Александр Ольшанский

Контакты
Телефон: +380669848090
Электронная почта: sander2411@ukr.net
GitHub: https://github.com/saNder2411

Желаемая зарплата: от 700 $

Образование

Профессиональные курсы HTML Academy
– Подготовительный курс, апрель 2019
– HTML и CSS, уровень I, июль 2019
– HTML и CSS, уровень II, сентябрь 2019
– JavaScript, уровень I, ноябрь 2019
– JavaScript, уровень II, январь 2020
– JavaScript, уровень III, май 2020

Сертификаты обучения:
https://htmlacademy.ru/profile/id880751/certificates

Стек технологий*: JavaScript, TypeScript, React, React/Hooks, Redux, Webpack, Git, NPM.*

О себе

Открытый, доброжелательный и целеустремленный. Обладаю высокой увлеченностью и интересом к технологиям. Постоянно занимаюсь улучшением своих профессиональных навыков. К работе отношусь очень ответственно. Самостоятельно нахожу и изучаю нужную информацию по проекту. Неплохо разбираюсь в чужом коде. Увлекаюсь созданием электронной музыки, люблю спорт и историю.

Личные достижения

Успешно окончил 12-ти месячный профессиональный курс в HTMLAcademy "React-разработчик". Во время обучения защитил все личный проект на 100 баллов. Разрабатываю на TypeScript React/Redux, React/Hooks, в небольших приложениях пользуюсь PropTypes. Тестирую с помощью Jest и Enzym. Оптимизировал производительность приложений с использованием встроенного браузерного профайлера, для разработки и отладки работаю с «React dev Tools», «Redux dev Tools».

Отлично знаком с работой над иммутабельными объектами: React(component state), Redux(srore state: добавление и удаление элементов в массиве, обновление свойств элемента и тд.). Знаком с Store Enhancers, активно использую Middleware. Для диспатча async call использую Thunk Middleware.

Взаимодействие с REST API: XHR, метод Fetch, библиотека для создания HTTP-клиента Axios, Promise, Async/Await. Хранение данных: Local Storage, Service Worker. DAL(Data access Layer) реализую классом на базе HTTP-клиента Axios или метода Fetch, с приватными полями и методами для

настройки клиента и задания дефолтных параметров, и публичными методами для доступа к API сервиса. Доступ к экземпляру DAL класса осуществляю через контекст. Создаю HOC(withAPIService) который принимает в параметры функцию для мапинга метода экземпляра и задает в props оборачиваемому компоненту нужный метод для запроса данных у сервиса. Затем это метод диспатчу в методе жизненного цикла компонента(HOC withData), после обработки данных в reducer, HOC withData через mapStateToProps передает в props UI компонента состояние запроса(loading, data, error). При использовании хуков, в двух словах, делаю практически также, только на кастомных хуках. Хук useService взаимодействует с экземпляром DAL класса и вызывает хук useRequest, в который передает необходимый метод для запроса. useRequest возвращает массив с объектом состояния запроса и функцией для вызова этого запроса([{loding, data, error}, doRequset]). doRequset вызываеться либо при монтировании/обновлении UI компонента в хуке useEffect или в обработчике какого либо события.

Разрабатывал проекты на native JS с объектно-ориентированным подходом, с архитектурой на основе шаблонов проектирования MVC и MVP(с иерархией презентеров и моделей HMVC(P)). ООП: Adapter, Singltone, Observer, Controller, Factory Method, Abstract Factory, Decorator. В контроллерах и презентерах реализовывал data-binding, помимо callback-ов, с помощью объектов Proxy и Reflect. Считаю одним из главных принципов SOLID принцип единственной ответственности. По этому всегда разделяю архитектуру приложения на уровни: UI(React)⟺BLL(Redux)⟺DAL(API Service). UI компоненты не должны содержать никакой логики и отвечать только за рендеринг, это всегда простые функции которые легко тестировать.

Реализовывал: кастомную валидацию форм, формы с контролируемым и не контролируемым компонентом, Drag`n`Drop с событиями мыши на native JS (слайдеры, перетаскивание маркеров), диаграмму активности пользователя с библиотекой chart.js. Работал с интерфейсами HTMLVideoElement и HTMLAudioElement в проектах с видео и аудио данными. Стилизацию компонентов делаю с «styled-components» и «css-modules».

Пользуюсь вспомогательными библиотеками: «reselect», «query-string», «classnames», «nanoid», «he», «flatpickr.js», «chart.js», «moment.js». Помимо использования «create-react-app», создавал собственные конфигурации webpack и babel для сборки приложения.

Портфолио: https://sanDer-portfolio.web.app