راه اندازی برنامه نویسی محلی

مثال QuickStart live-coding یا download example اصطلاحاً playground هستند. در اینجا بنا نیست که یک برنامه‌ی واقعی را بنویسید. بهتر است که شما در سیستم خودتان به صورت محلی برنامه نویسی کنید، همان طور که ما از شما انتظار داریم که Angular را به این شیوه یاد بگیرید.

راه اندازی یک پروژه‌ی جدید در سیستم خودتان به کمک QuickStart seed کار آسانی است. حتماً Node.js و npm را در سیستمتان نصب داشته باشید.

Clone

مراحل Clone to launch را به کمک دستورات ترمینال زیر اجرا کنید.

```
git clone https://github.com/angular/quickstart.git quickstart
cd quickstart
npm install
npm start
```

دستور npm start برای ویندوزهایی که ورژن آن‌ها قدیمی‌تر از آوریل 2017 باشد، در Bash موفق عمل نمی‌کند.

دانلود

QuickStart seed را دانلود کلیک و آن‌ها در فولدر پروژه‌تان از حالت فشرده در بیاورید. سپس به کمک دستورات ترمینال زیر مراحل باقی مانده را انجام دهید.

```
cd quickstart
npm install
npm start
```

فایل‌های غیر ضروری را پاک کنید (اختیاری)

می‌توانید تمامی فایل‌های غیر ضروری که به نوعی به نگهداری از گنجینه‌ی QuickStart و آزمایش آن ارتباط دارند را پاک کنید (از جمله تمامی artifact های مرتبط به git مانند پوشه‌ی.git و.gitignore).

برای آنکه اتفاقاً فایل نصب git و فایل‌های آزمایشی آن را پاک نکنید این کار را تنها در ابتدای کار انجام دهید.

از داخل پوشه‌ی project یه پنجره‌ی ترمینال را باز کنید و دستورات زیر را در سیستم عامل خود وارد کنید.

OS/X (bash)

```
xargs rm -rf < non-essential-files.osx.txt
rm src/app/*.spec*.ts
rm non-essential-files.osx.txt
```

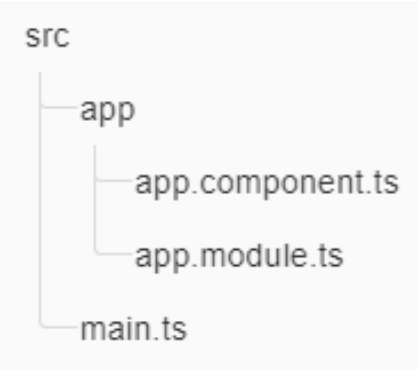
ویندوز

```
for /f%i in (non-essential-files.txt) do del%i /F /S /Q
rd.git /s /q
rd e2e /s /q
```

QuickStart seed شامل چه مواردی است؟

QuickStart seed دارای همان برنامه‌هایی است که QuickStart playground دارد، اما هدف اصلی آن فراهم کردن بنیان مستحکمی برای برنامه نویسی محلی است. به همین دلیل فایل‌های بیشتری در پوشه‌ی پروژه در دستگاه شما وجود دارد. در رابطه با اغلب این فایل‌ها در آینده بیشتر خواهید آموخت.

در پوشه‌ی src/ بر روی سه فایل (.ts) تایپ اسکریپت زیر متمرکز شوید.



src/app/app.component.ts

```
import { Component } from '@angular/core';

@Component({
 selector: 'my-app',
 template: `<h1>Hello {{name}}</h1>`
})
export class AppComponent { name = 'Angular'; }
```

src/app/app.module.ts

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppComponent } from './app.component';

@NgModule({
 imports: [BrowserModule],
 declarations: [AppComponent],
 bootstrap: [AppComponent]
})
export class AppModule{ }
```

src/main.ts

```
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { AppModule } from './app/app.module';

platformBrowserDynamic().bootstrapModule(AppModule);
```

تمامی آموزش‌ها و کتاب‌های آموزشی حداقل این سه فایل اصلی را در خود جای داده‌اند. هر یک از این فایل‌ها هدف مشخصی دارند و به صورت مستقل هم زمان با رشد برنامه کامل می‌شوند.

فایل‌هایی که خارج از /src قرار دارند، مربوط به ساخت، گسترش و آزمایش برنامه‌ی شما هستند. این فایل‌ها شامل فایل‌های پیکربندی و وابستگی‌های خارجی هستند.

فایل‌های داخل /src «متعلق» به برنامه‌ی شما هستند. فایل‌های تایپ اسکریپت، HTML و CSS را داخل دایرکتوری /src قرار دهید و اغلب آن‌ها را به داخل src/app انتقال دهید، مگر آن که خلاف آن بیان شود.

تمامی فایل‌های زیر داخل /src قرار دارند.

فایل	هدف
app/app.component.ts	همان AppComponent ای را تعریف می‌کند که در QuickStart playground وجود دارد. این فایل جزء ریشه‌ای چیزی است که بعدها با پیشرفت برنامه به درختی از اجزای تو در تو تبدیل خواهد شد.
app/app.module.ts	AppModule را تعریف می‌کند، AppModule ماژولی ریشه‌ای است که نحوه‌ی سر هم کردن برنامه را برای Angular توضیح می‌دهد. در حال حاضر این فایل فقط AppComponent را اعلان می‌کند. در آینده‌ی نزدیک، اجزای بیشتری برای اعلان وجود خواهند داشت.
main.ts	برنامه را به کمک کامپایلر JIT کامپایل می‌کند و ماژول اصلی برنامه را bootstrap می‌کند تا برنامه بتواند در مرورگر اجرا شود. طی برنامه نویسی اغلب پروژه‌ها کامپایلر JIT انتخاب مناسبی بوده و در محیط‌های دارای کدنویسی زنده مانند Stackblitz تنها گزینه‌ی موجود است. در ادامه‌ی این آموزش بیشتر در رابطه با گزینه‌های جایگزین برای گسترش و کامپایل کردن برنامه خواهید آموخت.

پیوست: Node.js و npm

Node.js و npm package manager ابزارهایی ضروری جهت برنامه نویسی تحت وب مدرن به کمک Angular و پلتفرم‌های دیگر هستند. Node.js به برنامه نویسی کلاینت و ساختن ابزارها کمک می‌کند. npm package manager که خودش یک برنامه‌ی Node.js است، کتابخانه‌های جاوا اسکریپت را نصب می‌کند.

اگر تا به این لحظه آن‌ها را در سیستم خود نصب نکرده‌اید، همین حالا اقدام به این کار کنید.

حتماً چک کنید که ورژنNode.js 8 و بالاتر از آن و ورژن npm 5 و یا بالاتر باشد. برای آن که از این بابت مطمئن شوید، دستورات node –v و npm –v را در پنجره‌ی کنسول و یا ترمینال وارد کنید. ورژن های قدیمی‌تر خطا می‌دهند.

توصیه می‌کنیم که برای مدیریت نسخه‌های متعددی از Node.js و npm از nvm استفاده کنید. اگر همین الان پروژه‌های در حال اجرایی در سیستمتان دارید که از نسخه‌های دیگر Node.js و npm استفاده می‌کنند، باز هم بهتر است از nvm استفاده کنید.

پیوست: برنامه نویسی محلی چرا مهم است؟

کد نویسی زنده در مرورگر و یا استفاده از مثالی که برای دانلود قرار داده شده است، روشی عالی برای بررسی دقیق Angular می‌باشد.

لینک‌هایی که تقریباً در تمامی صفحات این آموزش وجود دارند، مثال‌های کاملی را در مرورگر باز می‌کنند. می‌توانید با این نمونه کدها بازی کنید، تغییرات خود را با دوستان خود به اشتراک بگذارید و یا کدهای آن را در سیستم خود دانلود و اجرا کنید.

در بخش «شروع کار» تنها فایل AppComponent نشان داده شده است. این فایل معادل app.module.ts و main.ts را به صورت درونی فقط برای playground ایجاد می‌کند. به همین دلیل خواننده می‌تواند بدون حواس پرتی به کشف Angular بپردازد. مثال‌های دیگر مبتنی بر QuickStart seed هستند.

هرچند این فرآیند جالب است، اما در نظر داشته باشید که ...

- نمی‌توانید در Stackblitz برنامه‌ی خود را برای فروش آماده کنید.
- هنگام کد نویسی همیشه آنلاین نیستید.
- ترنسپایل کردن تایپ اسکریپت در مرورگر کند انجام می‌شود.
- پشتیبانی از نوع، بازسازی کد و تکمیل کد تنها در IDE محلی خودتان کار می‌کنند.

به عنوان playground از این مثال زنده و یا لینک دانلودی موجود استفاده کنید تا نمونه‌های موجود در این آموزش را امتحان کنید و خودتان آن‌ها را آزمایش کنید. زمانی که بخواهید یکی از مشکلات موجود در آموزش و یا یکی از مشکلات موجود در خود Angular را اصلاح کنید، این محیط‌ها بهترین مکان برای تکثیر باگ هستند.

برای برنامه نویسی واقعی به شما پیشنهاد می‌کنیم که از برنامه نویسی محلی استفاده کنید.

پیوست: برنامه نویسی محلی به کمک IE

اگر به کمک ng serve به صورت محلی در Angular در حال برنامه نویسی باشید، بین مرورگر و سرور برنامه نویسی محلی اتصال websocket به صورت خودکار بر قرار خواهد شد. به همین دلیل اگر کد شما تغییر کند، مرورگر می‌تواند به صورت خودکار رفرش شود.

یک برنامه در ویندوز به صورت پیش فرض می‌تواند تنها 6 اتصال websocket داشته باشد (MSDN WebSocket Settings). به همین دلیل در برخی از مواقع اگر IE به صورت دستی و یا خودکار توسط ng serve رفرش شود، بعد از آن که تعداد اتصالات websocket از حد مجاز فراتر رود، websocket به درستی بسته نخواهد شد و خطای SecurityError ایجاد می‌شود. این خطا اثری بر برنامه‌ی Angular ندارد و برای برطرف کردن آن تنها کافی است که IE را ری استارت کنید و یا برای آپدیت محدودیت تعداد websocket ها windows registry را تغییر دهید.

پیوست: آزمایش کد با استفاده از fakeAsync() و یا async()

اگر برای اجرای واحدهای آزمایشی (برای جزئیات دقیق به آموزش «آزمایش کردن» مراجعه کنید.) از تابع کمکی fakeAsync() و یا async() استفاده کنید، نیاز است که در فایل راه اندازی آزمایشی خود zone.js/dist/zone-testing را وارد کنید.

اگر پروژه‌ی خود را به کمک `Angular/CLI` ایجاد کنید، این بخش از قبل در `src/test.ts` وارد شده است.

و در ورژن های قدیمی‌تر Angular فایل‌های زیر از قبل به فایل html شما import و یا اضافه شده بوده‌اند:

```
import 'zone.js/dist/long-stack-trace-zone';
import 'zone.js/dist/proxy';
import 'zone.js/dist/sync-test';
import 'zone.js/dist/jasmine-patch';
import 'zone.js/dist/async-test';
import 'zone.js/dist/fake-async-test;'
```

همچنان می‌توانید این فایل‌ها را به صورت جداگانه بارگیری کنید اما ترتیب این فایل‌ها بسیار مهم است. شما باید پروکسی را قبل از sync-test، async-test , fake-async-test و jasmine-patch ایمپورت کنید. همچنین باید sync-test را قبل از jasmine-patch ایمپورت کنید. به همین دلیل توصیه می‌شود به جای آن که این فایل‌ها را به صورت جداگانه بارگیری کنید، فقط zone-testing را import کنید.