Angular Quick Reference

Single Page Application
- Web application that fits in a single page
- Navigation between pages performed without page reload
- Can be implemented using HTML and AJAX
- Example apps
  - Gmail
  - Google Maps
  - Facebook
- Frameworks to build SPA
  - AngularJS
  - Angular
  - Ember.js
  - ExtJS
  - Knockout.js
  - Meteor.js
  - ReactJS
  - Vue.js
- Development is done in TypeScript

Node.js
- Open-source runtime environment to build server-side application using JavaScript
- Global companies like Netflix, Facebook use Node.js as a server.

Node Package manager (npm)
- Node package manager helps to package JavaScript applications
- npm is used as tool to build and package angular applications
- Node.js and npm helps to compile TypeScript files to JavaScript

Command Line Interface (CLI)

New Project
`ng new [project-name]`

Start App
`ng serve`

Create Component
`ng generate component [component-name]`

Create Service
`ng generate service [service-name]`

Create Pipe
`ng generate pipe [pipe-name]`

Create Guard
`ng generate guard [guard-name]`

package.json
- Configuration file for JavaScript applications
- Helps to define dependencies
- A default configuration file can be created using the following command.

  `npm init`

TypeScript
- Typed superset of JavaScript
- TypeScript files saved with .ts extension
- TypeScript gets compiled to JavaScript
- Command to install TypeScript
  `npm install -g typescript`

- Command to compile TypeScript to JavaScript:
  `tsc hello.ts`

- TypeScript supports inheritance
- TypeScript supports access modifiers:
  - private
  - public
  - protected

Structural Directives
- Used to manipulate DOM
- Directives
  - ngIf – Displays an element if expression is true
  - ngFor – Repeats an element
  - ngSwitch – Similar to switch statement, additionally requires ngSwitchCase and ngDefault.

Attribute Directives
Change the appearance or behavior of DOM using HTML element attributes.

Built-in attribute directives
- ngClass
  `[ngClass]="Boolean-expression ? 'class-name' : 'another-class-name'"`

- ngStyle
  `[ngStyle]="{ 'color' : movie.bookingsOpen ? 'green' : 'red'}"`

- ngModel
  `[(ngModel)]="[component-property]"`

Pipes
Transform display of data from one format to another.
Built-in pipes
- DatePipe
- UpperCasePipe
- LowerCasePipe
- CurrencyPipe
- DecimalPipe
- PercentPipe
- JsonPipe

DatePipe example
`{{movie.budget | currency:'USD':'symbol':'3.0'}}`

Custom Pipe
Create a class with @Pipe decorator that implements PipeTransform interface.

Example code to transform an array to appropriate display.

Step #1: array-transform.pipe.ts

```
import { Pipe, PipeTransform } from "@angular/core";

@Pipe({
    name: 'arrayTransform'
})
export class ArrayTransformPipe implements PipeTransform
{
    transform(array:string[], separator:string):string {
        return array.join(' ' + separator + ' ');
    }
}
```

Step #2: Include pipe in app.module.ts under declarations:

```
@NgModule({
  declarations: [
    AppComponent,
    ArrayTransformPipe,
    MovieComponent
  ],
```

Step #3: Define an array of string as property of component

```
fruits: string[] = ['apple', 'orange', 'grapes'];
```

Step #4: Definition in Template

```
{{fruits | arrayTransform : '|'}}
```

One way data binding
Data traverses one way, either from component to template or from template to component.

One way data binding can be implemented using:

- Interpolation
- Property Binding
- Event Binding

Interpolation
Data transfers from component to view.
`{{component-property}}`

Property Binding
Assign component property value to an HTML element attribute using square brackets.
`[attr-name]="component-property"`

Event Binding
Any event triggered from the view to the component by defining event in normal brackets as a HTML element attribute.
`(click)="buttonClicked()"`

Two way data binding
Achieved by combining Property Binding and Event Binding.

- Add FormsModule in the import section of app.module.ts
- Syntax to implement two way binding. Here 'city' is a component property.
  `<input type="text" [(ngModel)]="city">`

Services

- Services help to organize share a business logic or feature across various component.
- Angular services are singleton implementation and can be injected using @Injectable decorator.
- Steps to implement
  - Generate service
    `ng generate service [service-name]`
  - Include a function in the generated service class
  - Inject the service in the constructor of the component:
    `constructor(private empService:EmpService) {}`

HTTP Client
An API to make AJAX and REST API calls. Steps to implement:

- Include HttpClientModule in app.module.ts under the imports section.
- Inject HttpClient in Service class constructor.
- Methods get(), post(), put() and delete() in HttpClient help to make REST API call.
- Refer example below for post:

```
addUser(user: string): Observable<any> {
  let header : HttpHeaders = new HttpHeaders();
  header.set('Content-Type', 'application/json');
  return this.httpClient.post<any>(this.addUrl, user,
    {headers: header});
}
```

- The first parameter of the function is the endpoint.
- The second parameter is included in the body part of the HTTP request as JSON.
- The third parameter defines the content type header.
- The HttpClient function returns an observable.
- In the component it is subscribed and data is received in an asynchronous way.

```
this.profileService.getUser().subscribe(
  response => {
    this.user = response.data;
  },
  error => {
    this.error = error;
  }
);
```

- The first parameter of subscribe function gets the response data.
- The second parameter of the subscribe function handles the error response.

Routes

- Helps to navigate from one view to another.
- Steps to implement:
  - Define a route in app-routing.module.ts

```
const routes: Routes = [
  { path: 'movie', component: MovieComponent }
];
```

  - Anchor tag in template to navigate to a route:

```
<a routerLink="movie" routerLinkActive="active-link">
    Movie
</a>
```

  - Code in component to navigate to a route:

```
constructor(private router: Router) { }
...
// within any function
this.router.navigate(['/movie']);
```