



(<https://facebook.com/keyholesoftware>)



(<https://twitter.com/keyholesoftware>)



(<https://www.linkedin.com/company/keyhole-software>)



(<https://www.youtube.com/channel/UCAIUNkXmnAPgLWnqUDpUGAQ>)



(<https://keyholesoftware.com/feed>)

H



[\(HTTPS://KEYHOLESOFTWARE.COM/\)](https://keyholesoftware.com/) > [KEYHOLE DEVELOPMENT BLOG \(HTTPS://KEYHOLESOFTWARE.COM/\)](https://keyholesoftware.com/) > [GETTING STARTED WITH ANGULAR CLI COMMANDS \(HTTPS://KEYHOLESOFTWARE.COM/2017/04/24/GETTING-STARTED-WITH-ANGULAR-CLI-COMMANDS/\)](https://keyholesoftware.com/2017/04/24/getting-started-with-angular-cli-commands/)

[\(HTTPS://KEYHOLESOFTWARE.COM/2017/04/24/GETTING-STARTED-WITH-ANGULAR-CLI-COMMANDS/\)](https://keyholesoftware.com/2017/04/24/getting-started-with-angular-cli-commands/) <  
[ELK-IN-NET-APPLICATIONS/](https://keyholesoftware.com/2017/04/24/getting-started-with-angular-cli-commands/)  
[\(HTTPS://KEYHOLESOFTWARE.COM/2017/04/24/GETTING-STARTED-WITH-ANGULAR-CLI-COMMANDS/\)](https://keyholesoftware.com/2017/04/24/getting-started-with-angular-cli-commands/) >  
[FOR-NET-DEVELOPERS/](https://keyholesoftware.com/2017/04/24/getting-started-with-angular-cli-commands/)

# Getting Started With Angular CLI Commands



□ BRETT SMITH ([HTTPS://KEYHOLESOFTWARE.COM/AUTHOR/BSMITH/](https://keyholesoftware.com/author/bsmith/)) / 📅 APRIL 24, 2017 /  
🔖 ANGULAR ([HTTPS://KEYHOLESOFTWARE.COM/CATEGORY/TECHNOLOGY-SNAPSHOT/JAVASCRIPT/ANGULAR/](https://keyholesoftware.com/category/technology-snapshot/javascript/angular/)),  
🔖 JAVASCRIPT ([HTTPS://KEYHOLESOFTWARE.COM/CATEGORY/TECHNOLOGY-SNAPSHOT/JAVASCRIPT/](https://keyholesoftware.com/category/technology-snapshot/javascript/)),  
🔖 PROGRAMMING ([HTTPS://KEYHOLESOFTWARE.COM/CATEGORY/GENERAL-PROGRAMMING/](https://keyholesoftware.com/category/general-programming/)),  
🔖 SINGLE-PAGE APPLICATION ([HTTPS://KEYHOLESOFTWARE.COM/CATEGORY/ARCHITECTURE/SINGLE-PAGE-APPLICATION/](https://keyholesoftware.com/category/architecture/single-page-application/)) /  
💬 LEAVE A COMMENT ([HTTPS://KEYHOLESOFTWARE.COM/2017/04/24/GETTING-STARTED-WITH-ANGULAR-CLI-COMMANDS/#RESPOND](https://keyholesoftware.com/2017/04/24/getting-started-with-angular-cli-commands/#RESPOND)).

I have been a developer for the past 20+ years and have used some of the ‘big’ languages (C, C++, C#, Java) throughout. I dabbled with JavaScript back before there were frameworks, when its use was to dynamically hide this control or disable that control. While it can still do those trivial tasks, JavaScript has come a long way with a framework for everything now: front-ends, back-end, sockets, 2D/3D games, connecting to all kinds of databases, and robots, among others.

Over the last few years, I have started looking back into JavaScript. The projects I have been working didn’t require any JavaScript, so I have been trying to pick it up here and there through online tutorials, videos, and podcasts. I chose Angular to get my feet wet. I’m not saying that Angular is the ultimate framework and that everyone should use it, but it can most certainly do quite a few things.

I keep coming back to a tool that has helped my journey through Angular-land: **the Angular CLI**. The CLI is a useful tool that can help set up and add different elements to your projects. It follows some of the best practices that have been laid down by the Angular team, even handling some of the plumbing for you so that things will work well together.

This post will show some of the basic commands available within the CLI. While we won’t go into every different command provided, we will look at some of the basic commands that can help get a project started and built. We will pay particular attention to the different commands and what they produce, as far as application structure and file layout is concerned.

So do keep in mind that it is not my purpose to describe all the different elements and how they are used (maybe a topic for future articles); we will just be looking at what the CLI lays down. Let’s get started.



# What is Angular CLI?

Angular CLI (<https://github.com/angular/angular-cli>) is a command line tool that can aid in creating new Angular projects from scratch or adding various elements to an existing Angular application. This tool is not essential in building an Angular project, but it does provide several benefits, especially for someone who does not have much experience with Angular.

This tool will create a new project that is ready to run immediately and will generate a *Hello World-esque* project. It will create all needed plumbing to get everything up and running in a matter of minutes, generating an application structure that is based on best practices for an Angular project. When adding new elements, it will create these elements in the appropriate directory structure, generate source code, and in some cases add code to other elements within the project so that the new elements can be used wherever needed.

## Prerequisites

In order to install Angular CLI, the following should be installed in the development environment:

- node (<https://nodejs.org/en/>) (at least version 4.0)
- npm (<https://www.npmjs.com/>) (at least version 3.0)


## Installation

Installing Angular CLI is as easy as typing the follow command:

```
npm install -g angular-cli
```

This will globally install angular-cli. This will allow for the use of all the `ng` commands. As of this writing, this installs version 1.0.0-beta.28.3 of the Angular CLI.





[\(https://keyholesoftware.com/2017/12/18/angular-developer-javascript-to-typescript/\)](https://keyholesoftware.com/2017/12/18/angular-developer-javascript-to-typescript/).

## Create New Project

To create a new project, either `ng new` or `ng init` can be used. The main difference between these commands is that `ng new` will create a new directory and `ng init` will create a new project in the current directory.

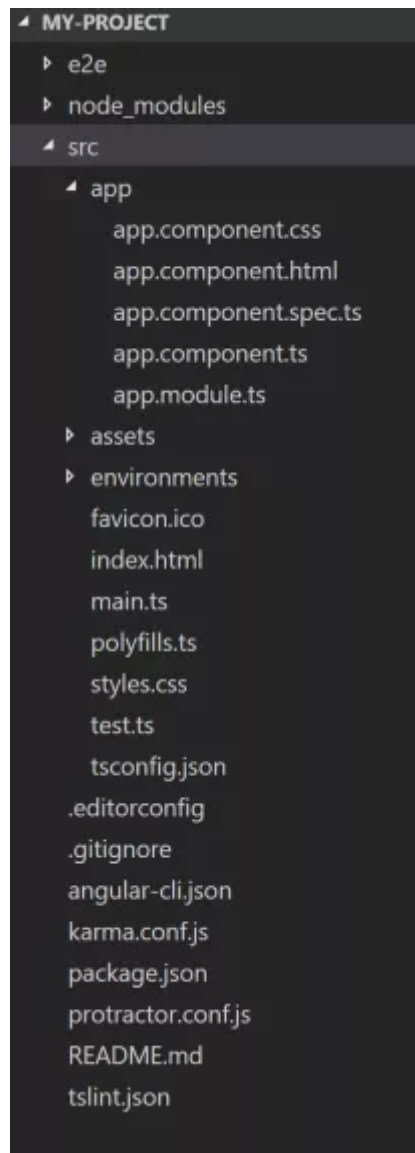
Issue the following command to create a new project called `my-project`.

```
ng new my-project
```

This will cause several things to happen:

- directory `my-project` will be created
- directory structure and source files will be generated
- any needed dependencies will be installed
- TypeScript will be configured
- Karma test runner will be configured
- Protractor end-to-end test runner will be configured
- environment files will be created

Here is a screenshot of the project structure that was generated:



The purpose of this article is not to go into all the files that are generated but there are a few keys files to point out:

- `src/app` directory: This directory contains source code, css, and html files. This is where most of the work will be done.
- `app.module.ts`: Main `NgModule` for the application. Everything must belong to a module (even other modules).
- `main.ts`: Bootstraps your application.

- `index.html` : Main html file for your application.

## Run Application

Change to the `my-project` directory and run the following:

```
ng serve
```

Then open a browser and point it to `http://localhost:4200/`. The following should be displayed:



Nothing spectacular, but a running application nonetheless.

Note: Running `ng serve` from the command line will not return control back to the command prompt. This will detect changes that are made to the project, recompile the project, and then refresh the web page.

## Adding Other Elements

To add other elements to your application, either `ng generate` or `ng g` command can be used. There are many elements that can be created, but only the following elements will be covered in this article:

- class
- interface
- enumeration
- component
- service
- module

In order to use Angular CLI to add these elements to your application, go to the `my-project` directory and issue the following commands.

Note: After issuing commands, Angular CLI will display files that are generated and where they are created at.

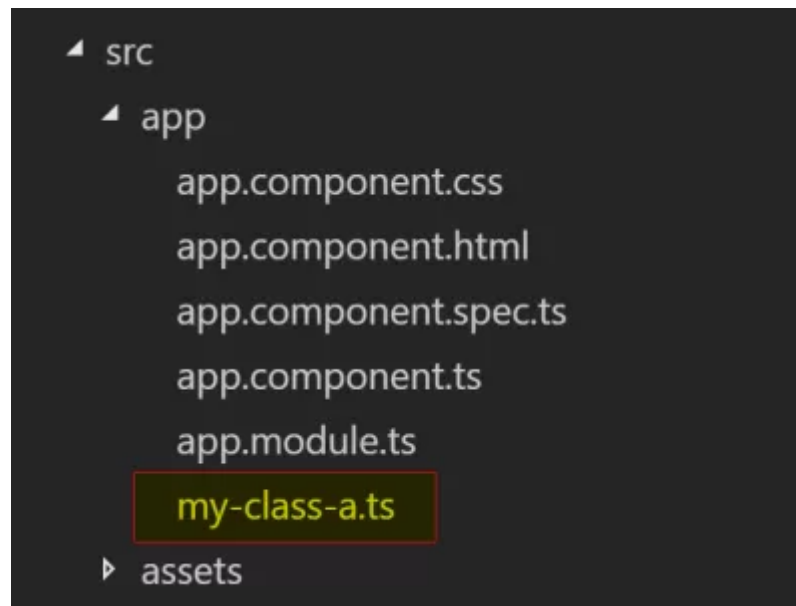
## Create A Class

To create a new class, run the following:

```
ng g class my-class-a
```

This will create a file called `my-class-a.ts` under the `src/app` directory. Notice that it will add `.ts` after the name provided.





This file is a basic class file with no decorators generated. It will export a class called `MyClassA` (removes dashes and uses camel case).

## my-class-a.ts

```
1 | export class MyClassA {  
2 | }
```

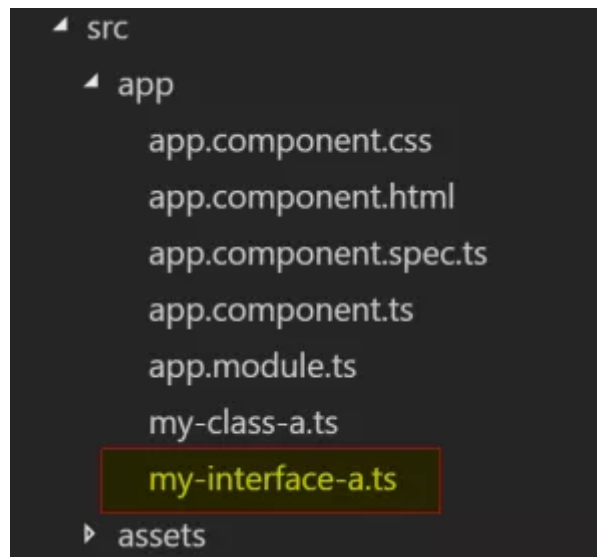
## Create An Interface

To create a new interface, run the following:

```
ng g interface my-interface-a
```

This will create a file called `my-interface-a.ts` under the `src/app` directory. Notice that it will add `.ts` after the name provided.





This file is a basic interface file with no decorator generated. It will export a class called `MyInterfaceA` (removes dashes and uses camel case).

## my-interface-a.ts

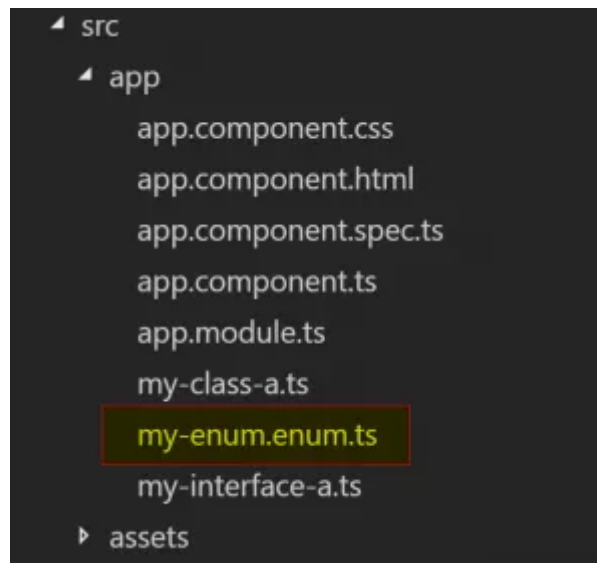
```
1 | export interface my-interface-a
2 | }
```

## Create An Enumeration

To create a new enumeration, run the following:

```
ng g enum my-num
```

This will create a file called `my-enum.enum.ts` under the `src/app` directory. Notice that it will add `.enum.ts` after the name provided.



This file is a basic enumeration file with no decorator generated. It will export a class called `MyEnum` (removes dashes and uses camel case).

## my-enum.enum.ts

```
1 | export enum MyEnum {  
2 |   }  
  |
```

Creating a class, interface, or enumeration are basic constructs and could be created manually just as easily, but the CLI will format them according to best practices. Now on to things that need a little more to them.

**See Also:** The Executable Code Review

(<https://keyholesoftware.com/2017/10/04/the-executable-code-review/>).

To create a new component, run the following:

```
ng g component my-component-a
```

This will do several things:

- create a directory called `my-component-a` under `src/app` directory
- generate four files under that directory
  - `my-component-a.component.css`
    - Contains any css that would be needed for this component
    - Optional file that is pointed to by the `component.ts` file
  - `my-component-a.component.html`
    - Contains any html that would be needed for this component
    - Optional file that is pointed to by the `component.ts` file
    - html could be contained within the `component.ts` file, if desired
  - `my-component-a.component.spec.ts`
    - unit test skeleton for this component
  - `my-component-a.component.ts`
    - exports a class called `MyComponentAComponent`
    - implements an interface called `OnInit`
    - generates empty function called `ngOnInit` for `OnInit` interface
    - generates empty constructor function
    - decorates class with `@Component`
      - add selector for component, `app-my-component-a`
      - adds `templateUrl`, points to generated html file for component
      - adds `styleUrls` array, points to generated css file for component
- modifies `app.module.ts` file, added `MyComponentAComponent` to declarations (every component has to belong to a module)



```
└─ src
  └─ app
    └─ my-component-a
      my-component-a.component.css
      my-component-a.component.html
      my-component-a.component.spec.ts
      my-component-a.component.ts
    └─ my-module-a
      └─ my-subcomponent-a
        my-subcomponent-a.component.css
        my-subcomponent-a.component.html
        my-subcomponent-a.component.spec.ts
        my-subcomponent-a.component.ts
      my-module-a.module.ts
      app.component.css
      app.component.html
      app.component.spec.ts
      app.component.ts
      app.module.ts
      my-class-a.ts
      my-enum.enum.ts
      my-interface-a.ts
      my-service-a.service.spec.ts
      my-service-a.service.ts
  └─ assets
```

my-component-a.component.ts



```
1 import { Component, OnInit } from '@angular/core';
2
3 @Component({
4   selector: 'app-my-component-a',
5   templateUrl: './my-component-a.component.html',
6   styleUrls: ['./my-component-a.component.css']
7 })
8 export class MyComponentAComponent implements OnInit {
9
10   constructor() { }
11
12   ngOnInit() {
13   }
14
15 }
```

app.module.ts



```

1 import { BrowserModule } from '@angular/platform-browser';
2 import { NgModule } from '@angular/core';
3 import { FormsModule } from '@angular/forms';
4 import { HttpClientModule } from '@angular/http';
5
6 import { AppComponent } from './app.component';
7 import { MyComponentAComponent } from './my-component-a/my-component-a.component';
8
9 @NgModule({
10   declarations: [
11     AppComponent,
12     MyComponentAComponent
13   ],
14   imports: [
15     BrowserModule,
16     FormsModule,
17     HttpClientModule
18   ],
19   providers: [],
20   bootstrap: [AppComponent]
21 })
22 export class AppModule { }

```

## Create A Service

To create a new service, run the following;

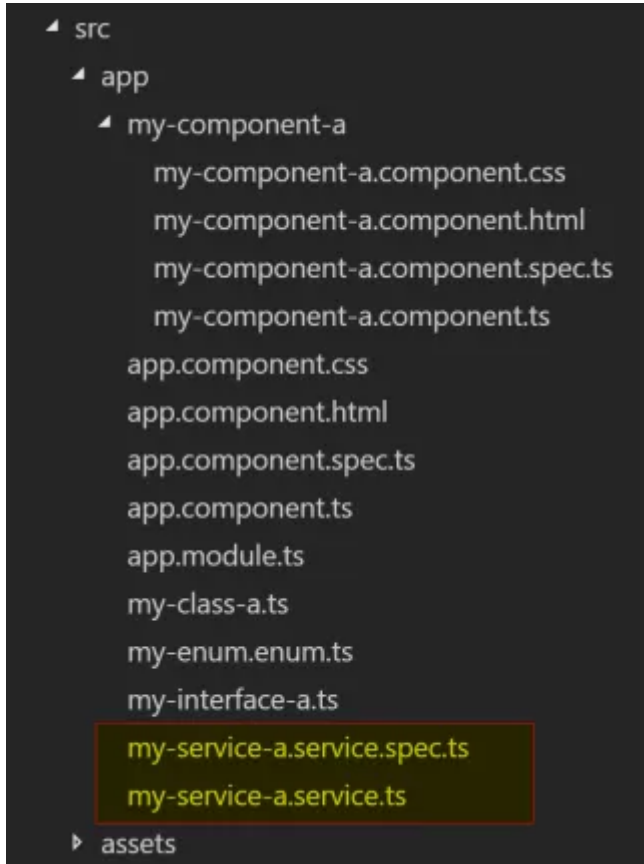
```
ng g service my-service-a
```

This will generate a couple of files under the `src/app` directory:

- `my-service-a.service.spec.ts`
  - unit test skeleton for this service
- `my-service-a.service.ts`
  - exports a class called `MyServiceAService`
  - generates empty constructor function



- decorates class with `@Injectable`



## my-service-a.service.ts

```
1 import { Injectable } from '@angular/core';
2
3 @Injectable()
4 export class MyServiceAService {
5
6     constructor() { }
7
8 }
```



# Create A Module

To create a new module, run the following:

```
ng g module my-module-a
```

This will do a couple things:

- create a directory `my-module-a` under `src/app`
- generate a file under that directory called `my-module-a.module.ts`
  - exports a class name `MyModuleAModule`
  - decorates that class with `@NgModule`





```
└─ src
  └─ app
    └─ my-component-a
      my-component-a.component.css
      my-component-a.component.html
      my-component-a.component.spec.ts
      my-component-a.component.ts
    └─ my-module-a
      my-module-a.module.ts
    app.component.css
    app.component.html
    app.component.spec.ts
    app.component.ts
    app.module.ts
    my-class-a.ts
    my-enum.enum.ts
    my-interface-a.ts
    my-service-a.service.spec.ts
    my-service-a.service.ts
  └─ assets
```

my-module-a.module.ts



```
1 import { NgModule } from '@angular/core';
2 import { CommonModule } from '@angular/common';
3
4 @NgModule({
5   imports: [
6     CommonModule
7   ],
8   declarations: []
9 })
10 export class MyModuleAModule { }
```

## Create Component In A Module

Components can be added to generated module by changing to the module directory:

```
cd src/app/my-module-a
```

```
ng g component my-subcomponent-a
```

or by prefixing the module name to the front of the new component name:

```
ng g component my-module-a/my-subcomponent-a
```

This will do several things:

- create a directory `my-subcomponent-a` under the `src/app/my-module-a` directory
- generate all the component files under this directory (see [Create a component](#) section for description of files)
- add `MySubcomponentAComponent` to the `my-module-a.module.ts` file



```
└─ src
  └─ app
    └─ my-component-a
      my-component-a.component.css
      my-component-a.component.html
      my-component-a.component.spec.ts
      my-component-a.component.ts
    └─ my-module-a
      └─ my-subcomponent-a
        my-subcomponent-a.component.css
        my-subcomponent-a.component.html
        my-subcomponent-a.component.spec.ts
        my-subcomponent-a.component.ts
      my-module-a.module.ts
      app.component.css
      app.component.html
      app.component.spec.ts
      app.component.ts
      app.module.ts
      my-class-a.ts
      my-enum.enum.ts
      my-interface-a.ts
      my-service-a.service.spec.ts
      my-service-a.service.ts
  └─ assets
```

my-module-a.module.ts



```
1 import { NgModule } from '@angular/core';
2 import { CommonModule } from '@angular/common';
3 import { MySubcomponentAComponent } from './my-subcomponent-a/my-subcomponent-a.component';
4
5 @NgModule({
6   imports: [
7     CommonModule
8   ],
9   declarations: [MySubcomponentAComponent]
10 })
11 export class MyModuleAModule { }
```

This same pattern can be used for a class, interface, enum, service, or even another module.

## Summary

As you can see, the Angular CLI has a lot to offer. It will create something that will get you up and running within a matter of minutes. It will follow best practices as related to layout and naming conventions. It will allow you to add elements where they are needed and even do some of the plumbing for you.

The CLI is not something that you have to use, but it is definitely helpful. If you haven't used it before, it is worth checking out especially for someone getting their feet wet.

## Topics We Write About

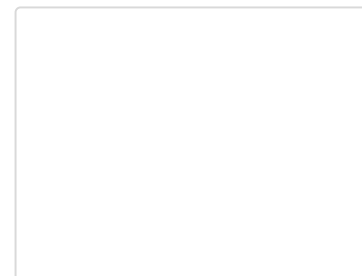
### Microservices

(<https://keyholesoftware.com/category/architecture/microservices>)

### Blockchain

(<https://keyholesoftware.com/category/architecture/blockchain>)

## Recent Posts



Privacy Policy



### JavaScript

(<https://keyholesoftware.com/category/technology-snapshot/javascript/>)

### React

(<https://keyholesoftware.com/category/technology-snapshot/javascript/react/>)

### Java

(<https://keyholesoftware.com/category/technology-snapshot/java-tools/>)

### .NET

(<https://keyholesoftware.com/category/technology-snapshot/net/>)

### Azure

(<https://keyholesoftware.com/category/technology-snapshot/cloud/azure/>)

### AWS

(<https://keyholesoftware.com/category/technology-snapshot/cloud/aws/>)

### Generate Strongly Typed React Components with GraphQL

SEPTEMBER 24, 2018

(<https://keyholesoftware.com/generate-strongly-typed-react-components-with-graphql/>)

### Keyhole Announces Gold Dev Up 2018 Sponsorship & Speaker

SEPTEMBER 20, 2018

(<https://keyholesoftware.com/keyhole-announces-gold-dev-up-2018-sponsorship-speaker/>)

### State Management with MobX and React

SEPTEMBER 17, 2018

(<https://keyholesoftware.com/state-management-with-mobx-and-react/>)

### Keyhole Labs Releases Open Source Blockchain Browser For Real-Time Metrics

SEPTEMBER 12, 2018

(<https://keyholesoftware.com/keyhole-labs-blockchain-browser/>)

## ABOUT THE AUTHOR



## Brett Smith

I have been a software developer for the last 20+ years. I have used the Microsoft .NET technologies as well as Java across several industries (retail, education, pharmacy, document management, government).

I have done desktop and web development as well as dabbling in some Android development. Recently

Privacy Policy



have been learning JavaScript and Angular.

---

SHARE THIS POST



(mailto:?  
subject=http://Getting%20Started%20Wit  
thought  
you  
might  
enjoy  
this!  
Check  
it  
out  
when  
you  
have



a

chance:

<https://keyholesoftware.com/2017/04/24/>

started-

with-

angular-

cli-



commands/).

[ANGULAR \(HTTPS://KEYHOLESOFTWARE.COM/TAG/ANGULAR/\)](https://keyholesoftware.com/tag/angular/)

[ANGULARCLI \(HTTPS://KEYHOLESOFTWARE.COM/TAG/ANGULARCLI/\)](https://keyholesoftware.com/tag/angularcli/)

[ANGULARJS \(HTTPS://KEYHOLESOFTWARE.COM/TAG/ANGULARJS/\)](https://keyholesoftware.com/tag/angularjs/)

[JAVASCRIPT \(HTTPS://KEYHOLESOFTWARE.COM/TAG/JAVASCRIPT/\)](https://keyholesoftware.com/tag/javascript/)

## What Do You Think?

Enter your comment here...

Privacy Policy



---

➤ Proud To Be

# Microsoft Partner

Silver Application Development

(<https://keyholesoftware.com/company/about/microsoft-competency-partner/>).

## Subscribe To Dev Blog

Receive notifications of new posts by email.

Subscribe

 What We're Talking About

Privacy Policy





[Generate Strongly Typed React Components with GraphQL](https://keyholesoftware.com/2018/09/24/generate-strongly-typed-react-components-with-graphql/) (<https://keyholesoftware.com/2018/09/24/generate-strongly-typed-react-components-with-graphql/>)

[Keyhole Announces Gold Dev Up 2018 Sponsorship & Speaker](https://keyholesoftware.com/2018/09/20/keyhole-announces-gold-dev-up-2018-sponsorship-speaker/) (<https://keyholesoftware.com/2018/09/20/keyhole-announces-gold-dev-up-2018-sponsorship-speaker/>)

[State Management with MobX and React](https://keyholesoftware.com/2018/09/17/state-management-with-mobx-and-react/) (<https://keyholesoftware.com/2018/09/17/state-management-with-mobx-and-react/>)

[Keyhole Labs Releases Open Source Blockchain Browser For Real-Time Metrics](https://keyholesoftware.com/2018/09/12/keyhole-labs-blockchain-browser/)  
(<https://keyholesoftware.com/2018/09/12/keyhole-labs-blockchain-browser/>)

## ➤ What We're Tweeting





**Keyhole Software**

@KeyholeSoftware

Modernization Lessons: FTP & the #Mainframe [bit.ly/2f8sSnH](https://bit.ly/2f8sSnH) |  
Mainframe -> Java web with @springframework & #springbatch



Sep 28, 2018



**Keyhole Software**

@KeyholeSoftware

Free Case Study: Blockchain In Hyperledger: Better Than ETL?  
[bit.ly/2vsUr2U](https://bit.ly/2vsUr2U). Includes in-depth walkthrough of a #blockchain  
implemented with @Hyperledger #Fabric with a focus on the value  
blockchain brings to the enterprise #hyperledger #hyperledgerfabric

[Embed](#)

[View on Twitter](#)



 [.https://facebook.com/keyholesoftware](https://facebook.com/keyholesoftware)  [.https://twitter.com/keyholesoftware](https://twitter.com/keyholesoftware)  
 [.https://www.linkedin.com/company/keyhole-software](https://www.linkedin.com/company/keyhole-software)  
 [.https://www.youtube.com/channel/UCAIUNkXmnAPgLWnqUDpUGAQ](https://www.youtube.com/channel/UCAIUNkXmnAPgLWnqUDpUGAQ)  
 [.https://keyholesoftware.com/feed](https://keyholesoftware.com/feed)

© Keyhole Software 2018 + [Content Usage \(/company/creations/content-usage-guidelines/\)](/company/creations/content-usage-guidelines/) Guidelines

