React Native

yongqiang

自我介绍

丁勇强

yongqiang

Mac开发组

富途

课程简介

1、介绍RN的能力，能用来做什么

2、简要分析RN运行的原理

3、介绍RN中的基础概念，动手编写

4、概括下RN优缺点，介绍未来发展方向

目录

React Native 简介

RN简介

1、15年3月由Facebook正式开源

2、基于React web框架

3、使用JS写原生应用

4、Learn once, write anywhere

RN简介 - 案例

京东金融

金融生活服务平台

iOS - Android

漂漂共享书

与朋友同事分享纸质图书

iOS - Android

若爱

免费实名制相亲婚恋平台

iOS - Android

QQ空间

分享生活，留住感动

iOS - Android

一键康

您的公立医院体检平台

iOS - Android

找车场

简单方便的帮你找车场

iOS - Android

幕布

管理你的大脑

iOS - Android

宝点理财

互联网金融理财产品

iOS - Android

RN简介 - 案例


京东金融
金融生活服务平台
iOS - Android
漂漂共享书
与朋友同事分享纸质图书
iOS - Android
若爱
免费实名制相亲婚恋平台
iOS - Android
QQ空间
分享生活，留住感动
iOS - Android
一键康
您的公立医院体检平台
iOS - Android
找车场
简单方便的帮你找车场
iOS - Android
幕布
管理你的大脑
iOS - Android
宝点理财
互联网金融理财产品
iOS - Android

RN简介 - 生态

- Repositories 63K
- Code
- Commits 207K
- Issues 139K
- Topics 317
- Wikis 4K
- Users 2K

Languages

JavaScript	40,191
Objective-C	10,066
Java	3,373
Makefile	648
TypeScript	604
HTML	363
Python	218
Swift	163
Shell	162

React Native

React Native is a JavaScript mobile framework developed by Facebook.

See topic

63,599 repository results

Sort: **Best match**

facebook/*react-native*

A framework for building *native* apps with *React*.

JavaScript 65K ★ 65k

Updated an hour ago 47 issues need help

reactnativecn/*react-native*-guide

★ 12.3k

*React Native*指南汇集了各类*react-native*学习资源、开源App和组件

Updated 13 days ago

jondot/awesome-*react-native*

Ruby ★ 19.8k

Awesome *React Native* components, news, tools, and learning material!

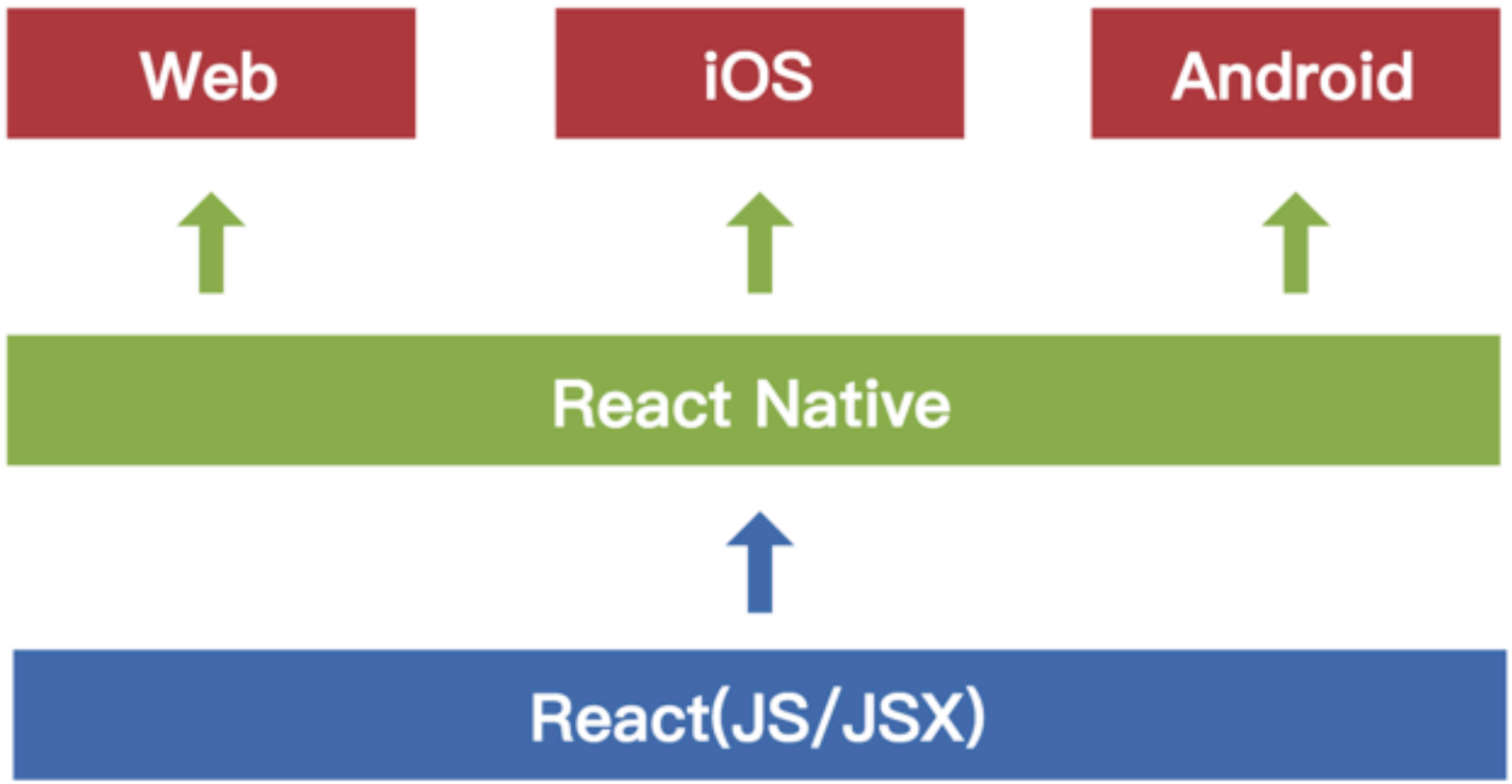
RN简介

纯RN App

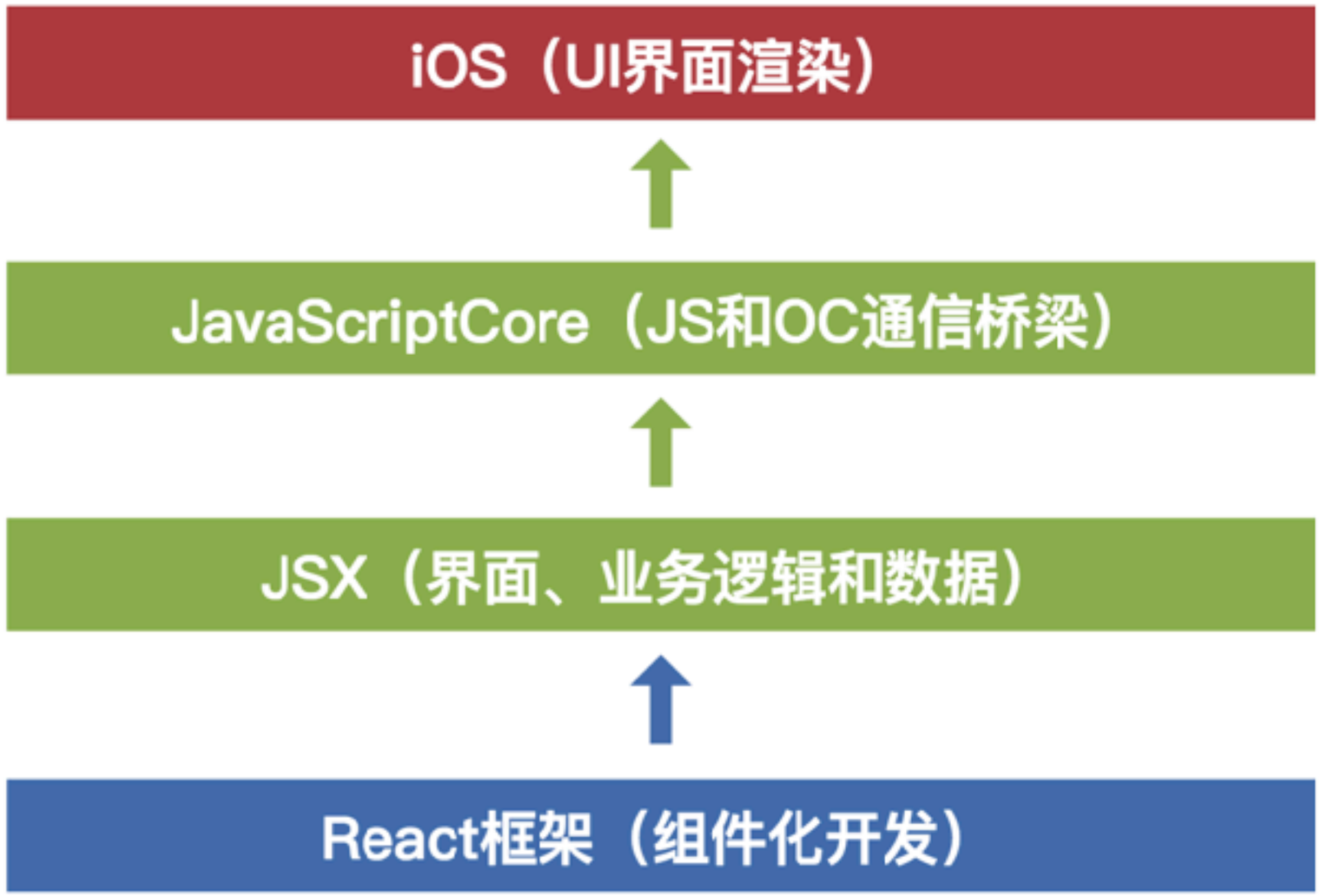
成熟App嵌入RN页面

React Native 原理篇

RN原理



RN原理 - iOS渲染流程



RN原理 - iOS渲染流程

如何将JS代码转换为OC可执行代码？？？

RN原理 - 通过JSCore模拟JS和OC的交互

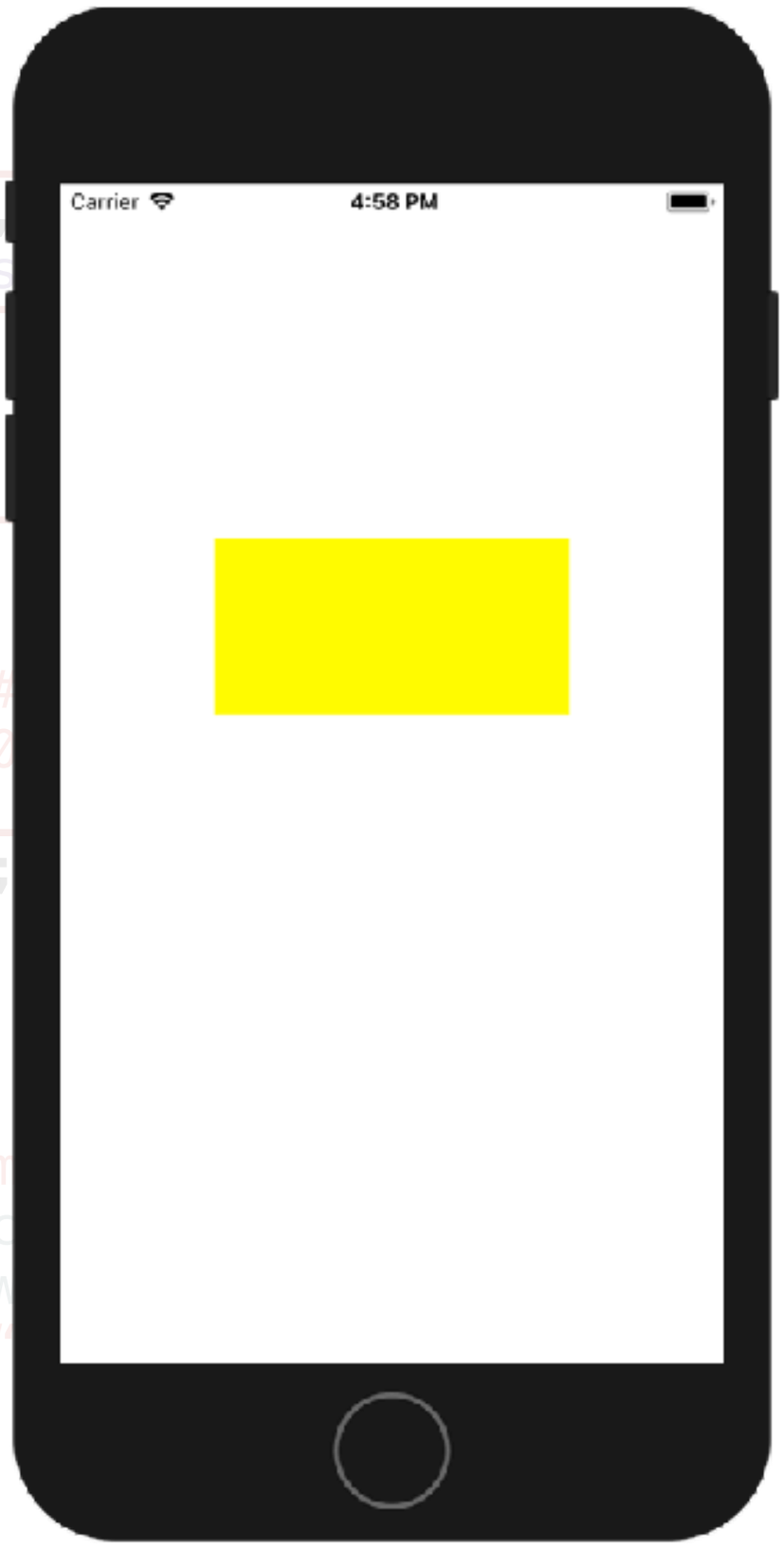
```
- (void)viewDidLoad {
    [super viewDidLoad];
    _moduleConfig = @{@"View": [UIView class],
                      @"Image": [UIImage class]};

    [self executeJSCode];
}

- (void)executeJSCode {
    NSString *jsCode = @"var renderData = [
     {'module':'View',
     'style':{'bgColor': '#FFFF00'},
     ‘rect’:{‘x’:75,'y':300,'w':200,'h':100}},
     ]";
    JSContext *ctx = [[JSContext alloc] init];
    [ctx evaluateScript:jsCode];
    NSArray *configInfo = [ctx[@"renderData"] toArray];
    for (NSDictionary *dic in configInfo) {
        NSDictionary *rect = dic[@"rect"];
        UIView *view = [[_moduleConfig[dic[@"module"]] alloc]
                        initWithFrame:[self convertRectDic:rect]];
        view.backgroundColor = [UIColor colorWithHexString:
                                        dic[@“style"][@"bgColor"]];
        [self.view addSubview:view];
    }
}
```

注册表

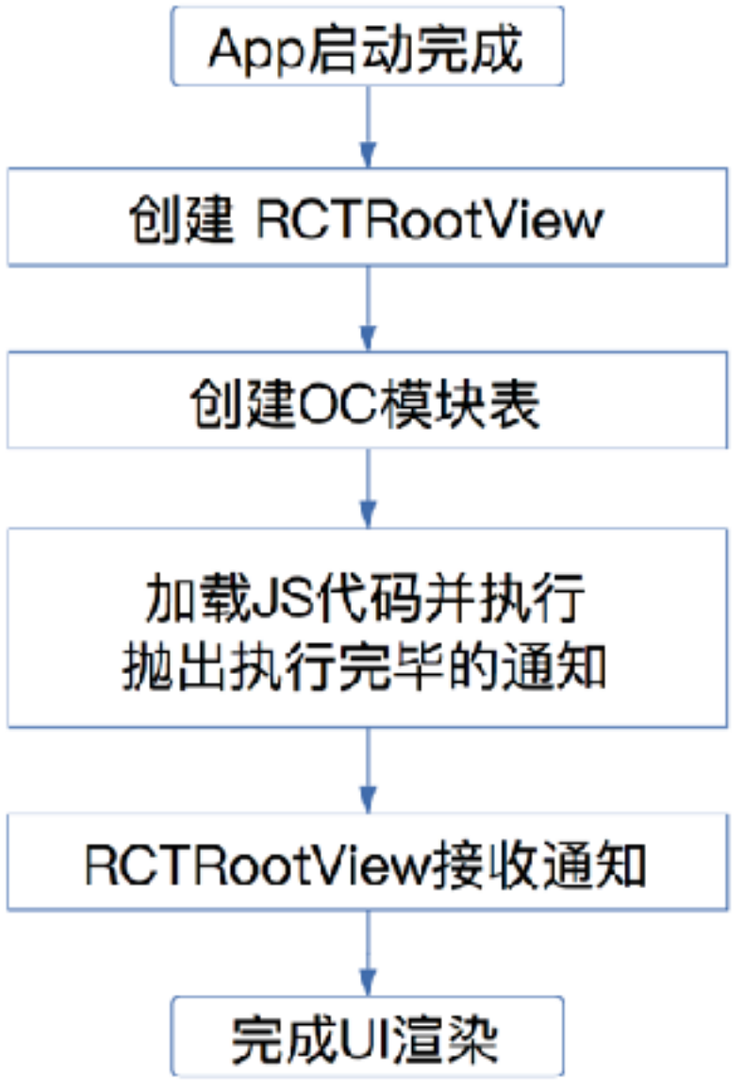
JS代码

RN原理 - 通过JSCore模拟JS和OC的交互

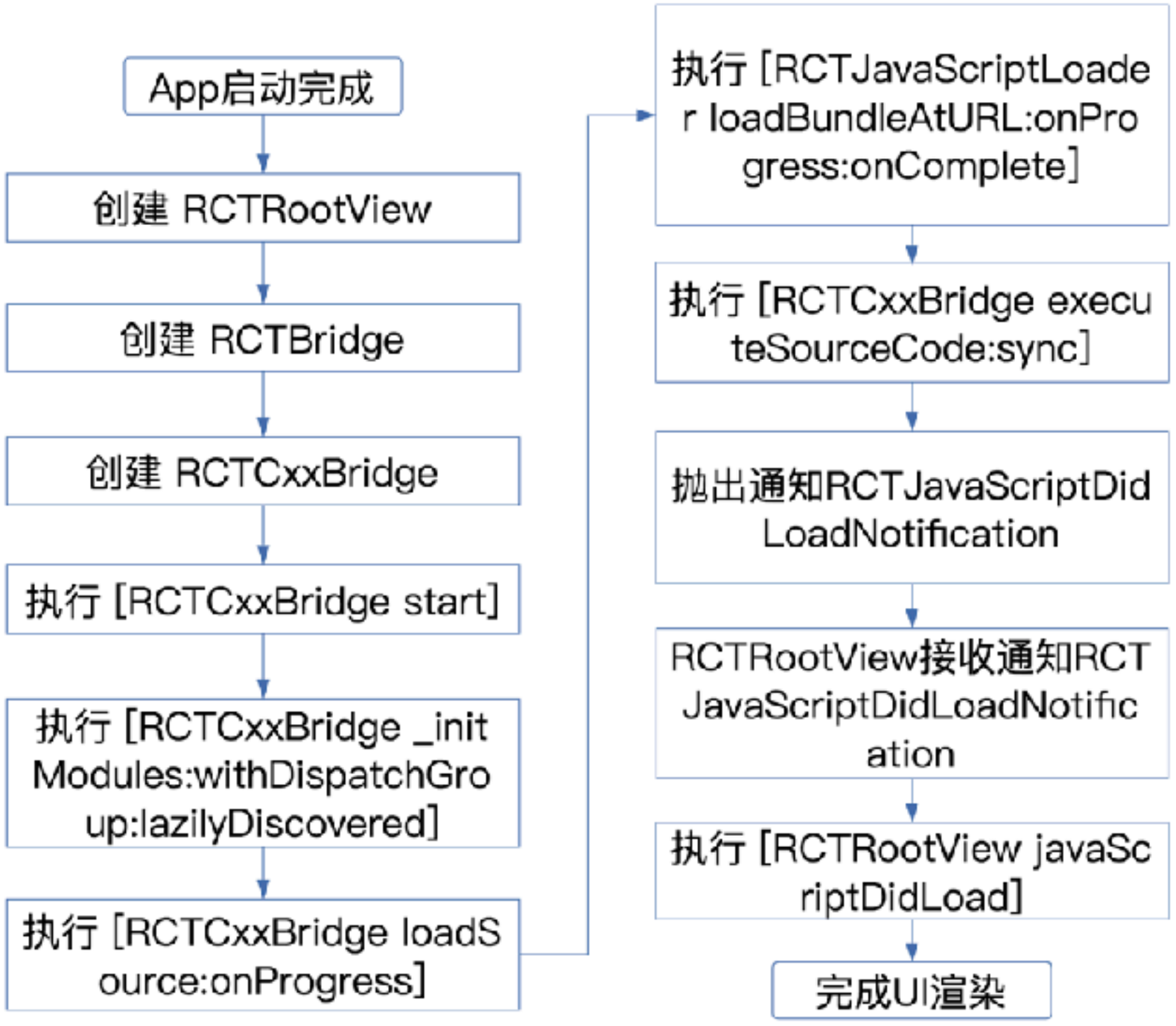
```
var renderData = [
    {'module':'View',
     'style':{'bgColor': '#FFFF00'},
     'rect':{'x':87.5,'y':200,
             'w':200,'h':100},
    },
    . . .
]
```



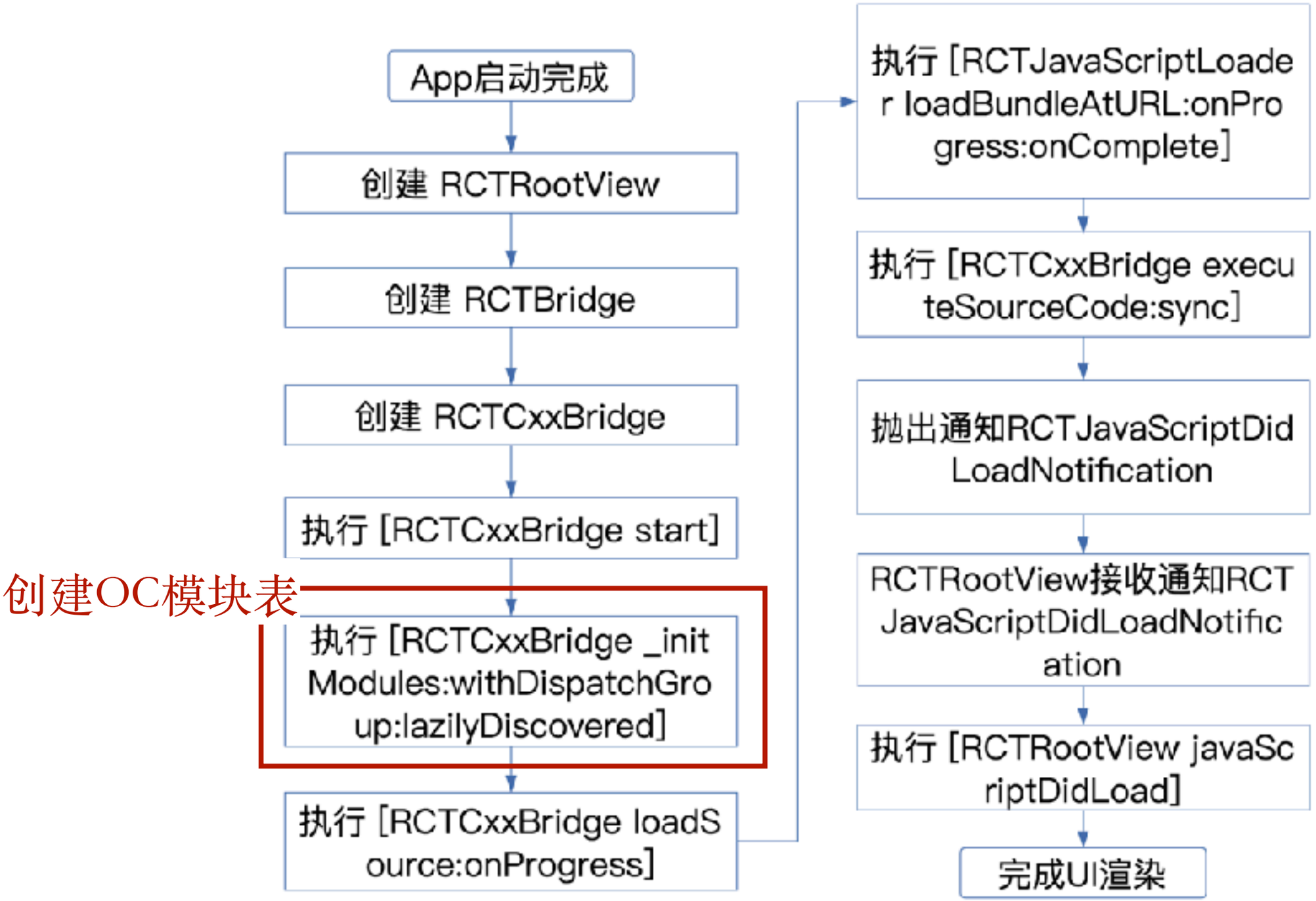
RN原理 - 渲染流程


App启动完成
创建 RCTRootView
创建OC模块表
加载JS代码并执行
抛出执行完毕的通知
RCTRootView接收通知
完成UI渲染

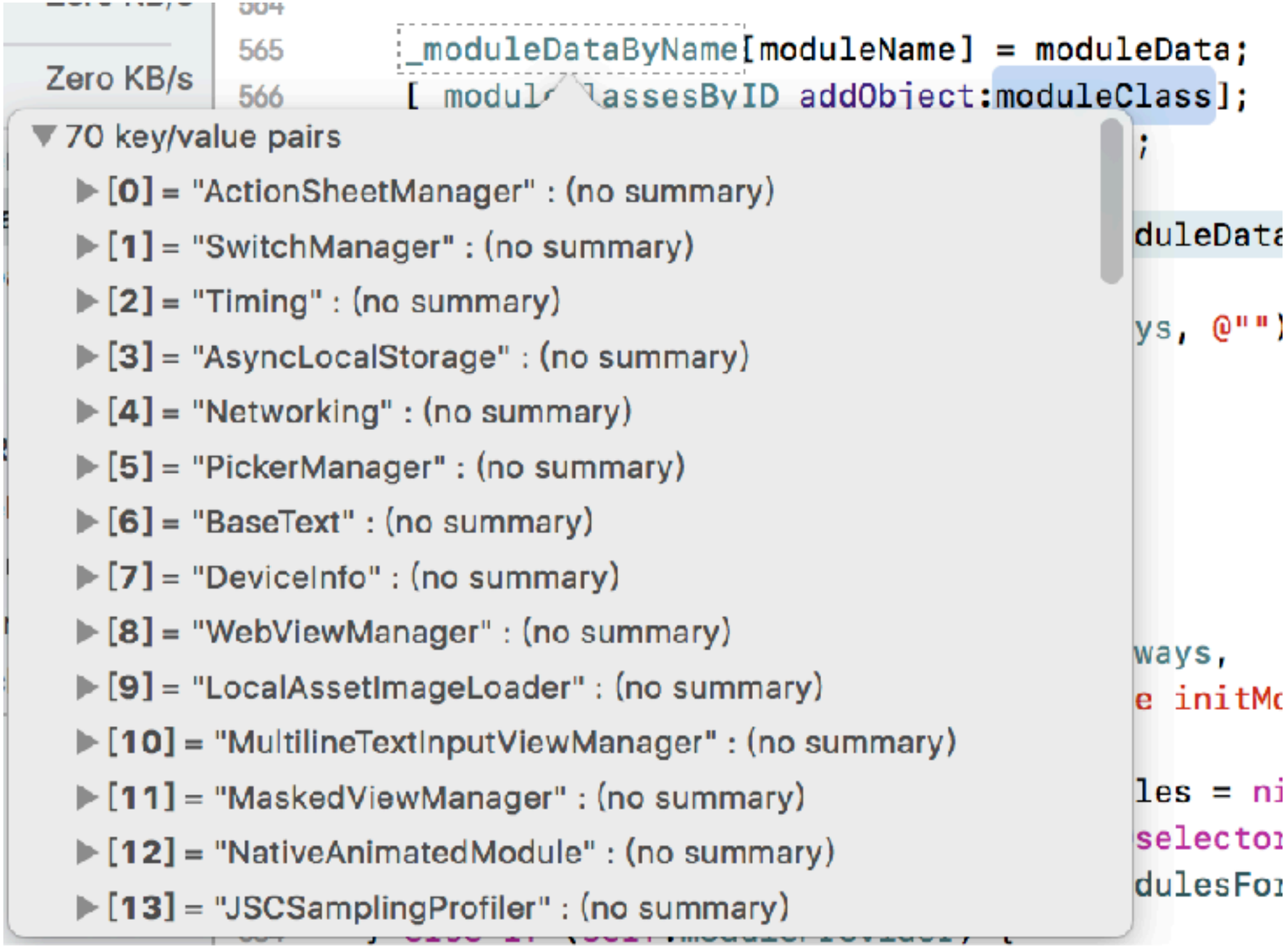
RN原理 - 渲染流程


App启动完成
创建 RCTRootView
创建 RCTBridge
创建 RCTCxxBridge
执行 [RCTCxxBridge start]
执行 [RCTCxxBridge _initModules:withDispatchGroup:lazilyDiscovered]
执行 [RCTCxxBridge loadSource:onProgress]
执行 [RCTJavaScriptLoader loadBundleAtURL:onProgress:onComplete]
执行 [RCTCxxBridge executeSourceCode:sync]
抛出通知RCTJavaScriptDidLoadNotification
RCTRootView接收通知RCTJavaScriptDidLoadNotification
执行 [RCTRootView javaScriptDidLoad]
完成UI渲染

RN原理 - 渲染流程

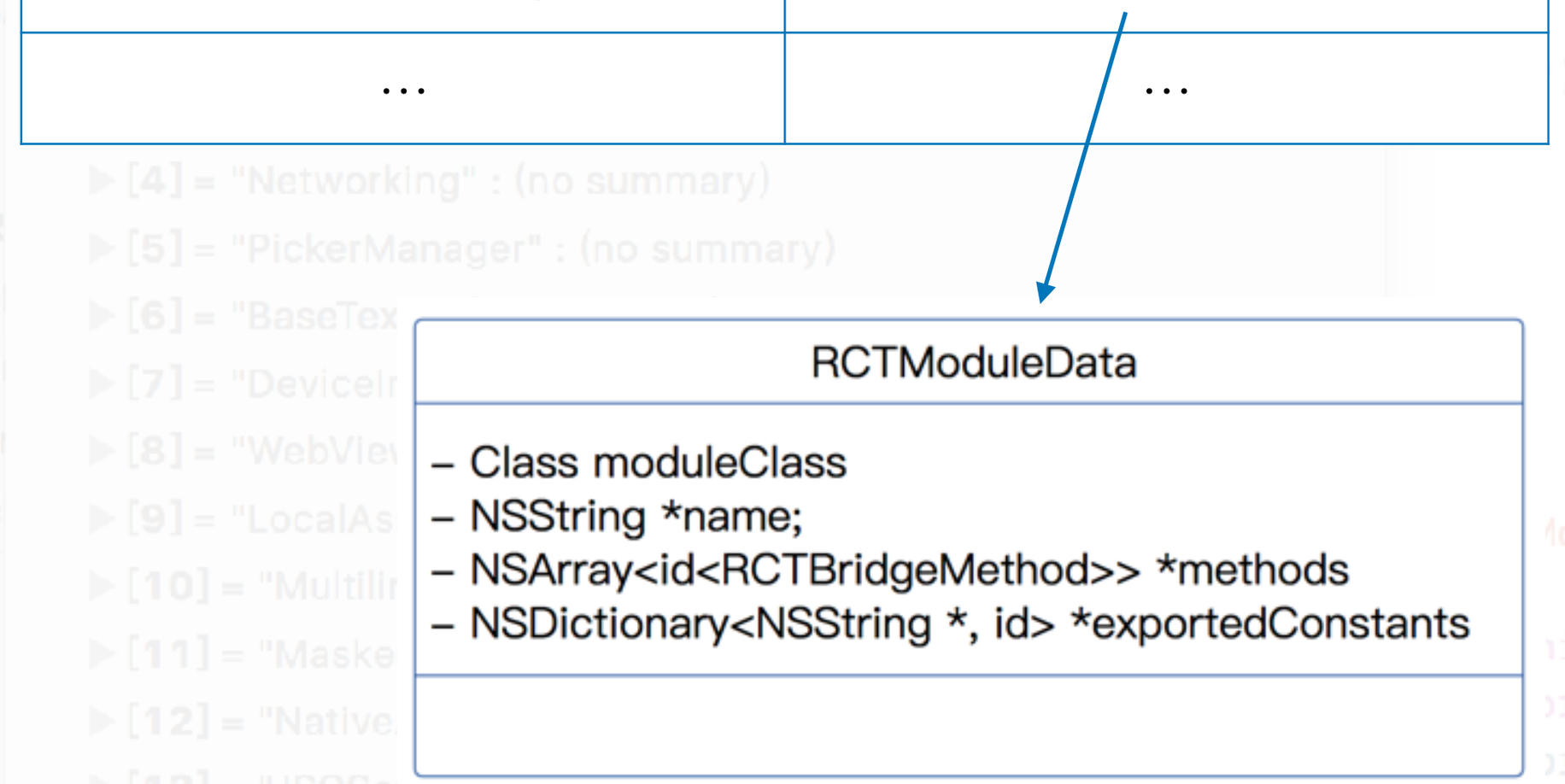

App启动完成
创建 RCTRootView
创建 RCTBridge
创建 RCTCxxBridge
执行 [RCTCxxBridge start]
创建OC模块表
执行 [RCTCxxBridge _initModules:withDispatchGroup:lazilyDiscovered]
执行 [RCTCxxBridge loadSource:onProgress]
执行 [RCTJavaScriptLoader loadBundleAtURL:onProgress:onComplete]
执行 [RCTCxxBridge executeSourceCode:sync]
抛出通知RCTJavaScriptDidLoadNotification
RCTRootView接收通知RCTJavaScriptDidLoadNotification
执行 [RCTRootView javaScriptDidLoad]
完成UI渲染

RN原理 - OC模块表



RN原理 - OC模块表

Key	Value
ActionSheetManager	RCTModuleData *
WebviewManager	RCTModuleData *
…	…

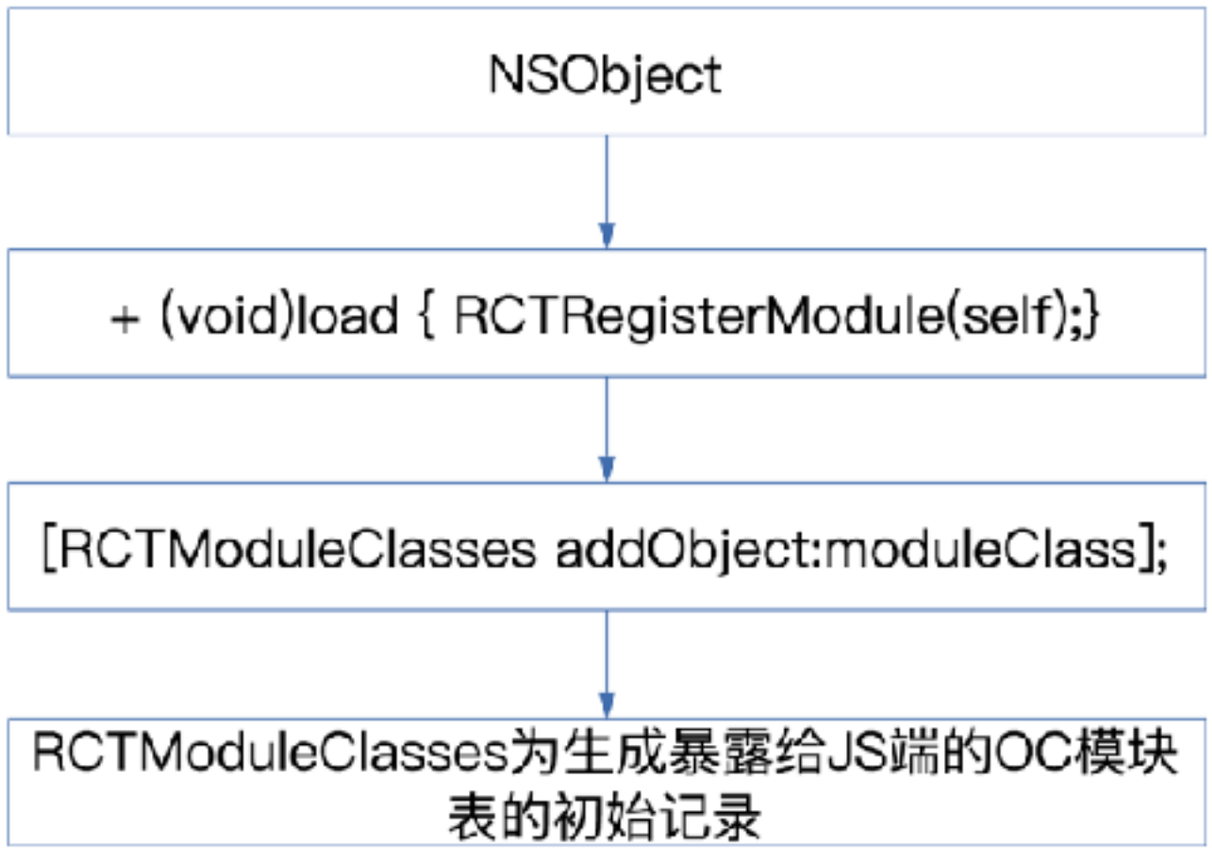


RN原理 - OC模块表

这张模块表OC是怎样生成的?

OC怎么知道这些类可以和JS交互

RN原理 - OC模块表



RN原理 - OC模块表

```
#define RCT_EXPORT_MODULE(js_name) \
RCT_EXTERN void RCTRegisterModule(Class); \
+ (NSString *)moduleName { return @#js_name; } \
+ (void)load { RCTRegisterModule(self); }
```

```
void RCTRegisterModule(Class moduleClass)
{
  static dispatch_once_t onceToken;
  dispatch_once(&onceToken, ^{
    RCTModuleClasses = [NSMutableArray new];
  });

  RCTAssert([moduleClass conformsToProtocol:@protocol(RCTBridgeModule)],
            @"%@ does not conform to the RCTBridgeModule protocol",
            moduleClass);

  // Register module
  [RCTModuleClasses addObject:moduleClass];
}
```

如果我们需要将自定义的类暴露给js调用，该如何做?

RN原理

如果我们需要将自定义的类暴露给js调用，该如何做?

```
@interface FTReactNativeHandler()<RCTBridgeModule>

@end

@implementation FTReactNativeHandler

RCT_EXPORT_MODULE()

RCT_EXPORT_METHOD(handleURLString:(NSString *)string) {

}
…
@end
```

RN原理

```
import {NativeModules} from 'react-native';

const rnHandler = NativeModules.FTReactNativeHandler

rnHandler handleURLString('ftnn://url/');
```

React Native 基础入门

RN新建项目

react-native init AwesomeProject

cd AwesomeProject

react-native run-ios

RN项目结构

AwesomeProject 〉 App.js 〉

Project

- AwesomeProject ~/Desktop/AwesomeProject
  - android
    - app
    - gradle
    - keystores
    - build.gradle
    - gradle.properties
    - gradlew
    - gradlew.bat
    - settings.gradle
  - ios
    - AwesomeProject
    - AwesomeProject.xcodeproj
    - AwesomeProject-tvOS
    - AwesomeProject-tvOSTests
    - AwesomeProjectTests
    - build
  - node_modules library root
  - .babelrc
  - .buckconfig
  - .flowconfig
  - .gitattributes
  - .gitignore
  - .watchmanconfig
  - App.js
  - app.json
  - index.js
  - package.json
  - yarn.lock
- External Libraries

App.js

```
1  /**
2   * Sample React Native App
3   * https://github.com/facebook/react-native
4   * @flow
5   */
6
7  import ...
14
15 const instructions = Platform.select({
16   ios: 'Press Cmd+R to reload,\n' +
17     'Cmd+D or shake for dev menu',
18   android: 'Double tap R on your keyboard to reload,\n' +
19     'Shake or press menu button for dev menu',
20 });
21
22 type Props = {};
23 export default class App extends Component<Props> {
24   render() {
25     return (
26       <View style={styles.container}>
27       <Text style={styles.welcome}>
28                     Welcome to React Native!
29                 </Text>
30                 <Text style={styles.instructions}>
31                     To get started, edit App.js
32                 </Text>
33                 <Text style={styles.instructions}>
34                     {instructions}
35                 </Text>
36
37       </View>
38     );
39   }
40 }
41
42 const styles = StyleSheet.create({
43   container: {
44     flex: 1,
```

RN - JSX

```
export default class App extends Component {
    render() {
        return (
            <View>
                <Text>
                    Welcome to React Native!
                </Text>
            </View>
        );
    }
}
```

RN基础控件

```
export default class App extends Component {
    render() {
        return (
            <View>
                <Text>
                    Welcome to React Native!
                </Text>
            </View>
        );
    }
}
```

RN基础控件

```
export default class App extends Component {
    render() {
        return (
            <View>
                <Text>
                    Welcome to React Native!
                </Text>
            </View>
        );
    }
}
```

Button
CheckBox
Image
Switch
……

Slider
ScrollView
FlatList
TouchableHighlight

RN自定义组件

```
class BriefView extends Component {
    static defaultProps = {
        description: '',
    };
    static propTypes = {
        description: PropTypes.string.isRequired,
    };
    render() {
        return (
            <Text>{this.props.description}</Text>
        )
    }
}

export default class App extends Component {
    render() {
        return (
            <View>
                <BriefView description={'Welcome to React Native!'}/>
            </View>
        );
    }
}
```

RN自定义组件 - props

```
class BriefView extends Component {
    static defaultProps = {
        description: '',
    };
    static propTypes = {
        description: PropTypes.string.isRequired,
    };
    render() {
        return (
            <Text>{this.props.description}</Text>
        )
    }
}

export default class App extends Component {
    render() {
        return (
            <View>
                <BriefView description={'Welcome to React Native!'}/>
            </View>
        );
    }
}
```

RN自定义组件 - props

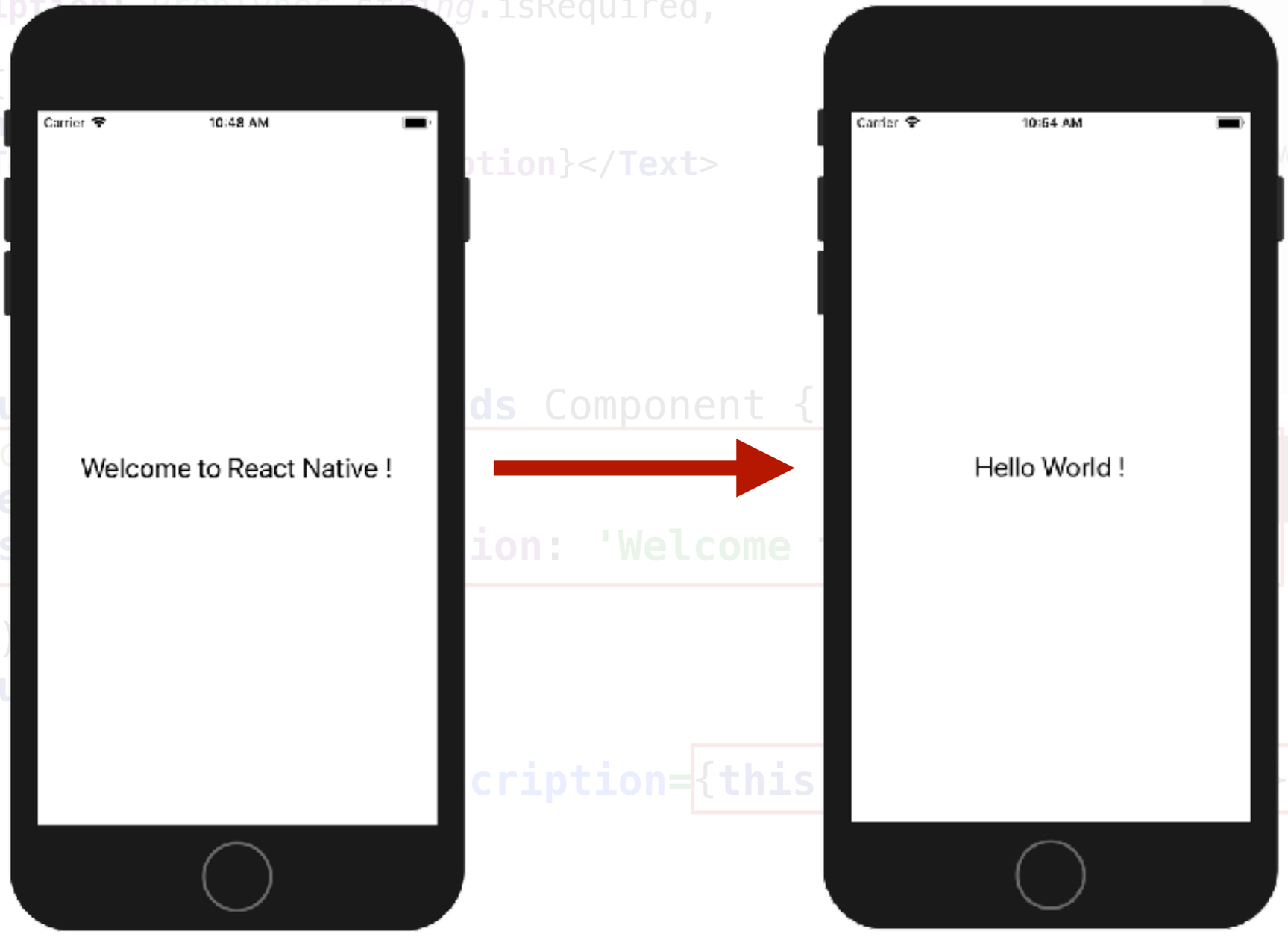
```
class BriefView extends Component {
    static defaultProps = {
        description: '',
    };
    static propTypes = {
        description: PropTypes.string.isRequired,
    };
    render() {
        this.props.description = 'Try to Modify';
        return (
            <Text>{this.props.description}</Text>
        )
    }
}

export default class App extends Component {
    render() {
        return (
            <View>
                <BriefView description={'Welcome to React Native!'}/>
            </View>
        );
    }
}
```

RN自定义组件 - state

```
class BriefView extends Component {
    static defaultProps = {
        description: '',
    };
    static propTypes = {
        description: PropTypes.string.isRequired,
    };
    render() {
        return (
            <Text>{this.props.description}</Text>
        )
    }
}

export default class App extends Component {
    constructor(props) {
        super(props);
        this.state = {description: 'Welcome to React Native!'}
    }
    render() {
        return (
            <View>
                <BriefView description={this.state.description}/>
            </View>
        );
    }
}
```

RN自定义组件 - state

```
this.setState({
    description: 'Hello World !'
})
```



RN组件生命周期

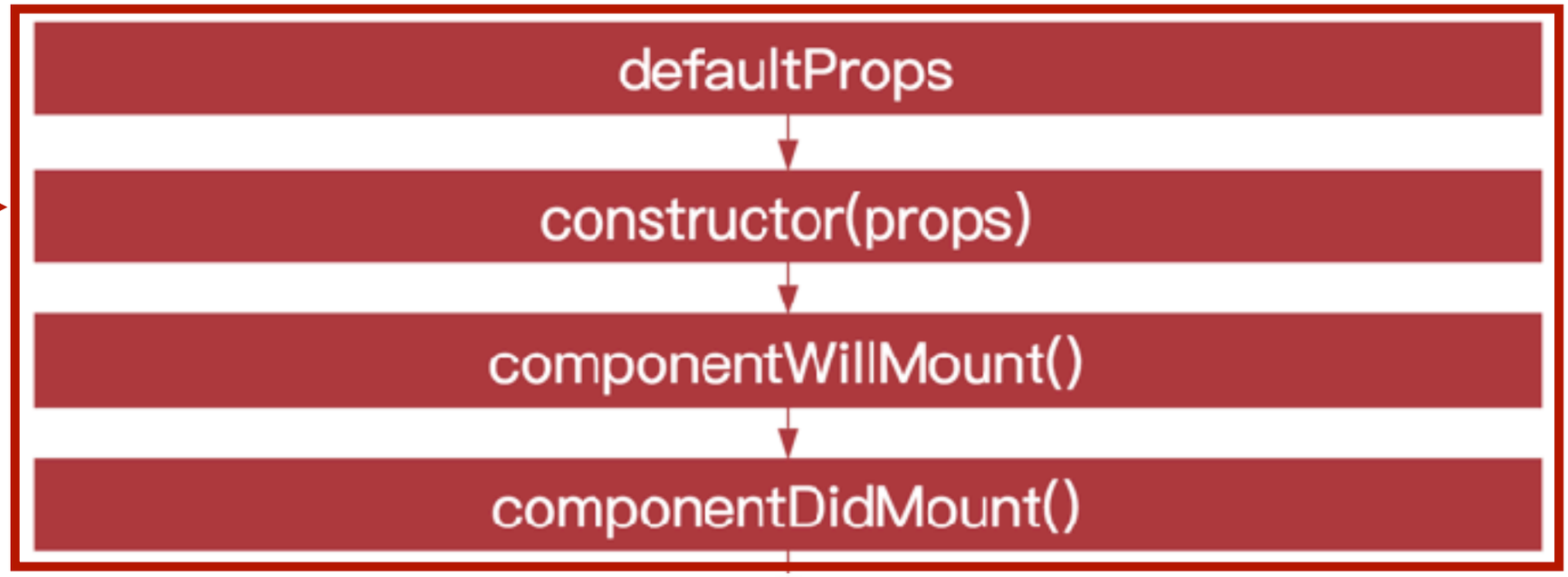
```
export default class App extends Component {
    static defaultProps = {
        title: '',
    };
    static propTypes = {
        title: PropTypes.string.isRequired,
    };
    constructor(props) {
        super(props);
        this.state = {description: 'Welcome to React Native !'}
    }
    componentWillMount () {}
    componentDidMount() {}
    componentWillReceiveProps(nextProps) {
        this.setState({/*description: 'Hello World !'*/});
    }
    shouldComponentUpdate(nextProps, nextState) {return true;}
    componentWillUpdate(nextProps, nextState) {}
    componentDidUpdate(preProps, preState) {}
    render() {
        return (
            <View>
                <BriefView description={this.state.description}/>
            </View>
        );
    }
    componentWillUnmount() {}
}
```

RN组件生命周期

- 初始化及挂载阶段
- 运行期阶段
- 卸载阶段

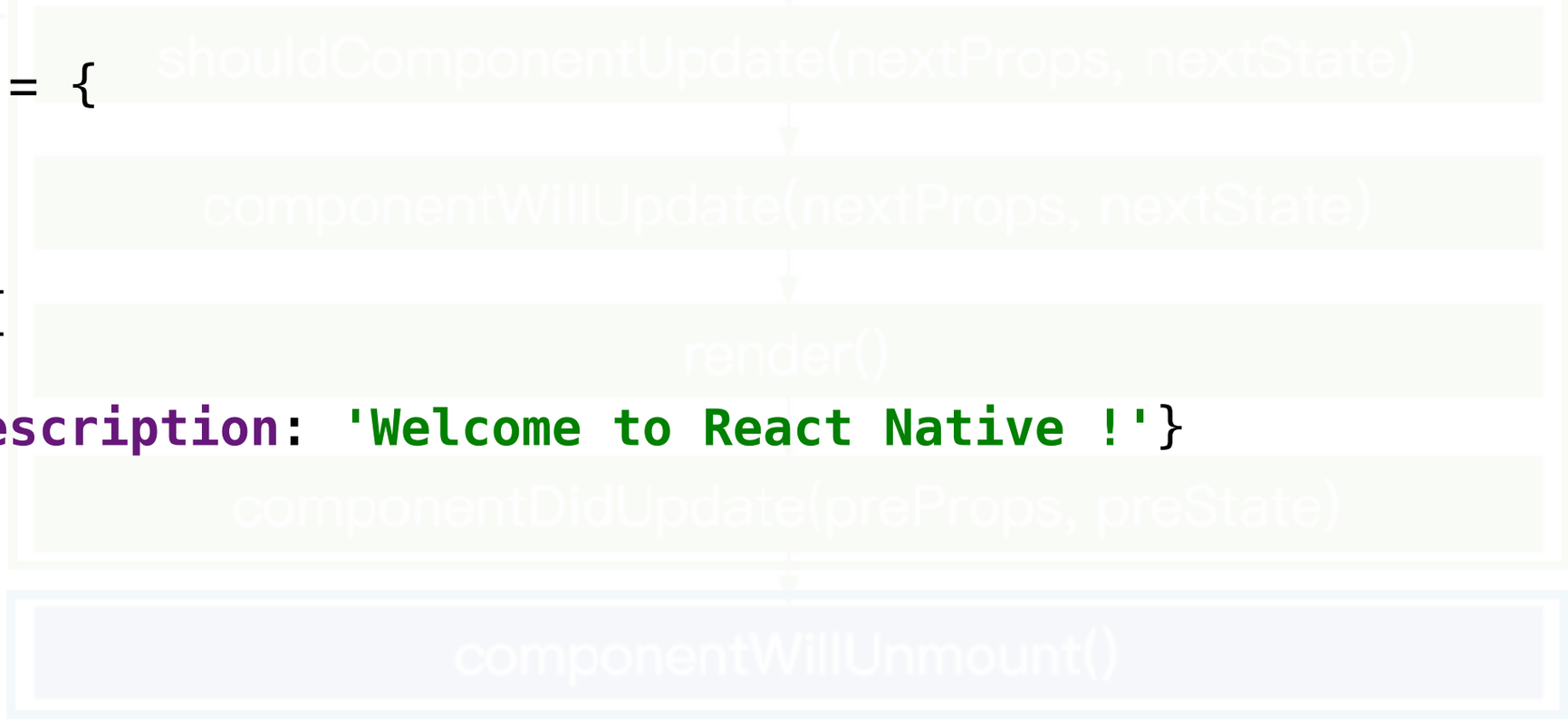
RN组件生命周期

- 初始化及挂载阶段



- 运行期阶段

```
static defaultProps = {
    title: '',
};

constructor(props) {
    super(props);
    this.state = {description: 'Welcome to React Native !'}
}
```



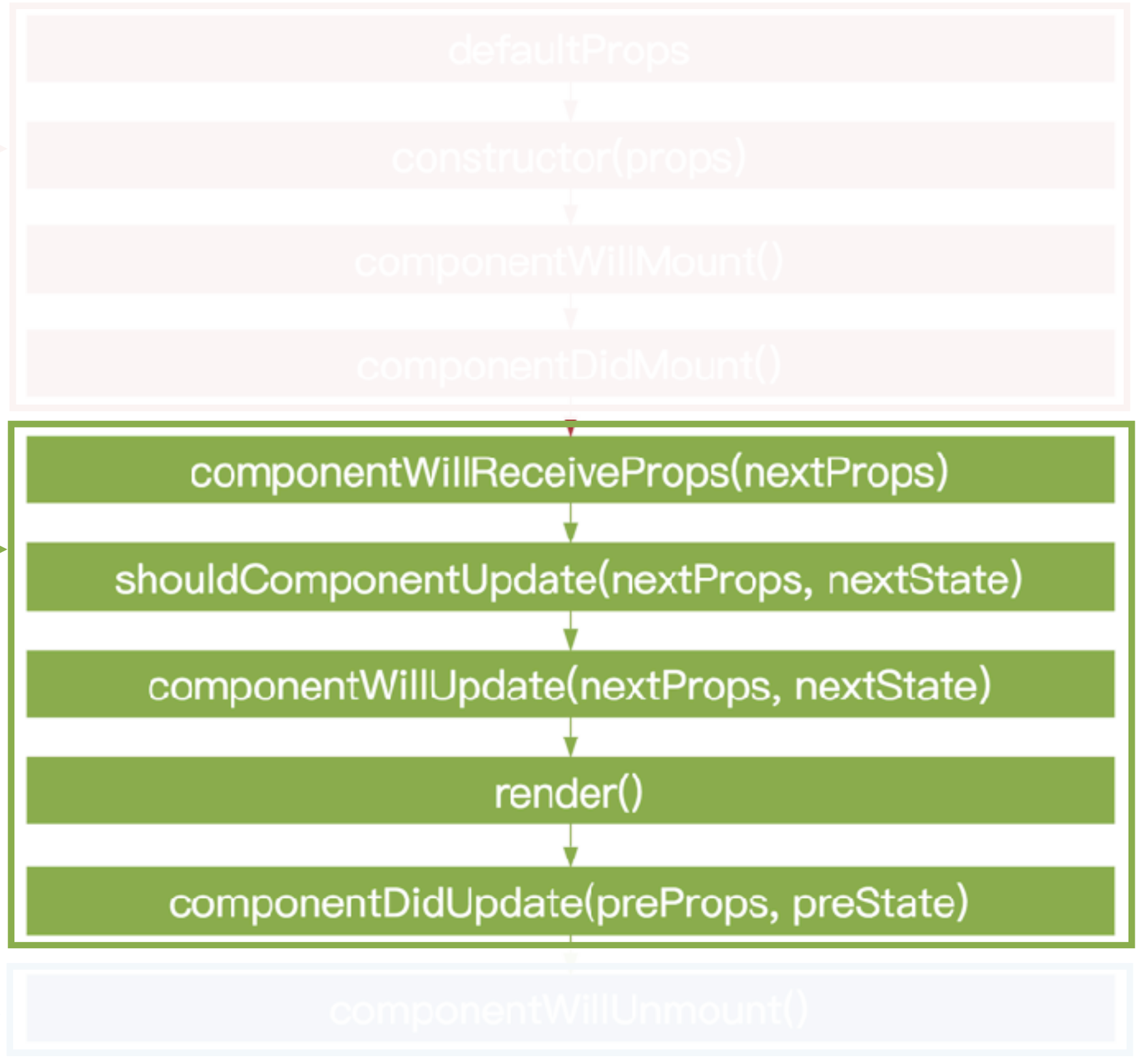
- 卸载阶段

RN组件生命周期

- 初始化及挂载阶段

- 运行期阶段

- 卸载阶段

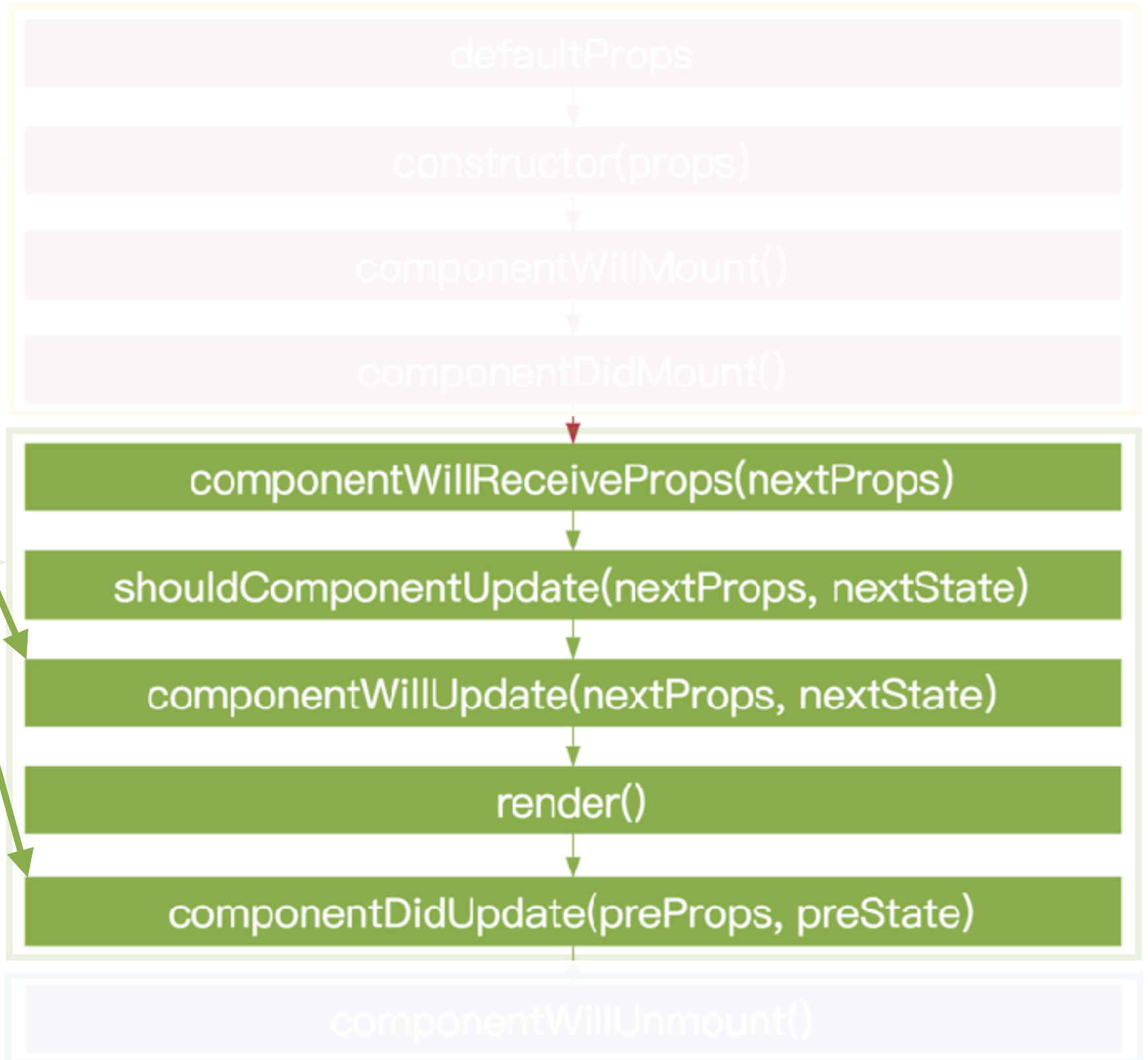


RN组件生命周期

- 如果在这两个方法中调用this.setState({});方法会出现什么问题?

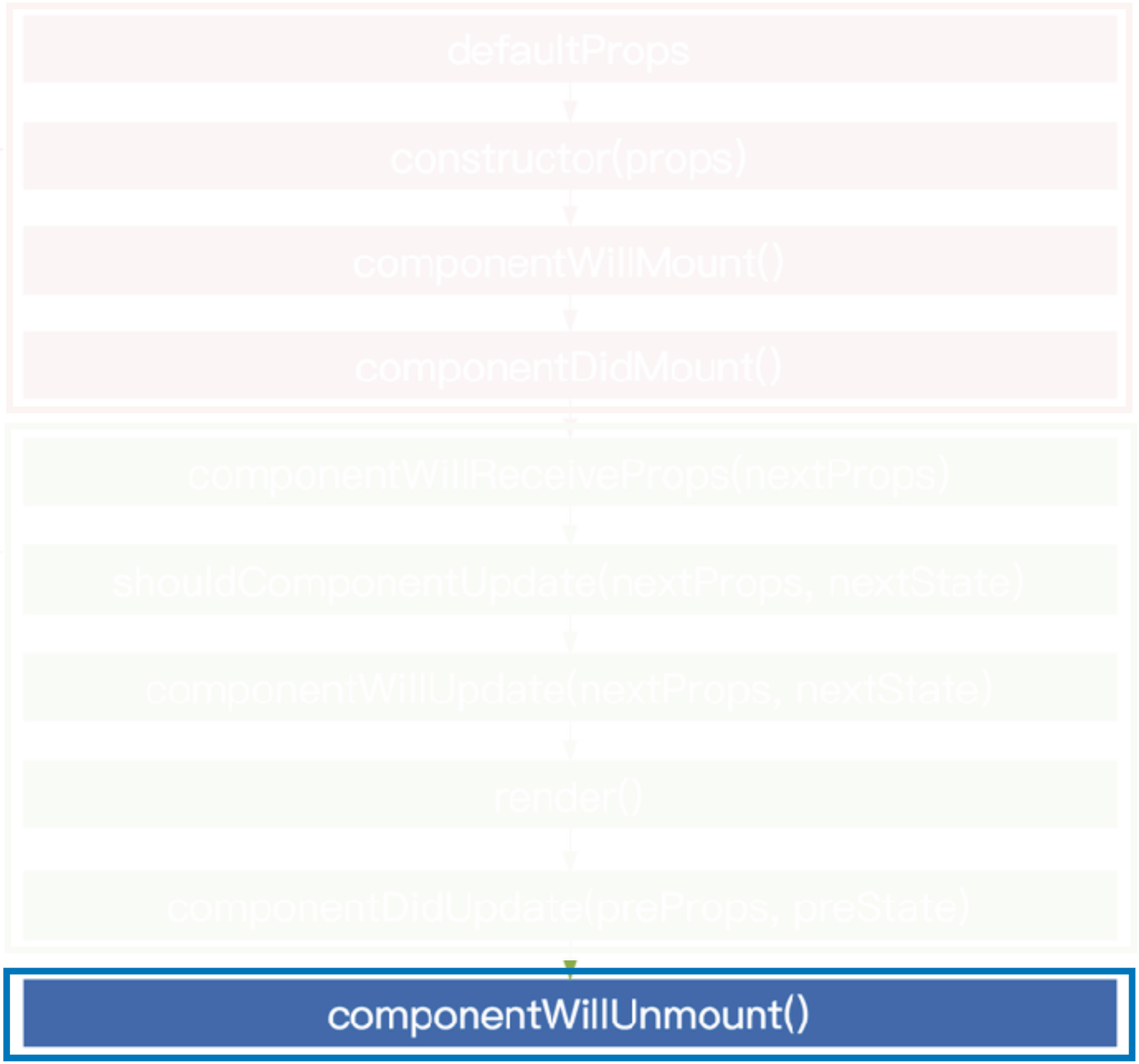
- 运行期阶段

- 卸载阶段



RN组件生命周期

- 初始化及挂载阶段



- 运行期阶段

- 卸载阶段

componentWillUnmount()

RN - CSS layout

```
export default class App extends Component {
    render() {
        return (
            <View>
                <Text style={styles.text}>
                    Welcome to React Native !
                </Text>
            </View>
        );
    }
}
const styles = StyleSheet.create({
    text: {
        fontSize: 32,
        color: '#000',
    }
})
```

RN - CSS layout

```
text: {
    fontSize: 32,
    color: '#000',
}
```


```
text: {
    fontSize: 32,
    color: '#F00',
}
```

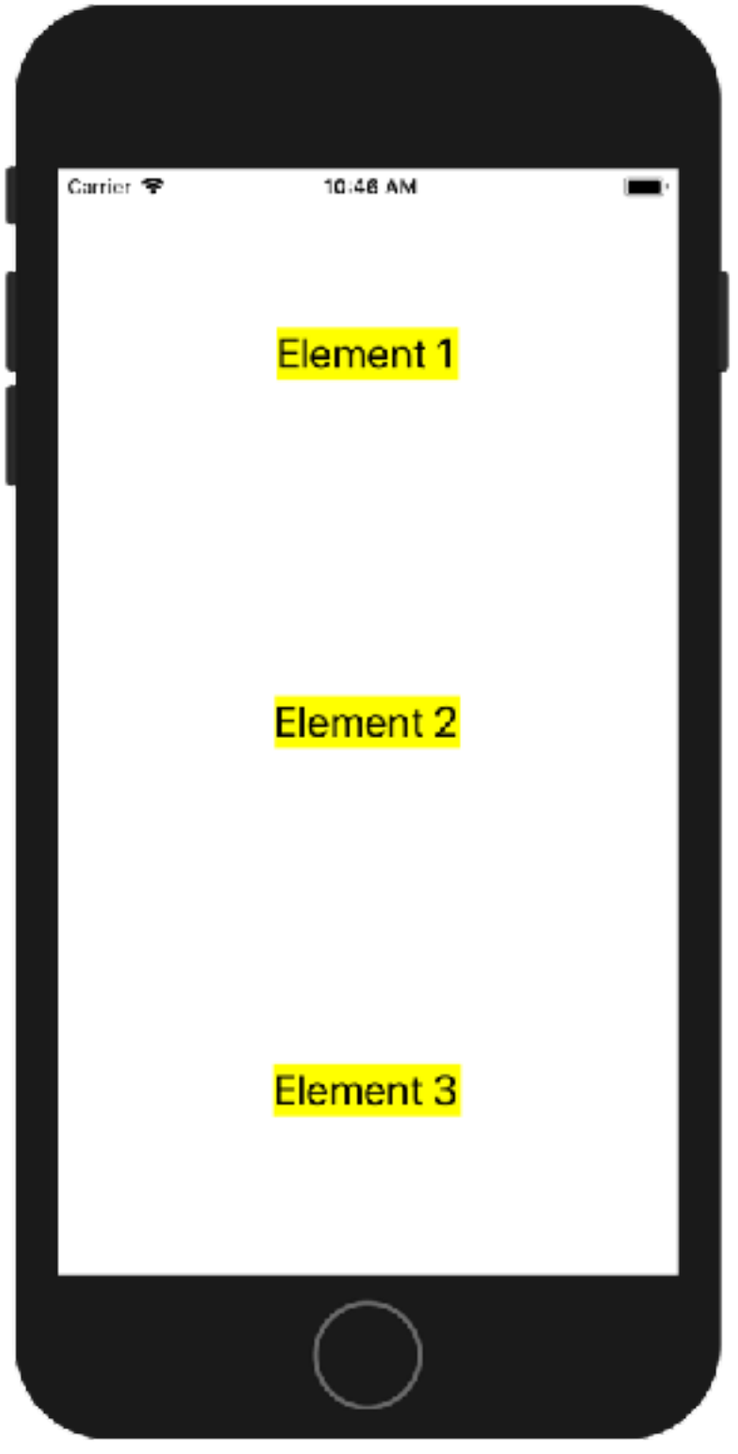
RN - CSS layout

width
height
backgroundColor
borderWidth
borderRadius
borderColor
...

fontWeight
shadowColor
shadowOffset
shadowRadius
shadowOpacity
margin

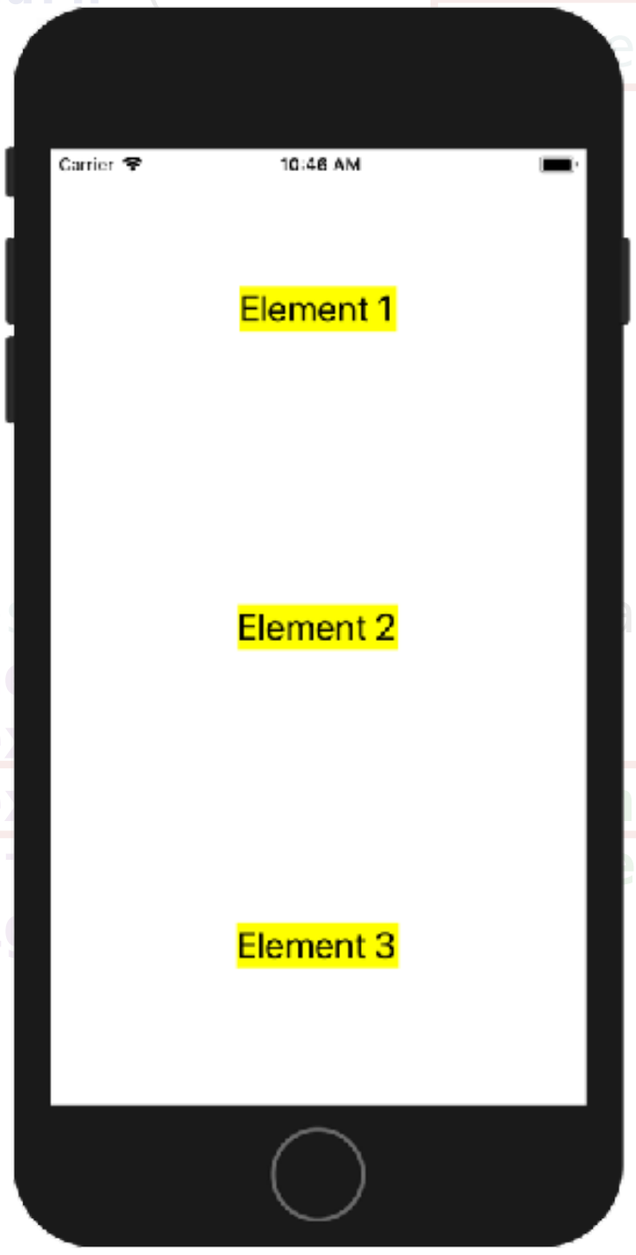
RN - CSS layout

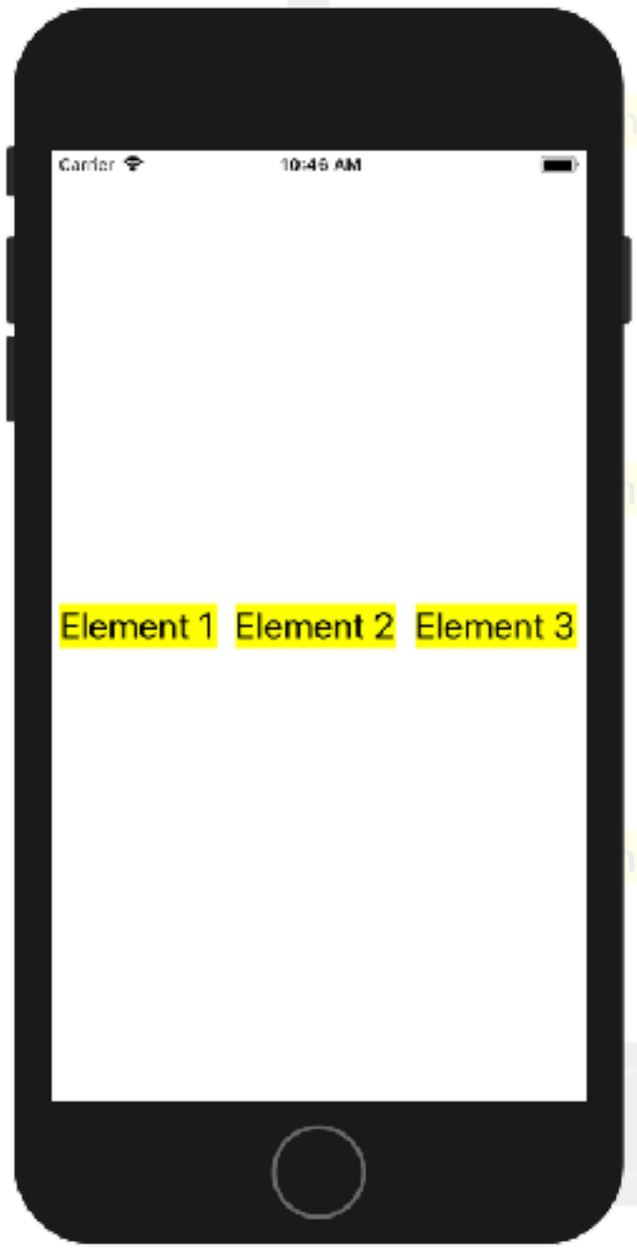
```
export default class App extends Component {
    render() {
        return (
            <View style={styles.container}>
                <Text>Element 1</Text>
                <Text>Element 2</Text>
                <Text>Element 3</Text>
            </View>
        );
    }
}

const styles = StyleSheet.create({
    container: {
        flex:1,
        flexDirection: 'column',
        justifyContent: 'space-around',
        alignItems:'center',
    }
})
```



RN - CSS layout

`flexDirection: 'column',` → `flexDirection: 'row',`





RN - CSS layout

```
export default class App extends Component {
    render() {
        return (
            <View style={styles.container}>
                <Text>Element 1</Text>
                <Text>Element 2</Text>
                <Text>Element 3</Text>
            </View>
        );
    }
}

const styles = StyleSheet.create({
    container: {
        flex:1,
        flexDirection: 'column',
        justifyContent: 'space-around',
        alignItems:'center',
    }
})
```

flex

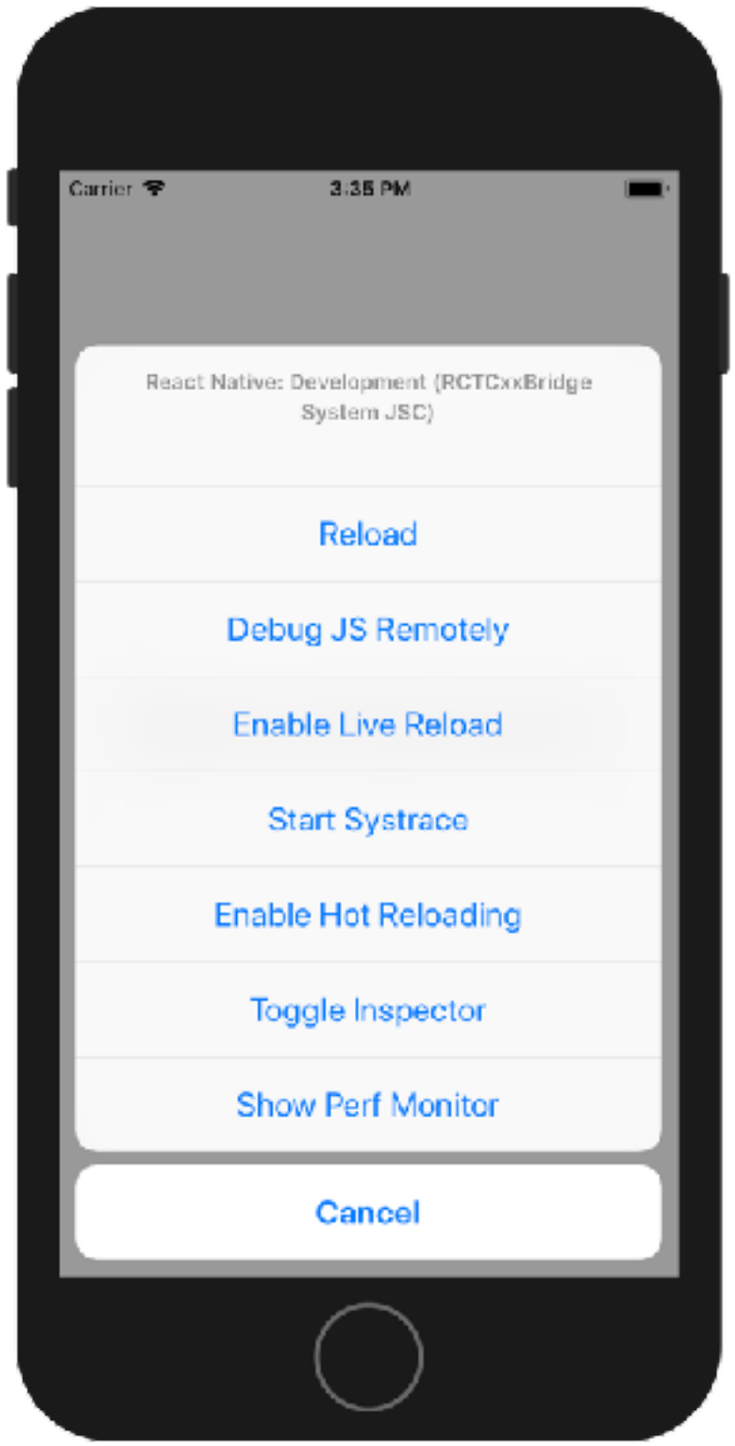
flexDirection

justifyContent

alignItems

...

RN - 调试

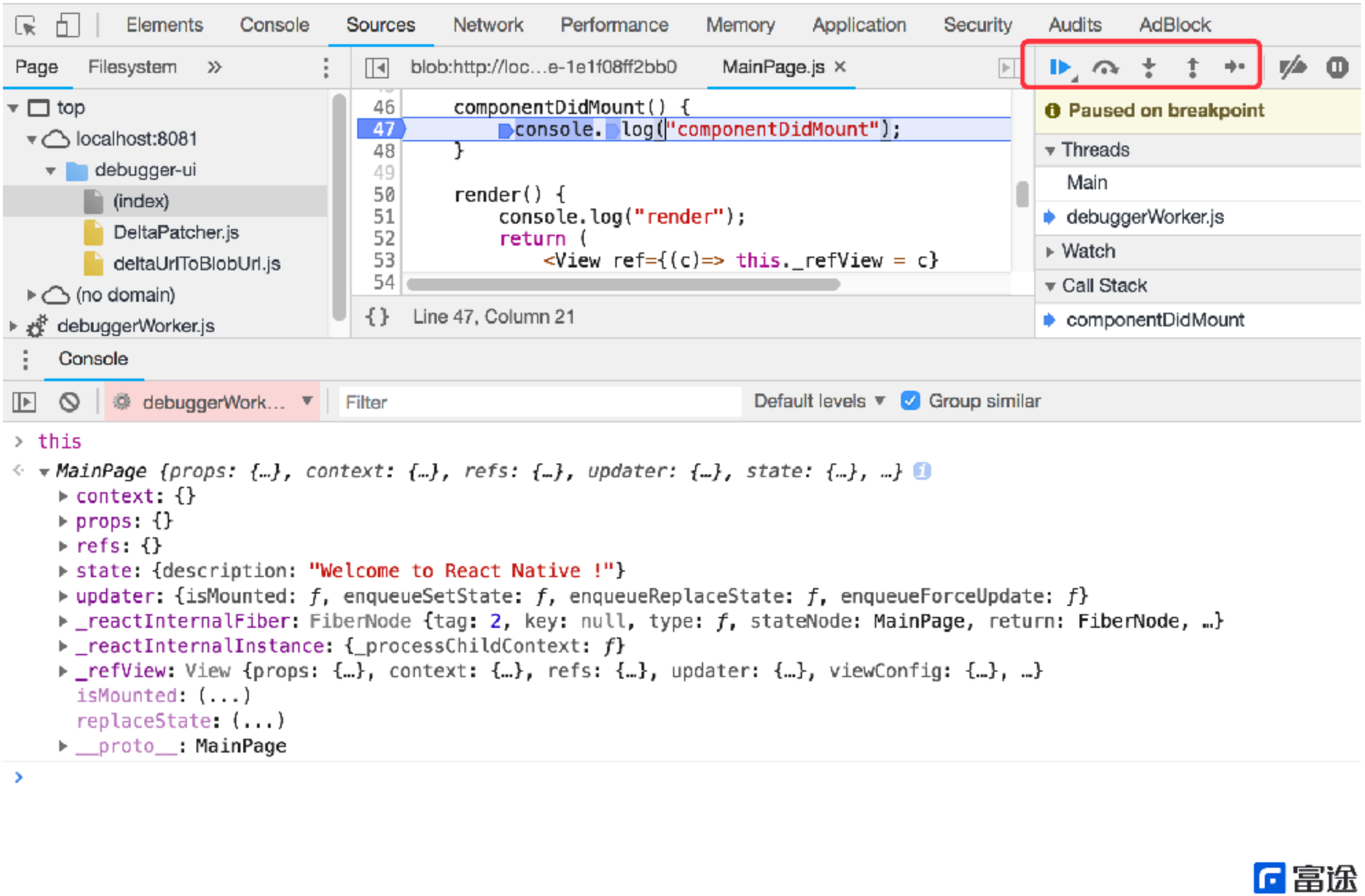


1、Command+D 调出开发者菜单

2、Command+R 刷新界面

3、支持远程调试JS代码

RN - 调试 - 断点调试


Elements
Console
Sources
Network
Performance
Memory
Application
Security
Audits
AdBlock
Page
Filesystem
blob:http://loc...e-1e1f08ff2bb0
MainPage.js
top
localhost:8081
debugger-ui
(index)
DeltaPatcher.js
deltaUrlToBlobUrl.js
(no domain)
debuggerWorker.js
46 componentDidMount() {
47 console.log("componentDidMount");
48 }
49
50 render() {
51 console.log("render");
52 return (
53 <View ref={(c)=> this._refView = c}
54
{} Line 47, Column 21
Paused on breakpoint
Threads
Main
debuggerWorker.js
Watch
Call Stack
componentDidMount
Console
debuggerWork...
Filter
Default levels
Group similar
> this
MainPage {props: {…}, context: {…}, refs: {…}, updater: {…}, state: {…}, …}
context: {}
props: {}
refs: {}
state: {description: "Welcome to React Native !"}
updater: {isMounted: f, enqueueSetState: f, enqueueReplaceState: f, enqueueForceUpdate: f}
_reactInternalFiber: FiberNode {tag: 2, key: null, type: f, stateNode: MainPage, return: FiberNode, …}
_reactInternalInstance: {_processChildContext: f}
_refView: View {props: {…}, context: {…}, refs: {…}, updater: {…}, viewConfig: {…}, …}
isMounted: (...)
replaceState: (...)
__proto__: MainPage
>

React Native 优缺点及新动态

RN优缺点

优点：

1、跨平台，iOS、Android

2、热更新，迭代速度快

3、拥有web开发的效率，同时兼顾了原生的性能

4、社区活跃

5、快速编译，hot reload

RN优缺点

缺点：

1、崩溃监控

2、Android兼容不理想

3、无法在Android发布64位应用

4、开发体验一般

RN新动态 - 底层重构

1、改变线程模型，优化对UI的响应

2、优化渲染

3、简化桥接，原生、JS间直接调用效率更高

4、更轻松的构建调试工具，如跨语言堆栈跟踪

附录

- React Native中文官网
- React Native学习指南
- Flex 布局教程：语法篇
- 优秀的第三方RN开源组件列表
- FB正在重构RN，将重写大量底层
- React Native 分析总结

Q & A

谢谢