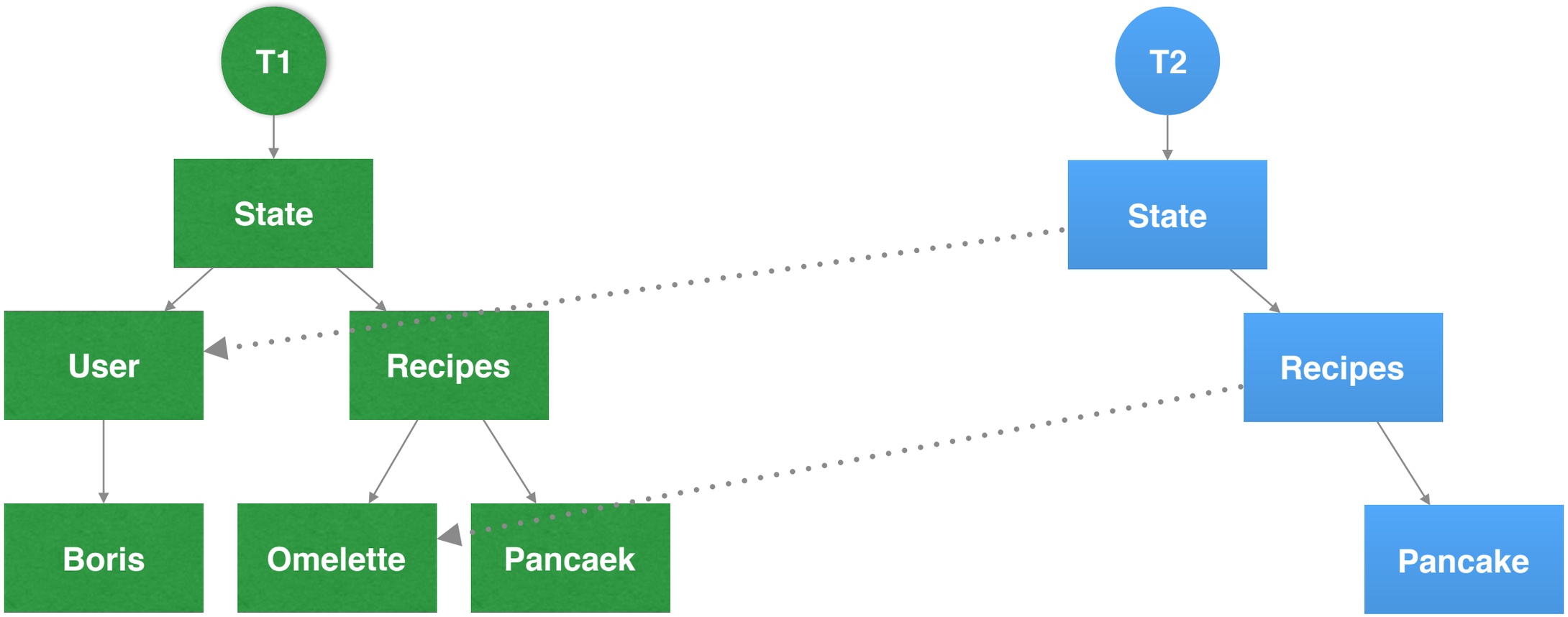


REFERENCE TREES

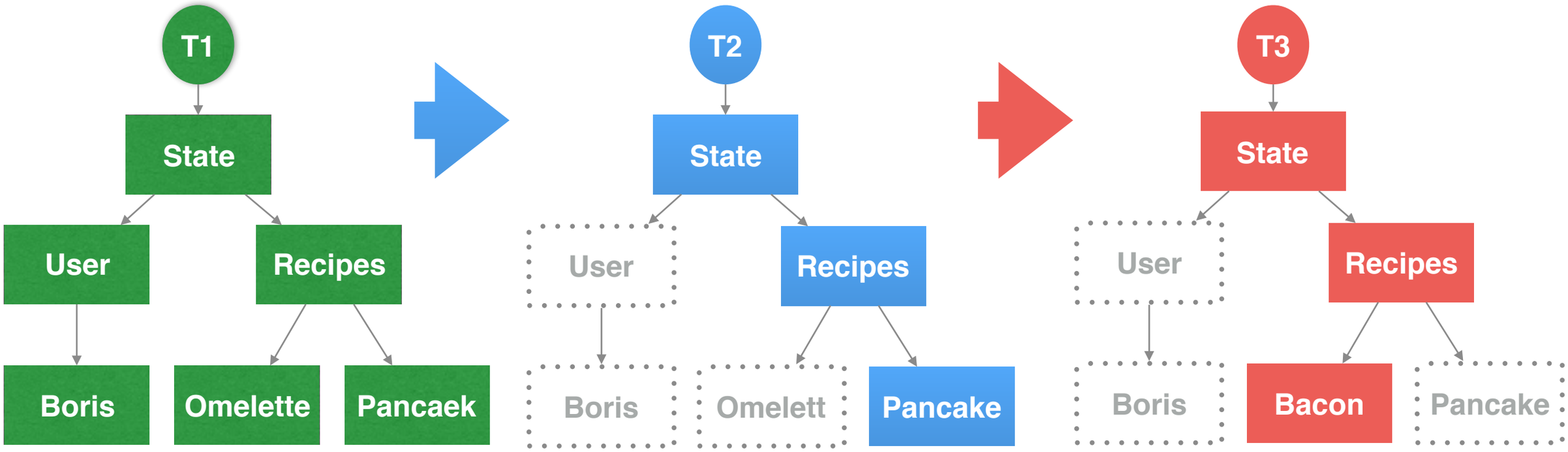




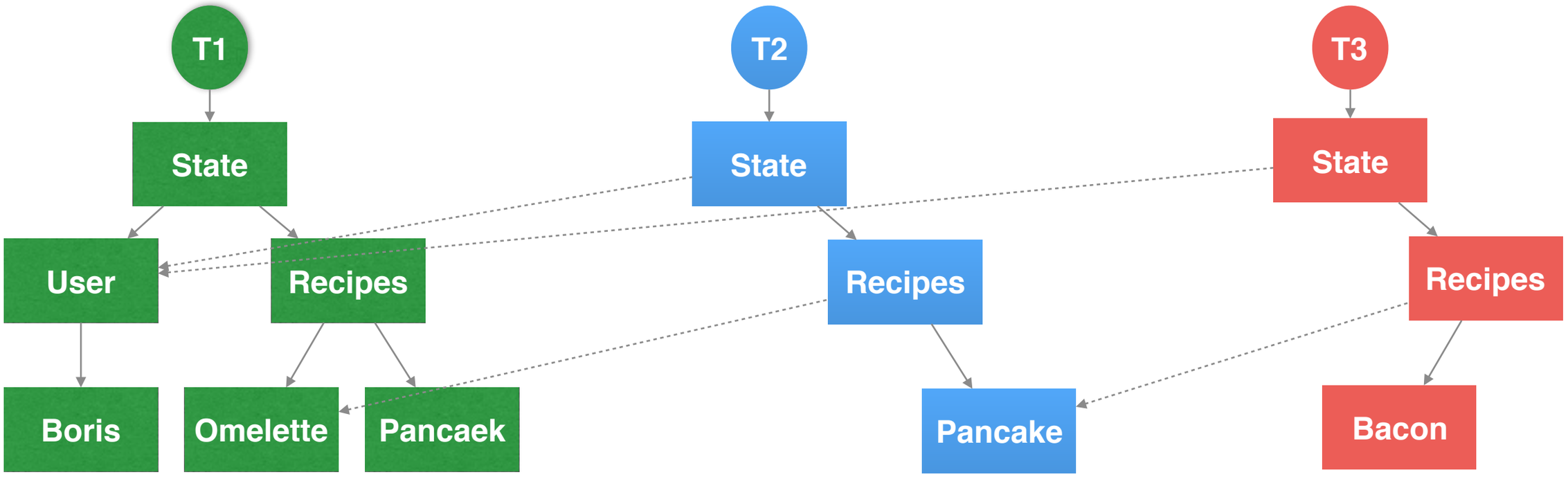
IN MEMORY


T1
State
User
Recipes
Boris
Omelette
Pancaek
T2
State
Recipes
Pancake

ACTION #2


T1
State
User
Recipes
Boris
Omelette
Pancaek
T2
State
User
Recipes
Boris
Omelett
Pancake
T3
State
User
Recipes
Boris
Bacon
Pancake

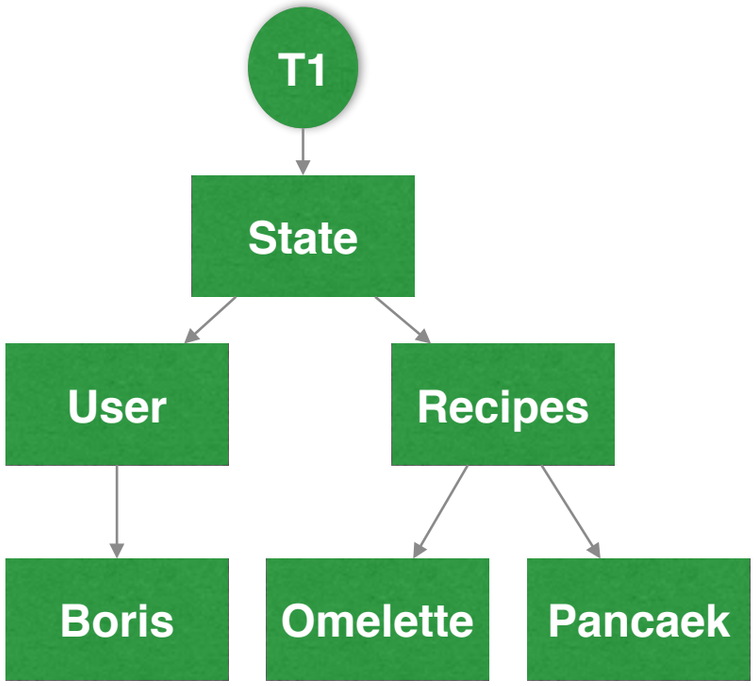
IN MEMORY

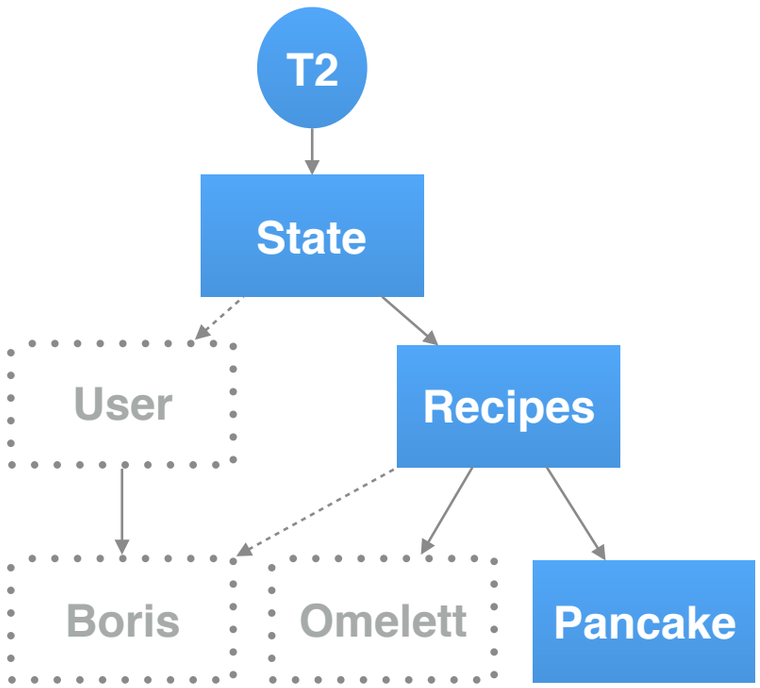

T1
State
User
Recipes
Boris
Omelette
Pancaek
T2
State
Recipes
Pancake
T3
State
Recipes
Bacon

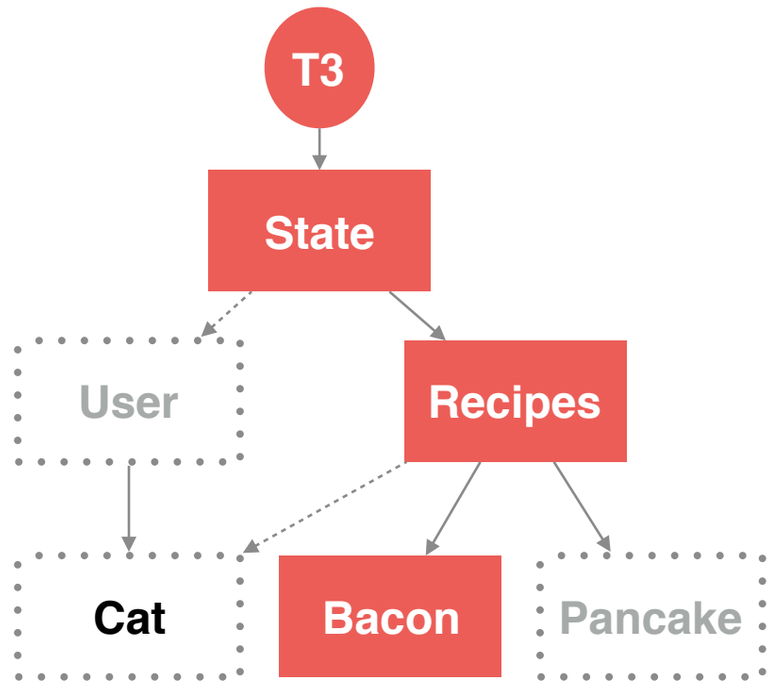
Why not direct?

```
state.recipes[recipeId].name = newName;
```

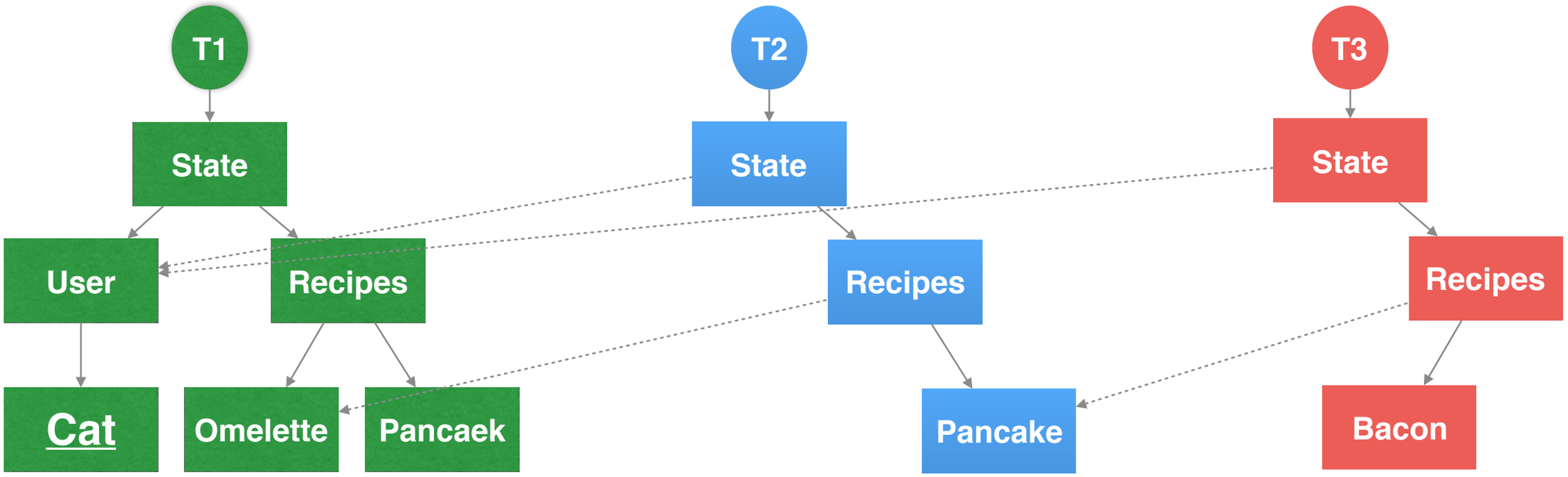
OH, NO!


T1
State
User
Recipes
Boris
Omelette
Pancaek

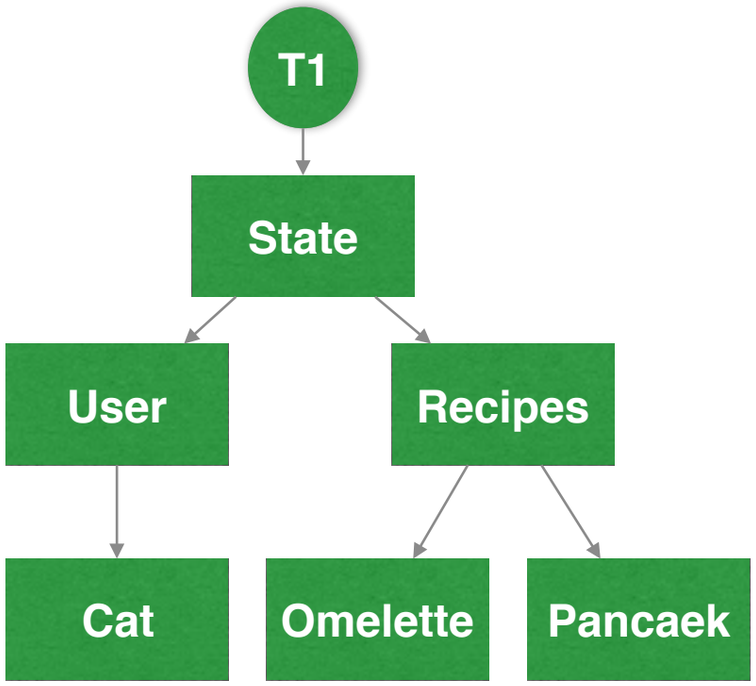

T2
State
User
Recipes
Boris
Omelett
Pancake

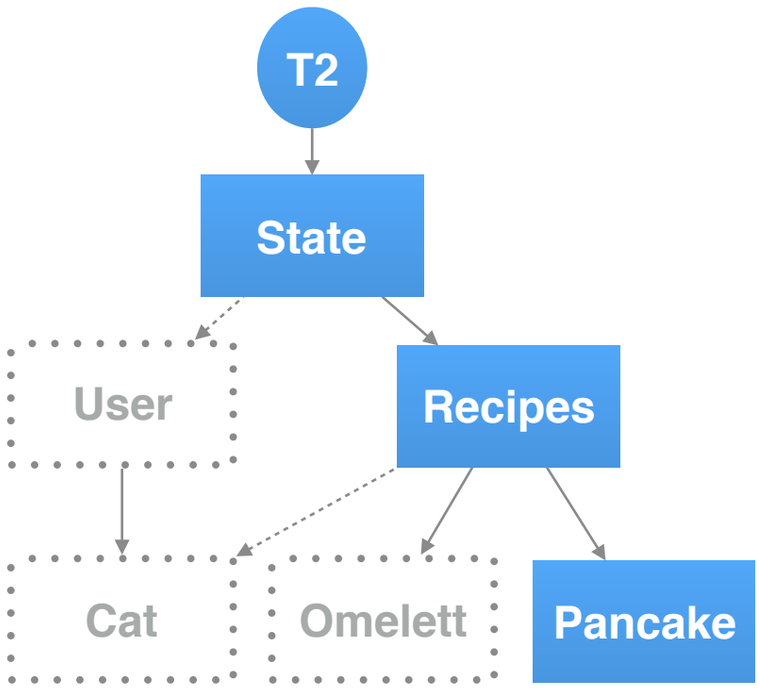

T3
State
User
Recipes
Cat
Bacon
Pancake

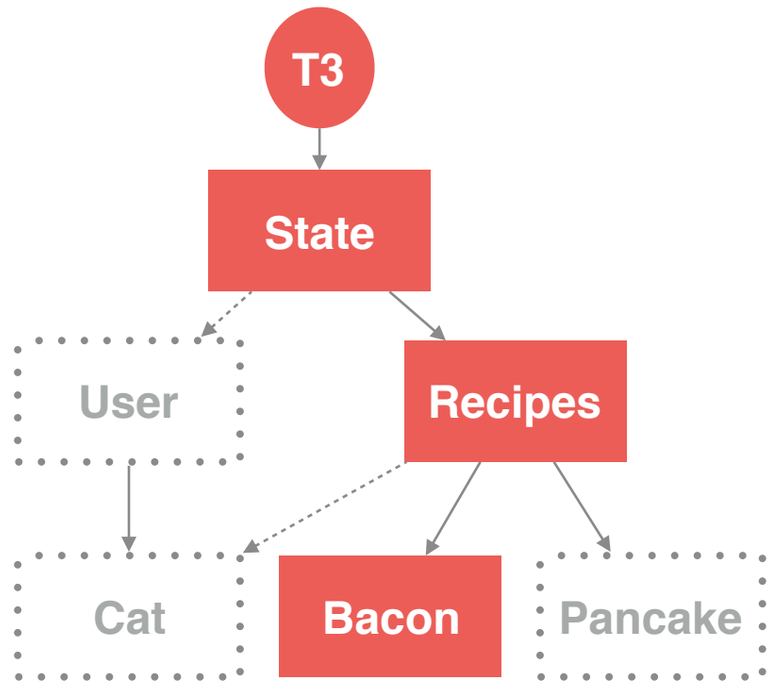
OUT SIDE CHANGE



OH, NO!


T1
State
User
Recipes
Cat
Omelette
Pancaek


T2
State
User
Recipes
Cat
Omelett
Pancake


T3
State
User
Recipes
Cat
Bacon
Pancake

Console work

```
s1 = store.getState()
```

Console work

```
s1 = store.getState()

- Do an action

s2 = store.getState()
```

Console work

```
s1 = store.getState()

- Do an action

s2 = store.getState()

- Do an action

s3 = store.getState()
```

Console work

```
s1 = store.getState()

- Do an action

s2 = store.getState()

- Do an action

s3 = store.getState()


store.dispatch({ type: ‘SET_STATE’, payload: s1 });
```

UNDO / REDO

BUT...

1. Actions like LOGIN
2. Actions from Middleware / Redux-Thunk
3. Layout / UI

Directory structure

reducers

store

components

ACTION CREATORS

Actions

```
export const addRecipe = (name) => ({
  type: 'ADD_RECIPE',
  name: name
});
```

ROUTING

React Router

```
import { Router, Route, browserHistory } from 'react-router'

render(
  <Provider store={ store }>
    <Router history={ browserHistory }>
      <Route path="" components={ App }>
        <Route path="/add" component={ AddRecipe } />
        <Route path="/" component={ Recipes } />
      </Route>
      <Route path="*" component={ NotFound } />
    </Router>
  </Provider>,
  document.getElementById('app')
);
```

TESTS

Testing

Karma - Test running - https://karma-runner.github.io/0.13/index.html

Enzyme - Test utils for React - https://github.com/airbnb/enzyme

Jasmine - Matcher and mocks - http://ricostacruz.com/cheatsheets/jasmine.html

Redux tests - http://redux.js.org/docs/recipes/WritingTests.html

SUMMARY

Useful stuff

Meetup: http://www.meetup.com/ReactJS-Israel/

Redux tutorial: https://egghead.io/series/getting-started-with-redux

The code we built: https://bitbucket.org/500tech/react-redux-course

ReactNext 2016

The largest ReactJS conference of the Startup Nation

September 15th

Dan Panorama Hotel, **Tel Aviv**

500Tech

React & Redux

Read our blog:

http://blog.500tech.com