React является одним из самых популярных JavaScript фреймворков. Фреймворк представляет собой заготовку, которая помогает вам писать код.

***Как он это делает*: фреймворк заставляет вас следовать определенным соглашениям по написанию кода, в результате этого все ваши проекты становится проще разрабатывать, а самое главное - проще поддерживать в дальнейшем.**

***Кроме того*: так как все проекты на фреймворке следуют одному и тому же соглашению, то любой другой человек сможет с легкостью разобраться в вашем коде и допилить что-либо в сделанном вами проекте.**

Фреймворки типа React используются для так называемых SPA (*single page application* - одностраничное приложение). Одностраничное приложение - это сайт, все действия в котором происходят на одной странице, без ее полного обновления.

Примером SPA может служить социальная сеть, работа с почтой и так далее.

Перед изучением React

Перед чтением данного учебника изучите уроки по работе с контекстом и учебник по ООП на классах в стиле ES6 в JavaScript. Это обязательно, без этого вы ничего не поймете.

Введение в React

Перед чтением также советую посмотреть вебинар по React, который является введением в этот фреймворк. Вебинар вы найдете по следующим ссылкам: 1, 2, 3, 4, 5, 6, 7, 8.

Фреймворки для модульного тестирования JavaScript

React работает на ES6, поддержки которого пока во всех браузерах нет. Поэтому для того, чтобы ваш код заработал, - следует использовать специальные программы-транспойлеры, которые преобразуют ваш код ES6 в код ES5.

Самым популярным транспойлером является Babel. Им мы и будем пользоваться.

На самом деле реальные проекты на React требуют некоторого знания nodejs и npm. Мы пока не будем с этим заморачиваться, потому что мне хотелось бы, чтобы вы сразу начали писать код.

Для самого простого запуска React нужно подключить такую кучку скриптов:

```
<script src="https://npmcdn.com/react@15.3.0/dist/react.js"></script>
<script src="https://npmcdn.com/react-dom@15.3.0/dist/react-dom.js"></script>
<script src="https://npmcdn.com/babel-core@5.8.38/browser.min.js"></script>
<script src="https://npmcdn.com/jquery@3.1.0/dist/jquery.min.js"></script>
<script src="https://npmcdn.com/remarkable@1.6.2/dist/remarkable.min.js"></script>
```

На *продакшене* (так называют рабочий сайт) так делать не стоит (так как здесь компиляция кода из новой версии в старую происходит в браузере, а это очень медленно). Мы так делаем для простоты - чтобы вы сразу начали писать код. Попозже мы разберем более сложный способ.

После подключения этих скриптов мы можем выполнять наш ES6 код в теге script с типом text/babel (это важно: не text/javascript, как мы привыкли):

```
<script type="text/babel">
    тут код
<script>
```

Если вы ошибетесь и напишите НЕ text/babel - все просто не заработает. Будьте внимательны - это распространенная ошибка новичков.

Введение в React

При написании кода в React принято использовать const вместо var. Напоминаю, что const - это константы, то есть переменные, тип и значение которых нельзя поменять. Последнее не относится к *массивам и объектам* - в этом случае нельзя будет сменить тип переменной, а изменять/добавлять/удалять элементы массива - можно будет.

Константы как раз-таки удобны при работе с массивами и объектами, их использование не дает сменить тип переменной - например, не дает затереть массив и записать туда что-нибудь другое, например, строку.

Будьте внимательны - при использовании const вас могут ждать неожиданные проблемы.

Например, проблемы могут возникнуть при работе с if, так как переменные с const, как и let, видны только внутри фигурных скобок (не так, как var) и не видны снаружи.

Смотрите пример:

```
if (тут условие) {
  const name = 'Коля';
    } else {
  const name = 'Вася';
        }

alert(name); //будет undefined
```

Если попробовать определить константу над ифом - это также приведет к ошибке, так как при определении константы обязательно должно быть указано ее значение:

```
const name; //выдаст ошибку

if (тут условие) {
    name = 'Коля';
    } else {
    name = 'Вася';
        }

alert(name);
```

Для того, чтобы поправить проблему вместо const напишем let, причем let обязательно над ифом, иначе из-за области видимости переменные не будут видны снаружи:

```
let name; //определяем переменную через let снаружи ифа


if (тут условие) {
    name = 'Коля';
    } else {
    name = 'Вася';
        }

alert(name); //все работает!
```

Компонентный подход

Пусть у нас есть сайт. На этом сайте мы можем выделить некоторые блоки: хедер, контент, сайдбар, футер и так далее. Каждый блок можно разделить на более мелкие подблоки. К примеру в хедере обычно можно выделить логотип, менюшку, блок контактов и так далее.

В React каждый такой блок называется компонентом. Каждый компонент может содержать в себе более мелкие компоненты, те в свою очередь еще более мелкие и так далее.

Каждому компоненту в React соответствует ООП класс:

```
class ИмяКласса {

}
```

Чтобы быть компонентом, этот класс обязательно должен наследовать от класса React.Component:

```
class ИмяКласса extends React.Component {
```

```
}
```

Класс React.Component встроен в React и доступен после подключения фреймворка. Просто наследуете от него - и все, остальное не должно вас волновать.

Подключение компонентов

Пусть у нас есть, к примеру, компонент с названием Test:

```
class Test extends React.Component {


}
```

Каждому компоненту соответствует свой тег. С помощью этого тега можно вставить на страницу результат работы компонента.

Тег имеет такое же название, как и имя класса компонента. То есть: если, к примеру, у нас компонент Test, то ему соответствует тег <Test />.

Обратите внимание на то, что этот тег пишется с большой буквы и сразу же закрывается с помощью обратного слеша (закрывать обязательно!).

Пусть у нас есть тег header:

```
<header></header>
```

Давайте в этот header вставим результат работы нашего компонента Test:

```
<header><Test /></header>
```

Результат работы компонента

Итак, мы выяснили, что там, где мы напишем тег <Test />, выведется результат работы нашего компонента, заданного классом Test.

Напоминаю, что сам компонент представляет собой класс. Внутри этого класса вы можете писать любые методы. Но есть один метод, который имеет особое значение: его название render(). То, что вернет метод render() и будет результатом работы любого компонента:

```
class Test extends React.Component {

render() {

return тут результат работы компонента;

}

}
```

Язык JSX

В React встроен специальный язык под названием JSX. Суть этого языка такая: *HTML теги являются корректным JavaScript кодом* и мы можем просто писать их прямо в скрипте, не беря в кавычки.

Например, запишем в переменную text кусочек HTML:

```
const text = <div>текст</div>;
```

Вот он прямо так и записывается - без кавычек. Подробнее об этом мы будем говорить в следующем уроке.

JSX в компонентах

Давайте сделаем так, чтобы результатом работы компонента Test было текст :

```
class Test extends React.Component {

render() {

return <div>текст</div>;

}

}
```

В итоге вместо этого кода:

```
<header><Test /></header>
```

Получим следующее:

```
<header><div>текст</div></header>
```

Основной компонент

Мы уже выяснили, что на странице есть компоненты, в них вложенные компоненты и так далее. Однако существует самый главный компонент верхнего уровня - в нем лежат все остальные компоненты.

Обычно этот компонент называют App (но не обязательно):

```
class App extends React.Component {


}
```

Чтобы вставить этот компонент на страницу, мало написать тег <App />. В этом случае необходимо указать элемент DOM, вовнутрь которого вставится результат работы компонента App.

Пусть у нас есть блок #content:

```
<div id="content"></div>
```

В этом случае следующий код заставит вставится результат работы компонента App в блок #content:

```
ReactDOM.render(
    <App />,
  document.getElementById('content')
);
```

Давайте соберем все вместе:

```
<div id="content"></div>

class App extends React.Component {
    render() {
  return <div>текст</div>;
    }
}
```

```
        ReactDOM.render(
            <App />,
        document.getElementById('content')
);
```

Результатом этого кода будет следующее:

```
<div id="content">
    <div>текст</div>
</div>
```