**SOLID**

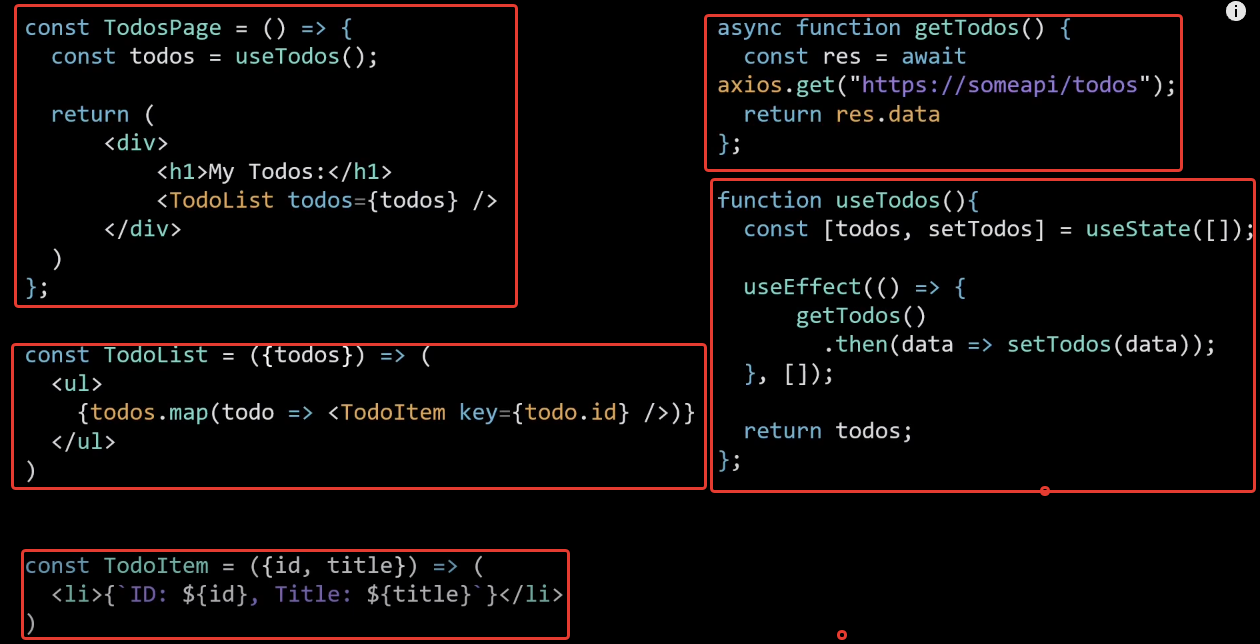
1. **SRP**
   1. **Код**

UI-компонент должен отвечать за решение одной отведенной ему задачи.

*Проблема*: имеется компонент, в котором всё перемешано (запрос на получение данных, обработка запроса, маппинг полученных данных):

****

*Решение*: такой компонент следует разбить на несколько компонентов, каждый из которых будет решать отведенную малую часть одной большой задачи:



* 1. **Архитектура**

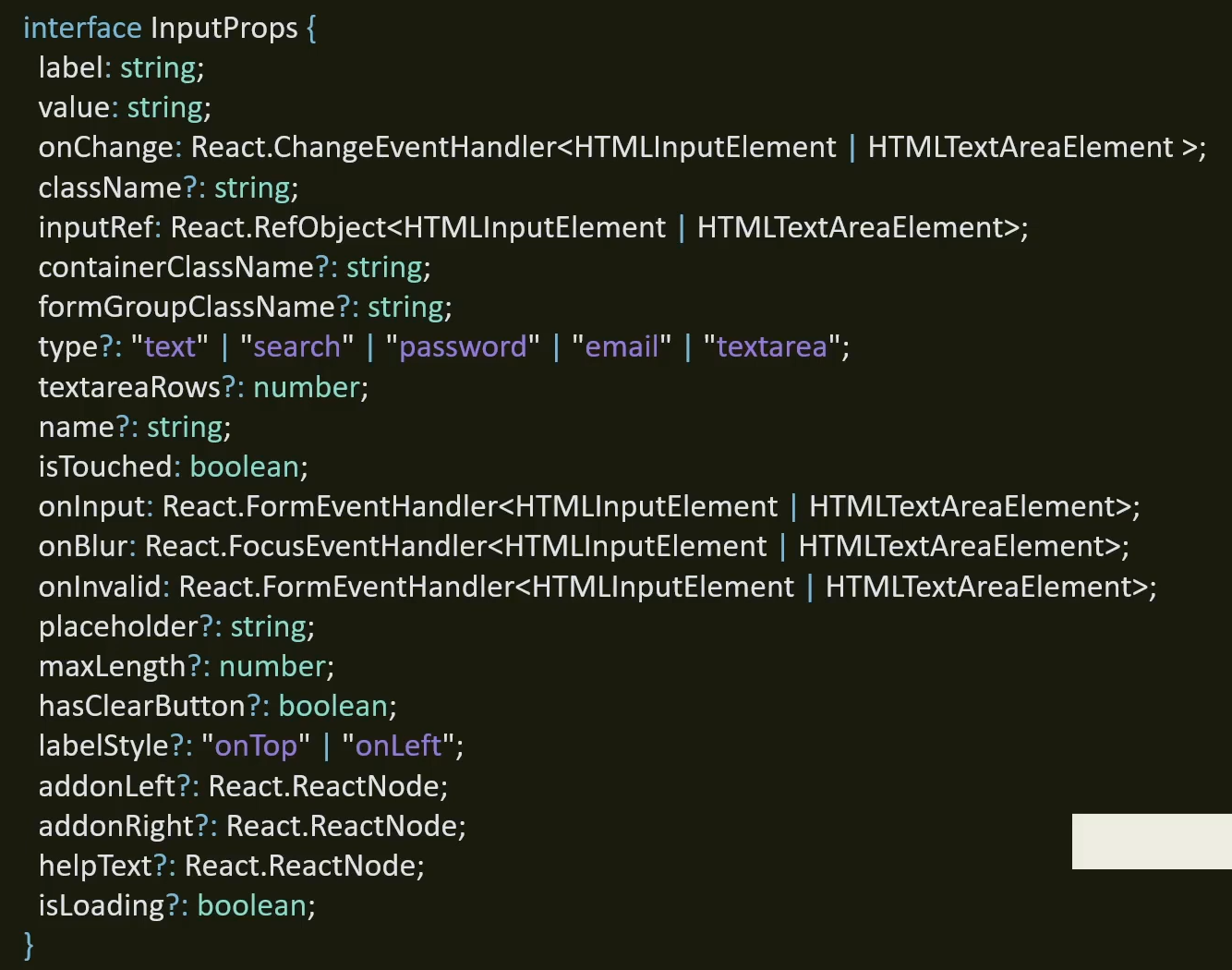
Модуль должен отвечать за одного и только за одного актора (под актором понимается группа пользователей и/или иных заинтересованных лиц).

*Проблема*: внесённые в функциональность модуля изменения по требования актора А отразились на функциональности, необходимой актору Б.

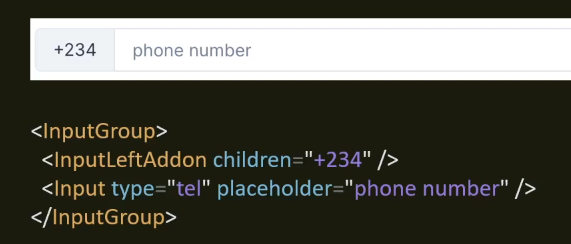
*Решение*: следует проверить модуль на соответствие SRP.

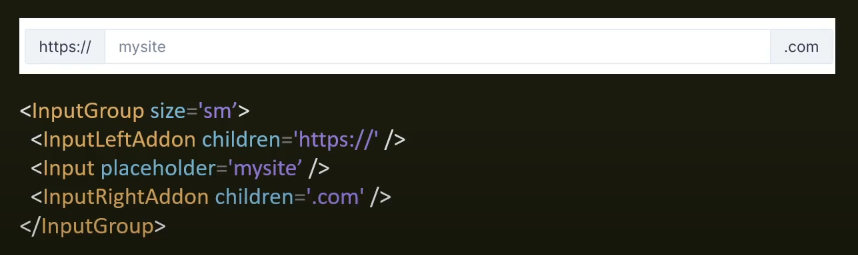
1. **OCP**
   1. **Код**

Вносимые изменения должны осуществляться **без изменения** уже существующей отлаженной и протестированной функциональности (в противном случае необходимо по окончанию работ проводить регрессионное тестирование).

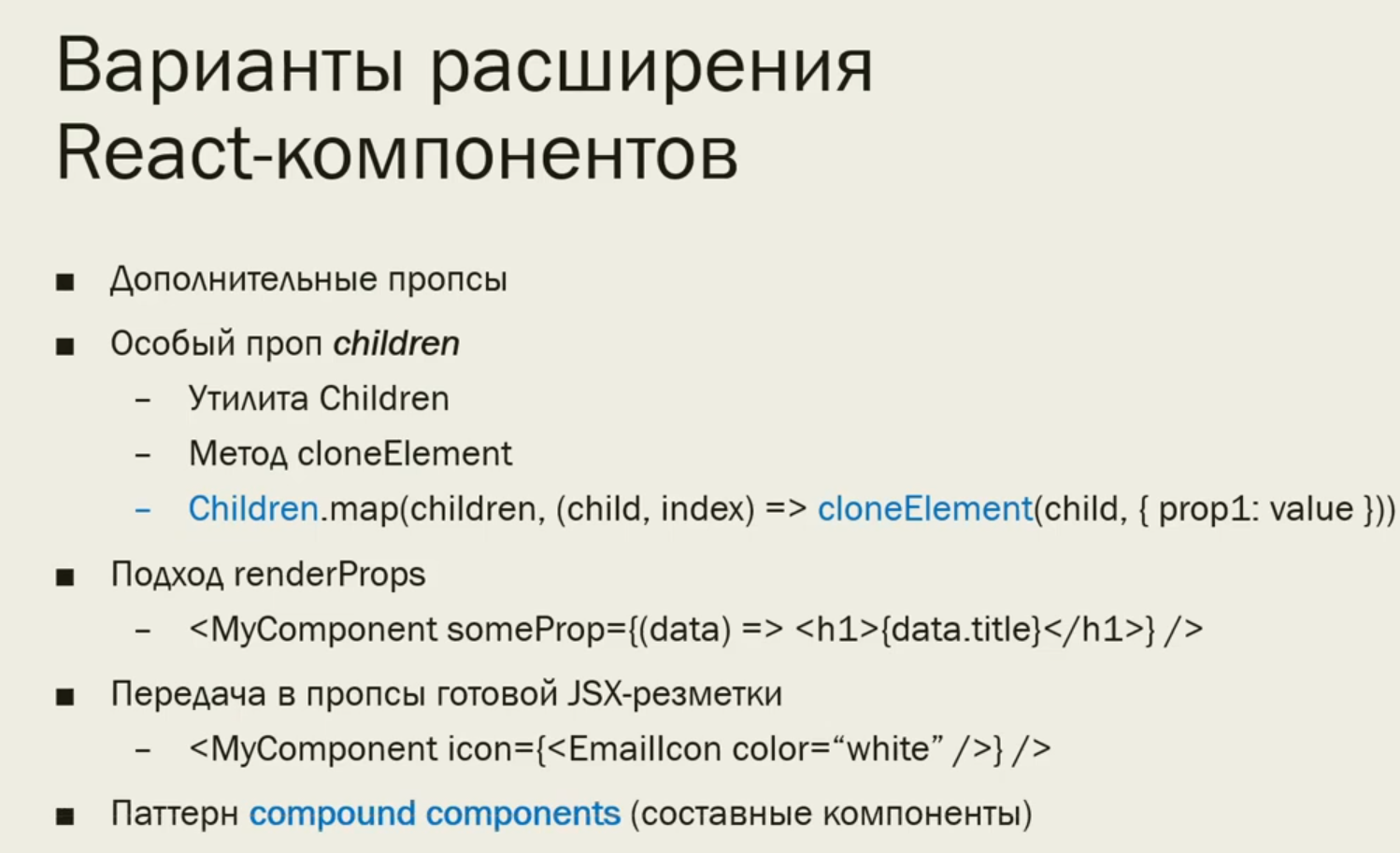
*Проблема*: имеется кастомный TextField-компонент с кучей пропсов, и возникает ситуация, когда при добавлении фич приходится править другой код:  


*Решение*: исходный компонент декомпозировать по SRP и расширение функциональности компонента осуществлять декомпозированными компонентами:







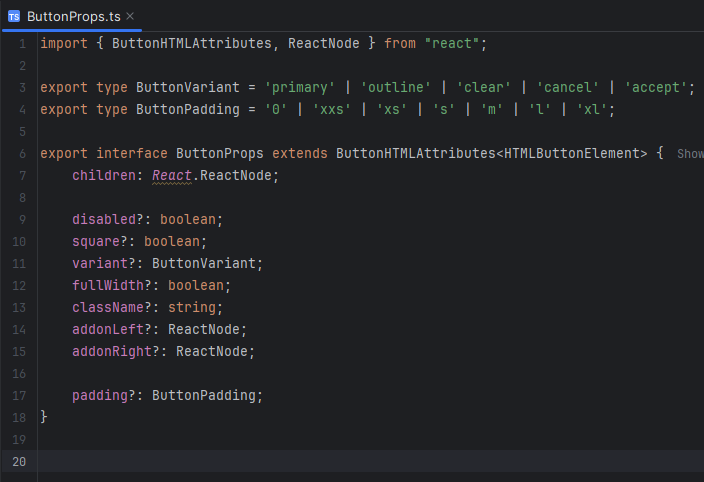


* 1. **Архитектура**

Пример – типовая архитектура FSD, в которой внесение изменений осуществляется посредством расширения: добавление слайсов в слои; добавление сегментов в слайсы.

1. **LSP**
   1. **Код**

Создание кастомного компонента, например, кнопки (под капотом <button></button>, типа React.ButtonHTMLAttributes<HTMLButtonElement>) и определение его пропсов, посредством интерфейса, например, IButtonProps, который расширяет тип базовой кнопки:



Таким образом можно сделать множество кнопок (табы, с загрузкой и т.д.)

* 1. **Архитектура**

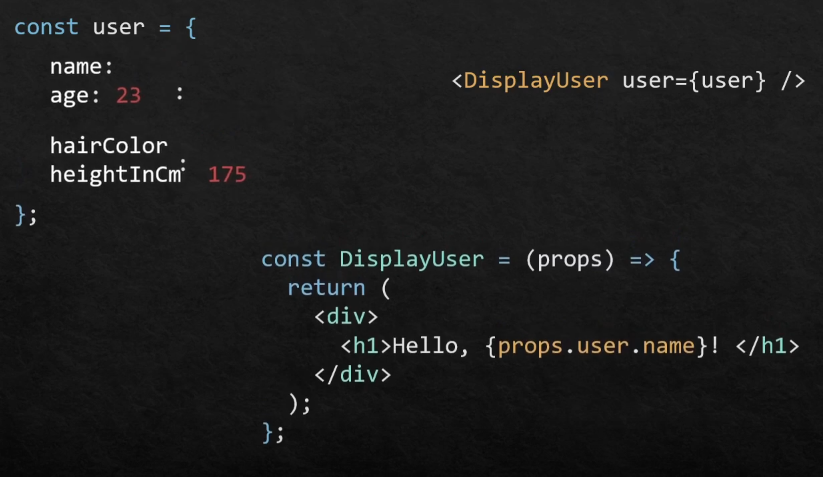
**???**

1. **ISP**

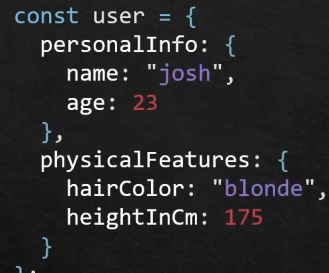
React-компоненты не должны зависеть от пропсов, которые они не используют; то есть для компонента должен быть определен такой набор пропсов, чтобы при изменении логики в родительском компоненте не пришлось рефакторить дочерний.

* 1. **Код**

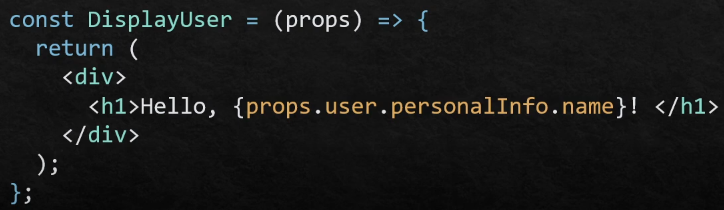
Имеется:



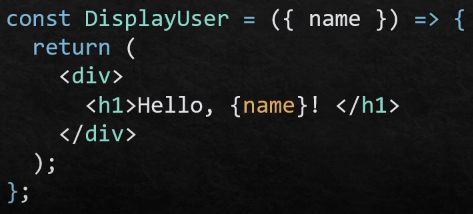
Структуру пропа поменяли, вследствие будет ошибка в компоненте DisplayUser и его придётся рефакторить:



Отрефакторенный DisplayUser (нарушен ISP):



Не нарушен ISP:



* 1. **Архитектура**

**???**

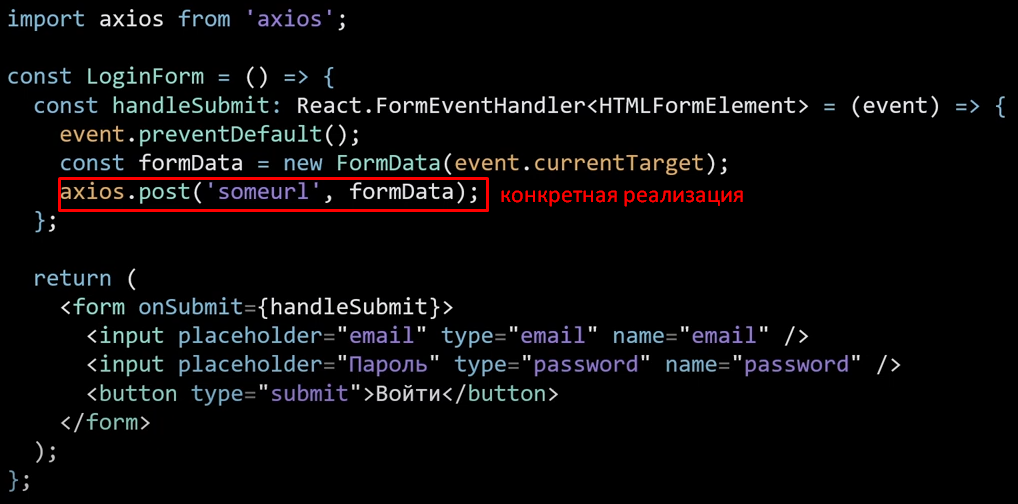
1. **DIP**

Модули верхних уровней не должны зависеть от модулей нижних уровней, оба типа модулей должны зависеть от абстракций.

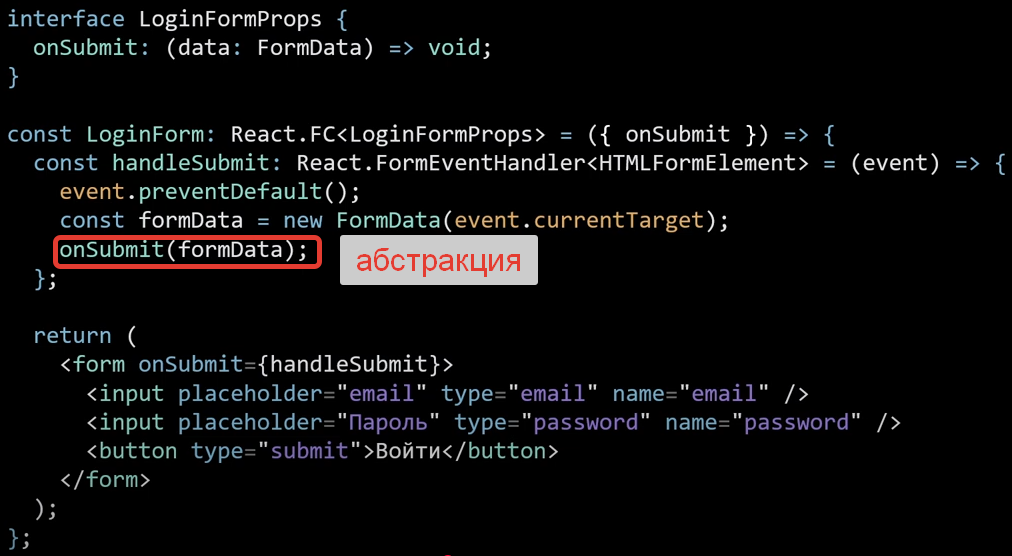
Абстракции не должны зависеть от деталей, детали должны зависеть от абстракций.

* 1. **Код**

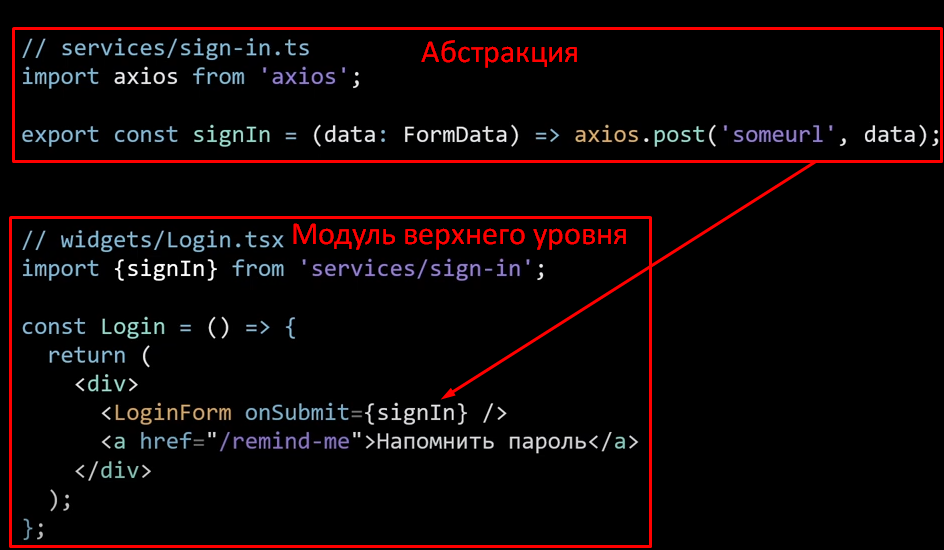
Нарушен DIP:



Не нарушен DIP

****

Пример зависимости модуля верхнего уровня от абстракции:



* 1. **Архитектура**

**???**