

IIT Madras

ONLINE DEGREE

Modern Application Development - II
Professor Thejesh G N
Software Consultant
Indian Institute of Technology, Madras
Components in VueJS

(Refer Slide Time: 00:14)



```
-code/vue_basics_2/application.html (application) - Sublime Text
File Edit Selection Find View Tools Project Preferences Help
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Hello World</title>
    <script src="https://cdn.jsdelivr.net/npm/vue@2/dist/vue.js"></script>
  </head>
  <body>
    <div id="app">
      <p> {{ message }} </p>
      Your name: <input type="text" v-model="visitor_name"/>
      <p> v-if="count == 0" No replies yet </p>
      <p> v-else-if="count > 3"> You are very popular </p>
      <p> v-else> Number of replies {{ count }} </p>
      <button v-on:click="sayHi">Say Hi</button>
      <br>
      <ul>
        <li v-for="name in visitors">{{name}}</li>
      </ul>
    </div>
  </body>
  <script type="text/javascript" src="application.js"></script>
</html>

L1n 1 Gramma
File Edit Selection Find View Tools Project Preferences Help
<!DOCTYPE html>
<html>
  <head>
    <title>Hello World</title>
  </head>
  <body>
    <div id="app">
      <h1>Hello World</h1>
      <input type="text" v-model="visitor_name" />
      <ul>
        <li v-for="name in visitors">{{name}}</li>
      </ul>
    </div>
  </body>
</html>
```

The image features three vertically stacked screenshots of a Sublime Text editor window, each displaying a different file from a Vue.js application. The background of the entire image is a large circular watermark containing the text "INDIAN INSTITUTE OF TECHNOLOGY MADRAS".

File 1: application.js

```

1 var app = new Vue({
2   el: "#app",
3   data: {
4     visitor_name: '',
5     visitors: []
6   },
7   methods: {
8     sayHi: function() {
9       this.visitors.push(this.visitor_name);
10      this.visitor_name = '';
11    }
12  },
13  computed: {
14    count: function() {
15      return this.visitors.length;
16    }
17  }
18 })
19

```

File 2: application.html

```

1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="utf-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1">
6   <title>Hello world</title>
7   <script src="https://cdn.jsdelivr.net/npm/vue@2/dist/vue.js"></script>
8 </head>
9 <body>
10 <div id="app">
11   <input type="text" v-model="visitor_name"/>
12   <p>v-if="count == 0"> No replies yet </p>
13   <p>v-else-if="count > 3"> You are very popular </p>
14   <p>v-else> Number of replies {{ count }} </p>
15   <button v-on:click="sayHi">Say Hi</button>
16   <ul>
17     <li v-for="name in visitors">{{name}}</li>
18   </ul>
19 </div>
20 </body>
21 <script type="text/javascript" src="application.js"></script>
22 </html>

```

File 3: application.html

```

1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="utf-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1">
6   <title>Hello world</title>
7   <script src="https://cdn.jsdelivr.net/npm/vue@2/dist/vue.js"></script>
8 </head>
9 <body>
10 <div id="app">
11   Your name: <input type="text" v-model="visitor_name"/>
12   Your message: <input type="text" v-model="visitor_message"/>
13   <button v-on:click="sayHi">Say Hi</button>
14   <ul>
15     <li v-for="name in visitors">{{name}}</li>
16   </ul>
17 </div>
18 </body>
19 <script type="text/javascript" src="application.js"></script>
20 </html>

```

The image shows three vertically stacked code editors from the Sublime Text application, each displaying a different file related to a Vue.js application.

File 1: application.js

```

1 var app = new Vue({
2   el: "#app",
3   data: {
4     visitor_name: '',
5     visitor_message: '',
6     visitors: []
7   },
8   methods: {
9     sayHi: function() {
10       this.visitors.push(this.visitor_name);
11       this.visitor_name = '';
12     }
13   },
14   computed: {
15     count: function() {
16       return this.visitors.length;
17     }
18   }
19 })
20 
```

File 2: application.js

```

1 var app = new Vue({
2   el: "#app",
3   data: {
4     visitor_name: '',
5     visitor_message: '',
6     messages: []
7   },
8   methods: {
9     sayHi: function() {
10       this.messages.push( { "visitor_name": this.visitor_name, "visitor_message": this.visitor_message });
11       this.visitor_name = '';
12       this.visitor_message = ''
13     }
14   },
15   computed: {
16     count: function() {
17       return this.messages.length;
18     }
19   }
20 })
21 
```

File 3: application.html

```

1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="utf-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1">
6     <title>Hello world!</title>
7     <script src="https://cdn.jsdelivr.net/npm/vue@2/dist/vue.js"></script>
8   </head>
9   <body>
10     <div id="app">
11       Your name: <input type="text" v-model="visitor_name"/>
12       Your message: <input type="text" v-model="visitor_message"/>
13       <button v-on:click="sayHi">Say Hi</button>
14       <ul>
15         <li v-for="message in messages">{message['visitor_name']} : {message['visitor_message']}</li>
16       </ul>
17     </div>
18   </body>
19   <script type="text/javascript" src="application.js"></script>
20 </html>
21 
```

The image shows two screenshots of a Sublime Text editor window. Both windows have the title bar '-code/work/vue_basics_2/Application.js (application) - Sublime Text'. The left screenshot shows the initial state of the code, and the right screenshot shows the code after modifications. The code is written in JavaScript and uses Vue.js components.

```

File Edit Selection Find View Code Tools Project Preferences Help
File Edit Selection Find View Code Tools Project Preferences Help
1 <template>
2   <div>
3     <h1>Hello</h1>
4     <input type="text" v-model="visitor_name" />
5     <button @click="sayHi">Say Hi</button>
6     <div>
7       <ul>
8         <li>{{ visitor_name }} : {{ visitor_message }}</li>
9       </ul>
10      <div>Count: {{ count }}</div>
11    </div>
12  </div>
13 </template>
14 <script>
15   var app = new Vue({
16     el: '#app',
17     data: {
18       visitor_name: '',
19       visitor_message: '',
20       messages: []
21     },
22     methods: {
23       sayHi: function() {
24         this.messages.push( { "visitor_name": this.visitor_name, "visitor_message": this.visitor_message } );
25         this.visitor_name = '';
26         this.visitor_message = ''
27       }
28     },
29     computed: {
30       count: function() {
31         return this.messages.length;
32       }
33     }
34   })

```

Welcome to the Modern Application Development II screencast. In this screencast, we will continue to experiment with our app, and componentize it, and see, how we can improve it by using various features of Vue. For this, you will need an editor. I am using Sublime Text here. As you can see, you need a browser. I have Firefox here, I have already installed the Vue, add-on, so that we can experiment with it. And, yeah, you can see here.

Let us start. To begin with the we will remember doing this app where every step can leave a name, and say hi, and then it adds up and shows the things. I have continued to use the same application, you can see the code here, this is a HTML part, this is a JS part, and this is same file that I have opened here. And if you can enter your name, say hi, it shows your name here and number of replies. Let us improve this a little bit.

You know, what I will do is, I will remove this message this top message we do not require. And I am going to remove this. So, I am going to add another attribute incoming attribute

called visitor message. Let us say we want to live name and message. And then, let us assume that we do not need to say any of this, we will just say, hi. And the say hi message is there, then visitor name, visitor message.

Let us call these messages, let us store all the messages visitor leaves, along with their name and actual message. So, this total message, we will make it plural, messages will have when the hi called it will store both the name and the message. So, let us call this attributor as visitor name and then no visitor messages, visitor message. This will store this dot visitor message and then we will empty that, we will empty this, we will this. And yeah, let this be there. I think this will not become visitors dot length become messages dot count.

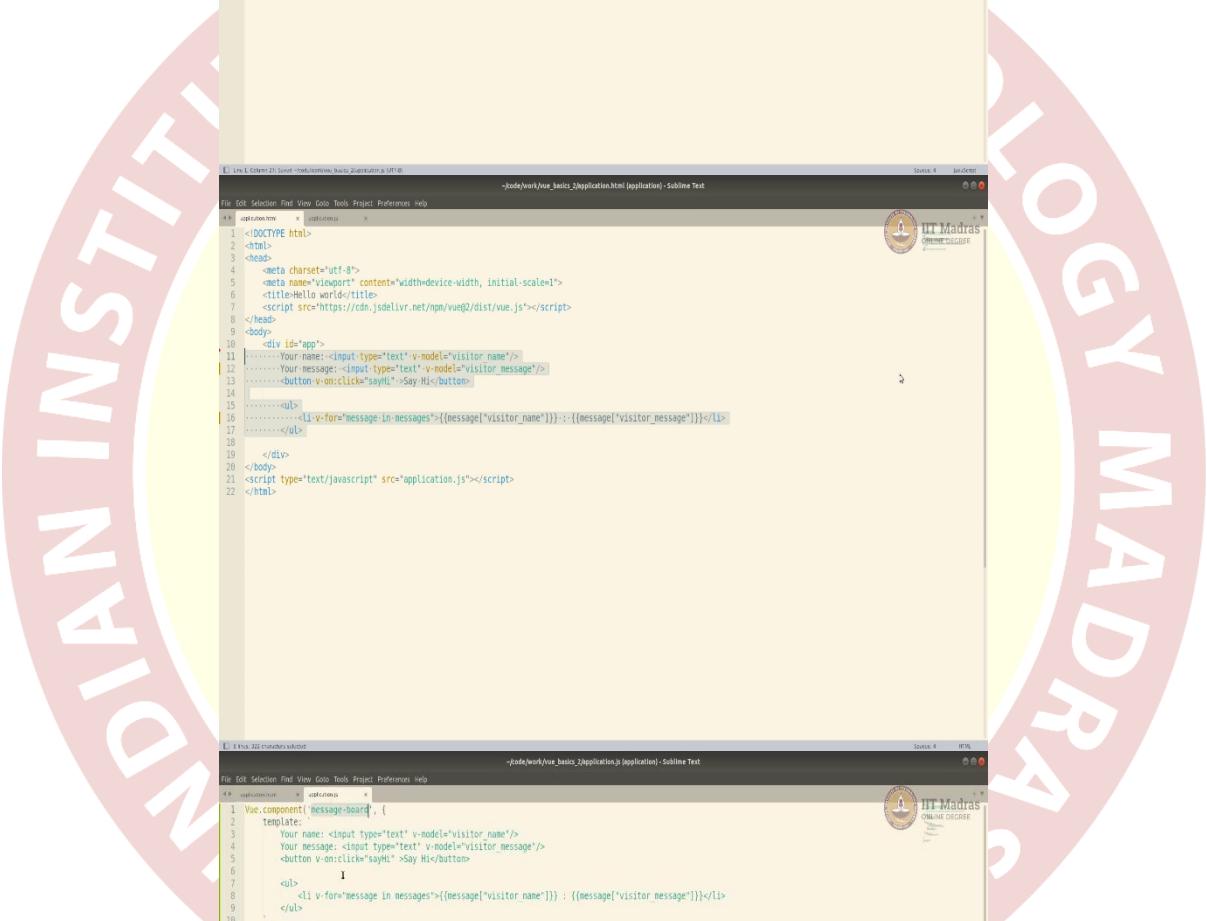
Okay, here let us just check once, messages. I am adding to messages when the hi is called. Okay, here, I think we are iterating through visitors, we will not be iterating through visitors anymore we will be iterating through messenger and each one called a message. And we can print message here, but we will not print the object we will print the this one visitor name and what message he left?

That will be message open bracket, this is just attribute. I think this should work. Let us just check. Reload, so there it is. Your name, let us say Tej welcome, say hi. So, it is showing Tej and welcome.

Now, let us assume. I mean, this is not really like a production level application, but let us assume that there are three folks who are hosting this website, and you want three books or three guest books where folks can leave a message for three folks. For example, there is let us say Tej's, messages board, Raj's messages board and Fatima's message board. And we want the same thing appear thrice, so that all of them can receive like a message here. It is a simple thing. We are not storing anywhere, it is just a display, but we let us say we have three boards, that would make you want to repeat all of this thrice, and write logic for that and all of that. The easier way to do that would be to convert this into a component.

Now, it is very easy to convert the Vue application into a component. We will have to just define a new Vue component. That can be done very easily, by calling Vue dot component. Like the way we have a new Vue, we will call Vue component. Now Vue component actually takes two important attributes. One is a name, what you want to call the name to be, we want to call it message board. Let us called message board and then it takes optional parameters, just like our app.

(Refer Slide Time: 05:57)



The image shows three screenshots of Sublime Text editors, each displaying a different file from a Vue.js application project:

- File 1 (application.js):** Contains the main Vue component definition. It sets up a new Vue instance with an element selector '#app', data properties for visitor name and message, and an array for messages. It includes methods for 'sayHi' (which pushes a new message object into the messages array) and 'count' (which returns the length of the messages array).

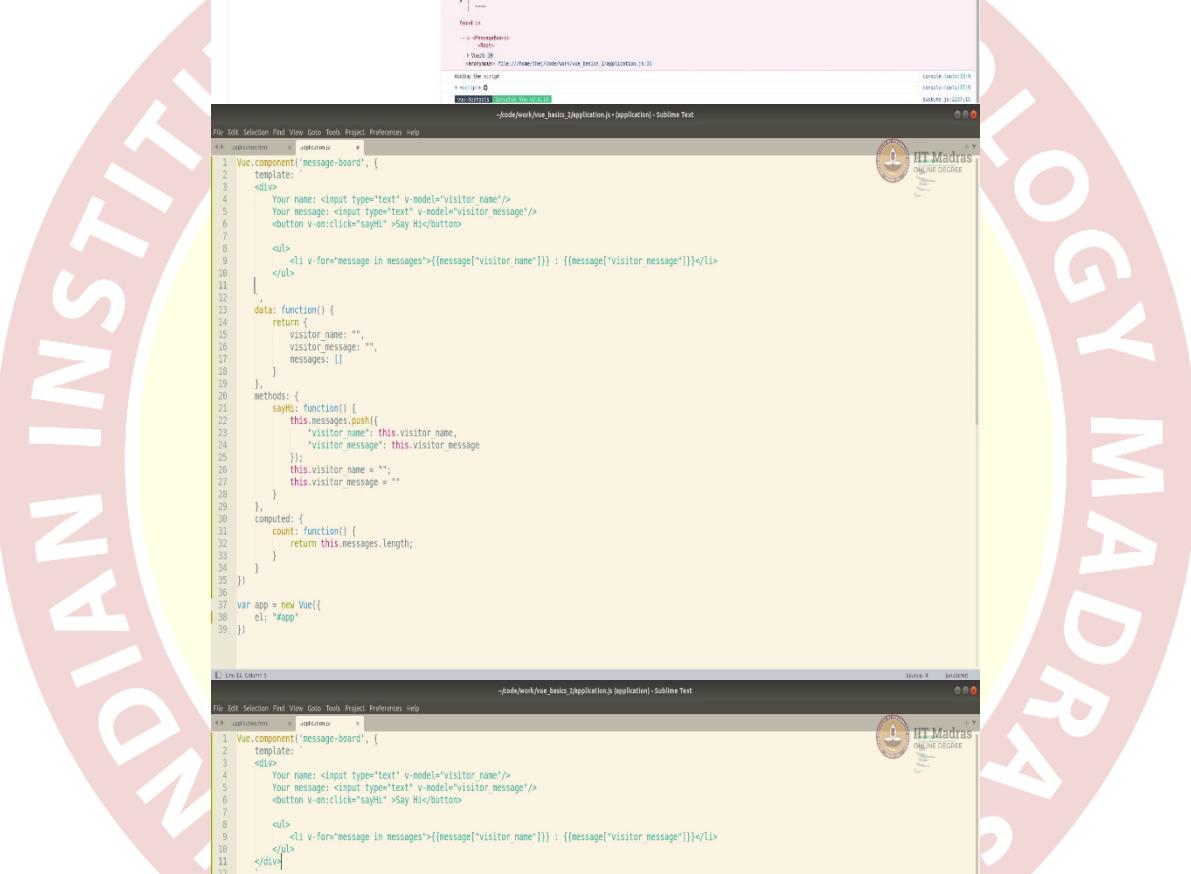
```
1 Vue.component('message-board', {
2   template: `
3     <div id="app">
4       <input type="text" v-model="visitor_name"/>
5       <input type="text" v-model="visitor_message"/>
6       <button v-on:click="sayHi">Say Hi!</button>
7       <ul>
8         <li v-for="message in messages">{{message.visitor_name}} : {{message.visitor_message}}</li>
9       </ul>
10    </div>
11  </div>
12  <script type="text/javascript" src="application.js"></script>
13 </body>
14 </html>
```
- File 2 (application.html):** Contains the HTML structure for the application. It includes a DOCTYPE declaration, meta tags for charset and viewport, a title, and a script tag pointing to a CDN URL for Vue.js.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>Hello world</title>
<script src="https://cdn.jsdelivr.net/npm/vue@2/dist/vue.js"></script>
</head>
<body>
<div id="app">
<input type="text" v-model="visitor_name"/>
<input type="text" v-model="visitor_message"/>
<button v-on:click="sayHi">Say Hi!</button>
<ul>
<li v-for="message in messages">{{message.visitor_name}} : {{message.visitor_message}}</li>
</ul>
</div>
<script type="text/javascript" src="application.js"></script>
</body>
</html>
```
- File 3 (application.css):** Contains CSS styles for the application. It includes a global style for the body and a specific style for the 'message-board' component's template.

```
body {
  font-family: sans-serif;
}

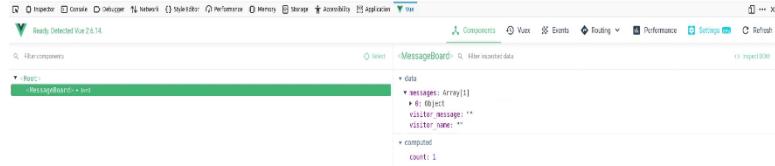
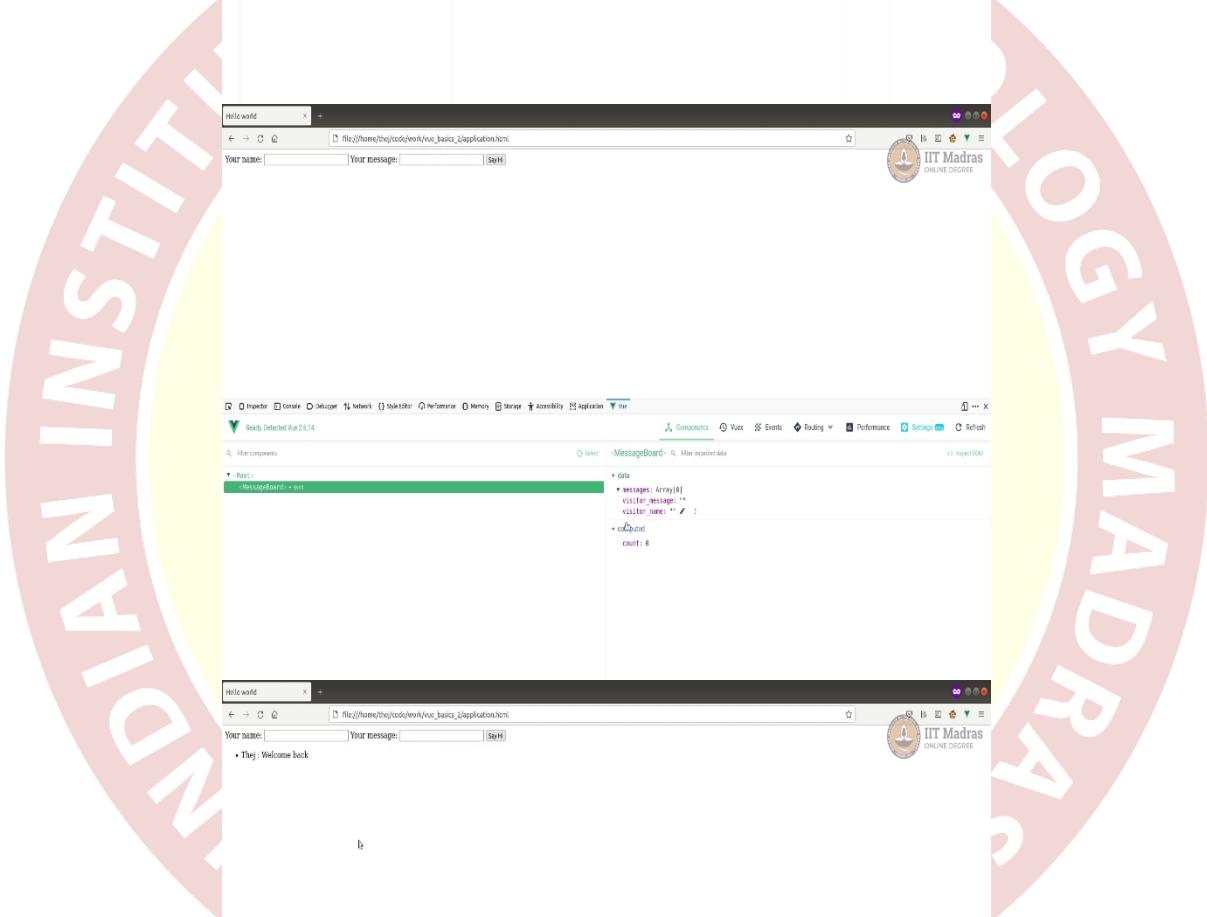
.message-board {
  border: 1px solid black;
  padding: 10px;
}
```





```
File Edit Selection Find View Code Tools Project Preferences Help
File Edit Selection Find View Code Tools Project Preferences Help
 1 <template>
 2   <div>
 3     Your name: <input type="text" v-model="visitor_name"/>
 4     Your message: <input type="text" v-model="visitor_message"/>
 5     <button v-on:click="sayHi">Say Hi</button>
 6   </div>
 7   <ul>
 8     <li v-for="message in messages">{{message["visitor_name"]}} : {{message["visitor_message"]}}</li>
 9   </ul>
10 </div>
11
12 data: function() {
13   return {
14     visitor_name: '',
15     visitor_message: '',
16     messages: []
17   }
18 },
19 methods: {
20   sayHi: function() {
21     this.messages.push(
22       {
23         "visitor_name": this.visitor_name,
24         "visitor_message": this.visitor_message
25       });
26     this.visitor_name = '';
27     this.visitor_message = ''
28   },
29 },
30 computed: {
31   count: function() {
32     return this.messages.length;
33   }
34 }
35 )
36
37 var app = new Vue({
38   el: "#app"
39 })
```

```
File Edit Selection Find View Code Tools Project Preferences Help
File Edit Selection Find View Code Tools Project Preferences Help
 1 <template>
 2   <div>
 3     Your name: <input type="text" v-model="visitor_name"/>
 4     Your message: <input type="text" v-model="visitor_message"/>
 5     <button v-on:click="sayHi">Say Hi</button>
 6   </div>
 7   <ul>
 8     <li v-for="message in messages">{{message["visitor_name"]}} : {{message["visitor_message"]}}</li>
 9   </ul>
10 </div>
11
12 data: function() {
13   return {
14     visitor_name: '',
15     visitor_message: '',
16     messages: []
17   }
18 },
19 methods: {
20   sayHi: function() {
21     this.messages.push(
22       {
23         "visitor_name": this.visitor_name,
24         "visitor_message": this.visitor_message
25       });
26     this.visitor_name = '';
27     this.visitor_message = ''
28   },
29 },
30 computed: {
31   count: function() {
32     return this.messages.length;
33   }
34 }
35 )
36
37 var app = new Vue({
38   el: "#app"
39 })
```



Now, Vue component is easy, we can just do it by just define the Vue component by calling Vue dot component, component. It takes two attributes, the first one being the name, and the

second is set of attributes. Let us say what do you want to call it, let us, you want you want to call it message board because we want each, each one to get a message board, like Tej, Supriya and Fatima to get a message board each, so we can call it message board. Board, now it takes input parameters, just like our Vue app. Now, our Vue app is doing almost, you know, single message board tasks, we have to move all of these into our view component.

Let us do that and see what happens. Okay, we are going to copy the same thing, remove this and paste it here. I am going to format it, so everything remains same, but here, we are now going to use data as an attribute, but data will actually become a function, which returns the data, because we want to keep a set of data separate for each component. Also, it will become a function which returns the data, you can return this. Just formatting a bit.

Now, we have converted data into a function this remains same, this remain same, everything else remains same. Now let us see how to use this. This is what we have defined, here, but the only thing that we missed is how we are going to write the content of the component, which is the template, which is actually here, this part.

So, let us define a template. Template is defined by define the attribute template, use backtick to put component details, and we could just because each message board will contain these details. So, we are going to take this and put this into like our component because this whole thing is going to become component so the whole thing, we are going to move into a template.

Okay, so we have moved into a template. Now, our whole application you kind of componentized into component. Now we can use this component multiple times, unlike our application, which is only one. So, let us see how to use it, and that's quite easy. That you can do by just calling message. What was the name of the component? Message Board?

Message board and slash message board. Okay, let us refresh and see. I have kept things on. Okay, there seems to be some error. Let us see what is the error? Okay, so now it is throwing can error can read the error here. Text your name is outside the root element will be ignored. Basically, there is no root element, there are many parallel elements, there is no root element, because a component can have only one root element and that one root element should be there. We do not have a root element here. So, we are going to just add a div as our root element.

Okay, let us add div as a root element. This should fix the template. Let us refresh this. Okay, error is gone. Now we can see if you actually click on view here and you can see there is root and there is message board. If you click on message board, you can see its attribute and its functions and stuff like that. Now you can give a name, and let us say welcome back, I say, hi, so it shows here.

(Refer Slide Time: 10:57)

The image shows a screenshot of a development environment. At the top, a Sublime Text window displays the code for 'application.html' (application.js). The code includes a DOCTYPE declaration, meta tags for charset and viewport, a title, a script tag for Vue.js, and a main section with a table and a script tag for application.js. Below this, a browser window shows a simple web application with two input fields ('Your name:' and 'Your message:') and a button ('Say Hi'). The application displays a list of messages in a table. At the bottom, a developer tools window (likely Chrome DevTools) shows the Vue.js component tree and the state of the 'MessageBoard' component, which has an array of messages containing one item: {id: 0, message: 'Hello world', name: ''}.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>Hello world</title>
<script src="https://cdn.jsdelivr.net/npm/vue@2/dist/vue.js"></script>
</head>
<body>
<div id="app">
<table>
<tr>
<td>message-board</td>
<td>message-board</td>
<td>message-board</td>
<td>message-board</td>
</tr>
</table>
</div>
<script type="text/javascript" src="application.js"></script>
</body>

```

Ctrl + Shift + I

File Edit Selection Find View Code Tools Project Preferences Help

Source HTML

DevTools Inspector Network Debugger Timeline Profiler Memory Storage Application Components Vue Events Routing Performance Settings Refresh

Detected Vue 2.6.14

Vue.js

MessageBoard

Data

messages: Array[1]

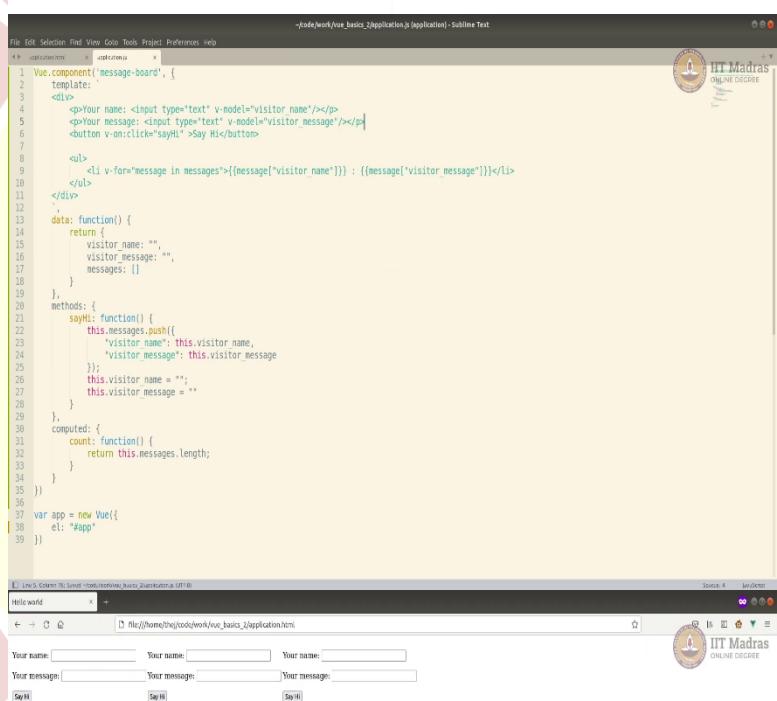
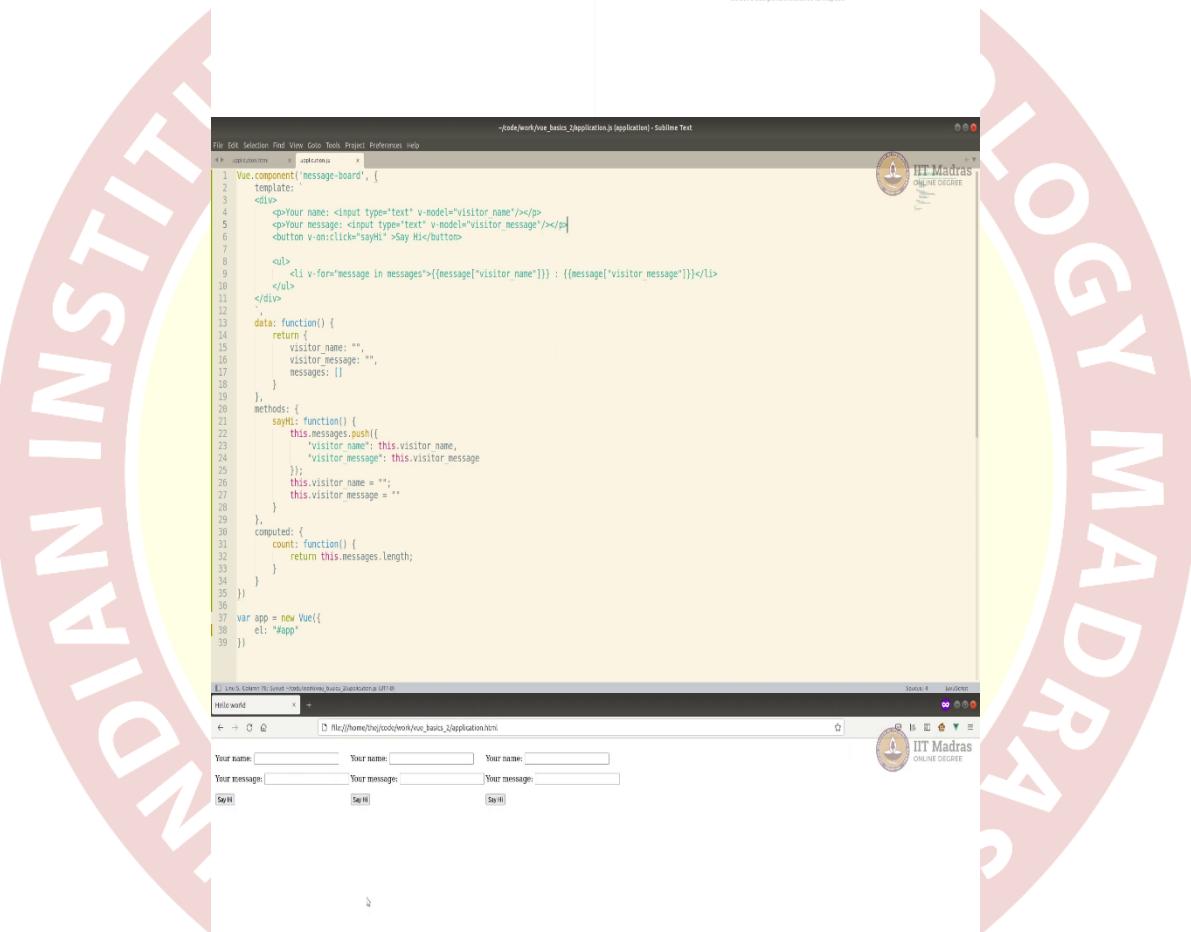
0: Object

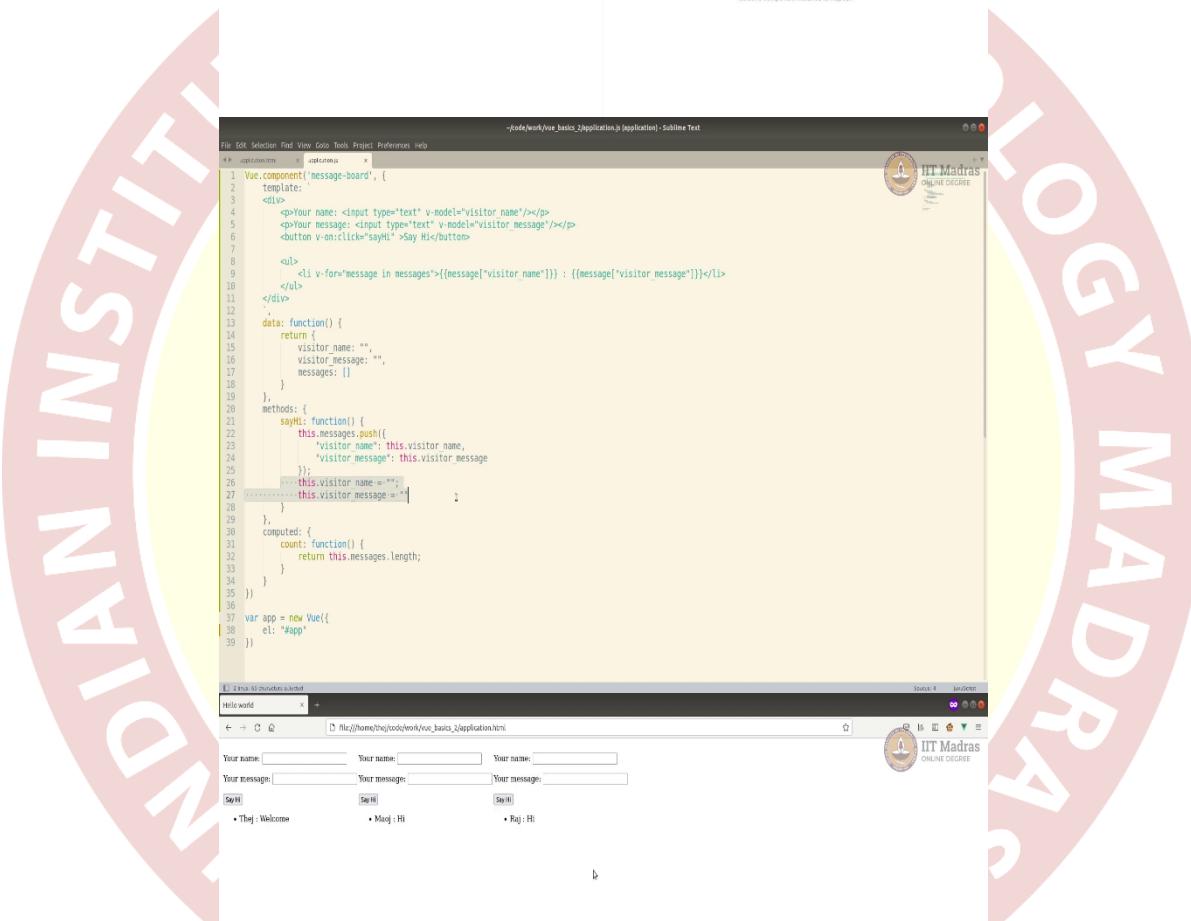
message: "Hello world"

name: ""

computed

Count: 1





Helworld

Your name: Your name: Your name:

Your message: Your message: Your message:

• Thej : Welcome • Raj : Hi • Raj : Hi

React Detected Vue 2.6.14

Components Events Routing Performance Storage Accessibility Application

Home components

Tree:

- Messageboard-
- Messageboard-
- Messageboard-

Select a component instance to inspect.

+code/work/vue_basics_2/Application.js (application) - Sublime Text

```

1 <template>
2   <div>
3     <p>Your name: <input type="text" v-model="visitor.name"/></p>
4     <p>Your message: <input type="text" v-model="visitor.message"/></p>
5     <button v-on:click="sayHi">Say Hi</button>
6   </div>
7   <ul>
8     <li v-for="message in messages">{{message["visitor.name"]}} : {{message["visitor.message"]}}</li>
9   </ul>
10  </div>
11  </div>
12  <script>
13    data: function() {
14      return {
15        visitor_name: '',
16        visitor_message: '',
17        messages: []
18      }
19    },
20    methods: {
21      sayHi: function() {
22        this.messages.push({
23          "visitor.name": this.visitor_name,
24          "visitor.message": this.visitor_message
25        });
26        this.visitor_name = '';
27        this.visitor_message = '';
28      }
29    },
30    computed: {
31      count: function() {
32        return this.messages.length;
33      }
34    }
35  </script>
36  <style>
37    var app = new Vue({
38      el: "#app"
39    })

```

2 files to inspects selected

Helworld

Your name: Your name: Your name:

Your message: Your message: Your message:

• Thej : Welcome • Raj : Hi • Raj : Hi

React Detected Vue 2.6.14

Components Events Routing Performance Storage Accessibility Application

Home components

Tree:

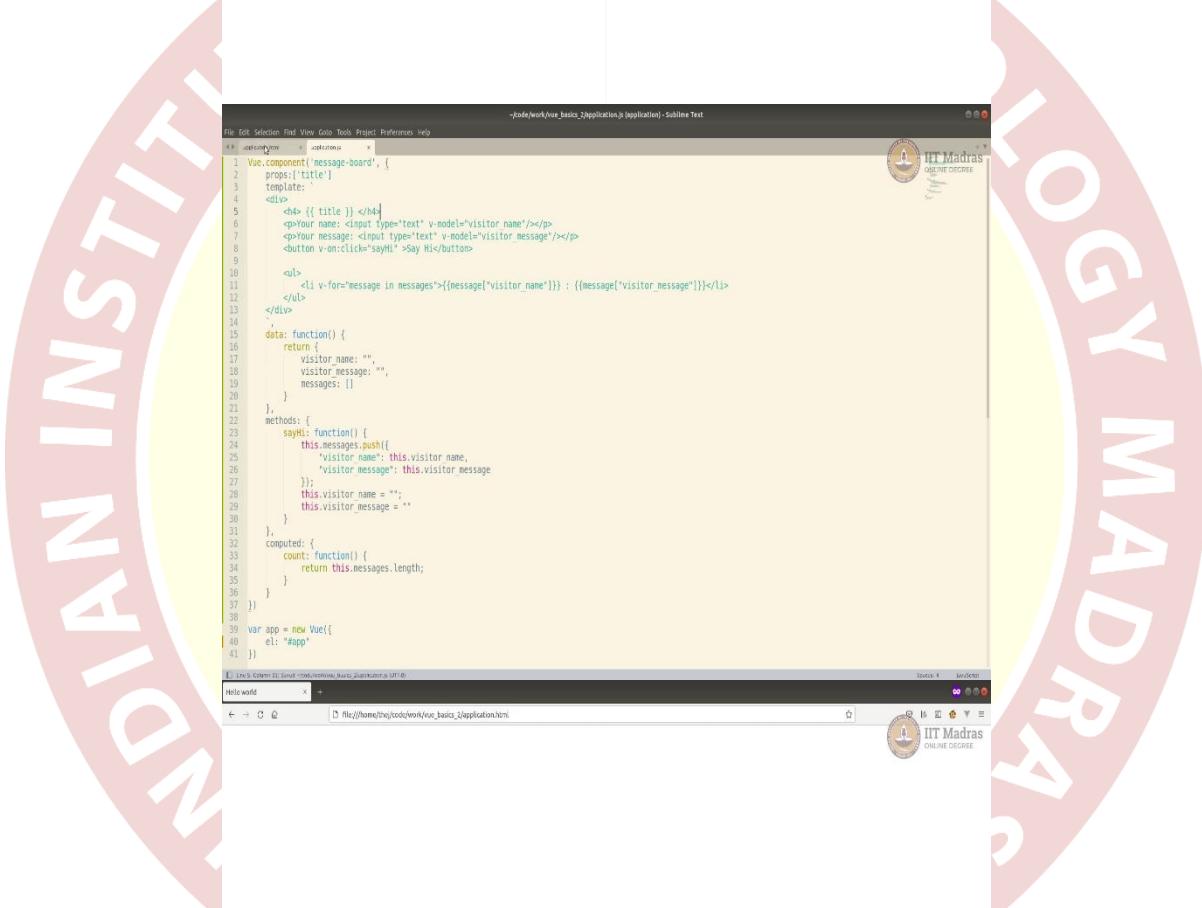
- Messageboard-
- Messageboard-
- Messageboard-

Select a component instance to inspect.



INDIAN INSTITUTE

BIOLOGY MADRAS



Helicord x +

File Edit View Code Tools Print Preferences Help

IT Madras ONLINE DEGREE

Your name: _____ Your name: _____ Your name: _____

Your message: _____ Your message: _____ Your message: _____

Say Hi Say Hi Say Hi

The screenshot shows a web browser window with a form for sending messages. The URL is http://127.0.0.1:5000/hellicord. The page includes fields for 'Your name' and 'Your message', and a 'Say Hi' button.

File Inspector Debugger Network Storage Application

Vue Devtools Vue 2.6.14

Components Events Routing Performance Refresh

Vue Components

Selected component: -MessageBoard- -MessageBoard- -MessageBoard-

Select a component instance to inspect.

The screenshot shows the Vue Devtools interface. It lists three instances of the 'MessageBoard' component. A dropdown menu is open over the first instance, showing options like 'Edit component' and 'Delete component'.

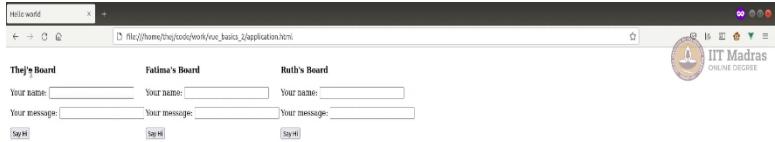
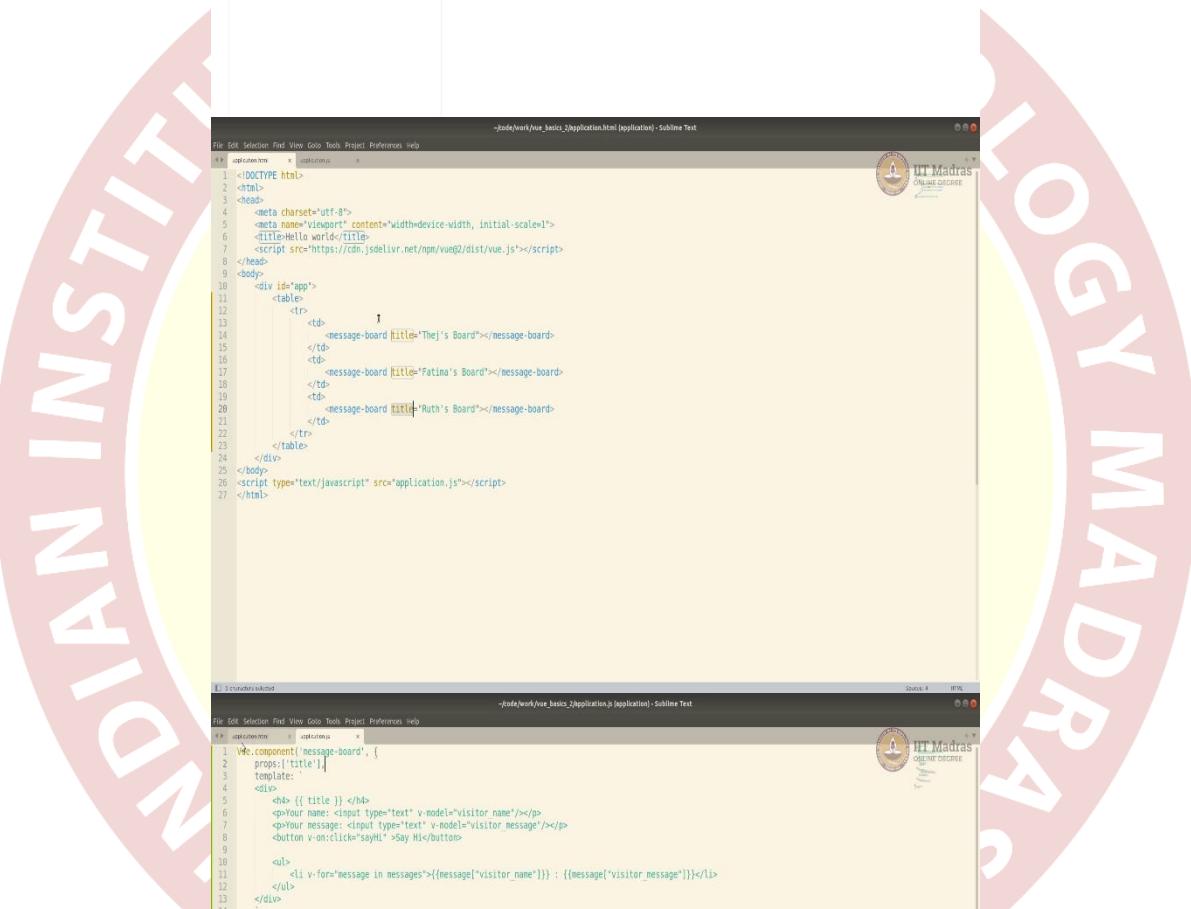
File Edit Selection Find View Code Tools Print Preferences Help

IT Madras ONLINE DEGREE

```
<code>/work/vue_basics_2/Application.js (application) - Sublime Text
```

```
1<script>new Vue({ el: '#app' })</script>
1  Vue.component('message-board', {
2    props:['title']
3    template: `
4      <div>
5        <h2>{{ title }}</h2>
6        <p>Your name:<input type="text" v-model="visitor_name"/></p>
7        <p>Your message:<input type="text" v-model="visitor_message"/></p>
8        <button v-on:click="sayHi">Say Hi</button>
9
10       <ul>
11         <li v-for="message in messages">{{message['visitor_name']}} : {{message['visitor_message']}}</li>
12       </ul>
13     </div>
14   }
15   data: function() {
16     return {
17       visitor_name: '',
18       visitor_message: '',
19       messages: []
20     }
21   },
22   methods: {
23     sayHi: function() {
24       this.messages.push({
25         "visitor_name": this.visitor_name,
26         "visitor_message": this.visitor_message
27       });
28       this.visitor_name = '';
29       this.visitor_message = ''
30     }
31   },
32   computed: {
33     count: function() {
34       return this.messages.length;
35     }
36   }
37 })
38 var app = new Vue({
39   el: "#app"
40 })</pre>
```





```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>Hello world</title>
<script src="https://cdn.jsdelivr.net/npm/vue@dist/vue.js"></script>
</head>
<body>
<div id="app">
<table>
<tr>
<td>
<message-board title="Thej's Board"></message-board>
</td>
<td>
<message-board title="Fatima's Board"></message-board>
</td>
<td>
<message-board title="Ruth's Board"></message-board>
</td>
</tr>
</table>
</div>
<script type="text/javascript" src="application.js"></script>
</body>

```

```
File Edit Selection Find View Goto Tools Project Preferences Help
File Edit Selection Find View Goto Tools Project Preferences Help
<script>
<template>
<div>
<h1>{{ title }}</h1>
<p>Your name: <input type="text" v-model="visitor_name"/></p>
<p>Your message: <input type="text" v-model="visitor_message"/></p>
<button v-on:click="sayHi">Say Hi</button>
<ul>
<li v-for="message in messages">{{message["visitor_name"]}} : {{message["visitor_message"]}}</li>
</ul>
</div>
</template>
<script>
  Vue.component('message-board', {
    props: ['title'],
    template: `
      <div>
        <h1>{{ title }}</h1>
        <p>Your name: <input type="text" v-model="visitor_name"/></p>
        <p>Your message: <input type="text" v-model="visitor_message"/></p>
        <button v-on:click="sayHi">Say Hi</button>
        <ul>
          <li v-for="message in messages">{{message["visitor_name"]}} : {{message["visitor_message"]}}</li>
        </ul>
      </div>
    `,
    data: function() {
      return {
        visitor_name: '',
        visitor_message: '',
        messages: []
      }
    },
    methods: {
      sayHi: function() {
        this.messages.push({
          "visitor_name": this.visitor_name,
          "visitor_message": this.visitor_message
        });
        this.visitor_name = '';
        this.visitor_message = ''
      }
    },
    computed: {
      count: function() {
        return this.messages.length;
      }
    }
  })
  var app = new Vue({
    el: '#app'
  })
</script>

```

Now, the whole purpose of doing this was to have three message borders three independent message boards. Let us add them, I am just going to make it simple table to add them as like columns. You could use divs and you know, rows and columns using the frameworks that we have learned before, using our standard framework, bootstrap, you can create rows and columns and then make them that way, but I am for a quick use, I am just going to make it into a table.

A simple table with one row and column, three columns, each column containing a message board. So, I can see message boards here. So, you can see three message boards. Let us make it little more. You can do any styling here, you can do what all the formatting that you want to add. So, I am just going to do like a P to bring them into next line. You can add your styles and other classes and other things. Let us refresh this. So, I can see that now there behind there is a p.

Let us say Tej, welcome back for this let us say Fatima's board and this is the then say, hi, so it is reloading, that is because it is clearing the things, not reloading actually, clearing the things. Let us say this one Manoj also says, hi, but to this board, and Raj again says, hi, to Fatima's board. But now we want to add like a name for each of the one name of the board. We can do that by just adding for example a name here. We could probably like let us do, add a name here for each of the board. I will show you a better way of doing it but essentially let us start I would say your had Tej's board, Fatema's board, Ruth's board. So, you can see here which board is whose.

But I really do not want to do this. I want it to be part of our definitions of part of our board. I wanted to send a message along with this to actually our component saying this is the board name, that you can do by sending a property or you can do, for example, say title of the board and say Tej's board instead of doing here we are going to send it and make it part of the component.

I will show you how to, how you will use title inside the template. Let us do this first. Fatima. So, I am going to remove that here, remove that here. I am going to refresh right and you can see nothing happens because we are sending it as title here, but we are not using it here. Now to use it, you have to define properties or they are called PROPS. You can for simple ones, you can just define it an arrays an array and we want to remove. We want to receive title, and we want to bring the title here.

Now I am going to do like a h4, or h3, whatever. And then, like our regular property we can use here, and then you can save. Now you can see that you are sending the property here to the message port, which is defined, and we are using that property instead the message board. Okay, there seems to be something wrong. Oh, okay. Syntax error. Okay. So, let us refresh.

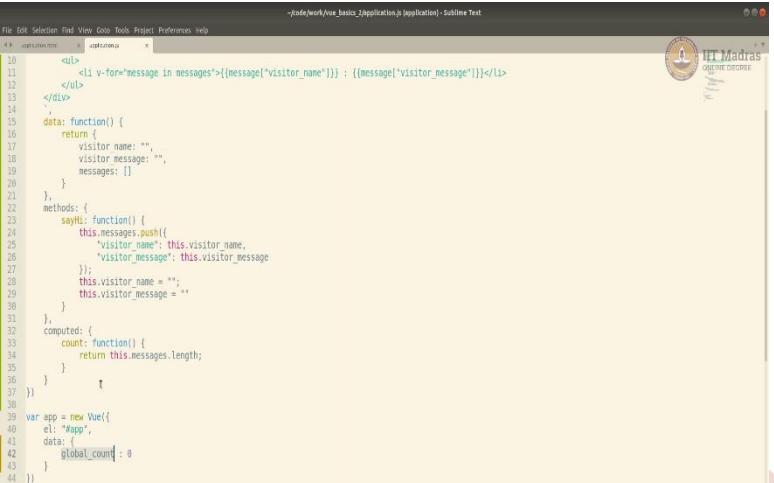
Okay, now we can see, you can see the name of the board, using the property. So, that is how you actually send values from the main app into the component. I mean, we just showed it as a title, but this is a way to send details or values, or some data from the main app into the component. You can send many things. You could probably send some style elements, for example, or it could send other properties like size, for example, how big or how small, you want the whole component to be things like that. And then you can use that here to format it or to make it the way you want, it. Basically, can send all kinds of data from here to here using properties.

You can expand it to, you know, add some default values, and add some property types, like whether it is a string, integer, etc. but here, I am using simple property. You can explore that by yourself.

(Refer Slide Time: 17:04)



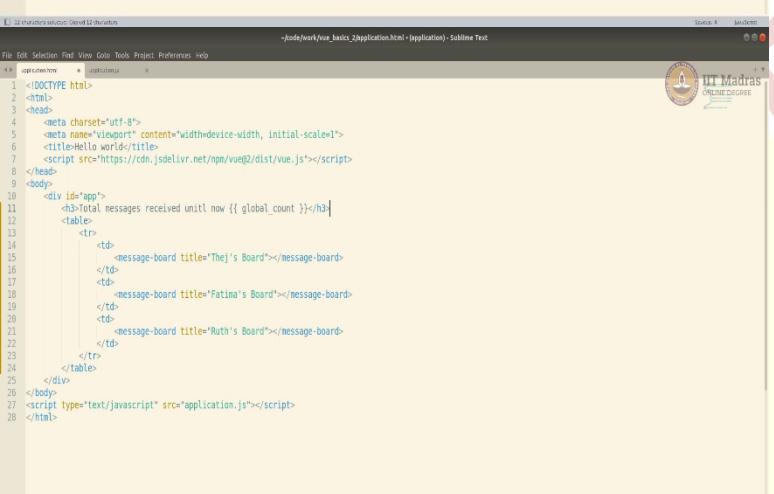
INDIAN INSTITUTE OF TECHNOLOGY MADRAS



```

10     <ul>
11       <li v-for="message in messages">{{message['visitor_name']}} : {{message['visitor_message']}}</li>
12     </ul>
13   </div>
14 }
15 data: function() {
16   return {
17     visitor_name: '',
18     visitor_message: '',
19     messages: []
20   }
21 },
22 methods: {
23   sayHi: function() {
24     this.messages.push({
25       'visitor_name': this.visitor_name,
26       'visitor_message': this.visitor_message
27     });
28     this.visitor_name = '';
29     this.visitor_message = ''
30   }
31 },
32 computed: {
33   count: function() {
34     return this.messages.length;
35   }
36 }
37 }
38
39 var app = new Vue({
40   el: '#app',
41   data: {
42     global_count: 0
43   }
44 })

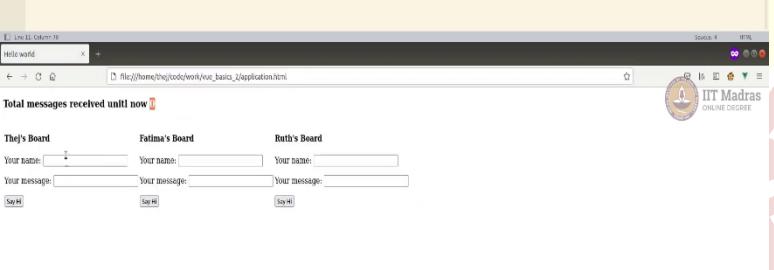
```



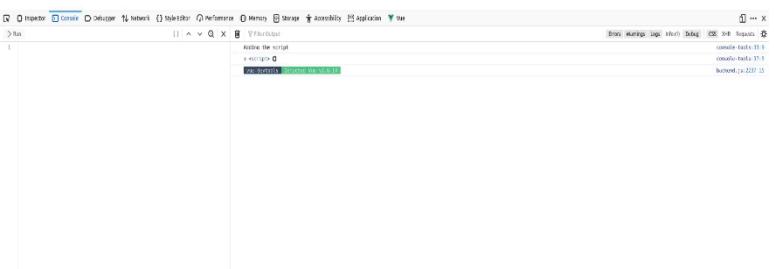
```

1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="utf-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1">
6     <title>Hello world</title>
7     <script src="https://cdn.jsdelivr.net/npm/vue@2/dist/vue.js"></script>
8   </head>
9   <body>
10   <div id="app">
11     <h3>Total messages received until now {{ global_count }}</h3>
12     <table>
13       <tr>
14         <td>
15           <message-board title="Thej's Board"></message-board>
16         </td>
17         <td>
18           <message-board title="Fatima's Board"></message-board>
19         </td>
20         <td>
21           <message-board title="Ruth's Board"></message-board>
22         </td>
23       </tr>
24     </table>
25   </div>
26 </body>
27 <script type="text/javascript" src="application.js"></script>
28 </html>

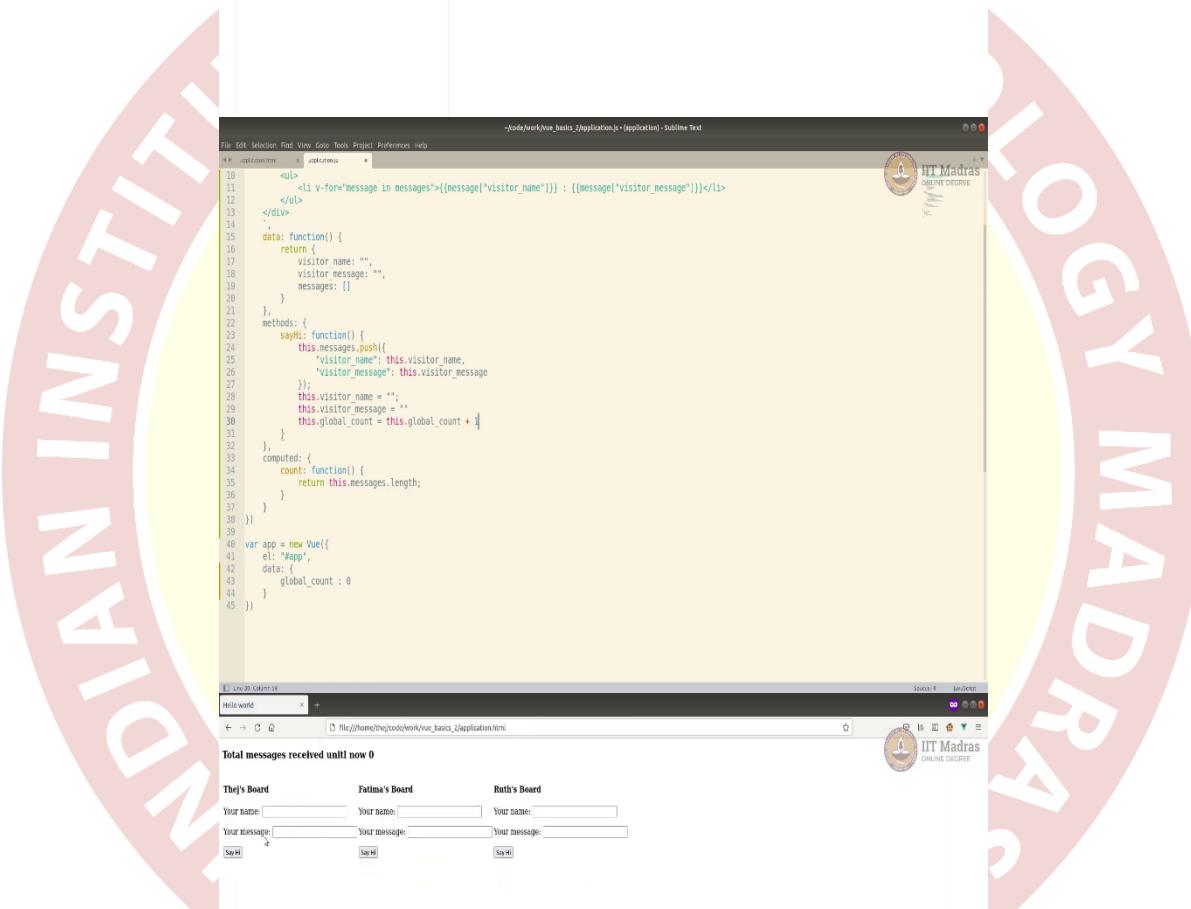
```



The screenshot shows a web browser window with the URL file:///home/bsg/code/woe/vue_basics_2/application.html. The page displays a table with three rows, each representing a message board. The first board is titled "Thej's Board" and contains fields for "Your name:" and "Your message:". The second board is titled "Fatima's Board" and contains fields for "Your name:" and "Your message:". The third board is titled "Ruth's Board" and contains fields for "Your name:" and "Your message:". There is a "Say Hi!" button next to each board.



The screenshot shows the Chrome DevTools Network tab. A single request is listed: "application.js" with a status of "OK" and a size of 13.1 kB. The "Timing" section shows a total time of 13.1 ms, with the "DNS" phase taking 13.1 ms.



HelloWorld

Thej's Board Fatima's Board Ruth's Board

Your name: _____ Your name: _____ Your name: _____

Your message: _____ Your message: _____ Your message: _____

SayHi SayHi SayHi

• Ram : 11



-ycode/work/vue_basic_2/Application.js • (application) • Sublime Text

```
File Edit Selection Find View Code Tools Project Preferences Help
40<script>
41  <ul>
42    <li v-for="message in messages">{{message["visitor_name"]}} : {{message["visitor_message"]}}</li>
43  </ul>
44  ,
45  data: function() {
46    return {
47      visitor_name: '',
48      visitor_message:'',
49      messages: []
50    }
51  },
52  methods: {
53    sayHi: function(){
54      this.messages.push({
55        "visitor_name": this.visitor_name,
56        "visitor_message": this.visitor_message
57      });
58      this.visitor_name = "";
59      this.visitor_message = ""
60      this.global_count = this.global_count + 1
61    }
62  },
63  computed: {
64    count: function() {
65      return this.messages.length;
66    }
67  }
68 }
69 var app = new Vue({
70   el: "#app",
71   data: {
72     global_count : 0
73   }
74 })
```

HelloWorld

Thej's Board Fatima's Board Ruth's Board

Your name: _____ Your name: _____ Your name: _____

Your message: _____ Your message: _____ Your message: _____

SayHi SayHi SayHi

Total messages received until now 0



Helcworld

Thej's Board Fatima's Board Ruth's Board

Your name: Your name: Your name:
 Your message: Your message: Your message:

* Ban :



IT Madras
ONLINE DEGREE



```

File Edit Selection Find View Code Tools Project Preferences Help
File Edit Selection Find View Code Tools Project Preferences Help
 7   <p>Your message: <input type="text" v-model="visitor_message"/></p>
 8   <button v-on:click="sayHi" >Say Hi</button>
 9
10  <ul>
11    <li v-for="message in messages">{{message["visitor_name"]}} : {{message["visitor_message"]}}</li>
12  </ul>
13
14  data: function() {
15    return {
16      visitor_name: '',
17      visitor_message: '',
18      messages: []
19    }
20  },
21  methods: {
22    sayHi: function() {
23      this.messages.push({
24        "visitor_name": this.visitor_name,
25        "visitor_message": this.visitor_message
26      });
27      this.visitor_name = '';
28      this.visitor_message = '';
29      this.global_count = this.global_count + 1;
30    },
31  },
32  computed: {
33    count: function() {
34      return this.messages.length;
35    }
36  }
37 }
38
39 var app = new Vue({
40   el: "#app",
41   data: {
42     global_count: 0
43   }
44 })

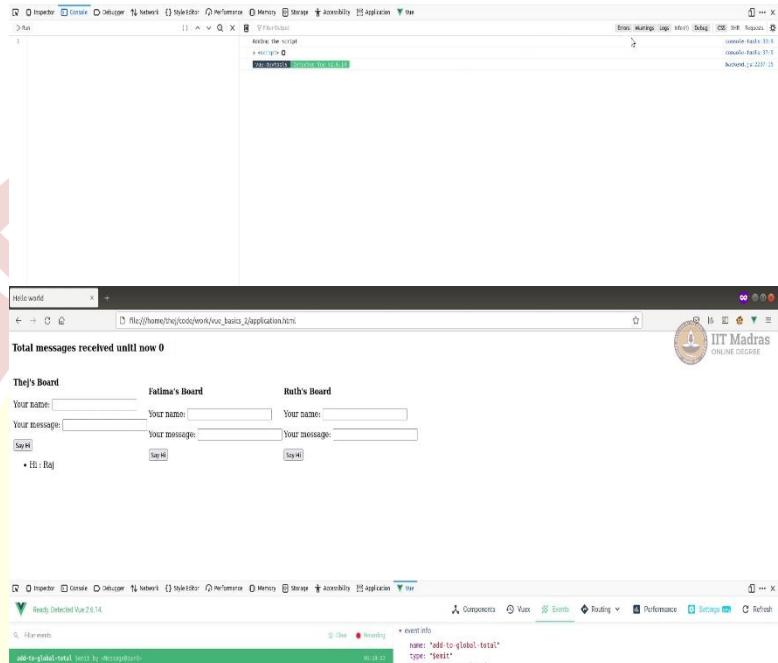
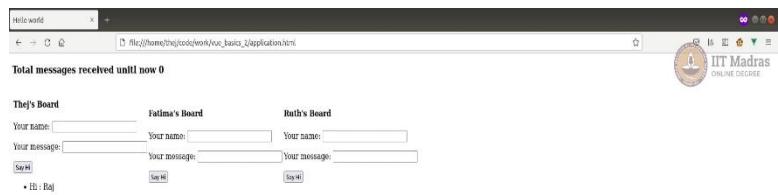
```

```

File Edit Selection Find View Code Tools Project Preferences Help
File Edit Selection Find View Code Tools Project Preferences Help
 10  <ul>
11    <li v-for="message in messages">{{message["visitor_name"]}} : {{message["visitor_message"]}}</li>
12  </ul>
13
14
15  data: function() {
16    return {
17      visitor_name: '',
18      visitor_message: '',
19      messages: []
20    }
21  },
22  methods: {
23    sayHi: function() {
24      this.messages.push({
25        "visitor_name": this.visitor_name,
26        "visitor_message": this.visitor_message
27      });
28      this.visitor_name = '';
29      this.visitor_message = '';
30      this.$emit("add-to-global-total");
31    }
32  },
33  computed: {
34    count: function() {
35      return this.messages.length;
36    }
37  }
38
39 var app = new Vue({
40   el: "#app",
41   data: {
42     global_count: 0
43   }
44 })

```

INDIAN INSTITUTE OF LOGIC MADRAS



Now, I want to do one more thing. Now we have added, like a message board, which works independently. Here is one, here, Raj says, hi. Asif, says, hi. Now, I want to show a global counter that says, how many messages were received in total? Currently, we know one message was here, one message is here, one message here. And if you see in our component we have computed, but it is at that individual level, that individual level, it is not the total overall application's total message received.

There is no way to count do that currently. Well, we could do if we could pass on while, when somebody says, hi, if you could tell that, we have received a message back to app saying you have received a message, and then we could count upon show them the total message, but there is no, currently, there is no way to send that back. We will see how to do that. We can do that by emitting the event.

I will show you how to emit an event from here to do that. You cannot really do for example, if I define data here, like we defined, which is a global app-level data, let us say global count is to 0. And then, I can use that here global or let us say total messages issued until now. I mean, you can do this let us say what happens. Let us say this is, it is 3 just to give some significance. It can be any tag.

And then if I do, did I save it? Yes. I have saved it and let me refresh, so it shows 0. And now if I do Ram, hi, it is not incrementing because we have not incremented the global count. Now to increment the global count like we did before, this dot global count is equal to this dot global count plus 1. If we do that, let us see what happens.

Okay, now, hi, Ram, hi. So now nothing happened. Because if you see, this dot global count does not exist here on this, it exists on the app. So, this nothing happens for this. Basically, there is no way to reach the main app using this. So, to reach that app, what we are going to do is, we are going to emit an event. What we are going to do is, we are going to call instead of this we are going to say, this dot dollar emit. This is how you raise an event. What is the name of the event that you want to raise? We want to raise add to global total.

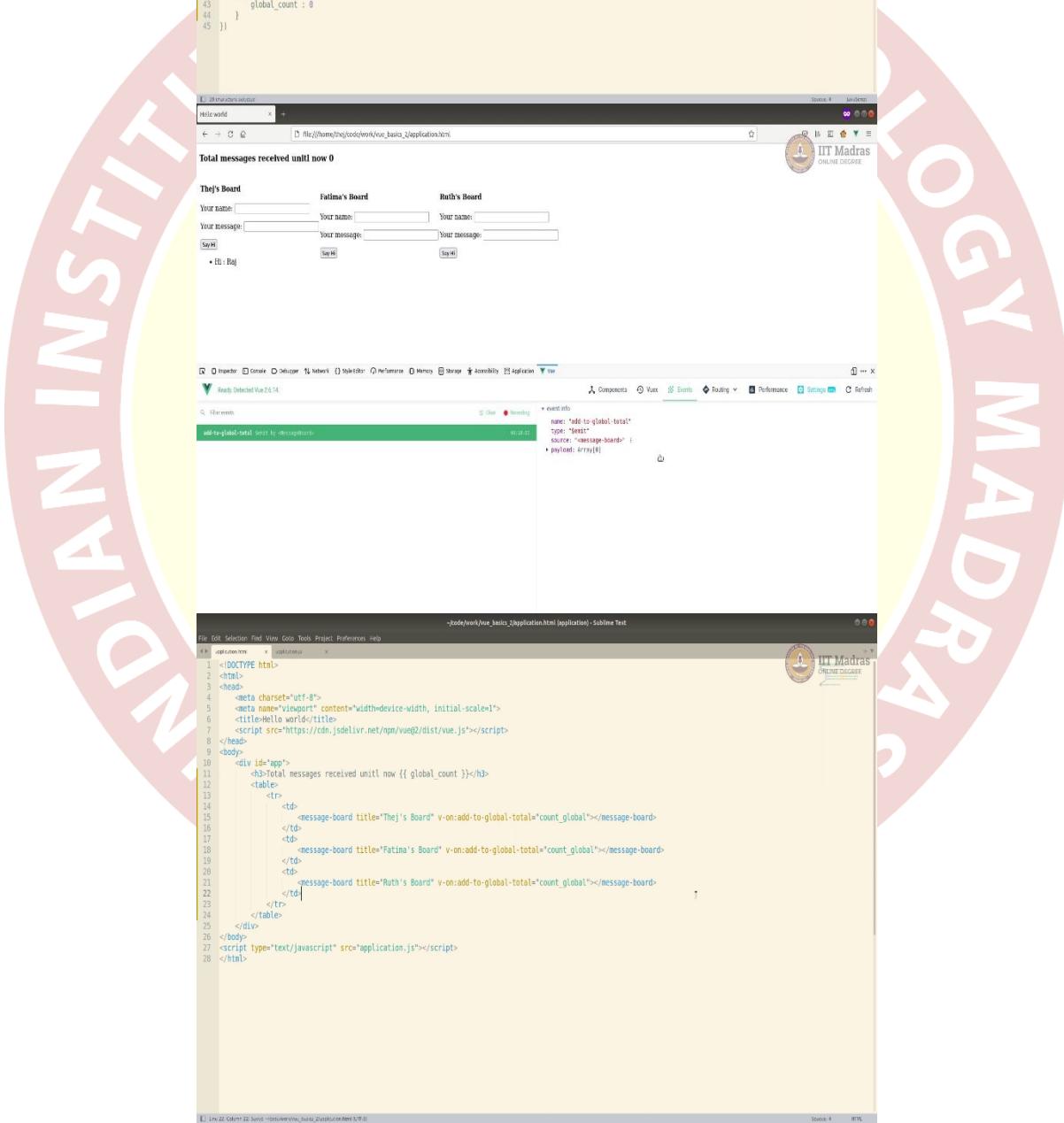
So, this is the event we want to raise. Let us see what happens now when we do this. We are now, hi, Raj. And if I say hi, it did, hi, but if you go to your view, if you go to event, you can see that there is an event. The event name was add to global total, and it got emitted and the message was message, the source of the emit was a message board, this message board or any message board and the payload was array.

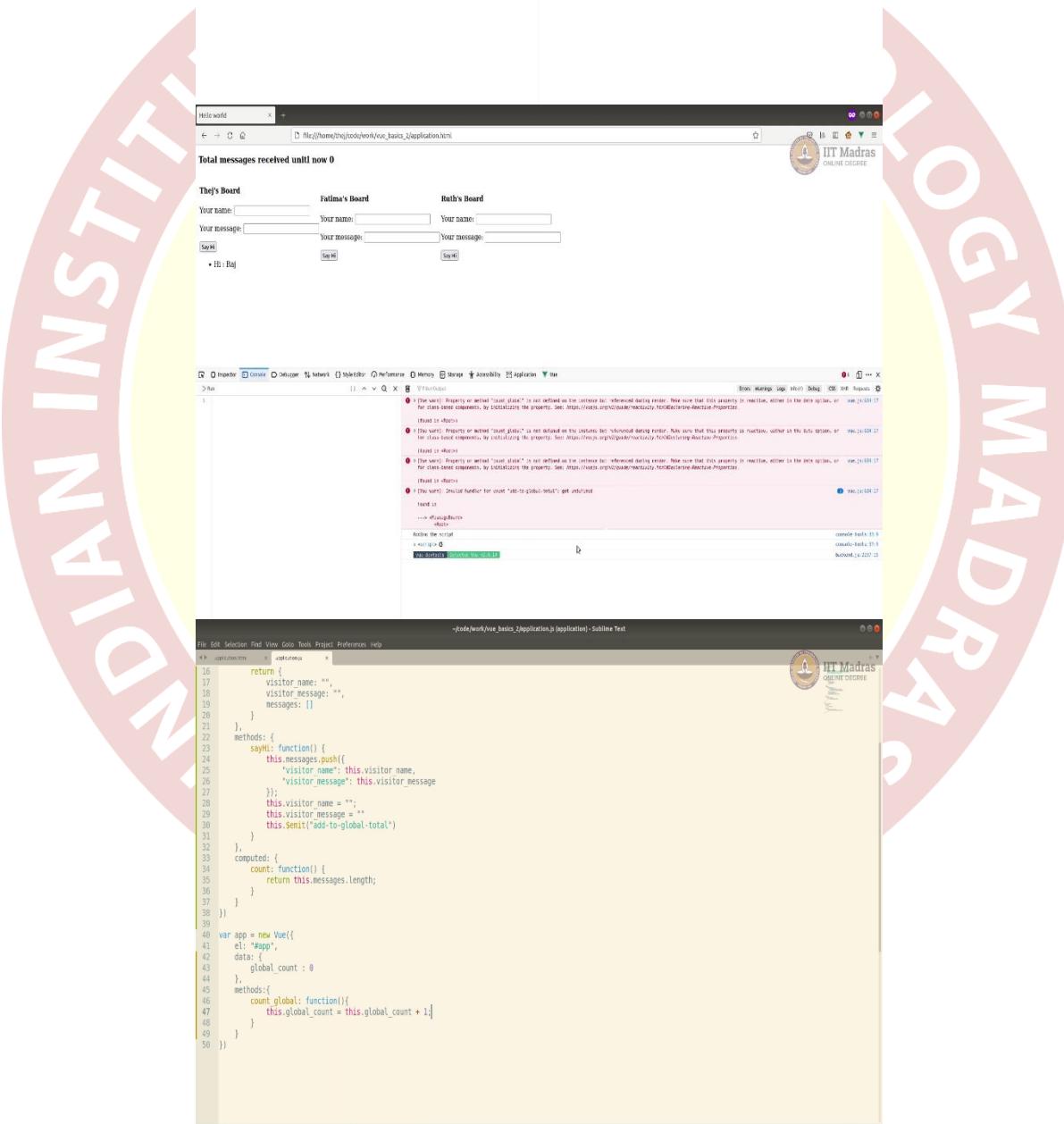
We are not currently worried about the payload, we just know that the event was raised and, but nothing was done. Because like any event, once the event is emitted, you need to handle it.

(Refer Slide Time: 22:06)

The screenshot shows a development environment with three main windows:

- Sublime Text (Top Window):** Displays the `application.js` file content. It contains Vue.js code for a component named `message-board`. The component has a title prop, a template with a `<div>` element containing a `<h2>` for the title and two `<p>` elements for visitor name and message. It includes a `<button v-on:click='sayHi'>Say Hi</button>`. The `data` function initializes `visitor_name`, `visitor_message`, and `messages` arrays. The `methods` section contains a `sayHi` method that pushes a new object into the `messages` array with properties `visitor_name` and `visitor_message`. It also has a `computed` block with a `count` getter that returns the length of the `messages` array.
- Browser Preview (Middle Window):** Shows a simple web page titled "Hello world". It features three input fields labeled "Your name", "Your message", and "Your name" again, followed by three buttons: "Say Hi", "Say Hi", and "Say Hi". Below the buttons are three radio buttons labeled "Tell Message", "Rej : HI", and "Asif : HI".
- Developer Tools (Bottom Window):** Shows the browser's developer tools with the "Console" tab selected. The console output shows three entries:
 - "console.info is 33:8"
 - "console.info is 33:8"
 - "console.log is 2207:15"





File Edit Selection Find View Go Tools Project Preferences Help

```

40 var app = new Vue({
41   el: "#app",
42   data: {
43     global_count : 0
44   },
45   methods:{
46     count_global: function(){
47       this.global_count = this.global_count + 1
48     }
49   }
50 })

```

Line 47 Col 115 Source: /code/work/no_basics_2/application.js (application) - Sublime Text

Stacks 4 last item

Thej's Board Fatima's Board Ruth's Board

Your name: _____ Your name: _____ Your name: _____
 Your message: _____ Your message: _____ Your message: _____

Say Hi **Say Hi** **Say Hi**

Braj : HI



```
File Edit Selection Find View Code Tools Project Preferences Help
File Edit Selection Find View Code Tools Project Preferences Help
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
```

```
File Edit Selection Find View Code Tools Project Preferences Help
File Edit Selection Find View Code Tools Project Preferences Help
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
```

Now, we did learn in the last thing, we could add an event handler by V on, by calling this V on. So, we can do the same thing. We can call we can use V on and handle the event. Just

give me a second, let us try something V on. It is not actually V colon it is V dash on colon. And what we want, what is the event, we usually used to do, click because we always handle click generally. But here, the event name is not click, event name is you know is add to global total. That is the name of the event, you can see here. Name of the event is add to global total.

So that is the same event name we are going to use it here, and then, we are going to call a function count global. This is a function that is going to handle this event. We are going to do the same thing for this too and for this too. Now, let us see what happened if I call this. I am going to refresh this.

Say hi, say Raj. Say hi. Event was raised, but nothing happened. It got raised, but no one handled it. In the console if you see, there is a event raised but nothing, no count global function to handle it. We have not defended function at all. So now we will just define this function. Where do we need to define this function? Because it is getting called emitted on the top of the event it is going to be defined at the app level, so we are going to define it here.

So, we are going to define like before, add an attribute called methods within that we will add a function, the function name will be count global and function. This function will not take any input, it will just increment this. This dot global count is equal to this dot global count plus 1.

Just checking all the brackets are closed. Now, I think it should work. Let us see whether it really works. Hi, not hi, Raj says, Hi see it got incremented here 1 globally. Tej says, Hi, Ruth says, Hi. See, this is how you actually pass the event. You do not have to just pass the event, you can also pass data along with the event. You can check out the details of emit on how to send the data and how to receive the data here and do the data.

For example, if you have a global pool of messages that you want to display on the side, not only individually, but also together, you can do that, I mean, just giving an example. But if I had a cart, you could add multiple products into the same single cart. Each cart could have, each for example, catalog could have different sections, but there could be one total catalog for the whole application. And that is how you send data back. Now, in this case, from a component back to the main app.

That is it for today. What you learnt is, how to create a component, how to handle component templates, how to send properties to the component from the main app to the component and

use that property. Then how to actually send emit the event and use that event to get data and update the main app. That is all in today's screencast, See you next time. Thank you.

