IPT – Intellectual Products & Technologies Ltd.
Bl. 44- office 2, Slatinska Str., Sofia 1574, Bulgaria
☏ (+359) 889 895196 ✉ office@iproduct.org http://iproduct.org/

Angular and TypeScript Web Application Development

Course domain:	**Software Engineering**
Number of modules:	**1**
Duration of the course:	**36 hours**

Sofia, 2019

Angular and TypeScript Web Application Development

STUDY PLAN

Module name	1. Angular and TypeScript Web Application Development
Lectures, astr. hours	18
Laboratory exercises, astr. hours	18
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: Duration of the course is 36 astronomical hours in total.

Course Description:

The course provides in-depth study of state-of-the-art *JavaScript (ES 6 , 7, 8 & 9), TypeScript* and *MV* Angular 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 builds on success of AngularJS web application development framework, but is significantly more efficient, 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 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. Using Angular Command Line Interface (Angular CLI) to* setup Angular project and add application components. Creating first Angular Todo application. Web components. Data binding. Angular application architecture – overview, main parts: *Module, Component, Template, Metadata, Data Binding, Service, Directive, Dependency Injection*. Component controllers, views & templates – syntax for defining templates for model data visualization, property bindings, event bindings, component styling. Using external template and style files. *Ahead-of-Time Compilation (AOT)*. (3 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. Template expression writing guidelines. Binding targets – properties, events, attributes, classes and styles. Input and output properties – using *@Input* and *@Output* decorators. Emitting custom events with *EventEmitter*. Built-in directives and templates *(*)* in depth. Local template variables.Template expression operators – *pipe (|)* and *safe navigation (?.)* operators. Data bindings modification using built-in pipes – *date, lowercase, uppercase, number, percent, currency, slice, async, json* pipes. Angular by example – building *Products Manager* sample application. (4 h.)

4. **Angular tempale-driven and reactive forms** – two-way data binding: *[(ngModel)]*, change tracking, validation, and error handling: 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.)

5. **Angular services.** Building custom services. Understanding *Dependency Injection (DI)* – DI using constructors, *Angular 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.)
6. **Angular HTTPClient and RxJS**. Asynchronous data requests (AJAX) using *XMLHttpRequest (XHR)*. Same origin policy and *CORS*. Using new Angular *HTTPClient , interceptors, logging, caching responses, handling ProgressEvents*. Building *Observable services* with Angular HTTPClient. Cross-origin requests – *Wikipedia Search* form demo. Reactive Extensions for JS (RxJS). Smoother UX using *RxJS Observables, Subscribers, Subjects*. Conversion to *Promises*. Composing functional operators for data stream transformation. *RxMarble* diagrams. (4 h.)
7. **Routing and navigation in Angular 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 active link *CSS* classes, 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, *lazy-loading* feature modules. 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 current route. Passing optional information in query parameters, jumping to anchors using fragments. Choosing between *"HTML5" (history.pushState)* and *"hash" ("#")* URL navigation styles. (5 h.)

8. **Advanced Angular topics** – custom pipes, custom directives, using *Renderer2* class. 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. Component *providers* and *viewProviders. Shadow DOM* and Angular view encapsulation modes – *ViewEncapsulation.None / Emulated / Native*. *Zones* & *change detection*. Angular component performance optimization strategies. (4 h.)
9. **Reactive Extentions for Angular (NGRX)** – *Flux* and *Redux* design patterns, main components and their interactions – *Actions, Action Creators, Reducers, Store, Selectors*. Predictable reactive state management with *NGRX/store*. Reselect-style computing of derived data with *RxJS*. Implementing asynchronous state transitions using *NGRX/effects* and *@Effect* decorator. NGRX Router integration. Using custom *RouterStateSerializer.* Using *Redux devtools* Chrome extension. (4 h.)
10. **Angular animations. I18n. Q&A**– quick-start example, states and transitions, entering and leaving from different states, animation timing, keyframe animations. Angular *i18n* – i18n pipes, template translations, source file generation with *ng xi18n*, merging app and translations with *JIT* and *AOT*. (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* framework with *TypeScript* and *Angular CLI* for building single page front-end 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.