



#ШПАРГАЛОЧКИ

# СОЗДАНИЕ САЙТОВ FRONT-END РАЗРАБОТКА

## Дополнительный уровень

Материалы подготовлены отделом методической разработки

Больше полезных материалов и общения в нашем комьюнити  
в Telegram: [https://t.me/hw\\_school](https://t.me/hw_school)





# Знакомство с props





# Структура приложения

Когда компонента растёт и становится громоздкой, её можно разбить на отдельные компоненты. Это снова принцип разделения ответственности.

Например, если в разделе Profile мы публикуем посты, то есть смысл создать в папке profile папку posts с компонентой Posts.js (она будет отвечать за отображение всех постов). А внутри posts - ещё одну папку post с компонентой Post.js, она уже будет отображать один пост.

Теперь в Posts.js нужно будет импортировать и подключить Post.js, а в Profile.js - Posts.js.



# props

**Props (properties)** - это объект, который можно использовать для передачи данных между компонентами.

Например, чтобы сделать текст постов различающимся, в теге **<Post/>** компоненты Posts.js можно использовать **атрибут** с текстом:

```
<Post message="Hello!"/>
```

А теперь в Post.js функция **Post** должна принимать **props** - тогда в ней можно будет использовать его значения:

```
function Post(props) {  
  return (  
    ...  
    <p> {props.message} </p>  
    ...  
  )  
}
```



# props

Важно понимать, что **props** хранит в себе пары **ключ-значение**. И при создании атрибута:

```
<Post message="Hello!"/>
```

В него запишется значение атрибута с ключом, совпадающим с именем атрибута:

```
{message: "Hello!"}
```

А чтобы получить значение из объекта, нужно обратиться к нему по ключу:

```
props.message // "Hello!"
```