React Native和React有啥区别?

8 个回答 默认排序

不知道
不太自由的自由职业者

11 人赞同了该回答

React native 是用React的方式开发mobileApp。
和phonegap不同，React Native 是Native 控件，但以React component 的方式expose 出来。
React Native 现在对Android 的支持还不太成熟，Ios版的不错，而且开源了

发布于 2015-08-15

图灵
IT/计算机书籍/IT人文书籍/Python/数据分析

13 人赞同了该回答

React Native 是一款用来开发真正原生渲染的 iOS 和 Android 移动应用的 JavaScript 框架。React 是一款 Facebook 公司开源的 JavaScript 用户界面开发框架，React Native基于React而产生。但 React 将浏览器作为渲染平台，而 React Native 的渲染平台则是移动设备。它们开发语法相似，都使用 JSX 语法进行开发，这种语法结合了 JavaScript 和类 XML 标记语言。

对于习惯了 Web 平台的 React 开发者来说，这意味着你可以使用熟悉的工具来开发真正原生的移动应用。下面我们来看一看React Native 与用于Web 平台的React 的主要区别，并顺带讲一些关键的概念。

React Native是如何工作的

在移动环境中使用React 是怎样实现的呢？为了更好地理解React Native 的工作原理，我们首先需要回顾一下React 的一个特点：Virtual DOM（虚拟DOM）。

在React 中，Virtual DOM 就像是一个中间层，介于开发者描述的视图与实际在页面上渲染的视图之间。为了在浏览器上渲染出可交互的用户界面，开发者必须操作浏览器的文档对象模型（DOM，document object model）。这个操作代价昂贵，对DOM 的过度操作将会给性能带来严重的影响。React 维护了一个内存版本的DOM，通过计算得出必要的最小操作并重新渲染。图2-1 展示了这个工作过程。

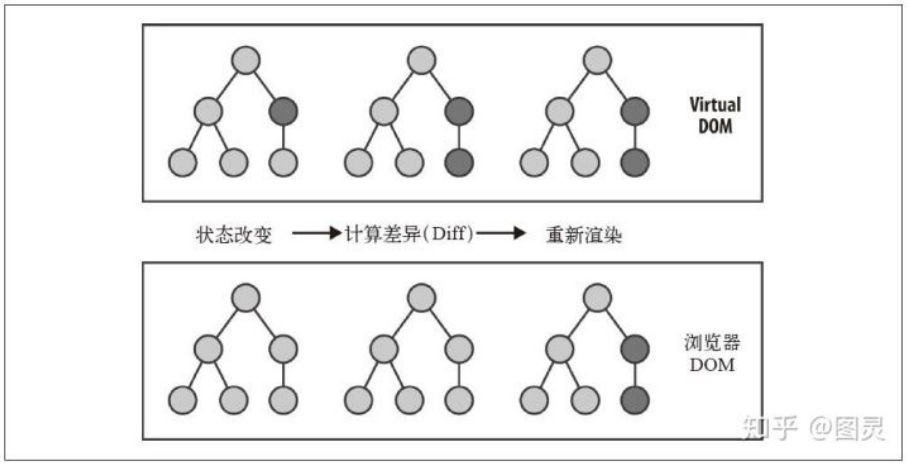

图2-1：执行Virtual DOM 的计算，减少浏览器DOM 的重复渲染

对于Web 环境的React 而言，大多数的开发者认为Virtual DOM 的出现主要是为了优化性能。Virtual DOM 确实能提升性能，但它主要的潜力在于提供了强大的抽象能力。在开发者的代码与实际的渲染之间加入一个抽象层，这带来了很多可能性。想象一下，如果React 能够渲染到浏览器以外的其他平台呢？毕竟，React 已经"理解"了你的应用应该如何展现。

确实，这就是React Native 的工作原理，如图2-2 所示。React Native 调用Objective-C 的API 去渲染iOS 组件，调用Java API 去渲染Android 组件，而不是渲染到浏览器DOM 上。

这使得React Native 不同于那些基于Web 视图的跨平台应用开发方案。

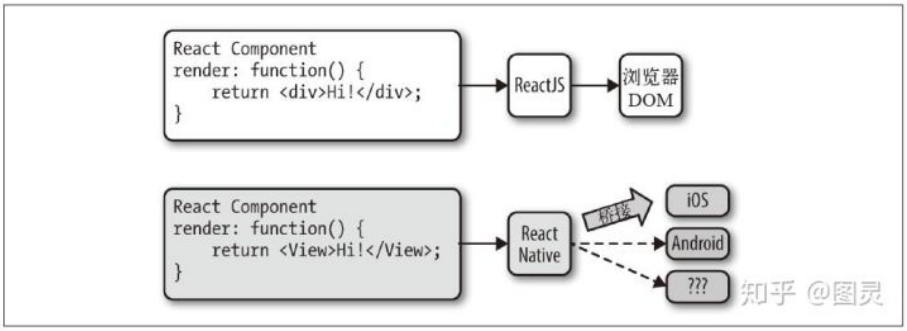

图2-2：React 可以渲染到多平台上

桥接令这一切成为可能，它使得React 可调用宿主平台开放的UI 组件。React 组件通过render 方法返回了描述界面的标记代码。如果是在Web 平台上，React 最终将把标记代码解析成浏览器的DOM；而在React Native 中，标记代码会被解析成特定平台的组件，例如<View> 将会表现为iOS 平台上的UIView。

渲染周期

如果你习惯使用React，那你应该熟悉React 的生命周期。当React 在Web 环境中运行时，渲染周期始于React 组件挂载之后（见图2-3）。

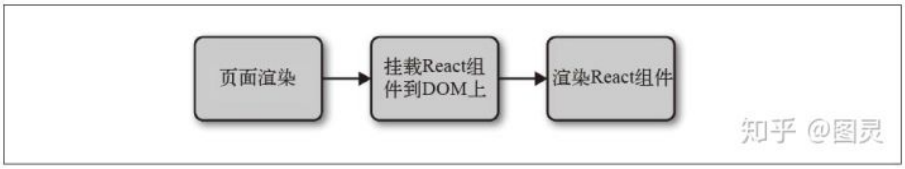

图2-3：React 组件挂载过程

接着，React 进入渲染周期并根据需要渲染组件（见图2-4）

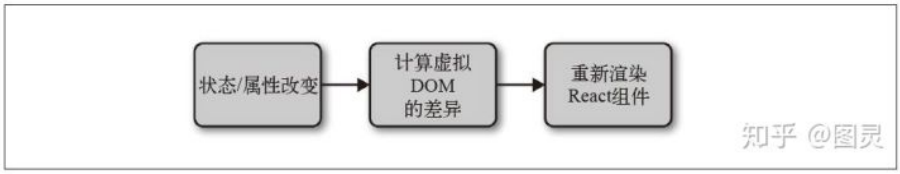

图2-4：React 组件重新渲染过程

在渲染阶段，React 将开发者在render 方法中返回的HTML 标记直接按需渲染到页面上。

至于React Native，生命周期与React 基本相同，但渲染过程有一些区别，因为React Native 依赖于桥接，正如先前图2-2 所示。JavaScript 通过桥接的解析，间接调用宿主平台的基础API 和UI 元素（也就是Objective-C 或Java）。由于React Native 不在UI 主线程运行，它可以在不影响用户体验的前提下执行这些异步调用。

创建组件

所有的React 代码都存在于React 组件中。React Native 组件与React 组件大体上一致，但在渲染和样式方面有一些重要的区别。

编写视图

当编写Web 环境的React 时，视图最终需要渲染成普通的HTML 元素（ 、<p>、、<a> 等）。而在React Native 中，所有的元素都将被平台特定的React 组件所替换（见表2-1）。最基础的组件是能跨平台的<View>，这是一个简单且灵活的UI 元素，类似于 标签。例如，在iOS 中，<View> 组件被渲染成UIView，而在Android 平台上则被渲染成View。

表2-1：React与React Native基础元素的比较

React	React Native
	<View>
	<Text>
、	<FlastList> 中的子条目
<img>	<Image>

其他组件则是平台特定的。例如，<DatePickerIOS> 组件显然将被渲染成iOS 标准的日期选择器（见图2-5）。下面是从RNTester 示例应用中摘录出来的代码，用来展示iOS 日期选择器。正如你期待的那样，用法相当直观：

```
<DatePickerIOS
  date={this.state.date}
  mode="time"
/>
```

图2-5：<DatePickerIOS>，顾名思义，是iOS 特有的组件

我们所有的UI 元素均为React 组件，而不是像 这样基础的HTML 元素，因此我们在使用每一个组件之前，都需要显式地进行导入。例如，我们可以这样导入<DatePickerIOS>组件：

```
import { DatePickerIOS } from "react-native";
```

RNTester 应用是一个打包的标准React Native 示例（facebook/react-nativetree/master/RNTester），可以让你查看它所支持的所有UI 元素，建议你体验一下其中包含的各种元素。除此之外，它还讲解了许多关于样式和交互的知识。

> 平台特定的元素和API 在官方文档中有特殊的标签，通常使用平台名称作为后缀，例如<TabBarIOS> 和<ToolbarAndroid>。

这些组件因平台而不同，因此在使用React Native 时，如何组织你的组件变得尤为重要。在Web 环境的React 中，我们通常混合各种React 组件，有的组件控制逻辑及其子组件，而有的则渲染原生标记。在使用React Native 时，如果你想复用代码，那么这些组件的抽象分离就至关重要。当然，如果一个组件渲染<DatePickerIOS> 元素，那它显然不能在Android 平台复用了。不过，如果一个组件封装的是关联逻辑，那就可以被复用。因此，视图组件可以根据平台进行替换选择。如果你乐意的话，还可以为组件设计平台特定的版本，例如picker.ios.js 和picker.android.js。

使用JSX

与React 相一致，React Native 也是通过编写JSX 来设计视图，并将视图标记和控制逻辑组合在一起成为一个文件。React 刚问世的时候，JSX 在业界引起了强烈的反响。对于许多Web 开发者来说，根据技术进行文件分离是理所当然的：保持CSS、HTML 和JavaScript文件的独立。然而将标记、控制逻辑，甚至样式合并成一门语言难免会让人觉得混乱。

JSX 认为减少心智负担比文件分离更有用。在React Native 中，这一点表现得更为明显。在一个没有浏览器的世界里，每个组件的样式、标记和行为被统一成单个文件的形式将会更有意义。因

此，React Native 中的.js 文件实际上就是JSX 文件。如果你正在使用原生JavaScript 编写Web环境的React，你可以考虑转换到JSX 语法来编写React Native 项目。

假如你之前从未使用过JSX，也不用太担心，它非常简单。举个例子，用纯JavaScript 编写React组件的代码看起来如下：

```
class HelloMessage extends React.Component {
  render() {
    return React.createElement(
      "div",
      null,
      "Hello ",
      this.props.name
    );
  }
}

ReactDOM.render(
  React.createElement(HelloMessage, { name: "Bonnie" }), mountNode);
```

我们可以通过使用JSX 使其更为简洁，使用类XML 标记来代替调用React.createElement方法并传入一组HTML 属性的做法。

```
class HelloMessage extends Component {
  render() {
      // 返回标记，而不是调用createElement方法
    return <div>Hello {this.props.name}</div>;
  }
}

// 我们不再需要调用createElement方法
ReactDOM.render(<HelloMessage name="Bonnie" />, mountNode);
```

以上两段代码最终都会在页面上被渲染为下面的HTML：

```
<div>Hello Bonnie</div>
```

原生组件的样式

在Web 中，正如使用HTML 标签一样，我们仍然使用CSS 来为React 组件添加样式。不论你是否喜欢CSS，它都已经成为Web 开发不可或缺的一部分。React 通常不影响我们编写CSS 的方式，并且它确实让样式的动态创建（通过props 和state）更加容易。除此之外，React 基本上不关心我们是如何处理样式的。

非Web 平台上有大量的方法来处理布局和样式。但好在我们使用React Native 时，只需要用一种标准的方法来处理样式。React 和宿主平台之间的桥接包含了一个缩减版CSS 子集的实现。这个CSS 子集主要通过flexbox 进行布局，做到了尽量简单化，而不是去实现所有的CSS 规则。有别于Web 平台，CSS 的支持程度因浏览器而不同，React Native 则做到了样式规则的一致。在React Native 配套的RNTester 应用中不仅可以查看许多UI 元素，还能看到许多支持的样式例子。

React Native 也坚持使用内联样式，通过JavaScript 对象进行样式组织。React 团队先前也提倡在Web 环境的React 中使用内联样式。如果你曾经在React 中使用过内联样式，那么下面的语法你一定非常熟悉了：

```
// 定义一个样式
const style = {
  backgroundColor: 'white',
  fontSize: '16px'
};

// 然后使用它
const txt = (
```

```
  <Text style={style}>
    A styled Text
  </Text>);
```

为了让样式更容易管理，React Native 为我们提供了创建和扩展样式的工具。

内联样式的写法让你觉得难受？它基于Web 背景而产生，被公认为标准实践的一个突破。相对于样式表来说，使用样式对象可能需要一些思维上的调整，从而改变你编写样式的方法。然而，在 React Native 中，这是一个实用的转变。

宿主平台API

使用Web 环境的React 与React Native 最大的不同，应该就在于宿主平台的API 了。在Web 中，我们通常要处理采纳标准的不一致和碎片化所引起的问题，并且大多数浏览器只支持部分核心的特性。然而在React Native 中，平台特定的API 在提供优秀原生的用户体验方面发挥了巨大的作用。当然，要考虑的方面还有很多。API 囊括了许多功能，从数据存储到地理服务，以及操控硬件设备（如摄像头）等。非常规平台上的API 会更有趣，例如，React Native 和虚拟现实头盔之间的API 会是什么样的呢?

默认情况下，iOS 和Android 版本的React Native 支持许多常用的特性，甚至可以支持任何异步的本地API。React Native 让宿主平台API 的使用变得更加简单和直观，你可以在其中自由地试验。同时，务必思考一下怎样做才符合目标平台的体验，并在心里设计好交互过程。

毋庸置疑，React Native 的桥接不可能暴露宿主平台全部的API。如果你需要使用一个未支持的特性，完全可以自己动手添加到React Native 中。另外，如果其他人已经集成，那就更好了，所以应该及时查看社区中的实现。

值得注意的是，使用平台API 也会对代码复用有帮助。同时，实现平台特定功能的React组件也是平台特定的。隔离和封装这些组件将会给你的应用带来更大的灵活性。当然，这对你开发Web 应用同样奏效，如果你想共享React 和React Native 的代码，请记住像DOM 这样的API 在React Native 中并不存在。

发布于 2019-04-19

王佳裕
INFJ | 程序员

11 人赞同了该回答

编程思路会有所不同，react 直接渲染dom，而rn生成id，用bridge（最新用c++实现了）变成一个表，等待 native 去调用，写react可以用前端知识直接上手，rn虽然也可以，但是深入下去没有native知识支持很难，有时候也无法理解一些现象，正在努力学习oc。

又比如单元测试，react可以用e2e的集成测，而rn要用到react中的shadow renderer，思路上也不同

利益相关：
前段时间写了一个较为完整的app，已上架
GitHub - fakefish/Weekly75: react-native iOS版奇舞周刊

编辑于 2016-05-13

题叶
ClojureScript 爱好者

27 人赞同了该回答

同样的 react virtual dom 作为 dsl，react.js 以浏览器 DOM 作为后端，react native 以 iOS 或 Android 原生控件作为后端。

发布于 2016-05-13