[IS216] Extra Exercises - Vue Basics/Events

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.

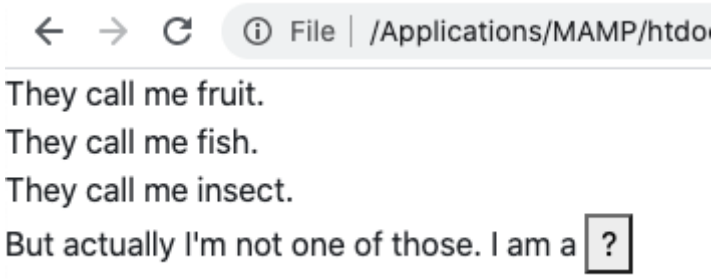
NOTE: If you spot any mistakes/errors in the questions, please contact your instructors by email and state the issues. We will try to address it as soon as possible.

Question 1 - Puzzle

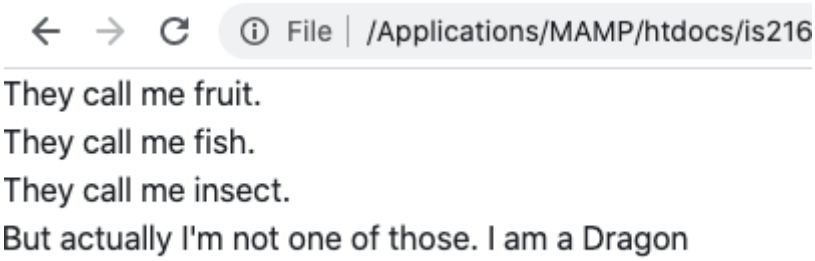
Go to puzzle directory.
Complete the following Part A in `puzzle.html` file.

Part A (*)

- puzzle.html shows the following web page:



- Add code in puzzle.html such that when the **?** button is clicked, it shows the following:



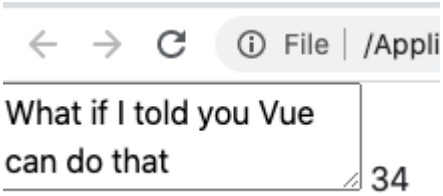
Question 2 - Maximum Characters

Go to `maxchar` directory.
Complete the following Parts A and B in `maxchar.html` file.

`maxchar.html` contains a textarea which is bound to a Vue instance. It has two properties: **text** and **limit**. You are to build a text area that warns the user when it is reaching the maximum allowed number of characters.

Part A (*)

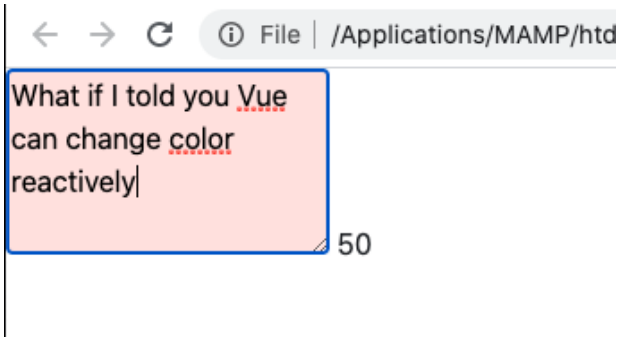
- Bind the given Textarea with **text** property of Vue instance so that it shows the following message in the textarea box



- Set the maximum character limit of the Textarea to the **limit** property of Vue instance so that the textarea cannot contain more than the limit value.

Part B (**)

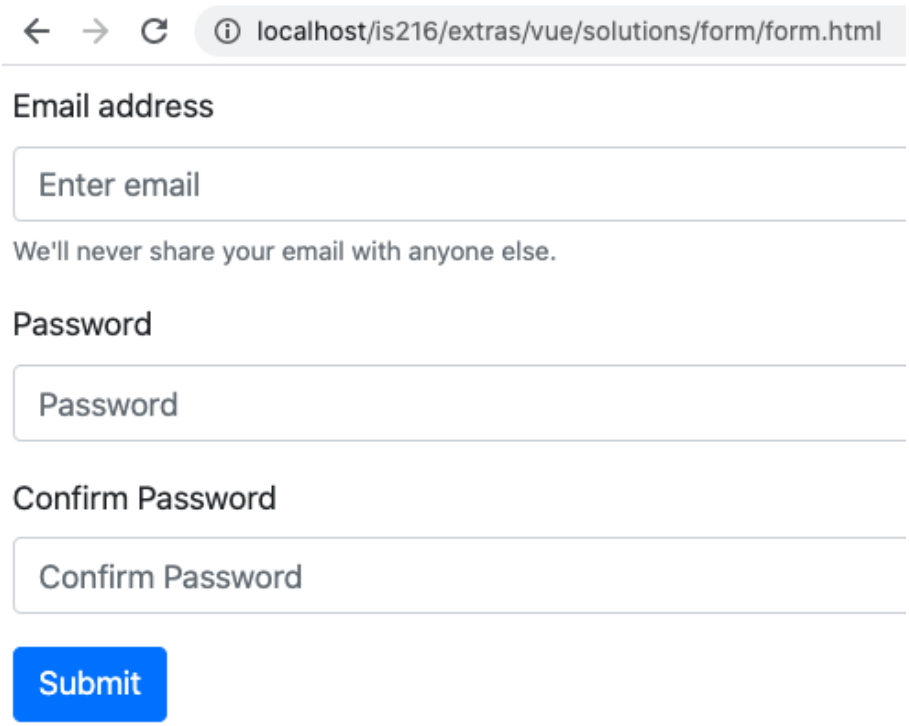
- Implement a computed property **longText** in the Vue instance, which computes the number of characters in the textarea box and return Boolean 'True' if the difference between the number of characters in the textarea box and the **limit** is less than 10; otherwise return Boolean 'False'.
- Add a class binding in the textarea box so that the textarea box changes the background color when there are only 10 characters left before the maximum number of allowed characters is reached. Use the CSS class **warn** given in maxchar.html



Question 3 - Form Validation

Go to `form` directory.
Complete the following Part A in `form.html` file.

`form.html` contains a form as shown in the following:



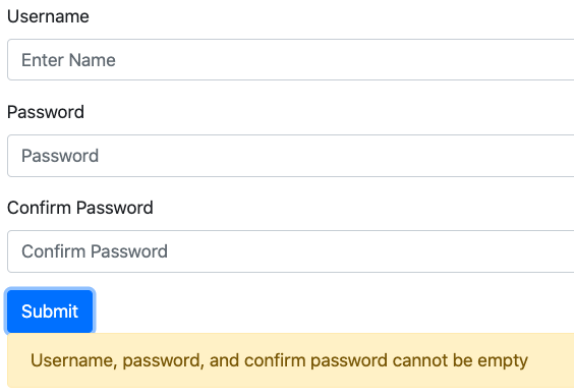
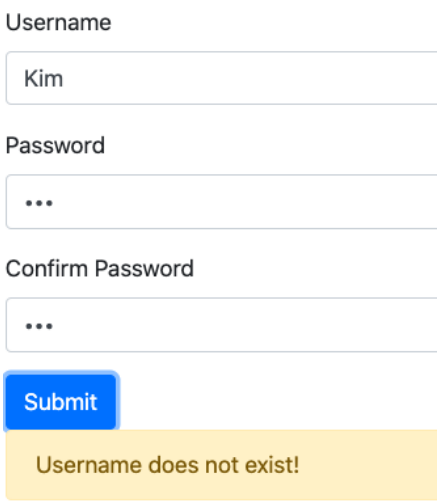
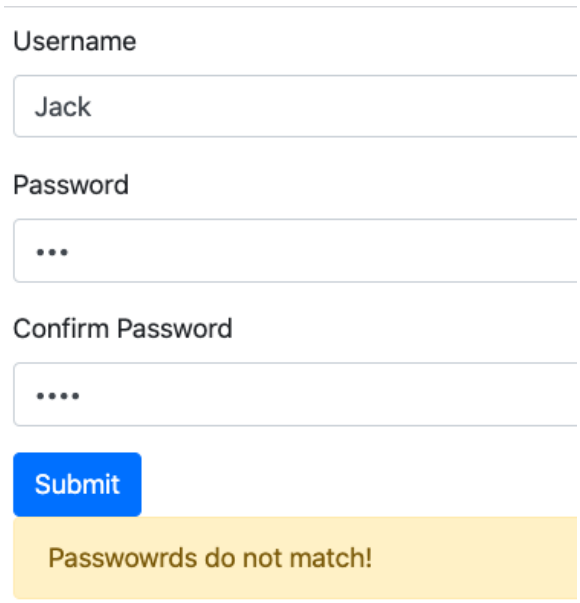
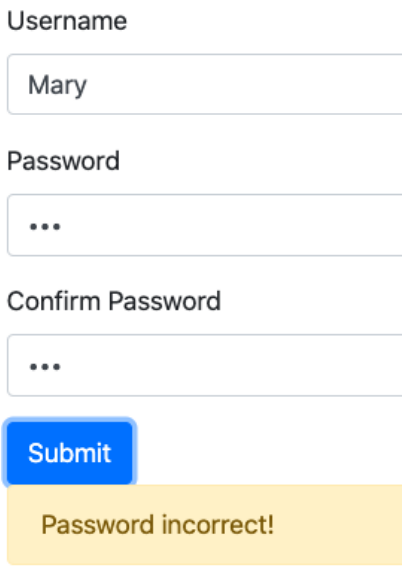
It also has a Vue instance which contains user information as shown below:

```
<script>
    // Add/Modify code below
    const app = Vue.createApp({
        data() {
            return {
                users: [{ "user": "Jack", "pwd": "abc" }, { "user": "Mary", "pwd": "def" },
                { "user": "John", "pwd": "123" }, { "user": "Cherry", "pwd": "456" }
                ]
            };
        }
    });
    const vm = app.mount('#app');
</script>
```

You are to implement form validation code using Vue that checks if the inputs entered by the user are valid.

Part A (*/)**

- Add/modify code in **`form.html`** so that it performs the **Form Input Validation** as shown below:

When the user does not enter one or more of the input fields  	When the user enters a name that does not exist in the `users` property specified in the Vue instance in **`form.html`** 
When the passwords do not match  	When the password is incorrect 

Note: Use Bootstrap's `<div class="alert alert-warning">` to display the error messages

Question 4 - Choose Background Color

Go to choose_color directory.
Given:

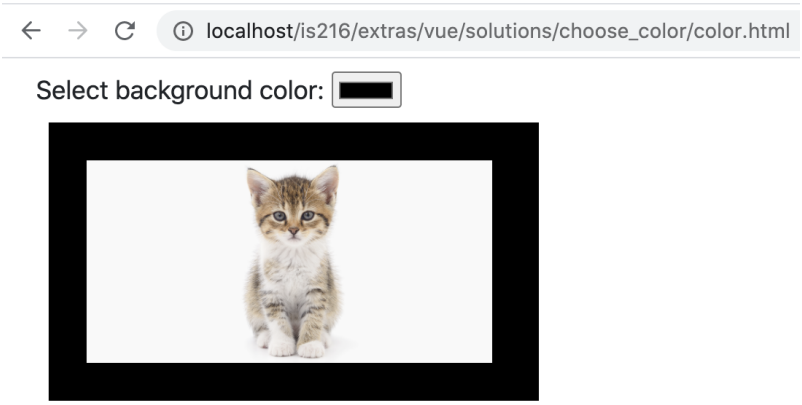
- **color.html**
- **img/kitten.png**

Complete the following Part A in `color.html` file.
You are to build a web page that allows the user to select the background color of a kitten image.

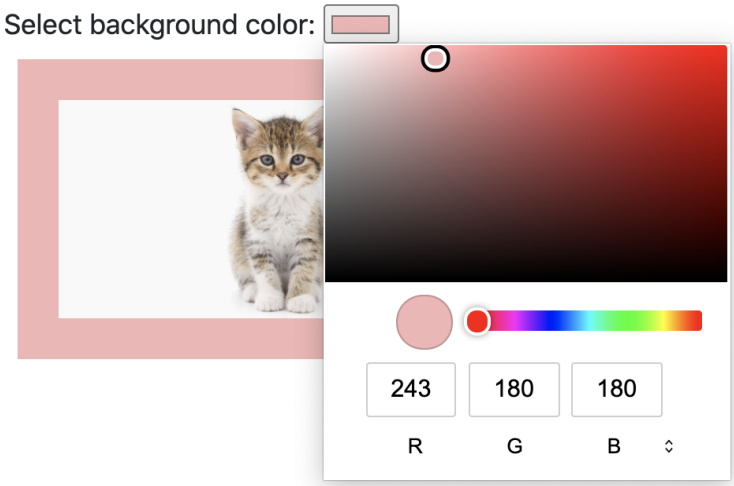
Part A (*)

- Create a Vue instance in **`color.html`** and add necessary Vue code so that it produces a web page as follows:

Initially, **`color.html`** produces the following web page



It contains an input field where the user can select a color



As soon as the user selects a color, the background color of the kitten image reactively changes accordingly

Note: for CSS styling, you may use any styles deemed necessary to produce the display as shown in the figures above. If necessary, use the internal CSS styles given in `color.html` and also explore the following Bootstrap classes:

- m-*
- p-*
- Img-fluid
- mx-auto

Question 5 - IS Curriculum

Go to curriculum directory.
Complete the following Parts A and B in `curriculum.html` file.

`curriculum.html` has a Vue instance in which its data is specified as below:

```
<script>
    Vue.createApp({
        // DO NOT MODIFY data
        data() {
            return {
                is_curriculum: {
                    "University Core": ["Statistics", "Computational Thinking", "Managing",
                        "Writing & Reasoning", "Internship",
                        "Economics & Society", "Technology & Society",
                        "Cultures of the Modern World", "Community Service",
                        "Ethics & Social / Corporate Responsibility",
                        "Big Questions", "Global Expoure"],
                    "IS Major Core": ["Information Systems & Innovation",
                        "Business Process Analysis and Solutioning",
                        "Enterprise Solution Development",
                        "Enterprise Solution Management", "Digital Business Technology and
Transformation",
                        "Introduction to Programming", "Software Project Management",
                        "Web Application Development I", "Web Application Development II",
                        "Data Management", "Interaction Design and Prototyping",
                        "Information Systems Project Experience"],
                    "Electives - Business Analytics": ["Analytics Foundation", "Data Mining &
Business Analytics",
                        "Geospatial Analytics & Applications",
                        "Managing Customer Relations with Analytics : Asian Insights",
                        "Social Analytics and Applicaitons", "Text Mining and Language Processing",
                        "Visual Analytics for Business Intelligence"],
                    "Electives - Digital Business Solutioning": ["Digital Transformation Strategy",
                        "Enterprise Analytics for Decision Support", "Enterprise Business
Solutions",
                        "Internet of Things: Technology and Applications",
                        "Managing Customer Relations with Analytics : Asian Insights",
                        "Systems for Intelligent Cities"],
                    "Electives - Financial Technology": ["Digital Banking Enterprise Architecture",
                        "Corporate Banking Technology & Smart Contracts",
                        "Digital Payments & Innovation",
                        "Financial Markets Processes & Technology",
                        "Retail Banking & Mobile Technology"]
                },
                selected_cat: 'University Core'
            };
        }
        // Add code here
    }).mount('#app')
</script>
```

You are to build a web page that shows the courses contained in Information Systems Major according to the course category selected by the user.

You are not allowed to modify the data section in the Vue instance. You are to add code in the place specified as `// Add code Here`

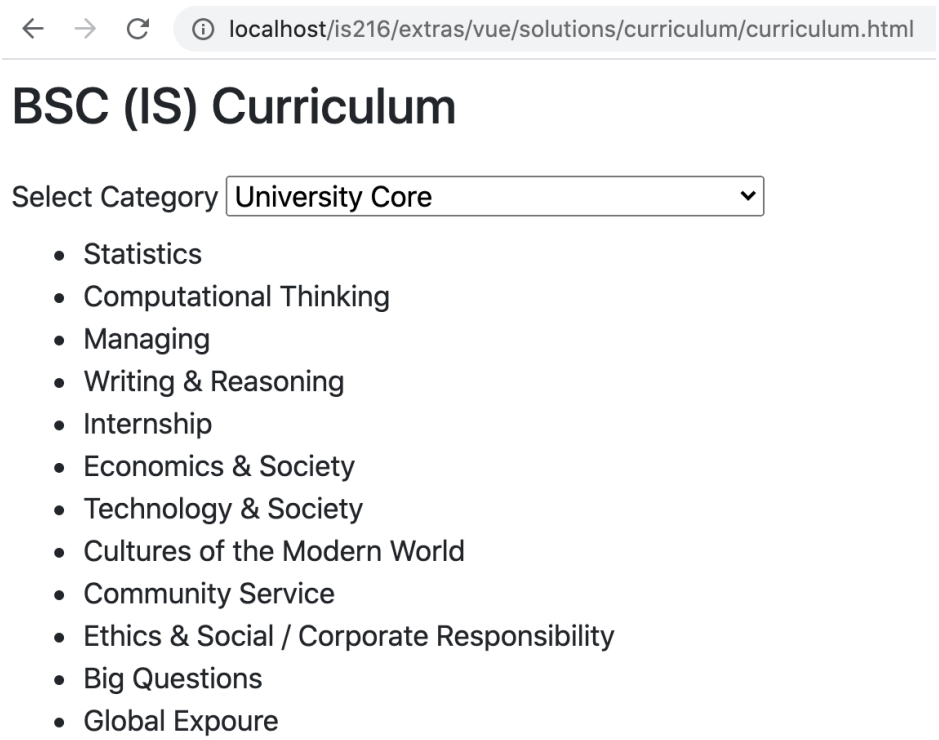
Part A (*)

- Add a computed property in the given Vue instance which returns the array of categories (i.e. `["University Core", "IS Major Core", …]`) contained in `is_curriculum`.
- Add another computed property in the given Vue instance which returns the array of courses corresponding to the selected category `selected_cat`. For example, if `selected_cat` is "University Core", the computed property returns `["Statistics", "Computational Thinking", "Managing"`, ...]

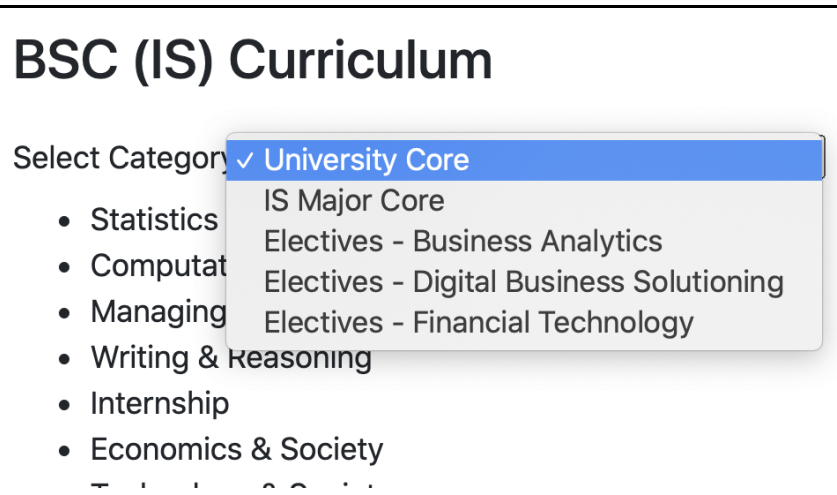
Part B (**)

- Use the two computed properties added in Part A and add code in the place asserted in **`curriculum.html`** so that it performs as follows:

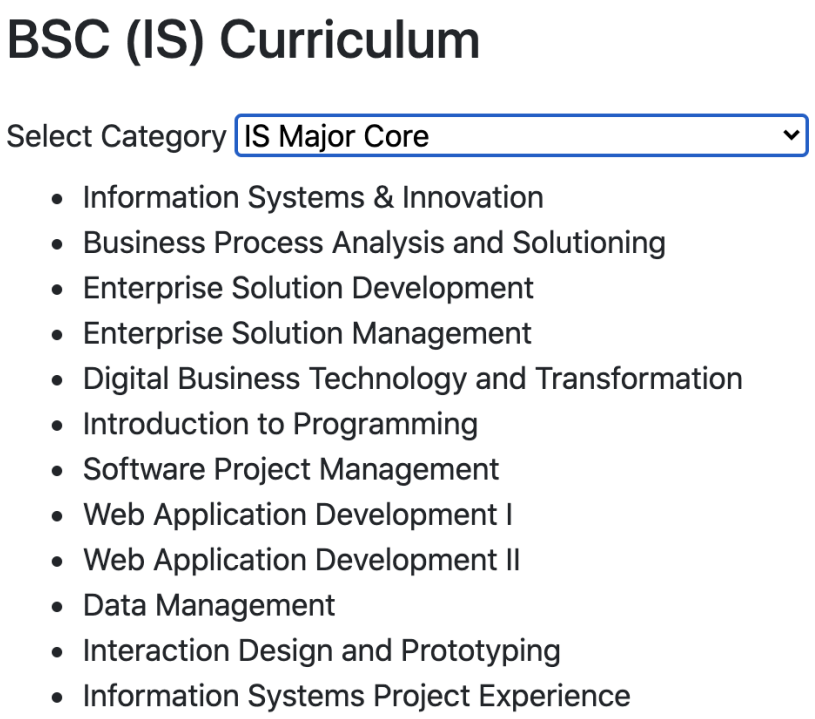
Initially, **`curriculum.html`** produces the following web page



It contains a dropdown menu where the user can select one of the categories



When the user selects “IS Major Core”



When the user selects an elective category

BSC (IS) Curriculum

Select Category Electives - Business Analytics

- Analytics Foundation
- Data Mining & Business Analytics
- Geospatial Analytics & Applications
- Managing Customer Relations with Analytics : Asian Insights
- Social Analytics and Applicaitons
- Text Mining and Language Processing
- Visual Analytics for Business Intelligence

Question 6 - Stars

Go to stars directory.
Complete the following Parts A and B in `stars.html` file.

`stars.html` has a Vue instance specified as below:

```
<script>
    // Add/Modify following code
    const vm = Vue.createApp({
        el: '#app',
        data: {
            num: 0,
            selected: ''
        }
    }).mount('#app');
</script>
```

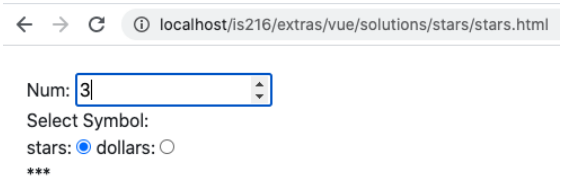
You are to build a web page that prints symbols (either stars or dollars) selected by the user, for a number of times specified by the user.

Part A (*)

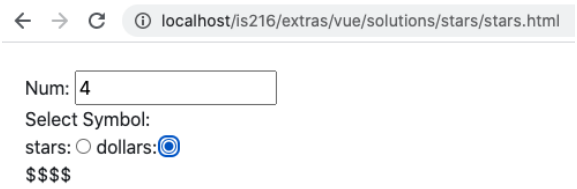
- Add code in `stars.html` file so that it shows the following web page initially:

- Bind num and selected properties of the Vue instance so that when the user enters a number (e.g. 3) and selects a radio button which corresponds to either *stars* symbol or *dollars* symbol, `stars.html` prints the corresponding symbol the number of times equal to the number entered by the user. Use appropriate Vue directives to achieve this functionality. See examples below:

When the user enters ‘3’ and selects ‘stars’



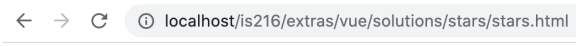
When the user enters ‘4’ and selects ‘dollars’



Part B (**)

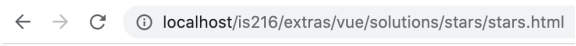
- Modify the Vue code in Part A such that when the user enters a number (e.g. 3) and selects a radio button which corresponds to either *stars* symbol or *dollars* symbol, `stars.html` produces the output as shown in the examples below:

When the user enters ‘3’ and select ‘stars’



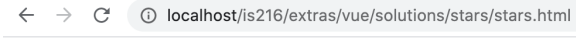
Num: 3
Select Symbol:
stars: dollars:
*
**
***

When the user enters ‘4’ and selects ‘dollars’



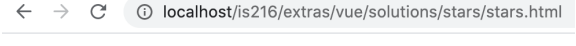
Num: 4
Select Symbol:
stars: dollars:
$
$$
$$$
$$$$

When the user enters ‘6’ and selects ‘stars’



Num: 6
Select Symbol:
stars: dollars:
*
**
***
****
*****
******

When the user enters ‘7’ and selects ‘dollars’



Num: 7
Select Symbol:
stars: dollars:
$
$$
$$$
$$$$
$$$$$
$$$$$$
$$$$$$$

Question 7 - Currency Converter

Go to the currency directory.
Complete the following Parts A and B in `currency.html` file.

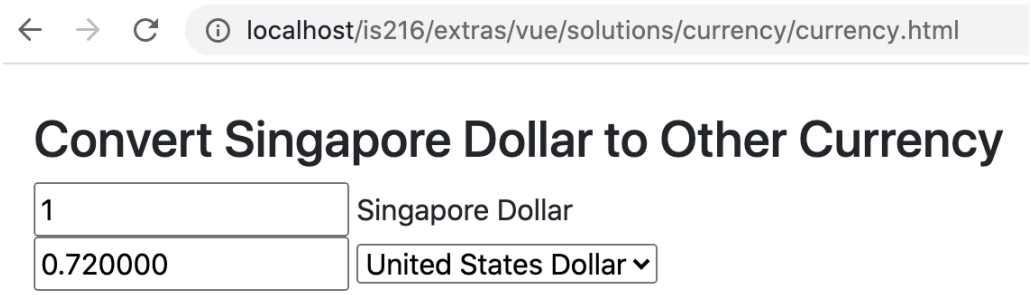
As shown below, **`currency.html`** has a Vue instance which is specified with `exchange_rates` data property:

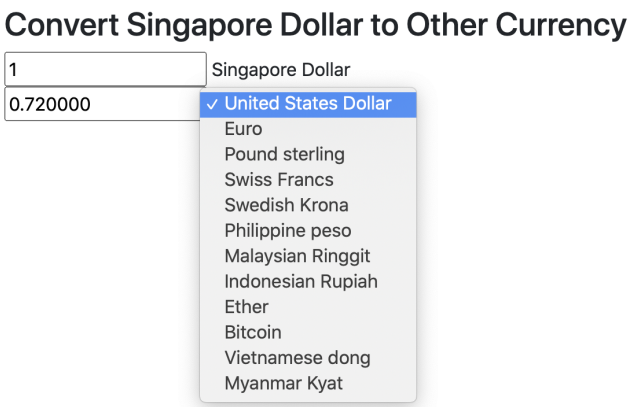
```
<script>
    // Add/Modify following code
    Vue.createApp({
        data() {
            return {
                exchange_rates: {
                    'United States Dollar': 0.72, 'Euro': 0.63, 'Pound sterling': 0.57,
                    'Swiss Francs': 0.67, 'Swedish Krona': 6.43, 'Phillippine peso': 35.62,
                    'Malaysian Ringgit': 3.08, 'Indonesian Rupiah': 10615.06, 'Ether': 0.0030,
                    'Bitcoin': 0.000077, 'Vietnamese dong': 16748.37, 'Myanmar Kyat': 1003.84
                }
            };
        }
    }).mount('#app')
</script>
```

You are to build a web page that converts a given amount of singapore dollars into its equivalent amount of the currency selected by the user.

Part A (*)

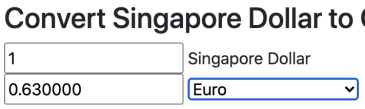
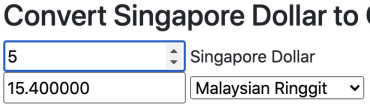
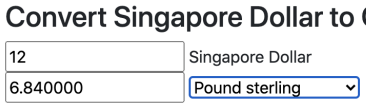
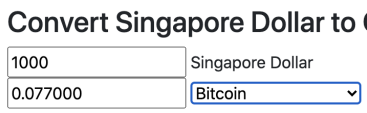
- Add code in `currency.html` file so that it shows the following web page initially:





Part B (**)

- Use appropriate Vue directives in HTML and add computed properties in the Vue instance so that `currency.html` reactively converts the amount of Singapore Dollar specified by the user to its equivalent amount of the currency selected by the user. The conversion must be reactive, i.e., as soon as the user makes a change, the web page must react to it and perform the computation. See examples below:

When the user selects 'Euro':  	When the user enters 5 for Singapore Dollar and selects 'Malaysian Ringgit': 
When the user enters 12 for Singapore Dollar and selects 'Pound sterling':  	When the user changes the selection to 'Phillippine peso': 
When the user enters 1000 for Singapore Dollar and selects 'Bitcoin':  	When the user changes the selection to 'Ether': 

- Note that the exchange rate results are shown in 6 decimal places.
- Hint: explore `Object.keys()`, `toFixed()`

Question 8 - Measurement Converter

Go to converter directory.
Complete the following Parts A and B in `converter.html` file.

As shown below, in **`converter.html`**, a Vue instance is specified with a set of data properties, which can be used to convert from one measure to another:

```
<script>
    const app = Vue.createApp({
        data() {
            return {
                Length: {
                    "Inch2centimetre": 2.54, "Inch2Yard": 0.0277778, "Inch2Mile": 0.000015783,
                    "centimetre2Inch": 0.393701, "centimetre2Yard": 0.0109361, "centimetre2Mile":
0.0000062137,
                    "Yard2Inch": 36, "Yard2centimetre": 91.44, "Yard2Mile": 0.000568182
                },
                Volume: { "litre2gallon": 0.219969, "gallon2litre": 4.54609 },
                Digital: { "Byte2Bit": 8, "Bit2Byte": 0.125 },
                Mass: {
                    "Ounce2Gram": 28.3495, "Ounce2Pound": 0.0625,
                    "Gram2Ounce": 0.035274, "Gram2Pound": 0.00220462,
                    "Pound2Ounce": 16, "Pound2Gram": 453.592
                },
                measurements: ["Length", "Volume", "Digital", "Mass"],
                selected_measurement: 'Length',
                unit1: 'Inch',
                unit2: 'centimetre',
                val1: 1
            };
        }

        // Add Code Here

    });
    const vm = app.mount('#app')
</script>
```

For example, in Length property, Inch to centimeter conversion is given as 2.54. That is, 1 inch is equivalent to 2.54 centimeters. In Volume property, gallon to litre conversion is given as 4.54609. That is 1 gallon is equivalent to 4.54609 litres.

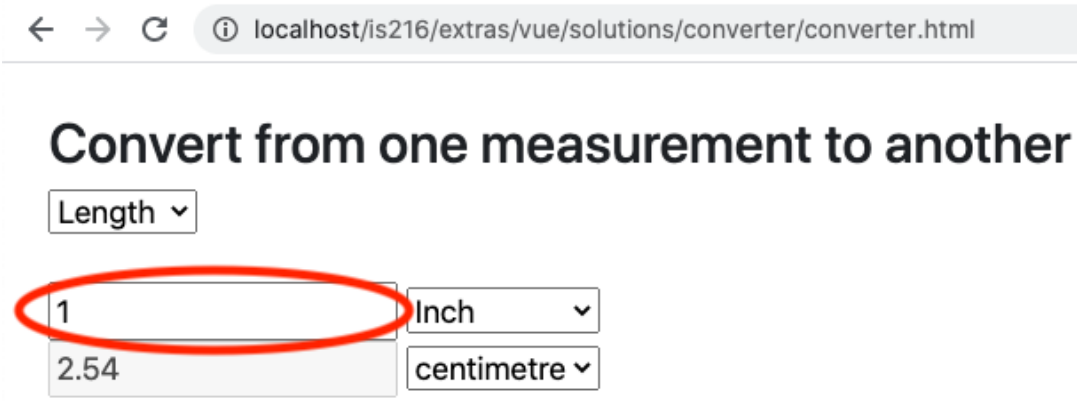
You are to build a web page that converts a given value of one measurement to its equivalent value of another measurement selected by the user.

You are not allowed to modify the data section in the Vue instance. You are to add code in the place specified as `// Add Code Here`

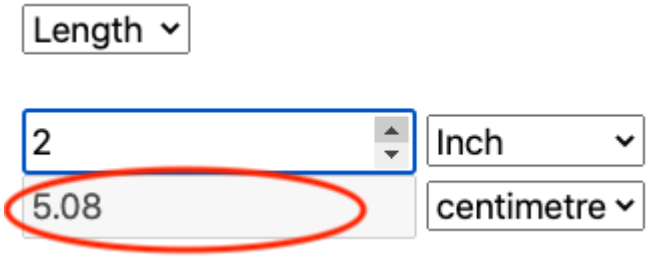
Part A (**)

- Add code in `converter.html` file so that it shows the following web page initially:

It has one input field where the user can provide the value of a measurement she/he wants to convert

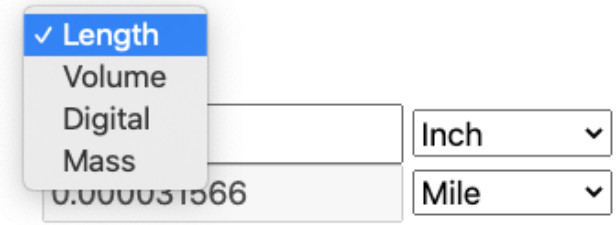


It has another field which shows the result of conversion:

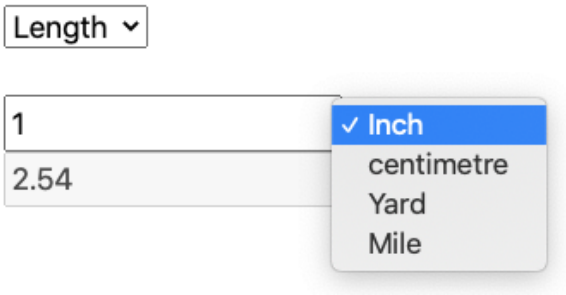


It has three dropdown menus as shown below. **The content of the dropdown menus must come from the data** given in the Vue instance. For example, “Length”, “Volume”, “Inch”, “Mile”, etc. Hard coding is not allowed.

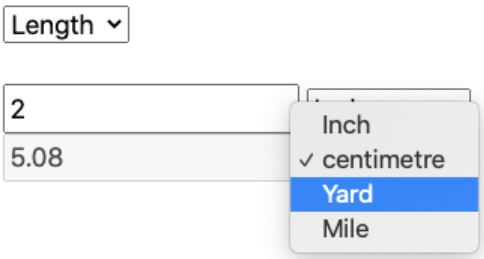
(a) Dropdown menu for selecting the measurement type:



(b) Dropdown menu for selecting the measurement:

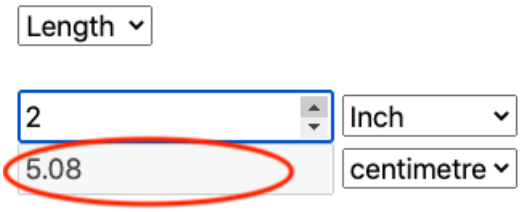
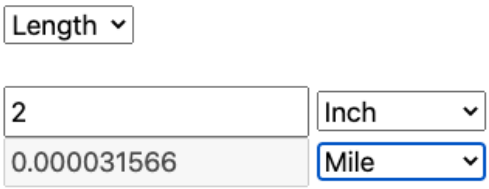
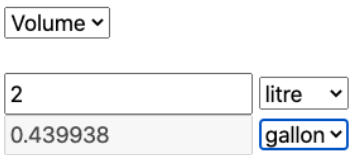
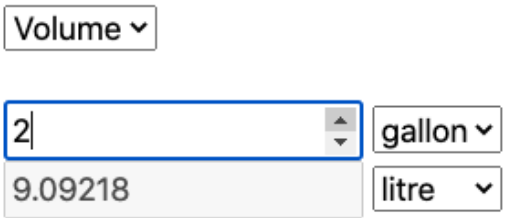
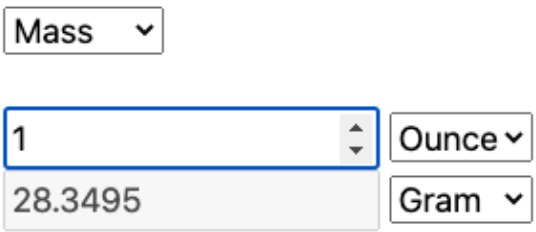


(c) Dropdown menu for selecting the measurement:



Part B (***)

- Use appropriate Vue directives in HTML and add computed properties in the Vue instance so that `converter.html` **reactively** converts a given value of one measurement to its equivalent value of another measurement selected by the user. The conversion must be reactive, i.e., as soon as the user makes a change, the web page must react to it and perform the computation. See examples below:

When the user selects 'Inch' for one measurement and 'centimetre' for another measurement, and enters the value '2': 	When the user then changes the measurement to 'Yard': 
When the user then changes the measurement to 'Mile' : 	When the user changes the measurement type to 'Volume' and selects 'litre' and 'gallon': 
When the user enters the value '4' for 'litre' measurement: 	When the user selects 'gallon' and 'litre' and enters the value '2': 
When the user changes the measurement type to 'Mass': 	When the user selects 'Ounce' and 'Pound' and enters the value '3': 

- Hint: explore `in`, `Object.keys()`, `split()`, `includes`

Question 9 - Farm

Go to `farm` directory.
Complete the following Parts A and B in `farm.html` file.

Suppose you are a farmer, and you start with zero animals in your farm. Every day, there are new animals on sale at the animal market. You can only buy one on any given day. We can express this choice with radio buttons!

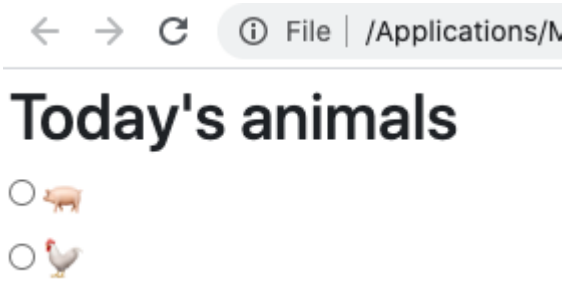
You are given an **animals** array in Vue and an **animal** property that will contain our choice for the day, and a farm array (initially empty) that will represent our hoarding:

```
data:{
        animals: ['🐖', '🐓', '🐄', '🐑'],
        animal: undefined,
        farm: []
    }
```

Emojis are used to represent the animals because they are fun. They are obtained from http://emojipedia.org/; simply look for animals and copy-paste.

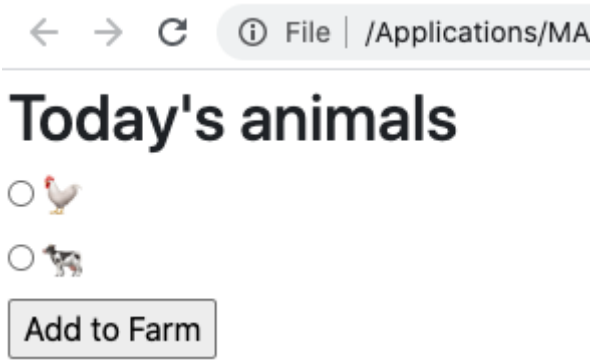
Part A (*)

- Add code in **`farm.html`** so that it displays a list of two animals randomly selected from **animals** property. Hint: use appropriate Vue directives to bind the properties and the given computed property **i**



Part B (**)

- Add a button in **`farm.html`** as shown below:



- Write code so that when an animal is selected and the ‘Add to Farm’ button is clicked, the selected animal is added to the **farm** property and shows the following result:

When ‘rooster’ is selected and button is clicked: Today's animals Add to Farm Your farm is composed by	When ‘pig’ is selected and button is clicked: Today's animals Add to Farm Your farm is composed by
When more ‘pig’ are selected: Today's animals Add to Farm Your farm is composed by	

Question 10 - Cat Food

Go to catfood directory.
Complete the following Parts A, B, and C in `catfood.html` file.

Given:

- **catfood.html**
- **img/cat.jpg**

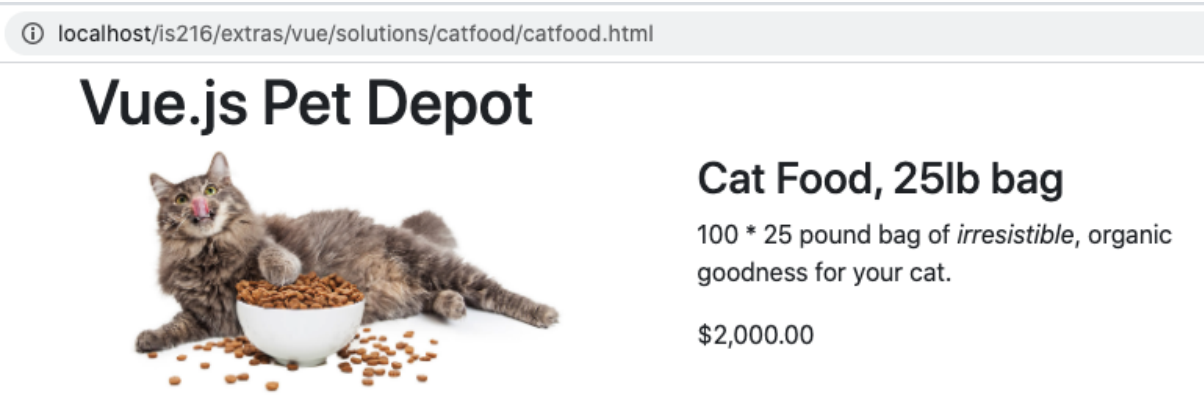
`catfood.html` contains a Vue instance which is defined as follows:

```
<script>
    // Add/Modify following code
    Vue.createApp({
        data() {
            return {sitename: "Vue.js Pet Depot",
            product: {
                id: 1234,
                title: "Cat Food, 25lb bag",
                description: "100 * 25 pound bag of <em>irresistible</em>," + " organic goodness for
your cat.",
                price: 200000,
                image: "img/cat.jpg",
            },
            count: 0};
        }
    }).mount('#app')

</script>
```

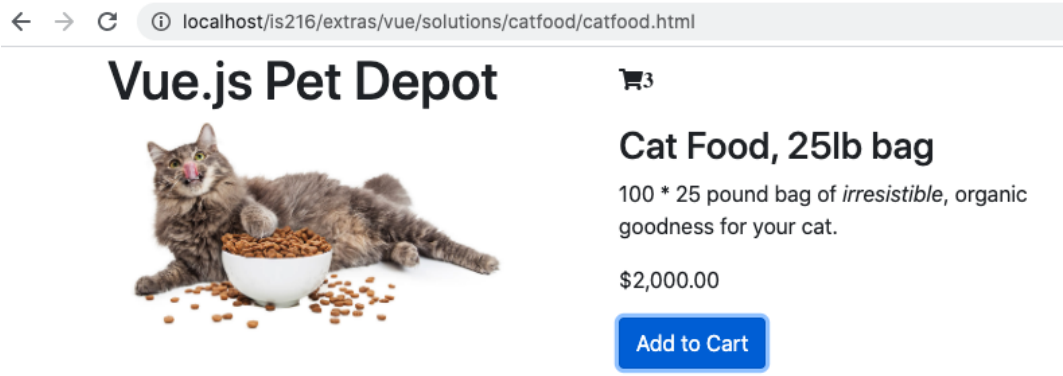
Part A (*/)**

- Display **sitename** in header2 (<h2>) tag using {{ }}
- Bind product's **image** property using v-bind directive to <img> tag so that the image is displayed as shown in the figure below.
- Also, bind product's **title** and **description** using appropriate Vue directives so that the information is displayed as shown in the figure below.
- Add a computed property called **formattedPrice** in the Vue instance, which formats the product's price value. It performs the following formatting:
    - It checks that the price is an integer. If it is not an integer, it returns an empty string
    - It converts the price value to a decimal and formats the last two digits of the price as decimal values (e.g. 200000 => 2000.00)
    - It adds commas every three places of the digits (e.g. 2000.00 => 2,000.00)
    - It adds '$' prefix and returns the result (e.g. 2000.00 => $2,000.00)
- Bind **formattedPrice** using an appropriate Vue directive so that the price is displayed as shown in the figure below.



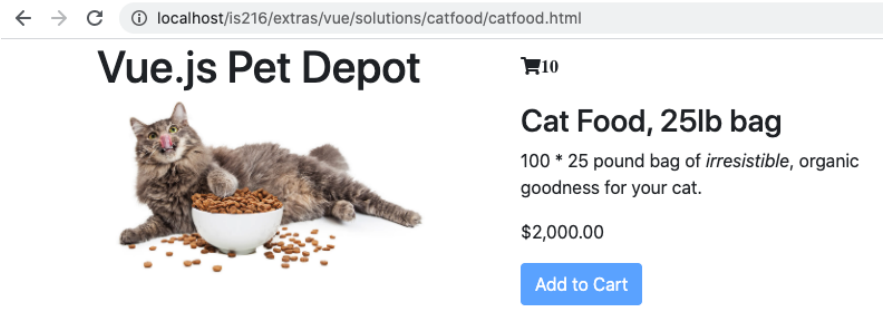
Part B (**)

- Add “Add to Cart” button and add code such that when the button is clicked, the value of the Vue instance’s **count** property is increased by 1
- Add code so that the web page shows a shopping cart icon together with the count value, as shown in the figure below.
- Note: use this code to display the shopping cart glyphicon: `<i class="fas fa-shopping-cart"></i>`



Part C (**)

- Assume that the inventory of Cat Food items is 10. Add code so that when the item count becomes 10, the “Add to Cart” button is disabled



Question 11 Colourful words (*)

#CSS #VueJS
Resource: nil

Create a HTML page

1. Display an input text field ‘Sentence’ on the first line.
2. Display the value of the field ‘Sentence' on the second line.
    a. When the user changes the text field’s value, the second line should be updated automatically.
    b. The first word is red in color, second word is orange, third is green. Fourth word onwards cycle through the colours in the same order of red, orange, green.
    c. Words at the even position are underlined.
    d. Note that the spaces are NOT underlined.

You may assume the words are separated by a single space.

Example 1

Sentence We love Bootstrap! We love Vue.js!

We love Bootstrap! We love Vue.js!

Example 2

Sentence World peace

World peace

Example 3

Sentence 1 2 3 4 5 6 7 8 9 11

1 2 3 4 5 6 7 8 9 11

Question 12 Flip coin (*)

#bootstrap #vue
Resource: Folder ‘flipcoin’

Start	After flipping twice
Flip  Flipped 0 times	Flip  Flipped 2 times

1.1. Flip many coins (*)

Start with 3 coins, randomly head or tails.

Number of coins 3 Flip

Flipped 0 times

After clicking ‘Flip” twice:

When the user changes the number of coins, display and flip the required number of coins; i.e. the number of flips is incremented by the new number of coins.

Minimum 1 coin

Question 13 Temperature (*)

#ajax #bootstrap #vue
Resource: nil

Create the following web page using https://data.gov.sg/dataset/realtime-weather-readings API to view air temperature across Singapore.

- Display the average of all temperatures recorded at the various stations in 2 decimal places..
- Display the date/time of retrieving the temperature data.
- Click button ‘Refresh’ to retrieve the temperature data again.

Singapore temperature

Average: 31.38

19/05/2020, 07:33:19 Refresh

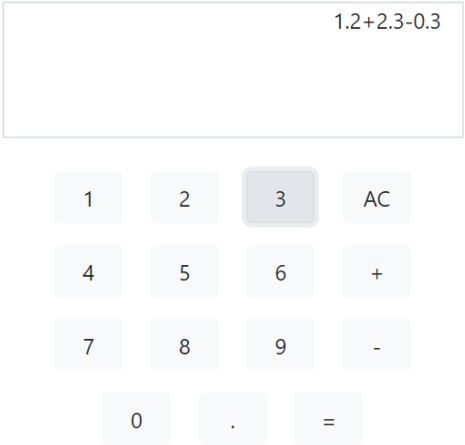
Station	Temperature
Banyan Road	32.2
Clementi Road	32.4
East Coast Parkway	29.7
Nanyang Avenue	32.6
Old Choa Chu Kang Road	33.7
Pulau Ubin	27.7
Scotts Road	31.4
Sembawang Road	28.1
Tuas South Avenue 3	32.7
West Coast Highway	31.4
Woodlands Avenue 9	30.2
Woodlands Road	34.5

Question 14 Calculator

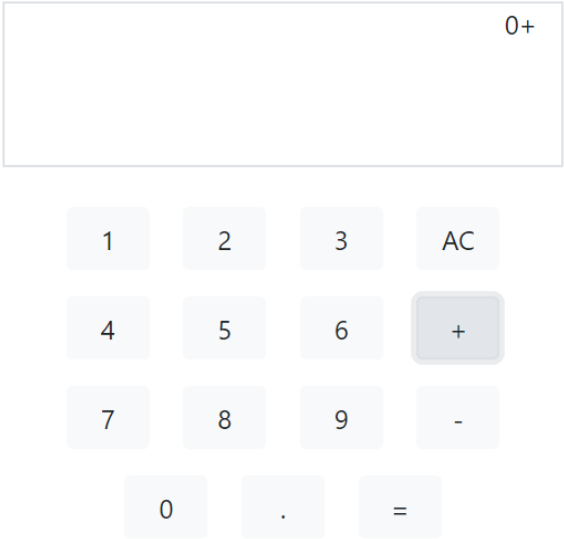
#bootstrap #vue
Resource: nil

Part 1 - Add/Minus (**)

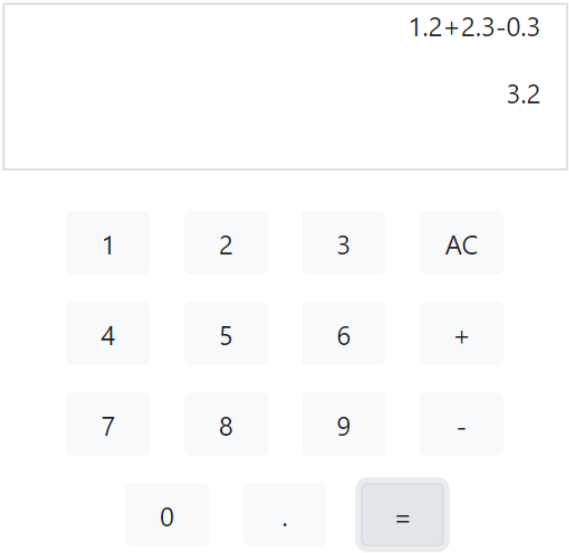
1. Click on the buttons to enter an expression.



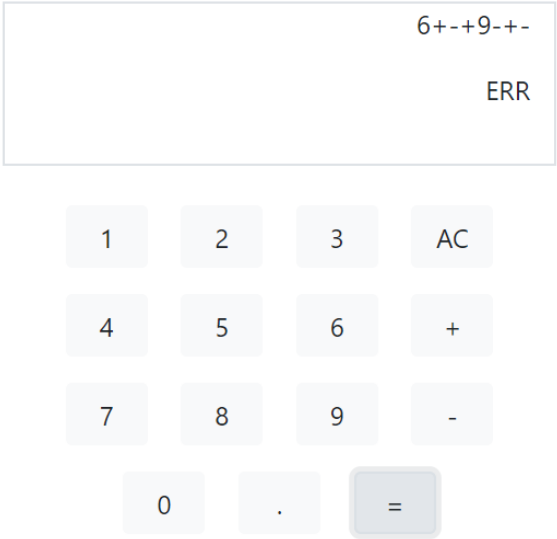
   a. If the first button clicked is '+' or '-', add 0 as the first operand to the expression automatically



2. When ‘=’ is clicked, show the answer.



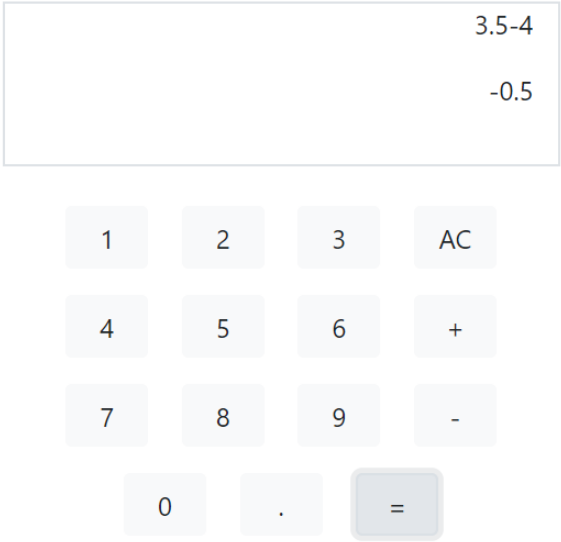
3. If the expression is invalid, display ‘ERR’ as answer



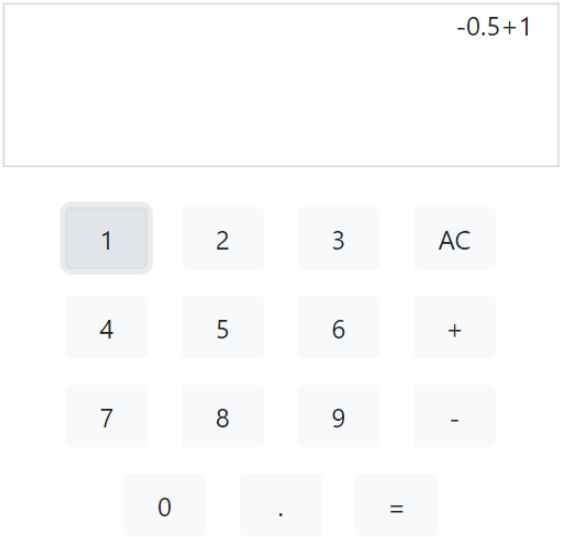
4. If you don’t know how to compute the answer, just use 12.5 as the answer and try the subsequent parts.

5. After having an answer (valid or invalid),
    a. Click ‘=’ shows the same answer.
    b. Clicking any other button is the start of a new expression.
6. If ‘AC’ is clicked, clear everything and restart.

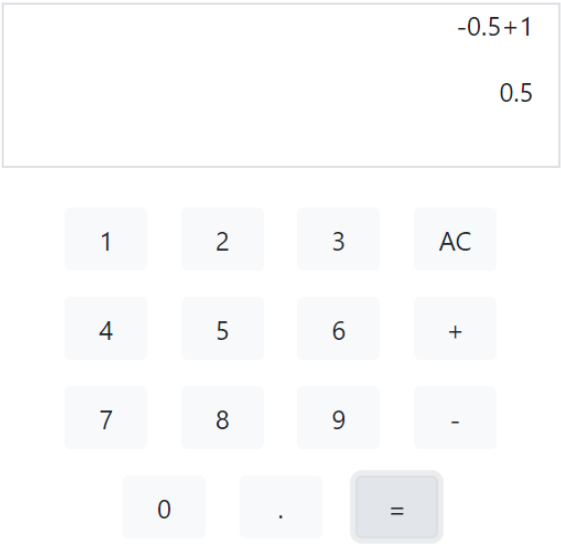
Part 2 - Friendly (*)



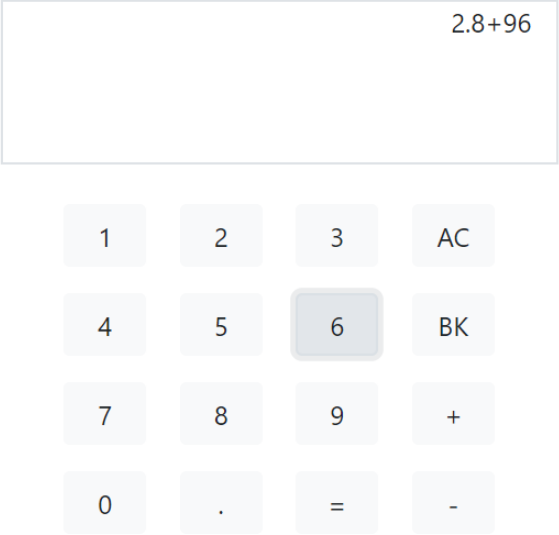
After having a **valid** answer, clicking '+' or '-', the answer will be used as first operand in a new expression



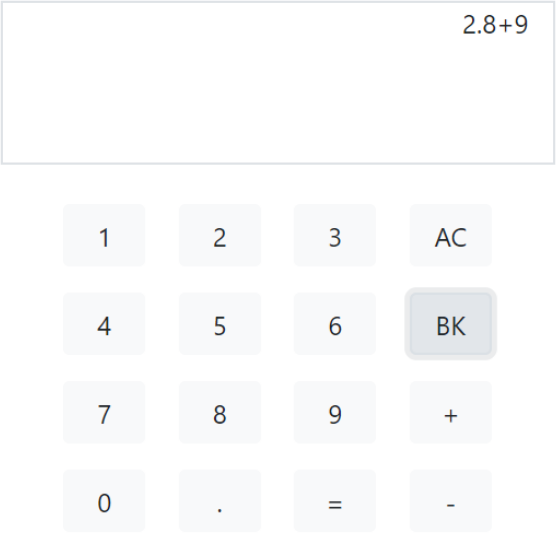
Click '='



Part 3 - Backspace (*)



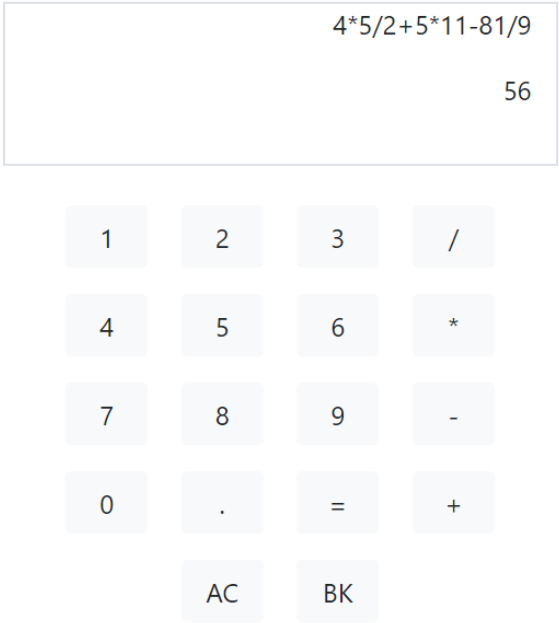
After ‘BK’ is clicked, clear the last character in the expression.



If there is an answer, clicking ‘BK’ clears everything and restarts.

Part 4 - Times and divide before add and minus (***)

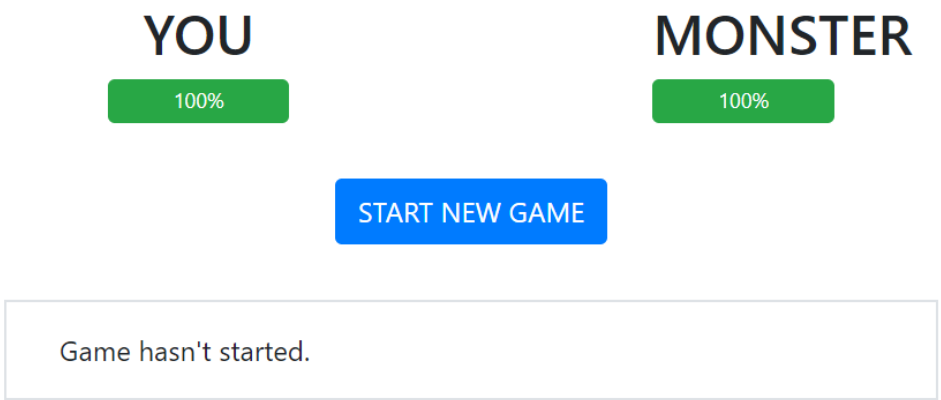
Add more logic to practise your javascript coding skills.

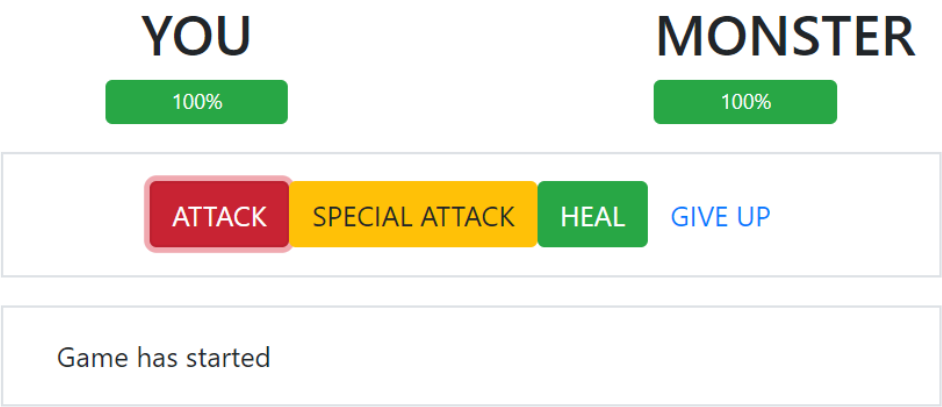


Question 15 Monster Slayer 2 (***)

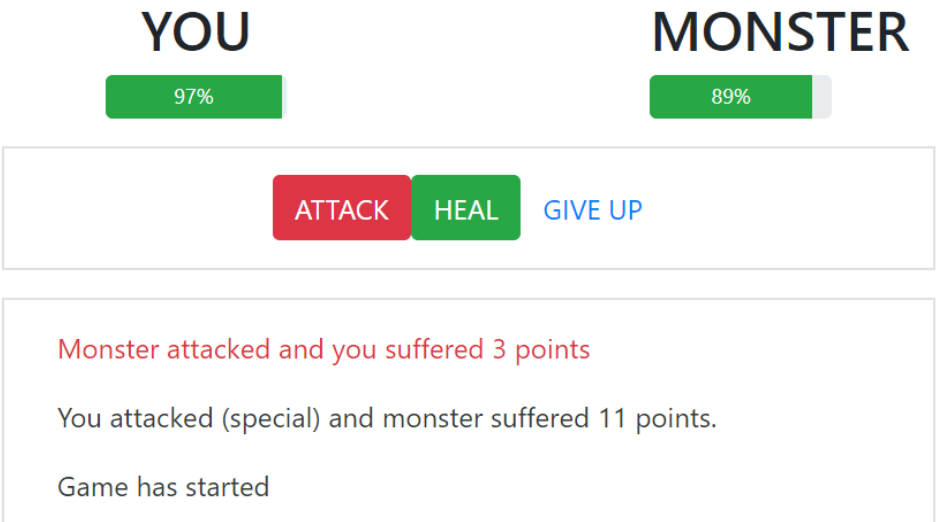
#bootstrap #vue
Resource: nil

Modified from 11-VueJS-PartI.pptx's "Putting all together" exercise.





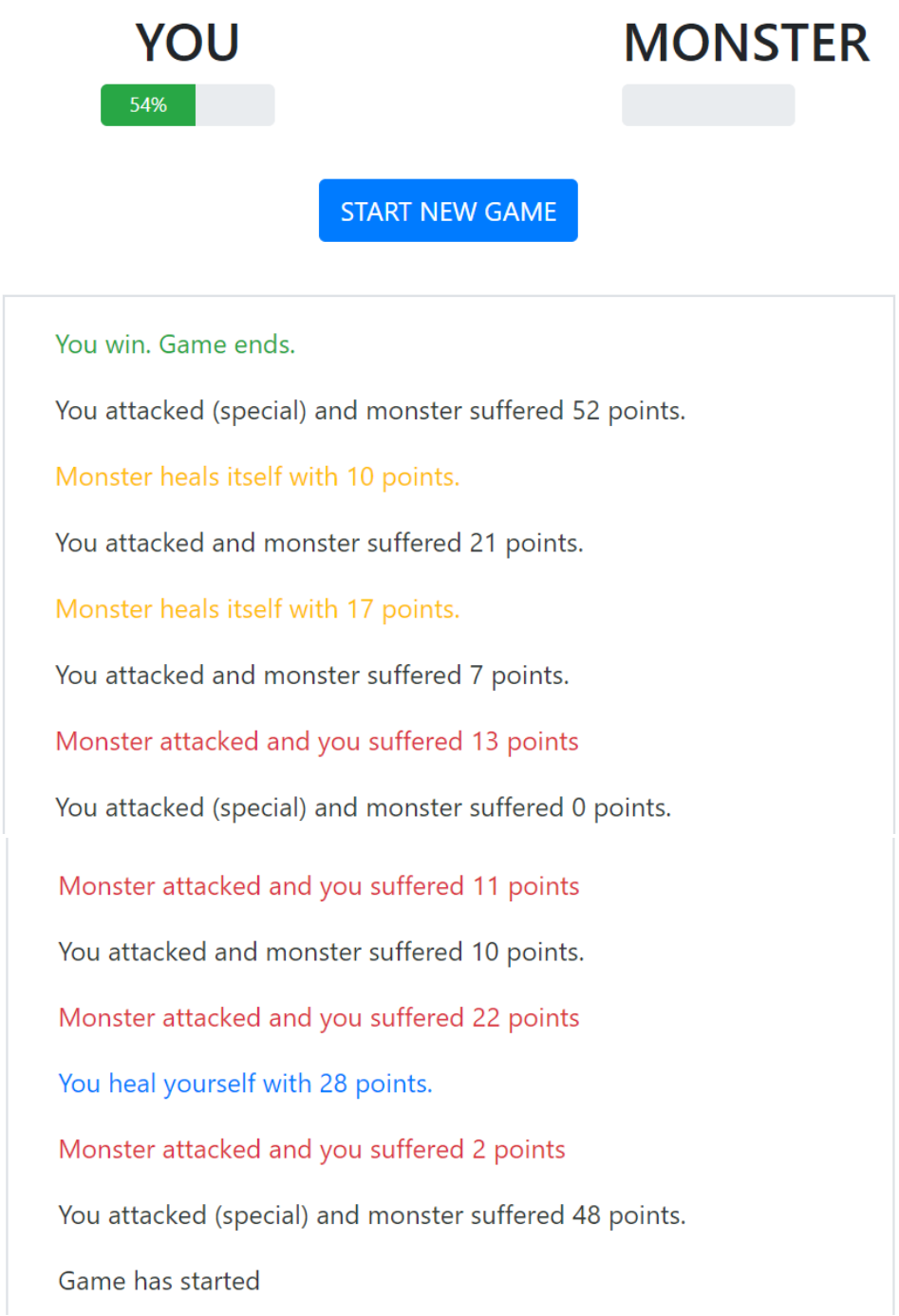
1. Attack
    a. Player attacks the monster by deducting a random integer value from its life. Maximum possible damage per attack is 30.
    b. Monster's move
        i. If the monster's life is more than 50, it attacks the player. Its attack has the same maximum possible damage of 30.
        ii. Otherwise, there is a 50% chance that it will heal its life a random integer value value.
2. Special attack is same as a normal attack with the following exceptions
    a. The player's randomly generated damage value is multiplied by 2.
    b. Once a special attack is used, there is a cooldown period of 2 rounds. The special attack button will be hidden; i.e. the player can only use attack or heal for subsequent 2 rounds.



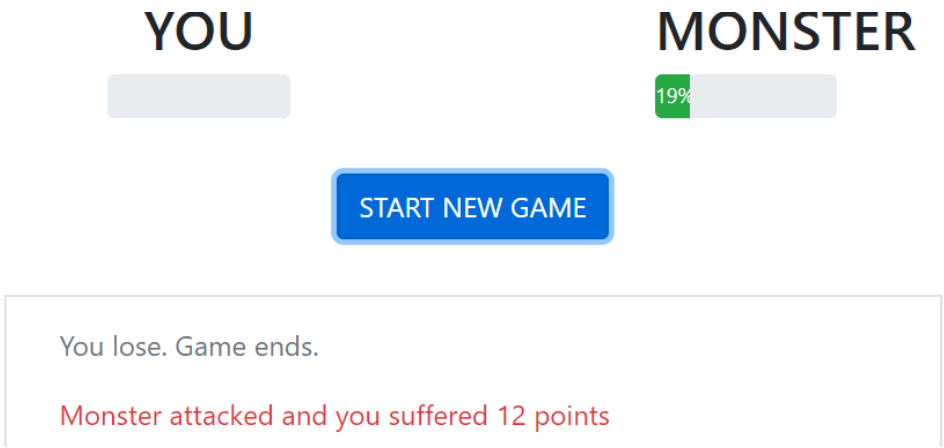
3. Heal
    a. Player heals her life at a random integer value.
    b. Monster's move.
4. Give up - End the game.

5. Status messages are displayed with the latest actions at the top.

Scenario: Win



Scenario: lose



Parts of the image are not shown to save space.

Scenario: Give up