React Complete Course Schwarzmuller

1. Getting started

2. JavaScript Refresher

**Classes**

Для чего нужен super() ?

Например, есть два класса, у обоих есть constructor. Он служит для того, чтобы у экземпляра появились свойства класса. Методы прописываются ниже в теле класса.

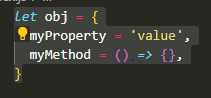
Если мы расширяем первый класс вторым, то, чтобы наследовался конструктор родительского класса, соответственно чтобы появились свойства у экземпляра объекта от первого (наследуемого) класса, используем слово super().

*Если кратко – используем super(), чтобы конструктор родительского класса передался дочернему.*

Именно поэтому в классовых реакт компонентах мы пишем каждый раз super у дочерних компонент.

**Classes, Properties, Methods**

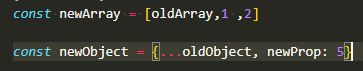
Современный синтаксис по написанию свойств и методов объектов:



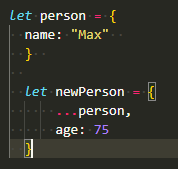
**Spread and rest operators**

Оператор …**spread**:

Копируем данные из **массива**:



Копируем данные из **объекта**:



**Destructuring**

Оператор …**spread**:



2. React Basics &

Working with components

Самая важная концепция в реакте это **компоненты**. В конечном итоге все состоит из них. Любой компонент в Реакте это простая ФУНКЦИЯ!

*Компонент:*

*- может быть переиспользован много раз (DRY)*

*- должен отвечать за что-то одно. Не надо реализовывать много функционала в одном компоненте. (SOLID)*

*Как построен компонент?*

Компонент состоит из: HTML, JS, CSS

Компонент должен быть написан в **декларативном стиле.** Тогда как приложение на обычном JS будет написано по большей части в императивном стиле.

1. The Concept of “Composition”

{props.children}

**Композиция – построение приложения по схеме на картинке. Т.е. каждый дочерний компонент явно указывается в родительском.**

Что, если мы хотим создать компонент – обертку? Для этого применяется **декомпозиция.** Например:

import './Card.css'

//now card is a CONTAINER component

*function* Card (*props*) {

  // так можно применять несколько стилей: стиль обертки и стиль родителя.

  // в строке ПРОБЕЛ!!!

*const* classes = 'card ' + *props*.className

  return <div className={classes}>{*props*.children}</div>

}

export default Card

**И обернем в него 2 компонента:**

import ExpenseItem from './ExpenseItem'

import Card from './Card'

import './Expenses.css'

*function* Expenses ({ *items* }) {

  return (

    <*Card* className={'expenses'}>

      <*ExpenseItem*

        title={*items*[0].title}

        amount={*items*[0].amount}

        date={*items*[0].date}

      />

      <*ExpenseItem*

        title={*items*[1].title}

        amount={*items*[1].amount}

        date={*items*[1].date}

      />

      <*ExpenseItem*

        title={*items*[2].title}

        amount={*items*[2].amount}

        date={*items*[2].date}

      />

      <*ExpenseItem*

        title={*items*[3].title}

        amount={*items*[3].amount}

        date={*items*[3].date}

      />

    </*Card*>

  )

}

**И**

import ExpenseDate from './ExpenseDate'

import './ExpenseItem.css'

import Card from './Card'

*function* ExpenseItem (*props*) {

  return (

    <*Card* className='expense-item'>

      <*ExpenseDate* date={*props*.date} />

      <div className='expense-item\_\_description'>

        <h2>{*props*.title}</h2>

      </div>

      <div className='expense-item\_\_price'>${*props*.amount}</div>

    </*Card*>

  )

}

export default ExpenseItem

**Зачем:**

* **Reusable component с набором стилей и без дочерних компонентов**

1. React State and Events

Если нужно обновлять **стейт с использованием предыдущего состояния** (например, **если** **стейт это объект**), то в функцию, сетающую стейт передать не просто значение, как обычно, а колл-бек:

const titleChangeHandler = (*event*) => {  
 setUserInput((*prevState*)=>({...*prevState*, enteredTitle: *event*.target.value}))  
};

Обычно мы делаем так:

const amountChangeHandler = (*event*) => {  
 setUserInput(*event*.target.value)  
};