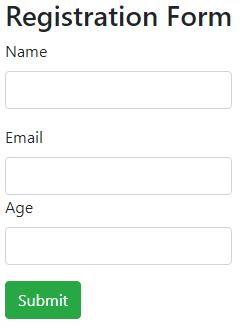
# Lab: Create custom form validation with Vue

## Html Form

We are provided with html registration form, which consist of 3 fields: Name, Age and Email. We must use Vue to validate the form. You are free to implement whatever logic you want.



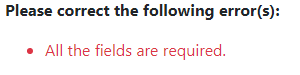
That's how our form should look like.  
The html is pretty simple you have 3 input fields and the labels for them. At the end we have submit button to submit the form.

### Validation upon submitting

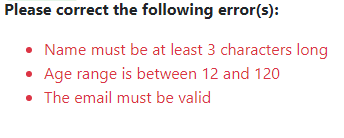
Let's start with basic validation when the form is submitted.  
First it's important to set the rules that we are going to follow.

* All the fields are required.
* Name: Name should be at least 3 characters long.
* Email: Should be validated through regex.
* Age: Age must be in range of 12 to 120.

After the form is submitted we would want to implement some kind of feedback so we will render the errors at the end of the form.  
They should look something like this.



Or



### Implementation

First of all let's start with implementing the requirements to make our html form to work with Vue.  
The **<form>** tag has an ID that we’ll be using for the Vue component.  
There’s a submit handler that you’ll see in a bit

This lab is only about validation so we won't implement action URL.

It should look something like this



After that you should include the **v-model** directive to bind the input with the data.

* Name:



* Email:



* Age: Include **.number** to parse age to number



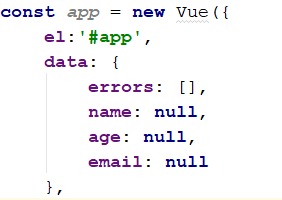
## The Vue instance

So far so good so now let's implement our Vue instance

We need the **el:"{element id}"** to bind the data of the form   
after that declare **data: {}** with the given properties

* **errors:** array that will include the errors required for the feedback
* **name, age, email:** all these properties will be null by default and they will hold the information of the input fields

It should look something like this:



## Validation Methods

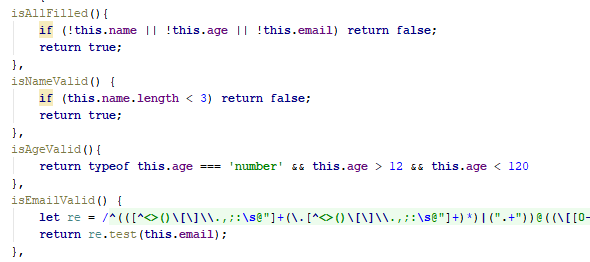
So now it's time to implement the validation methods.  
They will be implemented in "**methods**" property of the Vue instance and each of the methods will follow the following rules

* **isAllFilled() –** this method will check if all the fields have some data e.g. they are not null, returns **false** if there is any **null**
* **isNameValid() –** this method will check if the name is over **3 characters long**, returns **false** if name is under 3 characters
* **isAgeValid() –** this method will check if the age is in the range of **12** to **120**, returns **false** if the data don’t match the rule
* **isEmailValid() –** this method will check if the email is valid using **regex**, returns **false** if the email don’t match the **regex**

Since you already have the methods described try to implement them by yourself.  
We are going to use the following regex pattern, anyway feel free to implement any kind of additional logic!

**let** re = /^(([^<>()\[\]\\.,;:\s@"]+(\.[^<>()\[\]\\.,;:\s@"]+)\*)|(".+"))@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\])|(([a-zA-Z\-0-9]+\.)+[a-zA-Z]{2,}))$/;

If you face any kind of problems you can always check the fully implemented methods here:

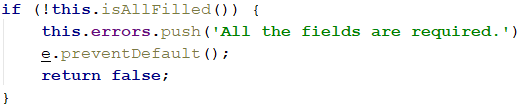
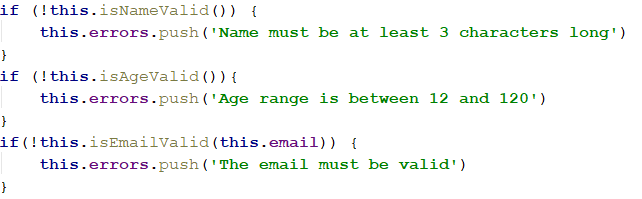
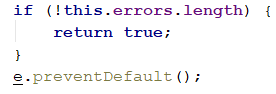


These are all the methods that will be used by our submit handler to validate the form.

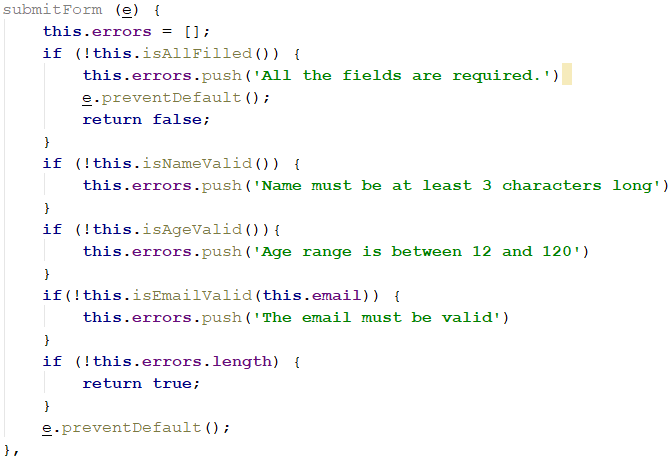
## Submit handler

You've probably already noticed that our submit handler is called **submitForm.**

Let's start implementing the validations step by step

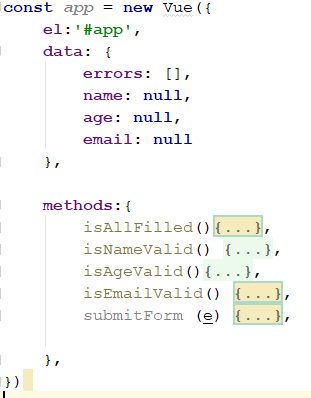
* First of all we must check if the fields have any kind of data, if they don’t we must warn the user that they are required  
  It should look something like this   
     
  Let us explain the validation. If **isAllFilled** returns false, means that one of the input fields are empty.  
  We will add an error message to the errors array and after that we must call **preventDefault** method to prevent the page of reloading.  
  The way our validation works is that we don’t want to show to the user any kind of error messages if all the fields are not filled,   
  so we will return false and this will stop the process.
* Head to the next validations.  
    
  Absolutely nothing special here, just check and if the result is false we are inserting an error to our collection.
* After all of the validations are passed we must check if our errors collection has any elements,  
  If it doesn’t, just return true. If it does invoke the prevent method.  
  

The whole submit handler should look like this:



**Don't forget to set an empty array at the start of the method.**

And the whole Vue instance should look like this:

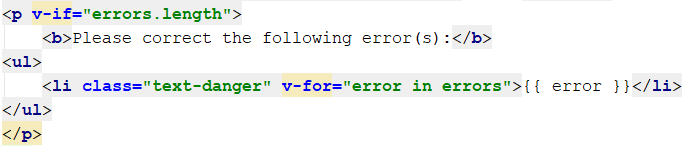


## Implement error rendering

Alright you've already noticed that we have errors collection but we have no way to show it to the users.  
Now it's time to do this.

The errors will be located right after the submit button it will be rendered using "**v-if**" checking if the errors collection has elements.  
After this we will render each of the errors in unordered list using "**v-for**"

Try it out by yourself, if you have issues look at the example bellow.



If everything is implemented correctly our form should render the errors and should not be send.  
Test it out before proceeding.

## Disable button

It would be a cool feature to disable our button based on some kind of validation so let us implement this feature.  
We can use **:disabled="{methodName}"**  in the button html tag to render **disabled** property based on **true** or **false**.

The given example will use **isAllFilled()** method so when any one of the fields is not filled the button will be disabled.  
Keep in mind that we want disabled to be rendered when one of the fields are empty, so our method will return false,   
disabled will be rendered when the result is true, this means we need "!" logical operator to revert the result.

The button should look something like this.

