Kirirom Institute of Technology

FULL STACK DEVELOPMENT BOOTCAMP

Module 1 (Front End) - Vue

Instructions:

- Each exercise carries 20 points.
- Each exercise contains Description of the problem, requirement, hint and expected output.
- Read the instruction clearly before practicing the exercises.
- The answer sheet must be named after the naming convention mentioned in each exercise.
- Each exercise done by student is expected to give the exact same output as mentioned in the document. Failing which may reduce points or assign 0 for that particular exercise.
- All the exercises must be must be completed on time.
- You may discuss about the problem conceptually with the trainer or fellow students or from internet **however getting direct answers or copying code from others or from internet is strictly prohibited.**
- All the program files must be submitted to google classroom.
- All the exercises will be evaluated and points will be updated by the end of each week.
- Based on the points obtained by each student the overall ranking will be maintained.
- **All the documents (including this document) used in this bootcamp are sole properties of KIT and for internal purposes only. Must not be shared to anyone outside of this bootcamp.**

Kirirom Institute of Technology

FULL STACK DEVELOPMENT BOOTCAMP

Module 1 (Front End) - Vue

<u>EXERCISE_01:</u>

Description:

- Research and write down the definitions for the topics mentioned below.
- What is Vue and its advantages?
- What is the role of Vue as a front-end framework?
- What is the difference between Vuejs and Bootstrap?
- What is a progressive framework?
- What are Single File Applications?
- What are the 3 parts of a component?
- What are the Differences between Vue CDN and Vue Cli?

Requirements:

- Document must be named as: **01_Mod_01_Vue_Intro.docx** and be submitted to google classroom.

Expected Outcome:

To understand the basic terminologies used in the Vue learning path.

EXERCISE_02:

Description:

- Create first Vue application by using CDN. Create a Vue instance inside the <script> tag of the html file. The id of the Vue instance must be **"app"**. Create a property named **"message"** in the Vue instance that contains the message **"Hello world"**. Now display the **"message"** property inside a div element**.**

Requirements:

- Program must be named: **02_index.html** and be submitted to google classroom.

Hint:

- Visit the official website https://vuejs.org/v2/guide/ and copy the script to use Vue in your html file.
- Learn to use {{ }}
- {{ message }}
- Creating a Vue instance

Output:

Run the **02_index.html** file in **google chrome**

"Hello World"

Expected Outcome:

To be able to run the first Hello world Vue application.

EXERCISE_03:

Description:

- Create a Vue application by using CDN. Create a Vue instance inside the <script> tag of the html file. The id of the Vue instance must be **“app”**. Create two properties **“name”** and **“age”** in the Vue instance that contains the message **“<Your Name>”** and **“<Your Age>”**. Now concatenate the **“name”** and **“age” separated with a: (Colon)** properties and display inside a div element**.**

Requirements:

- Program must be named: **03_string_interpolation.html** and be submitted to google classroom.

Hint:

- Use {{ }}
- Use + operator

Output:

Run the **03_string_interpolation.html** file in google chrome

[Example Output]

Arun:21

Expected Outcome:

To be able to understand the string concatenation among different data types.

Kirirom Institute of Technology

EXERCISE_04:

Description:

- Create a Vue application by using CDN. Create a Vue instance inside the <script> tag of the html file. The id of the Vue instance must be **"app"**. Create two properties **"google"** and **"facebook"** in the Vue instance that contains the following URLs as **strings "https://www.google.com/"** and **"https://www.facebook.com/"**. Now render the **"google"** and **"facebook"** properties as hyperlinks **(Clickable links)** and display inside two different **<h1>** elements**.**

Requirements:

- Program must be named: **04_data_binding.html** and be submitted to google classroom.

Hint:

- Use {{ }}
- v-bind:href or :href

Output:

Run the **04_data_binding.html** file in google chrome

https://www.google.com

https://www.facebook.com

Note. Links must be clickable

Expected Outcome:

To be able to understand the one way data binding using **v-bind**.

EXERCISE_05:

Description:

- Create a Vue application by using CDN. Create a Vue instance inside the <script> tag of the html file. The id of the Vue instance must be **"app"**. Create a property named **"message"** in the Vue instance that contains empty string**.** Create an html form with a text input field in it and a **<h1>** element under it. Both the input field as well as **<h1>** must be bound with **v-model.** If a text is entered in the input box of the form it should also appear in the **<h1> (Text must be reactive).** Even if the **"message"** property is updated from the google chrome console the change must reflect in the input box and the **<h1>** element.

Requirements:

- Program must be named: **06_two_way_data_binding.py** and be submitted to google classroom.

Hint:

- v-model
- Learn changing values from chrome console
- Reactivity

Output:

A form with text with a text input box must be displayed followed by an **<h1>** element.

Inputting text must appear in the **<h1>**

Also values to be updated in the web page if property changed from the Chrome console

Expected Outcome:

To be able to understand the two-way data binding using **v-model**.

EXERCISE_06:

Description:

- Create a Vue application by using CDN. Create a Vue instance inside the <script> tag of the html file. The id of the Vue instance must be **"app"**. Create a property named **"message"** in the Vue instance that contains empty string**.** Create an html form with a text input field in it and a **<h1>** element under it. Both the input field as well as **<h1>** must be bound with **v-model.** If a text is entered in the input box of the form it should also appear in the **<h1> (Text must be reactive).** Even if the **"message"** property is updated from the google chrome console the change must reflect in the input box and the **<h1>** element.

Requirements:

- Program must be named: **06_two_way_data_binding.html** and be submitted to google classroom.

Hint:

- v-model
- Learn changing values from chrome console
- Reactivity

Output:

A form with text with a text input box must be displayed followed by an **<h1>** element.

Inputting text must appear in the **<h1>**

Also values to be updated in the web page if property changed from the Chrome console

Expected Outcome:

To be able to understand the two-way data binding using **v-model**.

EXERCISE_07:

Description:

- Create a Vue application by using CDN. Create a Vue instance inside the <script> tag of the html file. The id of the Vue instance must be **"app"**. Create three check boxes with the value attributes namely (Menu 1, Menu 2 and Menu 3) and an **<h1>** element. The Checkboxes as well as **<h1>** must be bound with **v-model.** Create an Array named **"menus"** in the Vue instance to update the values of the selected check boxes. The **<h1>** element must be updated and show the value of the checkboxes selected immediately if there is any change in the selection.

Requirements:

- Program must be named: **07_two_way_data_binding_02.html** and be submitted to google classroom.

Hint:

- v-model
- Html Check Boxes and value attribute
- Array to store the values of checkboxes need to be created in the view instance
- toString() can be used to concatenate the values of checkboxes

Output:

Three checkboxes followed by an **<h1>** element to be displayed.

<h1> element must be updated with the values of checkboxes based on selection

Expected Outcome:

To be able to understand the two-way data binding using **v-model**.

EXERCISE_08:

Description:

- Create a Vue application by using CDN. Create a Vue instance inside the <script> tag of the html file. The id of the Vue instance must be **"app"**. Create a property named **"count"** in the Vue instance and assign 0 to it. Create two buttons namely increment and decrement and decrement. Create an **<h1>** element. The **<h1>** must be bound with **v-model.** Each time when the increment button is clicked the **"count"** must be incremented and updated in the **<h1>** element and **"count"** to be decremented each time when the decrement button is clicked.

Requirements:

- Program must be named: **08_button_click_event.html** and be submitted to google classroom.

Hint:

- v-on:click or @click

Output:

Two buttons namely "increment" and "decrement" followed by an **<h1>** element to be displayed.

<h1> element to be updated with the latest value of the **"count"** property from the Vue instance based on button clicks.

Expected Outcome:

To be able to understand and use v-on:click event.

EXERCISE_09:

Description:

- Create a Vue application by using CDN. Create a Vue instance inside the <script> tag of the html file. The id of the Vue instance must be **"app"**. Create a property named **"message"** with the default message **"Hello World"** in the Vue instance and display its value using **<h1>** element. Whenever the mouse hovers the **<h1>** element the string must be reversed and displayed and when the mouse is out then it must be bringing back to its previous state again. **Note. There shouldn't be any separator like Comma or space between letters if not provided in the original text.**

Requirements:

- Program must be named: **09_mouse_hover_mouse_leave.html** and be submitted to google classroom.

Hint:

- v-on:mouseover or @mouseover and v-on:mouseleave or @mouseleave
- Learn to use methods in Vue instances
- reverse()
- Method to reverse

Output:

Reversing the text when hovered on the **<h1>** element and put back to previous state if mouse left the **<h1>** element.

Expected Outcome:

To be able to understand and use mouse events.

EXERCISE_10:

Description:

- Create a Vue application by using CDN. Create a Vue instance inside the <script> tag of the html file. The id of the Vue instance must be **“app”**. Create a property named **“message”** with the default message **“Hello World”** in the Vue instance and display its value using a **<div>** element. Create an object named “**mystyle1”** in the Vue instance. In “**mystyle1”** create two properties namely “**color”** and “**fontSize”** with values **‘blue’** and **“30px”**. The style details “**color”** and “**fontSize”** from the Vue instance. Whenever the mouse hovers the **<h1>** element the “**color”** and “**fontSize”** properties must be updated with **“green”** and **“50px”** when the mouse left the style must be bringing back to the default color. Create two methods that does the required changes on the style properties.

Requirements:

- Program must be named: **10_working_with_styles.html** and be submitted to google classroom.

Hint:

- v-on:mouseover or @mouseover and v-on:mouseleave or @mouseleave
- v-bind:style or :style
- Method to change style properties in the Vue instance

Output:

Changing the style properties when hovered on the **<h1>** element and put back to previous state if mouse left the **<h1>** element.

Expected Outcome:

To be able to understand and use mouse events.
To be able to understand the v-bind:style

EXERCISE_11:

Description:

- Create a Vue application by using CDN. Create a Vue instance inside the <script> tag of the html file. The id of the Vue instance must be **"app"**. Create a property named **"message"** with the default message **"Hello World"** in the Vue instance and display its value using a **<div>** element. Create two css classes namely **.hovered** and **.nothovered** in the **<style>** element of the HTML with the "**color"** and "**font-size"** properties. In the .active class the properties must be assigned with **'blue'** and **'40px' and in the .notactive** class with **'red'** and **'30px'**. Create **isActive** property in the Vue instance and assign **"false"** as the default value. If the mouse is hovered on the **<div>** Whenever the mouse hovers the **<div>** element the "**color"** and element assign true to the **isActive** and **false** when the mouse left the div element. Whenever the **isActive** property changed to true assign the **.active** class for the **<div>** element and **.notactive** when the value is false.

Requirements:

- Program must be named: **11_working_with_style_classes.html** and be submitted to google classroom.

Hint:

- v-on:mouseover or @mouseover and v-on:mouseleave or @mouseleave
- v-bind:class or :class – Use ternary operator to take decision

Output:

Set the class to .**active** to **<div>** when the mouse hovered on it and **.notactive** when mouse left.

Expected Outcome:

To be able to understand and use mouse events.

To be able to understand the v-bind:class

To be able to understand applying css classes on an element

To be able to use Ternary operator for applying different css classes based on confition

EXERCISE_12:

Description:

- Create a Vue application by using CDN. Create a Vue instance inside the <script> tag of the html file. The id of the Vue instance must be **"app"**. Create a property named **"name"** and assign any name to it and another property named **"isActive"** to hold **true** or **false** in the Vue instance. Create an Html form with a text input field with a hint **"Enter your name here"** and an **<h1>** element. apply **two-way data binding** between the **text input field** and the property **"name"** in the Vue instance. Must use conditionals available in the Vue directory **(Ternary operator must not be used)** in the div element and when the name entered matches **<your name>** display **"Access Granted welcome Mr/Ms. <name>"** in **green color** and display **"Access Denied Sorry Mr/Ms. <name>"** in **red color.** Note. You may use any type of styling that you have learned before in this bootcamp.

Requirements:

- Program must be named: **12_working_conditionals.html** and be submitted to google classroom.

Hint:

- v-model
- v-if, v-else-if and v-else

Output:

Must show the desired output if the value of the **"name"** property in the Vue instance changes from the HTML form or from the console of the browser. Two way data binding must be applied for the properties.

Expected Outcome:

To be able to understand and use conditionals available in Vue.
To be able to revise the two way data binding learned before.

EXERCISE_13:

Description:

- Create a Vue application by using CDN. Create a Vue instance inside the <script> tag of the html file. The id of the Vue instance must be **"app"**. Create an array of object named **"table_details"** in the Vue instance and assign the objects of 5 students with **[id, name, age]** in it. Create an Html form with a <**table>** and an **<ol>**. Read the list from the Vue instance**, loop through** them and render each student object as a row in the <**table>** and list item in the **<ol>** (Each attribute of the object to be separated with **:(Colon)** for the list)**.**

Requirements:

- Program must be named: **13_loops_rendering.html** and be submitted to google classroom.

Hint:

- v-for
- Array of objects

Output:

Table must be rendered from the list of arrays in the Vue instance.

Expected Outcome:

To be able to understand and use v-for available in Vue.
To be able to render the array elements from the list.

EXERCISE_14:

Description:

- Create a Vue application by using CDN. Create a Vue instance inside the <script> tag of the html file. The id of the Vue instance must be **"app"**. Create a **methods** property in the Vue instance and assign a function named **"helloMethod"** to it **(Use ES6 way of creating functions)**. Display a log message **"Hello from the Vue method"** whenever it is called. Create a button and whenever the button is clicked it must call the **"helloMethod"** from Vue instance.

Requirements:

- Program must be named: **14_vue_methods.html** and be submitted to google classroom.
- Research about Vue methods and write a short note under the program on when to use methods in Vuejs app development.

Hint:

- Es6 function definition
- Console.log

Output:

When the button is clicked the methods property to be called and display the message to the console.

Expected Outcome:

To be able to understand the methods in a Vue instance.
To be able to differentiate methods, computed properties and watchers.

EXERCISE_15:

Description:

- Create a Vue application by using CDN. Create a Vue instance inside the <script> tag of the html file. The id of the Vue instance must be **"app"**. Create a property **"name"** in the Vue instance and create a **computed** property in the Vue instance and assign a function named **"computeName"** to it **(Use ES6 way of creating functions)**. Reverse the name and display the same as log message in the console when the app is loaded.

Requirements:

- Program must be named: **15_vue_computed_properties.html** and be submitted to google classroom.
- Research about Vue computed and write a short note under the program on when to use computed properties in Vuejs app development.

Hint:

- Es6 function definition
- Console.log

Output:

When the button is clicked the methods property to be called and display the message to the console.

Expected Outcome:

To be able to understand the methods in a Vue instance.
To be able to differentiate methods, computed properties and watchers.

<u>EXERCISE_16:</u>

Description:

- Create a Vue application by using CDN. Create a Vue instance inside the <script> tag of the html file. The id of the Vue instance must be **“app”**. Create a property **“status”** in the Vue instance and assign false as the default value. create a **watcher** property in the Vue instance and assign a function named **“watchStatus”** to it **(Use regular Javascript way of creating functions)**. Change the value of status from the Chrome console and display the log message on the console whenever the value of status is changed.

Requirements:

- Program must be named: **16_vue_watcher_properties.html** and be submitted to google classroom.
- Research about Vue **watchers** and write a short note under the program on when to use watchers on properties in Vuejs app development.

Hint:

- JS function definition
- Console.log

Output:

When the button is clicked the methods property to be called and display the message to the console.

Expected Outcome:

To be able to understand the methods in a Vue instance.
To be able to differentiate methods, computed properties and watchers.

EXERCISE_17:

Description:

- Create a Vue application by using CDN. Create a Vue instance inside the <script> tag of the html file. The id of the Vue instance must be **"app"**. Create your own tag/element (template) named **<kit>** using **Vue components** with a property **"schoolName"** assign **"Kirirom Institute of Technology"** to it and create a method named **getAscii** which sums up the acii value of each character of the **schoolName** and returns it. Use the **<kit>** element inside the div with the id **"app"**. When the **<kit>** element is used it must display a <**button>** and when the button is clicked it must call the **getAscii** method of the Vue component and display the number.

Requirements:

- Program must be named: **17_vue_components.html** and be submitted to google classroom.
- Research about Vue **components** and write a short note under the program on when to use **components** in Vuejs app development.

Hint:

- Vue.component
- data()

Output:

When the **<kit>** elements is used then the sum of ascii values of the school names must be displayed.

Expected Outcome:

To be able to understand and create Vue components.
To be able to create your own template in Vuejs.
To be able to create data and methods inside a Vue component.
To be able to call the method of a Vue component from template.

Kirirom Institute of Technology

EXERCISE_18:

Description:

- Create a Vue application by using CDN. Create a Vue instance inside the <script> tag of the html file. The id of the Vue instance must be **"app"**. Create an array 5 objects named **studentDetails** with id, name and age details. Create a Vue component template named **<inner>** which displays **<li>** when used**.** Create another own tag/element (template) named **<outer>** using **Vue components** which displays an **<ol>** when used**.** and it must use the **<inner>** to render and display <**li>** and need to render the objects from the array **studentDetails** using **v-for**.

Requirements:

- Program must be named: **18_nested_vue_components.html** and be submitted to google classroom.
- Research about **nested Vue components** and write a short note under the program on when to use **nested Vue components** in Vuejs app development.

Hint:

- Nested Vue.component
- data()

Output:

Use only **<outer>** and it must display all the student names as list items.

Expected Outcome:

To be able to understand and create nested Vue components.
To be able to create your own template in Vuejs.
To be able to create data and methods inside a Vue component.
To be able to call the method of a Vue component from template.

ALERT

Make sure that you follow all instructions properly. Be aware that each character is important. Upload all the exercises on the google classroom created and shared with you.