IPT – Intellectual Products & Technologies Ltd.
Bl. 44- Office 2 Slatinska Str., Sofia, Bulgaria
☏ (+359-2) 973-38-38 🖷 (+359-2) 971-28-50 ✉ office@iproduct.org 🖰 http://www.iproduct.org/

Advanced React JS + Redux Development

Course code:	**IJ - 27**
Course domain:	**Software Engineering**
Number of modules:	**1**
Duration of the course:	**40 astr. hours / 54 study[1] hours**

Sofia, 2017

1 Duration of a study hour is 45 minutes.

IPT – Intellectual Products & Technologies Ltd.
Bl. 44- Office 2 Slatinska Str., Sofia, Bulgaria
☏ (+359-2) 973-38-38 🖷 (+359-2) 971-28-50 ✉ office@iproduct.org 🖰 http://www.iproduct.org/

Advanced React JS + Redux Development

STUDY PLAN

Module name	1. Advanced React JS + Redux Development
Lectures, astr. hours	16
Laboratory exercises, astr. hours	24
Total, astr. hours	40

Lecturer: **Trayan Iliev**
IPT – Intellectual Products & Technologies Ltd.
E-mail: tiliev@iproduct.org

Target audience: Medium level JavaScript developers / students with practical experience using CSS and JavaScript

Course duration: Duration of the course is 40 astronomical (54 study) hours total. Training will be conducted in 10 sessions by 4 astr. hours.

Expected results:

- Experience using *ECMAScript 6 & 7* new features: Classes, Promises, lambda functions, iterators and generators, rest/spread, async/await
- Implementing reusable presentation components using *React JS*
- Experience building *single page web applications (SPA)* using *React Router*
- Planning state representation and management, best practices for maintainability and performance optimization using *immutability & Redux*
- Asynchronous state transitions and data-flows with *redux-thunk, redux-promise, redux-observable*
- Test-driven development of *React* and *Redux* components using *Jest, redux-mock-store, React Test Utils, Enzyme*
- Data query/mutation, automatic caching, pagination and infinite scrolling, optimistic UI, query batching, subscriptions, prefetching and server-side rendering using GraphQL & Apolo Client.

Bl. 44- Office 2 Slatinska Str., Sofia, Bulgaria
☏ (+359-2) 973-38-38 🖷 (+359-2) 971-28-50 ✉ office@iproduct.org 🖰 http://www.iproduct.org/

Course Description:

The course provides practical experience using React JS and Redux web component and application development frameworks. The main topics that will be covered during the course are:

1. **Novelties in ECMAScript 6 (ECMAScript 2015, Harmony)** – class and constructor syntax, static methods, inheritance, let and var, function lambdas (=>), template literals, destructuring assignment, default and rest parameters, spread operator, Promises. *ES6* development using *Babel* and *Webpack*. Writing reusable components as ES6 classes. Defining and loading JavaScript modules – requirements for code modularity, module systems, loaders, and JS design patterns: *CommonJS* and *ES 6* modules. Configuring, building and deploying *ES6* projects. Integration with build tools – *babel, npm, webpack, loaders, plugins, lazy loading & code splitting, cache busting, tree shaking*.(4 h.)
2. **Introduction to Single Page Applications (SPA) development** – *Model-View-Controller (MVC), Model-View-Presenter (MVC), Model-View-ViewModel (MVVM) – MV** patterns. Setting up a build system, defining front and back-end architecture (roter, models, views), application design and development (views, APIs and models, router config). React main features and advantages – simple (just the View) and superfast, component oriented development using pure JavaScript (ES 6), virtual DOM, one-way reactive data flow, MVC framework agnostic. (1 h.)
3. **Building React components and applications.** Development workflow, project setup and configuration – *package.json, babel (ES 6), webpack*. Building simple React components and application. Top level API: *React, ReacrDOM, ReactDOMServer. React.createElement()* and *ReactDOM.render()* methods. Using *JSX* in React. Showing dynamic values. Composing components using properties. Building sample Comments application using official React.js tutorial. Stateful components – react states, setting initial state, updating state. Events in React, managing DOM events. Using *React Developer Tools Chrome extension* . (3 h.)
4. **JSX in depth** – differences with HTML, transformation to JavaScript, namespaced components, expressions, child expressions and comments, props mutation anti-pattern, spread attributes, using HTML entities, custom attributes, *if-else*, immediately-invoked function expressions (*IIFE*). (2 h.)
5. **Components composition in depth** – ownership, *this.props.children, React.Children* utilities, child *reconciliation*, stateful children and dynamic children using keys. Transferring *props*. Using *mixins* and *ES6 classes*. *Stateless (pure) functions* as components. Where should we put state – common patterns. Integration with other

JavaScript libraries. Using *markdown* – *__html, dangerouslySetInnerHTML*. *Ajax* requests using *Axios*. (2 h.)

6. **React rendering lifecycle and life cycle callbacks**. Bidirectional owner-child component communication using callbacks passed as properties. (1 h.)
7. **Working with forms** – interactive props, controlled and uncontrolled components. Form building using *higher-order React components*. (3 h.)
8. **Novelties in React.js** – *Dependency Injection (DI)* of custom application services using *React Context*. Using *addons*. Two way binding helpers addon. *Refs* to components. (2 h.)
9. **React Router** – routes and components, named components, route parameters, index route, links, index links. Nested routes – nested UI components and routing URLs. Active links. *NavLink* wrapper component. Decoupling the UI from the URL by using routes without paths. Using route redirection to preserve changed routes. Controlling navigation and application state using *onEnter* and *onLeave* hooks. Route patterns and optional route params. Simultaneous navigation in multiple viewports using named routing components. Histories – *browserHistory* vs. *hashHistory*. Server-side support needed when using *HTML 5 History API*. Navigating routes programmatically. Using custom *History* implementations – coding state as route query parameters. Configuring *Router* programmatically using *RouteConfig* objects and setting *RouteLeaveHooks* to validate navigation. (4 h.)
10. **Performance optimization using immutability** – *shouldComponentUpdate()* component method, *PureRenderMixin* and *shallowCompare()* to boost performance. *Immutability Helpers* addon. Cloning ReactElements. (1 h.)
11. **Testing react components** with *ReactTestUtils* addon – events simulation, mocking, shallow rendering. (3 h.)
12. **Animation addon** – *ReactCSSTransitionGroup* and *ReactTransitionGroup*. Internationalization (i18n) and localization (l10n).(2 h.)
13. **Redux – predictable state container for JavaScript apps.** Predecessors: *Flux* design pattern – unidirectional data flow, main components (*Action creators, Actions, Dispatcher, Stores, Views* and *Controller-Views*). Building example TODO application using *Flux*. *Redux* – combining *Flux* reactive uni-directional data flows with event sourcing principles and state reduction. Three main principles of *Redux* – *single source of truth* (single store); *state is read-only* (only actions can change state); *changes are made using pure functions* – reducers: (state, action) => state. *Redux* main components – *Actions, Action Creators, Reducers, Stores*. Interaction between

Redux components – designing state representation, handling *Actions* using separate *Reducers*, combing reduces into single *Root Reducer* using *combineReducers()* helper function, registering listeners via *Store.subscribe()*, dispatching *Actions* using *Store.dispatch()*, unidirectional data flow. Connecting *Redux* with *React* using *React Redux bindings*. Presentational and container components. Implementing TODO application using *React* and *Redux*. (4 h.)

14. **Advanced Redux** – nested API responses normalization using normalizr. Implementing *async actions* using *redux-thunk, redux-promise/ redux-promise-middleware*, or *redux-observable*. Asynchronous data-flows using *applyMiddleware()*, examples. Using *redux-devtools*. Building *SPA* with *Redux* and *React Router*. Extended *Redux routing* support using *Redux Router* and *React Router Redux* libraries. *Server rendering* with Redux. Computing derived data and *memoization* using *Reselect*. Implementing *Undo History*. Testing *Redux* components using *Jest, redux-mock-store, React Test Utils, Enzyme*. (4 h.)

15. **Fetching data using GraphQL (Apolo)**. Introduction to *GraphQL* – queries, mutations, fragments, variables, directives, schemas and types. Using *Apollo Client* – Redux-based, flexible, production-ready, fully-featured *GraphQL* client for *React* platform. *React-Apollo integration*: queries and mutations, getting updates from the server, controlling the *Store*, multiple clients. Recipies for authentication, pagination, optimistic UI, using fragments, prefetching data, integrating with *Redux*, *server-side rendering*, *Webpack* integration. Using *apollo-client-devtools*. (4 h.)

The workshop contains lecture materials and lab exercises. Lectures and exercises will be conducted in parallel and will not be divided in separate sessions in order to achieve immediate reinforcement of theoretical discussions with practical examples and exercises.

During the workshop participants will get practical experience using ReactJS and Redux frameworks as well as developer tools like *create-react-app* and *redux-devtools* to develop and test (single page) web applications. The learning is conducted in small groups – up to 10 participants using problem-oriented methodology. During the course there will be opportunity for discussion of additional questions the participants are interested in.