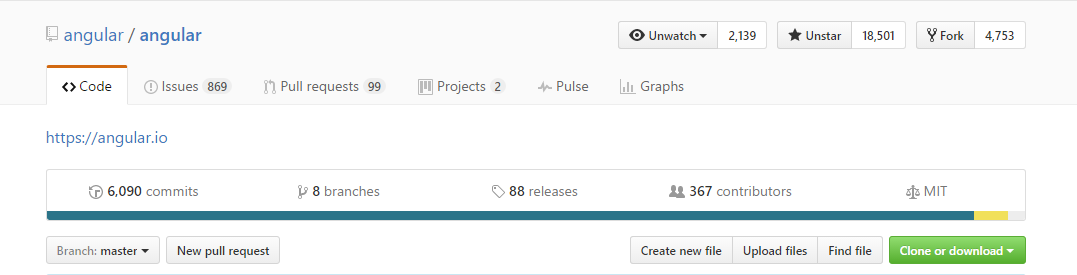
**Thử nghiệm với Angular 2**

# [Phần 0: Bạn Đã Sẵn Sàng Thử Nghiệm](#_Phần_0:_Bạn)

Angular 2 đã phát hành chính thức, tính tới thời điểm viết bài này, nó đã ra đến phiên bản 2.2.x, hầu hết các blog đều có những lời đánh tiếng về nó và cũng có vô vàn các hướng dẫn để bạn có thể tiếp cận với Angular 2. Vậy bạn đã sẵn sàng để bắt đầu với nền tảng hot này chưa? Hãy cùng mình thử nghiệm với nó nhé.

[](http://www.tiepphan.com/assets/uploads/2016/12/angular-angular.png)

Hình 1: Repository của Angular 2 trên Github

## 1. Giới thiệu.

Angular is a development platform for building mobile and desktop web applications.

Angular 2 giờ đây là một nền tảng, nó không chỉ là một framework như trước nữa. Giờ đây bạn có thể tạo ứng dụng đa nền tảng như web application, native application, desktop application với Angular 2. Ngoài ra, còn có một cộng đồng lớn mạnh, bạn rất dễ dàng để có thể tiếp cận.

Bằng việc thay đổi lại hoàn toàn kiến trúc của mình, Angular 2 dễ dàng tiếp cận hơn, hiện đại hơn – component based, hiệu năng cao hơn, nhanh hơn Angular 1. Angular 2 được phát triển từ hơn 5 năm đóng góp ý kiến của cộng đồng người dùng Angular, được viết lại trên TypeScript cho phép các nhà phát triển tận dụng các tính năng nổi bật của ES2015, ES2016+ càng khiến nó trở nên mạnh mẽ hơn trong quá trình phát triển tiếp theo của mình.

Một số tính năng mới của Angular 2 như:

* Form Builder
* Change Detection
* Templating
* Routing
* Annotations
* Observables
* Shadow DOM

## 2. Yêu cầu chung.

Trước khi bắt đầu thử nghiệm, chúng ta cần chuẩn bị một số công cụ cho quá trình phát triển phần mềm, trong trường hợp này là tạo các ứng dụng trên nền tảng của Angular 2.

– [Nodejs:](https://nodejs.org/en/)

Các bạn tải về phần cài đặt cho [Nodejs tại đây](https://nodejs.org/en/download/), mình ưu tiên sử dụng bản 6 trở lên và dùng bản LTS. Các bạn tải về và cài đặt theo hệ điều hành đang sử dụng.

– [TypeScript](https://www.typescriptlang.org/):

Sau khi cài đặt các bạn mở Command Prompt/Terminal lên và cài đặt TypeScript qua npm.

|  |
| --- |
| npm install -g typescript  # or short version  npm i -g typescript |

Quá trình cài đặt sẽ cài đặt TypeScript thành global package, có thể chạy command từ bất kỳ đâu.

– [Angular CLI](https://github.com/angular/angular-cli):

Angular CLI là command line tool giúp bạn tạo một số thành phần hoặc một project Angular 2 từ đầu. Ngoài ra, còn một số thiết lập để bạn bắt đầu với Angular 2 dễ dàng hơn.

Các bạn có thể sử dụng “[Quick start](https://github.com/angular/quickstart)” hoặc “[Angular 2 seed](https://github.com/angular/angular2-seed)” nếu muốn, các phần code cũng sẽ giống nhau. Mình sẽ sử dụng Angular CLI.

Để cài đặt Angular CLI, chúng ta cũng làm tương tự như khi cài đặt TypeScript.

|  |
| --- |
| npm install -g @angular/cli  # or  npm i -g @angular/cli |

– Version Control System: [Git](https://git-scm.com/) (không bắt buộc)

Hiện nay, việc dùng Git khi phát triển phần mềm rất phổ biến, bạn có thể quản lý các phiên bản một cách dễ dàng. Nếu chưa biết bạn hoàn toàn có thể học cách sử dụng khá nhanh.

Các bạn có thể tải về bản mới nhất từ trang chủ rồi cài đặt cho máy của mình.

– Editor:

Các bạn có thể dùng bất cứ editor nào mình thích và quen thuộc để làm việc, mình có một số gợi ý: [Visual Studio Code](https://code.visualstudio.com/), [Sublime Text](https://www.sublimetext.com/), [Atom](https://atom.io/), thậm chí [Vim](http://www.vim.org/).

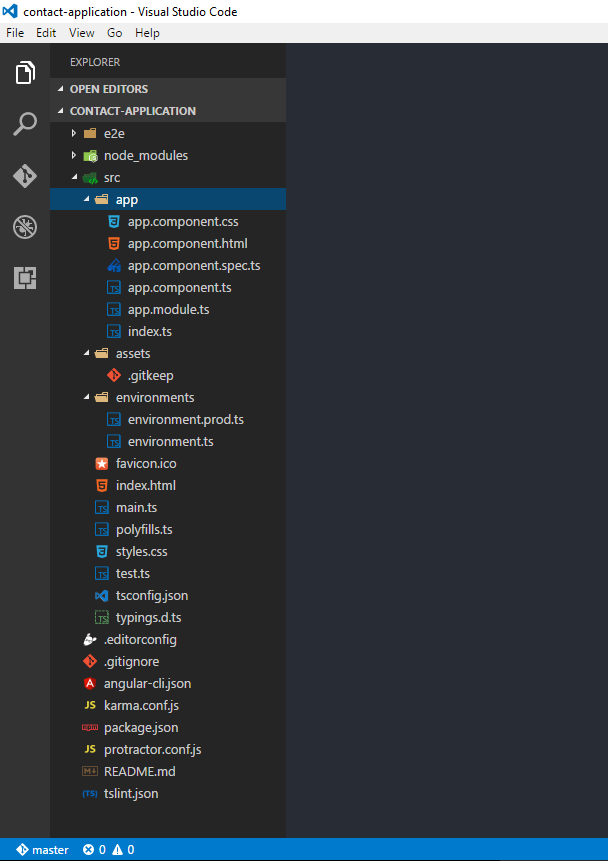
## 3. Tiến hành.

Đầu tiên, chúng ta tạo mới một project với sự trợ giúp của Angular CLI. Các bạn chạy lệnh dưới đây trong Command Prompt/Terminal.

|  |
| --- |
| ng new <app-name>  # example  ng new contact-application  # use preprocessor instead of css: scss, sass, stylus, less  ng new contact-application --style=scss |

Quá trình khởi tạo lần đầu tiên có thể mất vài (chục) phút, để tạo project và download các thư viện cần thiết để ứng dụng có thể thực thi.

Sau khi hoàn thành việc khởi tạo project, chúng ta sẽ có một project với cấu trúc dạng như sau:



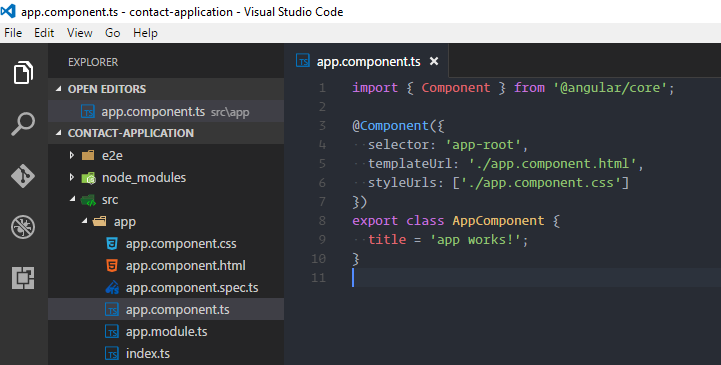
Hình 2: Project structure

**Vậy chúng ta phải bắt đầu từ đâu?**

Một ứng dụng Angular 2 xây dựng trên vô số Component.

A component controls a patch of screen called a view.

Để dễ hiểu thì Component là tất cả những gì mà end-user có thể nhận biết, nó có thể được sử dụng lại nhiều lần trong một ứng dụng. Sau khi tạo xong một project với Angular CLI, chúng ta đã có một Component có tên: AppComponent trong file src/app/app.component.ts.

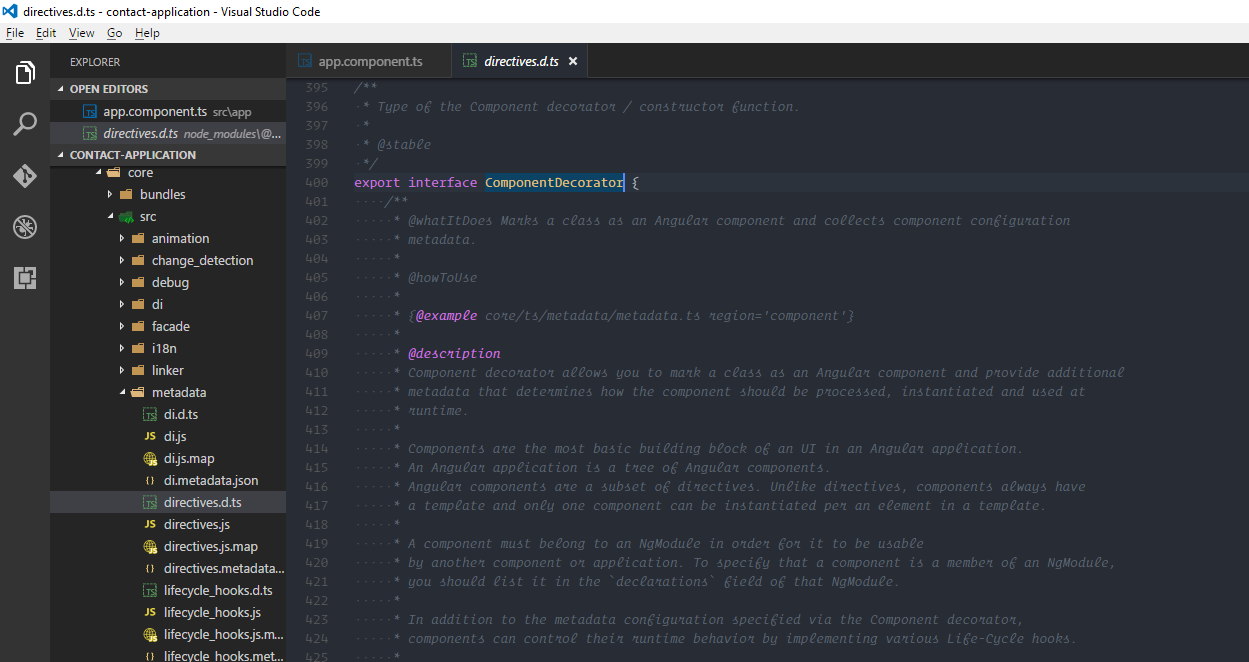


Hình 3: AppComponent

Đoạn code trên có ý nghĩa gì?

* Dòng 1: khi bạn muốn dùng một, một số module, thành phần nào đó từ module khác, bạn import chúng vào, đây là công việc để lấy về các thành phần mà module của bạn phụ thuộc vào. Tính năng này có trong ES2015: modules import. [Bạn có thể tìm hiểu thêm tại đây](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/import).
* Dòng 3 – 7: @Component để làm gì?

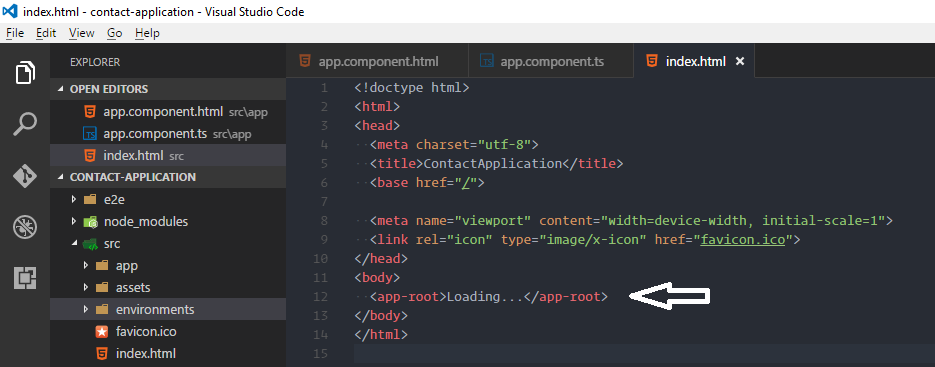
Nó là một khai báo (Decorator) cho class ngay sau đó, để chỉ ra rằng class đó là một Angular component, bên cạnh đó nó còn cung cấp các configuration metadata để Angular 2 biết cách tạo ra Component tương ứng.



Hình 4: *@Component*

@Component yêu cầu tối thiểu phải truyền vào một Javascript object với ít nhất thuộc tính: selector và templateUrlhoặc template.

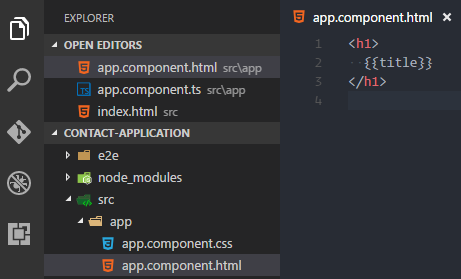
Dòng 4: là selector hay mục đích chỉ ra rằng, khi trong template có một thẻ dạng như thẻ HTML có tên app-root thì Angular sẽ hiển thị Component khai báo bên dưới vào đó. Bạn có thể nhìn thấy thẻ này trong file src/index.html. Trong trường hợp component này, khi render sẽ thay thế phần Text Node Loading&#8230.



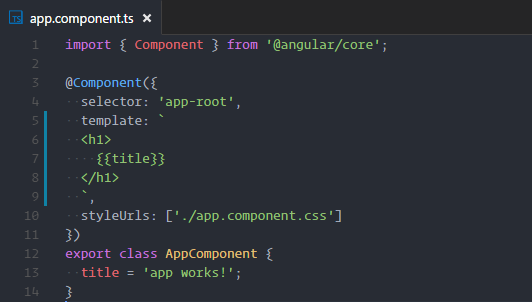
Hình 5: Custom tag *<app-root>* – selector

Dòng 5: templateUrl sử dụng để link đến phần template tương ứng của component này. Trong một số trường hợp, các bạn có thể sử dụng inline template mà không cần tạo file html template riêng lẻ, khi đó bạn thay thế việc sử dụng property templateUrl thành property template.

Lưu ý, nếu bạn dùng property template, bạn có thể sử dụng multi-line string bằng việc bao đóng string trong cặp dấu ``.



Hình 6: templateUrl content



Hình 7: inline template content

Dòng 6: styleUrls để link đến phần style tương ứng cho component này, giá trị của nó là 1 mảng các files style. Tương tự như template, style có thể sử dụng inline style bằng cách thay vì dùng styleUrls bạn sẽ dùng styles property, nó cũng nhận giá trị là một mảng các string khai báo rule, bạn có thể sử dụng multi-line string như template property.

Khi mới tạo project, component bằng Angular CLI, nội dung trong file style thường không có gì, các bạn có thể tùy ý thêm các rules của mình cho ứng dụng.

Dưới đây là list đầy đủ các properties của object trên. Mình sẽ không giải thích hết ý nghĩa của các properties này, các bạn tìm hiểu xem nhé.

|  |
| --- |
| {  animations: "list of animations of this component",  changeDetection: "change detection strategy used by this component",  encapsulation: "style encapsulation strategy used by this component",  entryComponents: "list of components that are dynamically inserted into the view of this component",  exportAs: "name under which the component instance is exported in a template",  host: "map of class property to host element bindings for events, properties and attributes",  inputs: "list of class property names to data-bind as component inputs",  interpolation: "custom interpolation markers used in this component's template",  moduleId: "ES/CommonJS module id of the file in which this component is defined",  outputs: "list of class property names that expose output events that others can subscribe to",  providers: "list of providers available to this component and its children",  queries: "configure queries that can be injected into the component",  selector: "css selector that identifies this component in a template",  styleUrls: "list of urls to stylesheets to be applied to this component's view",  styles: "inline-defined styles to be applied to this component's view",  template: "inline-defined template for the view",  templateUrl: "url to an external file containing a template for the view",  viewProviders: "list of providers available to this component and its view children",  } |

* Dòng 8 – 10:

Định nghĩa class tên là AppComponent, với property title sau đó export cho module khác sử dụng, đây là tính năng [module export](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/export) và [class](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/class) trong ES2015.

**Chúng ta vừa hoàn thành duyệt qua một vòng về Component, vậy làm gì tiếp theo đây?**

Câu trả lời là [Angular Module](https://angular.io/docs/ts/latest/guide/ngmodule.html) hay NgModule.

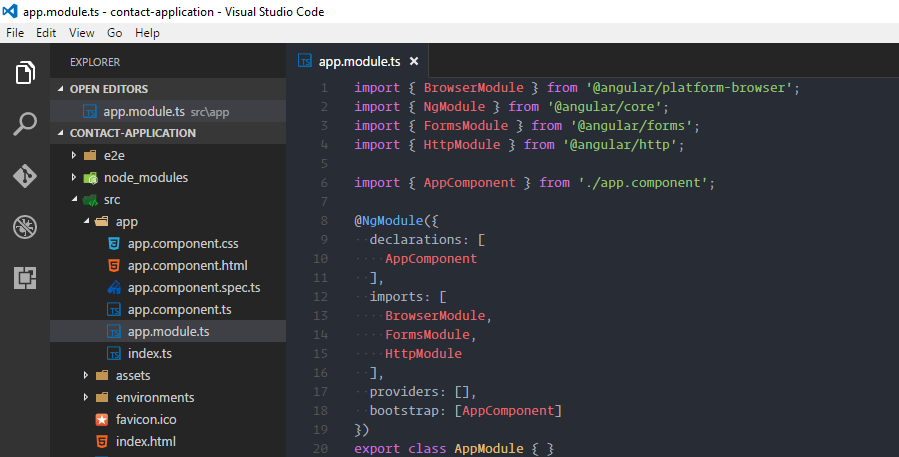
**Angular Modules** help organize an application into cohesive blocks of functionality.

An Angular Module is a class adorned with the @NgModule decorator function. @NgModuletakes a metadata object that tells Angular how to compile and run module code. It identifies the module’s own components, directives and pipes, making some of them public so external components can use them. It may add service providers to the application dependency injectors. And there are many more options covered here.

Trong quá trình phát triển ứng dụng, bạn cũng sẽ thường xuyên gặp các thư viện của Angular là các NgModule chẳng hạn: FormsModule, ReactiveFormsModule, HttpModule, RouterModule, etc.

Mỗi ứng dụng Angular đều có ít nhất một Module đó là root Module, là nơi để khởi chạy ứng dụng. Module có thể chứa các components, pipes, directives, services, …

Thông thường nó được đặt tên là AppModule, nhưng bạn hoàn toàn có thể đặt bất kỳ tên nào nếu muốn. Bạn có thể tìm thấy Module đó trong file src/app/app.module.ts.



Hình 8: *AppModule* – root Module

Trong class này chúng ta cũng nhìn thấy phần quen thuộc như trong định nghĩa của một Component: các phần import, export.

Ở đây, mình muốn các bạn chú ý đến @NgModule, nó là một decorator tương tự như @Component, nhưng object truyền vào thì có các properties khác với object của @Component.

Như hình 8, các properties imports, declarations, providers, bootstrap có ý nghĩa gì?

* **The imports array:**

Như mình đã nói, Angular chia thành nhiều Module tương ứng với các tính năng chẳng hạn. Khi một Module cần sử dụng đến tính năng đó, chúng ta cần nói cho Angular biết và import vào khi tạo Module.

Chẳng hạn, ứng dụng mình đã tạo ở trên, mình làm việc với platform là trình duyệt – browser, nên mình cần import BrowserModule, còn FormsModule để làm việc với form, HttpModule để làm việc với Http như call AJAX, …

**Lưu ý:**

**Only NgModule classes** go in the imports array. Don’t put any other kind of class in imports.

Don’t confuse the import statements at the top of the file with the Angular module’s imports array. They have different jobs.

The JavaScript import statements give you access to symbols exported by other files so you can reference them within this file. They have nothing to do with Angular and Angular knows nothing about them.

The module’s imports array tells Angular about specific Angular modules — classes decorated with @NgModule — that the application needs to function properly.

* **The declarations array:**

Dùng để khai báo các thành phần: components, directives and pipes mà nó thuộc về Module đó. Tất cả các loại class khác không thuộc ba nhóm trên đều không được cho vào array này.

* **The providers array:**

Khi làm việc với Dependency injection, chúng ta cần khai báo các Services cho Injector thực hiện việc nạp các dependencies.

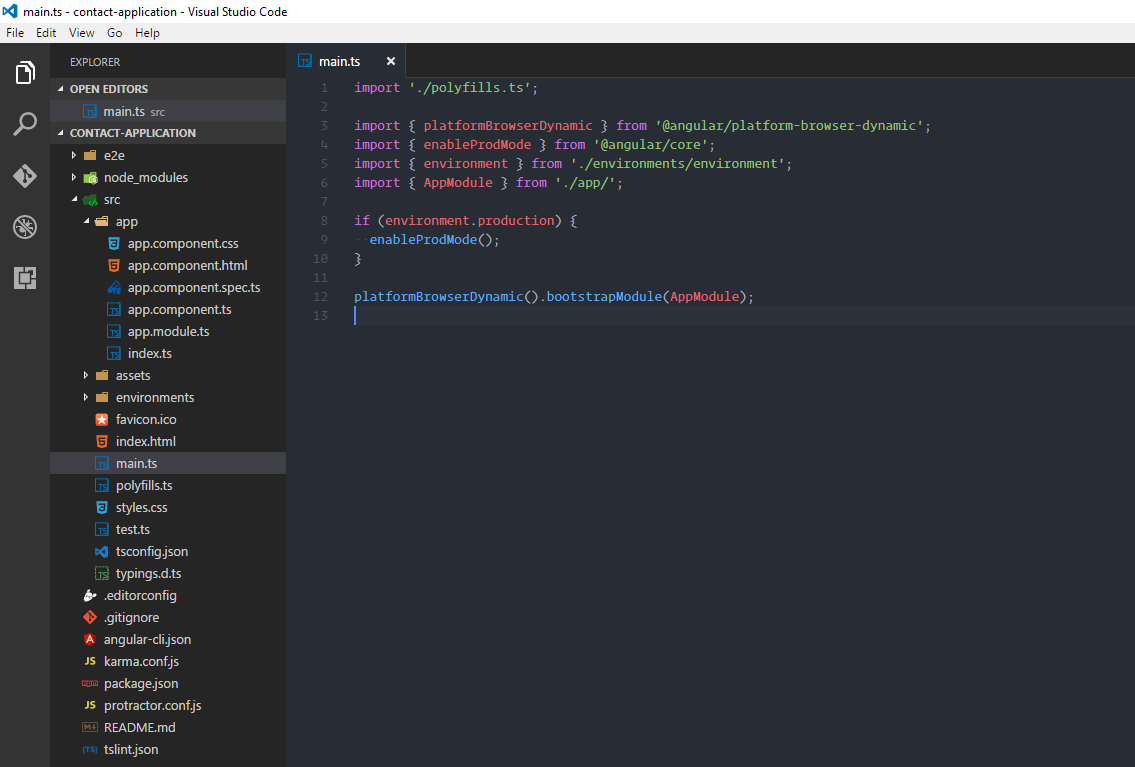
* **The bootstrap array:**

Khi định nghĩa root Module, chúng ta cần nói cho Angular biết chương trình cần bắt đầu ở đâu. Trong hình 8, chúng ta muốn Angular khởi chạy AppComponent.

Ngoài ra còn có **exports array** là list các component mà Module này trả về cho Module khác. Các bạn có thể xem thêm về [kiến trúc của Angular tại đây](https://angular.io/docs/ts/latest/guide/architecture.html).

**Chúng ta đã có Component, có Module, vậy khởi chạy chúng như thế nào?**

Các bạn có thể tìm thấy câu trả lời trong file src/main.ts.

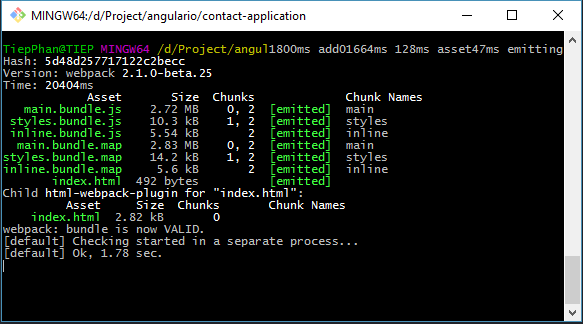


Hình 9: *main.ts*

Ở đây, chúng ta đã sử dụng platformBrowser, sau đó bootstrapModule là root Module – AppModule.

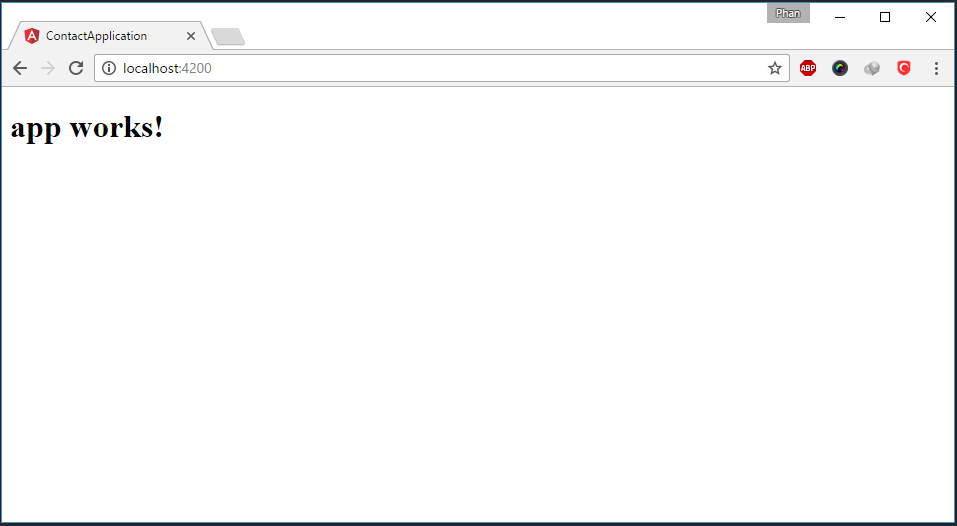
Vậy là tất cả đã rõ, chúng ta có Component, có root Module, có nơi thực hiện bootstrap. Bây giờ, chúng ta sẽ chạy thử chương trình để xem nó hình thù thế nào.

Các bạn vào Command Prompt/Terminal sau đó gõ lệnh ng serve và nhấn enter để Angular CLI thực hiện công việc của nó.



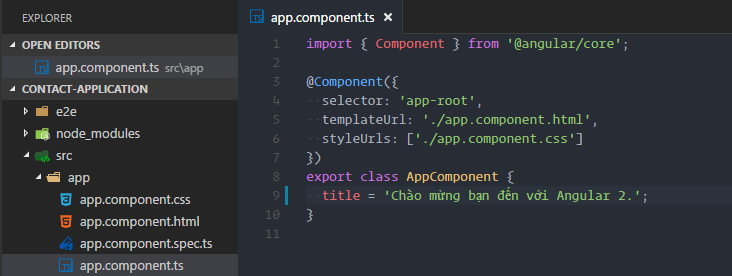
Hình 10: ng serve

Sau đó, các bạn mở trình duyệt vào truy cập vào địa chỉ <http://localhost:4200/>. Kết quả có dạng như sau.

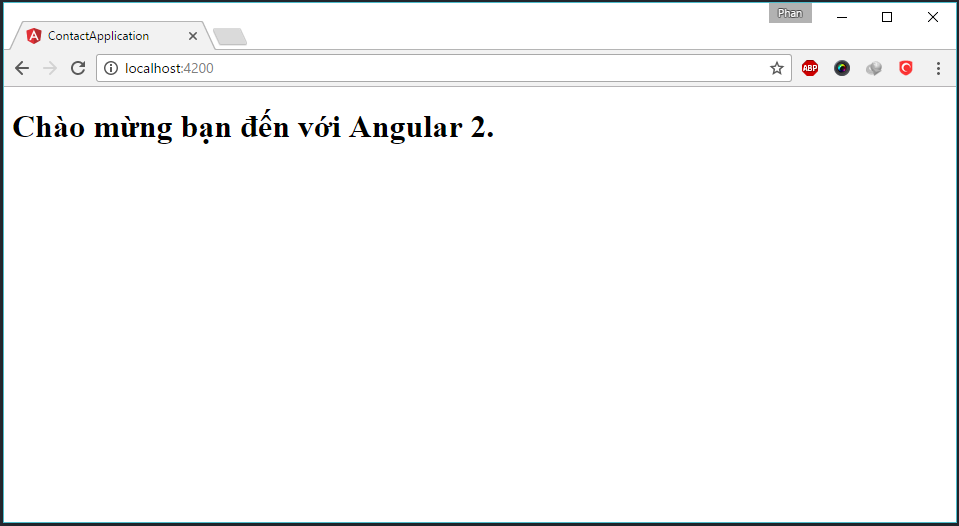


Hình 11: first app

Bây giờ, chúng ta thay đổi chút, mình sẽ thay đổi title trong AppComponent thành một chuỗi khác, và xem kết quả.



Hình 12: edit AppComponent



Hình 13: result

## 4. Lời kết

Angular 2 đang từng bước hoàn thiện và độ nóng của cô nàng này cũng có thừa. Hi vọng với những cái nhìn ban đầu sẽ có nhiều cơ hội để chúng ta từng bước cùng nhau chinh phục.

# [Phần 1: Component và Data Binding](#_Thử_Nghiệm_Với)

Xây dựng component bằng thủ công

Xây dựng component tự động bằng agular-cli

# [Phần 2: Built-in Directives NgIf, NgFor, NgSwitchCase](#_Thử_Nghiệm_Với_1)

Xin chào các bạn, đây là bài học thứ hai trong series **Thử Nghiệm Với Angular 2**, trong bài trước mình đã giới thiệu về Angular 2 Component và Data Binding. Bài học này chúng ta sẽ tìm hiểu về built-in directives NgIf, NgFor, NgSwitchCase trong Angular.

[Thử Nghiệm Với Angular 2 Phần 1: Component và Data Binding](#_Thử_Nghiệm_Với)

Trong bài này chúng ta sẽ đi tìm hiểu các Directives được cung cấp bởi Angular 2 là **NgIf**, **NgFor**, **NgSwitchCase** qua các ví dụ tiếp nối từ bài học trước.

Lưu ý: để sử dụng các directives này, bạn cần　import [BrowserModule](https://angular.io/docs/ts/latest/api/platform-browser/index/BrowserModule-class.html) hoặc

[CommonModule](https://angular.io/docs/ts/latest/api/common/index/CommonModule-class.html)

## NgIf

Sử dụng khi muốn thêm hoặc xóa bỏ một phần tử khi render. Ví dụ: hiển thị thông báo lỗi khi người dùng nhập form chưa đúng.

Cú pháp:

|  |
| --- |
| <h2 \*ngIf="printable">{{ message }}</h2> |

Lưu ý: đừng quên dấu \* phía trước ngIf directive

## NgFor

Sử dụng khi muốn render một list các phần tử. Ví dụ: render list các bài học trong một series chẳng hạn.

Cú pháp:

|  |
| --- |
| <div \*ngFor="let contact of contacts">  <h3>{{ contact.name }}</h3>  <div>  <img \*ngIf="contact.avatar?.url" [src]="contact.avatar?.url" alt="Avatar of {{ contact.name }}">  </div>  </div> |

Lưu ý: đừng quên dấu \* phía trước ngFor directive và sử dụng cấu trúc let &#8230; of &#8230;

## NgSwitchCase

Sử dụng thay thế việc if nhiều lần, tương tự như switch-case trong Javascript.

Cú pháp:

<div [ngSwitch]="conditionExpression">

<template [ngSwitchCase]="case1Exp">...</template>

<template ngSwitchCase="case2LiteralString">...</template>

<template ngSwitchDefault>...</template>

</div>

Hoặc:

<div [ngSwitch]="tabIndex">

<div \*ngSwitchCase="1">

<div>

Tab content 1

</div>

<p>

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Labore, rerum.

</p>

</div>

<div \*ngSwitchCase="2">

<div>

Tab content 2

</div>

<p>

Raw denim you probably haven't heard of them jean shorts Austin. Nesciunt tofu stumptown aliqua, retro synth master cleanse. Mustache cliche tempor, williamsburg carles vegan helvetica. Reprehenderit butcher retro keffiyeh dreamcatcher synth. Cosby sweater eu banh mi, qui irure terry richardson ex squid. Aliquip placeat salvia cillum iphone. Seitan aliquip quis cardigan american apparel, butcher voluptate nisi qui.

</p>

</div>

<div \*ngSwitchCase="3">

<div>

Tab content 3

</div>

<p>

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Ducimus a sequi cupiditate accusantium vitae impedit eum illo voluptatem neque, nisi.

</p>

</div>

</div>

Lưu ý: không có dấu \*ở phía trước ngSwitch directive. Thay vào đó, sử dụng property binding.

Đặt dấu \* ở phía trước ngSwitchCasevà ngSwitchDefault. Trường hợp sử dụng với thẻ template như ở ví dụ đầu tiên của ngSwitch thì không.