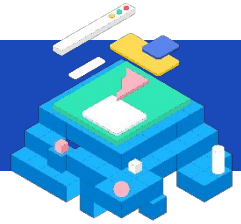
ДЕТСКАЯ ОНЛАЙН-ШКОЛА ПРОГРАММИРОВАНИЯ
HELLO WORLD

#ШПАРГАЛОЧКИ

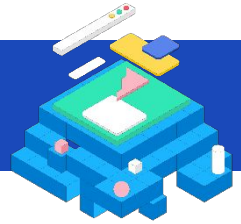
СОЗДАНИЕ САЙТОВ FRONT-END РАЗРАБОТКА

Материалы подготовлены отделом методической разработки

Дополнительный уровень



Перенос данных в index.js

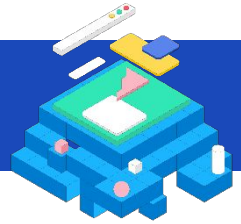


Перенос данных

Следующий шаг по разделению данных и **UI** - передача данных именно через **props** и вынос данных в **index.js**.

На примере компоненты Posts заменим использование массива **postsData** на **props** (postsData никуда не исчезнет - просто перенесется в index.js и перейдет **в** props):

{props.postsData.map((e) => <Post message={e.text} id={e.id}>)}



Перенос данных

А в компоненте Profile уже нужно прописать сохранение postData в props:

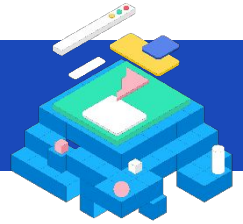
<Posts postsDate={props.postsData} />

Важно: какое имя атрибута используется здесь, такое же будет использоваться и в Posts при извлечении данных из props. А если изменить имя атрибута в Profile:

<Posts qwerty={props.postsData} />

То изменения потребуются и в Posts:

{props.qwerty.map((e) => <Post message={e.text} id={e.id}>)}



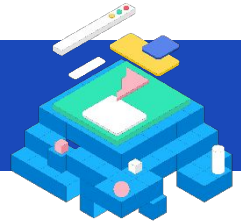
render

Остается только поправить **App.js**. Сейчас компонента Route вызывает другие компоненты как функции, но поскольку теперь эти функции принимают какие-то значения, код нужно немного изменить. Вместо атрибута component нужно использовать **render**, а ему передать анонимную функцию, возвращающую нужную компоненту:


```
function App(props) {

    ...

    <Route path="/profile" render={() => <Profile postsData={props.postsData} />}>
```



render

В index.js:

postsData [

{text: “Hello”, id: 1},

...

]

ReactDOM.render {

...

<App postsData={postsData}>