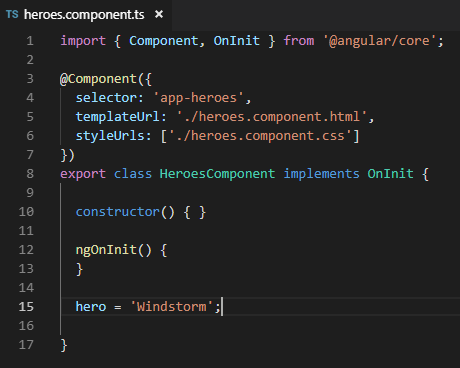
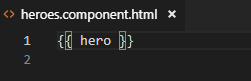
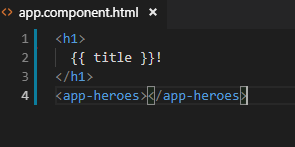
**III –** **ANGULAR**

1. **Những tính năng nổi bật của Angular:**
   * + Tính năng tự động reload brower khi code thay đổi
     + Giúp code có tính chặt chẽ cao hơn và ngắn gọn hơn
     + Phiên bản Angular 4 là phiên bản mới, được nâng cấp từ phiên bản Angular 2
     + Tài liệu tham khảo: <https://angular.io/>
2. **Cài đặt môi trường lập trình**
   * + Cài nodeJS: truy cập đường dẫn sau để download NodeJS phiên bản phù hợp với máy <https://nodejs.org/en/>
     + Cài đặt Code editor: VS Code: <https://code.visualstudio.com/download>
3. **Cài đặt Angular CLI**
   * + Mở Terminal và gõ lệnh : **npm install -g @angular/cli**
     + Kiểm tra xem đã cài đặt thành công chưa: **ng -v**
4. **Tạo không gian làm việc và ứng dụng đầu tiên**
   * + Tạo project mới**: ng new ProjectName**
     + Chạy project lên: **ng serve --open**
     + Mở trình duyệt với địa chỉ: <http://localhost:4200/> để xem kết quả
5. **Các thành phần của Angular**
   * + ./src/app/app.component.ts.
     + ./src/app/app.component.css
     + ./src/app/app.component.html
     + ./src/app/app.module.ts
     + ./src/index.html
6. **Khái niệm component**
   1. Component là gì?
      * Component là các thành phần được hiển thị trên màn hình
      * Điều chỉnh một phần của màn hình (view)
      * Component có thể chứa nhiều thẻ HTML
      * Component có thể chứa Component khác
      * Component có thể nhận tham số
      * Component có thể rất linh hoạt
   2. Lợi ích của việc sử dụng Component
      * Dễ quản lý code
      * Tái sử dụng code
7. **Tạo Component**
   * + Sử dụng Angular CLI, tạo ra một Component mới có tên là **heroes**.

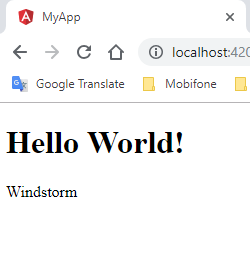
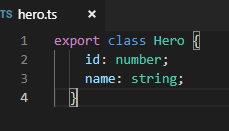
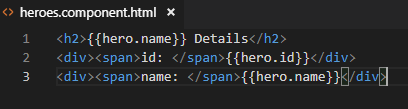
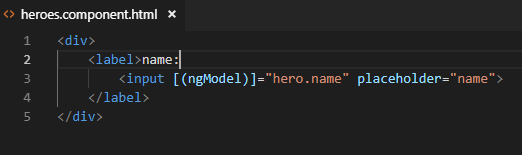
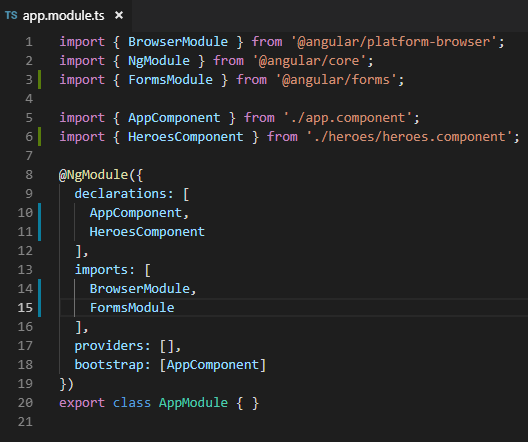
**ng generate component heroes** (Viết gọn **ng g c heroes**)

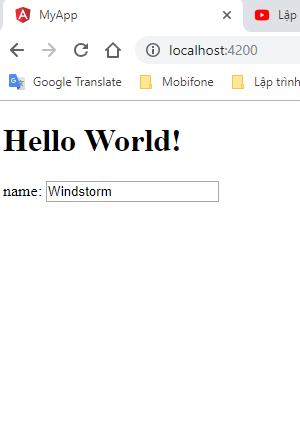
* + - CLI sẽ tạo một thư mục mới, **src / app / heroes /** và tạo bốn tệp của **HeroesComponent**.
    - Component này sẽ tự động được import vào file **app.module.ts**
    - Đây là nội dung của file **heroes.component.ts**



* + - Thêm thuộc tính ***hero = 'Windstorm';***
    - Mở file **heroes.component.html**. Xóa đoạn text mặc định được tạo bởi Angular CLI và thay thế nó bằng một **data binding** với thuộc tính **hero** vừa khởi tạo.
    - 
    - Để hiển thị **HeroesComponent**, phải thêm nó vào **AppComponent**
    - \* Lưu ý: các ***app-heroes*** là bộ chọn thành phần cho **HeroesComponent**. Vì vậy, hãy thêm phần tử **<app-hero>** vào tệp mẫu **AppComponent**, ngay bên dưới  **title**.
    - 

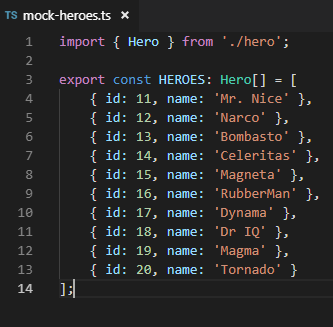
Đây là trình duyệt hiển thị kết quả:

* + - 
    - Tạo một lớp **hero** trong thư mục ***src / app***, với thuộc tính **id** và **name**.
    - 
    - Chỉnh sửa lại file **heroes.component.ts**, như sau:
    - 
    - Trong file **heroes.component.html**, chỉnh sửa lại để hiển thị **id** và **name**, như sau:
    - 
    - Định dạng với ***UppercasePipe***
    - Chỉnh sửa lại ràng buộc **hero.name** như sau:
    - 
    - Chỉnh sửa lại file html, như sau:
    - 
    - Trong đó [(ngModel)] là cú pháp two-way binding của Angular
    - Import **FormModule** vào file **app.module.ts**
    - 
    - Mở trình duyệt lên sẽ được giao diện này:

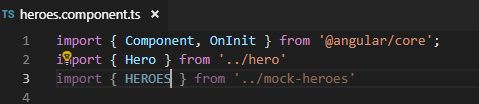


**2. Hiển thị danh sách các mục**

**-** ở trong thư mục **src/app**/, định nghĩa file **mock-heroes.ts** khai báo một hằng **HEROES** chứa danh sách các mục **heroes** như sau:



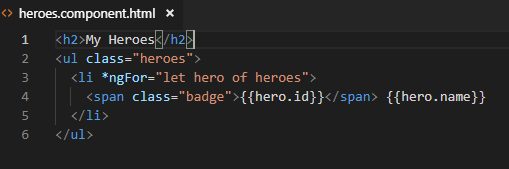
* + - Mở file **heroes.component.ts** và import **HEROES** vào



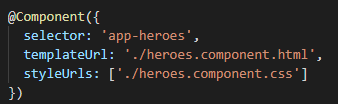
* + - Cũng trong file này, định nghĩa một biến **heroes** để lấy giá trị từ **HEROES**



* + - Mở file heroes.component.html, chỉnh sửa lại như sau:



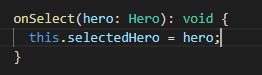
* + - **\*ngFor** là cú pháp lặp lại của Angular. Nó lặp lại phần tử cho từng phần tử trong danh sách.
    - **Định dạng CSS lại cho đẹp**
    - Khi CLI tạo **HeroesComponent**, nó đã tạo ra một bản định kiểu **hero.component.css** trống cho **HeroesComponent** và khai báo vào nó trong **@Component.styleUrls** như thế này.



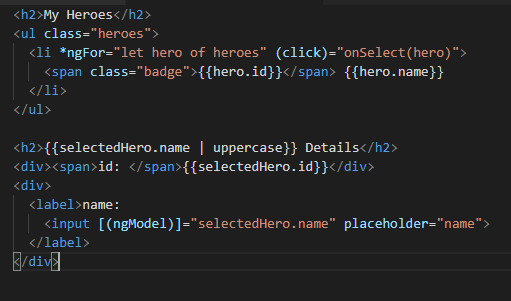
* + - Thêm đoạn code định dạng CSS cho Component vào file **heroes.component.css**
    - Đoạn định dạng CSS này chỉ có tác dụng đối với Component này mà thôi
    - Thêm ràng buộc sự kiện **(click)** vào thẻ <li> như sau:



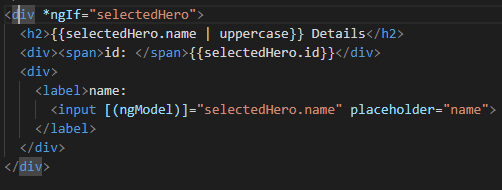
* + - Đây là một ví dụ về cú pháp ràng buộc sự kiện trong Angular, khi người dùng kích vào thẻ <li>, Angular sẽ thực thi hàm **onSelect(hero)**
    - **onSelect ()** là một phương thức của lớp **HeroesComponent** mà bạn sắp viết, Angular gọi nó với đối tượng **hero** được hiển thị trong <li> đã nhấp, là một đối tượng **hero** được xác định trước đó trong biểu thức **\*ngFor**.
    - **THÊM HÀM XỬ LÝ SỰ KIỆN**
    - Tạo thêm thuộc tính **selectedHero** có kiểu **hero**, nhưng không gán giá trị, nghĩa là không có giá trị nào được khởi tạo khi khởi động ứng dụng
    - Thêm phương thức **onSelect()** sau đây để gán **hero** được nhấp vào **selectedHero** của Component.



* + - **CẬP NHẬT CHI TIẾT CHO TỪNG ĐỐI TƯỢNG**
    - Chỉnh sửa lại file **heroes.component.html** để khi kích vào **hero** nào thì hiển thị chi tiết của **hero** đó



* + - Sau khi trình duyệt tải lại, ứng dụng sẽ bị lỗi, nhưng khi kích vào một mục bất kỳ, ứng dụng lại trở lại bình thường
    - Vì khi bắt đầu ứng dụng, không có mục nào được chọn, nên biến selectedHero là underfined theo mặc định, nên giá trị {{selectedHero.name}} không hiểu
    - Cách sửa lỗi: dùng **\*ngIf** để kiểm tra, nếu **selectedHero** có giá trị thì mới hiển thị chi tiết, đoạn code như sau:



* + - **ĐỊNH DẠNG CSS CHO HERO ĐƯỢC CHỌN**
    - Thêm đoạn này vào thẻ li của file html, để áp dụng class **selected** cho **hero** hiện tại đang chọn



* + - Cuối cùng, ta nhận được file html như sau:



**3. Hiển thị chi tiết các mục trong một component khác**

- Để tách các thành phần lớn thành các thành phần phụ nhỏ hơn, mỗi thành phần tập trung vào một nhiệm vụ hoặc quy trình công việc cụ thể.

- Sử dụng Angular CLI để tạo ra một component mới có tên là **hero-detail**.

**ng generate component hero-detail** (Viết gọn **ng g c hero-detail**)

* + - Câu lệnh trên tạo ra một thư mục hero-detail chứa 4 file:
      1. File CSS chứa các định dạng kiểu cho Component
      2. File HTML để viết code HTML
      3. File TypeScript
      4. File test cho Class **HeroDetailComponent**
    - Câu lệnh trên cũng tự động thêm **HeroDetailComponent**  vào trong file **app.module.ts**
    - Cắt đoạn HTML bên dưới của file **HeroesComponent** và dán vào file HTML của **HeroDetailComponent**