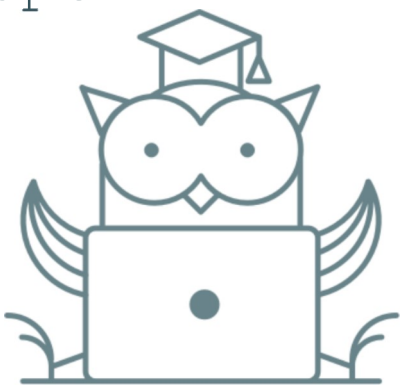
ОНЛАЙН-ОБРАЗОВАНИЕ

```
let lesson = {
  id:       '08'
  themes:   ['React Basics', 'JSX'],
  date:     '27.03.2018',
  teacher: {
    name:       'Юрий Дворжецкий',
    position: 'Lead Developer’
  }
};
```



Как меня слышно && видно?

Если нет – напишите, если слышите – смайлик в чат.

OTUS

Скажите пару слов о React

1. Что Вы знаете про React?
2. Пробовали?
3. Понравилось?
4. Подсели? ☺

О вебинаре:

Что сможем делать после вебинара?

- Настроить себе окружение для работы с React и использовать его.
- Понимать и применять синтаксис JSX
- Создавать простые приложения на React

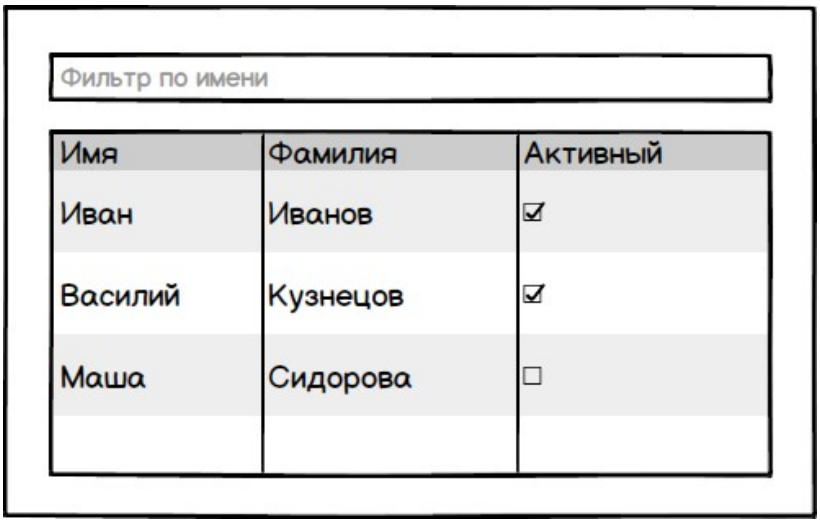
React

Компонентный подход

1. Компонент – логика + поведение.
2. Любая связная часть – это компонент
3. Все сейчас поддерживают компонентный подход к разработке UI приложения.
4. Всё приложение – дерево компонент.
5. Задача программиста – разработать удобные компоненты, желательно переиспользуемые и менее связные.

Компонентный подход (упражнение)

1. Назовите в чат компоненты, которые Вы выделили бы в следующем макете.
2. Поспорьте с коллегой.



OTUS

Как Вы думаете?

1. Чем отличается framework от библиотеки?
2. Что хорошего в использовании фреймворка?
3. Что хорошего в использовании библиотеки?

Классические приложения (jQuery)

1. Большая HTML с данными - для каждой страницы (генерируется сервером – PHP/JSP/ASP)
2. JS - подключается в HTML, анимирует HTML с данными
3. CSS - подключается в HTML, задаёт стили отображения

Шаблонные фреймворки

1. Небольшая статическая HTML
2. Данные загружаются с сервера отдельно
3. HTML шаблоны – статические фрагменты компонентов HTML, отображающие данные
4. JS компоненты - подключаются в HTML, добавляют поведение к HTML шаблонам
5. CSS – подключается в главной HTML

Фреймворки на базе JSX (React)

1. Минимальная HTML, только чтобы подключить JS и CSS (SPA – Single Page Application)
2. Данные загружаются с сервера отдельно
3. JSX компоненты – содержат и JS код и подобие HTML разметки для каждого компонента
4. CSS – подключается в HTML

React

1. Входит в ТОП-5 фреймворков/библиотек (Hackernoon 2017)
2. Библиотека, а не фреймворк.
3. Только для создания UI, вы не ограничены выбором для других частей (в терминах Model-View-Whatever – это V).
4. Component-based
5. Вся мощь на client-side но есть server-side.

OTUS

React

1. Использует JSX (“XML” в JS)
2. Virtual DOM и Reconcilation (может быстрее-быстрого)
3. Flux/Redux – модный паттерн/библиотека/реализация для хранения состояния.
4. Огроооооомнейшая экосистема из различных утилит, билиотек, плюшек и, конечно, компонент.
5. Служит для создания SPA

Пример кода на React (что нового?)

```
import React from 'react';
import ReactDOM from 'react-dom';

ReactDOM.render(
    <h1>Hello, world!</h1>,
    document.getElementById('root')
);
```

Пример кода на React (что нового?)

```
class HelloMessage extends React.Component {
    render() {
        return (
            <div>
                Hello {this.props.name}
            </div>
        );
    }
}

ReactDOM.render(
    <HelloMessage name="Taylor"/>,  mountNode
);
```

Create-react-app

1. Утилита для создания **React-**приложений
2. Глубоко запрятан webpack
3. Глубоко запрятан webpack dev-сервер
4. Ещё глубже тестовый фреймворк
5. Ещё глубже babel
6. Но зато очень просто создаётся приложение
7. eject – достать из глубин

OTUS

Create-react-app (пробуем)

npm install -g create-react-app

create-react-app my-app

cd my-app

npm start

npx create-react-app my-app

cd my-app

npm start

OTUS

Create-react-app (пробуем)

npm run eject

Webpack

- Сборщик
- На настоящее время самый крутой
- И самый быстрый
- Умеет tree-shaking, bundling, livereload и прочие страшные слова
- Загружает файлы с помощью loader-ов
- Позволяет писать импорт CSS в JS файлах

Babel

- Транспайлер
- Преобразует код на диалекте JS в код на JS
- Преобразует JSX в JS (посмотрим!)
- Можно подключать различные расширения JS

Вопросы?

Весь шик JSX

```
import React, { Component } from 'react';
import Logo from './Logo';
import Menu from './Menu';

export default class Header extends Component {
  render() {
    return (
      <div className="header">
        <Logo className="header__logo"/>
        <Menu className="header__menu"/>
      </div>
    );
  }
}
```

OTUS

JSX

```
<Menu className="header__menu"/> - строчные атрибуты

<User logged={true} /> - булевы атрибуты
<User logged={isLogged()} /> - произвольные атрибуты

/* <User /> */ - комментарии

<div style={{color:'red'}} /> - стили (объекты)

return flag && <div />; - это JS-объекты!
```

JSX

```
<User username="ydvorzhetskiy"/>
```

```
export default class User extends Component {
    render() {
        return (
            <div>
                My name is {this.props.username}
            </div>
        );
    }
}
```

JSX

```
<Page>
    <div/>
    <div/>
    <span/>
</Page>

export default class Page extends Component {
    render() {
        return (
            <div className="page">
                {this.props.children}
            </div>
        );
    }
}
```

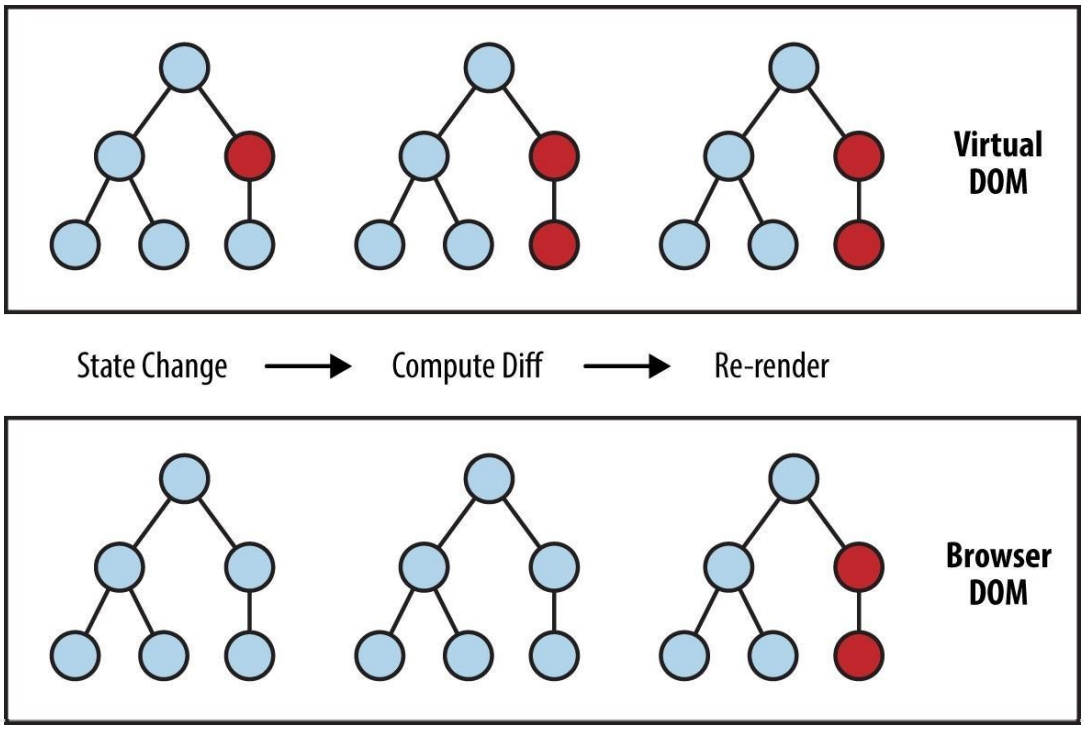
JSX

```
class Hello extends Component {
    render() {
        const {username} = this.props;
        return (
            <div>Hello, {username}</div>
        )
    }
}

// Эквивалентная запись

const Hello = ({username}) => (
    <div>Hello, {username}</div>
);
```

render()

- Чистая функция
- Из метода render() React компонента можно возвращать только один JSX элемент.
- Если хотите вернуть несколько элементов, то необходимо обернуть в ещё один элемент, например, ***.
- Помните, что JSX преобразуется в JS. Иногда непросто увидеть, почему разметка неправильная.
- Все открытые тэги должны закрываться или быть самозакрывающимися.

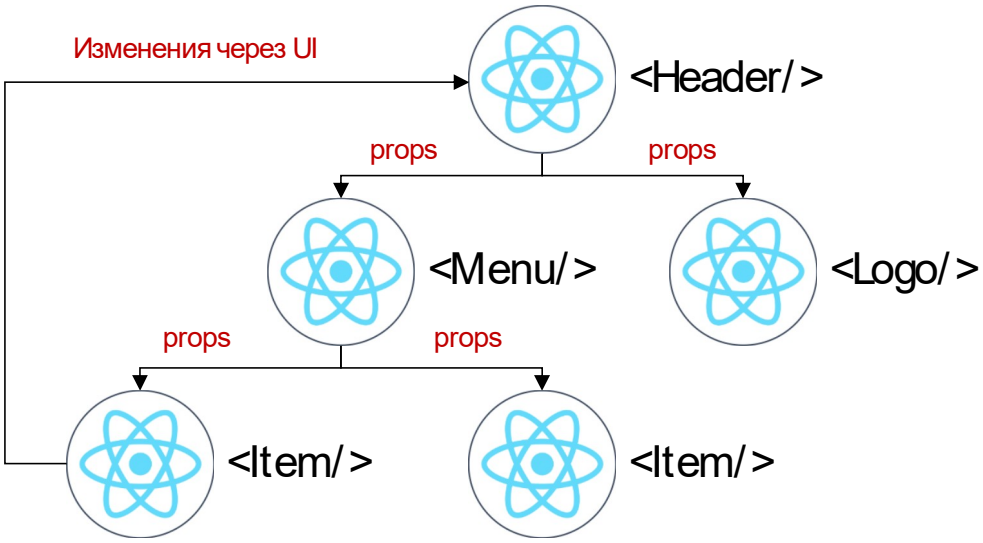
Diff-algorithm



Best-practices

- Данные лежат рядом с App.
- Данные спускаются до дочерних элементов через props
- Обычно хранятся в redux store (это именно то, что мы сейчас написали)
- Компоненты – переиспользуемые
- Используйте агрегацию вместо наследования.
- Компоненты прекрасно покрываются тестами (react-test-renderer, enzyme).

One-way Data-flow



Пример (и делаем)

```
let user = {
   username: 'Ivan',
   email: 'ivan@example.com'
};

class App extends Component {
  render() {
    return (
      <div className="App">
        <Header user={user}>

        ...
      </div>
    );
  }
}
```

OTUS

Упражнение

Вынести компонент Header с пользователем, вывести имя пользователя (передать через props).

Вопросы?

OTUS

Что прошли?

- Основы React
- create-react-app
- Webpack
- Babel
- JSX

Вопросы?

OTUS

Домашнее задание

Сделать Header, Footer, Navigation компоненты, определить ссылки для Navigation, передать их через Header

Что прочитать попробовать?

https://reactjs.org/

https://reactjs.org/docs/add-react-to-a-new-app.html

https://reactjs.org/docs/introducing-jsx.html

https://reactjs.org/docs/jsx-in-depth.html

Вопросы?

Спасибо
за внимание!