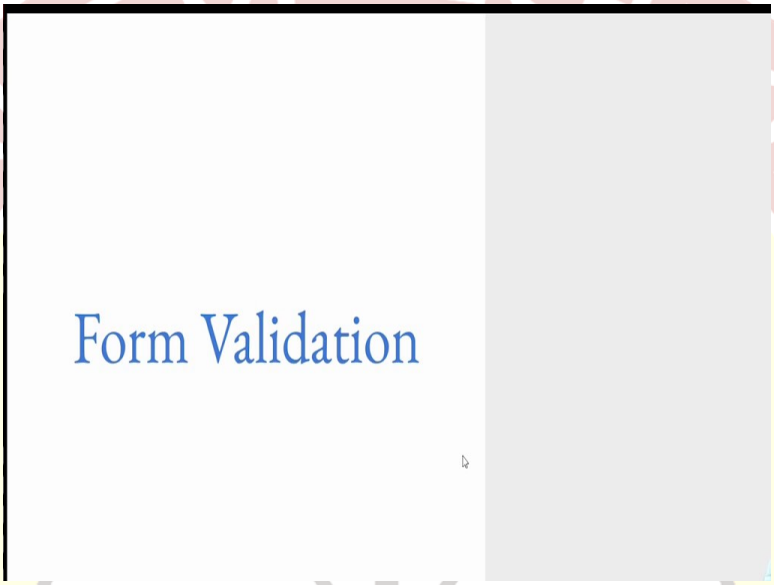
IIT Madras

ONLINE DEGREE

Modern Application Development-2
Professor. Nitin Chandrachoodan
Department of Electrical Engineering
Indian Institute of Technology, Madras
Vue - Form Validation

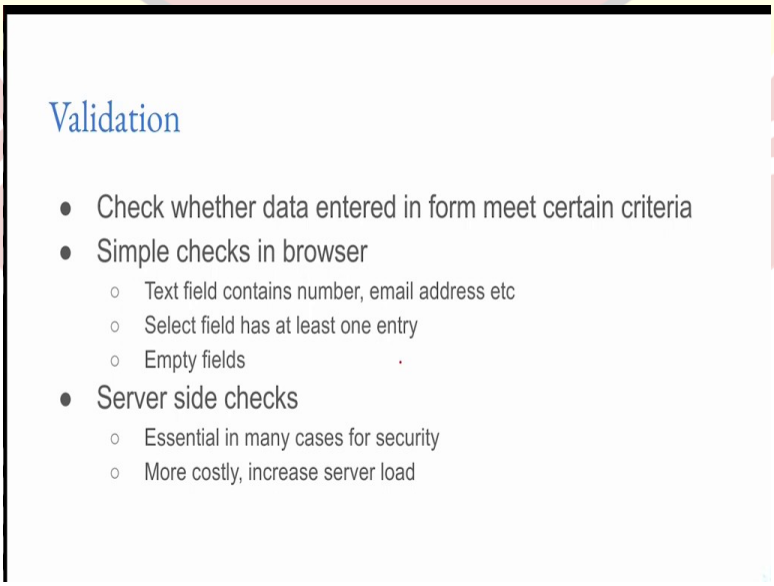
Hello everyone, welcome to Modern Application Development Part 2.

(Refer Slide Time: 0:14)



So, now let us look at the concept of form validation as it applies in the context of Vue.

(Refer Slide Time: 0:22)



So, first things first, what is form validation? We have already looked at this in different contexts even including in the case of Python flask applications. And ultimately, what is doing is checking whether the data that you have entered meets certain criteria. It could be, is this a valid email? Is it a number? If there is a selection field, there is a drop down over there, and have you at least selected one entry?

Are there any empty fields that were mandatory? Was there something where you are expected to type in something, but you had left it blank. All of those are typical examples of form validation. But you might even have some more complicated validation, you might say that, it is not just that it should be a valid email, it should be a valid email with a certain kind of a domain, or it should be a number within a certain range.

So, it is not just the sort of simple checks that we are interested in but, you might also, have something slightly more complex that you are interested in. Now, one thing to keep in mind over here of course is no matter what kind of validation you do in the browser, whenever you are sending data to a server, you have to once again revalidate on the server, you cannot assume that validation was completed on the client.

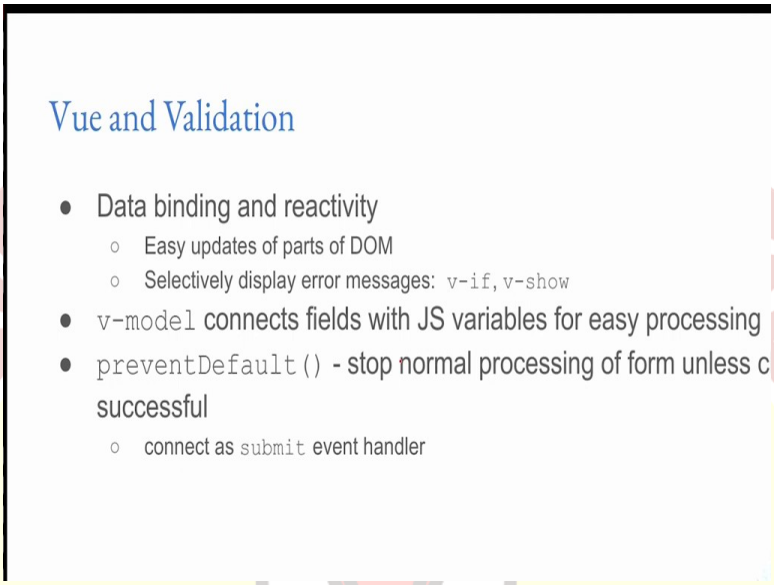
And that whatever you receive on the server from a client is valid. And the reason is simple. There is no way that a server can guarantee that it is only getting requests from a specific client. It could be some other script or even, somebody's browsing from a distant location, because after all, these are web enabled, meaning that in principle, at least they are remote accessible.

Even if you sort of put something which says that only local host is allowed to access it, maybe a script running on localhost could still try and access the server and feed it data which is invalid with the purpose of trying to break the server in some way either corrupt the database, or try and extract information from the server that should not be revealed. So, whatever validation you do on the client side has to be done on the server again or at least some parts of it.

The basic checks that ensure that you do not have a security problem have to be done on the server. Now, the reason why we still do client side checks is because if you do not do that, in a lot of cases, what you will end up with is too much server load. At least if you can have a scenario where most of the regular usage is filtered out on the client side, and validation is performed there, then only malicious attempts are likely to make it through to the server.

And those are typically going to be smaller in number. It is not that it becomes trivial at that point, but at least you can hope to reduce the load on the server significantly.

(Refer Slide Time: 3:19)



So, now what role does Vue play in the validation, there are a number of nice features of Vue that help in terms of not just validation, but also, the user experience during validation. In particular, this whole idea of data binding and reactivity means that you can very quickly update parts of the screen or parts of your document or parts of the Vue that the user sees in response to what has been entered in a form.

You can have things that selectively display error messages, there are this v-if and v-show directives, which can be used very effectively for that. And also, it can be reactive, meaning that the VF VA V show VL's and so, on can react to certain changes in variables or to computed values. Which means that you can very easily write code, which will cause part of the document to get updated and show you whether or not you are on the right track.

So, keep in mind that all of this could be done with pure JavaScript as well. After all, all that Vue is providing is a cleaner syntax and a nicer way of writing code. Which makes it easier to use certain aspects of JavaScript. So, it is not that Vue is providing something that fundamentally cannot be provided by something which is already there in the browser. It is a framework that builds on top of what already exists which by definition means that whatever can be done with

Vue could also have been done with pure JavaScript. But it is probably going to be a lot harder to write the code for that.

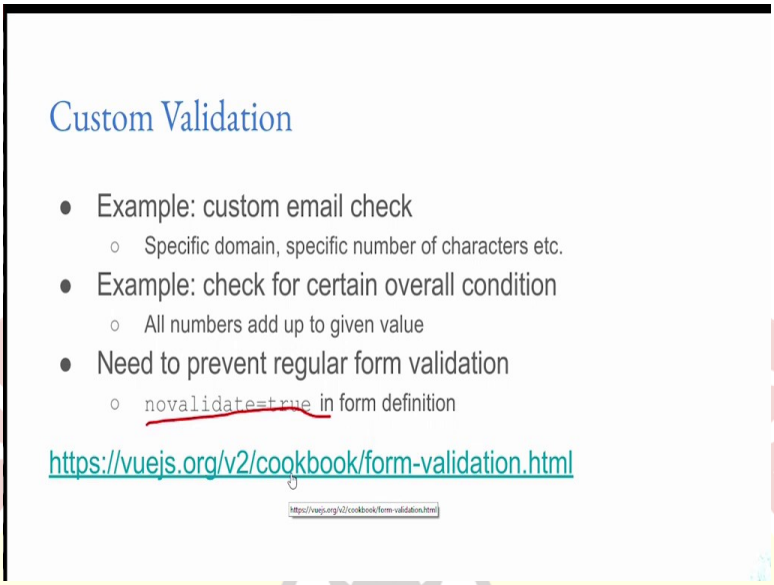
Now, in addition to of course, the directives for selectively displaying messages and so on, the V model can be used in order to connect a field, an input field or a select box or something like that directly with a data model some variable inside your Vue application which means that you could have computed features watchers and so on which can very easily sort of go and update things on the screen.

There is also, a JavaScript function called prevent default, which is very useful in many cases like this, because basically what happens is that the default operation, as defined by HTTP for a submit button is that it actually submits a form. So, if a form has a given target and the action for the form says post to a particular URL.

By definition, what is supposed to happen under the HTML HTTP standards is that when I click on that submit button, the post action should be triggered. But you can use the prevention default, the function call in JavaScript to prevent that from happening. Now, why would you want to do this? Let us say that you are actually trying to override some of the validation, which is there already in the browser.

You click on the submit button, the JavaScript takes over, it performs all the checks, if it finds that something is wrong, it blocks the operation and does not allow it to go through to the browser. So, you can essentially connect this entire check that we have as something which is an event handler for the submit event. So, what is a submit event? It is what happens when I click on the submit button. So, a form has a submit event, which is associated with the submit button of that form.

(Refer Slide Time: 6:56)

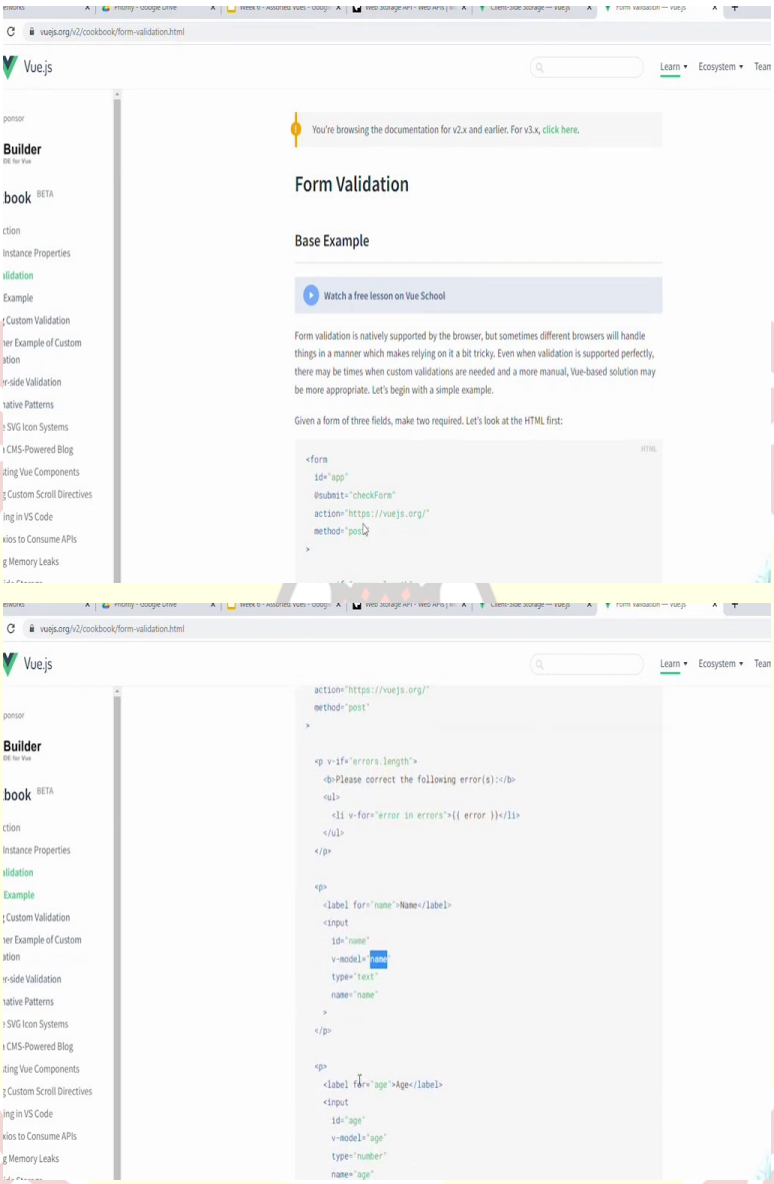


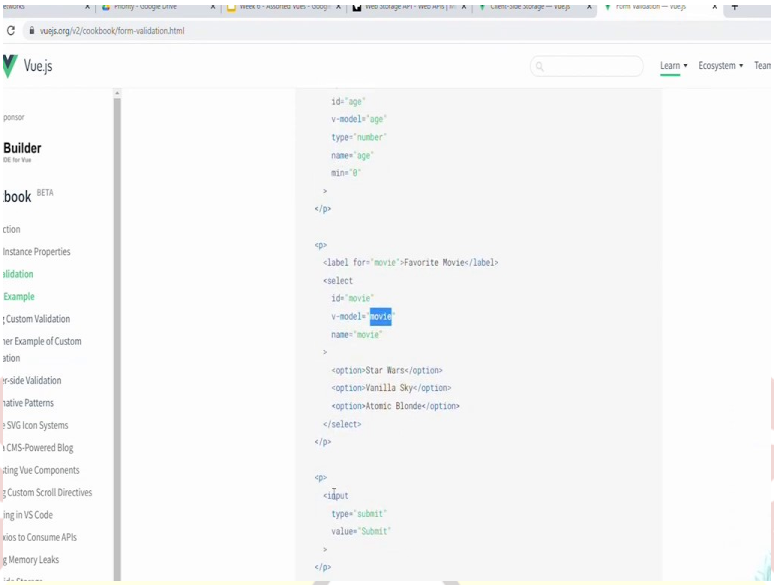
Now, what all this means is that we can also, write custom validation routines. So, for example, I could have an email check that, let us say checks whether you are coming from a specific domain or that there are a certain number of characters in your username. Or I could check for certain overall conditions to be satisfied. Let us say I have many numerical forms do they all add up to a given amount or less than a certain total? All of those can also, be checked.

Now, in these cases, one thing you need to do is prevent the regular form validation, you basically want to add these entries that says no validate equal to true which will basically result in the default form validation being turned off. Otherwise, the email when you specify a textbox, as email, the browser automatically tries to validate it as an email.

Here, you are turning it off and explicitly saying I will take care of the validation on my own. So, once again, let us take a look at something from the Vue cookbook to see how this can work in practice.

(Refer Slide Time: 8:01)





So, form validation, we are talking about form validation, not just the model validation, which means that we are assuming that we actually have a form, and as you can see, there has to be an action associated with it. And in this case, what they are saying is that if this particular form is submitted, it is going to perform a post operation to this URL http://vuejs.org/. Now, what happens when you post to vuejs.org?

Nothing actually, it is pretty much going to just ignore that. Which is why in this case, they have sort of just put it in there as an example. So, now let us look at the rest of this HTML, you can see that there is a nice snippet out here that basically, creates a paragraph, but that paragraph will exist only if there is this variable errors, which has a length property, which means that it is probably an array.

And only if that errors dot length is nonzero, meaning that it evaluates to true, keep in mind that in JavaScript, something which is 0 will evaluate to false or something which is not defined will evaluate to false. So, if errors is an array with no elements in it, errors dot linked will basically, evaluate to false in this context, which means that this entire paragraph will just disappear. It does not even show up in the final DOM.

Instead of VF, you could use a V show or something like that. But the VF is probably better it actually takes it out of the DOM entirely. Now, what are you doing with the rest of the thing you have defined another paragraph where there is an input, which in this case is a text box basically.

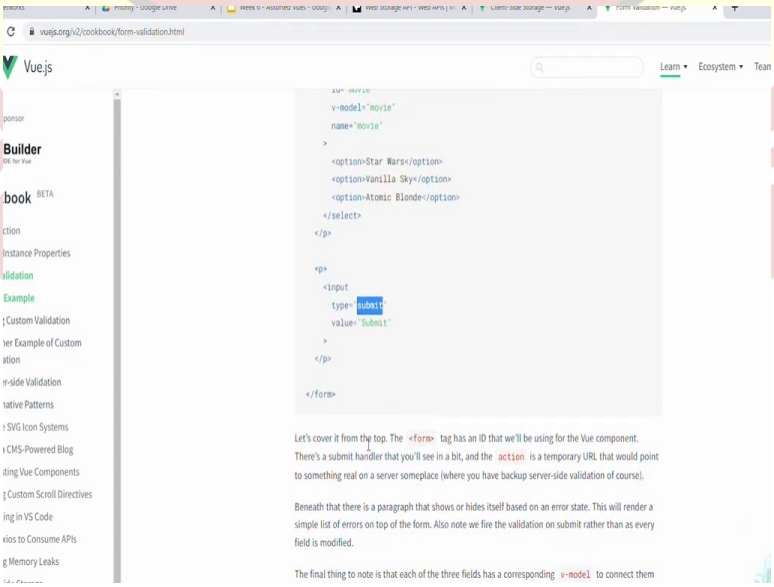
It has type equal to text, and is associated with this model Name, which means that inside the Vue app there is a variable called name which this input text boxes associated with.

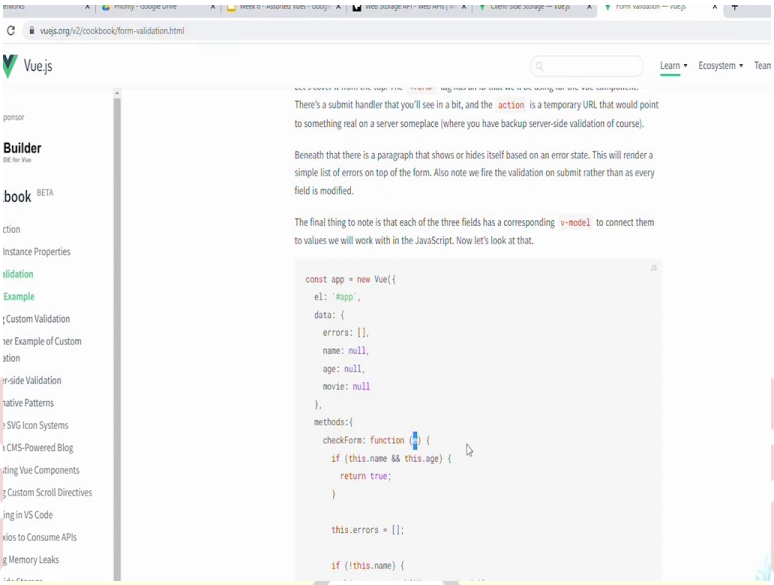
There is one more which is, again specified as an input is told to way of type number. Now this is interesting because what happens when you specify something here to be of type number, the browser essentially shows it to you with one extra box, which allows you to one extra handle on the side of the box that allows you to just click up and down arrows to increase or decrease the value of the number.

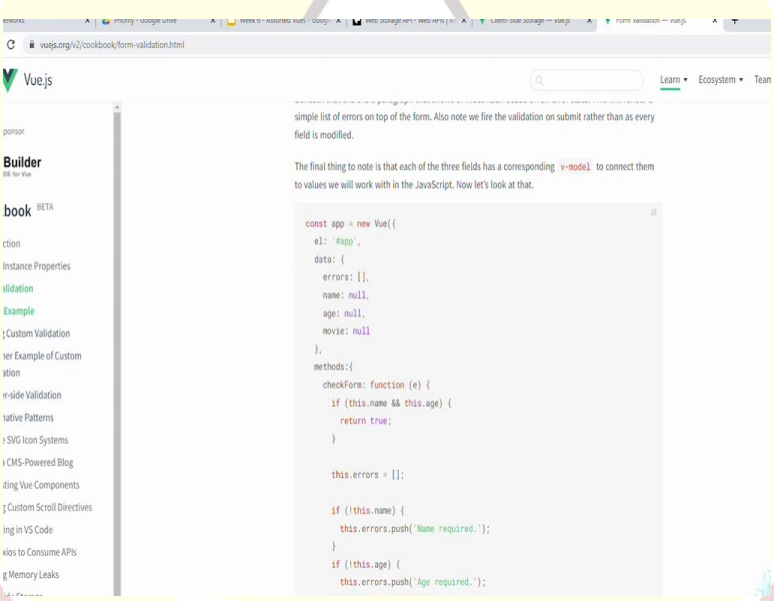
The V model over here is age, which means it is connected to the Vue app age. And there is also, a select box out here, which is just some movies. And the interesting thing is, you can also, have a V model associated with that. You have a bunch of options, and this V model will take the value of the correct option, whichever one it is. And finally, the important part over here is you do need to have this input with type equal to submit. Why?

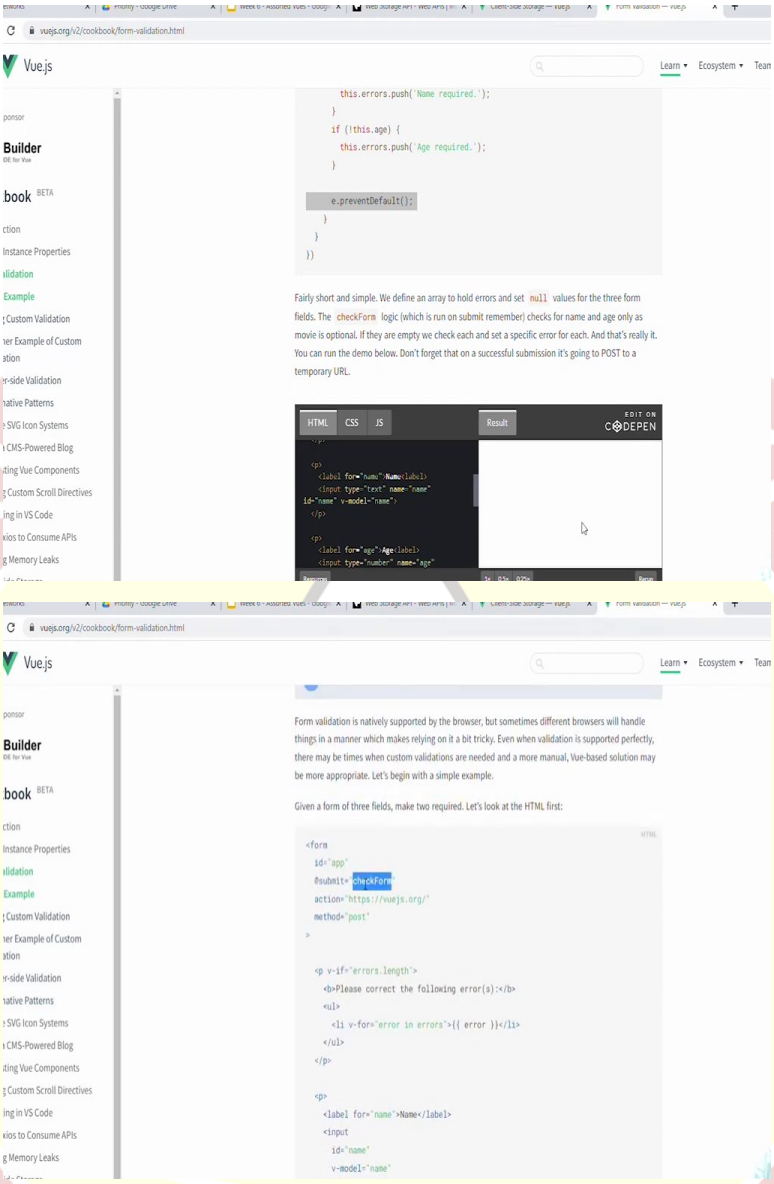
Because go back over here and take a look at it. What we have done is that whenever the at submit, that is the beyond colon, submit equals, that is essentially what this is a short form for, whenever the submit event happens on this form. This particular function check form is going to be called which means that I need to have this button submit only then it is actually going to trigger that.

(Refer Slide Time: 11:30)




vuejs.org/v2/cookbook/form-validation.html
Vue.js
Learn
Ecosystem
Team
Builder
book BETA
There's a submit handler that you'll see in a bit, and the action is a temporary URL that would point to something real on a server someplace (where you have backup server-side validation of course).
Beneath that there is a paragraph that shows or hides itself based on an error state. This will render a simple list of errors on top of the form. Also note we fire the validation on submit rather than as every field is modified.
The final thing to note is that each of the three fields has a corresponding v-model to connect them to values we will work with in the JavaScript. Now let's look at that.
JS
const app = new Vue({
el: '#app',
data: {
errors: [],
name: null,
age: null,
movie: null
},
methods:{
checkForm: function (e) {
if (this.name && this.age) {
return true;
}
this.errors = [];
if (!this.name) {


vuejs.org/v2/cookbook/form-validation.html
Vue.js
Learn
Ecosystem
Team
Builder
book BETA
simple list of errors on top of the form. Also note we fire the validation on submit rather than as every field is modified.
The final thing to note is that each of the three fields has a corresponding v-model to connect them to values we will work with in the JavaScript. Now let's look at that.
JS
const app = new Vue({
el: '#app',
data: {
errors: [],
name: null,
age: null,
movie: null
},
methods:{
checkForm: function (e) {
if (this.name && this.age) {
return true;
}
this.errors = [];
if (!this.name) {
this.errors.push('Name required.');
}
if (!this.age) {
this.errors.push('Age required.');



Now, what is the Vue part of the code look like? It basically creates a new Vue application, as we can see errors has been declared to be an array, name, age, movie, all of those are declared as nulls. Remember that this is required, if I do not do this, the reactivity is lost, if I have not declared these variable names in the Vue application to start with essentially, what it means is that, I am not going to react to these particular changes in these data elements.

If I create a new variable somewhere later, it will not trigger a change in the document by default. So, anyway, we have also, declared a method write the check form, which is a function. And in this case, it takes in one parameter e and that e is basically the event that triggered this

function. In this case, what was it? It was a submit event? So, I take that e as input to this function. And do I do anything with it? Well, not really, I am not really using it right now.

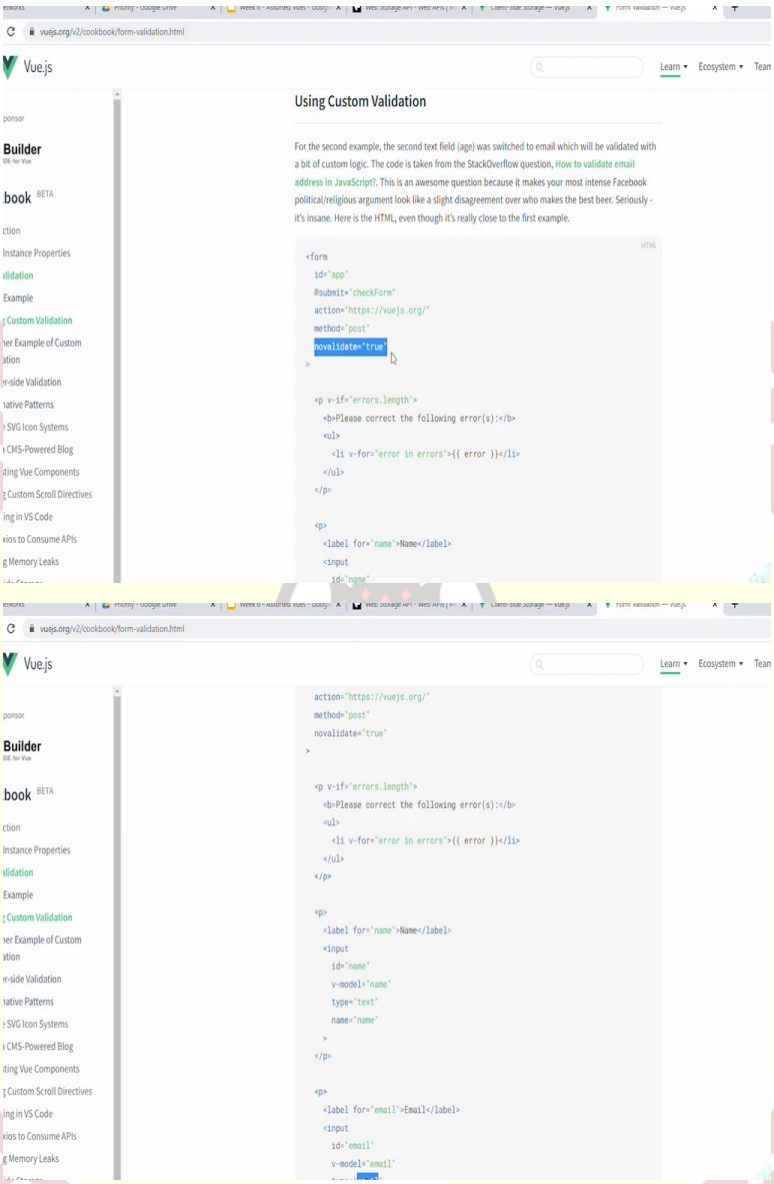
But we will see at the end that there is one particular use for this event that is passed in. The first thing we do is, we straight away check and say that if there is a name, and there is an age. Validation passed immediately return true. no problem. So, we are here. But if that did not happen, clearly one of them went wrong. Say, initialize this dot errors to a null array, write an empty array, and then check each one individually.

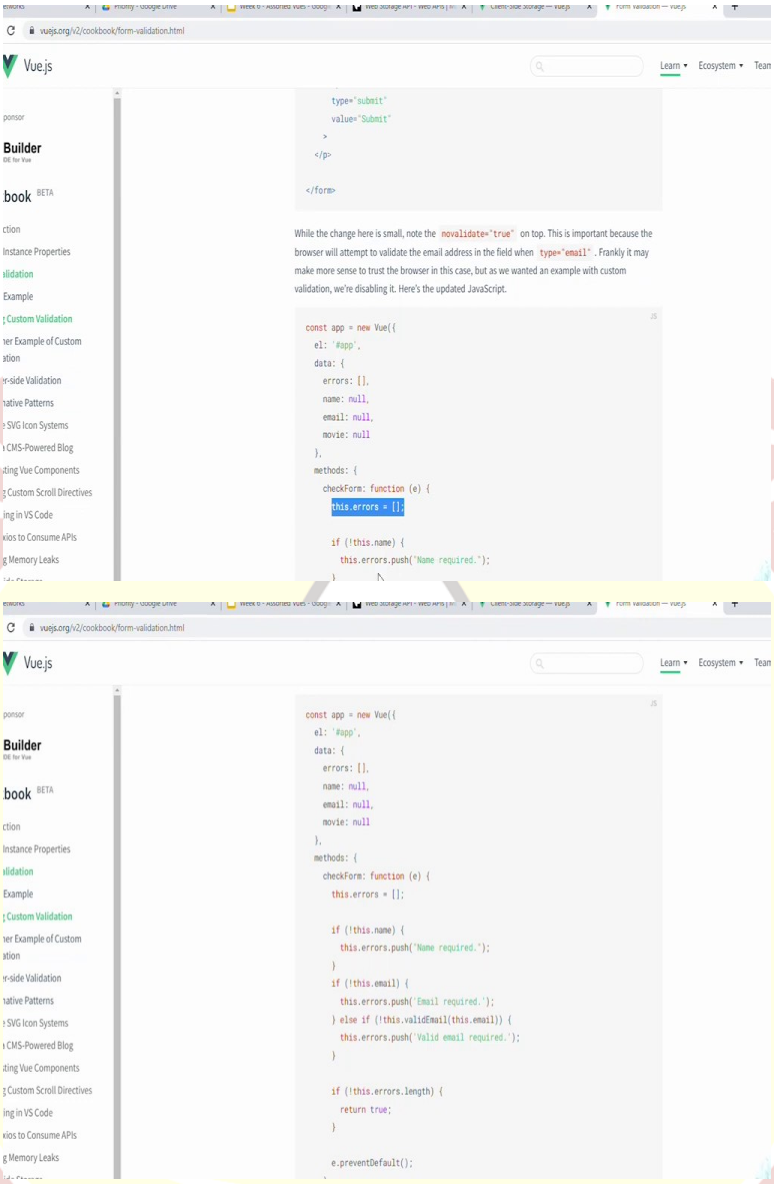
If this dot name is not present, then push a value name required into the error errors array. And if age is not present, then say age required and push that onto the errors array. And finally, see because something went wrong over here I also call e dot prevent default which means that the submit of operations prevent default failed. It will not actually cause a post operation to get executed.

So, what will happen when I run all of this? If I straightaway try typing submit, it immediately gives me the following errors name required age required. Let us say I try typing in something here, right for the name. And I can actually click on will enter which also, ultimately ends up triggering the submit functionality. Now you can see that the error has changed, it became age required.

And you can see what I meant about the age right now there is a button with an arrow out there that I can use in order to change the age. What if I put in an age but I delete the name. It now complains name required. So, it keeps going back and forth over there. And when I have got both of them well, it posts but nothing is really going to come back as a result of that because Vue js dot org by default just ignores posts and post requests which is why the page went blank.

(Refer Slide Time: 14:47)





So, let us take this further and say what happens if we want to do custom validation. It is exactly the same form as before, but you will notice that there is one extra entry out here the no validate equal to true. Now why is that? The reason is that instead of name and age, we now have name and email. And you can see that type is equal to email. Now, most modern browsers have built in validation.

And the moment that you say type is equal to email, it looks for something that looks like a valid email address and will reject it if you do not for example, have an add sign. And the information required for an email. So, the no validate is equal to true is set over here in the form to prevent

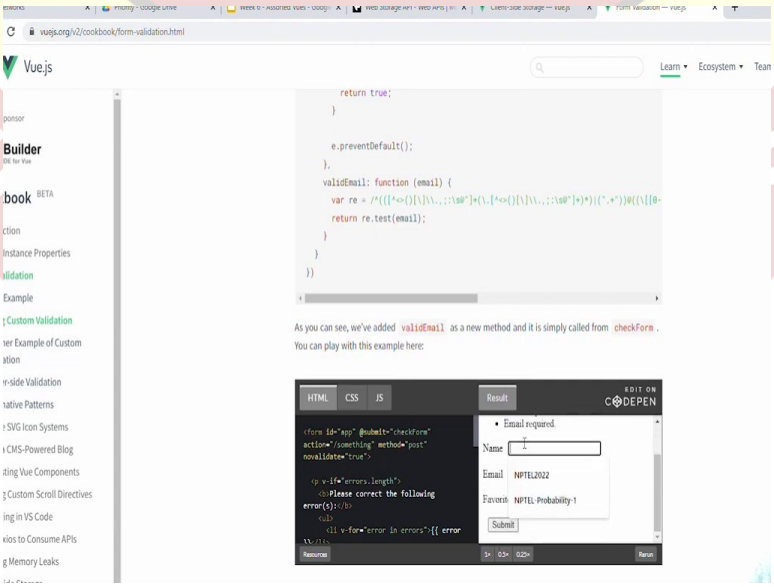
that from happening. We do not want the browser to intercept anything. Instead, what we do is we go into our JavaScript and in check form.

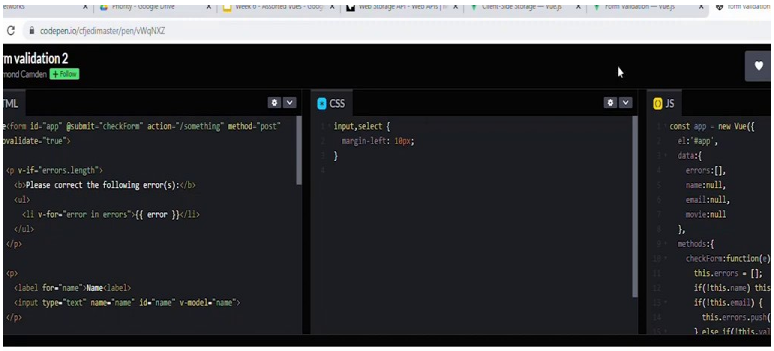
Now we do something slightly different, we basically start by setting the errors to 0. If there is a problem with the name, we basically push name required into the errors list. And if there is no email, then it says email required, and then it checks if there are no errors returned true. If there was an error, it will prevent default. So, this is another way of doing it. I mean, I could have, changed things around and sort of done it.

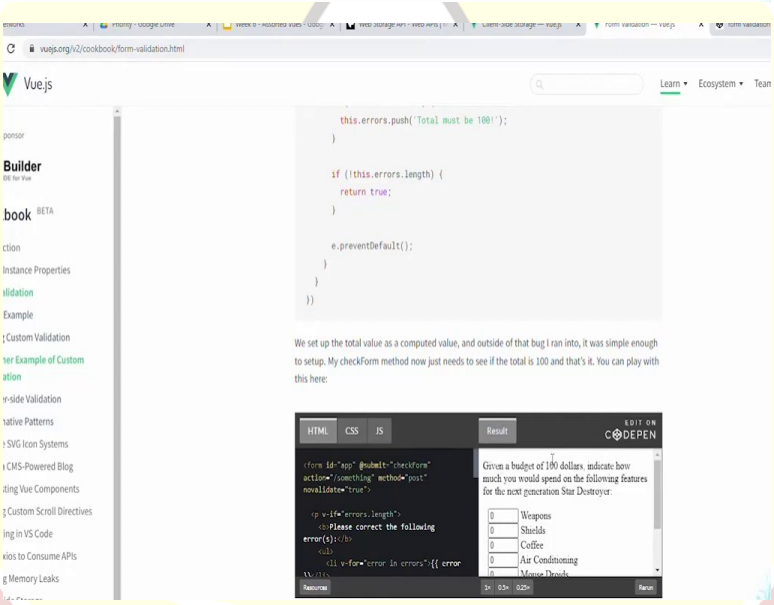
So, that I, in the previous code, what did I do, I basically had the check for name and age, whether they are present right at the beginning. This is probably a better way because you are checking each one of them individually. And only if there are no errors at the end, it returns true. But if there is even one error, it will prevent default, and basically return false in this case. It will not allow the form to submit.

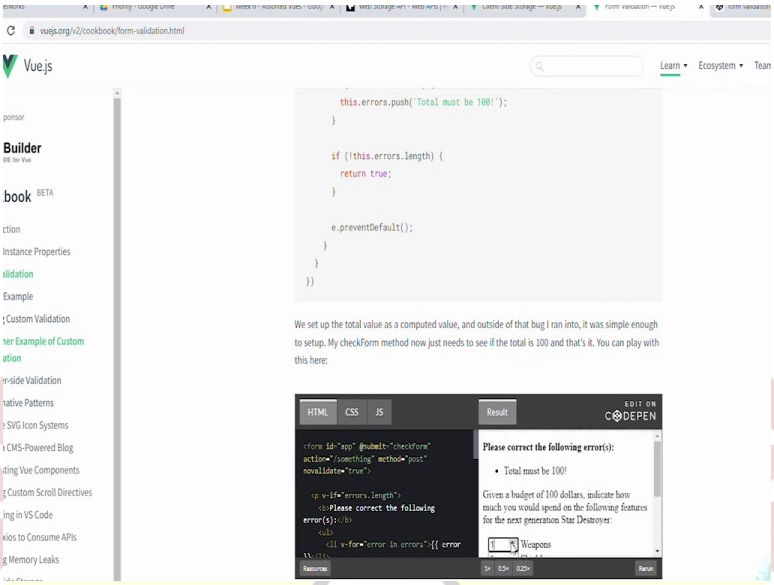
So, the valid email by itself is a function, it uses a regular expression, I am not even going to try to understand how this is working to validate it as an email. But whatever, apparently, this is something which has undergone a lot of discussion on StackOverflow. And for us it could be anything, I could even decide that my validate function just checks whether my name is present there.

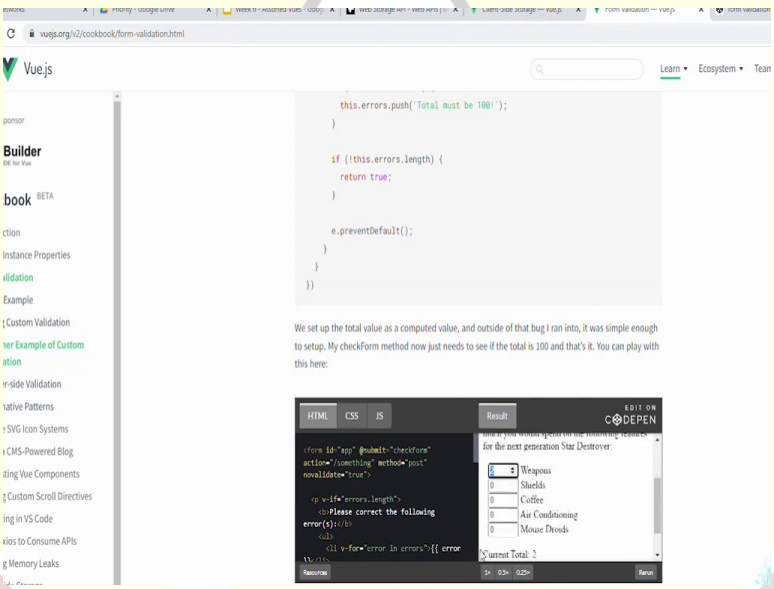
(Refer Slide Time: 17:05)



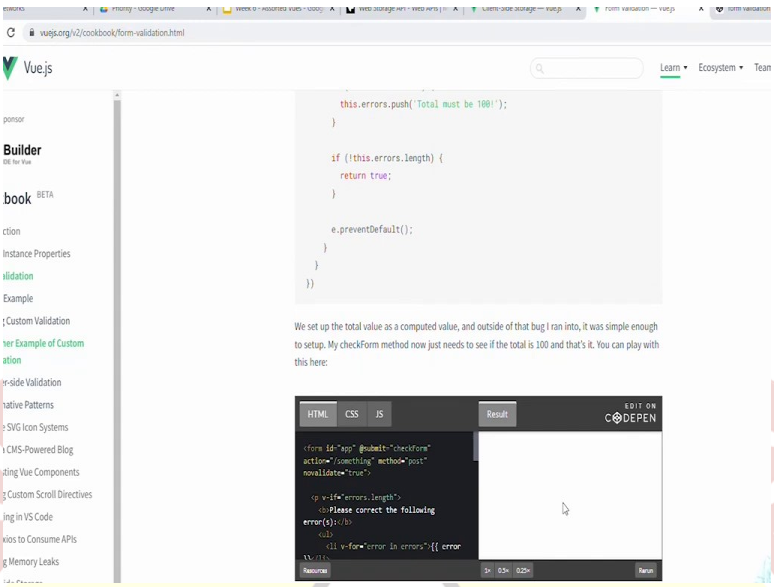

codepen.io/cfjedimaster/pen/vWqNXZ
m validation 2
HTML
CSS
input,select {
margin-left: 10px;
}
JS
const app = new Vue({
el:'#app',
data:{
errors:[],
name:null,
email:null,
movie:null
},
methods:{


vuejs.org/v2/cookbook/form-validation.html
Vue.js
Learn
Ecosystem
Builder
this.errors.push('Total must be 100!');
if (!this.errors.length) {
return true;
e.preventDefault();
We set up the total value as a computed value, and outside of that bug I ran into, it was simple enough to setup. My checkForm method now just needs to see if the total is 100 and that's it. You can play with this here:
HTML
CSS
JS
Result
EDIT ON CODEPEN
Given a budget of 100 dollars, indicate how much you would spend on the following features for the next generation Star Destroyer:
Weapons
Shields
Coffee
Air Conditioning


vuejs.org/v2/cookbook/form-validation.html
Vue.js
Learn
Ecosystem
Builder
book BETA
Instance Properties
Example
Custom Validation
Server-side Validation
SVG Icon Systems
CMS-Powered Blog
Custom Scroll Directives
Memory Leaks
this.errors.push('Total must be 100!');
if (!this.errors.length) {
return true;
e.preventDefault();
We set up the total value as a computed value, and outside of that bug I ran into, it was simple enough to setup. My checkForm method now just needs to see if the total is 100 and that's it. You can play with this here:
HTML
CSS
JS
Result
EDIT ON CODEPEN
<form id="app" @submit="checkForm" action="/something" method="post" novalidate="true">
<p v-if="errors.length">
Please correct the following error(s):

<li v-for="error in errors">{{ error
Please correct the following error(s):
Total must be 100!
Given a budget of 100 dollars, indicate how much you would spend on the following features for the next generation Star Destroyer:
Weapons
Resources
Rerun

vuejs.org/v2/cookbook/form-validation.html
Vue.js
Learn
Ecosystem
Builder
book BETA
Instance Properties
Example
Custom Validation
Server-side Validation
SVG Icon Systems
CMS-Powered Blog
Custom Scroll Directives
Memory Leaks
this.errors.push('Total must be 100!');
if (!this.errors.length) {
return true;
e.preventDefault();
We set up the total value as a computed value, and outside of that bug I ran into, it was simple enough to setup. My checkForm method now just needs to see if the total is 100 and that's it. You can play with this here:
HTML
CSS
JS
Result
EDIT ON CODEPEN
<form id="app" @submit="checkForm" action="/something" method="post" novalidate="true">
<p v-if="errors.length">
Please correct the following error(s):

<li v-for="error in errors">{{ error
for the next generation Star Destroyer:
Weapons
Shields
Coffee
Air Conditioning
Mouse Droids
Current Total: 2
Resources
Rerun

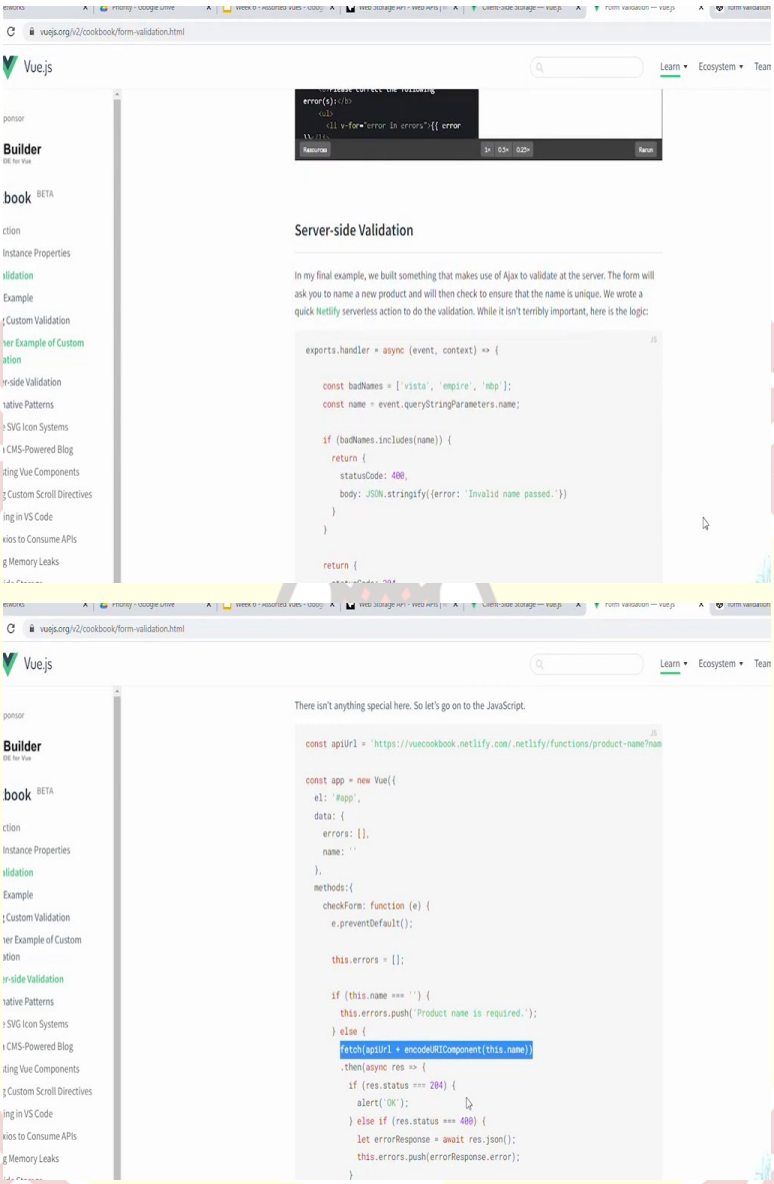


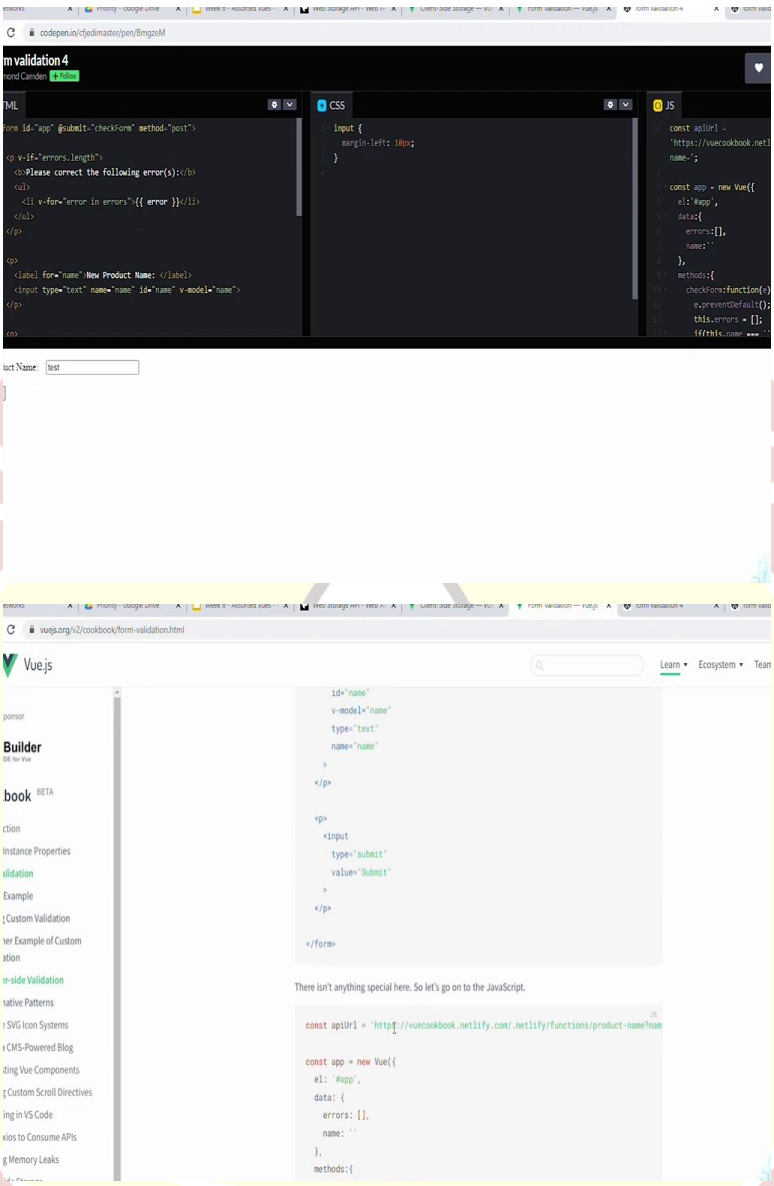
The bottom line is when I try running this once again, it says name required email required, I try typing in something here. Seems to be some problem with the box over there. I think we open it on this, it probably works better. And if I submit this, it says email required. So, that does not work. What happens if I go and type in an email like this says valid email required. So, test at abc still says valid email required? On the other hand, this makes it accepted.

So, in other words, we could have something which sort of goes through the process of validation and can sort of do a custom validation when needed. Another nice example, which is there on the same page basically asks you to enter several numbers, given a budget of 100 dollars indicate how much you would spend on the on each of these features for something. This is, I guess, something like a StarWars game or something like that.

And, it allows you to enter a few different numbers over here. If we submit this, it immediately says total must be 100. So, it does not matter what numbers I put in here, as long as the current total is less than 100, it is going to show me that error. But the moment I put in an actual number out there, and the current total becomes 100. It goes through, once again, it posts to some random page. So, it is not like you can see what happens but it succeeded.

(Refer Slide Time: 18:59)





Now finally, this is another example of even more complex validation. And what they have done over here is they have done something called a server side validation. And what they are doing in the server side validation is that they actually perform a fetch operation. And depending on whether or not that fetch operation actually works. You will get something which either passes or fails.

Now, there is a small problem with the code when corresponding to this. So, I am going to just edit it out here. What you will see is that in the JavaScript, they have given their API URL if you try typing in something now write test or something like that. It does not do anything. Ideally, it

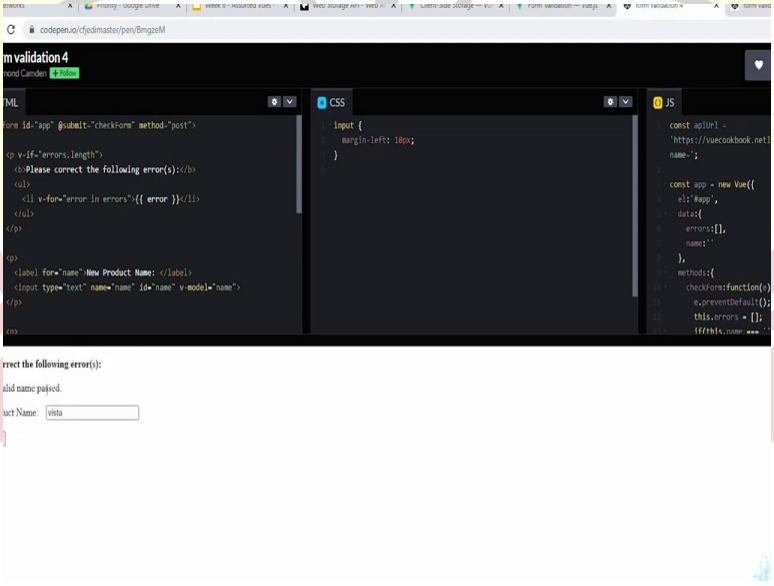
should have performed a post and you should have seen that blank page, but that is not happening.

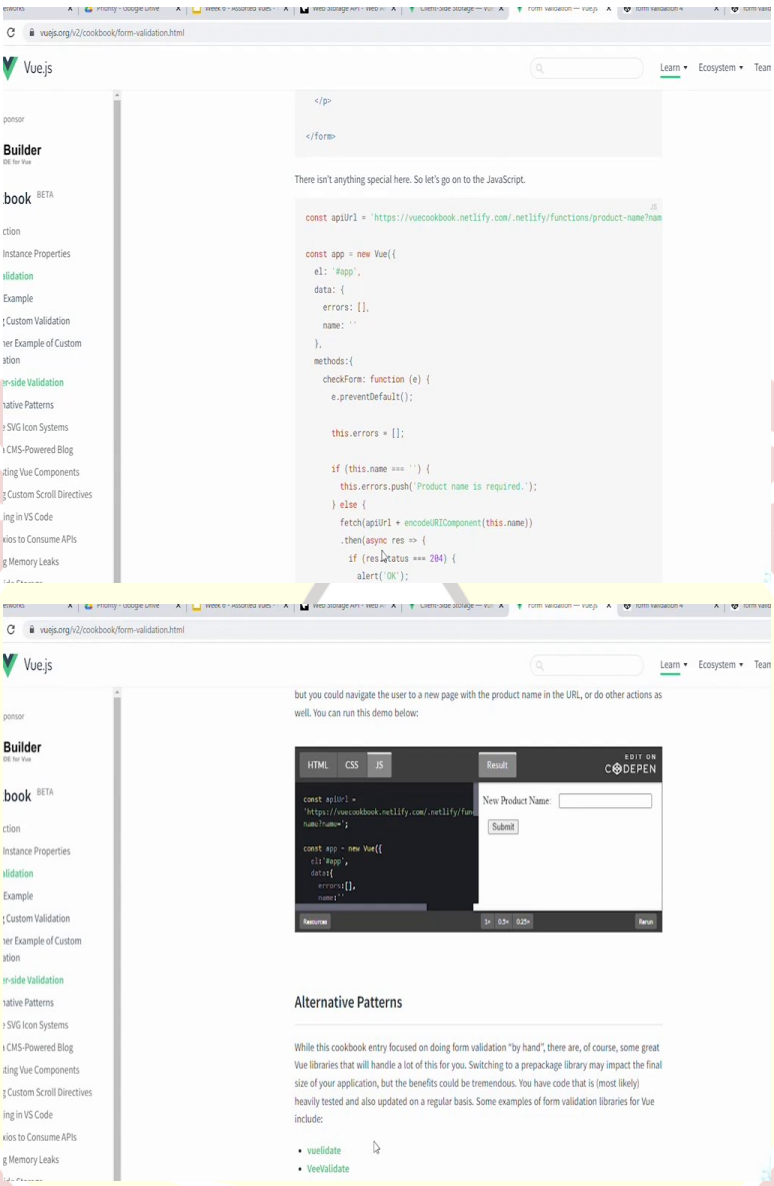
Actually not performed the post what it should have done according to the code out here. Is that if it succeeded. It should have shown an alert, whereas if it failed, it should basically, show you the response and error response. Now, how is it doing it? This is related to the fetch API that we discussed earlier. It actually performs a fetch on some URL, what is this URL, this is corresponding to some function, which they have already implemented on Netlify.

Which is a server that allows you to implement certain limited kind of applications for free and others on a paid basis. So, they have implemented this small sort of, this piece of JavaScript that either validates and returns either a 400 or a 204. And that is basically all that it does, it will return only one of these two error codes. So, we know that in our fetch API, the answer should be either 204, or 400. So, when we go and look at it, we should see one of those two coming.

Now what has happened is that Netlify changed the naming system, which means that instead of Netlify.com.

(Refer Slide Time: 21:05)





You need to actually go and change this to Netlify dot app. And once we have that, and you type in a valid name. It says, Okay, you will notice that there was a pause, it basically took a little bit of time before the okay came, that is because it basically had to go connect to Netlify and get the response back. On the other hand, there are some invalid names according to the function that they wrote.

And when you do that, it basically says invalid name passed. So, what was the purpose of this, basically it is telling you that the form validation can do something much more complicated than just, checking, running a JavaScript code locally. In the context of, let us say, the whether

checking application, you could have something where the moment I enter a new city, it actually first performs a check confirms that there is such a city.

And other that there is a city for which you can retrieve data based on this information. And only if there is it will show you the result, it will not even let you add the city otherwise. And as you can see, this is actually a fetch, which means that it is an asynchronous operation. It takes a certain amount of time for it to run. But all of that is taken care of pretty much nicely behind the scenes by the fact that, the validation is happening cleanly over there.

So, all this is the most basic form of validation. It turns out, there are also custom sort of libraries for Vue, v validate Vue redate, that provide even more features. So, including like how do you sort of manage and display error messages, all of those things are taken care of quite nicely by these libraries. So, if you want to do form validation in a way that is significantly more powerful than what a browser can do. There are support for that in the structure that Vue provide this for you.