



#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





Знакомство с ReactJS





ReactJS

ReactJS - это библиотека JavaScript. С его помощью можно быстрее и легче разрабатывать сложные приложения.

Отличительная особенность приложений на **React** - разбиение кода на отдельные компоненты, которые легко комбинировать и использовать повторно.



NodeJS

NodeJS - это специальное приложение, с помощью которого можно запускать JavaScript-код (включая **React**).

Чаще всего **NodeJS** используется для выполнения кода, написанного для веб-сервера.



Создание приложения React

Чтобы создать **React**-приложение, нужно открыть консоль (**cmd** в поиске), зайти в нужную папку:

```
cd Desktop\react
```

И выполнить следующую команду:

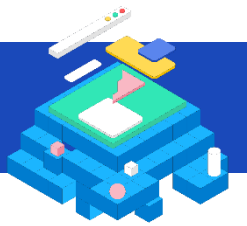
```
npx create-react-app app-name
```

Где **app-name** - имя приложения, оно может быть любым (на английском языке и без пробелов).

После установки всех необходимых пакетов нужно зайти в созданную папку и запустить сервер:

```
cd app-name
```

```
npm start
```



SPA

SPA (Single Page Application) - одностраничное приложение.

Именно такие создаются в **React**.

В одностраничном приложении все происходит на одной html-странице, просто ее содержимое будет меняться с помощью JavaScript кода.



JSX

JSX - это расширение JavaScript, позволяющее совмещать html и js-код.

Важно: чтобы Рес-приложения работали правильно, все одиночные теги (у которых нет закрывающего тега) нужно дополнить слэшем, вот так:

```

```

Иначе будет ошибка.



React DOM

Одна из причин быстроты приложений на React - он работает не с реальным DOM, а с виртуальным.

Это значит, что при внесении изменений в проект React не обновляет реальную страницу сразу. Вместо этого он редактирует сохраненную в памяти “копию”, сравнивает ее с предыдущей версией и только после этого меняет реальный DOM там, где это необходимо.

Такое сокращение работы с реальным DOM сильно повышает производительность и скорость работы приложения.



Структура приложения

Код, отвечающий за работу приложения, находится в папке **src**.

index.js - этот файл отвечает за отображение приложения. Этот код значит, что мы отрисовываем компоненту **App**:

```
ReactDOM.render(  
  <React.StrictMode>  
    <App />  
  </React.StrictMode>,  
  document.getElementById('root')  
);
```

App.js - а здесь хранится **jsx-код** - смесь **html** и **javascript**.



Компонента

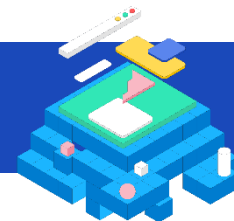
Компонента - это функция, которая возвращает **jsx-код** (именно он будет отрисован на странице). Важно, что одна компонента может вернуть только один элемент, это значит, что все код внутри нее должен быть обернут в общий тег (например, `div`).

Чтобы в компоненте можно было использовать React, его нужно импортировать:

`import React from 'react'`

Еще один важный момент - атрибут **`class`** в **jsx-коде** нужно заменить на **`className`**. Это связано с тем, что в JavaScript тоже есть классы.

`<div className="App"></div>`



Компонента

```
function App() {  
  return (  
    <div className="App">  
      <h1>Hello World!</h1>  
    </div>  
  );  
}
```



import/export

Чтобы компоненты можно было использовать, их нужно экспортировать из файла, где они созданы (App.js), и импортировать там, где они должны быть отрисованы (index.js).

Если компонента одна, то достаточно написать после ее создания:

```
export default App // App - имя компоненты
```

А в index.js прописать импорт:

```
import App from “./App”
```

Если же в одном файле создано несколько компонентов, то их экспорт будет другим:

```
export function Menu() {
```

...