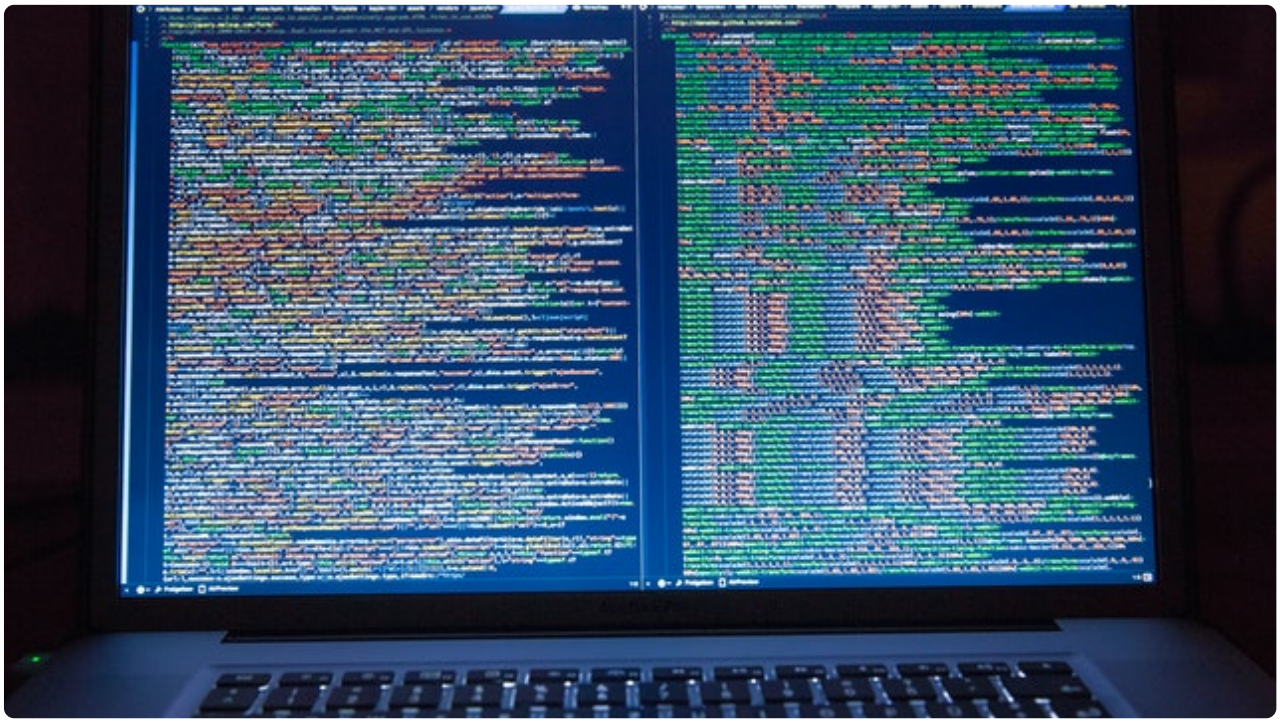
03 你对 ReactDOM.render(...) 操作了解多少

更新时间：2020-08-11 15:06:23



> 低头要有勇气，抬头要有底气。——韩寒

前言

ReactDOM.render 函数是整个 React 应用程序首次渲染的入口函数，你对它了解多少呢？本节主要介绍 ReactDOM 对象有哪些属性与方法，ReactDOM.render 函数在的三个重要参数分别是什么，以及函数返回值是什么。

ReactDOM 对象

ReactDOM 对象的定义，见代码示例 1.2.1。

> 注：本专栏引用的 React 源码版本为 v16.9。

```
// 源码位置：packages/react-dom/src/client/ReactDOM.js
const ReactDOM = {
  findDOMNode: function(...) { ... },
  hydrate: function(...) { ... },
  render: function (element, container, callback) {
    // 会先检验container是否有效，无效则停止执行且抛出错误
    // ...
    return legacyRenderSubtreeIntoContainer(null, element, container, false, callback);
  },
  unstable_renderSubtreeIntoContainer: function(...) {},
  unmountComponentAtNode: function(...) {}
  // ...
}
```

代码示例 1.2.1 ReactDOM 对象的定义

ReactDOM 对象上面有 findDOMNode 、 hydrate 和 render 等多个函数。其中 ReactDOM.render 函数有三个参数和一个返回值。下面内容将会对这三个参数和返回值进行详细说明。

理解 ReactDOM.render 函数的三个参数

ReactDOM.render(...) 的基本用法见代码示例 1.2.2。

```
import React from 'react';
import ReactDOM from 'react-dom';
import UpdateCounter from './pages/UpdateCounter';

ReactDOM.render(<UpdateCounter name="Taylor" />, document.getElementById('root'));
```

代码示例 1.2.2 ReactDOM.render 函数的使用

在代码示例 1.2.1 中，传入 ReactDOM.render 函数的两个参数分别是 <UpdateCounter name="Taylor" /> 和 document.getElementById('root') 。第二个参数很明显是 DOM 元素，也就是 React 应用程序最终渲染在页面中的容器。**那么，我们该如何理解第一个参数呢？**

UpdateCounter 是由 class 声明的一个「类」，它在 React 中被称为组件(component)。React 提供了 JSX 语法，基于 JSX 语法在函数或者「类」的两侧分别加上 < 和 /> 就变成了元素(element)。因此， <UpdateCounter name="Taylor" /> 就是一个 React 元素。在第二章中会详细介绍 React 组件和 React 元素。

第三个参数是应用程序渲染完成后的回调函数，这个参数是可选项，React 会在应用程序渲染完成后检查是否有回调函数，如果有则调用该回调函数。

ReactDOM.render 函数除了执行渲染任务外还有自己的返回值即 legacyRenderSubtreeIntoContainer 函数的执行结果。那么， legacyRenderSubtreeIntoContainer 函数的执行结果是什么呢？

ReactDOM.render 函数的返回值

在 React 源码中， legacyRenderSubtreeIntoContainer 函数内部通过 return 的形式又嵌套了多层函数。为了方便看到 ReactDOM.render 函数最终的返回值，使用 console.log(...) 将函数执行结果输出，见代码示例 1.2.3。

```
console.log('返回值',
  ReactDOM.render(
    <UpdateCounter name="Taylor" />, document.getElementById('root'), () => {console.log('渲染完成')}
  )
);

// 输出结果
UpdateCounter: {
  context: {},
  handleClick: ƒ (),
  props: {name: "Taylor"},
  refs: {},
  state: {count: 0, text: "点击计数"},
  // 更新触发器
  updater: {isMounted: ƒ, enqueueSetState: ƒ, enqueueReplaceState: ƒ, enqueueForceUpdate: ƒ},
  // 存储了首次渲染完成后对应的Fiber结点信息
  _reactInternalFiber: FiberNode {tag: 1, key: null, stateNode: UpdateCounter, elementType: …},
  _reactInternalInstance: {_processChildContext: ƒ},
  isMounted: (...),
  replaceState: (...),
 // 继承于React.Component
  __proto__: Component,
}
```

代码示例 1.2.3 ReactDOM.render 函数执行后的返回值

ReactDOM.render **函数的返回值是当前应用程序根组件的实例**。组件实例是 React 应用程序运行时在内存中的一种临时状态，组件实例的属性包括了自身类定义的属性以及继承于 React.Component 的属性。在 UpdateCounter 实例中， state 和 handleClick 为自身类的属性，而 context， props 和 updater 等则继承于 React.Component 。

小结

本章主要介绍了在研究 React 内部运行机制方面的一些思路与切入点以及 React 应用程序的首次渲染入口—ReactDOM.render 函数。下一章将会更加详细的介绍 React 世界中的一些基础概念，如 **React 组件，组件继承原理，React 元素设计思想，React 组件实例的作用以及组件与元素之间的关系** 等。

}

02 研究 React 应用程序的内部运行机制要找到切入点

04 React 世界中那些重要的概念