Angular 1

- HTML
- CSS
- JS
- Introduction to AngularJS:
    - How Angular.js is opinionated
    - Introduction of client side frameworks
    - Difference between Client side and server side frameworks
    - Introduction to MVC Frameworks
- Angular JS Building Blocks Features:
    - Controller Component
    - Model Component
    - View Component
    - Directives
    - Filters
    - Services
    - Providers (http://www.dotnettricks.com/learn/angularjs/understanding-angularjs-factory-service-and-provider)
    - Factory
    - DI in Angular JS
- Anatomy of an Angular JS Application:
    - Model View Controller
    - Binding controllers with views
    - Controller hierarchy
    - Templates and Data Binding
    - Repeating elements in templates
    - Sharing data between controllers
    - Using Expressions, CSS Classes and Styles
    - Using Controllers for UI responsibility separation
    - Responding to model changes
- Data Binding:
    - Understanding Built-in Directives
    - Scope resolution
    - $scope
    - $rootscope
    - Difference between $scope vs $rootscope
    - One way, One time and Two way data binding
    - JSON Advantages
    - Using Json in Angular JS
    - Using of $watch, $digest & $apply
- Dependency Injections:

- What is Dependency Injections
- Implicit Dependency Injections

- Filters:
  - Filters Overview
  - Understanding Filter Expressions & Directives
  - Building custom Filters
- Events
  - Introduction to ng-click
  - Hiding HTML Elements
  - ng-disabled, ng-show & ng-hide
- Modules:
  - Module Loading and Dependencies
  - Configuration Blocks
  - Run Blocks
  - $provide service
- Services & Factories:
  - Services Overview
  - Modularity using Services
  - Injecting Services
  - Creating Custom Factories and Providers
- Directives:
  - Directives Overview
  - Built in directives
  - Creating Directives
  - The Directive Definition Object
  - Compilation and Linking
  - Creating Components
- Forms:
  - Handling Forms
  - Forms Validations
  - $valid and $error
- Promises:
  - The premise of promises
  - Callbacks and Promises
  - Built in promises
  - Creating custom promises
- HTTP server requests:
  - Communicating over $http
  - Configuring and sending the http Headers
  - Caching Responses
  - Request and Response Transformation
  - Interceptors
  - Using RESTful Resources

- Using $resource Service
- Using Rest angular
- Communication over Web Sockets

- Routing and Views:
  - $route Provider and ng-view
  - Using $location Provider
  - Configuring routes
  - Accessing route values using $routeParams
  - Using Angular-UI's $state Provider
- Angular UI:
  - UI Bootstrap
  - Angular material
  -
- Project:
  - Project will be developed by the participants

Angular 4

- Introduction
- Why Angular?
- TypeScript
  - Getting Started With TypeScript
  - Working With tsc
  - Features
  - Typings
  - Variables
  - Operators
  - Decision Making
  - Loops
  - Functions
  - Classes
  - Interfaces
  - Objects
  - Shapes
  - Type Inference
  - Type Keyword
  - Decorators
- Bootstrapping an Angular Application
  - Understanding the File Structure
  - Bootstrapping Providers
- Components in Angular
  - Creating Components

- Application Structure with Components
  - Passing Data into a Component
  - Responding to Component Events
  - Using Two-Way Data Binding
    Accessing Child Components from Template
- Projection
- Structuring Applications with Components
- Using Other Components

- Directives
  - Attribute Directives
    - NgStyle Directive
    - NgClass Directive
  - Structural Directives
    - NgIf Directive
    - NgFor Directive
    - NgSwitch Directives
    - Using Multiple Structural Directives
  - Creating an Custom Attribute Directive
    - Listening to an Element Host
    - Setting Properties in a Directive
  - Creating a Custom Structural Directive
    - View Containers and Embedded Views
    - Providing Context Variables to Directives

- Advanced Components
  - Component Lifecycle
  - Accessing Other Components
  - View Encapsulation
- Pipes
  - Using Pipes
  - Custom Pipes
  - Stateful Pipes
- Forms
  - Getting Started
  - Template-Driven Forms
    - Nesting Form Data
    - Using Template Model Binding
    - Validating Template-Driven Forms
  - Reactive/Model-Driven Forms
    - FormBuilder Basics
    - Validating FormBuilder Forms
    - FormBuilder Custom Validation
  - Visual Cues for Users

- Observables
    - Using Observables
    - Error Handling
    - Disposing Subscriptions and Releasing Resources
    - Observables vs Promises

    - Using Observables From Other Sources
    - Observables Array Operations
- Angular Dependency Injection
    - What is DI?
    - DI Framework
    - Angular's DI (@Inject() and @Injectable)
- Http
    - Making Requests
    - CRUD Operations
    - Catching Rejections
        - Catch and Release
        - Cancel a Request
        - Retry
- Advanced Angular
    - Directives

- Immutable.js
    - What is Immutability?
    - Object.assign
    - Object.freeze
    - Immutable.Map
        - Map.merge
    - Nested Objects
        - Deleting Keys
        - Maps are Iterable
    - Immutable.List

- Modules
    - What is an Angular Module?
    - Adding Components, Pipes and Services to a Module
    - Creating a Feature Module
    - Directive Duplications
- Routing
- Why Routing?
    - Configuring Routes
    - Redirecting the Router to Another Route
    - Defining Links Between Routes
    - Dynamically Adding Route Components

- Using Route Parameters
- Defining Child Routes
- Controlling Access to or from a Route
- Passing Optional Parameters to a Route

- Angular UI:
  - Angular material
- Project:
  - Project will be developed by the participants