FarmData2 Onboarding Activity 04
Vue Data Binding Technology Spike

Introduction:

In the previous activity you learned how to add a sub-tab to FarmData2 and the basic HTML necessary to structure the content within that tab. This included basic HTML, HTML form elements and HTML Tables. You built up a static (hard coded) HTML page that *mocked up* a harvest report. I this activity we'll take the first steps toward making this page more "live." You'll learn about the Vue.js library and a little bit of JavaScript and how they work together to make interactive pages easy to build. The Vue.js library allows us to use data from a JavaScript object – called the *Vue instance* - to define some of the content of the HTML page, rather than hard coding it in the HTML. For this activity, we will still hard code information in the Vue instance. But you'll see that when the Vue instance is changed the page changes too – a very powerful idea called *data binding*. Then in the next activity you'll learn how to use JavaScript to modify the Vue instance and via data binding the page as well. After that, you'll learn how to get data from web services using application programming interfaces. Ultimately, you'll make your harvest report "live" by using JavaScript to modify the Vue instance using real data from the FarmData2 database.

Getting Started:

1. Create a text file `vuespike.html` **outside of the FarmData2 repository**. This can be in your development environment or on your host machine. It doesn't really matter as long as it is not in the FarmData2 repository. You'll use this as sort of a spike for your spike. It will be a place to experiment with some of the Vue.js stuff before trying to put it into your FarmData2 spike. Add the following content to your `vuespike.html` page

```
<!DOCTYPE html>
<html>
  <head>
    <title>Vue Data Binding Spike</title>
  </head>
  <body>
    <h1>An HTML Heading</h1>
  </body>
</html>
```

Then open your `vuespike.html` file in a browser to be sure you can see it.

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

1. Pull main branch from the upstream.
2. Push the main branch to your origin.
3. Merge the main branch into your feature branch (resolving any conflicts that arise)
4. Push your feature branch to your origin.

Give the sequence of commands that you used to complete this synchronization.

3. Synchronizing with the upstream is something you will want to get in the habit of doing every time you begin work. Why do you think it is a good idea to synchronize frequently?

Adding Another Sub-Tab:

4. Make sure you have your feature branch checked out. Add another new sub-tab named A03 to the FD2 Example tab. Have the contents of this new tab be provided by the file `a03.html`. Make a copy of your `a02.html` file into a file named `a03.html`. Don't forget to clear the Drupal cache when you are done. The result should be that you now have two sub-tabs A02 and A03 that are exactly the same. You'll be working on the A03 tab throughout this activity.

5. 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.

Getting Started with Vue.js

Vue School (https://vueschool.io/) is a site that provides both free and paid training for Vue.js developers. In this activity and the next, we'll be using their free *Vue.js Fundamentals* course (https://vueschool.io/courses/vuejs-fundamentals) to get started with Vue. 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.

The following sections will guide you through the videos, asking you to try things out in your `vuespike.html` file first and then having you add some Vue capabilities to your mocked up FarmData2 Harvest Report. Remember that the purpose of a spike is not to get things working, but to learn about a technology. So, your goal here is to learn about and understand Vue so that you'll be able to apply it later to building FarmData2 features. Getting the spikes to work should be a side effect of that learning.

6. Find the free *Vue.js Fundamentals* course and its first video, *Getting Started with Vue.js (3:02)*. Watch that video and follow along by adding content to your `vuespike.html` file. You'll probably want to pause and rewind the video frequently while you modify and test your vuespike.html file to experiment with the new content. When you've finished with the *Getting Started with Vue.js* video answer the following questions based on what you have done.

7. Give the HTML tag that you used to add the Vue library to your `vuespike.html` file.

8. The video said that the tag that added Vue used a CDN. Use your favorite search engine to learn a little about CDNs and write a few sentences in your own words explaining what a CDN is and what its purpose is. You do not have to go into how they work.

9. The `<div>` tag is used to create an HTML element that contains Vue content. Use the MDN HTML element reference (https://developer.mozilla.org/en-US/docs/Web/HTML/Element) from last activity to look up and read about the `<div>` tag. In a few sentences of your own words explain the general purpose of a `<div>` (i.e. not just with respect to Vue).

10. Give the opening HTML tag that defines the `div` element for the Vue content in your `vuespike.html` file.

11. Give the HTML code for the script element containing your Vue object. Be sure to include the code for the Vue object as well.

12. Give the HTML code for the header element that uses the *double mustache syntax* to bind the header content to the Vue object. Be sure that changing the value in the Vue object also changes the heading.

13. Give the HTML code for the text input element that you created. Be sure that you have the data binding setup correctly so that changing the contents of the text input element also changes the heading.

14. Give a brief explanation of how to open the Developer Tools (DevTools) in your browser.

15. Give a command that you could use in the DevTools console to change the property of the Vue object that is bound to the header and the text input element. Note: You will need to assign your Vue instance to a variable as shown in the video to get this to work! Be sure that using your command changes the content of both the text input element and the heading.

That Annoying Flash:

You may have noticed that each time you reload your `vuespike.html` file there is a little flash of unrendered Vue content (i.e. you see the double mustache before it is replaced with the data from your Vue instance). If you haven't noticed it, reload the page a few times and look for it.

16. We can fix that issue by making a few small changes to the page.

a. Add the following `<style>` element to the `<head>` element of your page:

```
<style>
  [v-cloak] {
    display:none;
  }
</style>
```

b. Add the `v-cloak` attribute to the `div` for your Vue content. Be sure that now when you reload the page you don't see the double mustache flash. Give your updated opening `div` tag here.

c. Optional: If you are curious about why this is necessary and how it works you can see https://codingexplained.com/coding/front-end/vue-js/hiding-elements-vue-instance-ready-v-cloak-directive for more details.

Adding Vue to the Harvest Report Spike 1:

Okay, now let's apply what you've learned about Vue thus far to your FarmData2 A03 sub-tab. But first, a couple of notes about the way that FarmOS and Drupal work that will matter when adding the Vue content:

1. Do not include the `script` element that loads the Vue.js library from the CDN.
    - You needed that in your vuespike.html file, but FarmData2 will add this to your page for you in the background. If you are curious, you can see how it does that in the `fd2_example_preprocess_page()` function in the `fd2_example.module` file.

2. Do not add the `style` element that defines the `v-cloak` attribute to the `head`.
    - Again, you needed this in your vuespike.html file, but FarmData2 will add this to your page for you in the background. If you are curious, you can see the the `fd2_example.info` file and the `fd2.css` file that is referenced there.

17. Add some basic Vue capabilities to your `a03.html` page in FarmData2. You should:
    - Add the `div` element for the Vue content and include the `v-cloak` attribute.
    - Create a Vue instance. Be sure to assign it to a variable so that you will be able to access it in the DevTools console.
    - Use your Vue instance to bind the title of the report that appears at the bottom of the page to the value entered in the text field at the top of the page. The text "My Mock Report" should appear by default. Ensure that both changes in the text field and changes to the Vue instance (through the console) change the report title.

    As you add to the data property of your Vue instance you should use consistent and meaningful property names. For example, if you have one named `reportTitle` then be consistent with other properties (e..g `reportStartDate`, `reportCrop`). But if you abbreviate (e.g. `rptTitle`) then abbreviate it in all names (e..g `rptCrop`). This will help others who have to read and modify your code later.

18. Commit your updates to your feature branch with a meaningful commit message and push it to your origin.

Adding Vue to the Harvest Report Spike 2:

The v-model directive can be used not just to bind the value of text field to the Vue instance, but it can be used with most HTML form elements.

19. Update your `a03.html` page in FarmData2 so that:

- The selected report start date is bound to a property in the Vue instance. Hint: you'll need to use v-model with the date input. But you will also need to remove the value attribute that gives it its initial value.
- When the start date is changed, the start date that appears on the report also changes.
- The selected report end date is bound to a property in the Vue instance.
- When the end date is changed, the end date that appears on the report also changes.
- The crop selected in the drop down is bound to a property in the Vue instance.
- When the selected crop is changed, the crop listed in the report also changes.

Note: None of these changes need to affect the table that appears in the report. It should still be static HTML for now.

20. Commit your updates to your feature branch with a meaningful commit message and push it to your origin.

Vue.js Template Syntax and Expressions:

20. Find the *Vue.js Template Syntax and Expressions (1:42)* video in the free *Vue.js Fundamentals* course. Watch that video and follow along by experimenting with similar content in your `vuespike.html` file.

21. Paste the HTML tag(s) that you used in your `vuespike.html` file to experiment with the ternary expression here.

22. As described in the video any JavaScript expression can be used inside the double mustache. Experiment with it a little more by trying the following:

a. Give a header tag using the double mustache that will set the header to be whatever is in the text field prefixed with "Like… " and suffixed with " …I know!" So, if the text field contained "Testing Stuff" then the header would be "Like… Testing Stuff …I know!"

- You'll need to use JavaScript string concatenation for this. You may be able to guess how to concatenate strings in JavaScript based on other languages that you know. But if not, skim the MDN page on the *Handling text — strings in JavaScript* page (https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/Strings).

b. Give a header tag using the double mustache with a ternary statement that will set the heading to 'Woooo!' if the text field contains 'yes' and to 'Boooo!' if the text field contains anything else.

- You'll need to use a JavaScript comparison operation in the condition part of the ternary operator. You may be able to guess how to do that based on other languages that you know. But if not, skim the W3Schools.com page on *JavaScript Comparison and Logical Operators* (https://www.w3schools.com/js/js_comparisons.asp).

c. Now give a header tag that makes your answer to part b case insensitive. So, typing 'Yes', 'YES' or 'YeS'or any other combination into the text field will now set the heading to "Woooo!'. Hint: The key is in the video!

Adding Vue to the Harvest Report Spike 3:

23. Update your `a03.html` page in FarmData2 to use a ternary statement so that if the title text field is empty the title that appears on the report will be "Mock Harvest Report". If any text is entered in the title text field, then the report title should be changed to match.

24. Commit your updates to your feature branch with a meaningful commit message and push it to your origin.

List Rendering:

Find the *List Rendering (1:42)* video in the free *Vue.js Fundamentals* course. Watch that video and then complete the following tasks your `vuespike.html` file.

25. Add the names array below to the data property of your Vue instance:

```
names: [ 'Joe', 'Arun', 'Ouwen', 'Anh', 'Sue' ]
```

Then use Vue's `v-for` to display an unordered list of the names. Give the HTML that you added to your page to display the list.

26. Arrays in JavaScript behave a lot like they do in most languages, but they also provide convenient methods for adding and removing elements from the end of an array. Skim the MDN *Arrays* page (https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/Arrays) to get a little familiar with them.

a. What method adds an element to the end of an array?

b. What method removes an element from the end of an array?

c. Give a command that when used in the DevTools console to adds another name to your list. Be sure to test it to be sure it works.

Adding Vue to the Harvest Report Spike 4:

27. Note that Vue's `v-for` can be used in any element that contains repeated nested elements (e.g. lists like `ul`, `ol`) but also importantly dropdowns created with the `<select>` tag. Update your `a03.html` page in FarmData2 so that:

- The drop down for the list of crops is generated from an array in your Vue instance instead of being hard coded in the HTML.
- The drop down for the list of fields is generated from an array in your Vue instance instead of being hard coded in the HTML.

28. You can also test your implementation by modifying the data in the Vue instance from the DevTools console.

a. Use the DevTools console to add a new crop to the crops drop down. Give the command that you used. Be sure to open the drop down and verify that the new crop is there.

b. Use the DevTools console to add a new field to the fields drop down. Give the command that you used. Be sure to open the drop down and verify that the new field is there.

29. Commit your updates to your feature branch with a meaningful commit message and push it to your origin.

Arrays of Objects:

In JavaScript an object can be created using an *Object Initializer*. For example, the code fragment shown below creates an object with two properties (suit and rank) each with a string value ('H' and '3').

```
let card = {suit: 'H', rank: '3'}
```

Like most other object-oriented languages, the properties of an object are accessed using the *dot notation*. For example, the statement below accesses the value of the rank object and assigns the value of 3 to the variable x.

```
let x = card.rank;
```

It is not required at this time (we'll see more later) but if you are interested in more details on JavaScript Objects and Object Initializers you can skim the following MDN Pages:

- *JavaScript Object Basics*: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/Basics
- *Object Initializer*: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Object_initializer

30. Add the following array of objects to the data property of your Vue instance in your `vuespike.html` file:

```
cards: [
   {suit: 'H', rank: '3'},
   {suit: 'S', rank: 'K'},
   {suit: 'D', rank: '7'}]
```

31. When you have an array of objects in the Vue instance you can use JavaScript's dot notation in the Vue directives to access the object properties. For example, the following will generate an `li` elements for each card containing the value of its `suit` property:

```
<li v-for='card in cards'>{{ card.suit }}</li>
```

Now, add HTML to your `vuespike.html` file that renders a list of the cards so that it looks like the following:

Cards:

- 3 of H
- K of S
- 7 of D

Give the HTML you added to generate the list of cards.

Adding Vue to the Harvest Report Spike 5:

32. Modify your `a03.html` page in FarmData2 so that the table in the Harvest Report is generated from an array of objects in the Vue instance. Do this by adding an array of objects with each object representing one harvest log (i.e. one row of the table.) Each harvest log object should have properties and values for the date, field, crop and yield. Include at least two harvest logs in your Vue instance.

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

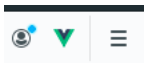
User Inputs & Vue DevTools:

While it is possible to observe and manipulate the Vue instance via the DevTools console, this can become pretty tedious. To help with this, the Vue DevTools can be added to the standard DevTools. The VueDev Tools make seeing and working with the Vue instance super simple.

34. Find the *User Inputs & Vue DevTools (2:34)* video in the free *Vue.js Fundamentals* course. Watch that video and follow along by experimenting by doing similar things in your `vuespike.html` file.

- Note: The video gives instructions for installing the Vue DevTools for Chrome. If you are using FireFox in Linux you can install them in a very similar way:
  - Choose “Add-ons” from the “Tools” menu.
  - Search for “Vue.js devtools”
  - Click through to the Vue.js devtools page
  - Click “+ Add to Firefox”

Once you have the Vue DevTools installed correctly you will (likely… seems a little flaky) see the stylized V in the upper right corner of your browser window. When you visit a page that contains Vue.js code that V will change from gray (left image below) to green (right image below).

- Note 2: Occasionally the Vue DevTools for Firefox on Linux will not show the Vue Tab in when you open the DevTools. If that happens some combination of the following will likely get the Vue Tab to appear:
    - Choose “Add-ons” from the “Tools” menu.
    - Disable the Vue devtools add in.
    - Re-enable the Vue devtools add in.
    - Open DevTools
    - Click the “…” on the right choose “Settings”
    - Check “Vue.js devtools” under “Developer Tools Installed by add-ons”

  More information about this glitch can be found here: https://github.com/vuejs/vue-devtools/issues/403 if you are interested.

Adding Vue to the Harvest Report Spike 6:

35. In order to use the Vue DevTools within FarmData2 you will need to add the following line at the bottom of your script, below your Vue instance, in `a03.html`.

```
Vue.config.devtools = true;
```

36. Commit your updates to your feature branch with a meaningful commit message and push it to your origin.

37. Open your A03 sub-tab, open the DevTools and the Vue DevTools tab. Click on the Vue instance to see its data property in the right hand pane. Scroll until you see the crops array. Paste a screenshot showing the part of the Vue instance that holds the crops array in the Vue DevTool.

37. Experiment with the VueDev tools by adding, removing, editing values in your Vue instance. This models what you will be doing in the next activity where you will be writing JavaScript code that manipulates the data values in the Vue instance, and thus modifies the page through the data bindings.