* I am working as a senior software engineer in Development in a US based MNC LeadVenture. It is in automobile domain catering to 70% B2B market. We also have a SAAS based B2C software product for point of sale. Wholesalers use them across USA and Canada.
* I have a diverse experience of 12+ years in product development and deployment.
* My key skills are Angular 2.x , Building Rest APIs, PHP, MySQL, MVC models, Jquery, Javascript Bootstrap, XML, Rest API

 I have good knowledge of Google Analytics and Google Webmaster tools.

Prior to this organization I was working as a Manager Technology in a startup called CollegeSearch in education space for five and a half years,

* I am a graduate from Delhi university and MCA from MDU.
* I have good knowledge of Angular framework having worked on directives, database binding, setting up local and production environments. I have knowledge of Components, Directives, environment, angular.json, dependencies, services.

**Angular Versions** : Worked on Angular 8.0.3 and Typescript version 3.4

Difference between ng serve and ng build

|  |  |
| --- | --- |
| **ng serve** | **ng build** |
| The command does not generate an output folder. | The output folder is – dist/. |
| The ng serve builds artifacts from memory instead for a faster development experience. | The ng build command generates output files just once and doesn't serve them. |

**Types of Decorators:**

* Class decorators like @Component, @NgModule
* Property decorators like @Input and @Output
* Method decorators like @HostListener
* Parameter decorators like @Injectable

**Dependency Injections:**

Dependencies are services or objects that a class needs to perform its function. Dependency injection, or DI, is a design pattern in which a class requests dependencies from external sources rather than creating them.

**Pipes**: A pipe takes in ***data as input*** and transforms it into an ***output***.

The pipe’s purpose is to **allow the transformation of an existing value** and **reusability**!

**Types of Pipes:**

Date

Uppercase

Lowercase

Currency

Decimal

PercentPipe

**AOT**

Since the application compiles before running inside the browser, the browser loads the executable code and renders the application immediately, which leads to **faster rendering**.

Synchronous and asynchronous Validators are very similar - the main difference is that a sync Validator returns an error object instance directly, while the **async version returns an Observable of the the same object**. The most common use case for async Validators is doing a server validation via an HTTP Callback.

@Component({

  selector: 'partners-client-add',

  templateUrl: './client-add.component.html',

  styleUrls: ['./client-add.component.scss']

})

**A class can be instantiated i.e**, objects of a class can be created. An Interface cannot be instantiated i.e, objects cannot be created. Classes does not support multiple inheritance. Interface supports multiple inheritance.

