03 - Vue Data Binding Technology Spike

COMP290 – Large Scale and Open Source Software Development
Dickinson College

Name:

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. In this activity you'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, you 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 thus the page as well, via the data binding. 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 data from the FarmData2 database.

Getting Started:

1. Be sure that you have started FarmData2 using the `fd2-up.bash` command and connect to it using the TigerVNC Viewer as you did in the last activity. You might find it helpful to refer back to the INSTALL.md file in the repository if you don't remember how to start and stop FarmData2.

2. Create a directory named `vuespike` in your home directory. Open the `vuespike` directory in VSCodium and create a text file `vuespike.html` in that directory. You'll use the `vuespike.html` as sort of a spike for your spike. It will be a place to do some small experiments with some of the Vue.js stuff before trying to put it into your actual Vue spike in FarmData2.

3. Add the following content to your `vuespike.html` file:

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

3. Open your `vuespike.html` file using the FireFox browser in the FarmData2 Development Environment. You can do this using the GUI file manager (the file cabinet icon in the launcher at the bottom of the screen). Or you can enter the URL into FireFox:

- file:///home/fd2dev/vuespike/vuespike.html

Type your name into the text field and paste a screenshot of the page rendered in the browser as your answer here.

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 reading, 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 to building FarmData2 features later. Getting the spikes to work should be a side effect of that learning.

4. 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.**

There is one place where you will need to do something different than what the video indicates. They instruct you to access Vue from the URL https://unpkg.com/vue. FarmData2 uses Vue version 2. To ensure that your code uses Vue version to you must add a "@2" to the end of this URL. So, you will instead use the URL: **`https://unpkg.com/vue@2`**

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.

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

6. 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.

7. In the video, the `<div>` tag is used to create an HTML element that contains Vue content. Use the MDN HTML element reference to look up and read about the `<div>` tag:

https://developer.mozilla.org/en-US/docs/Web/HTML/Element

In a few sentences of your own words explain the general purpose of a `<div>` (i.e. not just with respect to Vue.

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

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

10. 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 to try changing the value in the `Vue` object to ensure that the heading also changes.

11. 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 (via changing the `Vue` object).

12. Give a brief explanation of how to open the Developer Tools (DevTools) in the FireFox browser in the FarmData2 Development Environment.

13. 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.

14. 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.

It is not required reading, but if you are curious about why this `v-cloak` 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.

Synchronizing with the Upstream:

15. 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 `main` branch from the `upstream`.
3. Push the `main` branch to your `origin`.

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

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>-02-HTML`).
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).

Give the sequence of `git` commands that you used to complete this merge.

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

a. Switch to your prior feature branch `<name>-02-HTML` (if you are not there already).
b. Create a new feature branch named `<name>-03-Vue1` from your prior feature branch.
b. Switch to your new feature branch.

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

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

Adding a New FarmData2 School Sub-tab:

17. Make sure you have your feature branch checked out and then add another new sub-tab named **Vue1** to the FD2 School tab. Have the contents of this new tab be provided by the file `vue1.html`. Make a copy of your `html.html` file into a file named `vue1.html`. Don't forget to clear the Drupal cache when you are done. The result should be that you now have two sub-tabs in the FD2 School tab: HTML and Vue1. For now, these tabs will be exactly the same. You'll be modifying and extending the Vue1 tab throughout this activity.

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

19. 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.

Adding Vue to the Harvest Report - Spike 1:

Okay, now let's apply what you've learned about Vue thus far to your Vue1 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. You will 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 automatically. 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. You will 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 automatically. If you are curious, you can see the `fd2_example.info` file and the `fd2.css` file that is referenced there.

20. Add some basic Vue capabilities to your `vue1.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 Sample Harvest Report" should appear by default. Ensure that the report title changes both when the text field is changed and when the `Vue` instance is changed via the dev tools console.

    As you add to the `data` property of your `Vue` instance you should use consistent and meaningful property names. For example, if you have a property 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.

21. 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.

22. Update your `vue1.html` page in FarmData2 so that:
    - The report start date is bound to a property in the `Vue` instance. Hint: you'll need to use `v-model` with the date input element. But you will also need to remove the value attribute that you had added earlier to give it its initial value. You can use the Dev Tools to verify that this is working.

- When the start date form element is changed, the start date that appears on the report page also changes. Change the start date element and observe the report to ensure that this is working.
- The selected report end date is also bound to a property in the Vue instance and that 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 and that the selected crop is changed, the crop listed in the report also changes.

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

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

Vue.js Template Syntax and Expressions:

24. 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.

25. Paste the HTML elements and their contents that you used in your `vuespike.html` file to experiment with the ternary expression here.

26. 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 want 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
- If you want another example or two of how to use a ternary operator check out the MDN page for the *Conditional (ternary) Operator*:
  - https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Conditional_Operator

c. Optional: 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 VueSchool video you watched!

Adding Vue to the Harvest Report - Spike 3:

Currently if the text field for the title in your Harvest Report is empty, then there will be no title in the Harvest Report section of the report. It would be better for the report to always have a title.

27. Update your `vue1.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.

28. 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.

29. 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` operator to display an bulleted list (i.e. `<ul>`) of the names. Give the HTML that you added to your page to display the list. You do not need to show the `Vue` data property.

30. 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 to get a little familiar with these operations:

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

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. Test your answers to a and b by using commands in the DevTools console that add or remove items from the `names` array in the `data` property of your `Vue` instance. You should see the new element appear and disappear from the rendered HTML page as you change the `names` array. Give a command that you used to add a name and a command that you used to remove a name.

Adding Vue to the Harvest Report - Spike 4:

31. Note that Vue's `v-for` operator can be used with any element that contains repeated nested elements (e.g. with `li` in lists like `ul`, `ol` as above) but also importantly for `option` elements in dropdowns created with a `<select>` tag. Update your `vue1.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.

32. You can 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.

33. 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 (e.g. Java), the properties of an object are accessed using the *dot notation*. For example, the statement below accesses `rank` property of the `card` object assigns its value (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

34. 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'}]
```

35. 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` element for each card containing the value of its `suit` property:

```
<li v-for='c in cards'>{{ c.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 for the `<ul>` and `<li>` elements that you added to generate the list of cards. You do not need to include the `data` property or the `cards` array.

Adding Vue to the Harvest Report - Spike 5:

36. Modify your `vue1.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 the properties date, area, crop, yield and units with values for each. For example, one harvest log object might be:

```
{date: '2018-05-02', area: 'Chuau-1', crop: 'Kale', 'yield': 10, units: 'Bunches'}
```

Include at least two harvest logs in the array in your `Vue` instance.

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

Installing the Vue DevTools:

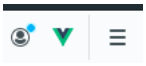
While it is possible to observe and manipulate the `Vue` instance via the DevTools console as you have been doing, this can become pretty tedious. To help with this, a set of Vue DevTools can be added to the standard DevTools. The Vue DevTools make seeing and working with the `Vue` instance super simple.

38. Find the *User Inputs & Vue DevTools (2:34)* video in the free *Vue.js Fundamentals* course. Watch that video and follow along to install the Vue DevTools.

- Note: The video gives instructions for installing the Vue DevTools for Chrome. You are using FireFox in Linux, but you will install these tools in a very similar way:
    - Choose "Add-ons and Themes" from the *hamburger* (☰) menu in FireFox.
    - Search for "Vue.js devtools"
    - Click through to the Vue.js devtools page
    - Click "Add to Firefox"
    - Approve the requests in the dialog boxes that pop up.

Once you have the Vue DevTools installed correctly you will see a 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 to green to indicate that Vue has been detected and that the Vue DevTools will be available.



- Note: Occasionally the Vue DevTools for Firefox on Linux will detect that a page contains Vue.js code but will not show the Vue Tab in the DevTools.

  If that happens some combination of the following will likely get the Vue Tab to appear:
    - Option 1:
        - Close the DevTools
        - Reload the page containing the Vue.js code
        - Reopen the DevTools
    - Option 2:
        - Close the DevTools
        - Choose "Add-ons and Themes" from the *hamburger* (☰) menu.
        - Disable the Vue devtools add in.
        - Re-enable the Vue devtools add in.
        - Reload the page containing the Vue.js code.
        - Reopen the DevTools

- Option 3:
  - Do Option 2 and then…
  - Click the "…" on the right choose "Settings"
  - Unceck and recheck "Vue.js devtools" under "Developer Tools Installed by add-ons"
  - Close the DevTools
  - Reopen the DevTools

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:

39. 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 but inside the `script` tag, in `vue1.html`.

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

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

41. Open your Vue1 sub-tab, open the DevTools and the Vue DevTools tab. Click on the `Vue` instance to see its `data` property. Scroll until you see the `crops` array that you created earlier. Paste a screenshot showing the part of the `Vue` instance that holds the `crops` array in the Vue DevTool.

42. 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 values in the `data` property of `Vue` instance, and thus modifies the rendered page through the Vue data bindings.

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.