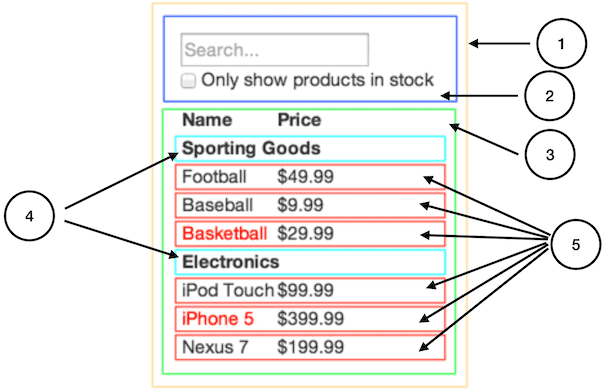
Нужно было сделать приложение, которое состоит из компонентов:



1. **FilterableProductTable (оранжевый):** контейнер, содержащий пример целиком
2. **SearchBar (синий):** поле *пользовательского ввода*
3. **ProductTable (зелёный):** отображает и фильтрует *список данных*, основанный на *пользовательском вводе*
4. **ProductCategoryRow (голубой):** наименования *категорий*
5. **ProductRow (красный):** отдельно взятый *товар*

Теперь, когда мы определили компоненты в нашем макете, давайте расположим их согласно иерархии. Компоненты, которые являются частью других компонентов, в иерархии отображаются как дочерние:

* FilterableProductTable
  + SearchBar
  + ProductTable
    - ProductCategoryRow
    - ProductRow

Чтобы построить статическое приложение, отображающее модель данных, нам нужно создать компоненты, которые используют другие компоненты и передают данные через пропсы. Пропсы — это способ передачи данных от родителя к потомку. Если вы знакомы с понятием состояния, то для статического приложения это как раз то, чего вам **использовать не нужно**. Состояние подразумевает собой данные, которые меняются со временем — интерактивность. Так как мы работаем над статическим приложением, нам этого не нужно.

Мы используем функциональных подход, а не объектный

Функциональный подход в React заключается в создании компонентов с использованием функций. Каждый компонент представляет собой функцию, которая возвращает дерево элементов React, описывающих пользовательский интерфейс. Функциональные компоненты не имеют состояния и методов жизненного цикла, что делает их более простыми и легкими в поддержке и тестировании.

В объектно-ориентированном подходе используются классы, которые наследуются от базового класса React.Component. Каждый компонент представляет собой класс, который содержит состояние и методы жизненного цикла. Объектно-ориентированный подход обеспечивает большую гибкость и масштабируемость, особенно при работе с большими приложениями.

1 компонент

Компонент FilterableProductTable импортирует типы свойств FilterableProductTablePropsType, IProduct, ProductTableType и StateType из модуля ../types/types, а также импортирует два других компонента: SearchBar и ProductTable.

Затем компонент FilterableProductTable определяет два локальных состояния, checked и text, с помощью хука useState. Первое состояние отвечает за флажок, который показывает, нужно ли отображать только товары, которые есть на складе, а второе состояние отвечает за текстовое поле, в которое пользователь может вводить текст для поиска.

Далее создается объект state, который представляет собой объект состояния для компонентов SearchBar и ProductTable. Этот объект содержит три свойства: filterText - текстовый запрос, inStockOnly - флаг, показывающий, нужно ли отображать только товары, которые есть на складе, а также две функции обратного вызова, onFilterTextChange и onInStockChange, которые будут вызываться при изменении текстового запроса и флага соответственно.

2 компонент

Компонент ProductCategoryRow импортирует тип IProduct из модуля ../types/types.

Затем определяется тип свойств ProductCategoryRowType, который содержит только одно свойство category - название категории продукта.

Далее компонент ProductCategoryRow принимает свойства типа ProductCategoryRowType, деструктурирует category и возвращает разметку JSX, которая представляет собой строку таблицы с названием категории продукта.

3 компонент

Компонент ProductRow импортирует тип IProduct из модуля ../types/types.

Затем определяется тип свойств ProductRowType, который содержит только одно свойство product типа IProduct - данные о продукте.

Далее компонент ProductRow принимает свойства типа ProductRowType, деструктурирует product и создает переменную name, которая содержит название продукта. Если продукт есть на складе, то name будет равно product.name, иначе будет возвращаться JSX элемент <span>, внутри которого имя продукта окрашивается красным цветом.

Затем компонент возвращает разметку JSX, представляющую собой строку таблицы с двумя ячейками. Первая ячейка содержит переменную name, а вторая ячейка содержит цену продукта, которая берется из свойства product.price.

4 компонент

Этот код представляет компонент ProductTable, который отображает таблицу продуктов, соответствующих заданным критериям поиска. Он принимает входные параметры через объект props, который содержит список продуктов и объект state, который содержит текущее состояние компонента FilterableProductTable.

Компонент проходит по списку продуктов и проверяет, соответствует ли каждый продукт критериям фильтрации, заданным в state. Если да, то он добавляет строку таблицы для каждого продукта. Если продукт относится к новой категории, то добавляется строка с названием категории.

Компонент ProductTable отображает таблицу с двумя столбцами: Name и Price, а также строками, сгруппированными по категориям продуктов. Каждая строка содержит имя продукта и его цену. В случае, если продукт не в наличии, его имя выделяется красным цветом.

5 компонент

Компонент SearchBar принимает свойства типа SearchBarType, который содержит одно свойство state типа StateType. StateType является пользовательским типом, который содержит свойства для фильтрации продуктов и функции обратного вызова для обновления состояния компонента родителя.

Компонент SearchBar возвращает разметку JSX, содержащую элемент <form> с текстовым полем для ввода текста поиска и флажком для фильтрации продуктов, которые есть на складе.

Значение текстового поля устанавливается на значение свойства state.filterText, а при изменении текста, вызывается функция обратного вызова state.onFilterTextChange, которая обновляет состояние родительского компонента.

Значение флажка устанавливается на значение свойства state.inStockOnly, а при изменении состояния флажка, вызывается функция обратного вызова state.onInStockChange, которая также обновляет состояние родительского компонента.

Хуки (hooks) в React - это функции, которые позволяют использовать состояние и другие возможности React в функциональных компонентах.

Некоторые из наиболее часто используемых хуков в React:

useState - позволяет добавить состояние в функциональный компонент. Этот хук возвращает массив из двух элементов: текущего значения состояния и функции для обновления этого состояния.