

## [IS216] Extra Exercises - Vue Components

### Objectives

- To master the concepts of reactive programming
- To be able to use features provided by Vue framework to build web applications more efficiently
- To practice on applying Vue directives to make web pages reactive

### Instructions

- Questions with no asterisk mark are easy peasy.
- Questions marked with \* are slightly challenging.
- Questions marked with \*\* are challenging.
- Questions marked with \*\*\* are very challenging.

**NOTE:** If you spot any mistakes/errors in the questions, please contact your instructors by email and state the issues. We will try to address it as soon as possible.

Open Weather (*)	2
Weather and Temperature (**)	2
Two Lists (**)	3
Vue component - Login (**)	5

## 1. Open Weather (\*)

*#vue-component*

**Resource:** folder 'open\_weather'

Look at the given index.html.

Create a Vue component 'weather-comp' in weather-comp.js

1. It has a property 'city'
2. It will retrieve the weather details of the specified city from openweathermap (refer to <https://openweathermap.org/current> for details)
3. Example, if the city is 'London', it will display the following details:

**London** Few clouds, 19 °C, 72% humidity, wind 3.6m/s.

If done correctly, the expected output index.html is as follows:

### World's weather

**Singapore** Light rain, 27 °C, 83% humidity, wind 2.1m/s.

**London** Few clouds, 19 °C, 72% humidity, wind 3.6m/s.

**Beijing** Light rain, 25 °C, 83% humidity, wind 4m/s.

**Washington** Few clouds, 21 °C, 30% humidity, wind 4.6m/s.

## 2. Weather and Temperature (\*\*)

*#vue-component #bootstrap*

**Resource:** nil

Convert your AJAX exercise "weather" into a Vue.JS app and combine with your Vue.JS exercise "temperature" to create your own Singapore weather pages.

Notice the top menu is common across all four pages and the menu item for the current page is disabled. Create a Vue component that can be reused across all four pages.

Temperature
2-hourly
24-hour
4-days

# Singapore temperature

01/06/2020, 02:00:25 Refresh

Station	Temperature
Banyan Road	28.6
Clementi Road	28.6
East Coast Parkway	28
Nanyang Avenue	27.1
Scotts Road	27.7

Temperature
2-hourly
24-hour
4-days

# 2-Hourly dated 2020-06-01

Area	00:30-02:30	01:00-03:00	01:30-03:30	02:00-04:00	02:30-04:30	03:00-05:00	03:30-05:30
Ang Mo Kio							
Bedok							
Bishan							
Boon Lay							
Bukit Batok							
Bukit Merah							

Temperature
2-hourly
24-hour
4-days

# 24-hour dated 2020-06-01 08:04

## Thundery Showers

	Low	High
Humidity	55	95
Temperature	25	34
Wind Speed(SSE)	5	15

Start	End	North	South	East
2020-06-01T06:00:00+08:00	2020-06-01T12:00:00+08:00	Thundery Showers	Partly Cloudy (Day)	Partly Cloudy (Day)

Temperature
2-hourly
24-hour
4-days

# 4-days

Date	Forecast	Temperature		Humid		Wind		
		Low	High	Low	High	Direction	Low	High
2020-06-02	Afternoon thundery showers.	25	34	50	95	S	5	15
2020-06-03	Afternoon thundery showers.	26	34	50	95	S	5	15
2020-06-04	Late morning and early afternoon thundery showers.	26	34	50	95	SSW	10	20

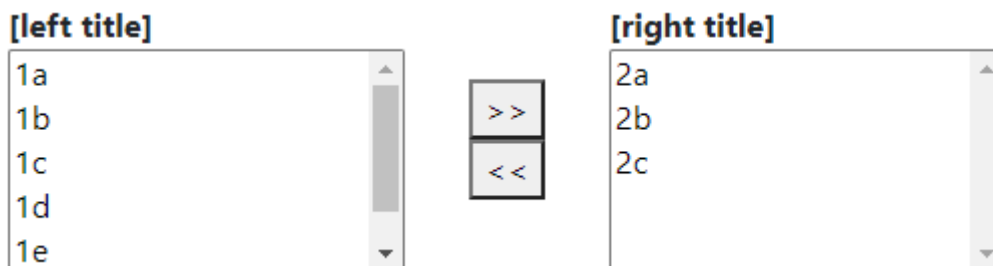
### 3. Two Lists (\*\*)

#vue-component

**Resource:** folder 'two-lists'

Look at the given resource index.html first.

Create a Vue component 'two-lists' that looks like this



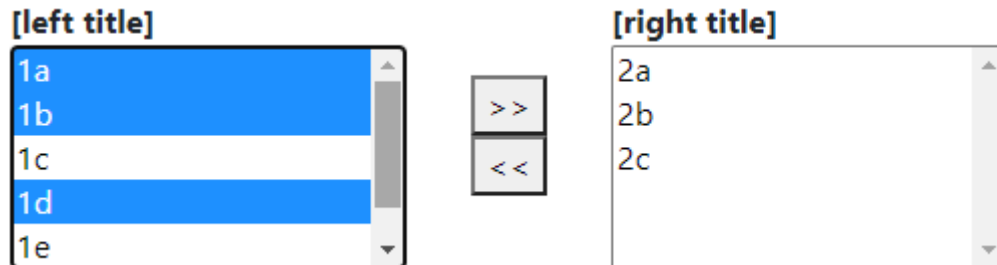
It has the following properties

- 'size' - the number of options to display in each dropdown list
- 'width' - the width of the whole component; e.g. 500px

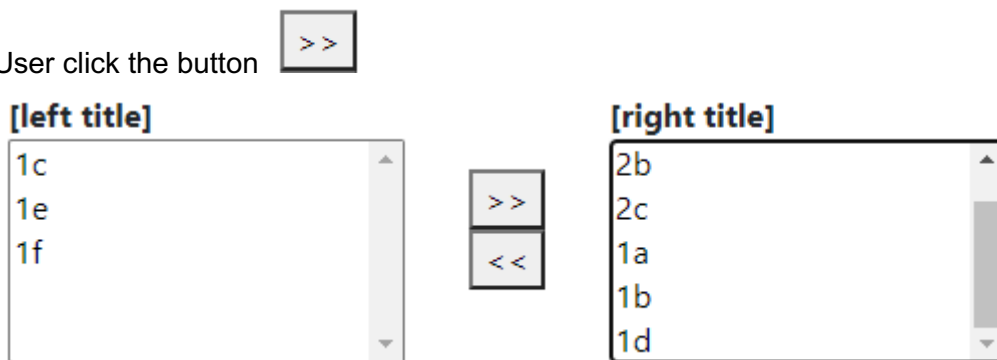
- 'left-title' - text to be displayed above the left dropdown list
- 'left-list' - array of strings to be displayed as options for the left dropdown list
- 'right-title' - text to be displayed above the right dropdown list
- 'right-list' - array of strings to be displayed as options for the right dropdown list

Implement the component such that it works as described in the below scenarios

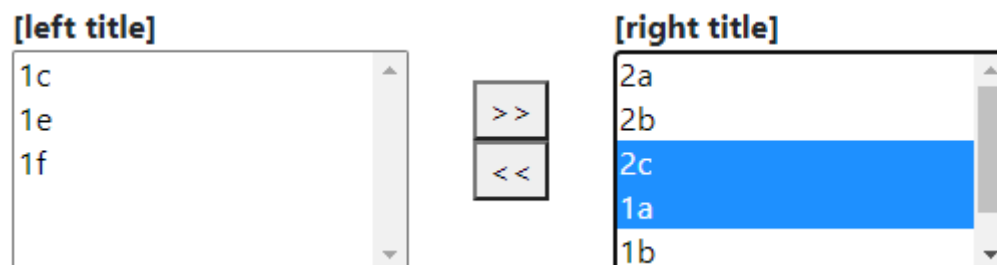
1. User selects a few options from the left list.



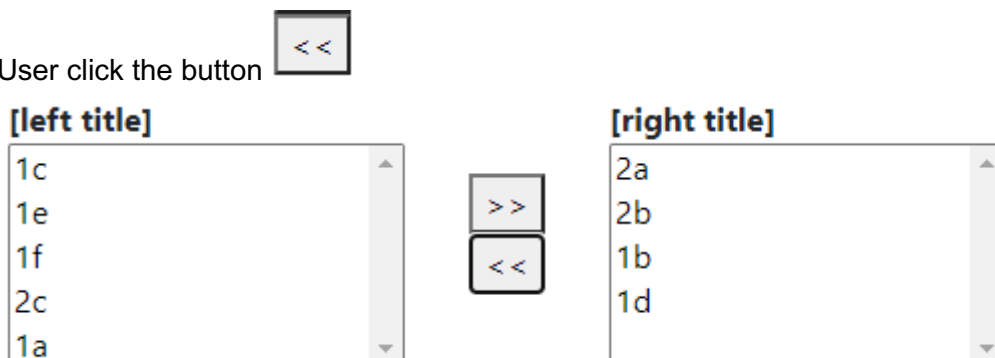
2. User click the button



3. User selects a few options from the right list.



4. User click the button

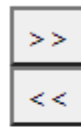


If done correctly, the given index.html should look like this.

## Fruits

### Likes

apple  
orange  
pear  
papaya  
rambutan



### Dislikes

starfruit  
jackfruit

## People

### Friends

Alfred  
Betty  
Cathy  
Desmond  
Ellen  
Fred  
George



### Acquaintances

Irene  
Julie  
Katty  
Leonard  
Margaret  
Norman

And you should be able to move items from left to right lists.

## Fruits

### Likes

papaya  
rambutan  
jackfruit  
durian  
jackfruit



### Dislikes

starfruit  
apple  
orange

## People

### Friends

Irene  
Julie  
Katty  
Leonard  
Margaret  
Norman



### Acquaintances

Alfred  
Betty  
Cathy  
Desmond  
Ellen  
Fred  
George

## 4. Vue component - Login (\*\*)

*#vue-component-event*      *#bootstrap*

You will build a vue component that allows the user to login with his user id and password.

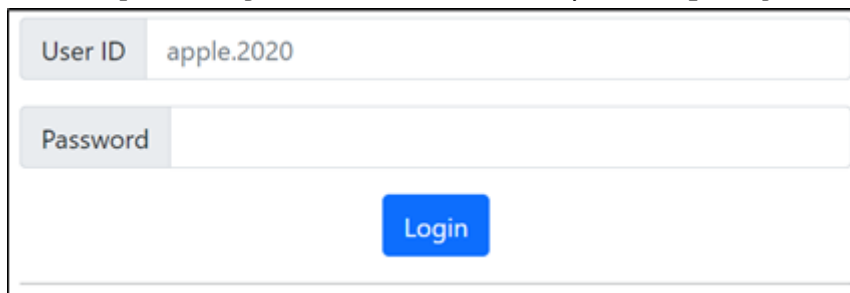
1. Create a folder 'login-component' in your WAMP web root folder and copy the given resource into it. If done correctly, the folder structure should look like this:

```
<web root>
+- login_component\
  |
  +- server\      (given resource)
  | +- model\
  | | +- Account.php
  | | +- AccountMgr.php
  | +- authenticate.php
  |
  +- component.js   (given resource, to edit)
  +- login.html     (given resource, to edit)
```

2. The `authenticate.php` is to be hosted on your local WAMP server. It simulates a mockup JSON Web API that authenticates user id and password. Look at the given code to understand the required HTTP protocol and parameters, and the expected response.
3. The following instructions will guide you to do the following:
  - a. Define a Vue component `my-login` (in `component.js`) for the login form.
  - b. update the given `login.html` to authenticate a user, display a welcome message and logout the user.

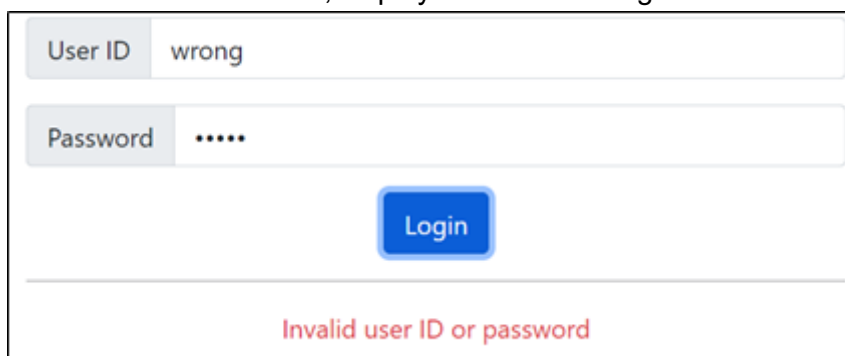
#### Part A: Vue component `my-login`

1. Edit `component.js` and define a Vue component `my-login` that looks like this:



A login form with two input fields: 'User ID' containing 'apple.2020' and 'Password' which is empty. Below the fields is a blue 'Login' button. The form is enclosed in a light gray border.

2. Upon clicking "Login", the component will authenticate the user's credentials by sending an AJAX request to `server/authenticate.php`.
3. If the authentication fails, display an error message "Invalid user ID or password".



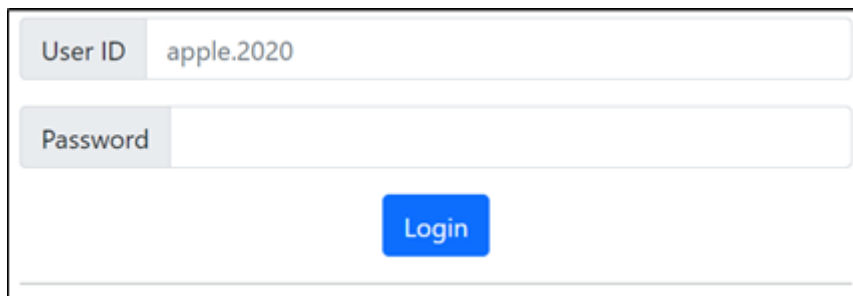
The same login form as above, but with 'User ID' containing 'wrong' and 'Password' containing five dots. Below the 'Login' button, the text 'Invalid user ID or password' is displayed in red. The form is enclosed in a light gray border.

4. If the authentication succeeds, the component will emit an event `login` with a JSON object whose values represent the authenticated user.

- a. An example of the JSON object: `{ userid: "apple.2020", name: "Apple TAN" }`

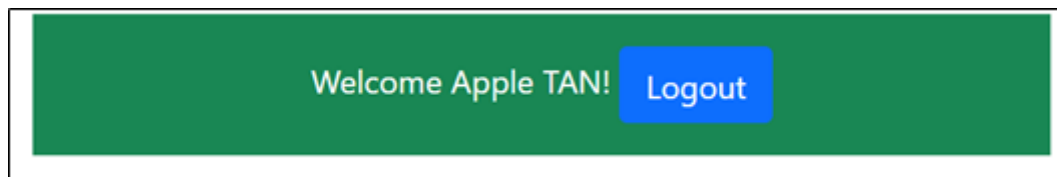
### Part B: login.html

1. Edit `login.html` to use the Vue component `my-login`. Upon loading, `login.html` looks like this:



The screenshot shows a login form with a light gray border. It contains two input fields with light gray labels on the left. The first field is labeled 'User ID' and contains the text 'apple.2020'. The second field is labeled 'Password' and is empty. Below these fields is a blue button with the text 'Login' in white.

2. Read and understand the given code for the Vue application object.
3. Edit the Vue app object in `component.js` to create a method `doLoginSuccess(...)`
  - a. Bind the method as the event handler for the Vue component `my-login`'s event `login`.
  - b. Upon receiving the event (i.e. login is successful), the event handler will do the following:
    - i. Hide the Vue component `my-login`.
    - ii. Display a welcome message, the user's name and a 'Logout' button.



4. Upon clicking the `Logout` button, the Vue component `my-login` is shown again.