# React 哲学

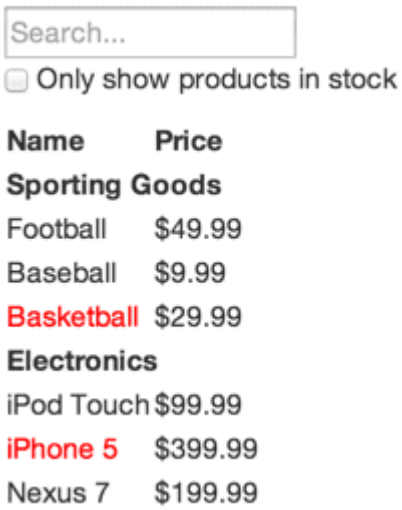
我们认为，React 是用 JavaScript 构建快速响应的大型 Web 应用程序的首选方式。它在 Facebook 和 Instagram 上表现优秀。

React 最棒的部分之一是引导我们思考如何构建一个应用。在这篇文档中，我们将会通过 React 构建一个可搜索的产品数据表格来更深刻地领会 React 哲学。

---

## 从设计稿开始

假设我们已经有了一个返回 JSON 的 API，以及设计师提供的组件设计稿。如下所示：




该 JSON API 会返回以下数据：

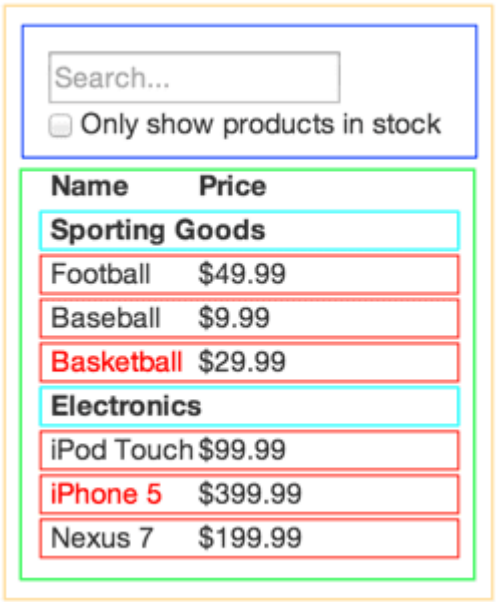
```
[
  {category: "Sporting Goods", price: "$49.99", stocked: true, name: "Football"},
  {category: "Sporting Goods", price: "$9.99", stocked: true, name: "Baseball"},
  {category: "Sporting Goods", price: "$29.99", stocked: false, name: "Basketball"},
  {category: "Electronics", price: "$99.99", stocked: true, name: "iPod Touch"},
  {category: "Electronics", price: "$399.99", stocked: false, name: "iPhone 5"},
  {category: "Electronics", price: "$199.99", stocked: true, name: "Nexus 7"}
];
```

---

首先，你需要在设计稿上用方框圈出每一个组件（包括它们的子组件），并且以合适的名称命名。如果你是和设计师一起完成此任务，那么他们可能已经做过类似的工作，所以请和他们进行交流！他们的 Photoshop 的图层名称可能最终就是你编写的 React 组件的名称！

但你如何确定应该将哪些部分划分到一个组件中呢？你可以将组件当作一种函数或者是对象来考虑，根据单一功能原则来判定组件的范围。也就是说，一个组件原则上只能负责一个功能。如果它需要负责更多的功能，这时候就应该考虑将它拆分成更小的组件。

在实践中，因为你经常是在向用户展示 JSON 数据模型，所以如果你的模型设计得恰当，UI（或者说组件结构）便会与数据模型一一对应，这是因为 UI 和数据模型都会倾向于遵守相同的*信息结构*。将 UI 分离为组件，其中每个组件需与数据模型的某部分匹配。




你会看到我们的应用中包含五个组件。我们已经将每个组件展示的数据标注为了斜体。

1. **`FilterableProductTable` (橙色):** 是整个示例应用的整体
2. **`SearchBar` (蓝色):** 接受所有的*用户输入*
3. **`ProductTable` (绿色):** 展示*数据内容*并根据*用户输入*筛选结果
4. **`ProductCategoryRow` (天蓝色):** 为每一个*产品类别*展示标题
5. **`ProductRow` (红色):** 每一行展示一个*产品*

你可能注意到，`ProductTable` 的表头（包含 “Name” 和 “Price” 的那一部分）并未单独成为一个组件。这仅仅是一种偏好选择，如何处理这一问题也一直存在争论。就这个示例而言，因为表头只起到了渲染*数据集合*的作用——这与 `ProductTable` 是一致的，所以我们仍然将其保留为 `ProductTable` 的一部分。但是，如果表头过于复杂（比如我们需为其添加排序功能），那么将它作为一个独立的 `ProductTableHeader` 组件就显得很有必要了。