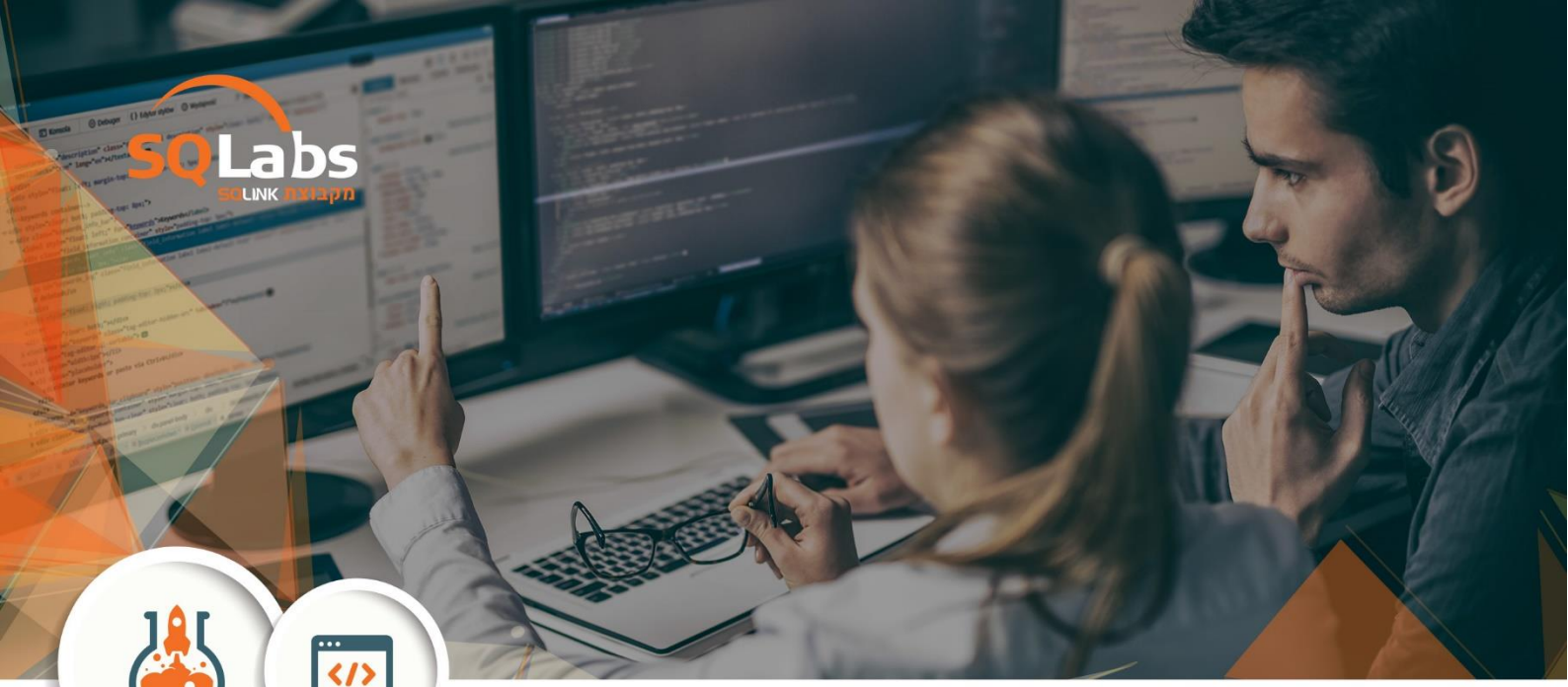


# Angular 7 – 40 Hours

## Overview

Angular brings an entirely new approach to web application development. We need to understand TypeScript, RxJS, and a new template syntax just to get started!

This course provides a developer-level introduction along with more advanced and useful features of JavaScript and Angular.

## Prerequisites

Front end developers with good knowledge in JS

## Course content

### Module 1: JavaScript 6.0 (ECMAScript 2015)

- Execution Context & Scope
- Syntax
- Functions
- Classes
- Collections
- Iterators & Generators
- Modules
- Template String
- Proxy

### Module 2: TypeScript

- Basic Syntax
  - Types
  - Classes
  - Functions

- Generics
- Type Inference & Compatibility
- Decorators

### Module 3: Async and Parallel in JavaScript

- Callbacks vs. Promises
- Multithreading with Workers
- Reactive Extensions (Rx.js)
- Zone.js

Angular is a development platform for creating applications using modern web standards. Angular includes a wealth of essential features such as mobile gestures, animations, filtering, routing, data binding, security, internationalization, and beautiful UI components. It's extremely modular, lightweight, and easy to learn.

### Module 1: Angular Overview

This module gives a quick overview of the main Angular components and how they work together.

- What are Web Apps (SPA)?
- Directives & Components
- NgModules
- Angular Bootstrap
- Routing
- Services
- Angular CLI Commands

### Module 2: Template Syntax

This module shows how to write templates that display data and consume user events with the help of data binding.

- Interpolation
- Binding syntax
- Property Binding
- Attribute, Class and Style Bindings
- Event Binding
- Two-way data binding with NgModel
- Built-in Directives
- * and <template>
- Local template variables

### Module 3: Components

Angular components use web standards (such as shadow DOM and the HTML5 template tag) in browsers that support them. Angular components are lightweight, reusable, self-contained UI components that have a single specific purpose.

- Components Overview
- Component Metadata (@Component)
- @Input

- @Output
- NgContent
- @ContentChild and @ContentChildren
- @ViewChild and @ViewChildren

## Module 4: Directives

There are two kinds of directives in Angular. **Attribute directives** can change the appearance or behavior of an element and **structural directives** can change the DOM layout by adding and removing DOM elements.

- Directives Overview
- Directive Metadata (@Directive)
- Attribute Directives
  - ElementRef
  - Renderer
  - @HostBinding
  - @HostListener
- Structural Directives
  - The <template> tag and TemplateRef class.
  - The asterisk (*) effect
  - ViewContainerRef class

## Module 5: Services & Dependency Injection (DI)

Angular ships with a powerful, yet simple-to-use dependency injection, allowing you to maintain modular applications without writing tedious glue code.

- What is a Service?
- @Injectable() & @inject
- Configuring the Injector
- The Injector Tree
- Component Injectors
  - Providers
  - View Providers
- Angular Modules (@NgModule)

## Module 6: Pipes

In this module we learn how to use pipes. Pipes transform displayed values within a template.

- Using Pipes
- Built-in pipes
- Parameterizing a Pipe
- Chaining pipes
- Custom Pipes
- Stateful Pipes
- Async Pipe

## Module 7: Forms

In this module we learn how to build a form that creates a cohesive, effective, and compelling data entry experience. An Angular form coordinates a set of data-bound user controls, tracks changes, validates input, and presents errors.

- Model Driven Development:
  - FormControl & FormGroup classes
  - FormBuilder service
  - Validations & Errors
- Template Driven Development:
  - Two-way data binding with ngModel directive
  - ngForm directive
  - Change tracking
  - Validations & Errors
- Value Accessors

## Module 8: Routing & Navigation

In this module we learn how to navigation from one view to the next as users perform application tasks.

- Configuring the router service
- RouterLink & RouterOutlet directives
- Relative Navigation
- Link parameters & query parameters
- Lazy loading & Pre Loading
- Preloading
- Confirming or canceling navigation with guards:
  - CanActivate
  - CanActivateChild
  - CanDeactivate
  - Resolve
  - CanLoad

## Module 9:  Server Communication (Http service)

In this module we learn how to communicate over HTTP with the server.

- HttpModule Overview
- Enable RxJS Operators
- Http API's
- Http Default Options
- Catch Operator
- JSON Web Token (JWT)
- Communication with JSONP
- Http Internal

## Module 10: Lifecycle Hooks

A component has a lifecycle managed by Angular itself. Angular creates it, renders it, creates and renders its children, checks it when it's data-bound properties change, and destroys it before removing it from the DOM. Angular offers lifecycle hooks that give us visibility into these key moments and the ability to act when they occur.

- OnChanges
- OnInit
- DoCheck
  - The Differs classes:
    - KeyValueDiffers
    - IterableDiffers
- AfterContentInit
- AfterContentChecked
- AfterViewInit
- AfterViewChecked
- OnDestroy

## Module 11: Change Detection

In Angular 1 the main reason for performance issues was the digest loop ($apply method). Angular 4.0 brings an entirely new approach called change detection. In this module we learn how the change detection mechanism works to update the screen and how to use it to improve the performance for high frequency data changes.

- What is tick in Angular Application?
- NgZone
- ChangeDetectorRef class
  - markForCheck & detectChanges methods
  - detach & reattach methods
- Ferformance Optimizations:
  - Change Detection Strategy
  - High Frequency Problem
  - High Expressions Problem

## Module 12: Ahead-of-Time Compilation

- Compiling in Angular 1
- Abstract Syntax Tree (AST)
- Just-in-time Compilation (JIT)
- Ahead-of-time Compilation (AOT)
- AOT Build Process

## Module 13: Upgrading From 1.X (Optional)

Angular applications can be incrementally upgraded to Angular. Having an existing Angular application doesn't mean that we can't enjoy everything Angular has to offer. That's because Angular comes with built-in tools for migrating Angular projects over to the Angular platform.

- Preparation for migrating.

- Component method vs. Directive method.
- Upgrading with The Upgrade Adapter.
  - How The Upgrade Adapter Works.
  - Using Angular  Components from Angular  Code.
  - Using Angular Component Directives from Angular  Code.