**Ref : https://www.youtube.com/watch?v=asX-0wvVM5Y**

There are two types of approach for the form validation :

a)Template driven approach

b)Data driven approach

Template driven :

Angular2 finds the form tag there in our page and then invisibly replace it with the directive. This form tag by default act as a selector for the angular 2 form directive.

Wecan assign the local reference by using

<form (ngSubmit)=”onSubmit()” #f=”ngForm”>

We can access the form created by angular2 by using the local reference f.

Controls property contains all the controls the form have. Angular2 manages the state of each form control. One important property is the

a)valid property.

b)touched : User has touched form or not

c)untouched

d)submitted : true

Validators in angular2 . Angular 2 maintains it’s own form in the back.

a)html5 validator **required :** When angular2 finds the html5 attribute required , it adds it’s own validator there.

b)pattern :

c)

**Form state :**

Right click on the text field and see the class. We have added only form-control. Rest are added by the angular 2. ng-touched ng-dirty ng-valid

Q1)Create a form with fields username, email and password . Add the style to different states of the field.Like for ng-valid , ng-invalid , ng-pristine

Hint ..

styles: [`

.ng-invalid {

border : 1px solid red;

} `]

**One way data binding (Template-driven form validation)**

Q1)Just initialize the username , password and email in the component and perform the one way data binding. Check the output on button click. (Value **(this.user)** not changes. It does not care about the changes done inside field)

Hint .. [ngModel]="user.password"

**Two way data binding (Template-driven form validation)**

Q1) Just initialize the username , password and email in the component and perform the two way data binding. Check the output on button click. (Value **(this.user)** changes. It does not care about the changes done inside field)

Hint .. [(ngModel)]="user.password"

**Form Group in angular2**

Grouping two fields into same group. This creates the nested object.

ngModelGroup = “userData”

**Radio button controls :**

Q1)Create gender radio button and get it inside the form submit.

Q2)Initialize the user object with username , password , email and gender. The form gets automatically populated

**Form state**

**#email = ‘ngModel’** reference

Q1)Apply the validation for username , password , email

Hint .. <input class="form-control" name="username" [ngModel]="user.username" required #username="ngModel">

<p \*ngIf="!username.valid">Username is required</p>

Q2)Disable button when form is not valid

<button [disabled]=”!f.valid”>Submit</button>

**Hints**

<form novalidate #f="ngForm">

<input class="form-control" name="username" [ngModel]="user.username" #username="ngModel" required maxlength="10" minlength="6">

<div \*ngIf="username.errors && username.touched">

<span [hidden]="!username.errors.required">Username is required</span>

<span [hidden]="!username.errors.minlength">6 characters required</span>

<span [hidden]="!username.errors.maxlangth">Not exceed 10 charactrs</span>

</div>