FarmData2 Onboarding Activity 05
Vue Events and JavaScript Functions Technology Spike

Introduction:

In the last activity you learned about *Vue Data Binding* and how Vue + JavaScript can be used to build pages with content that is rendered from the *Vue instance* (a JavaScript object). You saw that when changes are made to the Vue instance through the DevTools console or the Vue DevTools, the rendered page also changes. In this set of activities, you will build upon that by having JavaScript code modify the Vue instance in response to events (e.g. button clicks). You'll also see another way to do data binding, a few new Vue directives, and you'll learn some more JavaScript. Then in the next two activities you'll learn how to use JavaScript to get data from web services using APIs and then how to bring live data from the FarmData2 database into your harvest report.

Getting Started:

1. Synchronize the main branch of your local and origin FarmData2 repositories with the upstream and merge any changes to main into your feature branch (refer to past Activities if necessary). Give the sequence of commands that you used.

Adding Another Sub-Tab:

2. Make sure you have your feature branch checked out. Add another new sub-tab named A04 to the FD2 Example tab. Have the contents of this new tab be provided by the file `a04.html`. Make a copy of your `a03.html` file into the `a04.html` file. Don't forget to clear the Drupal cache when you are done. The result should be that you now have an A04 sub-tab that is (for now) identical to your A03 sub-tab. You'll be working on the A04 tab throughout this activity.

3. Commit your changes to your feature branch with a meaningful commit message and push it to your origin. Recall that this also updates your Draft Pull Request.

Vue School:

We'll be continuing with the free *Vue.js Fundamentals* course (https://vueschool.io/courses/vuejs-fundamentals) from Vue School (https://vueschool.io/). It is not required, but if you would also like a textual source that covers much of the same material you might find the *Introduction to the Vue.js Guide* (https://vuejs.org/v2/guide/index.html) helpful.

Handling User Events:

Find the *User Events (4:04)* video in the free *Vue.js Fundamentals* course. Watch that video and complete the activities below by continuing your `vuespike.html` file from A03.

4. What Vue directive do you add to a button in order to respond to a click? What is the shorthand for this directive?

5. Add HTML to your `vuespike.html` that adds a new label, field and button as shown below:

Give the lines of HTML that you added to produce this label, field and button.

6. Now, modify your page so that when the button is clicked whatever is in the text field is added to the list of names. As shown in the video you will need to add a property to your Vue instance, bind it to the text field and add an event handler to the button.

a. Give the updated HTML for your text field.

b. Give the code that defines and initializes the new property that you added to your Vue instance.

c. Give the updated HTML for your button.

7. Modify your text field it so that the name is also added if you press Enter or Return. Give the updated HTML for your text field.

8. Hopefully that all works now. But from a User Interface/User Experience (UI/UX) perspective it has some issues. We'll see them here and then fix them shortly.

a. What happens if you click the "Add Name" button or press Enter/Return multiple times?

b. What happens if the field is empty when you press Enter/Return or click Add name?

9. Let's also gather a little more information about the behavior of your text field, Vue instance property and button that will help when improving the UI/UX later. You'll want to use the Vue DevTools for these.

a. What is the initial value stored in the Vue instance property that is bound to your text field?

b. What value is in that property in the Vue instance when you type something into the text field and then delete it?

c. If your answer to parts a and b are not the same, modify the value assigned to the property in your Vue instance so that they are. This will simplify things later. Give code from your Vue instance that creates and initializes the property bound to your text field.

Adding to the Harvest Report Spike 1:

10. In your `a04.html` page in FarmData2:

- Set property of your Vue instance that holds the harvest logs to be an empty array.
- Then have the "Generate Report" button add one harvest log to the Vue instance when it is clicked.

When this works the table in the harvest report should initially be empty (just the headers). Then when the "Generate Report" button is clicked the new row should appear in the table. If you click the button multiple times, then multiple copies of the row should appear. Not quite the real-deal yet, but we are getting closer. Eventually, the JavaScript that runs when this button is clicked will retrieve data from the FarmData2 database using a web API and load it into the Vue instance.

11. Commit your changes to your feature branch with a meaningful commit message and push it to your origin.

Vue Methods and JavaScript Functions:

Find the *Vue Methods (3:00)* video in the free *Vue.js Fundamentals* course. Watch that video and complete the activities below by continuing to modify your `vuespike.html` file.

12. Mimicking what is done in the video, move the code that runs when your "Add Name" button is clicked from the button tag to a function in the methods object in your Vue instance.

a. Give the full methods object that you added to your Vue instance here including the function that handles the button click.

b. Give the updated HTML for your button tag here.

c. Give the updated HTML for your text field here.

13. Add code to your function from #12 so that the text field is cleared after the name is added to the list. Hint: Don't change the text field, just modify the Vue instance and let the data binding do the work! Give the line you added to your function.

14. Extend your function from #13 so that a name is only added to the list if there is something in the text field (i.e. no more adding blanks). To complete this you'll need to use a JavaScript conditional (`if`) statement. Using what you know about other programming languages a quick skim of the MDN resource below should be enough to get you started with JavaScript conditionals:

- https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Building_blocks/conditionals

Give the conditional statement that you added to your function here. Hint: Your answer to question #9 will also be helpful here.

15. Optional Extra Practice: Add a text field and a button for adding a new card to the list. The input to the text field should contain information for a new card (e.g. A H or 3 D, etc.). Your JavaScript function should split the text and add a new object to the array of cards in your Vue instance. You might find the MDN page on Useful String Methods helpful:

- https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/Useful_string_methods

Adding to the Harvest Report Spike 2:

16. In your A04.html file in FarmData2, move the JavaScript code that handles the "Generate Report" button click to a function in the methods object in your Vue instance. Then modify the handler function so that it adds a different harvest record to the Vue instance (and thus to the table as well) each time the button is clicked. Your code should add at least three different records. Hint: Use a JavaScript conditional (if/else) and the length of the array to add a different record each time. A quick scan of the MDN page on arrays should reveal how to determine the length of an array:

- https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/Arrays

Once your function has added all of the records, additional clicks of the button should not do anything.

17. Commit your changes to your feature branch with a meaningful commit message and push it to your origin.

v-for Indices and Functions with Parameters:

Sometimes when using the `v-for` directive to render a list of items it will be useful to have a variable telling us the index of each element as we are using it. The v-for directive has another form that will provide this index. This section will introduce you to this form of the `v-for` directive and show you a handy way to use it.

18. Modify the `li` tag that generates the list of names in your `vuespike.html` page so that it is similar to the one shown below. You may need to adapt it depending upon what you called the array of names in your Vue instance.

```
<li v-for='(name,index) in names'>{{ name + ' (' + index + ')' }}
```

This modified format for the `v-for` directive defines two variables, `name` and `index`, as it iterates over the array. The `name` variable takes on each value in the `names` array, just as it did before. The `index` variable is a counter that starts at 0 for the first name and increasing by one as the `v-for` goes through each additional name.

19. Think about what you would expect the list to look like when you reload the page based on the change you made in question #18. Then reload the page.

a. Paste a screen shot of just the list of names below.

b. Was the output what you expected? Briefly explain why it was or was not what you expected.

We can use the `index` variable created not just in the double mustache but anywhere within the tag where it is defined (e.g. between `<li>` and `</li>` in this case). That can be very useful for creating additional HTML elements that know the index that they are associated with. For example, the `v-on:click` (i.e. `@click`) event handler of the following `button` uses the `index` as an argument to the `splice` array function:

```
<button type='button' @click='names.splice(index,1)'>Delete</button>
```

20. Use the MDN documentation for the JavaScript `Array.splice` method to answer this question.

- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/splice

Based on the documentation, what do you expect to happen when you click the button beside a name? Hint: Pay particular attention to the `start` and `deleteCount` parameters.

21. Add the button tag above between the `<li>` and `</li>` tags that generate the list of names in your `vuespike.html` page. Again, you may have to adapt the statement to match the name of the array that holds the names in your Vue instance. Does clicking the button do what you expected it to do? If not, revisit and revise question #20.

As you see with the `splice` method, like functions in other languages you have used, Java script functions can have parameters. The above call to `splice` passes the value of the `index` variable and the value `1` as values for the `splice` function's parameters. The functions that we write ourselves can also have parameters. For example, consider the following function that might appear in the `methods` object of the Vue instance:

```
deleteName: function(nameIndex) {
  this.names.splice(nameIndex, 1);
}
```

This function accepts one argument as the value for the `nameIndex` parameter. The value of that parameter is then used in the call to `splice`.

22. Add the `deleteName` function above to the `methods` object of the Vue instance in your `vuespike.html` page.

23. Modify the button tag from question #21 so that it now calls your `deleteName` function instead of using `splice` directly. Paste the code for your `button` tag here.

Adding to the Harvest Report Spike 3:

When working with reports (e.g. like the Harvest Report) it is convenient for the farmer to be able to delete and edit records. So each row of a report in FarmData2 will have a "Delete" and an "Edit" button something like shown below:

Date	Field	Crop	Yield	Unit	Hours	Comments	User	Edit	Delete
2018-07-02	GHANA-3	COLLARDS	15.00	BUNCH	0.50		learyc	Edit	Delete
2018-07-02	GHANA-2	KALE	20.00	BUNCH	0.33		learyc	Edit	Delete

24. Add a column to the harvest report table in your `a04.html` file in FarmData2. That column should contain a delete button. You do not need to add the edit button or color the button red at this point. When the delete button in a row is clicked, that row should be removed from the table. Hint: Just remove the row from the Vue instance and let the data binding to the rest!

25. Commit your changes to your feature branch with a meaningful commit message and push it to your origin.

Conditional Rendering:

Find the *Conditional Rendering (3:13)* video in the free *Vue.js Fundamentals* course. Watch that video and complete the activities below by continuing your `vuespike.html`.

26. Set the value of the array you use to hold the list of names in your Vue instance in your `vuespike.html` file to be an empty array (E.g. something like: `names: [ ],`). Now what appears in the names list when you reload the page?

27. Instead of having the list not appear at all (as it should have in #26), it would be better to display a message that there are no names yet. Add an li to the list using a `v-if` directive so that that message "No names yet!" appears in the list if no names have been added. This message should only appear when there are no names in the list. Confirm that it is hidden when you add a name and that it reappears if you delete all of the names. Give the HTML for your `li` element using `v-if` that displays the message.

28. Mimicking what was done in the video, modify your `vuespike.html` so that there are two buttons "Show Cards" and "Hide Cards". By default the page should not show the list of cards but should show the "Show Cards" button. When show cards is clicked the list of cards and the "Hide Cards" button should be displayed and the "Show Cards" button should be

hidden. Hint: Add a state variable to your Vue instance and use it in the `v-if`, `v-else` and `@click` directives. Hint2: Enclose your list in a `div` that has a `v-if` directive to make it easier to show and hide the whole list! Give the HTML for your "Show Cards" and "Hide Cards" buttons.

29. Optional Extra Practice: Add a "Clear Names" button that appears when there are names in the list and is hidden when the names list is empty. When clicked the list of names should be made empty.

Adding to the Harvest Report Spike 4:

30. Modify your `a04.html` file in FarmData2 so that the table headings for the Harvest Report only appear when there are harvest logs to be displayed. If there are no harvest logs to be displayed, then a message should appear indicating that there are no matching records.

31. Commit your changes to your feature branch with a meaningful commit message and push it to your origin.

32. Modify your `a04.html` file in FarmData2 so that the whole section of the page showing the Harvest Report only appears after the Generate Report button is clicked. The report section of the page should remain visible even if all of the rows of the table are deleted (i.e. once shown, the report will remain visible.) Hint: Enclose your entire report section in a `div` and add a new variable to your Vue instance.

33. Commit your changes to your feature branch with a meaningful commit message and push it to your origin.

34. Optional Extra Practice: Modify your `a04.html` file in FarmData2 so that the first click of the Generate Report button just display the report but not add any harvest logs (i.e. the report should be empty when first displayed). Each additional click should then add a harvest log as before.

35. Commit your changes to your feature branch with a meaningful commit message and push it to your origin.

Attribute Bindings:

Find the *Attribute Bindings (1:41)* in the free *Vue.js Fundamentals* course. Watch that video and complete the activities below by continuing your `vuespike.html`.

36. What Vue directive is used to bind an attribute of an HTML element to a data value in the Vue instance? What is the shorthand for this directive?

37. Modify the Add Name button so that it is disabled until something is typed in the text field. Give the HTML for your updated button tag here.

38. Optional Extra Practice: The Clear Names button you added earlier (if you did that option part) appearing and disappearing when the list changes is a little distracting and not the best UI/UX design. Instead, modify the Clear Names button so that it is disabled when the list is empty and enabled when there are names in the list.

Adding to the Harvest Report Spike 5:

You may have noticed that when working with the start and end dates for the Harvest Report it is possible to set a start date that comes after the end date or vice versa. This of course would result in an empty report. So it would be better UI/UX design if it were not possible to choose a start date that comes after the end date or an end date that comes after the start date. Because the `v-bind` (i.e. `:`) Vue directive can be used with any HTML element attribute, it can help us with this.

39. Modify your `a04.html` page so that the start date cannot be set to after the end date and that the end date cannot be set to be set to before the start date.

40. Commit your changes to your feature branch with a meaningful commit message and push it to your origin.

Computed Properties:

Find the *Computed Properties (3:08)* in the free *Vue.js Fundamentals* course. Watch that video and complete the activities below by continuing your `vuespike.html`.

41. When does the video say that you should use a computed property?

42. When does the video say that you should use a method?

43. What does the video say that a computed property should not do?

44. In this question you will modify your `vuespike.html` page to add a computed property that displays the total length of all of the names in the list. For example, the list might now appear as:

Names:

- Sue
- Ahmad
- Jooik
- Anh

Total Name Length: 16

a. Add the following computed object to your Vue instance:

```
computed: {
   totalNameLength() {
     return 0;
   },
 },
```

b. Add HTML to your page that displays the total length of the names using the computed property. Note: It should just display 0 at this point. Give the HTML you added here.

c. Modify the `totalNameLength` function so that it computes the total length of all of the names in the list. You'll need to use a for loop to iterate over the names in the array and add up their lengths into a total. Using what you know about for loops already, a quick scan of the MDN *Loops and Iteration* section should give you what you need to know about for loops in JavaScript:

- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Loops_and_iteration

Similarly, with quick a scan of the MDN *Useful String Methods* section you should be able to determine how to find the length of a string in JavaScript:

- https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/Useful_string_methods

Paste your `totalNameLength` function here.

Adding to the Harvest Report Spike 6:

When viewing reports in FarmData2 the farmers would like to be able to see some summary information at the bottom of the table. It might eventually look something like the image below. In that image, the Totals, Averages and Hours at the bottom are computed from the values in the table. For example, if you add up the Yield values in the four rows you get 43.83.

Date	Field	Crop	Yield	Unit	Hours	Comments	User	Edit	Delete
2018-07-02	GHANA-2	KALE	20.00	BUNCH	0.33		learyc	Edit	Delete
2018-07-09	S	KALE	10.83	BUNCH	0.02		learyc	Edit	Delete
2018-07-09	S	KALE	10.00	BUNCH	0.02		learyc	Edit	Delete
2018-07-11	S	KALE	3.00	BUNCH	0.02	3 bunches of 8 leaves	horowitk	Edit	Delete

Total Yield	Average Yield (bed feet)	Average Yield (row feet)	Average Yield (acres)	Hours	Hours/Unit
43.83 BUNCH(S)	0.193 BUNCH(S)/Bed Foot	0.109 BUNCH(S)/Row Foot	1356.890 BUNCH(S)/Acre	0.38	0.01

45. Modify your `a04.html` page to display the total yield from your Harvest Report.

Note that your table has the yield and the text for the units combined (e.g. 10 Bunches) so you will need to extract the integer part of each of the values to compute the total. You should be able to adapt the expression below to perform this task. The split function as called below divides the string `log.yield` into an array of strings delimited by spaces. The `[0]` gets the first element of that array, which will be the number. The `parseInt` function converts the string holding the number into an integer value that can be used in arithmetic.

```
let yield = parseInt(log.yield.split(" ")[0])
```

46. Commit your changes to your feature branch with a meaningful commit message and push it to your origin.