**Lesson16 Angular reactive form custom validator with parameter**

**Notes:-**

**1-The Problem of this custom validator is that the domain name is hard coded, so if we want to use the custom validator with dynamic domain like the Microsoft.com google.com**

function emailDomain(control: AbstractControl): { [key: string]: any } | null {

const email: string = control.value;

const domain = email.substring(email.lastIndexOf('@') + 1);

if (email === '' || domain.toLowerCase() === 'pragimtech.com') {return null;}

else { return { 'emailDomain': true };}}

**2-So the Solution is using the custom validator with parameter**

email: ['', [emailDomain('pragimtech.com')]]

**3-The following built-in validators have parameters.**

**A-min B-max C-minlength D-maxlength**

**Notice the min() built-in validator function. It takes in a number as a parameter and returns ValidatorFn.**

**What is the ValidatorFn?**

**Answer:-**

**1-ValidatorFn stands from validator function. So this min() function is taking in a number as a parameter and returns a validator function. If you understand the concept of closure in JavaScript, then this is very easy to understand.**

**2-In simple terms, you can thinks of a closure as, a function inside another function i.e an inner function and an outer function. The inner function has access to the outer function’s variables and parameters in addition to it's own variables and parameters.**  
  
**3-Now that task at hand for us, is to convert our emailDomain() function to take in the domain name as a parameter and return a validator function. To be able to do this we are going to take the advantage of**[**closures in JavaScript**](https://www.youtube.com/watch?v=HjJQ-lvTgWg&list=PL6n9fhu94yhUA99nOsJkKXBqokT3MBK0b&t=0s&index=28)**.**

**4-ValidatorFn is an interface and the signature of the function it returns is as shown below. It takes the AbstractControl that we want to validate as an input parameter and returns null or ValidationErrors object. Null if the validation succeeds and a ValidationErrors object is the validation has failed.**

(c: AbstractControl): ValidationErrors | null;

//we apply the custom validation with using parameter with closures by using the ValidatorFn

function emailDomain(domainName:string){

//we see that we apply anonymous function inside the outer function

return (control: AbstractControl): { [key:string]: any } | null => {

const email :string = control.value;

const domain = email.substring(email.lastIndexOf("@")+1);

if(email === '' || domain.toLowerCase() === domainName){return null;}

else{return {'emailDomain':true};}}