

**Master Copy**

Contents

**[1](#_Toc493181508)****[Prerequisites](#_Toc493181508)** [3](#_Toc493181508)

**[1.1](#_Toc493181509)****[Language Prerequisites](#_Toc493181509)**[: 3](#_Toc493181509)

**[1.2](#_Toc493181510)****[Architecture / Concept Prerequisites](#_Toc493181510)** [3](#_Toc493181510)

**[1.3](#_Toc493181511)****[Software Prerequisites](#_Toc493181511)** [3](#_Toc493181511)

**[1.4](#_Toc493181512)****[Access Prerequisites](#_Toc493181512)** [3](#_Toc493181512)

**[2](#_Toc493181513)****[Most Important Concepts](#_Toc493181513)** [3](#_Toc493181513)

**[2.1](#_Toc493181514)****[Dependency Injection](#_Toc493181514)** [3](#_Toc493181514)

**[2.2](#_Toc493181515)****[JavaScript Based Framework](#_Toc493181515)** [3](#_Toc493181515)

**[2.3](#_Toc493181516)****[Single Page Application Framework](#_Toc493181516)** [4](#_Toc493181516)

**[2.4](#_Toc493181517)****[Typescript Language](#_Toc493181517)** [4](#_Toc493181517)

**[3](#_Toc493181518)****[Node.js , npm Architecture Model](#_Toc493181518)** [4](#_Toc493181518)

**[3.1](#_Toc493181519)****[Traditional Web Application Model](#_Toc493181519)** [4](#_Toc493181519)

**[3.2](#_Toc493181520)****[Nodejs Model – Single Threaded Event Loop](#_Toc493181520)** [4](#_Toc493181520)

**[3.3](#_Toc493181521)****[Node.js basic](#_Toc493181521)** [5](#_Toc493181521)

**[3.4](#_Toc493181522)****[Npm (Node Package manager)](#_Toc493181522)** [5](#_Toc493181522)

**[3.5](#_Toc493181523)****[Nvm (Node version Manager)](#_Toc493181523)** [5](#_Toc493181523)

**[4](#_Toc493181524)****[Node.js , npm Installation](#_Toc493181524)** [5](#_Toc493181524)

**[5](#_Toc493181525)****[IDE](#_Toc493181525)** [6](#_Toc493181525)

**[6](#_Toc493181526)****[Typescript](#_Toc493181526)** [6](#_Toc493181526)

**[7](#_Toc493181527)****[Angular JS 1.x (Old) Basics](#_Toc493181527)** [6](#_Toc493181527)

**[8](#_Toc493181528)****[Angular (2+, 4) Basics](#_Toc493181528)** [7](#_Toc493181528)

1. **Prerequisites**
   1. **Language Prerequisites**:

* JavaScript – Basic Level
* Command line scripting
* HTML / HTML5
* CSS
* jQuery – Basics (Not required thought for Angular)
  1. **Architecture / Concept Prerequisites**
* Object Oriented Programming
  1. Specifically in JavaScript the concept of inheritance
* MVC Framework – Basics
* MVVM Framework – Basics
* DOM (Document Object Model)
* Dependency Injection (DI) Design Pattern - you **can't** build an Angular application without it
* AJAX – Asynchronous JavaScript and XML
  1. **Software Prerequisites**
* Visual Studio Code – May be Installed – IDE
* Minimum Node version 6.9.x LTS (Long Term Support version) installer – prefer to install it by myself and need admin access to the PC
* Google Chrome version 59+
* Prefer Windows OS version 7+
  1. **Access Prerequisites**
* Admin access in the PC
* Internet access
* Download Access of packages from github and via npm command
* A board and marker / chalk / pen to write on

1. **Most Important Concepts**
   1. **Dependency Injection**

Best Site with Example: <https://blog.sstorie.com/experimenting-with-angular-2-dependency-injection/>

It is the core concept of Angular which lets you understand how Angular provider instances work, number of instances created per component depending on the level of provider definition.

Remember “**Bubbling Up**”

Injectable Service is always injected via **Constructor Injection**.

* 1. **JavaScript Based Framework**
* JavaScript mainly is used for client side programming (Although it has evolved to be used for ser-side programming as well)
* JavaScript deals with dynamic content, e.g., it can detect the device and render the particular device version of the website
  1. **Single Page Application Framework**

Provides more dynamic interactions, rather than a full page refresh, an AJAX way to communicate and Update a section of a page.

Resembles more like a mobile / desktop application

An example is Gmail

* 1. **Typescript Language**

The language is used in Angular and thereby Ionic

1. **Node.js , npm Architecture Model**
   1. **Traditional Web Application Model**

It is also called a Request Response Stateless Model which uses HTTP protocol (Stateless Protocol).

1 Client Request to Web Server

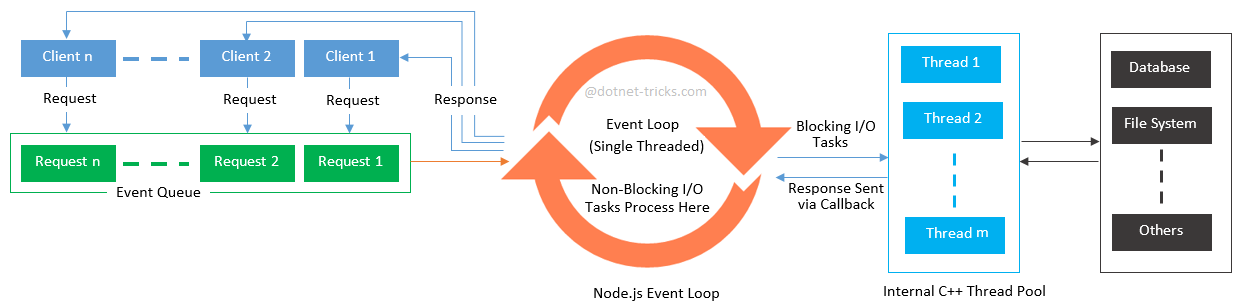
1 Thread assigned from the Thread Pool by Web Server

In this model:

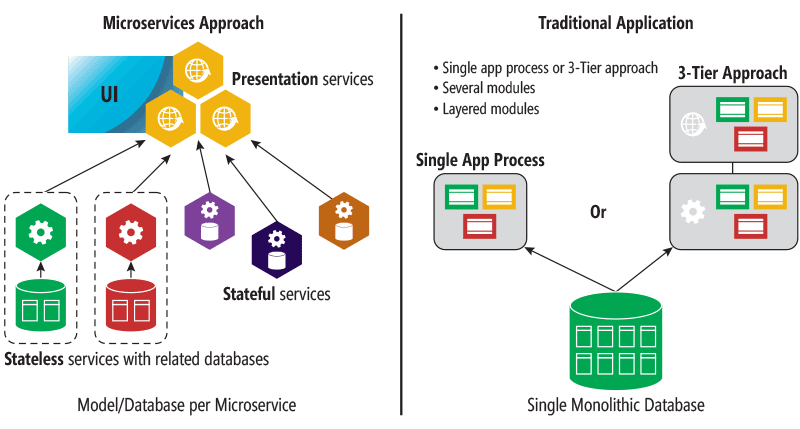
* Server waits in infinite loop
* Assigns one thread to one client request, which takes up system RAM and eventually maxing out as per RAM size
* Blocking I/O request, it stalls the current thread, if multiple threads are stalled, then either more threads are created by Thread pool or waits for available thread
  1. **Nodejs Model – Single Threaded Event Loop**

The root of nodejs processing model is the ‘**event loop**’ which is single threaded.

* All client requests are stored in an ‘Event Queue’. The ‘Event Loop’ which is single threaded is responsible for processing those requests. If the requests are non-blocking (i.e., non I/O) then the event loop itself processes and creates the response. If however it is blocking then it sends it to a separate I/O thread from thread pool and continues running itself.



* Node.js takes the advantage of JavaScript Callbacks and the coding should also take the advantage
* Best suited for Micro services Architecture



**Cons of Node.js**:

* It is not designed for heavy computations (CPU intensive operations)
* If the code is not written properly, it can create “**callback hell**” (nested callbacks several levels deep).
  1. **Node.js basic**

It is an open source server framework which allows JavaScript to run on server. It simplifies the development of web applications by providing a rich library of JavaScript modules. Most important features of node.js:

* **Asynchronous API** : All APIs in nodejs are asynchronous. It means an API call from nodejs server will never wait for the API response.
* **Event-based Response** : A response from an API is captured via notification mechanism of Events
* **Fast Compilation** : Node.js is created from Google’s V8 JavaScript engine
* **Lightweight**
* Easily configurable and highly customizable

Creator Ryan Dahl

* 1. **Npm (Node Package manager)**

npm is a CLI for managing node modules and packages. It basically downloads packages from an online database called **npm registry** which is an online website (<https://www.npmjs.com/> ).

* 1. **Nvm (Node version Manager)**

nvm is a command line interface (CLI) to install and maintain different versions of nodejs in the system. It allows running multiple version of node in isolated environment.

1. **Node.js , npm Installation**

Visit the node.js website <https://nodejs.org/en/> and download the LTS (Long Term Support) if you are not planning on development on node.js modules itself. LTS is the most stable version used by all the companies.

|  |
| --- |
| **Hands On** |
| The installation is pretty basic stuff.  After installation check the versions by opening a command prompt and executing the following commands   * npm –v * node –v   A simple way to test node.js is working is:   * Create a folder ‘TestNode’ * ‘cd’ to the folder * Create a simple javascript file called ‘helloworld.js’ * Add the following code:   Console.log(‘Hello World’);   * From the command line (with the current directory as ‘TestNode’) execute: * node helloworld.js   **NOTE:** Do not installed it too deep from the root or inside a long folder name(s) path, it sometimes do not work. (A known issue) |

1. **npm Scope**

When you want to publish multiple packages together under a single name (say a company wants to publish multiple packages), one uses a scope.

A scope name is preceded by @. A package name under a scope maybe defined as,

@scopename/packagename

An example will be angular and ionic packages we will be using in this course. Some common angular packages we will use like,

@angular/cli

@angular/http

@angular/common

@angular/forms

@angular/animation

[@angular/material](https://www.npmjs.com/package/@angular/material)

@angular/core

1. **IDE**

There are many free editors which can be used:

* Visual Studio Code
* Atom
* Sublime Text 3
* Visual Studio 2017
* Angular IDE
* Brackets
* …

Personally I prefer **Visual Studio Code** (*It’s just a personal preference*!!), It is built on Electron Framework.

Electron is a cross platform framework to build native applications with web technologies like JavaScript, HTML and CSS.

|  |
| --- |
| **Hands On** |
| Install Visual Studio Code (If not installed already) |

1. **Typescript**

|  |  |
| --- | --- |
| License | Open Source Apache 2 |
| Framework | Super-set of JavaScript |
| Maintained By | Microsoft |
| First Release | October 2012 |
| Compiles To | JavaScript |
| Standard | Follows ECMAScript 2015 Standard to some extent |

ECMA (**European Computer Manufacturers Association**) standard was created to standardize java script. The ECMAScript specification is a standardized specification of a scripting language **developed by [Brendan Eich](https://en.wikipedia.org/wiki/Brendan_Eich" \o "Brendan Eich) of [Netscape](https://en.wikipedia.org/wiki/Netscape_Communications_Corporation" \o "Netscape Communications Corporation)**.

Note all browser still do not support ECMAScript2015 (ES6) standard but instead supports an older version of ECMA Standard called ES5 published 2009.

Current version of Typescript --> 2.5

**Advantages of using Typescript are**:

* OOPS concept support - partially, those who are familiar with OOPS can dive into typescript easily, e.g., class, modules, namespaces, interface support
* Static type checking (compile time type checking)
  1. **Installation**

|  |
| --- |
| **Hands On** |
| npm install -g typescript |

* “-g” – global saves it in the global “npm” cache in windows which is generally “%AppData%\npm\node\_modules”
* The package includes the “**t**ype**s**cript to javascript **c**ompiler” (tsc)

**Rest is in Typescript\_Training1.1.docx**

1. **Angular JS 1.x (Old) Basics**

|  |  |
| --- | --- |
| Language | JavaScript |
| License | Open Source MIT |
| Framework | Front End Web Application Framework (Client Side) |
| Maintained By | Google’s Angular Team |
| First Release | October 2010 |

* AngularJS was one of the first most popular JavaScript frameworks for building Single-page applications
* *“Two-way data binding”* first introduced was a “killer feature”
* Most important feature introduced is the **Dependency Injection**
* It has evolved from only a Single Page Application framework to multi-page framework development as well.

1. **Angular (2+, 4) Basics**

|  |  |
| --- | --- |
| Language | Typescript |
| License | Open Source MIT |
| Framework | Front End Web Application Framework |
| Maintained By | Google’s Angular Team |
| First Release | September 2016 |
| Platform | Cross Platform |
| Browser latest | IE 11+, Chrome, Firefox, Safari |

A funny name used by the Angular Team for the type of Framework it supports --> **MVW** (Model View Whatever or **MV\***)

Angular gives enough flexibility to separate presentation (UI layer) logic from business logic and presentation state.

Current Angular version is 4.3

1. **Similar Frameworks like Angular**

1. **Miscellaneous**

AngularJS is the frontend part of the **MEAN** stack, consisting of MongoDB database, Express.js web application server framework, Angular.js itself, and Node.js server runtime environment.

1. **Miscellaneous**

[Microsoft Typescript Home](https://www.typescriptlang.org/docs/home.html) – Typescript tutorial and download