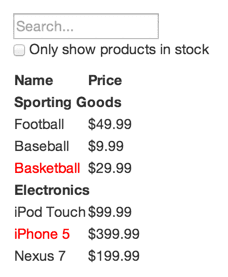
React 中许多重要部分之一是思考如何构建应用程序。 在本文档中，我们将引导你完成用 React 构建一个可搜索的产品数据表的思考过程。

## 从一个线框图开始

试想我们已经有一个 JSON API，和从设计者那里得来的一个线框图 。这个线框图如图所示：

[](http://www.css88.com/react/static/thinking-in-react-mock-1071fbcc9eed01fddc115b41e193ec11-4dd91.png)

我们的 JSON API 返回像这样的一些数据：

[

{category: "Sporting Goods", price: "$49.99", stocked: true, name: "Football"},

{category: "Sporting Goods", price: "$9.99", stocked: true, name: "Baseball"},

{category: "Sporting Goods", price: "$29.99", stocked: false, name: "Basketball"},

{category: "Electronics", price: "$99.99", stocked: true, name: "iPod Touch"},

{category: "Electronics", price: "$399.99", stocked: false, name: "iPhone 5"},

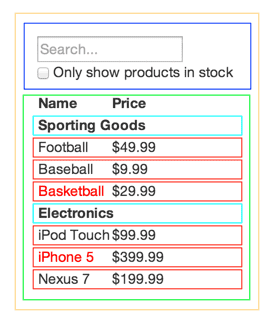
{category: "Electronics", price: "$199.99", stocked: true, name: "Nexus 7"}

];

## 步骤1：将 UI 拆解到组件层次结构中

你该如何拆分组件呢？其实只需要像拆分一个新方法或新对象一样的方式即可。一个常用的技巧是[单一职责原则](https://en.wikipedia.org/wiki/Single_responsibility_principle)，即一个组件理想情况下只处理一件事。如果一个组件持续膨胀，就应该将其拆分为多个更小的组件中。

由于你经常向用户显示 JSON 数据模型，你应该发现如果模型构建正确，则对应的 UI（以及组件结构）将很好地映射。这是因为 UI 和数据模型往往遵循相同的信息结构，这意味将UI拆分成组件的工作通常比较琐碎，只需要将其拆分为精确对应数据模型的片段即可。

[](http://www.css88.com/react/static/thinking-in-react-components-eb8bda25806a89ebdc838813bdfa3601-82965.png" \t "http://www.css88.com/react/docs/_blank)

你可以看到，在这个简单的小应用中，我们有 5 个组件。 我们已经对每个组件代表的数据进行了斜体标注。

1. **FilterableProductTable **（orange）：**** 包含整个示例
2. **SearchBar**（蓝色）：**** 接收所有的 用户输入
3. **ProductTable**（绿色）：**** 根据 用户输入 显示和过滤 数据集合
4. **ProductCategoryRow**（宝石绿）：**** 显示每个 类别 的标题
5. **ProductRow**（红色）：**** 显示每个 产品 的行数据

仔细观察 ProductTable ，你将会发现表头（包含 “Name” 和 “Price” 标签）不是独立的组件。这是个人偏好问题，当然也存在用其他方式实现的争论。在这个例子中，将其作为 ProductTable 的一部分来处理，因为它是 数据集合 渲染的一部分，是 ProductTable 的职责。当然，如果这个表头变得复杂的时候（比如，如果需要添加排序功能），这时候创建一个独立的 ProductTableHeader 组件更合适。

## 步骤2： 用 React 构建一个静态版本

目前为止你已经有了组件层次结构，现在是时候实现你的 app 了。最简单的方法是构建一个采用数据模型并渲染 UI 但没有交互性的版本。最好解耦这些处理，因为构建静态版本需要 大量的代码 和 少量的思考，而添加交互需要 大量思考 和 少量的代码。我们将看到原因。

要构建你 app 的一个静态版本，用于渲染数据模型， 您将需要构建复用其他组件并使用 props 传递数据的组件。props 是将数据从 父级组件 传递到 子级 的一种方式。如果你熟悉 state 的概念，在构建静态版本时 \*不要使用 \*state \*\* 。state 只用于交互，也就是说，数据可以随时被改变。由于这是一个静态版本 app，所以你并不需要使用 state 。

您可以 自上而下 或 自下而上 构建。

## 步骤3： 确定 UI state(状态) 的最小（但完整）表示

为了你的 UI 可以交互，你需要能够触发更改底层的数据模型。React 通过 ****state**** 使其变得容易。

找出你的应用程序所需 state(状态) 的绝对最小表示，并且可以以此计算出你所需的所有其他数据内容。例如，如果你正在构建一个 TODO 列表，只保留一个 TODO 元素数组即可;不需要为元素数量保留一个单独的 state(状态) 变量。

想想我们的示例应用中的所有数据。 我们有：

* 原始产品列表
* 用户输入的搜索文本
* 复选框的值
* 过滤后的产品列表

让我们仔细分析每一个数据，弄清楚哪一个是 state(状态) 。请简单地提出有关每个数据的 3 个问题：

1. 是否通过 props(属性) 从父级传入？ 如果是这样，它可能不是 state(状态) 。
2. 是否永远不会发生变化？ 如果是这样，它可能不是 state(状态)。
3. 是否可以由组件中其他的 state(状态) 或 props(属性) 计算得出？如果是这样，则它不是 state(状态)。

所以最终，我们的 state(状态) 是：

* 用户输入的搜索文本
* 复选框的值

## 步骤4：确定 state(状态) 的位置

记住：React 单向数据流在层级中自上而下进行。这样有可能不能立即判断出状态属于哪个组件。****这常常是新手最难理解的一部分，****试着按下面的步骤分析操作：

对于你应用中的每一个 state(状态) ：

* 确定每个基于这个 state(状态) 渲染的组件。
* 找出公共父级组件（一个单独的组件，在组件层级中位于所有需要这个 state(状态) 的组件的上面。愚人码头注：父级组件）。
* 公共父级组件 或者 另一个更高级组件拥有这个 state(状态) 。
* 如果找不出一个拥有该 state(状态) 的合适组件，可以创建一个简单的新组件来保留这个 state(状态) ，并将其添加到公共父级组件的上层即可。

我们在我们的应用中贯穿这个策略：

* ProductTable 需要基于 state(状态) 过滤产品列表，SearchBar 需要显示 搜索文本和选中状态 state(状态) 。
* 公共的父级组件是 FilterableProductTable
* 它从概念上讲适用于过滤文本和复选框选中值应该存在于 FilterableProductTable。

## 步骤5：添加反向数据流

目前，构建的应用已经具备了正确渲染 props(属性) 和 state(状态) 沿着层次结构向下传播的功能。现在是时候实现另一种数据流方式：层次结构中深层的 form(表单) 组件需要更新 FilterableProductTable 中的 state(状态) 。

想想我们希望发生什么。我们期望当用户改变表单输入的时候，我们更新 state(状态) 来反映用户的输入。由于组件只能更新它们自己的 state(状态) ，FilterableProductTable 将传递回调到 SearchBar，然后在 state(状态) 被更新的时候触发。我们可以使用 input 的 onChange 事件来接收通知。而且通过 FilterableProductTable 传递的回调调用 setState()，然后应用被更新。

## 就这么简单

非常希望，这篇文章能给你一些使用 React 构建组件和应用的想法。有可能这种写法会比你通常的写法多几行代码，但切记阅读代码的重要性远远高于写代码，模块化、结构清晰的代码最利于阅读。当创建一个大组件库的时候，你将感激模块化、结构清晰和可以重用的代码，同时你的代码行数会慢慢减少。:)