Software Engineering

08:30-10:00

45-B311

Computer Network

10:20-11:50

46-A108

College Japanese

13:30-15:00

47-B228

Free time

Your own time

Design Psychology1

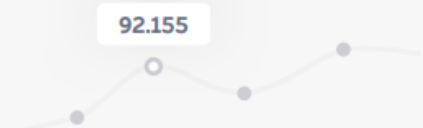
Time:
2018-08-08

User Experience

Time:
2018-07-29

The visual design

Time:
2018-07-26



Total Weighted

89.869

Total Grade

3.869

Bicycle

Tel Num

Learning

More

Presentation Agenda

SECTION 1

React-Native

SECTION 2

Redux

SECTION 3

TypeScript

SECTION 4

4.0 Specs

SECTION 1

React-Native

身为优秀的软件工程师

我们一直在尽力**减少冗余、复用代码**

A&T 21:42

It’s always
Nice to
Have you back.

Username

Password

21:42

It’s always
Nice to
Have you back.

Username

Password

Username

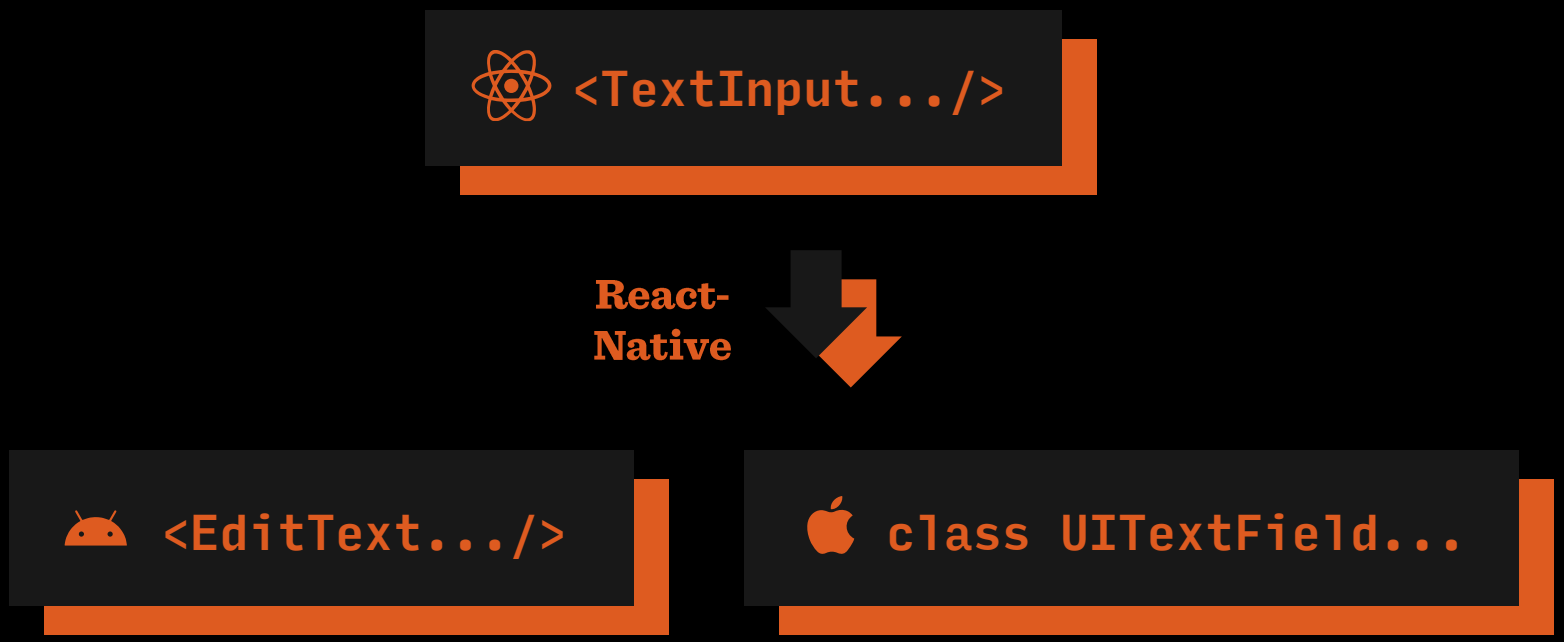
```
class UITextField : UIControl
```

```
<EditText android:id=.../>
```

<TextInput.../>

 <EditText.../>

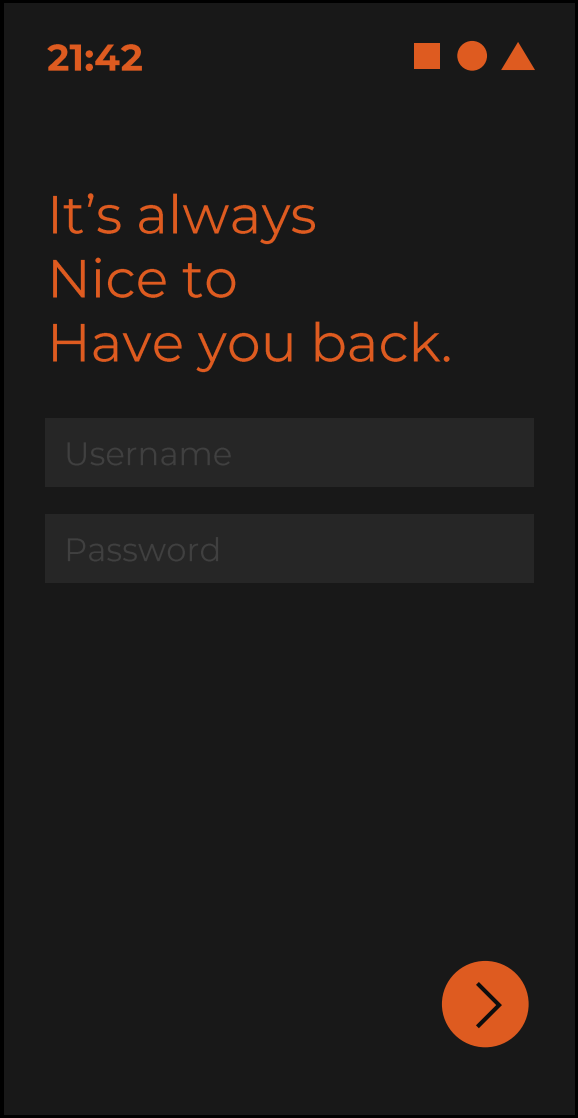
 class UITextField...


<TextInput.../>
React-
Native
<EditText.../>
class UITextField...

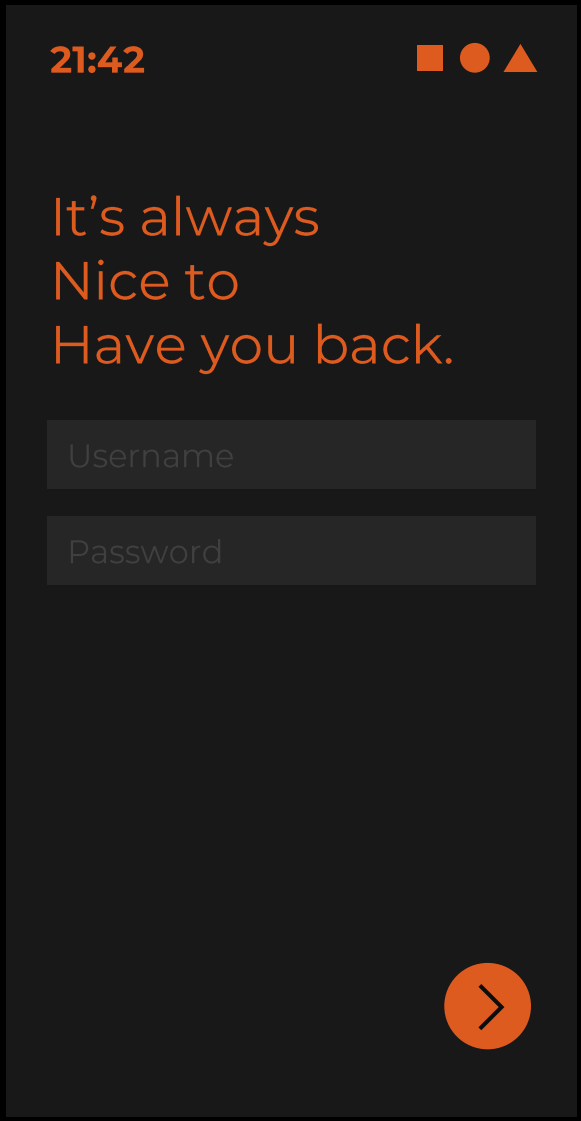
我们似乎可以不再 **Repeat yourself** 了
虽然这可能带来其它问题
但相比代码复用的潜在效益
这一切应该还是值得的

Component-Based Thinking

基于组件的设计思想



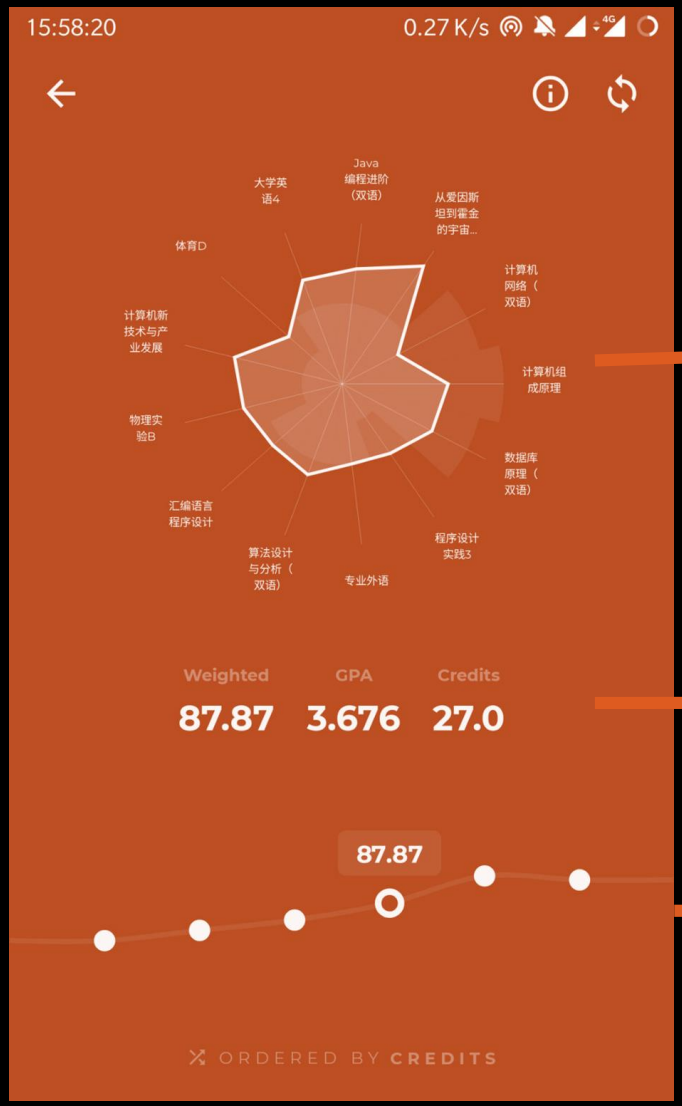

```
Class LoginScreen:
<Screen>
    <StatusBar />
    <Container>
        <Text />
        <TextField />
        <TextField />
        <Button />
    </Container>
</Screen>
```




```
Class LoginScreen:
<Screen>
    <StatusBar />
    <Container>
        <Text preset="h2"/>
        <TextField />
        <TextField />
        <Button onPress="..."/>
    </Container>
</Screen>
```

Props & State

“属性”与“状态”




```
<GpaRadar />
<GpaStats />
  <GpaStat />
  <GpaStat />
  <GpaStat />
<GpaCurve />
```

Props are
set by the parent and
they are fixed **throughout the lifetime**
of a component.

Weighted	GPA	Credits
87.87	3.676	27.0

DEFINE

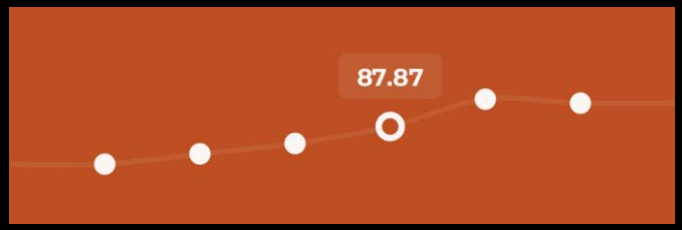
```
class GpaStat extends Component {
  render() {
    return (
      <View>
        <Text preset="small" text={this.props.type} />
        <Text preset="h2" text={this.props.score} />
      </View>
    )
  }
}
```

USE

```
<GpaStat type="weighted" score="87.87"/>
```

请注意，本演示中所有举例使用的 JavaScriptX 代码均为极度简化版本，且和项目本身的 TypeScriptX 实际实现存在差异。尽管使用 TypeScript 实现需要不同的写法，但这些例子仍然能够很好地帮助你理解 State 和 Props 的概念。

States can be
set by the component itself and
they **can be changed**.



DEFINE

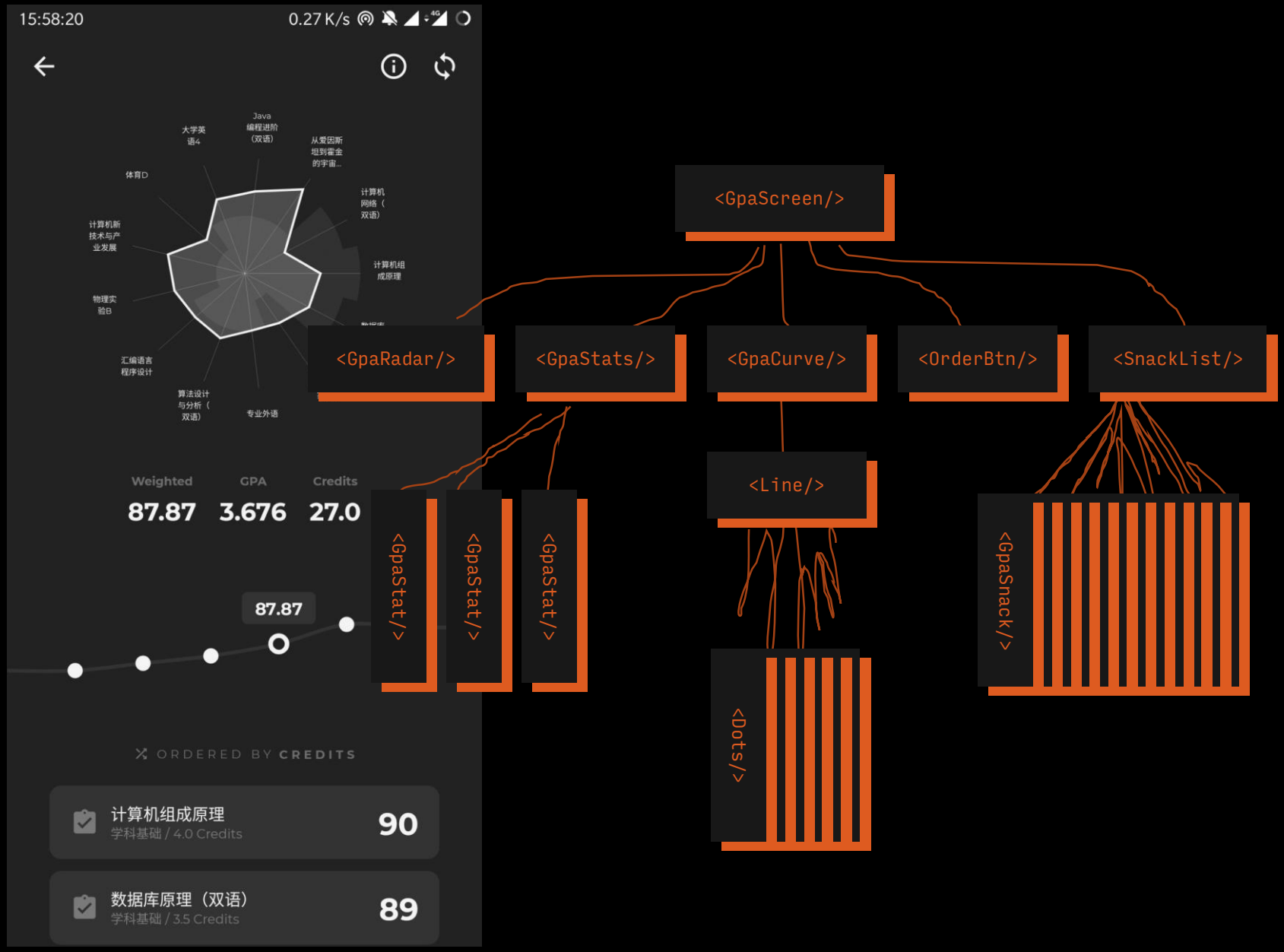
```
class GpaCurve extends Component {
  state = { semesterIndex: 0 }
  render() {
    return (
      <Line>
        this.props.data.map((semester, i) => (
          <Dot
            style={this.state.semesterIndex === i ?
              activeStyle : basicStyle}
            onPress={() => setSemesterIndex(i)}
          />
        ))
      </Line>
    )
  }
}
```

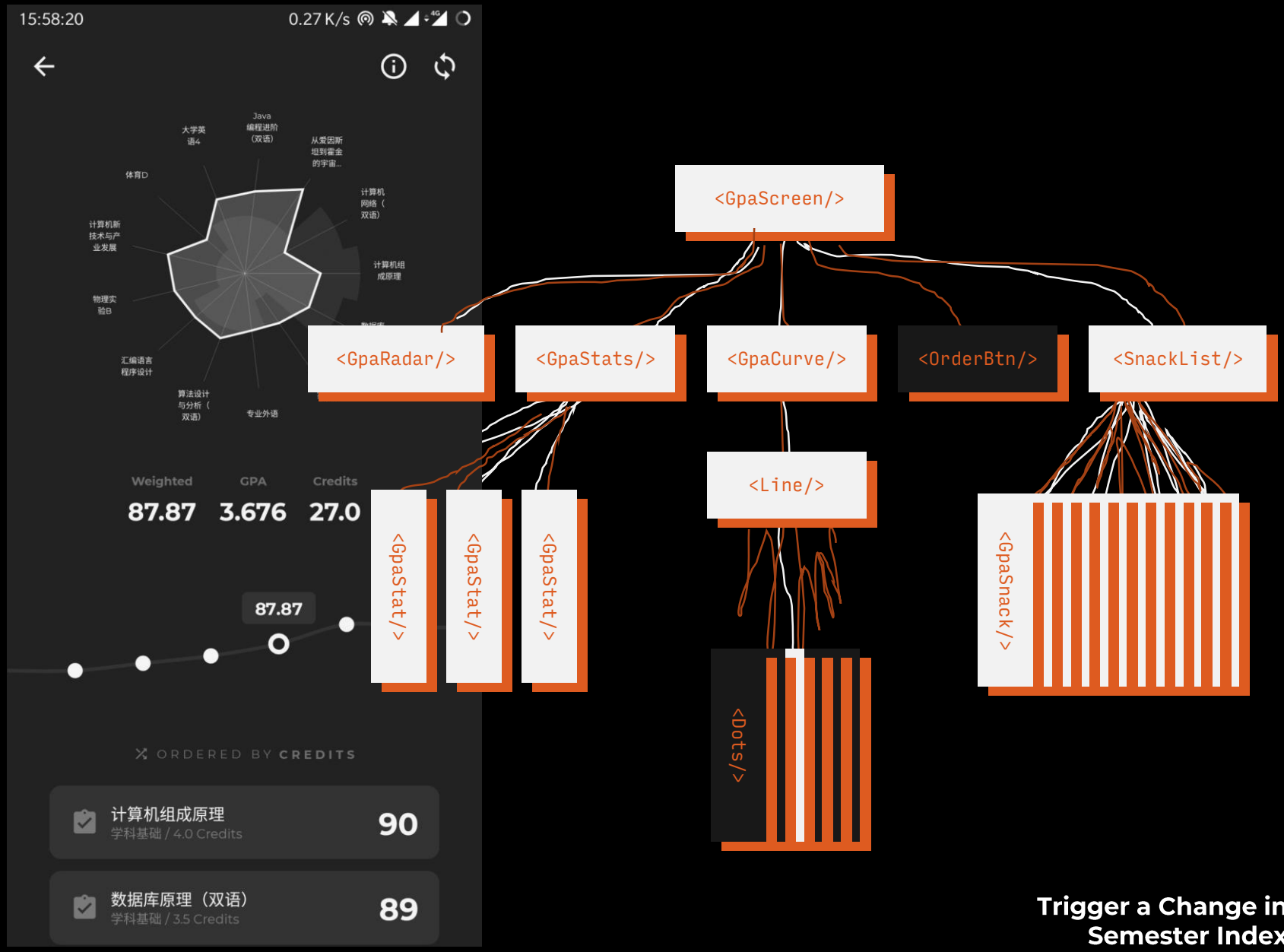
SECTION 2

Flux & Redux

Component Communication For Large Scale Applications

维护大型应用
组件间通信的解决方案

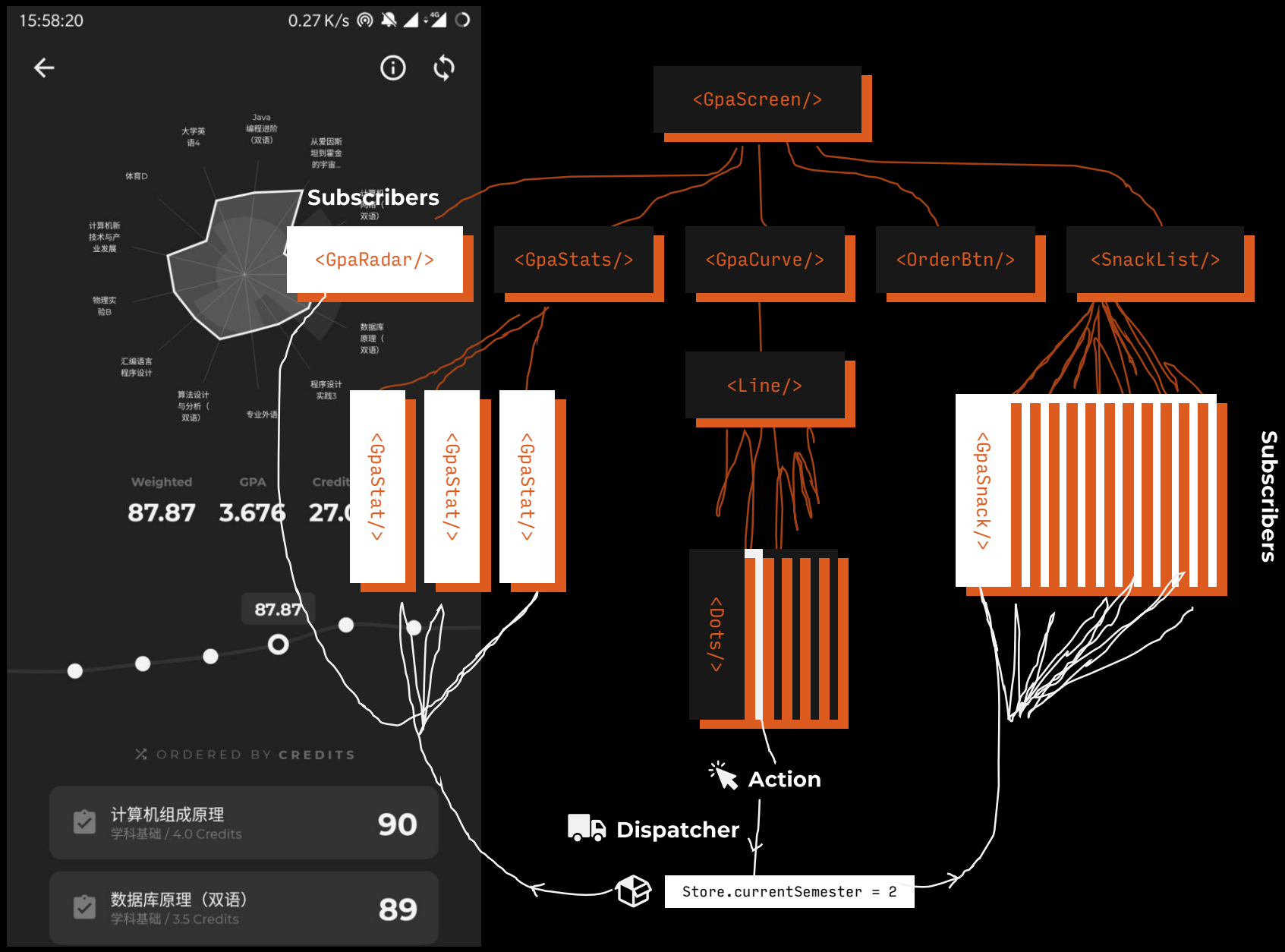

15:58:20
0.27 K/s
Java 编程进阶（双语）
大学英语4
从爱因斯坦到霍金的宇宙...
体育D
计算机网络（双语）
计算机新技术与产业发展
计算机组成原理
物理实验B
汇编语言程序设计
算法设计与分析（双语）
专业外语
Weighted
GPA
Credits
87.87
3.676
27.0
87.87
ORDERED BY CREDITS
计算机组成原理
学科基础 / 4.0 Credits
90
数据库原理（双语）
学科基础 / 3.5 Credits
89
<GpaScreen/>
<GpaRadar/>
<GpaStats/>
<GpaCurve/>
<OrderBtn/>
<SnackList/>
<GpaStat/>
<GpaStat/>
<GpaStat/>
<Line/>
<Dots/>
<GpaSnack/>

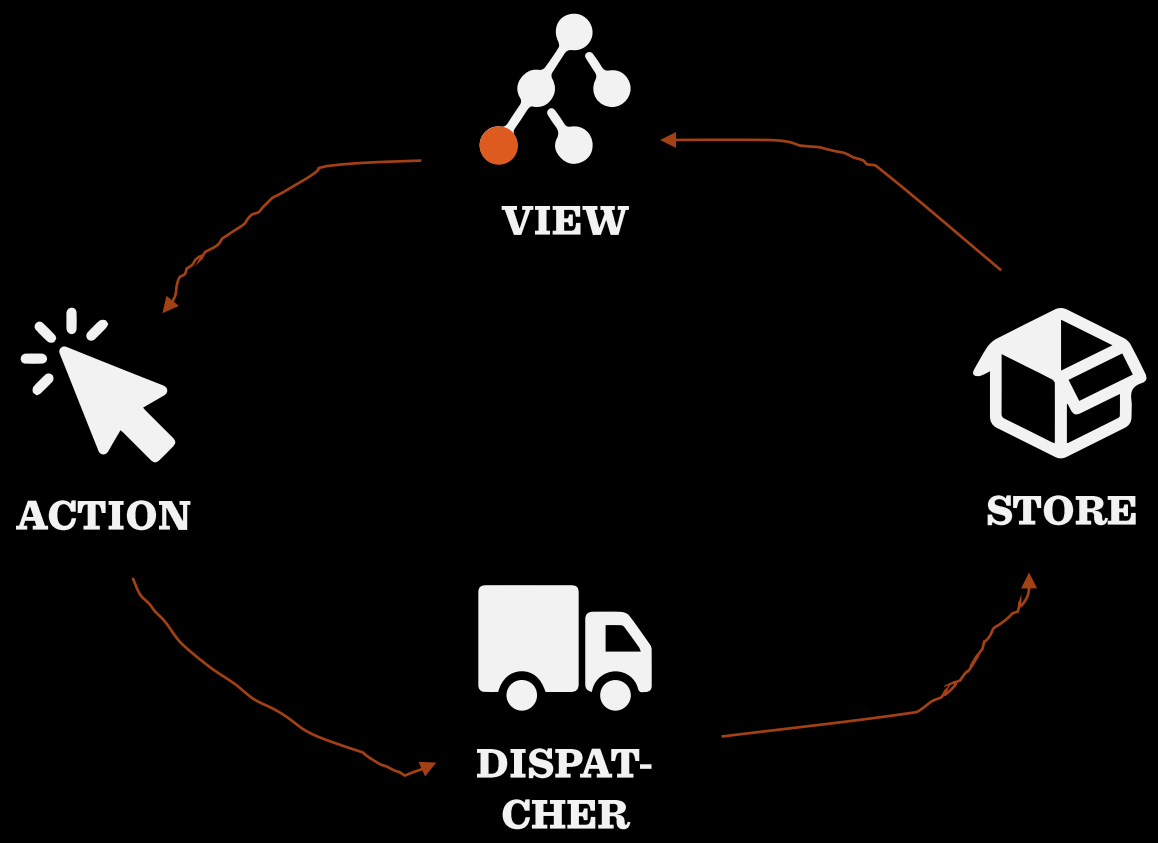


Trigger a Change in Semester Index

大型应用中，
需要管理复杂的组件通信逻辑。
父组件、子组件和兄弟组件
甚至跨越好几个层级的远亲组件。
这之中，除了父向子的单向通信
可以很容易地通过 **props** 完成，
其它的都不太好办。

Flux 设计模式


15:58:20
0.27 K/s
Java 编程进阶（双语）
大学英语4
从爱因斯坦到霍金的宇宙...
体育D
计算机新技术与产业发展
物理实验B
汇编语言程序设计
算法设计与分析（双语）
专业外语
程序设计实践3
数据库原理（双语）
Weighted
GPA
Credit
87.87
3.676
27.
87.87
ORDERED BY CREDITS
计算机组成原理
学科基础 / 4.0 Credits
90
数据库原理（双语）
学科基础 / 3.5 Credits
89
Subscribers
<GpaRadar/>
<GpaScreen/>
<GpaStats/>
<GpaCurve/>
<OrderBtn/>
<SnackList/>
<GpaStat/>
<GpaStat/>
<GpaStat/>
<Line/>
<Dots/>
<GpaSnack/>
Subscribers
Action
Dispatcher
Store.currentSemester = 2


VIEW
STORE
DISPAT-
CHER
ACTION

connect()

在把 Redux 和 React 使用 React-Redux 连接之后，你不能再直接调用 store.dispatch() 等底层函数，而需要通过 React-Redux 提供的 **mapStateToProps** 和 **mapDispatchToProps** 将它们映射到组件的 props 上，再使用 Props 来访问这些功能。直接访问 store 将会产生权限泄露、循环依赖等问题。

SECTION 3

TypeScript

TypeScript =
JavaScript + Type

静态类型编译检查的
优势与妥协

“JavaScript that Scales”

```
function greeter(person) {
    return "Hello, " + person;
}

document.body.textContent = greeter("Eric");
```

```
function greeter(person) {
    return "Hello, " + person;
}

document.body.textContent = greeter("Eric");
```

```
function greeter(person: string) {
    return "Hello, " + person;
}

document.body.textContent = greeter("Eric");
```

```
function greeter(person) {
    return "Hello, " + person;
}

document.body.textContent = greeter("Error");
```

[1,2,3]

Uncertain & potentially destructive behavior

```
function greeter(person: string) {
    return "Hello, " + person;
}

document.body.textContent = greeter("Error");
```

[1,2,3]

Friendly Error message In time

```
error TS2345:
Argument of
type
'number[]'
is not
assignable
to parameter
of type
'string'.
```

JavaScript 的 class 很难用

其实我对 ES6 Class 很无感，一来它不过是个语法糖，二来这个语法糖很坑爹，一些原来能做的事情用纯 class syntax 是做不了的，比如 prototype property 和 static property…… 所以 ES6 版本的 class 其实很废的。另外说句不中听的话，搞不懂原型继承就别给 ES6 class 拍手叫好啦，还是去写 java 吧。

—— **Evan You**

JS 里的 class 功能残缺、使你无法理解 JS 的本质、不能带来额外的好处，如类型安全。

—— 方应杭

```
class Student {
    fullName: string;
    constructor(public firstName: string, public
middleInitial: string, public lastName: string) {
        this.fullName = firstName + " " +
middleInitial + " " + lastName;
    }
}

interface Person {
    firstName: string;
    lastName: string;
}

function greeter(person: Person) {
    return "Hello, " + person.firstName + " " +
person.lastName;
}

let user = new Student("Jane", "M.", "Smith");

document.body.textContent = greeter(user);
```

SECTION 4

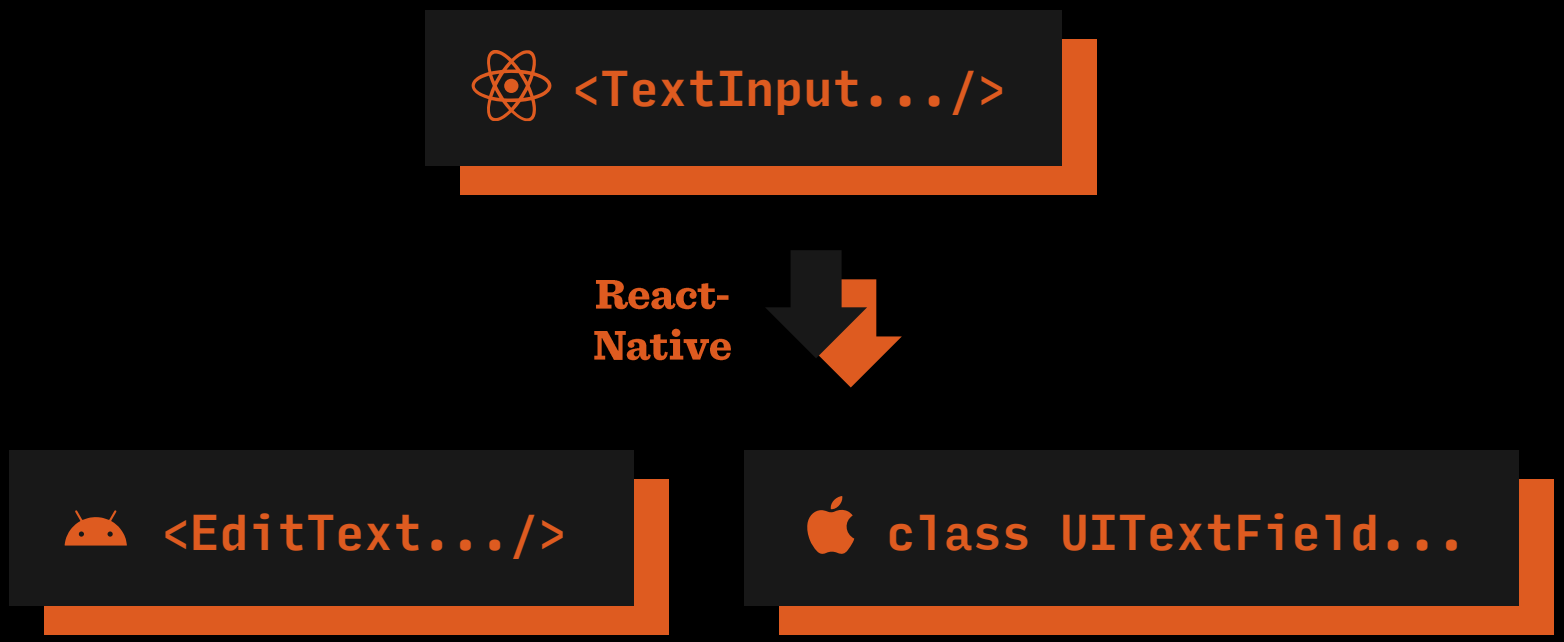
WePeiyang 4.0

几个事实

微北洋 **4.0**

- 使用了 **React-Native**
- 使用了 **Redux** 来管理全局状态
- 使用了 **React-redux** 来连接 **Redux** 状态和 **React** 组件
- 使用了 **TypeScript** 作为开发语言，但并没有 **Strictly-typed**

```
WePeiyang-RN
├── android
├── ios
├── app
│   ├── i18n
│   ├── store.ts
│   ├── actions
│   ├── reducers
│   ├── components
│   ├── navigation
│   ├── screens
│   ├── services
│   ├── theme
│   ├── utils
│   ├── app.tsx
│   ├── environment-variables.ts
├── __test__
├── README.md
├── index.js
└── package.json
```


<TextInput.../>
React-
Native
<EditText.../>
class UITextField...

app

<TextInput.../>

android

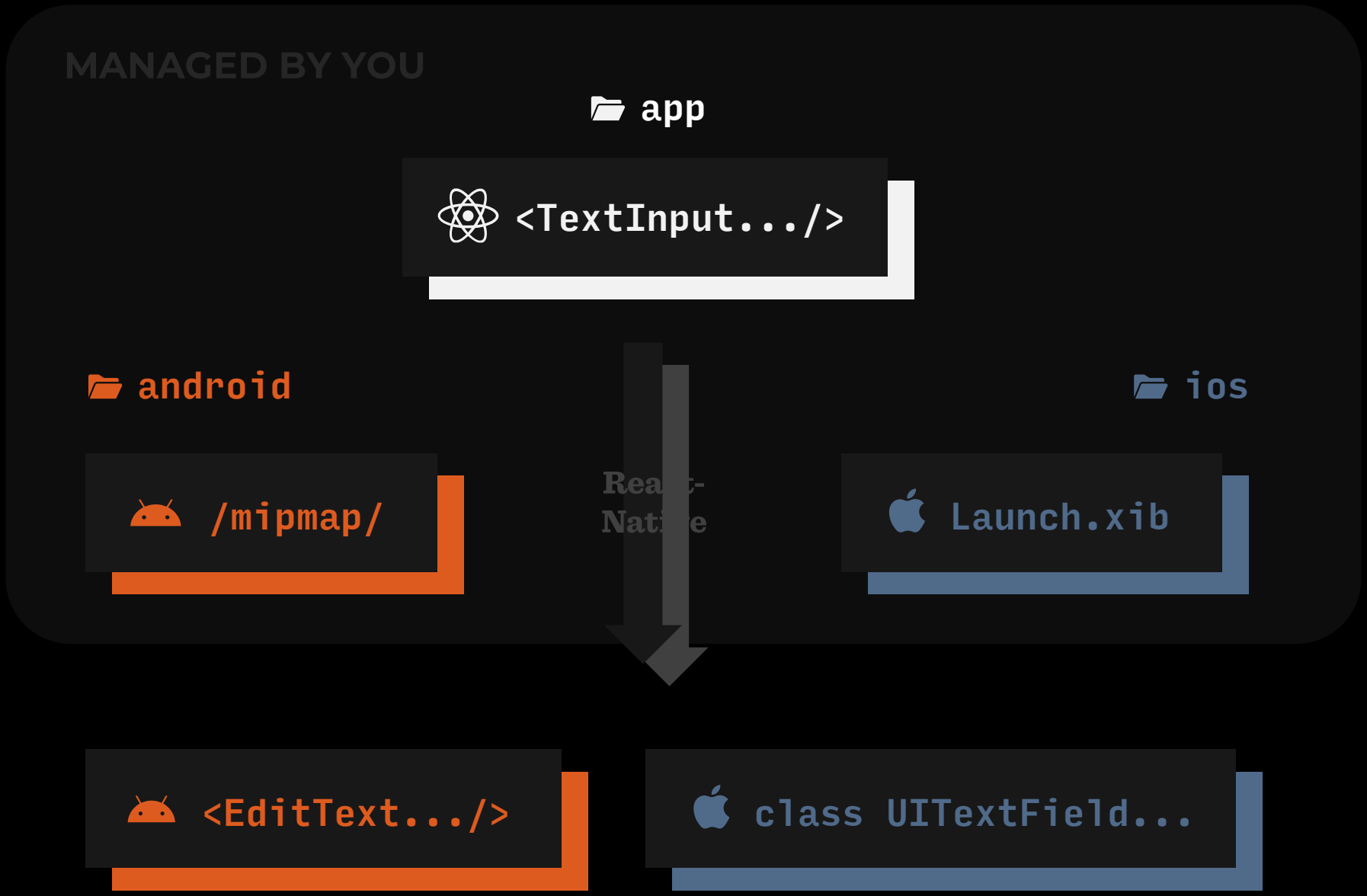
/mipmap/

ios

Launch.xib

<EditText.../>

class UITextField...


MANAGED BY YOU
app
<TextInput.../>
android
ios
/mipmap/
React-Native
Launch.xib
<EditText.../>
class UITextField...

Location: app/components/ian/ian.tsx

The Most Simple Component: <Ian/>

```
/*
 * Ian
 * Created by Tzingtao Chow
 * ---
 *
 * Iana (Ian 的复数形式) 是用于填充空 List 的提示性组件。
 * 如主页的「你还没有借阅书籍」, Schedule Screen 的「今天无课」等。
 * 样式上, 它表现为淡灰色的圆角矩形。
 *
 */

import * as React from "react"
import { TextStyle, View, ViewStyle } from "react-native"
import { Text } from "../text"
import { color, layoutParam } from "../../theme"

export interface IanProps {
  tx?: string
  text?: string
  style?: ViewStyle
  palette?
}

export function Ian(props: IanProps) {
  const { tx, text, style, palette } = props
  let colors = palette || [color.washed, color.lightGrey]

  const ianView: ViewStyle = {
    backgroundColor: colors[0],
    borderRadius: layoutParam.borderRadius,
    alignItems: "center",
    justifyContent: "center",
    padding: 20,
  }

  const ianText: TextStyle = {
    color: colors[1],
    fontWeight: "bold",
    textTransform: "uppercase",
  }

  return (
    <View style={[ianView, style]}>
      <Text tx={tx} text={text} style={ianText} />
    </View>
  )
}
```

Location: app/components/ian/ian.tsx

The Most Simple Component: <Ian/>


```
if (courseDaily.length <= 0) {
    return (
      <View style={[ss.predefinedStyle, style]}>
        <Ian tx="schedule.noCourseToday" />
      </View>
    )
  }
```

Location:
app/components/ecard-block/ecard-block.tsx

A Connected Component: <Ecard/>

```
import...

export interface EcardBlockProps {
  style?: ViewStyle
  ecard?
  palette?
  onPress?
}

class _EcardBlock extends React.PureComponent<EcardBlockProps, {}> {
  render() {
    let { ecard, style, palette, onPress } = this.props

    return (
        <View style={ss.containerStyle} pointerEvents="box-only">
          <View style={ss.top}>
            <View style={ss.bar}>
              <Text>
                <Text tx="ecard.card" style={ss.barTextPre} />
                <Text text={" NO." + ecard.profile.cardnum}
style={ss.barTextSub} />
              </Text>
            </View>
            <Text>
              <Text text="¥" style={ss.yen} />
              <Text text={ecard.profile.balance} style={ss.balance} />
            </Text>
          </View>
          <View style={ss.bottom}>...</View>
        </View>
    )
  }
}

const mapStateToProps = state => {
  return {
    ecard: state.dataReducer.ecard,
  }
}
const mapDispatchToProps = () => {
  return {}
}

export const EcardBlock = connect(
  mapStateToProps,
  mapDispatchToProps,
)(_EcardBlock)
```

Location:
app/components/text/text.tsx

A Hacked-over Component: <Text/>

```
// Using plain text
// balance = "78.50"
<Text text={balance} />

// Using i18n text
// Edit translation resources in app/i18n/xx.json
<Text tx="ecard.card" />

// Using preset
<Text text="Ordered by score" preset="lausanne" />

// Controlling autospacing between letters & ideographs
// raw = "我的名字是Spencer Hendricks"
<Text text={raw} spacing={false} />

// Using custom style override
<Text text="Fancy Text" style={someCustomStyleObj} />
```

REFERENCES

WePeiyang 4.0

https://github.com/Cyphexl/WePeiyang-RN

Instructions - 如果你觉得本仓库很赞，点亮 Star
如果你发现了一个问题且想让开发团队修复它，发起 Issues
如果你想衍生自己的版本，或是对仓库做自己的修改，Fork 它
如果你发现了一个问题，想自己修复它，Fork 它并发起 Pull Request

React-Native

https://facebook.github.io/react-native/docs/getting-started
按照指示，自己 Build App 并运行

For Anything Else

https://www.google.com/
搜索 “Redux” “TypeScript”，因为官方的 Getting Started 总是靠谱

LOVE & PEACE