09 元素：从设计者的角度理解 React 元素

更新时间：2020-08-20 10:00:03

> 人生太短，要干的事太多，我要争分夺秒。——爱迪生

前言

React 元素是应用程序运行过程中用于描述页面结构的关键「数据/对象」。React 用元素来描述页面结构也是 React 拥有良好的性能表现的关键因素之一。本节将会介绍 React 元素的设计原理。

什么是 React 元素

React 官方文档 - 元素渲染 中给出了元素的描述：元素是构成 React 应用的最小砖块，它描述了你在屏幕上想看到的内容。在程序中它可以是代码示例 2.6.1这样的。

```
const element = (
  <h1 className="greeting">
    Hello, world!
  </h1>
);
```

代码示例 2.6.1 使用 JSX 定义元素

上面这个有趣的标签语法既不是字符串也不是 HTML，它被称为 JSX，是一个 JavaScript 的语法扩展，它可以生成 React 元素。React 元素在程序中也可以是代码示例 2.6.2 这样的。

```
const element = React.createElement(
  'h1',
  {className: 'greeting'},
  'Hello, world!'
);
```

代码示例 2.6.2 使用 createElement 函数定义元素

Babel 会把 JSX 转译成一个名为 `React.createElement()` 的函数调用。因此上面两种示例代码完全等效。`React.createElement( ... )` 会预先执行一些检查，以帮助我们编写无错代码，最终它创建的元素结构是代码示例 2.6.3 这样的对象。

```
// 这是简化后的结构
const element = {
  type: 'h1',
  props: {
    className: 'greeting',
    children: 'Hello, world!'
  }
};
```

代码示例 2.6.3 简化后的元素结构

代码示例 2.6.3 中的对象 **React 元素** 最终的形态，它们 **描述** 了我们希望在屏幕上看到的内容。React 对元素的详细定义见代码示例2.6.4。

```
// 源码位置：packages/shared/ReactElementType.js
var ReactElement = function (type, key, ref, self, source, owner, props) {
  var element = {
    // 这个标记将其标识为React元素，且保证唯一性
    $$typeof: REACT_ELEMENT_TYPE,
    // 元素类型，如宿主元素（div，span），组件元素（UpdateCounter）
    type: type,
    key: key,
    // 组件实例的引用
    ref: ref,
    // 元素属性
    props: props,
    _owner: owner
  };
  // ...
  return element;
}
```

代码示例 2.6.4 React 对元素的定义

现在来思考一个问题，**为什么 React 要用元素这样的对象结构来描述页面结构？**

对象方便处理（添加/修改/删除属性）。应用程序渲染时，React 首先读取应用程序的元素对象，然后使用它们来构建 DOM 以及保持随时更新。与浏览器的 DOM 元素不同，React 元素是创建开销极小（因为属性少）的普通对象。React 会负责更新元素来保证与 DOM 保持一致。

元素的种类与组成

React 元素主要可以分为两大类，分别是 React 组件元素和宿主组件（如`div`，`span`等）元素。这两类元素可以组合嵌套构成各种元素来描述任何 DOM 结构。

> 注：宿主组件元素（HostComponent）在 React 中一般表示的是 DOM 元素。

一、React 元素全部由宿主组件元素构成的情况

React 元素全部由 DOM 元素组成，JSX 的写法见代码示例 2.6.5。

```
const element = (
  <button class='button button-blue'>
    <b>
      OK!
    </b>
  </button>
)
```

代码示例 2.6.5 纯 DOM 元素组成的 React 元素

上面 JSX 的代码实现的元素被编译后的形态见代码示例 2.6.6。

```
element = {
 type: 'button',
 props: {
  className: 'button button-blue',
  children: {
   type: 'b',
   props: {
    children: 'OK!'
   }
  }
 }
}
```

代码示例 2.6.6 纯 DOM 元素组成的 React 元素最终形态

这里需要注意的是 **元素之间通过 children 属性进行嵌套**。在创建元素（树）时，可以指定一个或多个子元素作为其包含（父）元素的 children 属性值。

二、React 元素由组件和 DOM 元素组合构成的情况

事实上，在组件的两边加上 < 和 /> 就变成了元素，它代表的 UI 部分就是组件 render 函数的返回值。组件元素就像 DOM 元素一样，它们也可以相互嵌套和混合。图 2.6.1一个删除提示组件，程序见代码示例 2.6.7。

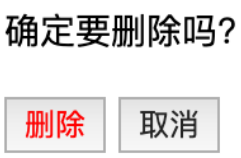

图 2.6.1 删除提示组件

```
// DeleteButton组件
class DeleteButton extends React.Component {
  render() {
    return <button className="redBtn">{this.props.children}</button>;
  }
}
// Operation组件
class Operation extends React.Component {
  render() {
    return (
      <div>
       <p>确定要删除吗?</p>
        <DeleteButton>删除</DeleteButton>
        <button>取消</button>
      </div>
    )
  }
}

const element = <Operation />
ReactDOM.render(element, document.getElementById('root'));
```

代码示例 2.6.7 删除提示组件代码结构

代码示例 2.6.7 中先定义了 `DeleteButton` 组件，然后在 `Operation` 组件中使用。`Operation` 组件的 UI 部分由 `DeleteButton` 组件和 宿主组件元素（ `p` ， `button` ）组成，该组件最后会被编译成的元素结构见代码示例 2.6.8。

```
// Operation组件返回的元素
{
  type: 'div',
  props: {
    children: [{
      type: 'p',
      props: {
        children: '确定要删除吗？'
      }
    }, {
      type: DeleteButton,
      props: {
        children: '删除'
      }
    }, {
      type: Button,
      props: {
        children: '取消'
      }
    }]
};

// DeleteButton组件返回的元素
{
  type: Button,
  props: {
    className: 'redBtn',
    children: '取消'
  }
}
```

代码示例 2.6.8 删除提示组件元素结构

小结

React 用元素来描述 DOM 结构的优点在于它们很容易遍历，不需要解析，并且它们比实际的 DOM 元素轻量得多！React 组件是由 UI 部分加逻辑部分组成，其中 UI 部分就是 React 元素，元素在 render 会被转换成 React Fiber 对象（结点）。Fiber 对象的层层嵌套形成了应用程序的 **Fiber树**，所有更新的处理都在这颗「树」中计算。

React 中组件和元素的根本区别就是 **元素普通对象，而组件是类和函数**，元素是组件的一部分。

}

08 组件实例：组件实例到底是什么？

10 React 如何定义更新队列以及它们之间的相互作用关系？