



Reactの最新動向とベストプラクティス

HTML5 Conference 2016 (2016/09)

@koba04

@koba04



Toru Kobayashi
koba04

📍 Tokyo
✉ koba0004@gmail.com
🌐 <http://koba04.com/>
🕒 Joined on Apr 23, 2010

104 Followers
306 Starred
64 Following

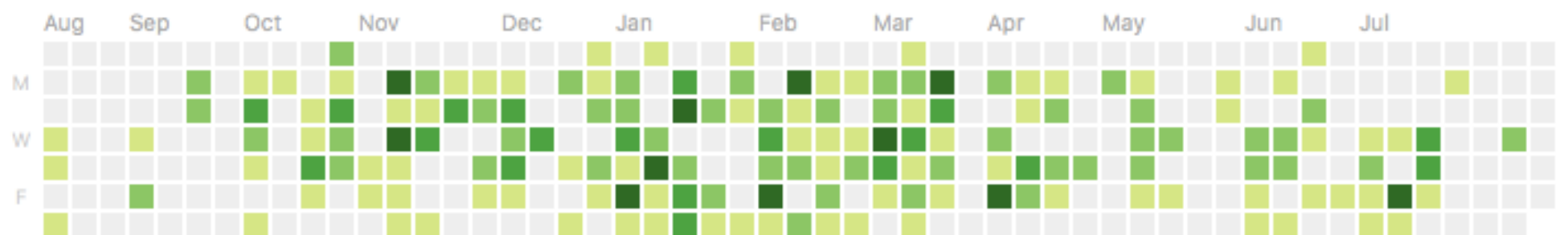
Overview Repositories Public activity

Follow

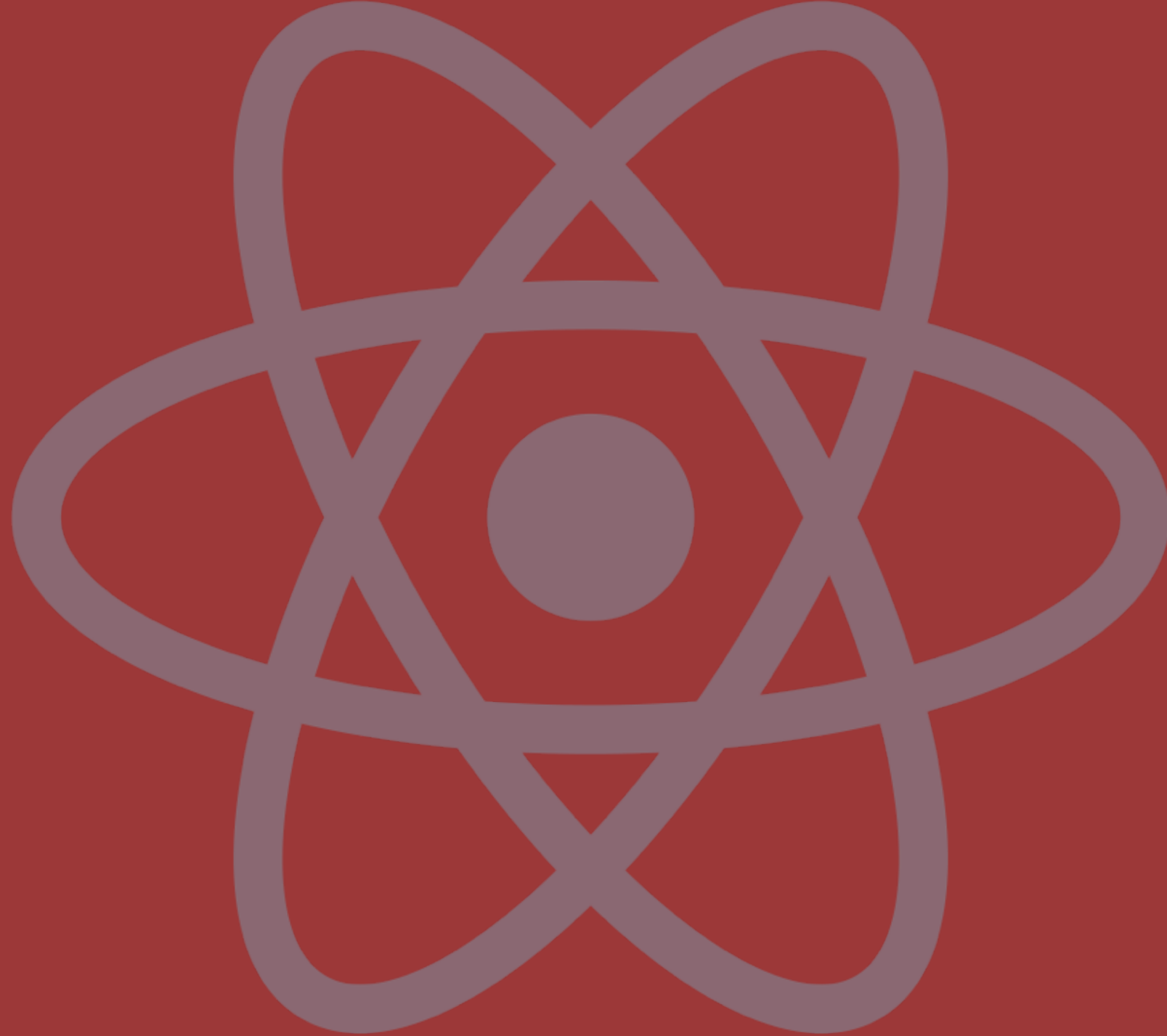
Pinned repositories

	facebook/react A declarative, efficient, and flexible JavaScript library for building user interfaces.	47,510 ★
	airbnb/enzyme JavaScript Testing utilities for React	6,192 ★
	segmentio/nightmare A high-level browser automation library.	8,377 ★
	mjackson/history Manage browser history with JavaScript	1,554 ★
	facebook/flux Application Architecture for Building User Interfaces	12,196 ★

565 contributions in the last year



React事例



Facebook

The image shows a Facebook profile page for Toru Kobayashi. The profile picture shows a man on a beach. The cover photo shows a child kissing a baby. The page has tabs for Timeline, About, Friends (157), Photos, and More. A React DevTools overlay is visible, showing the ReactComposer component with a text input "What's on your mind?". The React DevTools interface includes a toolbar with options like Trace React Updates, Highlight Search, and Use Regular Expressions. The component tree on the left shows the structure of the ReactComposer component, and the props panel on the right lists its properties.

ReactComposer 510px x 149px

Status Photo / Video Life Event

What's on your mind?

Friends Post

<ReactComposer>

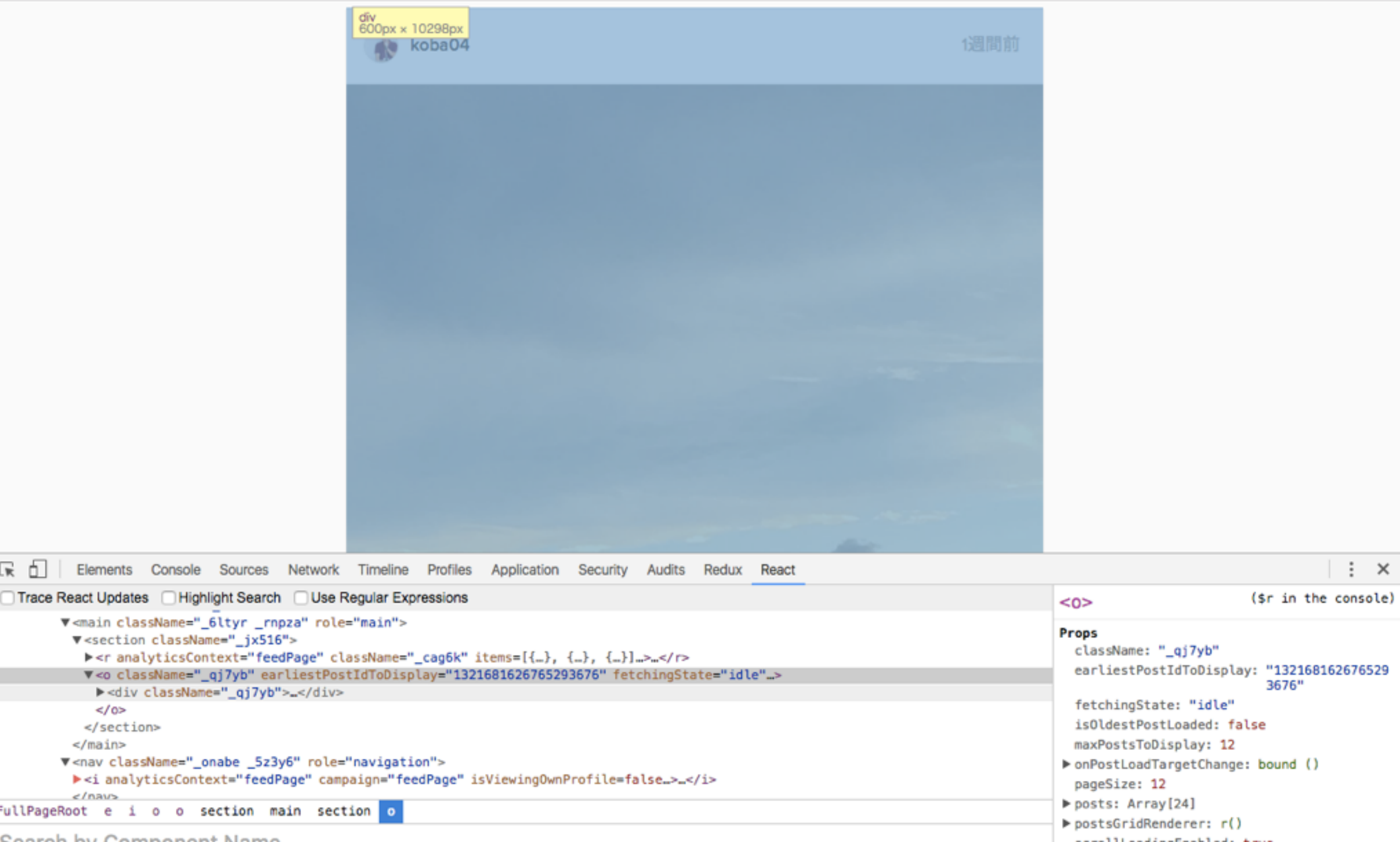
Ref "root"

Props

- activateOnInit: false
- children: Array[2]
- className: "_s7h"
- hideBorders: true

```
<ReactComposer ref="root" className="_s7h" hideBorders=true loggingConfig={targetType: "feed", ref: "timeline", composerVersion: "www_re...>
  <k className="_s7h _36bx _143o" data-testid="react-composer-root" id="rc.u_jsonp_18_l">
    <div className="_s7h _36bx _143o _4-u2 _4-u8" data-testid="react-composer-root" id="rc.u_jsonp_18_l">
      <ReactComposerTimelineHeader background="light-wash" noHorizontalMargin=true showDelimiter=false...></ReactComposerTimelineHeader>
      <ReactComposerBodyContainer expanded=true hasMinHeight=true>
        <k className="_4cw">
          <k className="_4zoz _4cw" background="transparent">
            <div className="_4zoz _4cw _4-u3" background="transparent">
```

Instagram



The screenshot shows the Instagram web application with a blue header bar. The header contains a profile picture, the username "koba04", and the text "1週間前". The main content area is a large blue rectangle. The React DevTools component inspector is open at the bottom, showing the component tree and the props for the selected component.

Component Tree:

```
<main className="_6ltyr _rnpza" role="main">  
  <section className="_jx516">  
    <r analyticsContext="feedPage" className="_cag6k" items=[{...}, {...}, {...}]...>  
      <o className="_qj7yb" earliestPostIdToDisplay="1321681626765293676" fetchingState="idle"...>  
        <div className="_qj7yb">...</div>  
      </o>  
    </section>  
  </main>  
  <nav className="_onabe _5z3y6" role="navigation">  
    <i analyticsContext="feedPage" campaign="feedPage" isViewingOwnProfile=false...>...</i>  
  </nav>  
</main>
```

Props for <o>:

- className: "_qj7yb"
- earliestPostIdToDisplay: "1321681626765293676"
- fetchingState: "idle"
- isOldestPostLoaded: false
- maxPostsToDisplay: 12
- onPostLoadTargetChange: bound ()
- pageSize: 12
- posts: Array[24]
- postsGridRenderer: r()
- scrollLoadingEnabled: true

Netflix

AkiraLayout 1218px x 7287.28125px

Netflix

メニュー

キッズ

検索

通知

マイページ

ストレンジャー・シングス 未知の世界

Netflix オリジナル作品

ピックとドラゴン 新たな世界へ!

Netflix オリジナル作品

デアデビル

Netflix オリジナル作品

TERRACE HOUSE

Netflix オリジナル作品

コング 猿の王者

Netflix オリジナル作品

Elements Console Sources Network Timeline Profiles Application Security Audits Redux React

☐ Trace React Updates ☐ Highlight Search ☐ Use Regular Expressions

<div ref="main" lang="ja" dir="ltr" onTouchStart=bound _handleTouchStart()>
<e model={title: "Netflix", pathEvaluator: {...}, node: {...}} renderSource="client" uiView="homeScreen">
<e>
<AkiraLayout ref="__routeHandler__" model={title: "Netflix", pathEvaluator: {...}, node: {...}} renderSource="client" uiView="homeScreen">
<div className="bd dark-background" lang="ja">
<TimeoutTransitionGroup transitionName="pageTransition" enterTimeout=0 transitionEnter=false>...</TimeoutTransitionGroup>
<Header ref="header" model={title: "Netflix", pathEvaluator: {...}, node: {...}} pathEvaluator={_streaming: false, __next: Array}>
<div className="mdx-mount-point nfp hidden"/>
<FreezeWrapper shouldFreeze=false>
<TimeoutTransitionGroup key="mainView" className="mainView" role="main" component="div">
<ReactTransitionGroup className="mainView" role="main" component="div">

Router LayoutContext DetectFonts(BaseRouteHandler) div BaseRouteHandler div e e AkiraLayout

Search by Component Name

<AkiraLayout> (\$r in the console)

Ref "__routeHandler__"

Props

isInitialRender: true
isKidsPage: false
isOverlayPage: false
model: {...}
params: {...}
profileGateAppState: {...}
promotionalVideoHasPlayed: false
query: {...}
renderSource: "client"

Twitter (mobile)



Elements

Console

Sources

Network

Timeline

Profiles

Application

Security

Audits

Redux

React

Trace React Updates

☐ Highlight Search

☐ Use Regular Expressions

</t>

▼<r ref=fn() history={listenBefore: _(), listen: x(), transitionTo: a(), ...} location={pathnam

▼<t namespace={page: "profile", section: "tweets"}>

▼<Connect(t) history={listenBefore: _(), listen: x(), transitionTo: a(), ...} location={path

▼<t history={listenBefore: _(), listen: x(), transitionTo: a(), ...} location={pathname: "

▶<t accessibilityLabel="@koba04さんのツイートを読み込み中" displayMode="loaded" render=fn()>...

</t>

</Connect(t)>

</t>

</r>

Router

RouterContext

t

SideEffect(DocumentMeta)

DocumentMeta

Connect(t)

t

t

div

t

t

div

t

main

Connect(t)

t

t

SideEffect(DocumentMeta)

DocumentMeta

Connect(t)

t

t

t

t

div

t

div

t

div

r

t

Connect(t)

Search by Component Name

<Connect(t)>

(\$r in the console)

Props

children: null

displayBlocked: false

▶ history: {...}

initialBufferHeight: 0

▶ location: {...}

▶ params: {...}

▶ route: {...}

▶ routeParams: {...}

▶ routes: Array[5]

▶ scribeNamespace: {...}

▶ user: {...}

Abema.tv

t 1086px x 2157px

現在CM放送中！

ホーム番組表通知予約リストオンデマンドランキング今日のみどころその他

8月30日（火）のみどころ
【AbemaTV初登場】阿部寛 宮迫博之
「アットホーム・ダッド」
ドラマチャンネル

22:14

現在27チャンネル放送中！

Abema news/

松井恵理子 松崎麗 Abema SPECIAL

Abema SPECIAL2

Elements Console Sources Network Timeline Profiles Application Security Audits Redux React

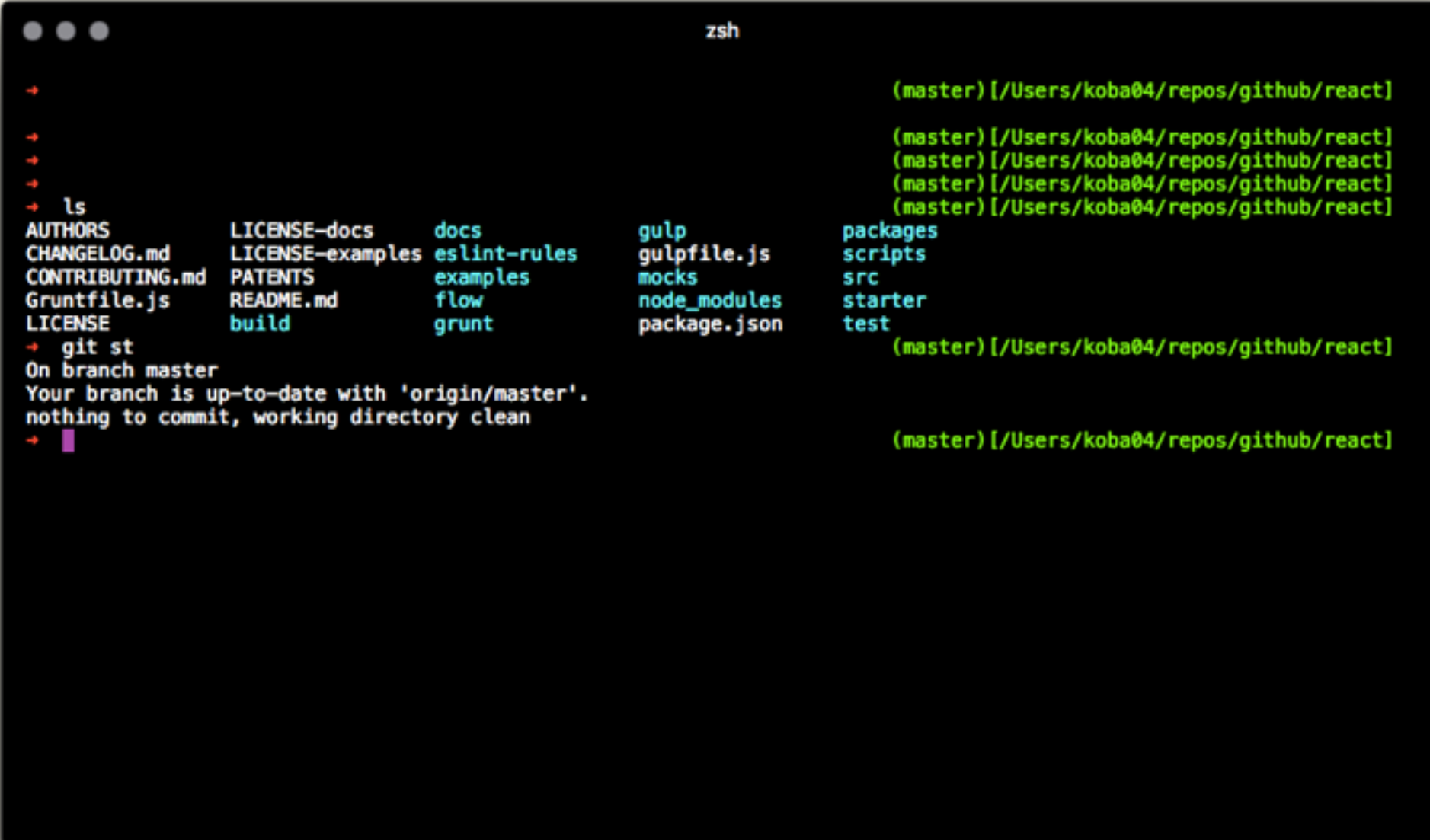
☐ Trace React Updates ☐ Highlight Search ☐ Use Regular Expressions

<Router history={listenBefore: r(), listen: i(), transitionTo: u(), ...} render=render()>
<RouterContext history={listenBefore: r(), listen: b(), transitionTo: u(), ...} router={listenBefore: r(), listen: i()
 <t history={listenBefore: r(), listen: b(), transitionTo: u(), ...} location={pathname: "/", search: "", hash: "", ...
 <div ref="wrapperContainer">
 <div ref="headerContainer" className="AppContainer__header-container__39c wd AppContaine...>
 <r className="AppContainer__header__3nc9H" onCloseHighlightsBalloonButton=bound value() highlights={tableEndA
 </div>
 <t type="userEnvironment" data=null close=bound value()...></t>
 <t key="main" history={listenBefore: r(), listen: b(), transitionTo: u(), ...} location={pathname: "/", search: ""
 <r>...</r>
 </div>
 </t>
 </RouterContext>
</Router>

<t> (\$r in the console)
Props
 ▶ history: {...}
 ▶ location: {...}
 ▶ main: {...}
 ▶ params: {...}
 ▶ route: {...}
 ▶ routeParams: {...}
 ▶ routes: Array[2]
 ▶ splash: {...}
State
 hasShowSplash: true

Hyperterm

- <https://hyperterm.org/>
- <https://github.com/zeit/hyperterm>

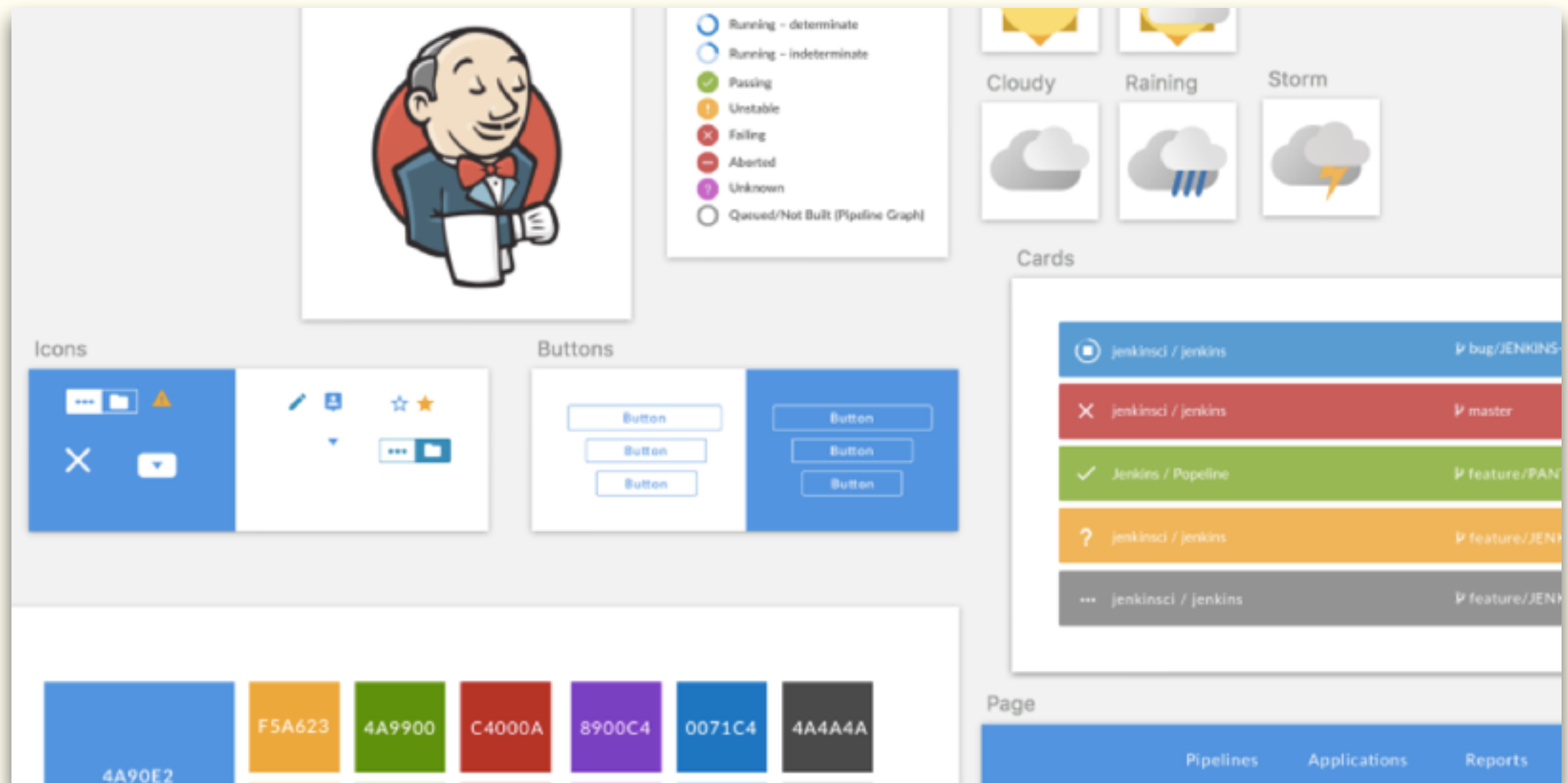


```
zsh

→ (master) [/Users/koba04/repos/github/react]
→ (master) [/Users/koba04/repos/github/react]
→ (master) [/Users/koba04/repos/github/react]
→ (master) [/Users/koba04/repos/github/react]
→ (master) [/Users/koba04/repos/github/react]
→ ls
AUTHORS          LICENSE-docs     docs            gulp            packages
CHANGELOG.md     LICENSE-examples eslint-rules     gulpfile.js    scripts
CONTRIBUTING.md PATENTS         examples       mocks          src
Gruntfile.js     README.md       flow           node_modules  starter
LICENSE          build           grunt          package.json   test
→ git st
On branch master
Your branch is up-to-date with 'origin/master'.
nothing to commit, working directory clean
→ (master) [/Users/koba04/repos/github/react]
```

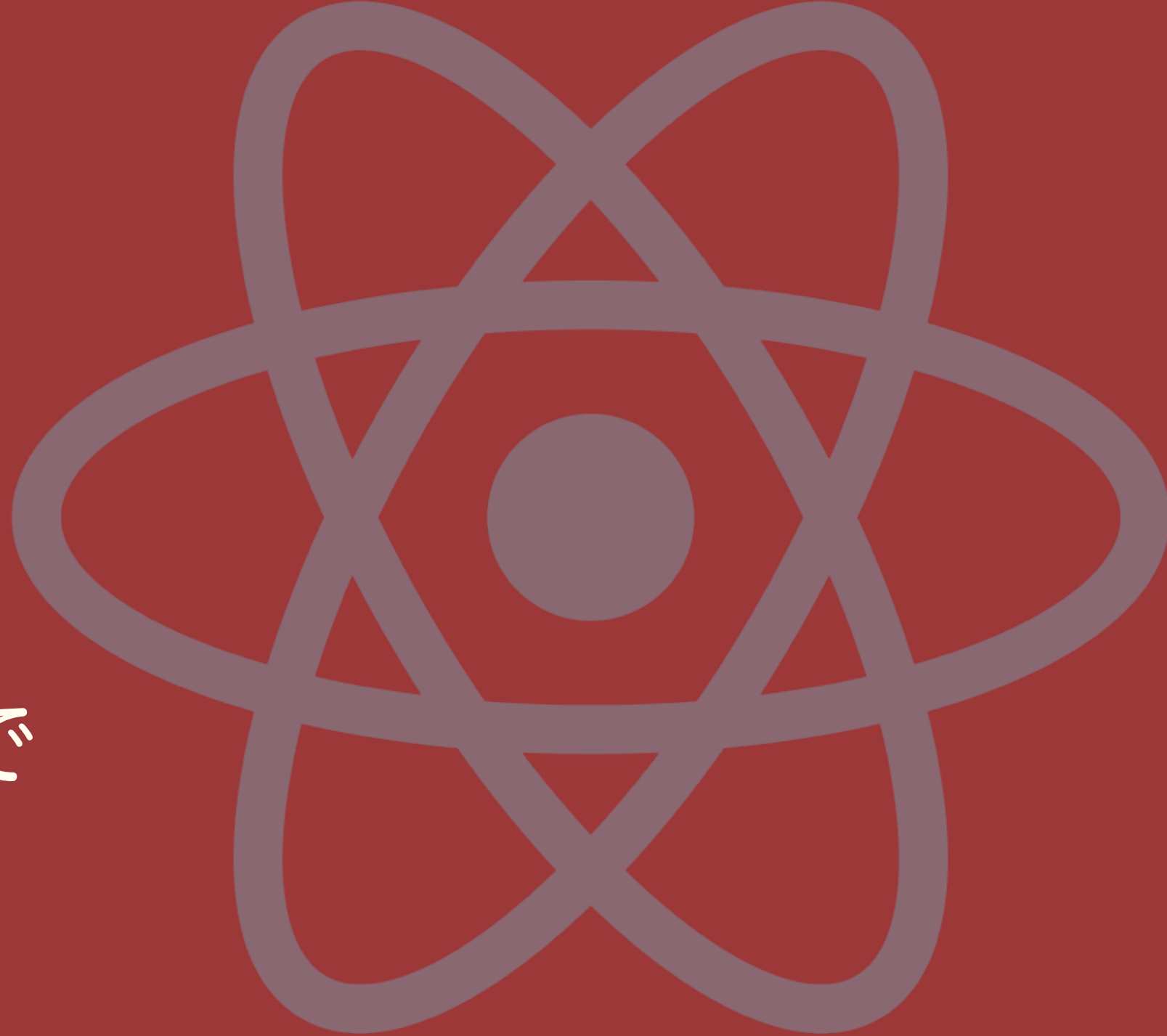
Jenkins Design Language

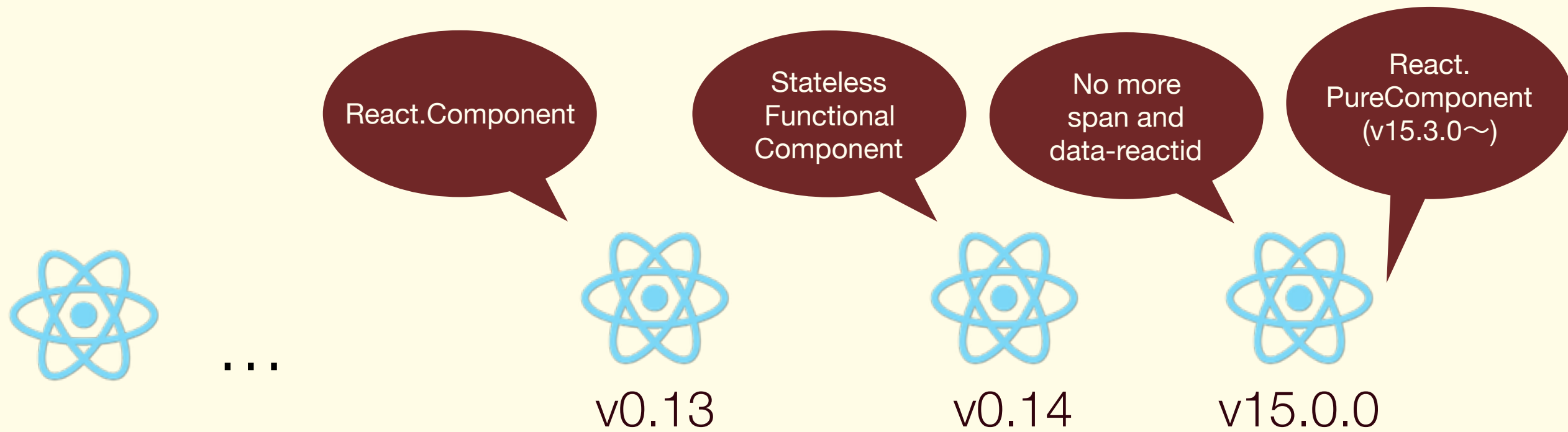
- <https://github.com/jenkinsci/jenkins-design-language>



Reactは”今”使われているライブラリー 🌐

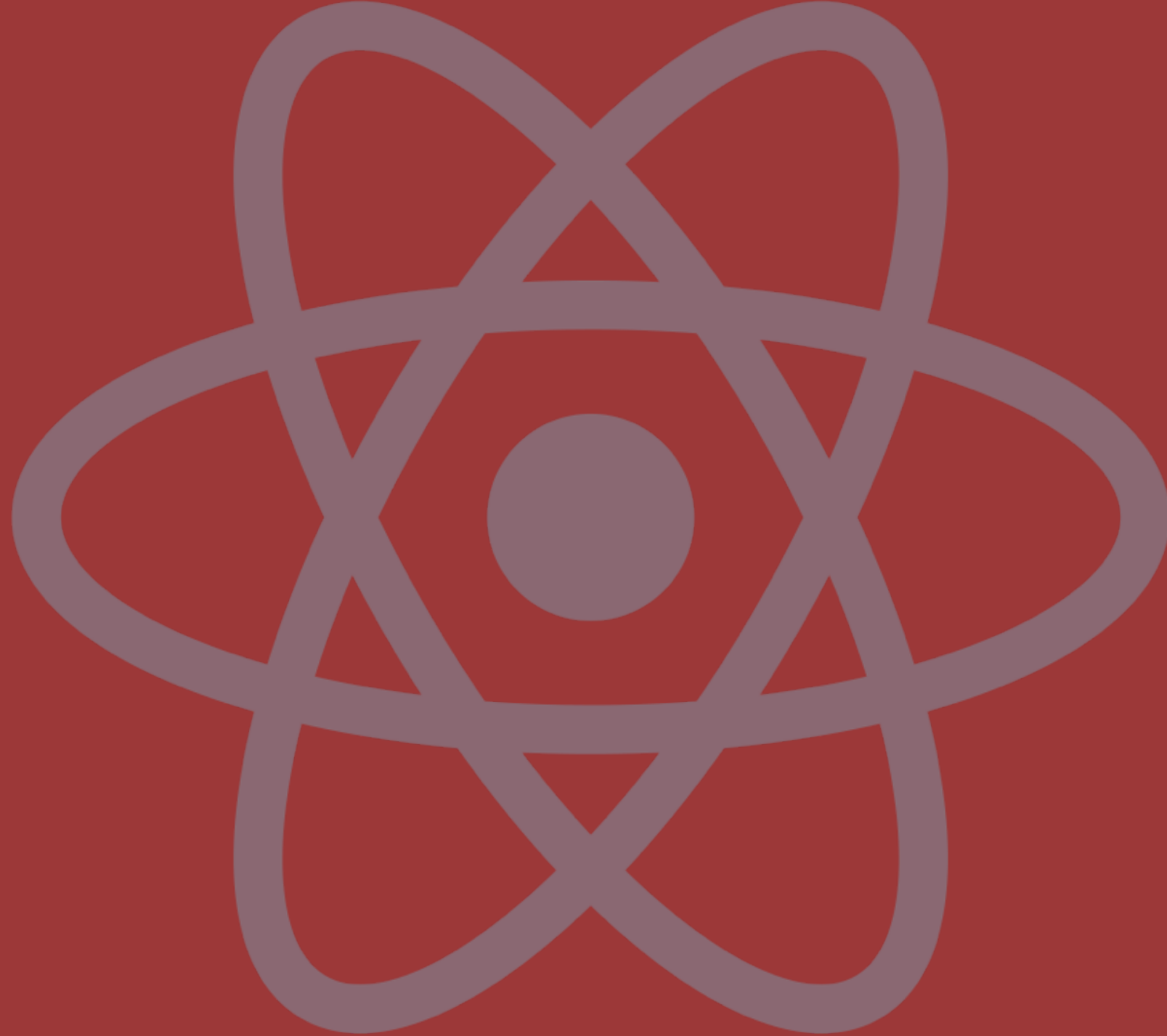
Reactのこれまで





2013 2014 2015 2016

Reactの特徴



Reactの特徴

React

A JAVASCRIPT LIBRARY FOR BUILDING USER INTERFACES

Get Started

Download React v15.3.0

Declarative

React makes it painless to create interactive UIs. Design simple views for each state in your application, and React will efficiently update and render just the right components when your data changes.

Declarative views make your code more predictable and easier to debug.

Component-Based

Build encapsulated components that manage their own state, then compose them to make complex UIs.

Since component logic is written in JavaScript instead of templates, you can easily pass rich data through your app and keep state out of the DOM.

Learn Once, Write Anywhere

We don't make assumptions about the rest of your technology stack, so you can develop new features in React without rewriting existing code.

React can also render on the server using Node and power mobile apps using **React Native**.

```
class App extends React.Component {
  constructor(...args) {
    super(...args);
    this.state = {
      text: '',
    };
  }
  render() {
    return (
      <TextBox
        text={this.state.text}
        onChange={text => this.setState({text})}
      />
    );
  }
}

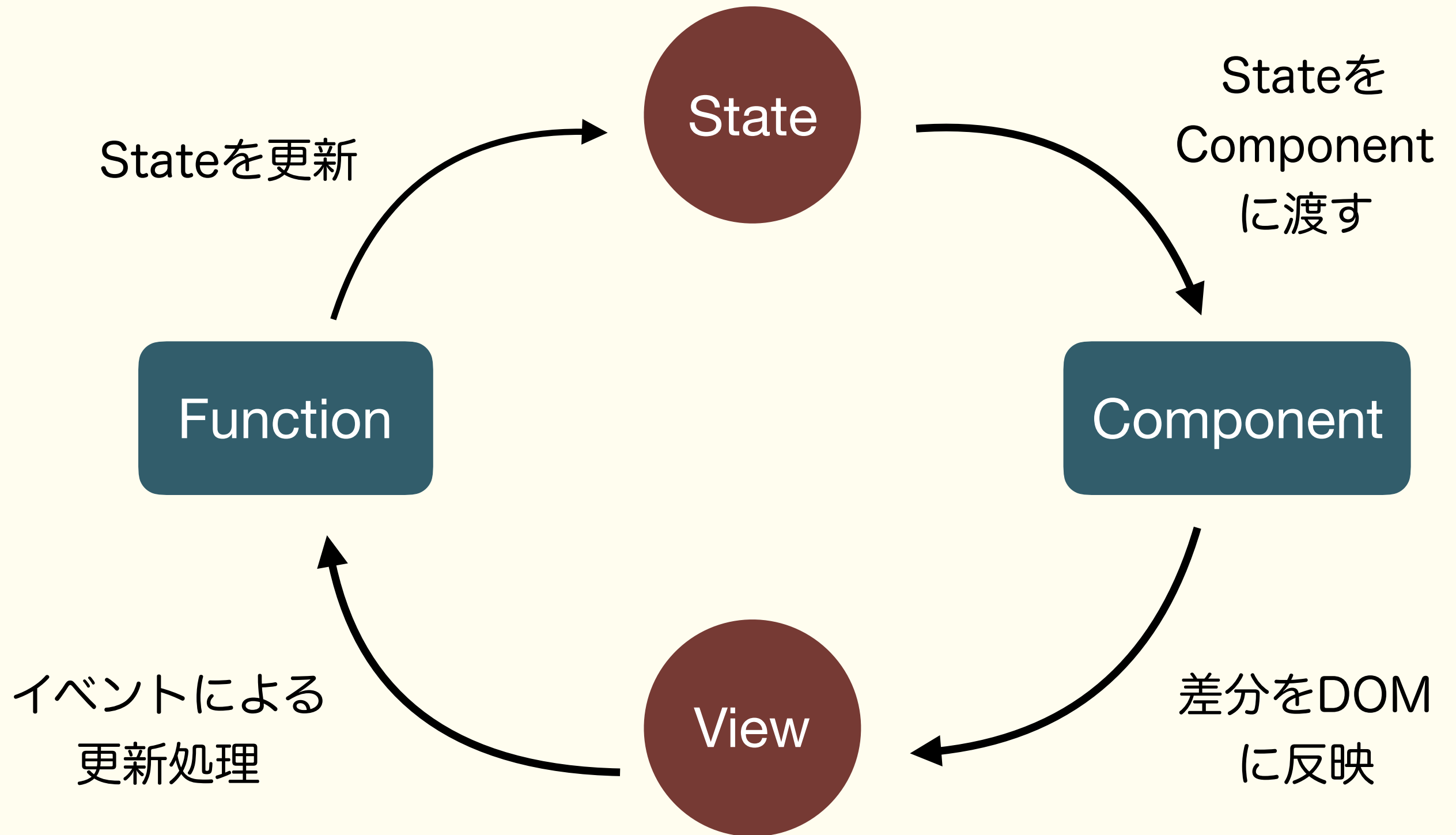
const TextBox = (text, onChange) => (
  <div>
    <input type="text" onChange={e => onChange(e.target.value)} />
    <p>{text}</p>
  </div>
);

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

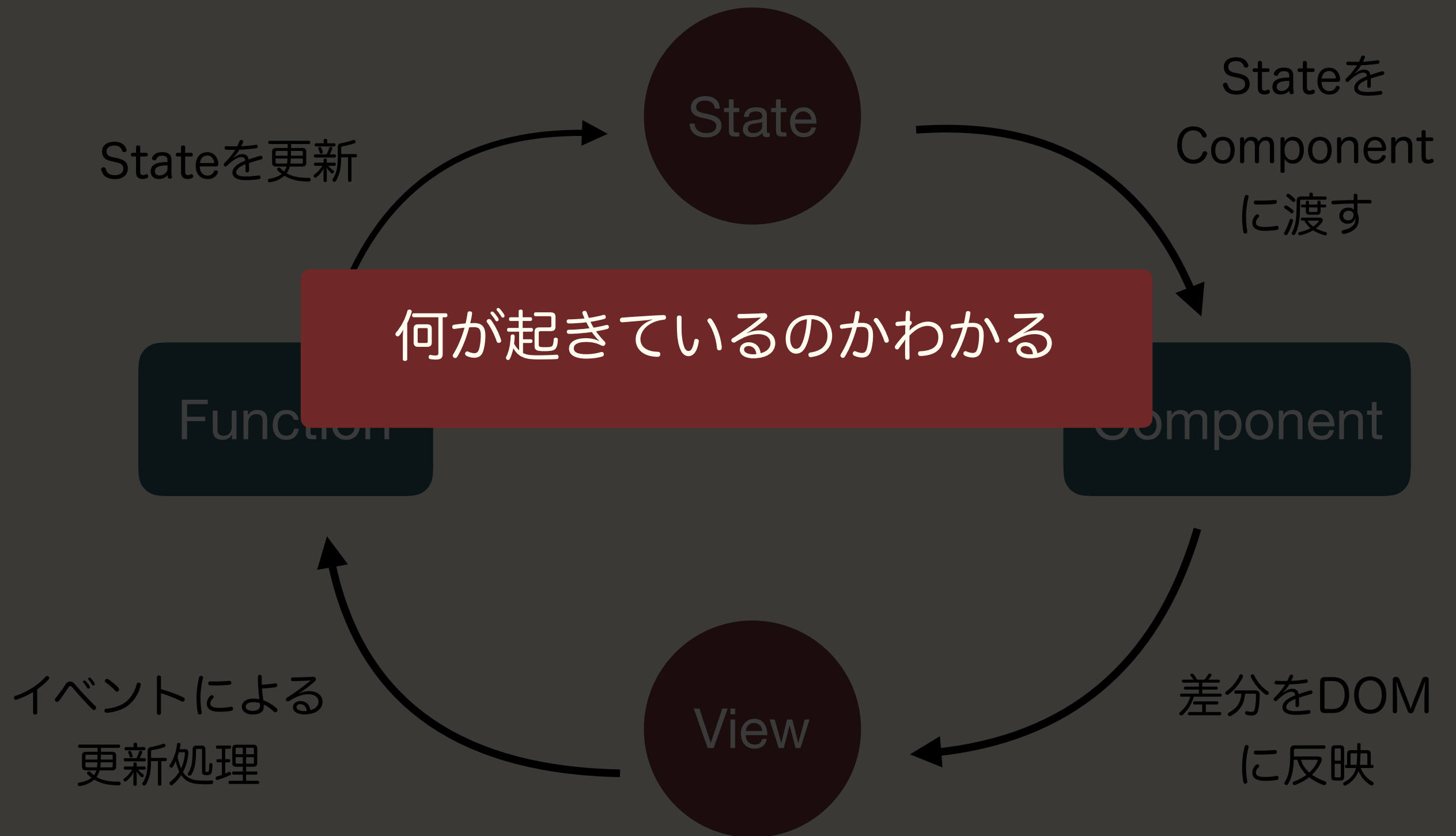

Reactの特徴

- View = Component(State)
 - ComponentはStateからView(ReactElement)を作る
 - ComponentはViewを作る関数
 - Reactが差分を計算してDOMに反映する
 - I/FはProps
- 宣言的にViewを構築する
 - 宣言的？

Reactの特徴



Reactの特徴



宣言的？

- ・「変化」ではなく「状態」を定義する

```
// 命令的
// elの変化を書いている
button.on('click', () => el.append(child));

// 宣言的
// stateに対するあるべき表示を書いている
render = state => {
  el.innerHTML = state.map(child => `<div>${child}</div>`).join('');
};

button.on('click', () => {
  state.push(child);
  render(state);
})
```

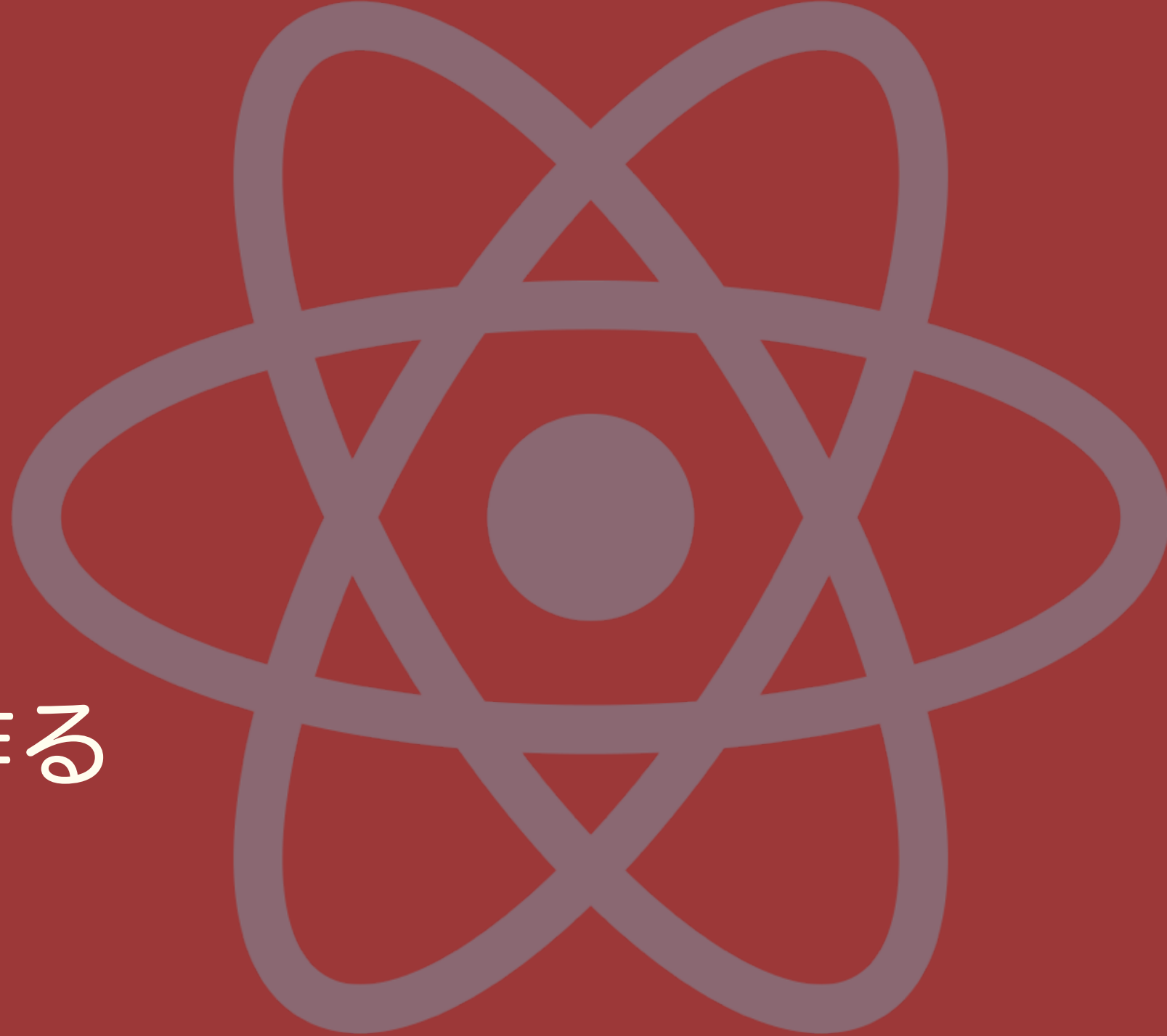

宣言的？

- ・ 初期表示も更新も常にComponent(State)でViewを作る

// 入力に対して常に同じ出力を返す

```
const CommentBox = ({comment, onChange}) => (  
  <div>  
    <p>{comment}</p>  
    <input  
      type="text"  
      value={comment}  
      onChange={e => onChange(e.target.value)}  
    />  
  </div>  
);  
  
const render = comment => (  
  ReactDOM.render(  
    <CommentBox comment={comment} onChange={render} />,  
    document.getElementById('app')  
  )  
);
```

Componentを作る



Componentの定義方法

- Stateless Functional Components
- React.Component
- React.PureComponent
- React.createClass

Stateless Function Components(SFC)

- First Choice
- インスタンスも持たないただの関数
 - `ReactElement = Component(Props)`
 - No state, No lifecycle methods, No refs.
- 将来的な最適化も

```
// Propsを受け取り、ReactElementを返す関数
const Item = ({item}) => (
  <div>
    <div>{item.name}×{item.count}</div>
  </div>
);
// <Item item={{name: 'foo', count: 1}} />
```


React.Component

- Stateやライフサイクルメソッドが必要になったら

```
class App extends React.Component {
  constructor(...args) {
    super(...args);
    this.state = {
      user: null,
    };
  }
  componentDidMount() {
    fetch('/api/user')
      .then(res => res.json())
      .then(user => this.setState({user}))
      ;
  }
  render() {
    if (this.state.user == null) return <Loading />;
    return (
      <div>
        <User user={this.state.user} />
      </div>
    );
  }
}
```

React.Component

- Stateには表示に必要なデータだけ

```
class App extends React.Component {
  constructor(...args) {
    super(...args);
    // this.state = {
    //   timerId: null,
    // };
  }
  componentDidMount() {
    this.timerId = setInterval(() => {}, 1000);
    // this.setState({
    //   timerId: setInterval(() => {}, 1000)
    // });
  }
  componentWillUnmount() {
    clearInterval(this.timerId);
    // clearInterval(this.state.timerId);
  }
  render() {
  }
}
```

React.PureComponent (v15.3.0～)

- ShouldComponentUpdateにPropsとStateに対するShallowEqualが適用される
- 将来的にはChildrenのSFCに対しての最適化が入るかも？

```
class Counter extends React.PureComponent {
  constructor(...args) {
    super(...args);
    this.state = {count: 0};
  }
  render() {
    return (
      <div>
        <p>{this.props.label}:{this.state.count}</p>
        <button
          onClick={() => this.setState({count: this.state.count + 1})}
        />
      </div>
    );
  }
}
```

PureComponentの落とし穴

```
class Item extends React.PureComponent {
  render() {
    return (
      <div>
        <p>{this.props.name}</p>
        <button onClick={this.props.onClick}>click</button>
        {this.props.children}
      </div>
    );
  }
}

// props.onClick !== nextProps.onClick
<Item name="foo" onClick={() => console.log('click')} />

// props.children !== nextProps.children
<Item name="foo" onClick={onClick}>
  <div>foo</div>
</Item>
```

推測するな計測せよ

- react-addons-perf

```
import Perf from 'react-addons-perf';

Perf.start();

ReactDOM.render(<App name="React" />, el);

setTimeout(() => {
  ReactDOM.render(<App name="React" />, el);
  Perf.stop();
  Perf.printWasted();
}, 1000);
```

(index)	Owner > Component	Inclusive wasted time (ms)	Instance count	Render count
0	"App"	0.23	1	1
1	"App > Item"	0.14	1	1

React.createClass??

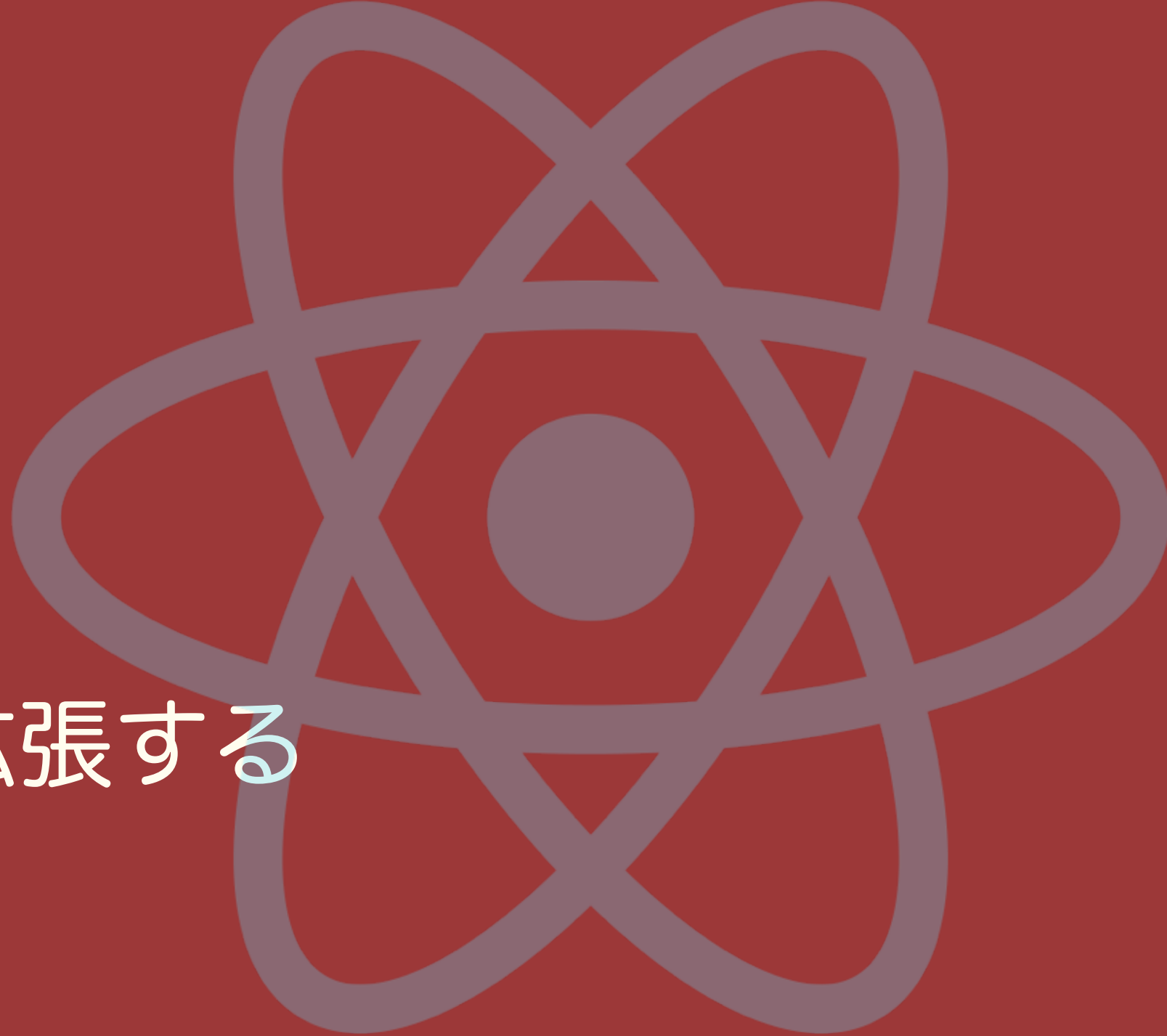
- かつての作成方法。緩やかに非推奨へ
 - 最終的には別パッケージに？
- Facebook内部もReact.Componentへの移行中
- React.createClassの便利機能のマイグレーションパス
 - mixin ▶ High Order Components
 - autobinding ▶ Public Class Fields



Stage2

```
class Button extends React.Component {  
  onClick = () => this.setState({count: this.state.count + 1});  
}
```

Componentを拡張する



High Order Components

- Composition over Inheritance
- High Order Function

```
// 何かする関数を受け取って、結果をログ出して返す
const logger = operation => (...args) => {
  const result = operation(...args);
  console.log(result);
  return result;
};

const add = logger((a, b) => a + b);

add(10, 20);
// 30
```

High Order Components

- High Order Components

```
// Componentを受け取って、PureComponentでラップして返す
const pure = Component => class Pure extends React.PureComponent {
  render() {
    return <Component ...{this.props} />;
  }
}

const Item = ({name}) => <div>{name}</div>;

const PureItem = pure(Item);

// <PureItem name="foo" />
```

High Order Componentsを使う

- acdlite/recompose
 - High Order ComponentsのUtility集
 - <https://github.com/acdlite/recompose>

```
// isLoadingによってComponentを出し分ける
const enhance = branch(
  props => props.isLoading,
  Component => Component,
  () => Loading
);
const LoadUser = enhance(User);

// <LoadUser isLoading={isLoading} user={user} />
```

High Order Components事例

- reactjs/react-router (withRouter)

```
const MyPage = ({router}) => <div>hoge</div>;  
const WithRouterMyPage = withRouter(MyPage);
```

- reactjs/react-redux (connect)

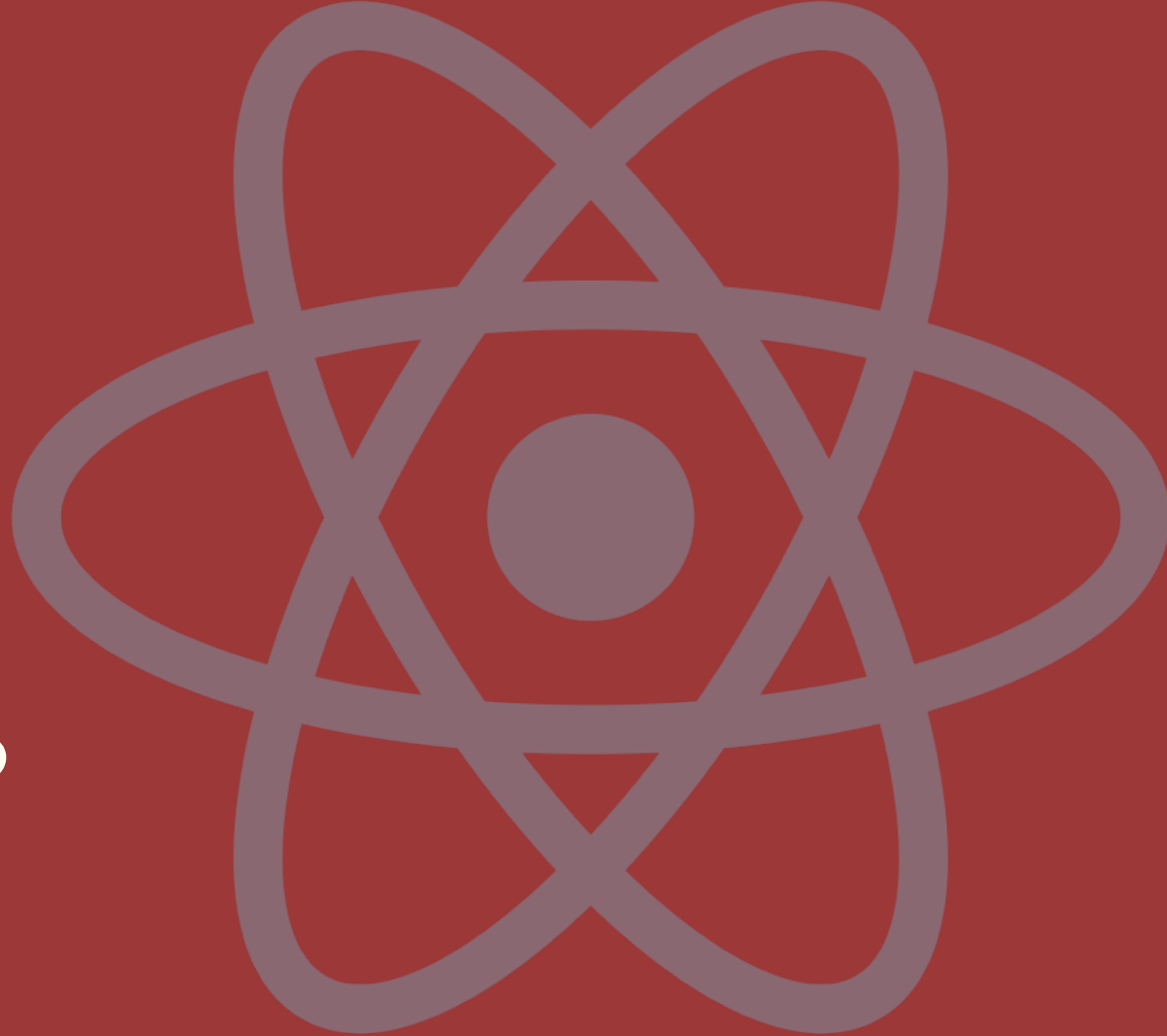
```
const MyPage = ({user, updateName}) => <div>{user.name}</div>;  
const ConnectedMyPage = connect(  
  mapStateToProps,  
  mapDispatchToProps  
) (MyPage);
```


Function as Child Components

- props.childrenに関数を渡すことで拡張する
 - 無駄なComponentによるラップが発生しない
 - Propsとの衝突を考えなくていい
 - 最適化が難しい
 - react-motion
- <https://medium.com/merrickchristensen/function-as-child-components-5f3920a9ace9>

```
FuncAsChild = props => <div>{props.children('foo')}</div>;  
// <FuncAsChild>{foo => <span>{foo}</span>}</FuncAsChild>
```

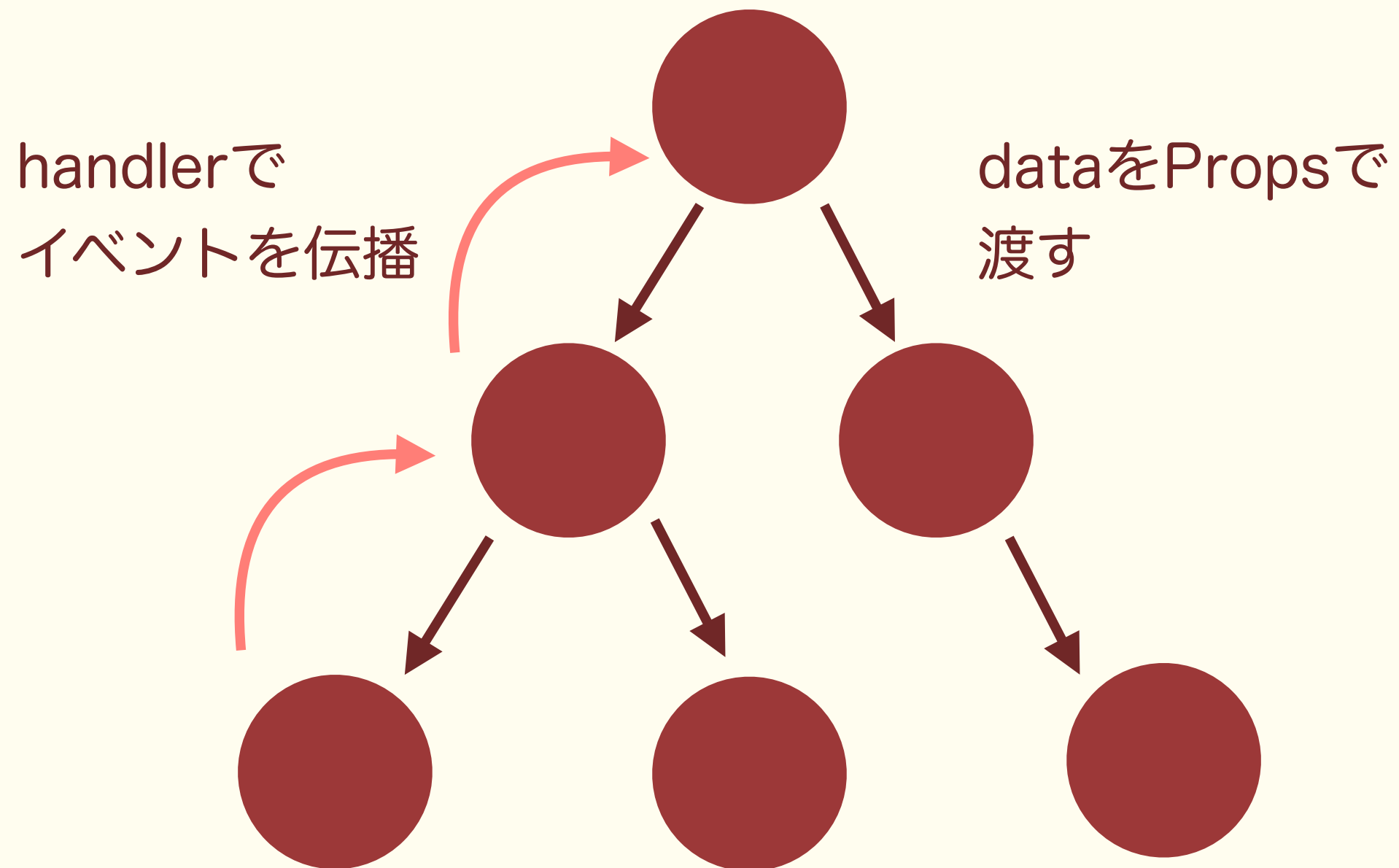
Stateを管理する



State管理

- ・ アプリケーションの状態はなるべく一箇所に。
 - ・ 親のComponent
 - ・ 親は子にデータと操作（関数）を渡す。子はそれらを使うだけ
- ・ 完全にComponent内に閉じる状態はComponentのStateにすることも考える
- ・ アプリケーションが大きくなると、親のComponentで管理しきれなくなる

Dataflow

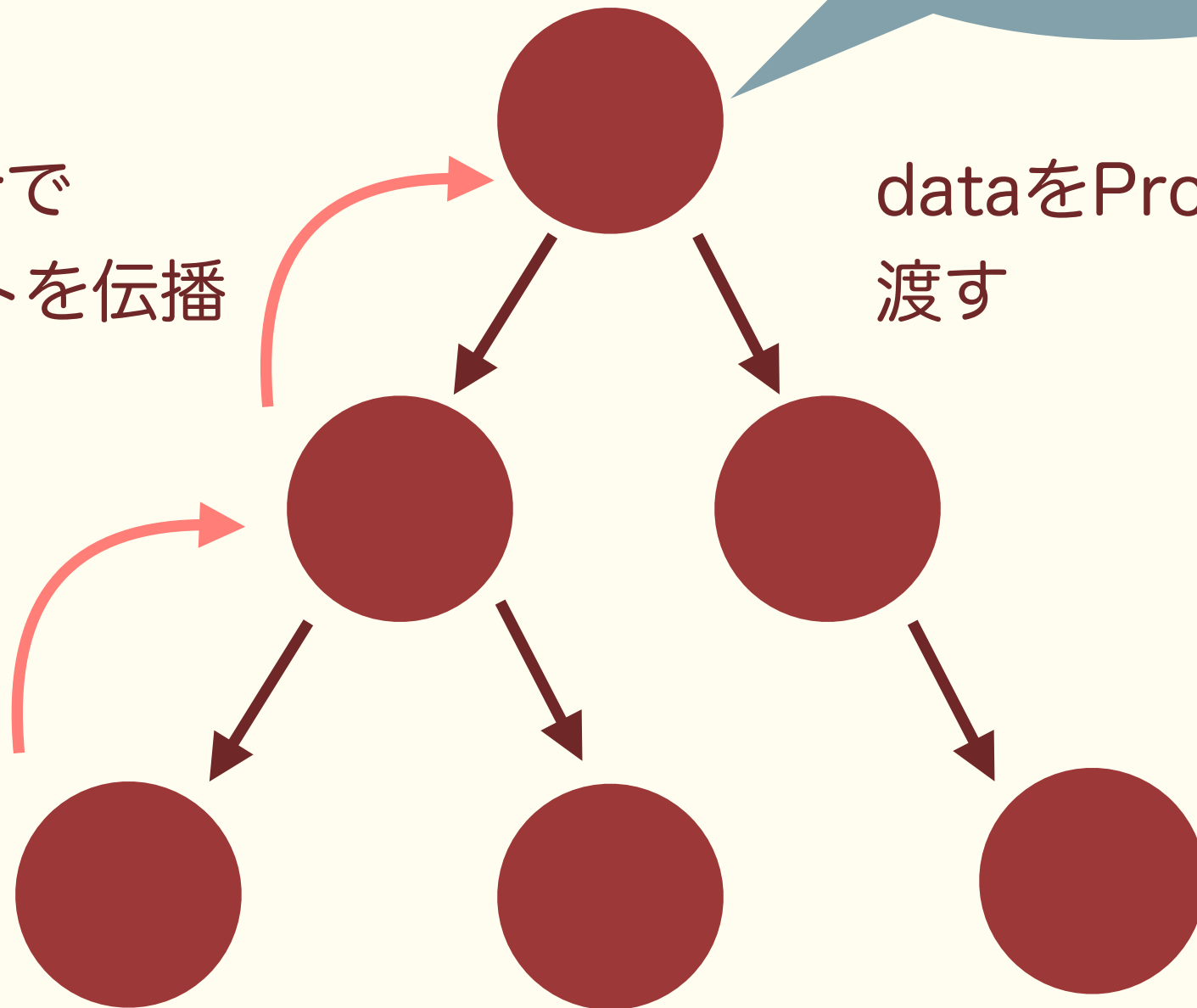


Dataflow

アプリケーションの規模が
大きくなってきたら？

handlerで
イベントを伝播

dataをPropsで
渡す



Redux

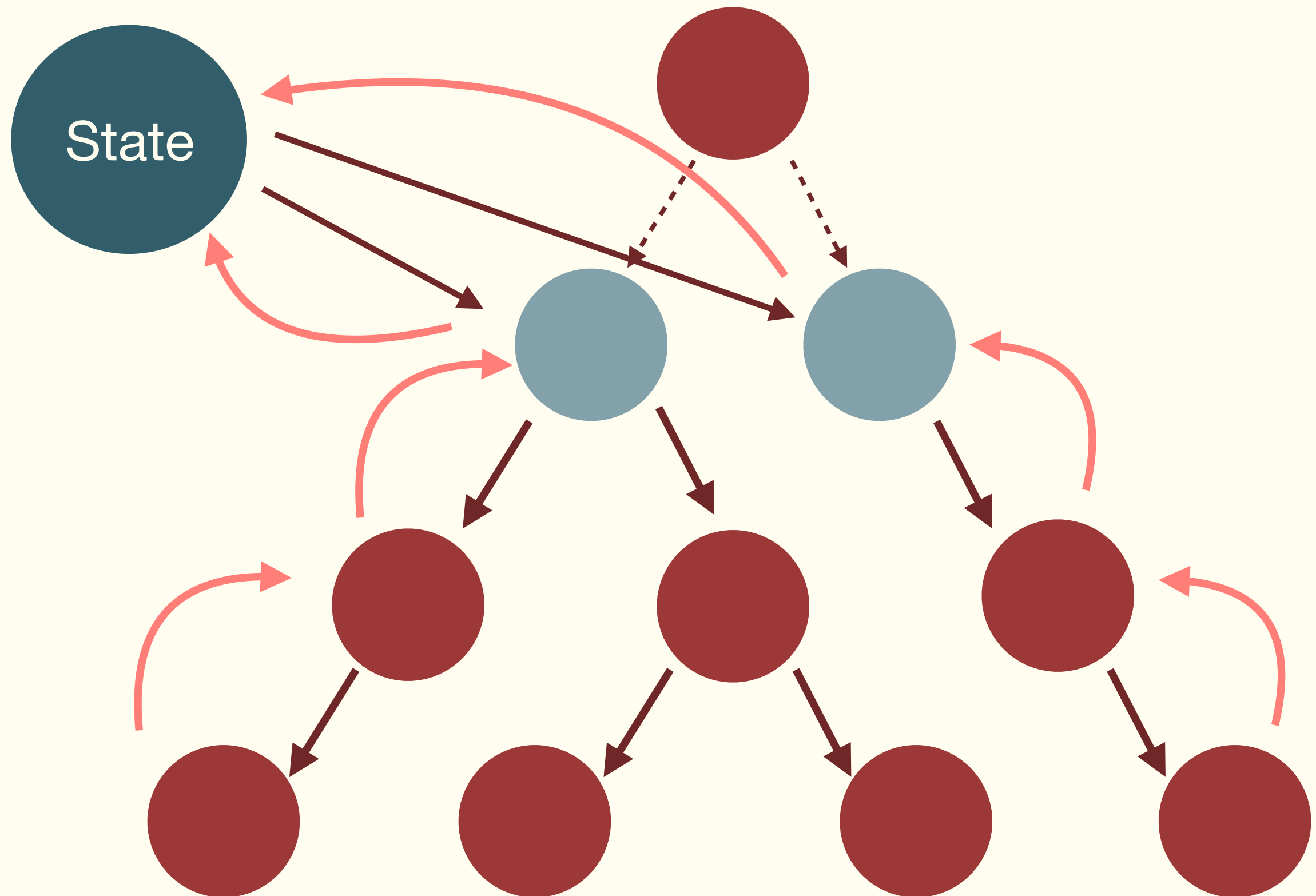
- ・ StateをComponentから切り離す
 - ・ アプリケーションの状態 (State) をSingle Storeで管理
- ・ Stateの更新をActionとして表現する
 - ・ StateはActionで更新される
 - ・ Stateの更新はReducerと呼ばれるPure Functionで
- ・ Container ComponentとPresentational Component

Redux

- ・ StateをComponentから切り離す
 - ・ アプリケーションの状態 (State) をSingle Storeで管理
- ・ Stateの更新
 - ・ Stateはimmutableで更新される
 - ・ Stateの更新はReducerと呼ばれるPure Functionで
- ・ Container ComponentとPresentational Component

必要になるまで使う必要はない

Dataflow



Container ComponentとPresentational Component

- ・同じComponentでも役割は全く異なる

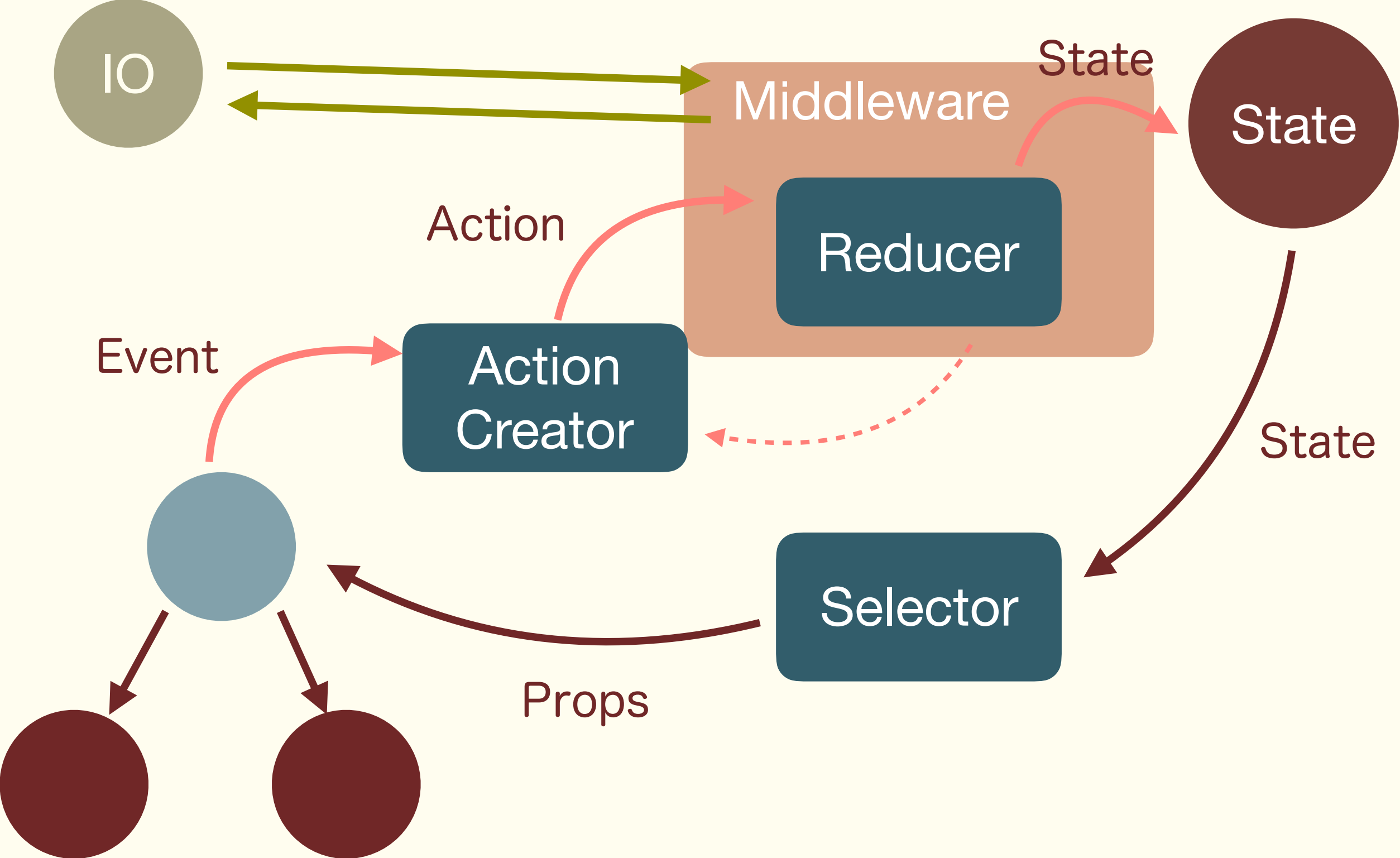
Container Component

- ・更新処理、状態管理を担う
- ・React.(Pure)Component
- ・DOMを知らない
- ・connectで作成(Redux)

Presentational Component

- ・Viewの構築を担う
- ・SFC
- ・Domain Logicを知らない
- ・State管理のことは知らない

Redux



Redux

- `action = ActionCreator([event]);`
 - ・アプリケーションで発生するイベント
- `newState = Reducer(state, action);`
 - ・アプリケーションの状態更新
- `props = Selector(state);`
 - ・Viewに必要なデータ

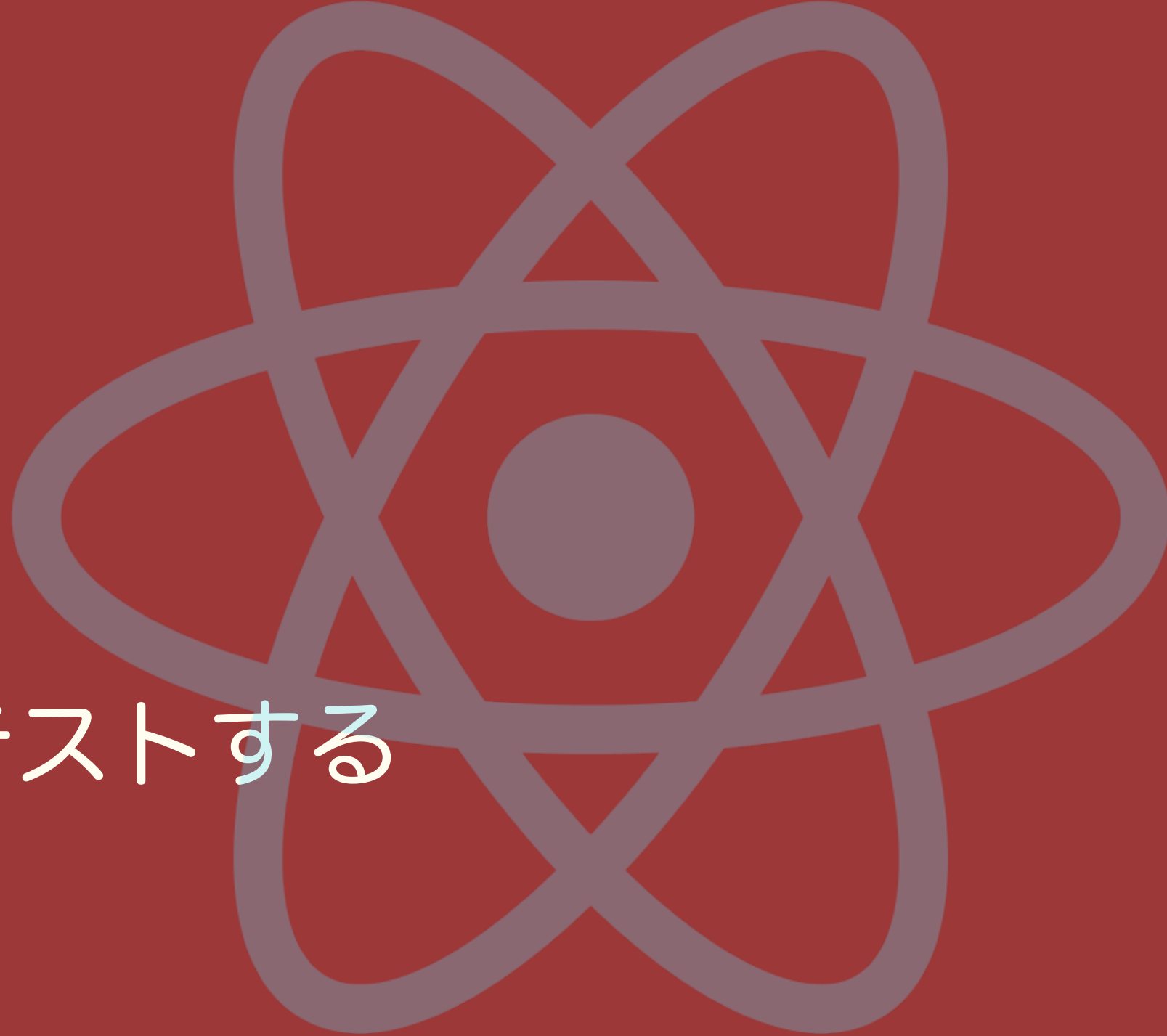
Redux

- 単一のStateに全てが詰まっている
 - ▶ Stateを見ればアプリケーションの状態がわかる
 - ▶ View = Component(State)
- データとロジックの分離
- 多くの部分が副作用のない関数になるのでテストが簡単
- 副作用は??

Redux Middleware

- 副作用や非同期処理は、Middlewareを使って処理する
 - Middlewareでは、ReducerがActionを処理する前後に割り込み、Actionの加工やキャンセル、別のActionの発行などが可能
 - `redux-thunk`, `redux-promise`, `redux-saga`, `redux-observable`, `redux-loop`, `redux-effects`...
 - `redux-persist`, `redux-logger`, `redux-analytics`

Componentをテストする



Test

- TestUtils.renderIntoDocument
 - ・ DOMが必要なテスト
- TestUtils.ShallowRender
 - ・ Componentの単体テスト
- react-test-renderer
 - ・ Component Treeに対するテスト

ShallowRender

- ・ Component単体に対するテストが可能
 - ・ 子Componentはrenderされない
- ・ Node環境でテストが可能（jsdomなどは不要）
- ・ Refsは未サポート、Lifecycle Methodsは一部のみサポート
 - ・ “Shallow” Rendering

```
const shallowRenderer = TestUtils.createRenderer();  
const elementTree = shallowRenderer.render(<YourComponent />);  
assert(elementTree.props.children[0].type === 'div');
```

airbnb/enzyme

- React Componentsに対するテストのUtility
- shallow, mount, render
- TestUtilsのAPIを使うより、簡単に、わかりやすく書くことが可能

```
import {shallow} from 'enzyme';

const wrapper = shallow(<YourComponent />);
assert(wrapper.find(Link).prop('to') === '/foo');
wrapper.find('button').simulate('click');
```

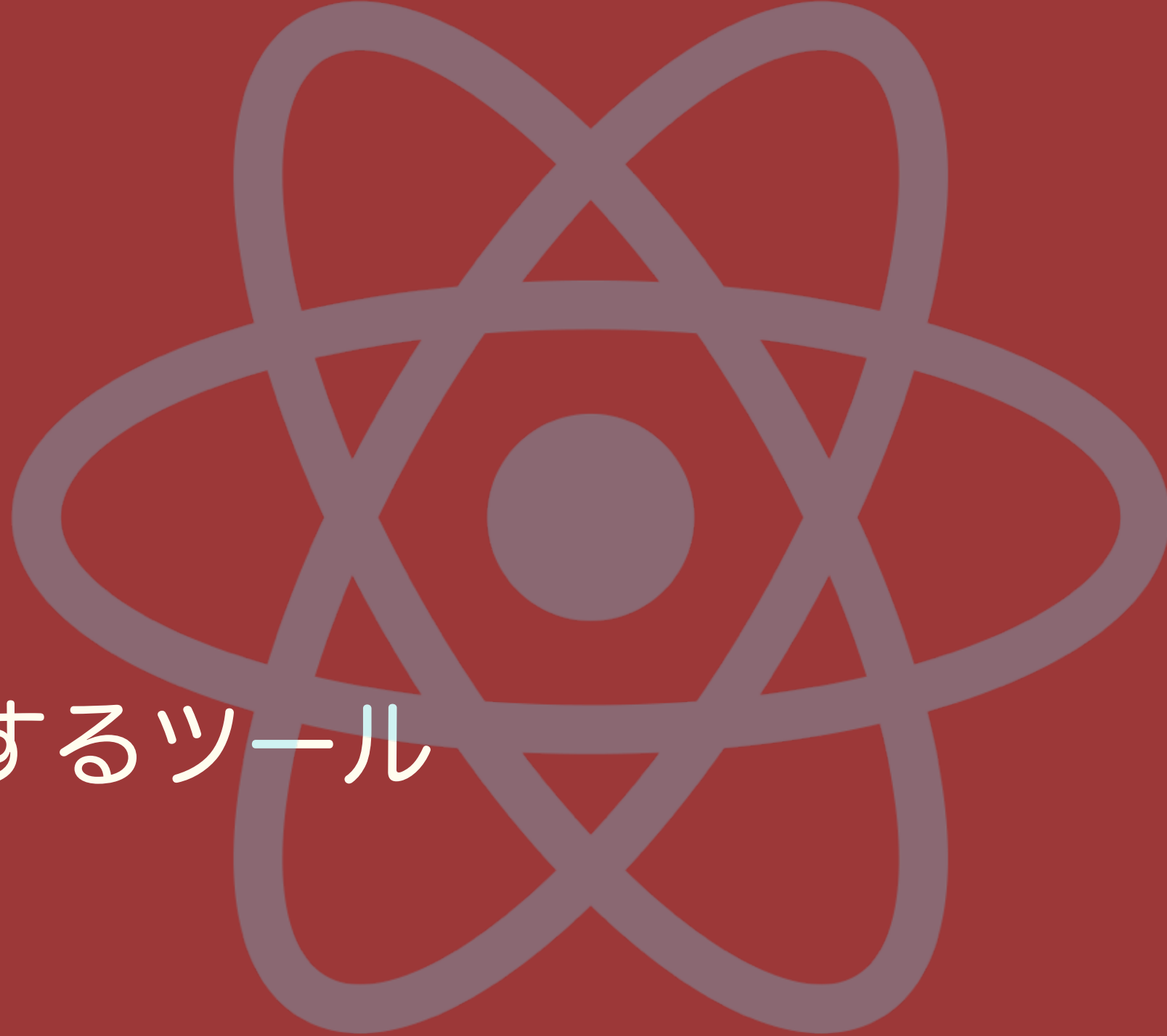
react-test-renderer (v15.3.0～)

- ・ ReactElementのツリーをJSONにして返す
 - ・ Not “Shallow”
 - ・ Lifecycle Methodsもサポートされている
- ・ APIはまだまだ整備中
- ・ Jestのsnapshot testingで使われている

```
import renderer from 'react-test-renderer';

const tree = renderer.create(<YourComponent />).toJSON();
assert(tree.props.children[0].type === 'div');
```

開発をサポートするツール



ESLint 🧡 React

- eslint-plugin-react
 - Reactに関するESLintのルール集
 - Reactのベストプラクティスを知ることができる
 - <https://github.com/yannickcr/eslint-plugin-react>
- eslint-plugin-jsx-a11y
 - JSXに対するaccessibilityのチェックができる
 - <https://github.com/evcohen/eslint-plugin-jsx-a11y>

ESLint ❤️ React

```
1  import React from 'react';
2  import ReactDOM from 'react-dom';
3
4  • const App = ({name}) => (
5    <div
6    •   onClick={() => console.log('click')}
7      style={{width: '100'}}
8    >
9    •   <p>Hello {name}</p>
10   •   
11   </div>
12 );
```

ESLint Error react/jsx-no-bind JSX props should not use arrow functions

ESLint Error react/prop-types 'name' is missing in props validation at line 4 col 15

ESLint Error react/jsx-no-bind JSX props should not use arrow functions at line 6 col 5

ESLint Error react/jsx-indent Expected indentation of 4 space characters but found 6. at line 9 col 7

ESLint Error react/jsx-indent Expected indentation of 4 space characters but found 6. at line 10 col 7

ESLint Error jsx-a11y/img-has-alt img elements must have an alt prop or use role="presentation". at line 10 col 7

ESLint React

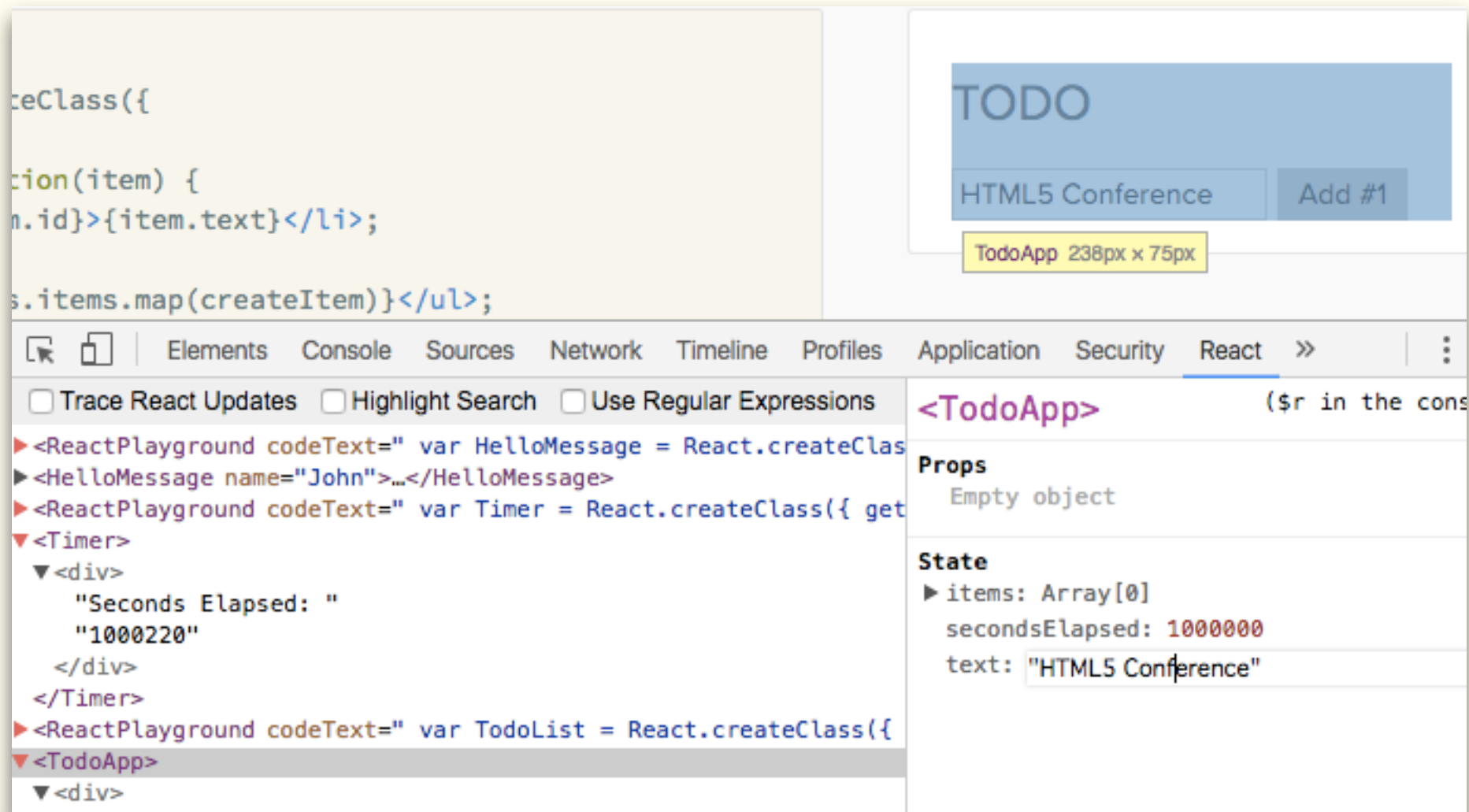
- no-string-refs … 文字列によるRefs指定の禁止
- prefer-stateless-function … SFCによる定義を優先する
- no-direct-mutation-state … this.stateを直接更新を禁止
- sort-comp … Component内のメソッド定義順をチェック
- jsx-no-bind … Propsでの.bind()やArrowFunctionの禁止
- jsx-key … keyのPropsが必要な場面で指定されているか
- See more rules
 - <https://github.com/yannickcr/eslint-plugin-react>

React StoryBook

- <https://getstorybook.io/>
- Componentをアプリケーションから切り離した形で開発できる
- StoryとしてComponentの表示パターンを定義できる
- Storybooks.io
 - <https://storybooks.io/>

React Developer Tools

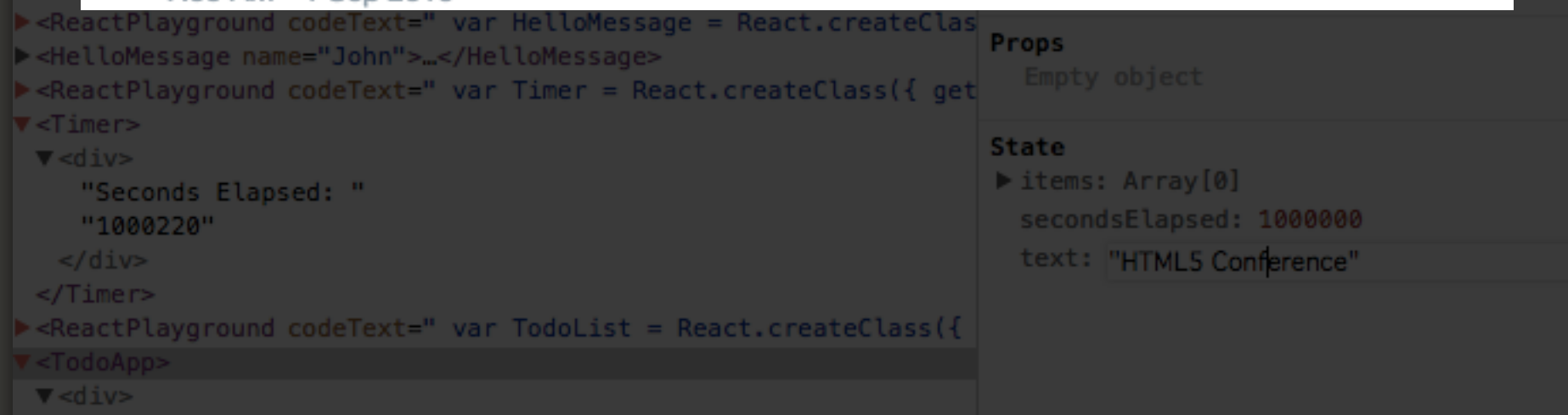
- ReactElementのTreeをinspectできる
- Stateの書き換えも可能



React Developer Tools

- ReactElementのTreeをinspectできる

- Stateの管理



Redux DevTools

- <https://github.com/gaearon/redux-devtools>
- タイムトラベルデバッキング
- Actionの書き換え
- https://twitter.com/dan_abramov/status/761549682178916352



Dan Abramov
@dan_abramov

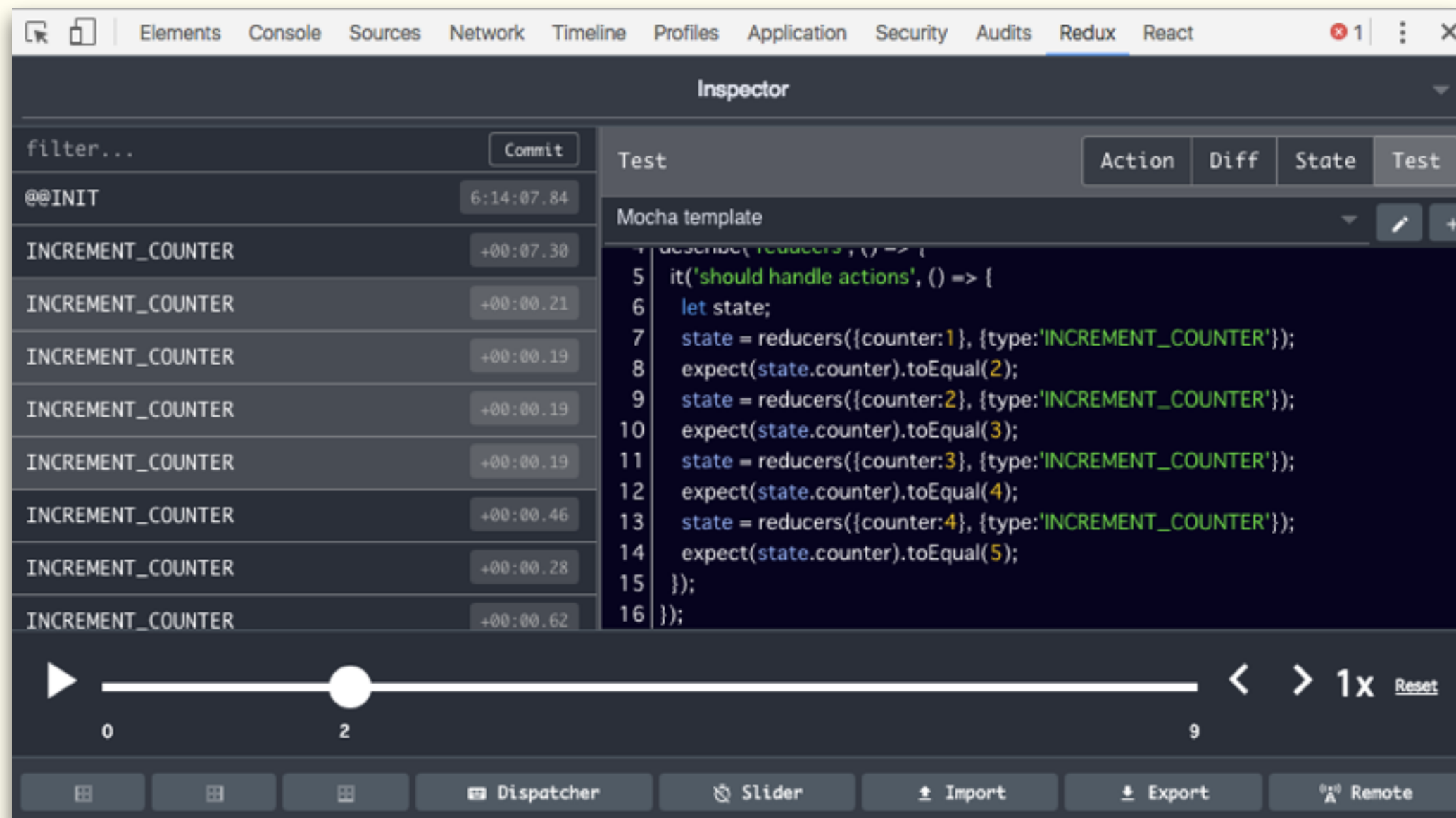


Following

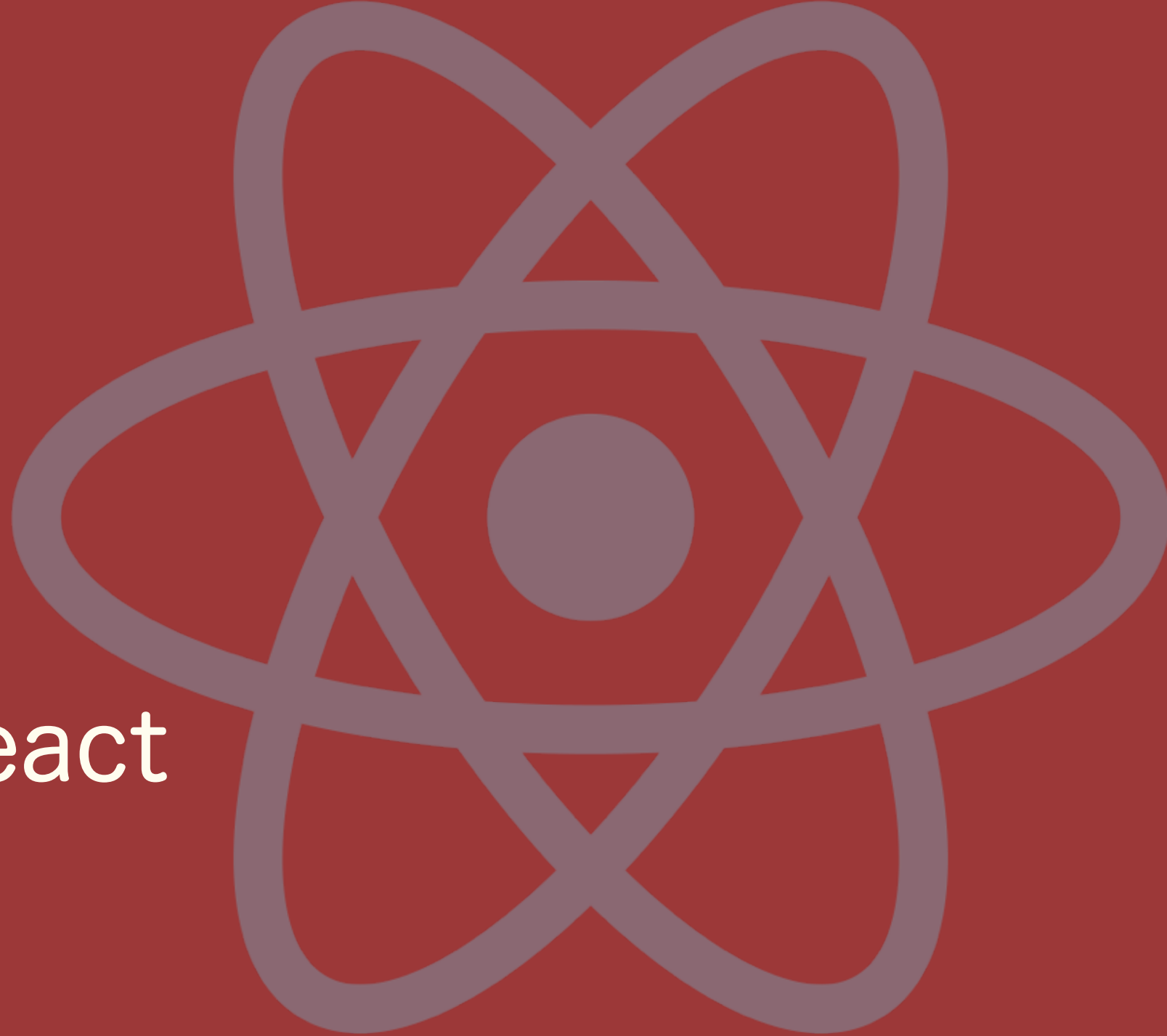
Redux DevTools make me sad because I intended it for a totally different workflow, but failed to explain it in the UI.

Redux DevTools Extension

- <https://github.com/zalmoxisus/redux-devtools-extension>
- 履歴のExport/Import、テストケースの作成...



Real WorldとReact



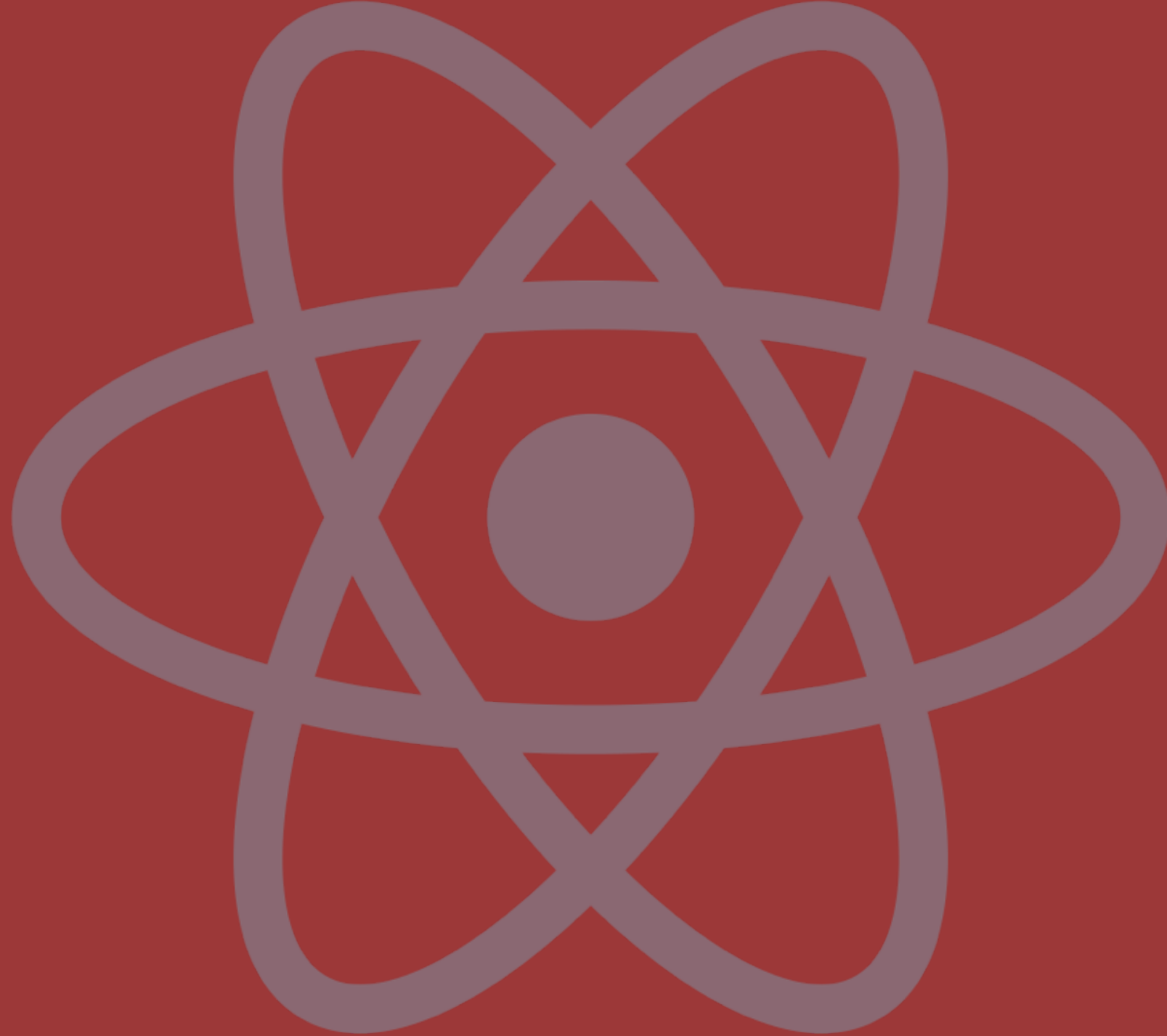
FacebookとReact

- ReactはFacebookが使うために作られているライブラリー
 - over 25,000 components
 - “We only open source what we use”
- OSSとしてのReact
 - SVG, custom elements
 - facebookincubator/create-react-app
 - reactjs/core-notes

Real WorldとReact

- Deprecate Process
 - 1つ前のメジャーバージョンで警告してから廃止される
 - 移行パスを提供
- Gradual Adoption
 - 既存のアプリケーションに組み込みやすいAPIの提供
 - React with jQuery, Canvas...
- Escape Hatches
 - Refs, dangerouslySetInnerHTML, Context

Future?



ReactCore

- Coreはどんどん小さくAPIは最低限に
- react
 - ComponentやReactElementを作成する
- renderer
 - Componentをプラットフォームに応じて処理する
 - DOMもただの1ターゲット
 - react-dom, react-native, react-art …

```
% tree src -L 3 -d
```

```
.
```

```
├── addons
```

```
├── isomorphic
```

```
│   ├── children
```

```
│   ├── classic
```

```
│   ├── hooks
```

```
│   └── modern
```

```
├── renderers
```

```
│   ├── art
```

```
│   ├── dom
```

```
│   │   ├── client
```

```
│   │   ├── fiber
```

```
│   │   ├── server
```

```
│   │   └── shared
```

```
│   ├── native
```

```
│   ├── noop
```

```
│   ├── shared
```

```
│   │   ├── fiber
```

```
│   │   ├── hooks
```

```
│   │   ├── shared
```

```
│   │   ├── stack
```

```
│   │   └── utils
```

```
│   └── testing
```

```
├── shared
```

```
├── test
```

```
└── umd
```

[WIP]ReactFiber 🙄🙄

- ・ 内部アルゴリズムの全面書き換え(Reconciler)
- ・ 優先度に応じてスケジューリングされた非同期なViewの更新
- ・ 現在は差分の検出・適用が同期的に行なわれている
 - ・ 16ms以内を目指そうとすると...
- ・ 優先度の高い処理(requestAnimationFrame)
- ・ 優先度の低い処理(requestIdleCallback)

[WIP]ReactFiber 🙄🙄

```
// renderers/shared/fiber/ReactPriorityLevel.js

module.exports = {
  // No work is pending.
  NoWork: 0,

  // For controlled text inputs. Synchronous side-effects.
  SynchronousPriority: 1,

  // Needs to complete before the next frame.
  AnimationPriority: 2,

  // Interaction that needs to complete pretty soon to feel responsive.
  HighPriority: 3,

  // Data fetching, or result from updating stores.
  LowPriority: 4,

  // Won't be visible but do the work in case it becomes visible.
  OffscreenPriority: 5,
};
```



<Session author="@koba04">

小さくはじめて、必要に応じてエコシステムを
利用しましょう！

</Session>

speakerdeck.com/koba04