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/

Angular 2 and TypeScript Web Application Development

Course code:	**IJ -23**
Course domain:	**Software Engineering**
Number of modules:	**1**
Duration of the course:	**48 study[1] (36 astr.) hours**

Sofia, 2017

1 Duration of a study hour is 45 minutes.

Angular 2 and TypeScript Web Application Development

STUDY PLAN

Module name	1. Angular 2 and TypeScript Web Application Development
Lectures, astr. hours	17
Laboratory exercises, astr. hours	18
Test , astr. hours	1
Total, astr. hours	36

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

Target audience: Medium to expert level JavaScript developers with practical experience in building web applications using JavaScript libraries.

Course duration and schedule: Duration of the course is 36 astr. hours total. Training will be conducted in *5 weekends* (Saturday and Sunday, 9 days), 4 astr. hours each day: *9.00 – 13.15 h*.

Course Description:

The course provides in-depth study of state-of-the-art *JavaScript (ES 6 & 7), TypeScript* and *MV* Angular 2 JS/TS framework* for rapid development of modern, component-based, mobile-first, responsive single-page applications that are easy to extend and maintain in long run. Angular 2 builds on success of AngularJS web application development framework, but is significantly more efficient (up to 10-15 times faster for big number of updated components), streamlined, elegant and easy to learn.

The covered topics include:

1. **Introduction to TypeScript** – functions, interfaces, classes and constructors. Common types: *Boolean, Number, String, Array, Enum, Any, Void*. *Duck typing*. Declaring contracts using Interfaces – properties and optional properties, function types, class types, array types, hybrid types, extension of interfaces. Classes – constructors, public/private properties, get and set accessors, static and instance sides. Functions and function types – optional default and rest parameters, function lambdas and use of this, overloads.

Using *Enums*. *Modules in TypeScript* – namespaces and modules (former internal and external modules), using *import*, *export* and *require*. Interoperability with external JS libraries – ambient type declarations and ambient modules. Module resolution. *Generic type parameters* – writing generic functions and classes, generic constructors, bounded generics. Type inference and type compatibility. Declaration merging. Decorators. Configuring, building and deploying TypeScript project – typings for *npm* packages, compiler options. Integration with build tools – *npm, webpack*. (4 h.)

2. **Introduction to Single Page Applications (SPA) development using Angular 2 and TypeScript** – using *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 *Visual Studio Code*. Creating Angular 2 Hello World Application. Web components. Data binding. Differences between Angular 1 and Angular 2. Advantages of Angular 2. Data architecture in Angular 2 – overview, main types of components: *Module, Component, Template, Metadata, Data Binding, Service, Directive, Dependency Injection*. Component controllers, views & templates – syntax for defining templates for model data visualization, property bindings, class & style property bindings, event bindings, component styling. Using external template and style files. Using *Angular Command Line Interface (CLI)* to setup Angular 2 project and add application components. Angular 2 by example – building from scratch *Products Manager* sample application. (4 h.)
3. **Displaying data using template interpolation and built-in directives**: *NgModel, NgFor, NgIf, NgSwitch, NgStyle, NgClass*. Handling user input – binding event handlers, getting data from the *$event* object, using local template variables, event filtering, handling multiple events. Building forms – two-way data binding: *[(ngModel)]*, change tracking, validation, and error handling: *ngControl*, providing custom styles for different control states, resetting and submitting forms, edit/display forms. Using *NgForm, NgControlName, NgFormControl, NgControlGroup*, and *NgFormModel* directives to bind a domain model to the form. *Reactive Forms* using *FormGroupDirective, FormControlDirective*, *FormControlName, FormGroupName, FormArrayName*, and *FormBuilder*. Organizing *FormControls* using *FormGroups* and *FormArrays*. Using form validators – *RequiredValidator, MaxLengthValidator, MinLengthValidator, PatternValidator* directives and *Validators* class. Building custom validator directives by implementing *Validator* interface. Building *asynchronous form validators*. Demos and exercises. (4 h.)

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/

4. **Angular 2 services.** Building custom services. Understanding *Dependency Injection (DI)* – DI using constructors, *ng2 DI framework*, configuring the injector, registering providers in components, declarative and programmatic dependency injection, injecting service dependencies in a service using *@Injectable* decorator, registering custom providers using provide function, dependency injection tokens. Using hierarchical injectors. Demos and exercises. (2 h.)

5. **Angular 2 template syntax in depth** – interpolation and template expressions. Expression writing guidelines: no visible side effects, quick execution, simplicity, idempotence. Template statements, statement guidelines. Binding syntax and semantics. Difference between HTML attributes and DOM properties. Binding targets – properties, events, attributes, classes and styles. Examples for different types of bindings. Emitting custom events with *EventEmitter*. Built-in directives in depth, * and tags. Local template variables. Input and output properties – using *@Input* and *@Output* decorators. Aliasing input and output properties. Template expression operators – *pipe (|)* and *Elvis (?.)*. Data bindings modification using built-in pipes – *date, lowercase, uppercase, number, percent, currency, slice, async, json pipes*. Building custom pipes. (2 h.)

6. **Routing and navigation in Angular 2 single page applications** – setting the base href and importing *'@angular/router'* module. Configuring a router using *RouterModule.forRoot()* and *RouterModule.forChild()* with *Routes* – paths and components. Building Routing Components – injecting Router service, providing router outlets using *RouterOutlet* directive, link parameters array that propels router navigation, navigating by user clicks – data-bound *RouterLink* directive, programmatic navigation using *Router* service, toggling css classes for the active router link, embedding important information in the URL with route parameters, adding child routes under a feature section, using wild-cards, redirecting from one route to another, matching strategies (prefix vs. full), using empty-path route configurations, setting default routes, using multiple named outlets, componentless routes. Accessing information about activated route using *ActivatedRoute* service – Observable params, data and resolves vs. non-observable (snapshot) alternative. Confirming or canceling navigation with guards – *CanActivate* to prevent navigation to a route, *CanDeactivate* to prevent navigation away from the current route. Passing optional information in query parameters, persisting information across routes with global query parameters, jumping to anchor elements using a fragment. Choosing between *"HTML5" (history.pushState)* and *"hash" ("#")* URL styles for SPA navigation. (4 h.)

7. **Angular 2 HTTP client and RxJS.** Asynchronous data requests (AJAX) using *XMLHttpRequest (XHR)* and *JSONP*. Same origin policy and *CORS*. Injecting *ng2 HTTP / JSONP* services. Building simple JSON-based CRUD REST(-like) application – *Product Manager HTTP Client*. JSONP cross-origin requests – *Wikipedia Search* form demo. Smoother user experience using reactive event streams processing with *RxJS Observables*. *Observables, Subscribers, Subjects*. Conversion to *Promises*. Composing functional operators for event stream transformation – *RxMarbles* examples. *IPT Reactive WebSocket* implementation – exposing it as Angular 2 service. (4 h.)

8. **Advanced Angular 2 topics** – writing custom attribute and structural directives. Component lifecycle and use of custom lifecycle hooks: *ngOnInit, ngOnChanges, ngDoCheck, ngOnDestroy, ngAfterContentInit, ngAfterContentChecked, ngAfterViewInit, ngAfterViewChecked*. Usage examples. Accessing component's *ViewChildren* and *ContentChildren* using *@ViewChild, @ViewChilderen, @ContentChild, @ContentChildren* decorators. Understanding difference between using *viewProviders* and *providers* component configuration option. *Shadow DOM* and Angular 2 view encapsulation modes – *ViewEncapsulation.None, ViewEncapsulation.Emulated* and *ViewEncapsulation.Native*. (2 h.)

9. **Angular 2 Redux (ngrx)** – using *Reactive Extentions for Angular (ngrx)* to build large web apps with predictable state management. *Flux* and *Redux* design patterns. Redux main components – *Actions, Action Creators, Reducers, Store*. Interaction between *Redux* components. Using *Redux devtools* Chrome extension. (4 h.)

10. **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 *Angular 2* platform. *Angular 2 – Apolo 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*, defining *proxy* configuration with *Angular CLI*, *Ahead-of-Time Compilation (AOT), Webpack* integration. Using *apollo-client-devtools* Chrome extension. (4 h.)

11. **Angular 2 animations** – quick-start example, states and transitions, entering and leaving from different states, animatable properties and units, automatic property calculation, animation timing, multi-step animations with keyframes, parallel animation groups. (2 h.)

The workshop 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 theoretical discussions with practical examples and exercises.

During the workshop participants will get practical experience using Angular 2 JS/TS framework and Angular CLI for building single page applications. The learning is conducted in small groups – up to 10 participants using problem-based methodology. During the workshop there will be opportunity for discussion of additional questions the participants are interested in.