**React за 60 минут**

**Содержание семинара:**

1. ReactJS знакомство
2. TODO приложение

**Теоретическая часть:**

1. **ReactJS знакомство**

**React** — JavaScript-библиотека.

#### **Концепции** React имеет небольшой API, он прост в изучении использовании. Однако, сперва давайте взглянем на несколько концепций.

#### **Элементы** — это объекты JavaScript, которые представляют HTML-элементы. Их не существуют в браузере. они описывают DOM-элементы, такие как h1, div, или section.

#### **Компоненты** — это элементы React, написанные разработчиком. Обычно это части пользовательского интерфейса, которые содержат свою структуру и функциональность. Например, такие как NavBar, LikeButton, или ImageUploader. Компоненты — это душа и сердце React. Это пользовательские элементы. Обычно, они имеют свою уникальную структуру и функциональность.

#### **JSX** — это техника создания элементов и компонентов React. Например, это React-элемент, написанный на JSX:

<h1>Hello</h1>

Такой же элемент может быть написан на JavaScript:

React.DOM.h1(null, 'Hello');

**Virtual DOM** — это дерево React элементов на JavaScript. React отрисовывает Virtual DOM в браузере, чтоб сделать интерфейс видимым. React следит за изменениями в Virtual DOM и автоматически изменяет DOM в браузере так, чтоб он соответствовал виртуальному.

#### **Рендеринг** Первым делом происходит рендер виртуального элемента (элемента, или компонента React). Помните, пока виртуальный элемент содержится только в памяти JavaScript. мы должны явно сообщить React отрисовать его в браузере.

#### **Состояние** Состояние — это специальный объект внутри компонента. Он хранит данные, которые могут изменятся с течением времени

#### **Композиция** Композиция означает комбинирование меньших компонентов при формировании большего. Например, компонент **Photo** может быть использован внутри компонента **PhotoGallery**.

**Практическая часть:**

1. **TODO приложение**

Создаем простой html файл и подключаем файлы библиотеки (react.js, react-dom.js) и создаем контейнер, где будет размещаться TODO приложение на Реакте.

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width">

<title>ToDo App</title>

</head>

<body>

<div id="root"></div>

<script src="https://cdnjs.cloudflare.com/ajax/libs/react/15.1.0/react.min.js"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/react/15.1.0/react-dom.min.js"></script>

</body>

</html>

Давайте создадим наше первое Реакт приложение и добавим код после подключения библиотек:

<script>

ReactDOM.render(

React.createElement('h1', null, 'Привет, Мир!'),

document.getElementById('root')

);

</script>

Работать с таким синтаксисом очень не удобно, по этому в приложениях используют JSX. Для этого подключим еще одну библиотеку

***<script src="https://cdnjs.cloudflare.com/ajax/libs/babel-standalone/6.24.0/babel.js"></script>***

<script src="https://cdnjs.cloudflare.com/ajax/libs/react/15.1.0/react.min.js"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/react/15.1.0/react-dom.min.js"></script>

Наше приложение можно переписать использую JSX (нужно еще добавить *type="text/babel"*):

<script type="text/babel">

ReactDOM.render(

<h1>Привет, Мир!</h1>,

document.getElementById('root')

);

</script>

Перепишем наше приложение с учетом создания React компоненты:

class TodoApp extends React.Component {

constructor(props) {

super(props);

this.state = {};

}

render() {

return (

<div>

<h1>Привет, Мир!</h1>

</div>

);

}

}

ReactDOM.render(

<TodoApp />,

document.getElementById('root')

);

В конструкции «*this.state*» будет храниться текущее состояние компоненты. Это свойство используется только для чтения, для записи его можно использовать только в конструкторе.

Давайте добавим немного динамичности в приложение:

class TodoApp extends React.Component {

constructor(props) {

super(props);

this.state = {

data: [

{title: "buy milk",completed: false},

{title: "send letter",completed: true},

{title: "meet with friend",completed: false}

]

}

}

render() {

return (

<div>

<h1>ToDo List ({this.state.data.length})</h1>

</div>

);

}

}

ReactDOM.render(

<TodoApp />,

document.getElementById('root')

);

Выносим Title в отдельный компонент, данные в который мы будем передавать через «props». Так как этот компонент не будет иметь своего отдельного состояния то мы его создадим через простую константу.

const Title = (props) => {

return (

<div>

<h1>ToDo List ({props.count})</h1>

</div>

);

}

class TodoApp extends React.Component {

constructor(props) {

super(props);

this.state = {

data: [

{title: "buy milk",completed: false},

{title: "send letter",completed: true},

{title: "meet with friend",completed: false}

]

}

}

render() {

return (

<div>

<Title count={this.state.data.length} />

</div>

);

}

}

ReactDOM.render(

<TodoApp />,

document.getElementById('root')

);

Добавим еще два компонента – контейнер для задач и компонент для отображения задачи. Также предусмотрим логику удаления задачи из списка.

const Title = ({count}) => {

return (

<div>

<h1>ToDo List ({count})</h1>

</div>

);

}

const Todo = ({todo, remove}) => {

// Each Todo

return (

<li>

{todo.title}

<b style={{ cursor: 'pointer' }} onClick={()=> {remove(todo)}}> X </b>

</li>

);

}

const TodoList = ({todolist, remove}) => {

// Map through the todolist

const todoItem = todolist.map((todo) => {

return (<Todo todo={todo} remove={remove} />)

});

return (<ul>{todoItem}</ul>);

}

class TodoApp extends React.Component {

constructor(props) {

super(props);

this.removeTodo = this.removeTodo.bind(this)

this.indexOfElement = this.indexOfElement.bind(this)

this.state = {

data: [

{title: "buy milk",completed: false},

{title: "send letter",completed: true},

{title: "meet with friend",completed: false}

]

}

}

indexOfElement(todo){

let data = this.state.data;

let indexOf = -1

data.forEach((item, index) => {

if(item.title === todo.title){

console.log(item.title)

indexOf = index;

}

})

return indexOf;

}

removeTodo(todo){

let data = this.state.data;

let indexOfRemovedElem= this.indexOfElement(todo)

console.log(data.indexOf(todo))

data.splice(indexOfRemovedElem, 1)

this.setState({data: data})

}

render() {

return (

<div>

<Title count={this.state.data.length} />

<TodoList todolist={this.state.data} remove={this.removeTodo} />

</div>

);

}

}

ReactDOM.render(

<TodoApp />,

document.getElementById('root')

);

Добавим логику пометки задачи «выполнена-невыполнена». Добавим функцию ***completeTodo,*** и добавим кнопку для отметки выполнения задачи.

completeTodo(todo) {

let data = this.state.data;

let indexOfRemovedElem = this.indexOfElement(todo)

data.splice(indexOfRemovedElem, 1, { title: todo.title, completed: !todo.completed })

this.setState({ data: data })

console.log(this.state.data)

}

Изменения в верстке:

const Todo = ({ todo, remove, complete }) => {

// Each Todo

return (

<li>

<span className={todo.completed ? 'completed' : null}>{todo.title}</span>

<b style={{ cursor: 'pointer' }} onClick={() => { complete(todo) }}> V </b>

<b style={{ cursor: 'pointer' }} onClick={() => { remove(todo) }}> X </b>

</li>);

}

Теперь добавим форму добавления новой задачи в список. Финальный вариант нашего приложения:

<div id="root"></div>

<script src="https://cdnjs.cloudflare.com/ajax/libs/babel-standalone/6.24.0/babel.js"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/react/15.1.0/react.min.js"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/react/15.1.0/react-dom.min.js"></script>

<script type="text/babel">

const Title = ({count}) => {

return (

<div>

<h1>ToDo List ({count})</h1>

</div>

);

}

const TodoForm = ({addTodo}) => {

// Input tracker

let input;

return (

<div>

<input ref={node => {

input = node;

}} />

<button onClick={() => {

addTodo(input.value);

input.value = '';

}}>

+ Add

</button>

</div>

);

};

const Todo = ({todo, remove, complete}) => {

// Each Todo

return (

<li>

<span className={todo.completed?'completed':null}>{todo.title}</span>

<b style={{ cursor: 'pointer' }} onClick={()=> {complete(todo)}}> V </b>

<b style={{ cursor: 'pointer' }} onClick={()=> {remove(todo)}}> X </b>

</li>);

}

const TodoList = ({todolist, remove, complete}) => {

// Map through the todolist

const todoItem = todolist.map((todo) => {

return (<Todo todo={todo} remove={remove} complete={complete} />)

});

return (<ul>{todoItem}</ul>);

}

class TodoApp extends React.Component {

constructor(props) {

super(props);

this.removeTodo = this.removeTodo.bind(this)

this.indexOfElement = this.indexOfElement.bind(this)

this.completeTodo = this.completeTodo.bind(this)

this.addTodo = this.addTodo.bind(this)

this.state = {

data: [

{title: "buy milk",completed: false},

{title: "send letter",completed: true},

{title: "meet with friend",completed: false}

]

}

}

indexOfElement(todo){

let data = this.state.data;

let indexOf = -1

data.forEach((item, index) => {

if(item.title === todo.title){

console.log(item.title)

indexOf = index;

}

})

return indexOf;

}

removeTodo(todo){

let data = this.state.data;

let indexOfRemovedElem= this.indexOfElement(todo)

console.log(data.indexOf(todo))

data.splice(indexOfRemovedElem, 1)

this.setState({data: data})

}

completeTodo(todo){

let data = this.state.data;

let indexOfRemovedElem = this.indexOfElement(todo)

data.splice(indexOfRemovedElem, 1, {title:todo.title, completed:!todo.completed})

this.setState({data: data})

console.log(this.state.data)

}

addTodo(todoTitle){

let data = this.state.data;

data.push({title: todoTitle, completed: false})

this.setState({data: data})

console.log(this.state.data)

}

render() {

return (

<div>

<Title count={this.state.data.length} />

<TodoForm addTodo={this.addTodo} />

<TodoList

todolist={this.state.data}

remove={this.removeTodo}

complete={this.completeTodo} />

</div>

);

}

}

ReactDOM.render(

<TodoApp />,

document.getElementById('root')

);

</script>

**Дополнительные ресурсы:**

* Репозиторий готового приложения - <https://github.com/FullStack-Academy-Kiev/React60min/>
* Готовое приложение - <https://fullstack-academy-kiev.github.io/React60min/>