# Routing

* It provides a way for us to navigate from one view to another view in the application
* Allows us to create SPAs (Single Page Application)
* We define the paths that when called upon will quickly swap in that component assigned to that path without refreshing the page.
* You must provide RouterModule that has the necessary service providers and directives for navigating through the application
* You need two components achieve routing

# Angular HttpClient

* It is an API for Angular application that handles HTTP request in order to download/upload data and access other back-end services.
* All HttpClient methods returns an **Observable** of something.
* Note: you must import HttpClientModule in app.module.ts

## Observables

* It is like promises in C#
  + Promise is a place holder for a future value that handles a single event when an asynchronous operation complete or failure
  + Not cancellable and has only 1 activation at a time.
* It follows a publisher and subscriber model
  + It defines a function for publishing values, but it will not execute until a consumer is subscribe to it
* It can have 0 or 1 or many activations
* It has an array that stores the data that arrive asynchronous over time
* It can help us manage our asynchronous data that is coming from the back-end server

## Publisher and subscriber model

* Describes the flow of message between applications
* A message is published by the **publisher** to an **event channel**, to any **Subscribers** that is monitor to that **Channel** will consume that message

# Services

* It is a class that is not dependent on any component.
* They are used to share data/logic across multiple components and encapsulate external interactions.
* Services, if properly used, will be singletons
  + Meaning they will only have one instance always exist.
* It will be injected into multiple components and those components can utilize the functions defined in the service class.

# Dependency Injection in Angular

* It is a design pattern where we declare the dependencies in the constructor of the dependent class.
* In Angular, Angular Injector is the one responsible for handing the initializing of the class the the application will need and handle all the dependencies that an application might use.
* Any class that has Injector and component decorator will be handled by the Injector.
* In this case, we are giving up control over part of our application to the Angular Injector.

# Angular Forms

* There two types of Forms in Angular.
* You must import FormsModule in app.module.ts

## Template Driven Forms

* All validation form element is all defined in the HTML.
* Uses two-data binding
* Directives:
  + ngModel – Allows us to two-way data bind the html and component variable
  + ngForm – Assigned a reference variable to the entire form

## Reactive Forms

* All form validation will be handled in the component class.
* Directives:
  + formGroup – Assigns the form group in the typescript file to the html
  + formControl – Assigns the form control in that form group from the typescript file to the html file
* Easier to do data binding
* Easier to create flexible and complex validation requirements in your form
* You must import ReactiveFormsModule in app.module.ts

## Validation in Forms in Angular

* Angular forms it will have 3 different states of a form control
* Form control has been visited
  + False – ng-untouched
  + True – ng-touched
* Form control value has been changed
  + False – ng-pristine
  + True – ng-dirty
* Form control’s value is valid
  + False – ng-invalid
  + True -ng-valid

# Pipes

* They provide a way to transform values before it is displayed
  + “transform” you can change, modify, or reformat the values into what you want.
* They are used in conjunction with interpolation so syntax “{{targetData | pipeType }}”