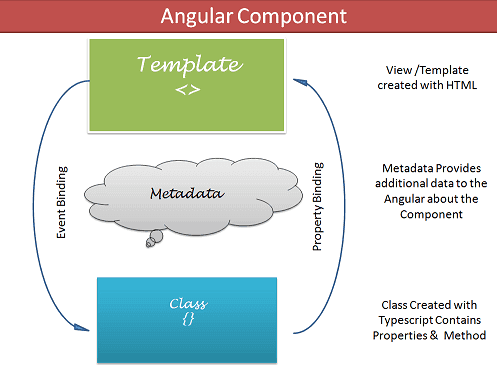
Aug 2nd, 2022



Template: HTML codes along with the

Angular Template Syntax: Angular directives, Angular pipes.

Data to template comes from Component, in term from Service vs data binding.

The Template can use event binding or two way binding to notify the component, when user changes something on the view.

Metadata: provide additional information about component to Angular.

1. Selector
2. Providers
3. Directives
4. Styles
5. Template/templateUrl

The Angular Module organizes the components, [directives](https://www.tektutorialshub.com/angular/angular-directives/), [pipes](https://www.tektutorialshub.com/angular/angular-pipes/), and [services](https://www.tektutorialshub.com/angular/angular-services/)that are related and arrange them into cohesive blocks of functionality.

We use [@NgModule](https://www.tektutorialshub.com/angular/angular-modules/) class decorator to define a [Angular Module](https://www.tektutorialshub.com/angular/angular-modules/) and provide metadata about the Modules.

The **declaration arrays** is where we include the components, [pipes](https://www.tektutorialshub.com/angular/angular-pipes/) and [directives](https://www.tektutorialshub.com/angular/angular-directives/) that are part of this module.

We add all the other [Angular Modules](https://www.tektutorialshub.com/angular/angular-modules/) that this module uses in the **imports array**.

Include all the [Angular Services](https://www.tektutorialshub.com/angular/angular-services/) that are part of this module in the [**providers**](https://www.tektutorialshub.com/angular/angular-providers/)**‘ array**.

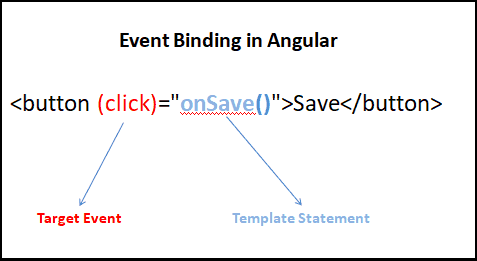
We want appComponent to be loaded when Angular starts, thus we assign it to bootstrap property.

[binding-target]=”binding-source”

## Property Binding Vs Interpolation

## Event Binding

Event binding allows us to bind events such as keystroke, clicks, hover, touche, etc to a method in component. It is one way from view to component.



Component.html:

<h2>Example 1</h2>

<button (click)="clickMe()">Click Me</button>

<p>You have clicked {{clickCount}}</p>

Component.ts:

clickCount=0

  clickMe() {

**this**.clickCount++;

  }

Unlike the [Property Binding](https://www.tektutorialshub.com/angular/property-binding-in-angular/) & [Interpolation](https://www.tektutorialshub.com/angular/interpolation-in-angular/), where we use the template expression is used, in the case of event binding we use template statement.

## $event Payload

<h2>Template Reference Variable</h2>

<input #el (input)="handleInput1(el)">

<p>You have entered {{val}}</p>

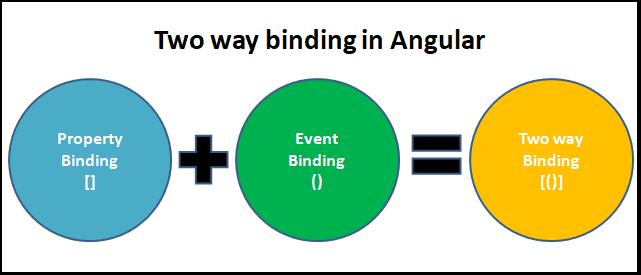
With component.ts:

val="";

handleInput1(element) {

**this**.val=element.value;

}



Good Custom Two-Ways Binding Example:

<https://www.netjstech.com/2020/06/angular-custom-two-way-data-binding.html>



<https://medium.com/codex/how-to-share-data-between-components-in-angular-a-shopping-cart-example-b86ce8254965>

Demo here:

<https://stackblitz.com/edit/angular-ivy-zyhakh?file=README.md>

After calling the getItems() method on the component we can use the async pipe in the component template to subscribe to the returned Observable:

https://blog.angular-university.io/angular-2-ngfor/

How to set up Angular project:

1. Create the project directory
2. Ng new –no-strict bootstrap-example
3. Install Bootstrap CSS framework #npm install bootstrap
4. Add the following to angular.json file: to the styles[]  array as shown in the next video, but the path should be node\_modules/bootstrap/dist/css/bootstrap.min.css
5. If you're getting errors when running npm install, you can often solve them by running npm install --legacy-peer-deps instead of npm install

**Reactive Forms**

Creating a form using FormControl, FormGroup, and FormArray are said to be reactive forms. Therefore, they use the ng module as ReactiveFormsModule.

**Template-Driven Form**

Template Drive Forms are just Model Driven Forms *driven* by directives in the template versus code in the component.

In template-driven, we use directives to create the model. In model-driven, we generate a model on the component and then use directives to map elements in the template to our form model.

Put simply, form controls in Angular give the developer all the control, and nothing is implicit anymore — every choice about inputs and controls must be made intentionally and, of course, explicitly.

Form groups wrap a collection of form controls. Just as the control gives you access to the state of an element, the group gives the same access but to the state of the wrapped controls. Every single form control in the form group is identified by name when initializing.

<https://blog.logrocket.com/reactive-form-controls-form-groups-angular/>

<https://blog.logrocket.com/angular-formbuilder-reactive-form-validation/>

Validator functions can be either synchronous or asynchronous:

* Synchronous validators take a control instance and return either a set of errors or null. When you create a FormControl, you can pass sync functions in as the second argument.
* Asynchronous validators take a control instance and return a Promise or an Observable that later issues either a set of errors or null. You can pass async functions in as the third argument when you instantiate a FormControl.

Running unit tests

Run ng test to execute the unit tests via [Karma](https://karma-runner.github.io/).

## Running end-to-end tests

Run ng e2e to execute the end-to-end tests via [Protractor](http://www.protractortest.org/).

Observable Notes:

### Subscribing to Observable

An Observable function called only when someone subscribes to it.

And to subscribe we should call subscribe() method of the observable instance and additionally we should pass an observer object to read the data from the observable.

how to make use of different rxjs operators with HttpClient Observable.

Great Example: <https://www.angularjswiki.com/httpclient/observable/>

<https://www.angularjswiki.com/httpclient/get/>

<https://reqres.in/api/users?page=1>

<https://fakestoreapi.com/users?limit=2>

export class UserComponent {

userInformation = new UserInformation();

users = new Array<User>();

constructor(userService: UserService) {

userService.getUsers().subscribe(response => {

this.userInformation.page = response.page;

this.userInformation.per\_page = response.per\_page;

this.userInformation.support = response.support;

this.userInformation.total = response.total;

this.userInformation.total\_pages = response.total\_pages;

this.userInformation.data = response.data.map(item => {

var user = new User();

user.avatar = item.avatar;

user.email = item.email;

user.first\_name = item.first\_name;

user.last\_name = item.last\_name;

user.id = item.id;

return user;

});

this.users = this.userInformation.data;

});

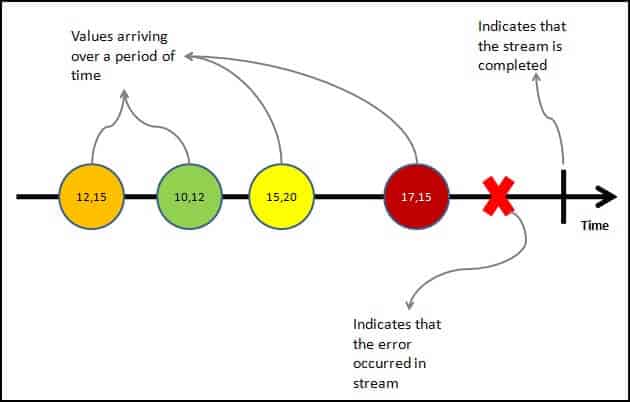
}

}

RxJs Operators:

The operators are very important components of the Rxjs library. They are functions that take an observable as input and transform it into a new observable and return it. We use them to manipulate the observable data stream.

import { map, filter, tap } from 'rxjs/operators'

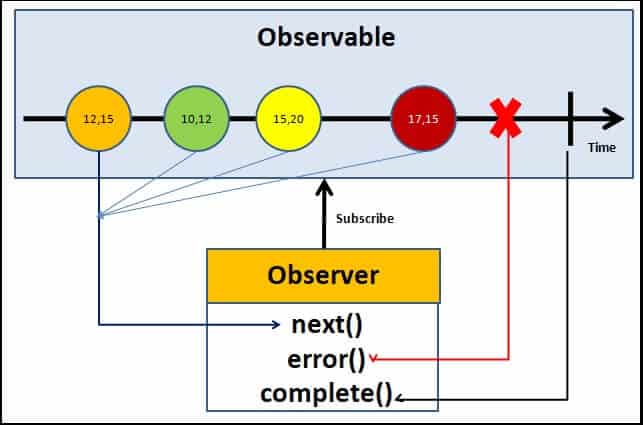


Observable is a function that converts the **ordinary stream of data** into an **observable stream of data**. You can think of Observable as a wrapper around the **ordinary stream of data**.

The observable starts to emit values only when **someone subscribes to it**.

The observers communicate with the Observable using callbacks

The observer must subscribe with the observable to receive the value from the observable. While subscribing it optionally passes the three callbacks. next(), error() & complete()



Angular and springboot-web project

<https://www.baeldung.com/spring-boot-angular-web>

on the eclipse side: C:\UdemyLearning2022\WorkSpace\springboot-web-angular

on the angular side:

C:\AngularProject2022\springboot-web-client

Graphical user interface

Description automatically generated

CLI Overview and Command Reference

<https://angular.io/cli>

Angular Debugging:

Developer Tools 🡺 source => webpack => find .ts file to set up a break point.