React Native 初探

by johnwatsondev

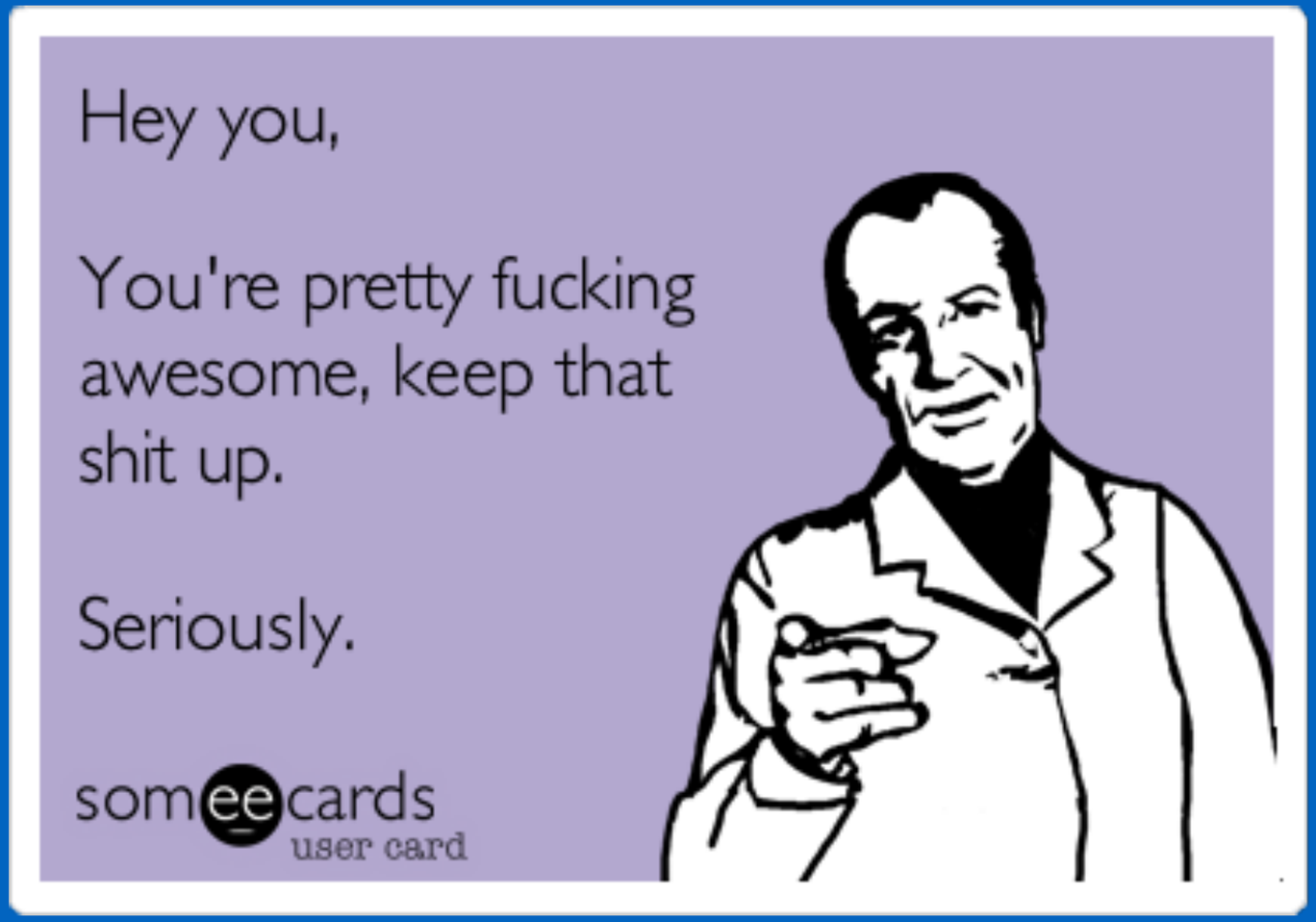
Prerequisites

- HTML & CSS
- JavaScript
  Some ES6 Feature: arrow function/classes/let/const
- React
  props/state/lifting state up/JSX
- Tips
  Please check **Official Doc** or use **Google** to comprehend these concepts

Quiz

```
class DisplayText extends React.Component {
    render() {
        let displayText = this.props.isFamiliarBasicConcepts ? 'Let\'s go baby~' : 'Go home please.';
        return (
            <View style={{flex: 1, flexDirection: 'column'}}>
                <Text>{displayText}</Text>
            </View>
        );
    }
}
```

Result - Positive



Result - Negative

What

- **Learn once, write anywhere**
- Build native mobile apps using **JavaScript** and **React**

Talk is cheap

```
class DisplayText extends React.Component {
    _renderPicture() {
        let picUrl = 'dummyImgUrl';
        if (!this.props.isFamiliarBasicConcepts) {
            return (
                <Image style={{width: 200, height: 100}} source={{uri: picUrl}} resizeMode='stretch'/>
            );
        } else {
            return null;
        }
    }
    render() {
        let displayText = this.props.isFamiliarBasicConcepts ? 'Let\'s go baby~' : 'Go home please.';
        return (
            <View style={{flex: 1, flexDirection: 'column'}}>
                <Text>{displayText}</Text>
                {this._renderPicture()}
            </View>
        );
    }
}
```

Talk is cheap

```
class DisplayText extends React.Component {
    _renderPicture() {
        let picUrl = 'dummyImgUrl';
        if (!this.props.isFamiliarBasicConcepts) {
            return (
                <Image style={{width: 200, height: 100}} source={{uri: picUrl}} resizeMode='stretch'/>
            );
        } else {
            return null;
        }
    }
    render() {
        let displayText = this.props.isFamiliarBasicConcepts ? 'Let\'s go baby~' : 'Go home please.';
        return (
            <View style={{flex: 1, flexDirection: 'column'}}>
                <Text>{displayText}</Text>
                {this._renderPicture()}
            </View>
        );
    }
}
```

Getting value from the outer world

```
class DisplayText extends React.Component {
    _renderPicture() {
        let picUrl = 'dummyImgUrl';
        if (!this.props.isFamiliarBasicConcepts) {
            return (
                <Image style={{width: 200, height: 100}} source={{uri: picUrl}} resizeMode='stretch'/>
            );
        } else {
            return null;
        }
    }
    render() {
        let displayText = this.props.isFamiliarBasicConcepts ? 'Let\'s go baby~' : 'Go home please.';
        return (
            <View style={{flex: 1, flexDirection: 'column'}}>
                <Text>{displayText}</Text>
                {this._renderPicture()}
            </View>
        );
    }
}
```

Getting value from the outer world

```
class DisplayText extends React.Component {
    _renderPicture() {
        let picUrl = 'dummyImgUrl';
        if (!this.props.isFamiliarBasicConcepts) {
            return (
                <Image style={{width: 200, height: 100}} source={{uri: picUrl}} resizeMode='stretch'/>
            );
        } else {
            return null;
        }
    }
    render() {
        let displayText = this.props.isFamiliarBasicConcepts ? 'Let\'s go baby~' : 'Go home please.';
        return (
            <View style={{flex: 1, flexDirection: 'column'}}>
                <Text>{displayText}</Text>
                {this._renderPicture()}
            </View>
        );
    }
}
```

Storing value in your inner world

```
class MyInput extends React.Component {
    constructor() {
        super();
        this.state = {inputText: ''}
    }
    render() {
        return (
            <View style={{marginHorizontal: 40}}>
                <TextInput style={{alignSelf: 'stretch', height: 40}}
                           placeholder="Please input here!"
                           onChangeText={text => {
                               console.log("MyInput onChangeText = " + text);
                               this.setState({inputText: text});
                           }}/>
                <Text>{`Your inputText is ${this.state.inputText}`}</Text>
            </View>
        );
    }
}
```

Storing value in your inner world

```
class MyInput extends React.Component {
    constructor() {
        super();
        this.state = {inputText: ''}
    }
    render() {
        return (
            <View style={{marginHorizontal: 40}}>
                <TextInput style={{alignSelf: 'stretch', height: 40}}
                           placeholder="Please input here!"
                           onChangeText={text => {
                               console.log("MyInput onChangeText = " + text);
                               this.setState({inputText: text});
                           }}/>
                <Text>{`Your inputText is ${this.state.inputText}`}</Text>
            </View>
        );
    }
}
```

Props vs State

Only give props if your data is not going to change over time/in response to user interaction

We can build a completed RN App now

- Text, Image
- Button, TextInput, Switch
- Picker, Slider, SectionList, StatusBar
- View, FlatList, ListView, ScrollView, WebView, etc

Is that enough
in production environment?

Hybrid App

- Part Native
- Part React Native

What's the most important thing?

Communication between JS and Native

- Native Module
- JS Module
- InitialProps

Native Module

JS side

invoke

Native code

JS Module

Native side

invoke

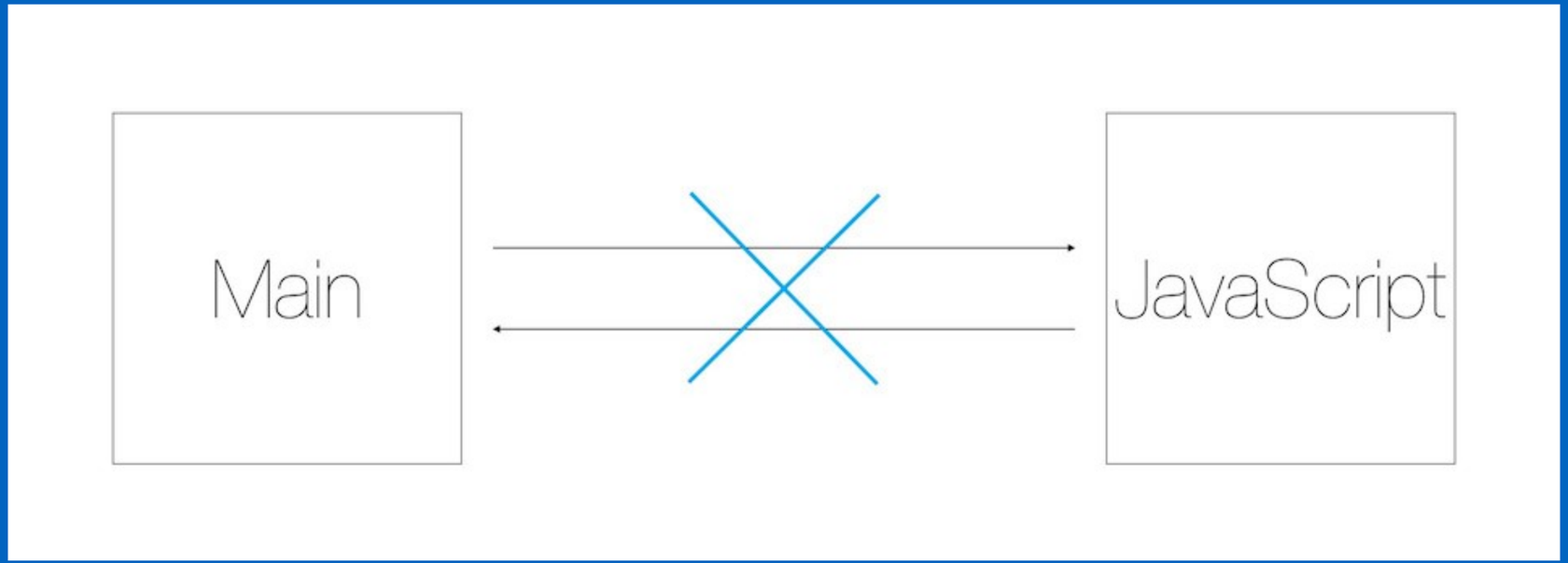
JS code

InitialProps

Native code pass any value as props to the React Native Component

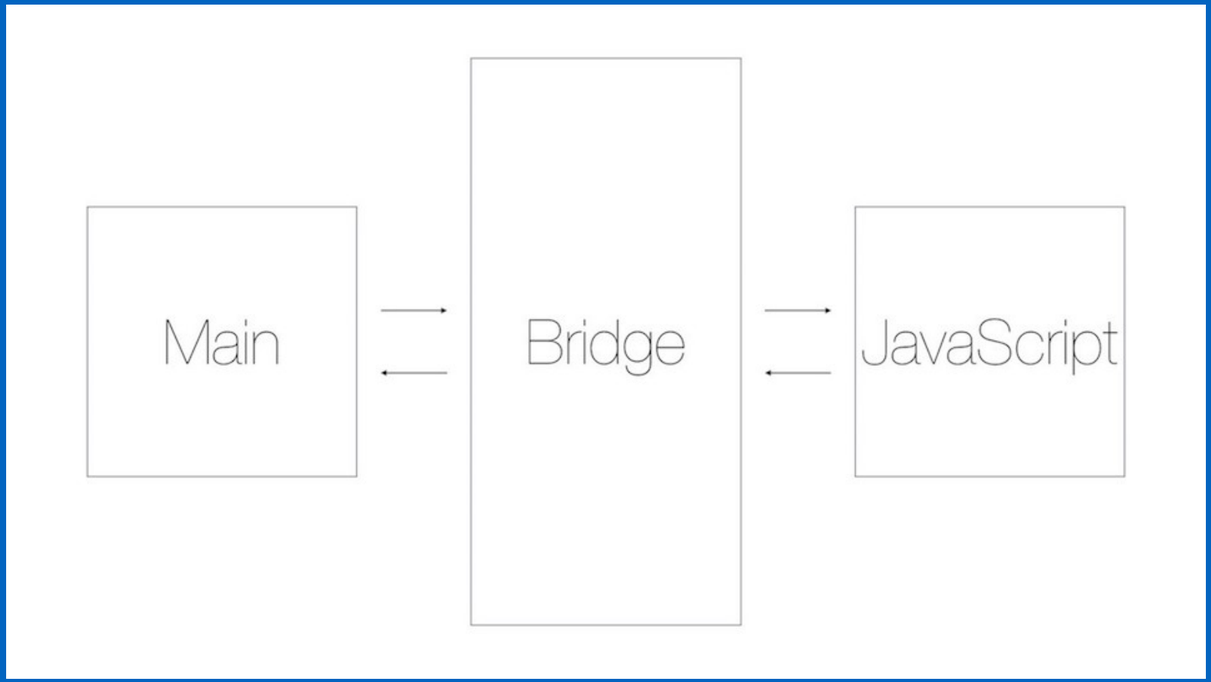
- Bonus: We can invoke **different** React Native Page from the **same native entry**(Activity/UIViewController).

How to communicate?



Indirectly | No-Blocked

How to communicate?



Asynchronous | Batched Message | Serializable

How to communicate?

Please dive into the source code
to get more details. 😎😈😂

React → React Native

- React ***forces*** us to break our applications down into ***discrete components***, each representing a single view.
- React ***raises the level of abstraction and simplifies the programming model***.
- Code is more predictable. (credit: JSX)

Efficient Tools

- WebStrom (Keymap, Live Templates)
- Shell Scripts (Debugging, Build Artifacts)
- Vysor (Just For Android Physical Device)

More Resource

- Awesome React Native
- More Resource

Question

Homework

请各位同学自己搭建一套 RN 环境，成功运行 HelloWorld。

心有余力的同学可以重构下现有业务界面。

Thanks