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/

Single Page Application Development with AngularJS

Course code:	**IJ - 30**
Course domain:	**Software Engineering**
Number of modules:	**1**
Duration of the course:	**36 astr. (48 study[1]) hours**

Sofia, 2016

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/

Single Page Application Development with AngularJS

STUDY PLAN

Module name	1. Single Page Application Development with AngularJS
Lectures, astr. hours	17
Laboratory exercises, astr. hours	17
Final test and practical problem, astr. hours	2
Total, astr. hours	36

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

Target audience: Medium level JavaScript developers with practical experience in building web applications using HTML 5, CSS 3, JavaScript/ECMAScript 6, and Bootstrap.

Course duration: Duration of the course is 36 astr. (48 study) hours total. Training will be conducted in 9 days – 4 astr. hours each day.

Course Description:

The course provides in-depth study of state-of-the-art *JavaScript (ES 5 & 6)* and *AngularJS 1.5 MV* framework* for rapid development of modern, mobile-first, responsive single-page applications that are easy to extend and maintain in long run. The main topics that will be covered during the course include:

1. **Single Page Applications (SPA).** Horizontal and vertical SPA. Introduction to *Model-View-Controller (MVC), Model-View-Presenter (MVC), Model-View-ViewModel (MVVM) – MV** patterns for development of more complex, extensible and easy to maintain web applications using *AngularJS framework*. Advantages and typical use-cases of *AgnularJS* as a *complete solution* for client-side *MV* SPA* development. Developing *AngularJS* applications with *VS Code*. Creating and bootstrapping simple *AngularJS TODO Application*. (3 h.)

2. **Step-by-step practical introduction to *AngularJS***, following the official *AngularJS* tutorial – *Phone Catalog App*. Starting with *angular-seed project*. Bootstrapping, expressions, templates, models, controllers, scopes, and directives. *Web components*. Using and testing *AngularJS* components. Project directory and file structure – *domain-driven design* and organizing code *by feature/section*. Using *AngularJS modules* to manage feature/section dependencies. Using external templates. Using filters. Two-way data binding using *ngModel* and *ngModelOptions* directives. XHR & Dependency Injection (DI) – using *$http* service. Minification-safe DI using *$inject* constructor property, and using *array with named dependencies*. Templating links and images – using *ngSrc* directive. Multiple views and introduction to routing - using *ngRoute* module and *ngView* directive. Configuring routing service provider. Uri templates and *$routeParams* object. Defining custom filters. Adding interactivity with event handlers – *ngClick* directive. Using *REST* and custom services. Adding animations using *ngAnimate* module, *CSS transitions* and *keyframe animations*. (5 h.)

3. **AngularJS in depth** – *angular-seed project* and advanced *AngularJS* project setup and application component development with *npm, webpack,* and *yeoman* generator – using *generator-angular-webpack-es6* (*angularOcLazyLoad*, tree shaking feature enabled, *ES6, ES2017, SASS, UI router*, *bootstrap-sass*, *ngAnnotate, ngTemplateCache*, optional angular modules installation, development and production mode configurations). Development and production builds, package dependencies and project configuration with *package.json. Webpack* configurations. *ES6* and *AngularJS* modules – dependencies, *imports/exports*, creation versus retrieval. Views and templates, using external templates with *templateUrl*. Contexts and expressions, interpolation - *$scope* and *$parse. Scope* as *ViewModel* – main methods (*$new, $eval, $watch, $digest, $apply, $on, $destroy*), isolate scopes, scope hierarchies, events propagation, lifecycle, integration within browser loop. Data binding – unidirectional vs. bidirectional. Controllers – integration with scopes. Main directives. Directive types and normalization. Directives that create scopes. Components. Angular events. Forms and directives. Binding to form and control state. Form validation – builtin and custom validators as *validation directives*. *CSS* classes. Custom model update triggers, event debouncing. Implementing custom form controls using *ngModel*. (6 h.)

4. **AngularJS Services** – built-in and custom services. Providing and consuming services. Dependency Injection (DI) – factory methods, module methods *config* and *run*, implicit, inline array and *$inject* annotations, using *ngAnnotate* tool to add, remove

and rebuild DI annotations, using strict dependency injection. Using *$injector, $provide*, provides and decorators. Object persistence and query using services. *REST* and *JSON APIs*. Using *$log, $q, $http, $resource* built-in services. Promises and asynchronous event pipelines. Building custom services. Repository design pattern. Filters – using filters in templates, controllers, services, and directives. Creating custom filters. (4 h.)

5. **AngularJS routing** – components, routes, outlets, URLs, links, navigation. Three *AngularJS router* choices: *ngRoute* module, *component router*, and *UI router* (state-based + extra features). *UI router* as de-facto standard for more complex *AngularJS* apps. State management and navigation – *$stateProvider*, *ui-view* directive. Activating a state – *$state.go()*, *ui-sref* directive, and by navigation in the browser. State configuration – *name, template, templateUrl, controller, controllerProvider, resolve*, and *data* properties, *onEnter* and *onExit* callbacks. State change and view load events. Nested states and nested views – *children* and *parent* properties, object (. dot) state naming notation, inherited resolved dependencies and custom data, abstract states. Multiple named views and paired states navigation – view reference scheme: *viewname@statename*, relative and absolute names. URL routing – URL basic and regex parameters, path, query, and state specified parameters, wildcards, using parameters in links and with programmatic navigation. Absolute routes (^). *$stateParams* service. *$urlRouterProvider* service – *when* (redirection), *otherwise* (page not found handling), *rule* (custom navigation rules). *$urlMatcherFactory* , *UrlMatchers*, and *$templateFactory*. Building SPA with more complex navigation requirements using *UI router.*(4 h.)

6. **Building custom directives** – isolating the scope of a directive, manipulating *DOM*, wrapping existing content, adding event listeners. Using built-in services *$location, $sce* and *$compile*. Transclusion. Creating embedded custom directives that communicate – using *$compile service* and *Directive Definition Object* properties (*priority, restrict, template, scope, bindToController, controller, controllerAs, require - ?, ^, ^^, ?^, ?^^, transclude, compile, link*). Pre- and post-*linking functions*. Controller lifecycle hooks – *$onInit, $onChanges, $onDestroy, $postLink*. (4 h.)

7. **Test Driven Development (TDD)** – unit/end-to-end testing AngularJS classes and components using *Jasmine, Karma, angular-mocks (ngMock)* and *Protractor.* Separation of concerns and dependency injection. Testing controllers, services, filters and directives. Using *beforeAll()*. Testing transclusion, external template, and embedded directives – tips and tricks, debugging. Using *karma-ng-html2js-*

preprocessor to pre-compile HTML templates. Testing promises and asynchronous behavior. Review of sample AngularJS GitHub example projects (bigger ones). (4 h.)

8. **Wrapping up** – developing *AngularJS SPA*. *i18n* and *l10n.* Includes. Security. Applying animations – CSS and JavaScript animations. Advanced concepts – *Shadow DOM, lazy loading, Google Material Design.* (4 h.)
9. **Final test + practical problem** + discussion of additional questions. (2 astr. h.)

The course contains 50% lecture materials and 50% 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 new concepts with practical examples and problem solving activities.

During the course participants will get practical experience using *AngularJS 1.5 framework* for building *single page applications* by solving problems and exercises. The learning is conducted in small groups – up to 8 participants using problem-based methodology. During the workshop there will be opportunity for discussion of additional questions the participants are interested in. At the end of the course participants are expected to solve a practical problem – develop simple SPA with AngularJS and routing.