# Alexander Olshansky

**Contacts**
phone: +380669848090
email: sander2411@ukr.net
GitHub: https://github.com/saNder2411

**Desired Salary:** from 700$

## Education

Professional courses HTML Academy
– JavaScript, level 3 #3, may 2020
– JavaScript, level 2 #10, january 2020
– JavaScript, level 1 #18, november 2019
– HTML & CSS, level 2 #17, september 2019
– HTML & CSS, level 1 #25, july 2019
– HTML, CSS & JavaScript, preparatory course, april 2019
Certificates of education:
https://htmlacademy.ru/profile/id880751/certificates

**Technology stack**: *JavaScript, TypeScript, React, React/Hooks, Redux, Webpack, Git, NPM.*

## About myself

Open, benevolent and purposeful. I have a high passion and interest in technology. Constantly improving my professional skills. I am very responsible for work. Independently find and study the necessary information on the project. I'm good at someone else's code. Fond of creating electronic music, love sports and history.

## Personal achievements

Successfully completed a 12-month professional course in HTMLAcademy "React-developer". During the training defended the entire all personal projects by 100 points. Developing on TypeScript React/Redux, React/Hooks, in small applications I use PropTypes. Testing with Jest and Enzym. Optimized application performance using the built-in browser profiler, for development and debugging I work with React dev Tools, Redux dev Tools.

I am very familiar with the work on immutable objects: React (component state), Redux (srore state: adding and removing elements in an array, updating element properties, etc.). Familiar with Store Enhancers, actively using Middleware. For dispatch async call I use Thunk Middleware.

Interaction with REST API: XHR, Fetch method, library for creating Axios HTTP client, Promise, Async/Await. Storage: Local Storage, Service Worker. I implement DAL (Data access Layer) with a class based on the Axios HTTP client or the Fetch method, with private properties and

methods for configuring the client and setting default parameters, and public methods for accessing the service API. Access the DAL class implements through the context. I create a HOC (withAPIService) that takes a function in the parameters for mapping the instance method and sets the required method in props to the wrapped component to request data from the service. Then this is the dispatch method in the component life cycle method (HOC withData), after processing the data in reducer, the HOC withData passes the request state (loading, data, error) to the props UI component. When I use hooks, in a nutshell, I do almost the same, only on custom hooks. A useService hook that interacts with an instance of the DAL class and calls the useRequest hook, which passes the necessary method for the request. useRequest returns an array with a request status object and a function to call this request ([{loding, data, error}, doRequset]). doRequset is called either when mounting / updating the UI component in the useEffect hook or in an event handler.

Developed projects on native JS with an object-oriented approach, with an architecture based on MVC and MVP design patterns (with a hierarchy of presenters and models HMVC(P)).OOP: Adapter, Singltone, Observer, Controller, Factory Method, Abstract Factory, Decorator. In controllers and presenters, I implemented data binding, besides callback, use Proxy and Reflect objects. I consider one of the main principles of SOLID - Single Responsibility Principle. Therefore, I always divide the application architecture into levels: UI(React)⟺BLL(Redux)⟺DAL(API Service). The user interface components should not contain any logic and should be responsible only for rendering, these are always simple functions that are easy to test.

Implemented: custom form validation, forms with controlled and uncontrolled components, Drag`n`Drop with mouse events on native JS (sliders, drag and drop handles), user activity chart with chart.js library. Worked with the HTMLVideoElement and HTMLAudioElement interfaces in projects with video and audio data. I do component styling with «styled-components» and «css-modules».

I use auxiliary libraries: «reselect», «query-string», «classnames», «nanoid», «he», «flatpickr.js», «chart.js», «moment.js». In addition to using the create-react-app, created his own Webpack and Babel configuration for building the application.