Week 15 codify academy

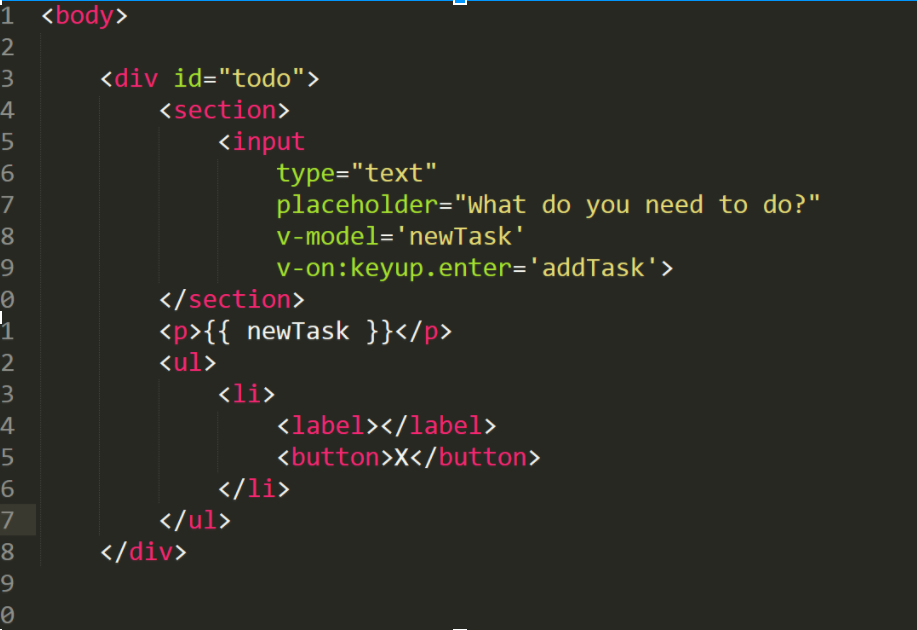
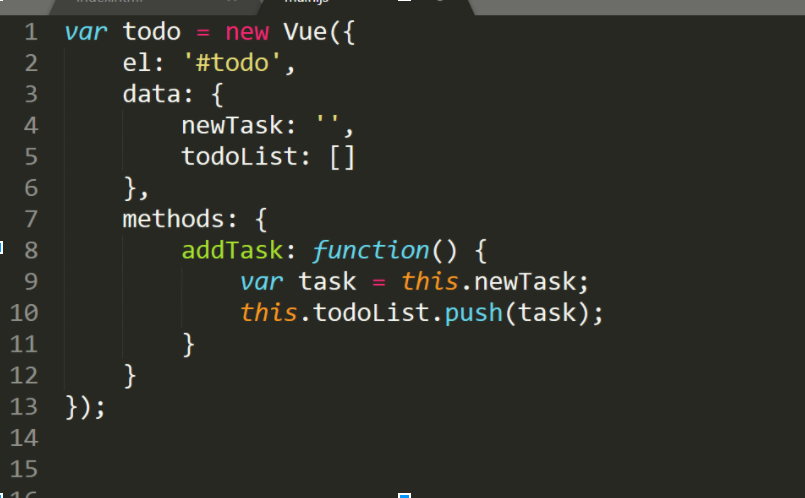
**Week 15**

**Group Discussion Questions**

1. What is Vue.js, and what can it do for us?
2. What does MVC stand for?
3. What are directives, and what do they do?
4. Where can you go to learn more about Vue.js and read its documentation?
5. What is interpolation? What else is it called? What is it used for?
6. When coding a method inside the Vue instance, what does the "this" variable refer to?
7. What does v-for do? What's the syntax for using it?

**Exercises**

***Important: Make sure to add comments to your code to help your future self.***

1. Create a copy of your template folder rename it to vue-template.
2. First thing we need to do is link VueJS to our HTML in a script tag using the Vue CDN. Place it at the bottom of your body like any other script: [https://cdn.jsdelivr.net/npm/vue](https://cdn.jsdelivr.net/npm/vue" \t "_blank).
3. We’re going to need to create the HTML element in which the code inside our Vue instance will run. In your HTML, create a div and give it an id of “todo”.
4. Inside of our todo div, let’s create a section to contain our to-do input. Create a section, create one text input. Give this input the placeholder text, “What do you need to do?”
5. Now we’ll need to create the unordered list which will contain all of our to-dos once we have entered them. Below the section, create a ul, and inside this ul, create one (only one) list-item. Think of this list-item as the “template” list-item for all of our future to-dos.
6. Inside of each to-do li, we want to have a label element which contains the to-do, and a button element for deleting the to-do. Add an empty label, and a button with the text-content of “X”, inside the empty list-item.
7. Now that our HTML is set up, let’s set up our Vue instance. In Javascript: store a new Vue instance in a variable called “todo” with the following code:   
     
    var todo = new Vue({ });
8. Now specify the element in which we want our Vue instance code to run with an “el” property, and create a “data” property and a “methods” property, each containing an object, below it:   
     
    var todo = new Vue({ el: '#todo', data: { }, methods: { } });
9. Now we need to create a few data properties which will update with user interaction; “newTask”, which will update as a user creates a new todo, and “todoList”, which will be an empty array in which we’ll store todo’s as they are created:   
     
    var todo = new Vue({ el: '#todo', data: { newTask: '', todoList: [] }, methods: { } });
10. In HTML: inside your text input inside your section, let’s bind the value of this input to the “newTask” data key (think of these data keys as plain javascript variables). Inside the input tag, add “ v-model=’newTask’ “. v-model is a Vue-specific directive which binds the value of this input to a data key. This means that, as a user types into this input tag, the newTask key will update reactively (in real-time).  
    (If you want to see this in action at this point, add a paragraph somewhere in your HTML INSIDE the todo div, but outside your section and li, and give it the text content of: {{ newTask }}. Now load your page, type into the text input, and watch the content of that paragraph element change as you type.)
11. In HTML: then, we'll need to write the trigger for the function which will actually create a todo item and store it in our todoList key. To do this, we'll reference a function called "addTask" (which we'll create in the next step), and cause it to fire when a user presses the enter key inside our text input. To do this, we'll use another Vue directive, v-on, which works just like an "onclick" or "onscroll" or "onmouseover" attribute/event listener. So, inside our text input, let's also write " v-on:keyup.enter='addTask' ".  
    By now, our HTML should look something like this:
12. In JS: now to define the addTask function, we'll add it as a method property, like this:   
      
     var todo = new Vue({ el: '#todo', data: { newTask: '', todoList: [] }, methods: { addTask: function() { } } });
13. Inside our new function, we'll define a variable, "task", equal to "this.newTask". Inside the Vue instance, "this" will almost always refer to our "data" object, so "this.newTask" refers to our newTask data key. Think about it. When this function is running, a user should have typed a new todo into our text input and hit the enter key, so the value of that newTask key is the new todo which the user typed in, and now we have stored that data in the "task" variable inside our "addTask" function.
14. After declaring our task variable, we're going to push task to the todoList array (don't forget, in order to target the todoList key inside the addTask function, we need to use "this.todoList"). Now our function should be good to go!
15. Now that our function is ready, we need to set up our list so that it updates with new todos as we add them. Inside the opening list-item tag, insert the following code: " v-for='task in todoList' ". This is a Vue directive which uses for-loop logic. What this will do is it will replicate this list-item element for every item inside our todoList key (we have also "named" each item in this array as a "task" in the same line of code). In other words, for each so-called "task" in the todoList array, there will be a list-item like this one.
16. Our list-item is now connected to our todoList data key. Inside the empty label element inside the list-item, insert "{{ task }}" (remember: without the quotations). This will cause each label in each list-item created with the v-for directive to contain the value of the so-called "task" in todoList specific to its parent list-item. To see it in action, refresh your page and try adding a few todos.
17. Almost done! Now we want to enable the button inside our list-item to delete a todo. We’re going to add a v-on directive to this button which listens for a click event and runs a function, “removeTask” which will receive an argument of “task” (specifically, this will be the task unique to the list-item containing the given button): “ <button type="button" v-on:click="removeTask(task)">X</button> “
18. In JS: then we define the removeTask method in our Vue instance. In this method, we’ll declare an index variable equal to the index of our argument, “task”, in our data’s todoList key: “var index = this.todoList.indexOf(task)”. Then we splice the todo we want to delete it from our todoList with: “ this.todoList.splice(index, 1) “   
      
     var todo = new Vue({ el: '#todo', data: { newTask: '', todoList: [] }, methods: { addTask: function() { var task = this.newTask; this.todoList.push(task) this.newTask = ''; }, removeTask: function(task) { var index = this.todoList.indexOf(task); this.todoList.splice(index, 1); } } });
19. With that, our VueJS to-do list is finished! Now think about how you can improve it.
    * Try adding a clear all button, which clears the list of all todos.
    * Try adding a checkbox to each to-do which, when checked, causes the to-do to become crossed out.
    * Try making a to-do which has already been added editable when clicked on.

***Once you have completed these exercises add them to Github.***

[Download Completed Week 15 Exercises](http://codifyacademy.com/wk15/week15exercises.zip)