08 - FarmData2 Components Spike

COMP290 – Large Scale and Open Source Software Development
Dickinson College

Name:

Introduction:

By this point you have a pretty good handle on how all of the front-end development technologies in FarmData2 (HTML, JavaScript, Vue.js, Axios, and Cypress) fit together. However, you've probably noticed that the Harvest Report page that you have created doesn't look much like the other FarmData2 pages (e.g. the Seeding Report). This is because, the actual FarmData2 pages use a collection of *Vue Components* for their UI elements. FarmData2 uses custom Vue Components for UI elements including dropdown menus, date selection, date range selection, formatted input (e.g. only integers), a message banner, and a data table. Using these components across all of the pages helps create a consistent look and feel across the pages. Also, by encapsulating the UI elements within Vue components the look, feel and behavior of the UI elements can be updated without touching the code for the pages.

In this activity you'll learn about some of these custom Vue Components and replace some of the UI elements in your Harvest report with them. You'll learn to use the documentation for these components as well as some of the example code that is built into FarmData2. Finally, you will write a Cypress test or two that interact with the custom Vue Components that you added to your Harvest Report page.

Exploring a FarmData2 Vue Component:

One Vue component used by FarmData2 is the `DropdownWithAllComponent`. This component is just like the HTML dropdown menu you have been using but, it can automatically insert an "All" option at the top of the list.

1. The FD2 Example tab contains a number of subtabs that illustrate techniques and common coding patterns that are used in FarmData2. The UI subtab includes examples of how the custom Vue Components are used. Find the section of the UI subtab that illustrates the `DropdownWithAllComponent`. Experiment with the component and the buttons near it to get a feel for what it does.

2. Based on what you know about the organization of the files that create tabs and sub-tabs in FarmData2, find the file that contains the content (i.e. a `.html` file) for the UI sub-tab on the FD2 Example tab. What is the relative path to this file from FarmData2 directory?

3. Look at the code in the `.html` page you found above. Find the `<dropdown-with-all>` HTML element that creates the `DropdownWithAllComponent` in the page and copy the HTML for that element here.

4. You may be able to intuit some of how the `DropdownWithAllComponent` works by looking at its HTML element.

HTML elements for Vue components have *attributes* and *event handlers* just like the other HTML elements you have been using.

- The attributes for a Vue component are called *props* (short for *properties*) and these are used to provide input to the component that customizes its content. Notice that props can be bound to Vue `data` properties using `v-bind` or its shorthand (`:`), just like normal HTML attributes.
- Like standard HTML element, Vue components emit *events* to signal changes and to provide values to code in the page that contains them. The Vue `v-on` attribute, or its shorthand (`@`) can be used to assign an *event handler* function that is invoked when a specific event occurs.

Use the HTML element you found to answer the following questions.

a. Complete the table below to indicate the four props that the `<dropdown-with-all>` element has, and the Vue `data` properties to which they are bound (if any) in this specific page.

Attribute	Data Property

b. Complete the table below to indicate the event that is emitted by the `<dropdown-with-all>` element and the Vue method that handles that event in this specific page.

Event	Event Handler

5. The implementation of the event handler you just identified is a function that is defined in the `methods` property of the Vue `instance`.

a. Find that function and copy it here.

b. Notice that this function has a parameter. When the `DropDownWithAllComponent` emits the `selection-changed` event it also provides a value, called the event *payload*. The payload provides the value for the parameter. Based on the name of the event and the parameter what do you think the value of this parameter represents?

c. Describe in a sentence what the code in the body of the event handler does.

6. While you can intuit some information from the HTML tag for a Vue component more complete information about this component, and the others that FarmData2 uses, can be found in the FarmData2 documentation.

Open the documentation in `farmdata2/doc/index.html` in the Firefox browser. You probably generated this documentation in a prior activity, but if not see the `README` in the `farmdata2` directory for instructions on how to generate the documentation.

Use the information on the page for the `DropDownWithAllComponent` in the documentation to answer the following questions.

a. How do the props and events documented for the `DropdownWithAllComponent` correspond to the attributes that appear in the `<dropdown-with-all>` HTML element?

b. Based on the documentation, what will happen if the `includesAll` prop is omitted from the `<dropdown-with-all>` HTML element? Try it to confirm your answer.

c. Earlier you guessed at the meaning of the payload for the `selection-changed` event. This documentation provides a little more information.

i. What is the data type of the event payload?

ii. Under what two conditions is the selection-changed event emitted?

d. What additional information is contained in the documentation for the `DropdownWithAllComponent`? How do you think that information will be useful to a FarmData2 Developer?

7. What other custom Vue Components does the documentation indicate are provided by FarmData2?

8. You have probably noticed that the name of the Vue component (`DropdownWithAllComponent`) and the corresponding HTML element (`<dropdown-with-all>`) are different. The `components` property in the `Vue` instance, not surprisingly, indicates the Vue components that will be used. When it does so, it also defines the name of the HTML element that will be used to add the component to the page.

Find the `components` property in the `Vue` instance of the `ui.html` page. Copy that property here and highlight the line the line that maps the `DropdownWithAllComponent` to the name of its HTML element.

Synchronizing with the Upstream:

9. Some time has passed since you created your fork and clone of the upstream FD2School-FarmData2 repository. It is possible that there have been updates to the upstream since you did so. So, as when working on any fork and clone it is important to synchronize your main branch with the upstream so that you have all of the recent changes.

a. Synchronize your local and origin FarmData2 repositories and your feature branch with the upstream. The steps you will need to do this include are:

1. Switch to the `main` branch.
2. Pull the `main` branch from the upstream.
3. Push the `main` branch to your origin.

If you don't remember the commands for this, you can refer back to the previous activity where you will have listed them.

b. Now because your `main` branch has been updated, you will need to merge those changes into the feature branch that you created in the prior activity. To merge the `main` branch into your feature branch:

1. Switch to your feature branch from the prior assignment (`<name>-07-E2E`).
2. Merge the `main` branch into your feature branch.
3. Resolve any conflicts that arise (If you are just doing the FD2School activities there should not be any).

If you don't remember the commands for this, you can refer back to a previous activity where you will have listed them.

c. Your work on this assignment builds from what you did in the prior assignment. So you now need to:

1. Switch to your prior feature branch `<name>-07-E2E` (if you are not there already).
2. Create a new feature branch named `<name>-08-FD2` from your prior feature branch.
3. Switch to your new feature branch.

You will do all of your work for this activity in this feature branch.

If you don't remember the commands for this, you can refer back to a previous activity where you will have listed them.

Adding a New FarmData2 School Sub-tab:

With a little Cypress experience you are now ready to begin working on some E2E Cypress tests using your harvest report.

10. Make sure you have your feature branch checked out and then add another new sub-tab named **FD2** to the FD2 School tab. Have the contents of this new tab be contained in the directory cypress with the page content provided by the file `fd2.html`. **Make a copy of your `e2e.html` file into a file named `fd2/fd2.html`. Also copy the `.spec.js` files from `e2e` to `fd2` and change their filenames to have the fd2 prefix.** Don't forget to clear the Drupal cache when you are done. The result should be that you now have four sub-tabs in the FD2 School tab: HTML, Vue1, Vue2, API, API2, e2e and FD2. For now, your e2e and FD2 tabs will be exactly the same. You'll be modifying and extending the FD2 tab throughout this activity.

11. Commit your changes to your feature branch with a meaningful commit message that describes what you have done and push it to your origin to update your PR.

12. On GitHub create a *Draft Pull Request* for your new feature branch to the upstream FD2School-FarmData2 repository. Be sure to link your PR to the issue for this activity by including a line like the following in the body of your PR:

Addresses #??

Replace the ?? by the issue number in the FD2School-FarmData2 issue tracker for this assignment.

Switching the Crop Dropdown - Spike 1:

In this section you will convert the crop dropdown from being a basic HTML `<select>` element to being a `DropDownWithAllComponent`.

13. The first step in converting the crop dropdown to be a `DropDownWithAllComponent` is to add a `components` property to the Vue instance that defines the name of the HTML element that will contain the component.

Using your answer to #8 as a guide, add a `components` property to the `Vue` instance in your `fd2.html` file so that the `DropDownWithAllComponent` can be used in the page. To be most efficient, your `components` property should only include the components that your page is actually using. So do not simply copy the entire components property from the `ui.html` page.

14. To prepare to convert your crop dropdown to use the `DropDownWithAllCompoent`, use the contents of your `fd2.html` file to answer the following questions:

a. What is the name of the computed property that provides the array of crops that appear in your current crop dropdown?

b. What is the name of the Vue `data` property that is bound to the selected value of your current crop dropdown?

c. What is the value of the `data-cy` attribute of your current crop dropdown element?

15. Replace the `<select>` element used to create the crop dropdown with a `<dropdown-with-all>` component. Your new dropdown should:

- have its contents generated using the computed property you identified above.
- include the "All" option.
- have its selected value bound to the Vue `data` property you identified above.
- have a `data-cy` attribute with the value you identified above.
- not be disabled.

You might find it helpful to refer to example `<dropdown-with-all>` on the `ui.html` page. What you need here is not exactly the same as that, but it will be quite similar.

16. Reload your FD2 sub-tab in FarmData2 and confirm that the `DropDownWithAllComponent` is being used in the page. If all is working well the page will look exactly the same, except the first item in the crop dropdown will now be "All".

17. Modify your page so that "All" is the option that is selected by default in the crop dropdown. Be sure to reload the page and confirm that "All" is now selected in the crop dropdown by default.

18. Generate a harvest report using the default date range (05/05/2020 through 05/15/2020) where we know that there are some crops that have been harvested. Does the "Crop" listed under "Details" in the report change when the crop dropdown is changed? Why not?

19. Add an event handler for the `selection-changed` event to the crops `DropdownWithAllComponent`. Your event handler should update the Vue `data` so that changes in the crop dropdown are reflected in the "Crop" section of the "Details" in the harvest report.

Hint: Use the `ui.html` example and what you learned about the `selection-changed` event handler in question #5 to add the event handler to your `fd2.html` page.

19. Once changes to the crop dropdown are reflected in the "Crops" section of the "Details" commit your changes to your feature branch with a meaningful commit message that describes what you have done and push it to your origin to update your PR.

20. Now modify the `computed` property that generates the rows of the harvest report table so that the table only shows the proper rows (e.g. just "BROCOLLI" or "ALL" crops).

21. Once changes to the crop dropdown properly filter the harvest report table by crop, commit your changes to your feature branch with a meaningful commit message that describes what you have done and push it to your origin to update your PR.

Testing the New Crop Dropdown - Spike 2:

In the prior assignment you wrote a test that checked the first (`0`), fifth (`4`) and last (`110`) crop in the crop dropdown to be (relatively) sure that it contains the correct crops. That test assumed that the dropdown was created as a `<select>` element in the page and used that assumption to test the elements by check that the children (the `<option>` elements) of the dropdown had the correct crop names. Thus, when you changed the crop dropdown to be a `DropdownWithAllComponent`, you broke the test that assumed it was a `<select>`.

22. Update the `beforeEach` in your `fd2.defaults.sepc.js` and `fd2.report.spec.js` so that it visits your new FD2 tab instead of the E2E tab from the previous assignment. Note: if you do not have these `spec` files, copy them from your `e2e` directory as described in #10.

23. Commit your changes to your feature branch with a meaningful commit message that describes what you have done and push it to your origin to update your PR.

24. Run the `fd2.defaults.sepc.js` Cypress tests. Refer to the previous assignment or the `README` in the `farmdata2` directory of the repository if you do not remember how to run the tests.

One of the tests will fail. Why did it fail? Hint: That test used the `children()` function to get the `<option>` elements inside the `<select>` element that is the crop dropdown. Which HTML element is that call getting the `children` of now?

25. Each custom Vue component can encapsulate multiple other HTML elements. For example, the `DropdownWithAllComponent` encapsulates a `<label>`, a `<select>` and all of the `<option>` elements. You could still access the appropriate elements using the `children()` of the `DropdownWithAllComponent`, however that would require that you know the internal details of the component (i.e. the component would not be a good abstraction). To facilitate testing the `DropdownWithAllComponent` (and all other FarmData2 components) they each assign `data-cy` attributes to the elements that they encapsulate.

In this question you will fix your failing test from #24 by changing it to use the `data-cy` attributes that are defined by the `DropdownWithAllComponent` to check that the dropdown contains the proper crops.

a. Use the documentation for the `DropdownWithAllComponent` to complete the following table indicating the `data-cy` attribute associated with the given HTML element.

Element	**data-cy attribute**
`<select>`	
1st `<option> (index 0)`	
5th `<option>`	

b. To access an element inside of a component using its `data-cy` tag you access the component and then use the > operator to access any nested elements. For example, to get the 3rd `<option>` element of a `DropdownWithAllComponent` you would use something like the following:

```
cy.get("[data-cy=crop-dropdown] > [data-cy=dropdown-input] > [data-cy=option2]")
```

In the example above:

- `crop-dropdown` is the `data-cy` value for the `DropdownWithAllComponent`.
- `dropdown-input` is the `data-cy` value of the `<select>` element that is inside the component.
- `option2` is the `data-cy` value of the 3rd `<option>` element in the `<select>` element.

Modify the test that failed so that it passes. The working test should:

- Check `<option>` elements 0, 1, 5 and the last one to be sure they contain the expected crop.
- Check the number of `<option>` elements. You will still need to use `children()` for this one. You just have to be careful with which element you get the children from.

26. Commit your changes to your feature branch with a meaningful commit message that describes what you have done and push it to your origin to update your PR.

Switching the Data Table - Spike 3:

27. Visit the Seeding Report in the BarnKit and generate a report for the date range 05/05/2020 to 05/15/2020. This report should show information about the same records that you have been seeing in your report. However, this table has more columns that your table. What are some of the other differences that you notice from your table?

28. Some of the differences that you may have noticed are because the table that is displayed in the Seeing Report is not a plain HTML table like the one that you created. It is created by a Vue component. This is what gives it its style (red headings, alternating rows highlighted) and its functionality (check boxes for the rows, export and delete buttons at the top and the edit button on the right).

Use the FarmData2 documentation to answer the following questions:

a. What is the name of the Vue component that is used for displaying report tables in FarmData2?

b. What props are required when using this component?

c. Use the examples given in the documentation for the `columns` and `rows` props to fill in the text that would appear in the empty cells in the table below.

		2019-01-10
	M	
Pants		

d. Imagine that you have a `CustomTableComponent` in a page and you want to make it possible for users to delete rows from this table. What prop would you need to set? What event would you need to handle?

29. Like the `DropdownWithAllComponent` the UI sub-tab in the FD2 Examples tab provides a working example of how to use the `CustomTableComponent`. Study the source code for the parts of the UI sub-tab that contain the `CustomTableComponent` to answer the following questions.

a. What HTML element name is used when adding a `CustomTableComponent` to this page? Where in the source code is that name assigned to be a `CustomTableComponent`?

b. Find the Vue `data` property that is bound to the `columns` prop of the `CustomTableComponent` and copy its value (i.e. the array) here.

c. Find the Vue `data` property that is bound to the `rows` prop of the `CustomTableComponent` and copy its value (i.e. the array) here.

30. Use the values you found in the previous question to answer the following questions:

How many columns does this table contain?	
How many columns will be visible in the table?	
Which column is not editable?	
Which column is editable using a dropdown?	
How many rows of data will appear in the table?	
Which item in the table has been purchased?	

31. Convert the HTML table in your FD2 sub-tab to be a `CustomTableComponent`. To do so you will need to make a number of changes. It is suggested that you follow incremental development practices. Complete each of the following steps one-by-one and use the Vue DevTools and `console.log` statements to ensure that each piece is working correctly.

- Create a Vue `data` property to specify the columns that appear in your table. For now, make all of the columns visible and not editable.
- Create a new `computed` property that generates the array of table rows in the proper format for a `CustomTableComponent`. Use the `id` property of the harvest log as the `id` for its table row - you'll probably have to refresh your memory about what a harvest log looks like and figure out how to access its `id` property. The VueDev tools can help here too!
- Add a property to the Vue `components` object so that your page can use the `CustomTableComponent`.
- Add the `CustomTableComponent` to the HTML for your page so that it appears.
- Clean up your page by removing the old HTML table and any Vue `data`, `methods` and/or `computed` properties that are no longer used.

32. Commit your changes to your feature branch with a meaningful commit message that describes what you have done and push it to your origin to update your PR.

Testing with the CustomTableComponent - Spike 4:

In this spike you will add some tests that check the contents of the harvest report table.

33. Create a new test file named `fd2.table.spec.js` in the `fd2` folder for your FD2 sub-tab. Add a `describe`, `beforeEach` and an empty `it` to this file and then confirm that you can run the test in the Cypress test runner.

34. Commit your changes to your feature branch with a meaningful commit message that describes what you have done and push it to your origin to update your PR.

Like the `DropdownWithAllComponent` the `CustomTableComponent` defines a number `data-cy` attributes that can be used in tests. For example, each table header (`<th>` element) is assigned a `data-cy` attribute with a value of `hi` where `i` is the number of the

header (e.g. the leftmost header is `h0`, then `h1`, then `h2`, etc.) The line below uses the `h0` `data-cy` attribute to get the leftmost table header and check that it contains "Date":

```
cy.get("[data-cy=h0]").should("have.text","Date")
```

Modify your empty test in `fd2.table.spec.js` so that it contains a test that checks that all of the table headers are correct. Hint: Remember that your test will need to click the "Generate Report" button to get the table to appear.

35. Commit your changes to your feature branch with a meaningful commit message that describes what you have done and push it to your origin to update your PR.

36. It would also be a good idea to check that the table has the correct number of columns. Use the documentation for the `CustomTableComponent` to find the `data-cy` attribute that for the `<tr>` element that contains all of the `<th>` elements. Then add a statement to your test for the table headers that checks that the `<tr>` element has the right number of children (i.e. `<th>` elements).

37. Commit your changes to your feature branch with a meaningful commit message that describes what you have done and push it to your origin to update your PR.

38. Add a new test (`it`) to your `fd2.table.spec.js` file that will test that filtering by crop works correctly. This test should:

- Generate a report using the default date range.
- Select one of the crops that has been harvested using the crop dropdown.
- Check that the filtered table has the correct number of rows.
- Check that the crop in every row is the one that was filtered for.

Hints:

- You'll need to use the > operator to select the dropdown-input in the `DropdownWithAllComponent` to select the crop.
- At the end of the previous assignment, you found a Cypress function that will let you select a value in a dropdown.
- Use the documentation for the `CustomTableComponent` to find `data-cy` attributes that:
    - Give you the body of the table so that you can check that there are the proper number of rows.
    - Give you the value that appears in a specific column of a specific row so that you can check the name of the crop.

39. Commit your changes to your feature branch with a meaningful commit message that describes what you have done and push it to your origin to update your PR.

Switching the Start and End Date Selection - Spike 5:

This section is optional.

40. There is also a custom Vue component for selecting date ranges (e.g. the start and end dates for a report). Find this component, read its documentation, and look at its example in the `ui.html` page. Then update your `fd2.html` page to use this component instead of the two separate `date` elements that you are currently using.

41. Run the E2E cypress tests test for your FD2 page and fix any of them that fail by updating them to interact with the component you are now using for the dates.

42. Commit your changes to your feature branch with a meaningful commit message that describes what you have done and push it to your origin to update your PR.

Enhancing the Harvest Report Table - Spike 6:

This section is optional.

43. Most of the tables in FarmData2 allow the user to delete rows from the table. For example, in both the UI sub-tab and in the Seeding Report you can select rows by their check boxes in the leftmost column and then click the delete (i.e. trash) button at the top right of the table. The selected rows are then deleted from the table. In this exercise you will modify your `CustomTableComponent` so that the user can delete rows from your harvest report table.

a. Use the documentation for the `CustomTableComponent` to find the prop that you must set to allow rows to be deleted. Add this prop to the `CustomTableComponent` in your harvest report page.

b. Reload the page. There should now be a column of checkboxes in the leftmost column of the table and a delete button at the top right of the table. Try selecting a row and deleting it. Does the row go away?

c. Simply enabling rows to be deleted is not sufficient to delete them. If we think about this, that will make sense. The data that is displayed in the table is generated from the harvest logs that are stored in your Vue `data` via a `computed` property. Thus, to delete a row, it must be removed from the Vue `data`. When it is removed, the computed property will regenerate the data for the table rows without the deleted rows and then the rendered table will update. So, you need to delete the row from the Vue `data`.

Recall that the `CustomTableComponent` emits an event when a row is deleted. So, to actually delete the row, you need to create an event handler for this event.

i. Create the following function in your Vue `methods`:

```
deleteTableRow(rowIDs) {
   console.log(rowIDs)
}
```

ii. Assign this function to be the event handler for the appropriate event from your `CustomTableComponent`. Try deleting a row from the table and inspect the console in the DevTools.

iii. As you have just seen the `rowIDs` parameter is the *event payload* for the event. In this case, that contains and array of the `id`s of the harvest logs corresponding to the rows that were deleted.

Implement the body of the `deleteTableRow` function so that it deletes those harvest logs from your Vue `data`. You might find the implementation of `deleteTableRow` in the UI example a helpful starting point.

d. Commit your changes to your feature branch with a meaningful commit message that describes what you have done and push it to your origin to update your PR.

44. **Extra Hard Challenge:** Note that when you completed the above exercise that deletes rows row from the table you removed them from the `data` from your Vue instance. This caused your computed property and then the rendered table to update. So, you no longer see those rows. However, you did not remove them from the database. Thus, if you were to reload the page and generate the report again, those rows would return.

a. Modify your `deleteTableRow` function so that it also deletes the log from the database. Hint: Look at the FarmData2 documentation there is a function there that will help you. Hint2: Look at the API sub-tab of the FD2 Examples tab, there is an example there that will help you.

45. Most of the tables in FarmData2 allow the user to edit some of the data in the table. For example, you can see this feature in both the UI sub-tab and in the Seeding Report. This challenge has you make one of the fields in your table editable.

a. Make the Crop value in your harvest report table editable using a dropdown that contains the list of all crops.

b. Click the edit icon (i.e. the pencil) in a row and change the crop. Does the value displayed in the table change when you change the crop and click the save button? If it did, great - well done!!! But it is quite possible that it did not.

If it did not add an event handler to your page that uses the emitted payload to update the `data` in your Vue instance by changing the data in the harvest log, then let the computed property regenerate the rows and then then the table will be rerendered with the updated value.

d. Commit your changes to your feature branch with a meaningful commit message that describes what you have done and push it to your origin to update your PR.

46. **Extra Hard Challenge:** Just like when you deleted a row above, when you save the edits from the table into the harvest logs in the Vue data they are not changed in the database.

a. Modify your event hander for row edits such that the harvest log is also updated in the database. Hint: Look at the FarmData2 documentation there is a function there that will help you. Hint2: Look at the API sub-tab of the FD2 Examples tab, there is an example there that will help you.

b. Commit your changes to your feature branch with a meaningful commit message that describes what you have done and push it to your origin to update your PR.

Optional: To help us improve and scope these activities for future semesters please consider providing the following feedback.

a. Approximately how much time did you spend on this activity outside of class time?

b. Please comment on any particular challenges you faced in completing this activity.