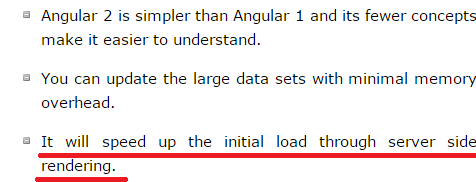
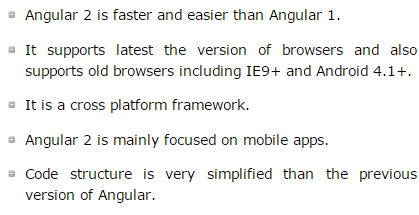
1. What is Angular 2 🡺Angular 2 is an open source JavaScript framework to build web applications in HTML and JavaScript and has been conceived as a mobile first approach
2. The beta version of Angular 2 has been released in the March of 2014.

## Why to use Angular 2?

1. 

## Features

1. 

## Disadvantages

1. Since Angular 2 is a newly introduced framework, there is less online community support.
2. **It takes time to learn if you are new to Angular 2.**

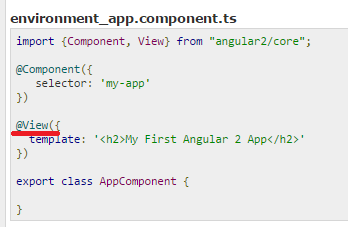
# Angular 2 - Environment

* **Angular uses *TypeScript* which is a primary language for developing of Angular applications.**
* The *TypeScript* is a super set of JavaScript which is migrated to TypeScript and code written in TypeScript makes less prone to run time errors.

1. Npm🡺Node Package Manager

COMPONENT

1. A component is the fundamental concept of Angular.
2. A component is a class that controls a view template - a part of a web page where information to the user is displayed and user feedback is responded.
3. Components are required to build Angular apps.
4. The files which you create need to be saved with **.ts** extension.



1. The *@view* contains a *template* that tells Angular how to render a view.
2. The *export* specifies that, this component will be available outside the file.
3. **To launch the application, we need to import both *Angular's browser bootstrap* function and *root component of the application*.**

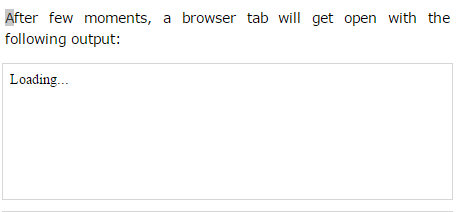


1. To run the application, type the below command in a terminal window:



The above command runs two parallel node processes as listed below:

* TypeScript compiler in the watch mode
* The **lite-server (static server)** loads the *index.html* in a browser and refreshes the browser as application files change.



# Angular 2 - Hello World

1. First let’s see the Html file

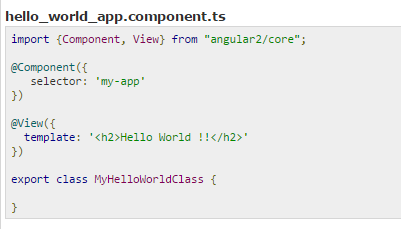


* 1. The SystemJS transpile the TypeScript to JavaScript before running the application by using the *transpiler* option🡺**If you do not transpile to JavaScript before running the application, you could see the compiler warnings and errors which are hidden in the browser.**
  2. *emitDecoratorMetadata*  🡺The TypeScript generates metadata for each and every class of the code when the *emitDecoratorMetadata* option is set. If you don't specify this option, large amount of unused metadata will be generated which affects the file size and impact on the application runtime.
  3. packages 🡺 Angular 2 includes the packages form the *app* folder where files will have the *.ts* extension
  4. Next it will load the main component file from the *app* folder. If there is no main component file found, then it will display the error in the console.
  5. When Angular calls the bootstrap function in main.ts, it reads the Component metadata, finds the 'app' selector, locates an element tag named app, and loads the application between those tags.

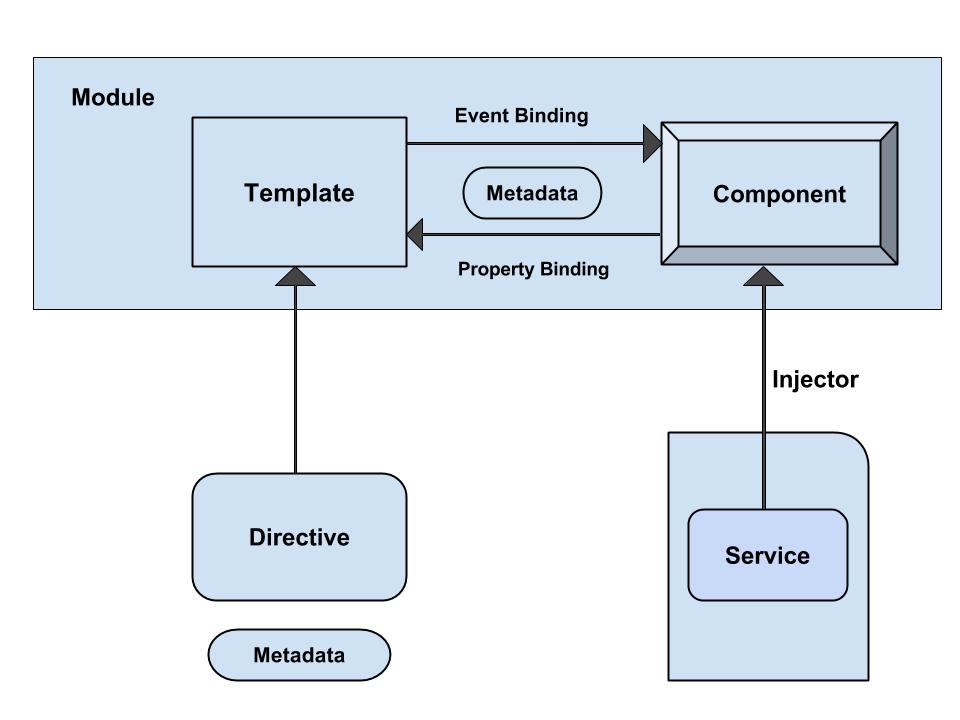
1. To run the code, you need the following *TypeScript(.ts)* files which you need to save under the *app* folder.



1. create a component in TypeScript(.ts) file as shown below:



# Angular 2 - Architecture

1. The following diagram shows architecture of Angular 
2. The architecture of Angular 2 contains following modules:

* **Module**
* **Component**
* **Template**
* **Metadata**
* **Data Binding**
* **Service**
* **Directive**
* **Dependency Injection**

## Module

1. The module component is characterized by a block of code which can be used to perform a single task.
2. you can **export** the value of something from the code such as a class.
3. The Angular apps are called as modules and build your application using many modules.
4. Use the *export* statement to export component class from module as shown below:



1. **The *export* statement specifies that it is a module and its *AppComponent* class is defined as public and can be accessible to other modules of the application.**

## Component

1. A component is a controller class with a template which mainly deals with a view of the application and logic on the page
2. Component knows how to render itself and configure dependency injection
3. You can associate CSS styles using component inline styles, style urls and template inline styles to a component.
4. To register component, we use *@Component* annotation and can be used to break up the application into smaller parts. **There will be only one component per DOM element**

## Template

1. The component's view can be defined by using the *template* which tells Angular how to display the component.
2. For instance, below simple template shows how to display the name:



1. **To display the value, you can put template expression within the interpolation braces.**

## Metadata

1. Metadata is a way of processing the class.
2. Consider we have one component called *MyComponent* which will be a class until we tell Angular that it's a component.  You can use *metadata* to the class to tell Angular that *MyComponent* is a component. The metadata can be attached to TypeScript by using the *decorator*.
3. For instance:



The *@Component* is a decorator which uses configuration object to create the component and its view.

The *selector🡺* creates an instance of the component where it finds <mylist> tag in parent HTML.

The *template* 🡺tells Angular how to display the component. The *directive* decorator 🡺is used to represent the array of components or directives.

## Data Binding

1. Data binding is the synchronization of data between the model and view components. There are four types of data binding

* **Interpolation**: It displays the component value within the div tags.
* **Property Binding**: It passes the property from the parent to property of the child.
* **Event Binding**: It fires when you click on the components method name.
* Example in the Data Binding has OnItemClicked method its userdefined name we can give any name as needed

NOTE🡺 FEW OF THE HTTP STATUS CODES

* 404 When server is not able to find the Resource
* 304 when server is not taking the Latest saved file but instead it takes the file content from its cache
* 200 status was fine, giving me the output
* **Two-way Binding: This form binds property and event by using the *ngModel* directive in a single notation.**

# Data Display

# User Input

## Example 🡺

## this.listItems.push({name:listItem.value}); 🡺Appends new elements to an array and returns the new length of the array

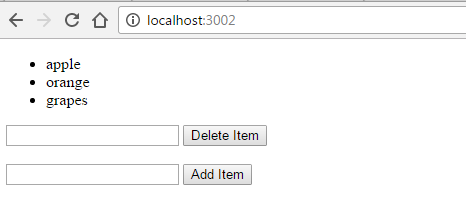
## 

## this.listItems.splice(this.listItems.indexOf(this.selectedItem),1); 🡺 Removes an element from array and if necessary inserts new element

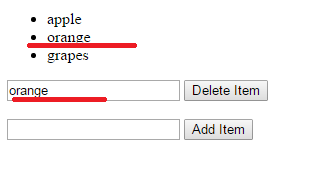
## 

# Angular 2 - Binding User Input

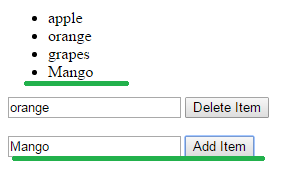
1. Initial on page loading



1. On selecting apple(any element from the list)



1. On adding a new elements 🡺 green newly added element



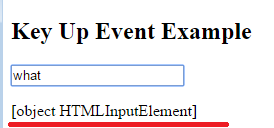
# Angular 2 - User Input from Event Object

1. Output 🡺 See the below Underlined code is due to my changes, Actually

(event.target).value was giving me the error so I had modified the code just to print the Event Object 🡺 val = event.target; is the modification I did

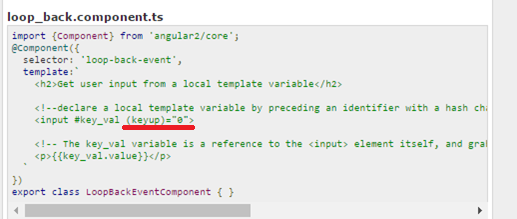


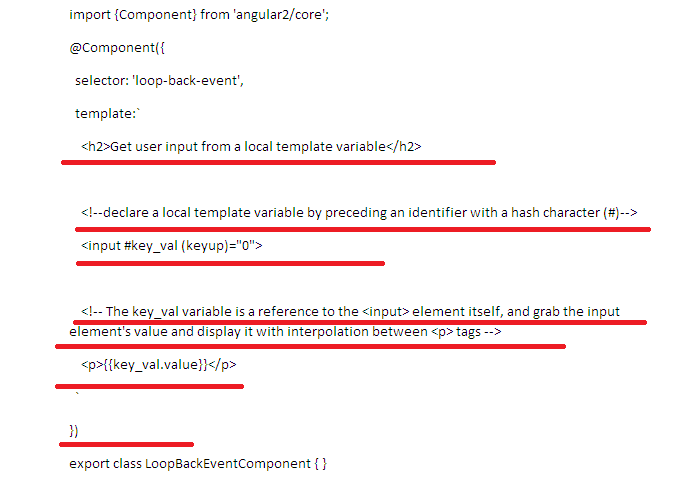
Output:



# Angular 2 - User Input from Local Template Variable

1. 
2. 



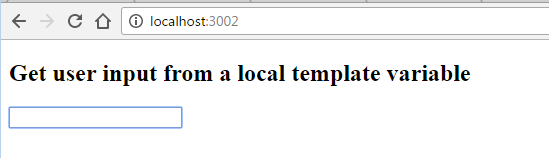


1. Explanation of the above code

* The *@Component* is a decorator which uses configuration object to create the component and its view.
* The *selector* creates an instance of the component where it finds <loop-back-event> tag in parent HTML.
* The *key\_val* is the template reference variable on *<input>* element.
* The *key\_val* variable grabs the input element's value and displayed with interpolation placed between <p> tags.

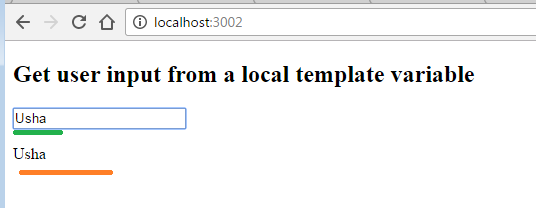
Output 🡺Here what a ever user enters in the textbox same is displayed in the below paragraph, suppose if user deletes all the characters or any characters on the textbox same thing will be reflected in the below paragraph also,

1. Initial on page load

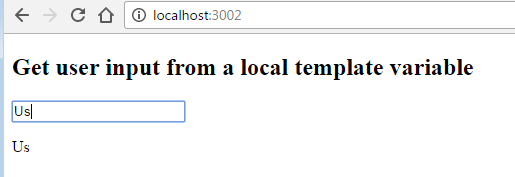


1. On entering the Text in the text box🡺 green🡺 user entered

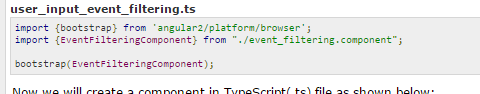
Orange => event triggered due to user event



1. On Deleting the character in the text field



# Angular 2 - Key Event Filtering

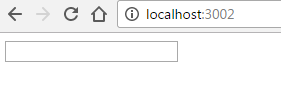
1. 
2. 
3. 
4. Explanation for the above code🡺

* The *@Component* is a decorator which uses configuration object to create the component and its view.
* When the user presses the 'Enter' key on the keyboard, Angular 2 calls the *keyup* event and display the text entered by the user.

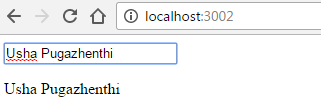
Output 🡺

It’s like once the user press the enter button, what a ever character entered by user will be displayed in the paragraph present below the text box

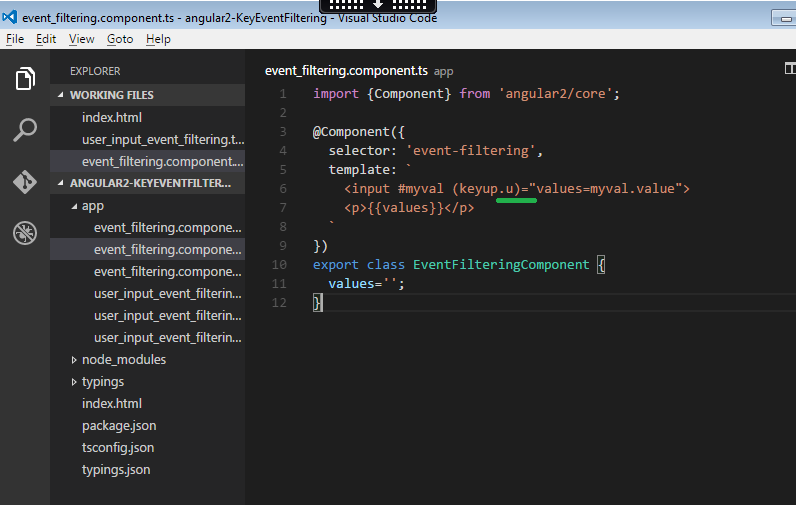
Initial🡺



After entering some character in the above text field press enter

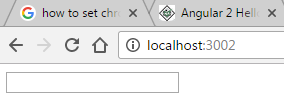


Following changes I did in the code🡺 that is to enter the user entered text once he enters “**u**” in the keyboard

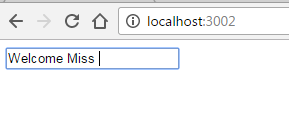


Output🡺

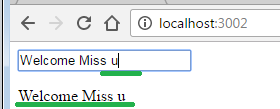
1. initial



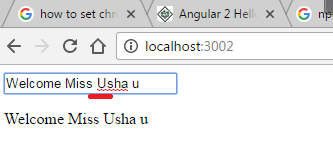
1. on entering the text in text box🡺 see no changes now



1. on entering the “U”🡺 see the changes



note it was case sensitive see the below image🡺 while entering “U” event was not triggered but the event was triggered only after entering ‘u’

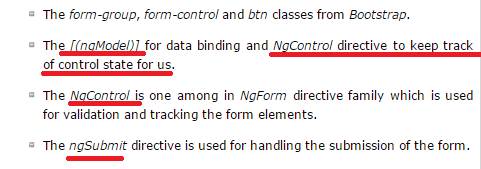


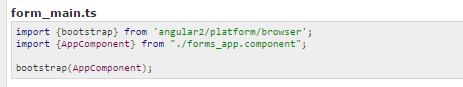
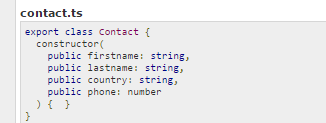
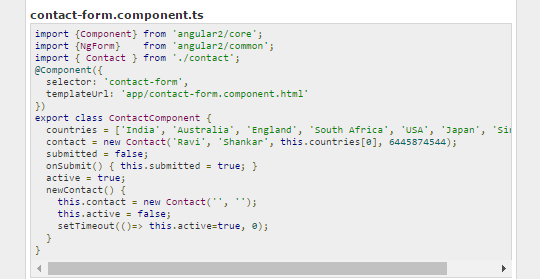
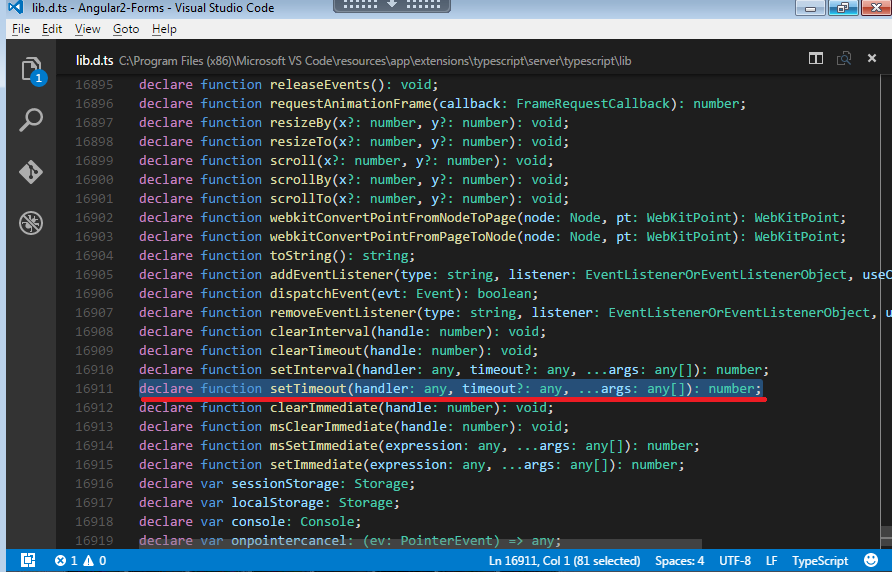
# Angular 2 - On Blur Event

1. not covered 🡺 see the Mosh Video tuto

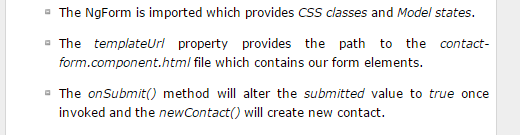
# Angular 2 - Forms

1. In this chapter let us study how to create a *form*.
2. We shall use the following classes and directives in our example.



1. 
2. 
3. 
4. 
5. 
6. 

Explanation🡺

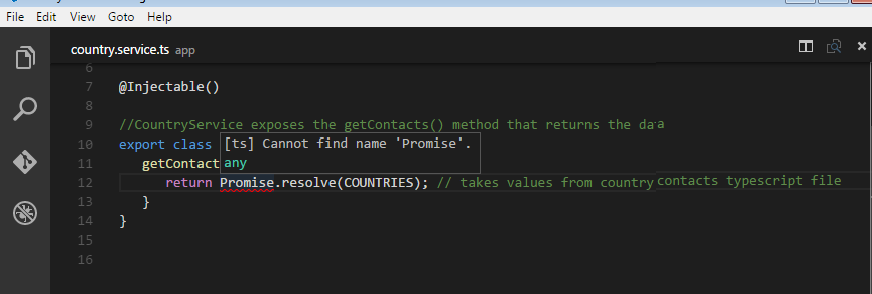


1. 
2. 

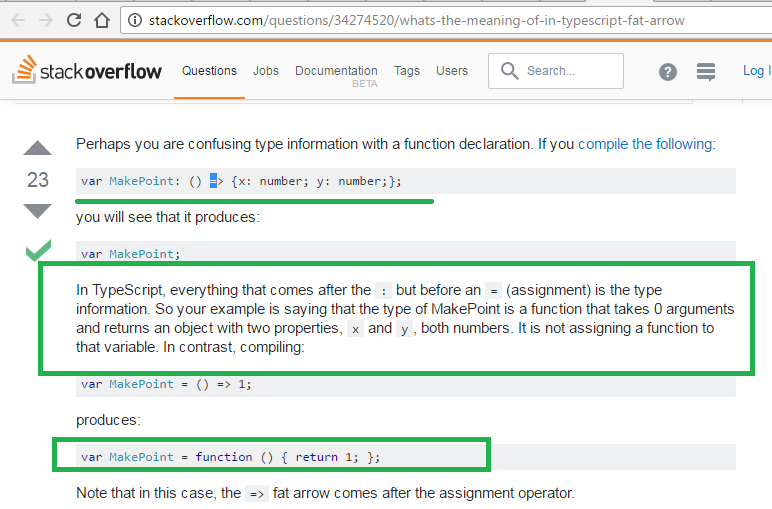
Output🡺 later we will execute once we are clear with the concept

## Service

1. Services are JavaScript functions that are responsible for doing a specific task only
2. Angular services are injected using Dependency Injection mechanism.
3. Service includes the value, function or feature which is required by the application. Generally, service is a class which can perform something specific such as logging service, data service, message service, the configuration of application etc.



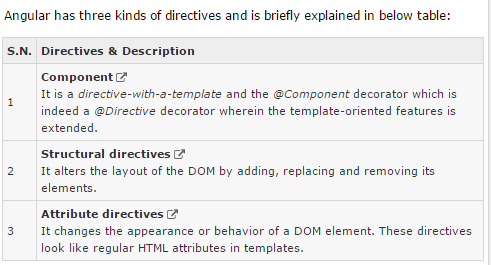
Note 🡺What is the meaning of “=>” in Angular2



## Directive

1. The directive is a class that represents the metadata. There are three types of directives:

* **Component Directive**: It creates custom controller by using view and controller and used as custom HTML element.
* **Decorator Directive**: It decorates the elements using additional behavior.
* **Template Directive**: It converts HTML into a reusable template.

1. Templates of the Angular are *dynamic*, when these templates are rendered by Angular, it changes the DOM according to the *directive* fed instructions
2. 

# Angular 2 - Attribute Directives

## Dependency Injection

1. Dependency Injection is a design pattern that passes an object as dependencies in different components across the application.
2. It creates new a instance of class along with its required dependencies.

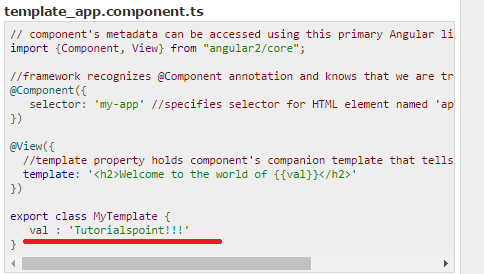
# Angular 2 - Modules

1. The applications in Angular follow modular structure.
2. The Angular apps will contain many *modules*, each dedicated to the single purpose.
3. A module *exports* some *classes*, *function* and *values* from its code.
4. **A module can be a library for another module. For instance, the *angular2/core* library which is a primary Angular library module will be imported by another *component*.**

# Angular 2 - Components

1. The component is a controller class with a template which mainly deals with a view of the application and logic on the page.
2. It is a bit of code can be used throughout an application.
3. **The component knows how to render itself and configure dependency injection.**
4. **The component contains two important things; one is *view* and another one is *some logic*.**

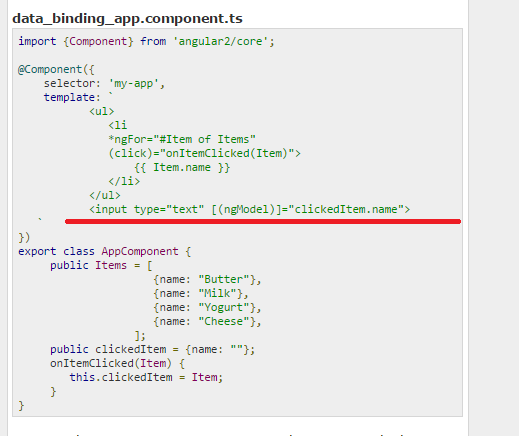
# Angular 2 - Templates

1. The component's view can be defined by using the *template* which tells Angular how to display the component.
2. The template describes how the component is rendered on the page.
3. 

# Angular 2 - Metadata

1. 

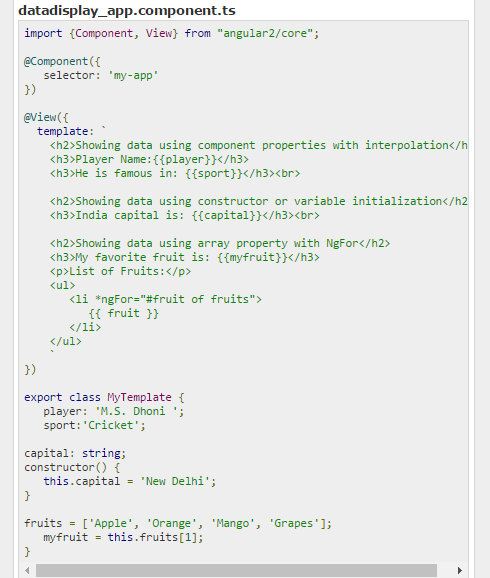
# Angular 2 - Data Binding

1. Data binding is the synchronization of data between the model and view components.
2. **To display the component property, you can put its name in the view template, enclosed in double curly braces.**
3. Two-way data binding merges property and event binding in a single notation using the directive *ngModel*.
4. 

Explanation

1. **Next is *\*ngFor* directive creates the view exports which we bind to in the template. The \* is a shorthand for using Angular 2 template syntax with the template tag.**

# Angular 2 - Data Display

1. You can display the data with the **help of binding** controls in the UI
2. 

# Angular 2 - User Input

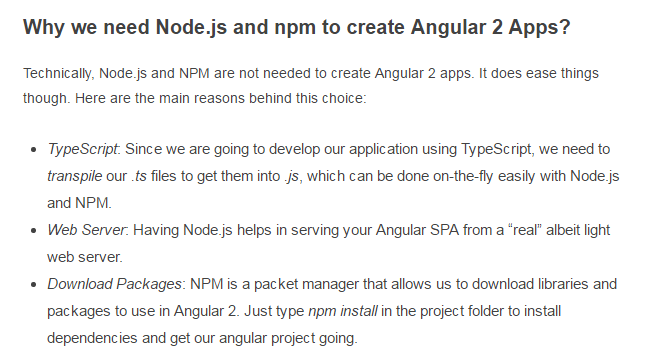
1. When the user clicks a button, enters text or clicks a link, these user interactions will trigger DOM events.
2. The below table describes how to bind to these events using Angular event binding syntax.

Following notes is prepared from the below Website

<http://www.geekhours.com/2016/07/25/creating-your-first-angular-2-app-detailed-step-by-step-tutorial/>

1. Node.js is a prerequisite for developing Angular 2 apps in TypeScript. Install [**Node.js® and npm**](https://nodejs.org/en/download/) if they are not already installed on your machine.

#### ****Why we need Node.js and npm to create Angular 2 Apps?****



#### ****What is transpile? How is it different from Compile?****

Transpiling is a specific kind of compiling. Compiling is the general term for taking source code written in one language and transforming into another. Transpiling is a specific term for taking source code written in one language and transforming into another language that has a **similar level of abstraction.**

Doubts

1. What is Promise
2. What is NgControl
3. What is NgForm
4. What is ngModel
5. Promise