Reactivity & Vue Lifecycle

Reactivity – it is nothing but programming paradigm which allows us to adjust changes in a declarative manner.

Vue3 composition API has a setup() function – it allows you to define the state in reactive and non-reactive manner and also you can have functions in setup itself.

Vue3 has introduced ref() and reactive() function to define reactive state and you can use it inside setup(){} only.

Ref() function is used on individual properties , you can use it for object.

let message = ref("Hello From Composition Reactive");

let data = ref(100);

Using ref() we are wrapping string value in an object.

Ref() uses object.defineProperty()

Original object and vue’s proxy object

Whenever you will use ref() function it will store that state inside proxy object of vue and it will have get() set(val) functions on that object.

Inside setup() function to access and update reactive state you have to use state.value always

What if I want to declare complete object as reactive

Let person = {

Name:’raj’,

Address:’Mumbai’

}

This object can not be made reactive using ref() function, you have to use reactive() function from vue to make your objects reactive

If you are using Vue2 options API the the lifecycle hooks are already available with your component but

If you are composition API then you need to import respective lifecycle hook depending on app need.

Options API Lifecycle hooks

beforeCreate()

created()

beforeMount()

mounted()

beforeUpdate()

updated()

beforeUnmount()

unmounted()

errorCaptured()

renderTracked()

renderTriggered()

**compositionAPI**

onBeforeMount()

onMounted()

onBeforeUpdate()

onUpdated()

onBeforeUnmount()

onUnmounted()

onErrorCaptured()

onRenderTracked()

onRenderTriggered()

beforeCreate() and created() lifecycle hooks from options API are not present in composition API reason of it is setup() method runs around these two so no need to have them explicitly.

4 phases

1. Creation – runs on components creation
2. Mounting – runs on component DOM mounting
3. Update – runs when reactive data / state is modified
4. Destruction – runs right before component is removed from DOM

.vue is a single file component because it is combination of <template> i.e html

<script.> and <style>

Component communication

Parent to child – can be done using props

Child to parent – by emmiting child component events

If you are using options API no need to import any lifecycle hook, you can use it directly.

In case composition API you need to import respective lifecycle hook and because of this it makes

Your project lightweight.

1. beforeCreate() – it initializes all reactive data and events. This hook does not have access to any component’s reactive data & events

this hook is useful if you want to make any api call and it is not necessary to assign the data

1. created() – here we can access component data and event. It is used for reading and writing reactive data
2. beforeMount() / onBeforeMount() – this hook is called before component DOM is rendered & mounted. This can used as a last step for making API call
3. mounted / onMounted – called right after first render of your component. Here you have direct access to DOM. We can have access to elements here, if it is options api then you can access elements using this.$el and if it is composition API then you can access element using ref
4. beforeUpdate/onBeforeUpdate – runs before data is changed and component is re-rendered. It is a place to update DOM manunally before any changes happens, e.g you can remove unwanted listeners. This is place where you can track number of edits to your component.
5. Updated/onUpdated – whenever change in data is there these hooks will be called
6. Destruction – beforeUnmount/ onBeforeUnmount – cleanup

Components / Component communication

To pass data from parent component to child component we have make use of props.

Change state of parent component using child component - listening events of child component

Reference link::

**Module 5 : Rendering Conditional Contents and Lists**

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

<https://v3.vuejs.org/guide/conditional.html>

Videos :

<https://www.vuemastery.com/courses/intro-to-vue-3/conditional-rendering-vue3>

<https://www.youtube.com/watch?v=6fIFXjpmWl8>

Using v-show Instead Of v-if

<https://v3.vuejs.org/guide/conditional.html#v-show>

<https://v3.vuejs.org/guide/conditional.html#v-if-vs-v-show>

Videos :

<https://www.youtube.com/watch?v=5OoyjkQ8YA4>

Rendering Lists of Data & Diving Deeper Into v-for

<https://v3.vuejs.org/guide/list.html>

Videos :

<https://www.vuemastery.com/courses/intro-to-vue-3/list-rendering-vue3>

<https://www.youtube.com/watch?v=pudi6M9eIng>

Removing List Items

<https://codesource.io/vue-js-snippets-delete-item-from-list/>

Lists & Keys

<https://v3.vuejs.org/guide/list.html>

Videos :

<https://www.youtube.com/watch?v=yaRKuxIHag4>

**Module 6 : Reactivity and Vue Lifecycle:**

An Introduction to Vue's Reactivity

<https://v3.vuejs.org/guide/reactivity.html#what-is-reactivity>

Videos :

<https://www.youtube.com/watch?v=HezB8UEU5Rg>

<https://www.danvega.dev/blog/2020/02/12/vue3-ref-vs-reactive/>

Proxy objects

<https://v3.vuejs.org/guide/reactivity.html#proxied-objects>

Videos :

<https://www.youtube.com/watch?v=HezB8UEU5Rg>

Proxy vs Original objects

<https://www.youtube.com/watch?v=HezB8UEU5Rg>

Videos :

<https://www.youtube.com/watch?v=HezB8UEU5Rg>

Watchers

<https://v3.vuejs.org/guide/reactivity.html#watchers>

<https://dev.to/ycmjason/recreating-vue-3-reactivity-api-roughly-1o6a>

<https://v3.vuejs.org/guide/reactivity-computed-watchers.html#computed-values>

<https://www.vuemastery.com/courses/vue-3-essentials/setup-and-reactive-references>

Working with Refs

<https://v3.vuejs.org/guide/reactivity-fundamentals.html#creating-standalone-reactive-values-as-refs>

Videos :

<https://www.youtube.com/watch?v=irpy9PX3jkU>

<https://www.youtube.com/watch?v=BFav9Z4lEXE>

Vue 3 App Lifecycle

<https://learnvue.co/2020/12/how-to-use-lifecycle-hooks-in-vue3/>

Videos :

<https://www.youtube.com/watch?v=ovGtB5eWsVI>

**Module 7 : Components & Component Communication:**

* Component Basics

<https://v3.vuejs.org/guide/component-basics.html#base-example>

Videos :

<https://www.vuemastery.com/courses/intro-to-vue-3/components-and-props-vue3>

* Single File Component

<https://v3.vuejs.org/guide/single-file-component.html>

Videos :

<https://www.youtube.com/watch?v=JKp76k_MmZc>

* Reusing Component

<https://v3.vuejs.org/guide/component-basics.html#reusing-components>

Videos :

<https://www.youtube.com/watch?v=gK8f_eRVurw>

* Organizing Components

<https://v3.vuejs.org/guide/component-basics.html#organizing-components>

* Passing data to Child components via Props

<https://v3.vuejs.org/guide/component-basics.html#passing-data-to-child-components-with-props>

Videos :

<https://www.youtube.com/watch?v=te-lFidcrcM>

* Listening to Child component events

<https://v3.vuejs.org/guide/component-basics.html#listening-to-child-components-events>

<https://v3.vuejs.org/guide/component-basics.html#emitting-a-value-with-an-event>

Videos :

<https://www.youtube.com/watch?v=__kSsJxllUw>

* Props Type & Validations

<https://v3.vuejs.org/guide/component-props.html#prop-types>

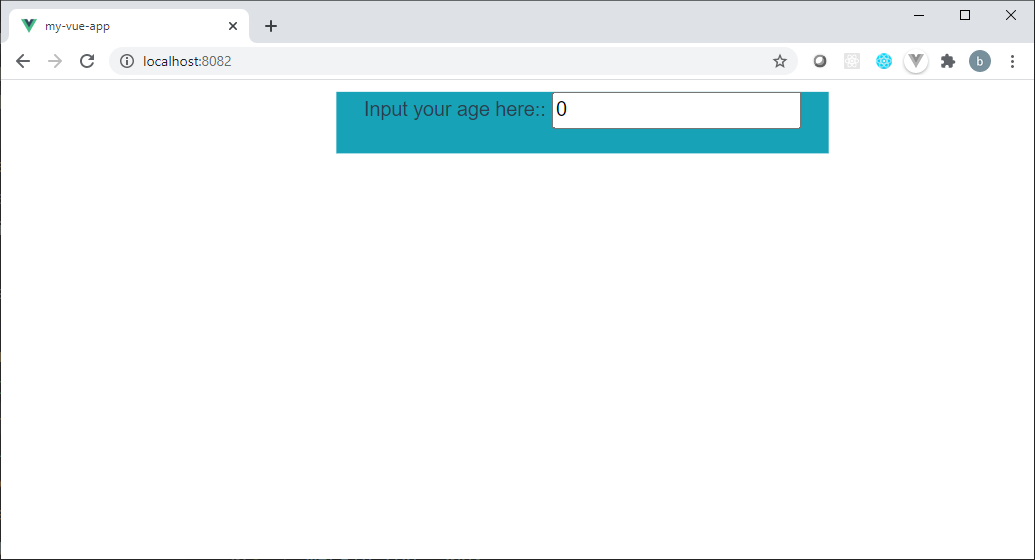
Videos :

<https://www.youtube.com/watch?v=gmgmRx6KsWA>

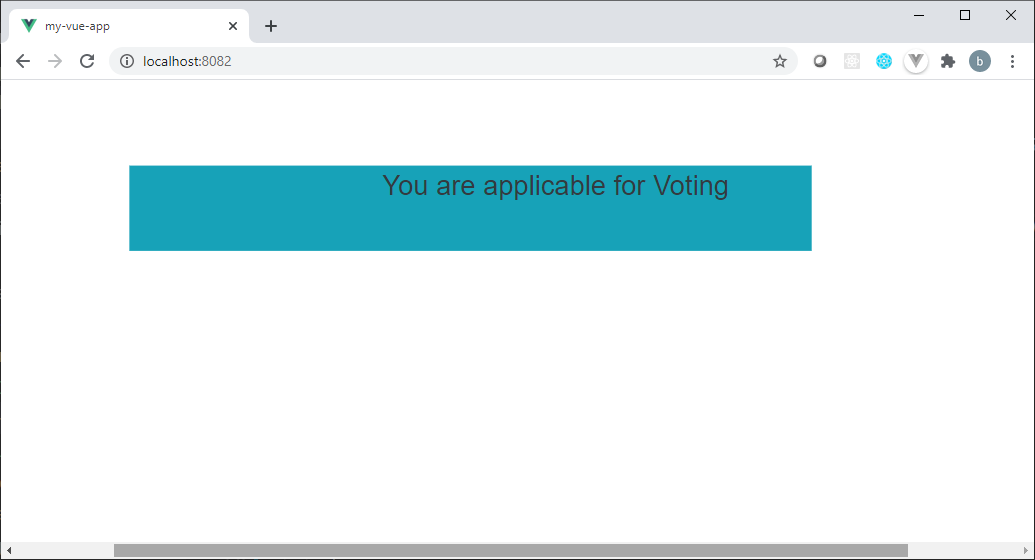
Assignments::

**Module 5** : **Rendering Conditional Contents and Lists**

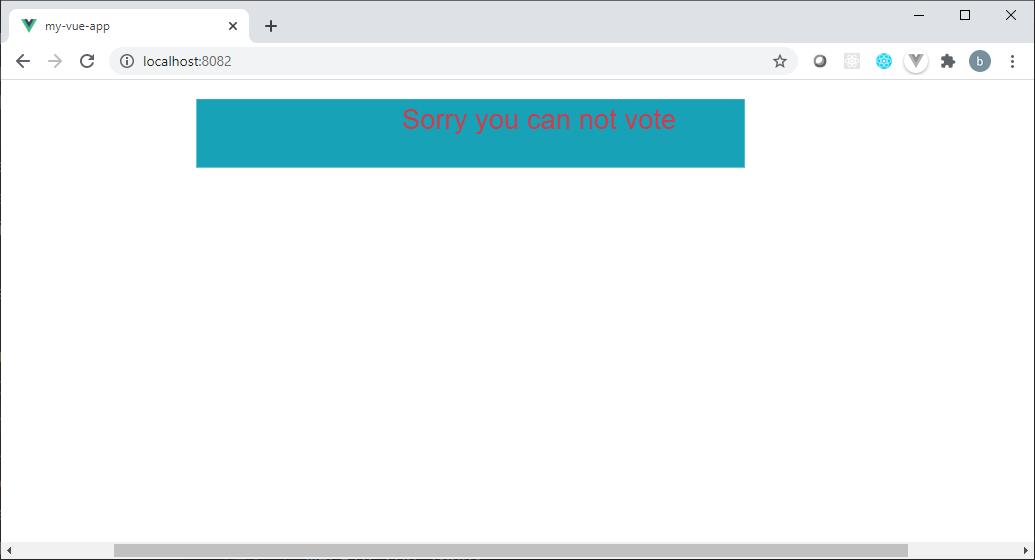
5.1 Create a Vue app that will generate below view to accept an age. Initialize data age in the vue 3 instance to 0 and check if age is 0 then input the age as given below.



if Entered age will be >=18 then generate below view.



else generate below view.



5.2 Create a vue app instance that will iterate below given object

personObject: {

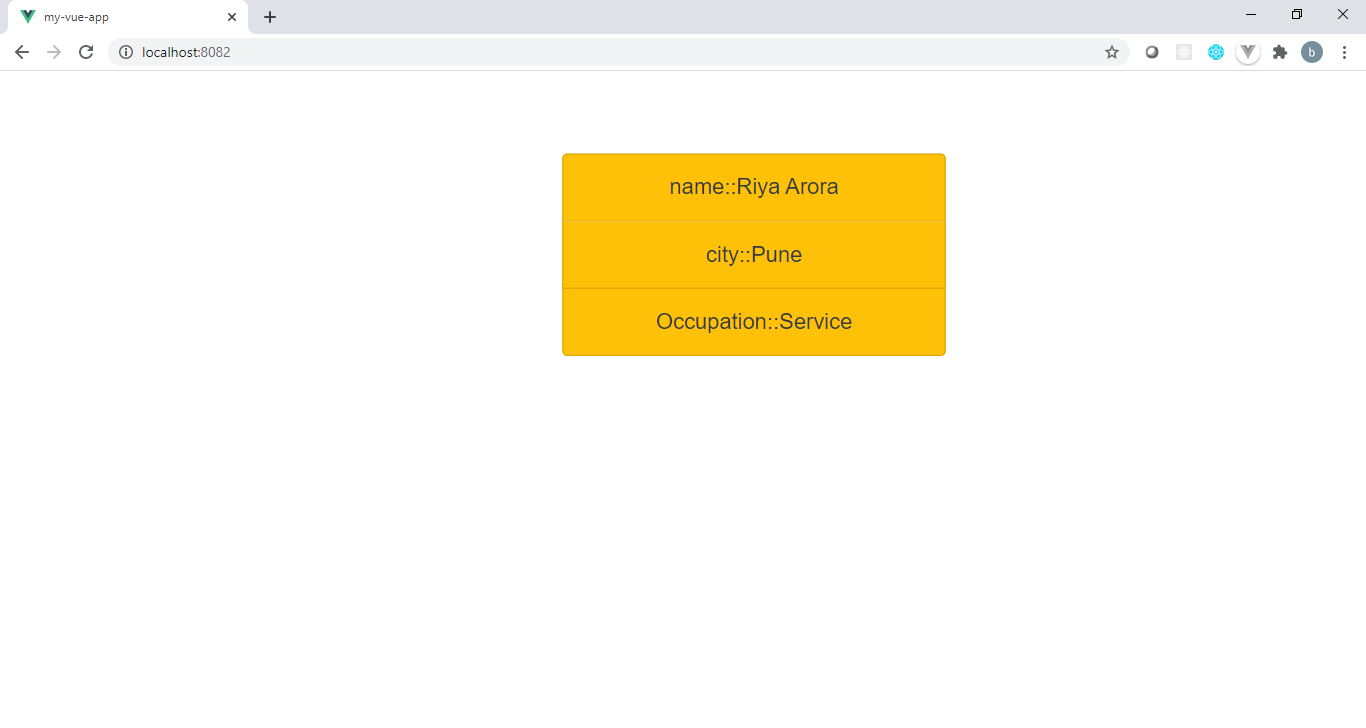
name: 'Riya Arora',

city: 'Pune',

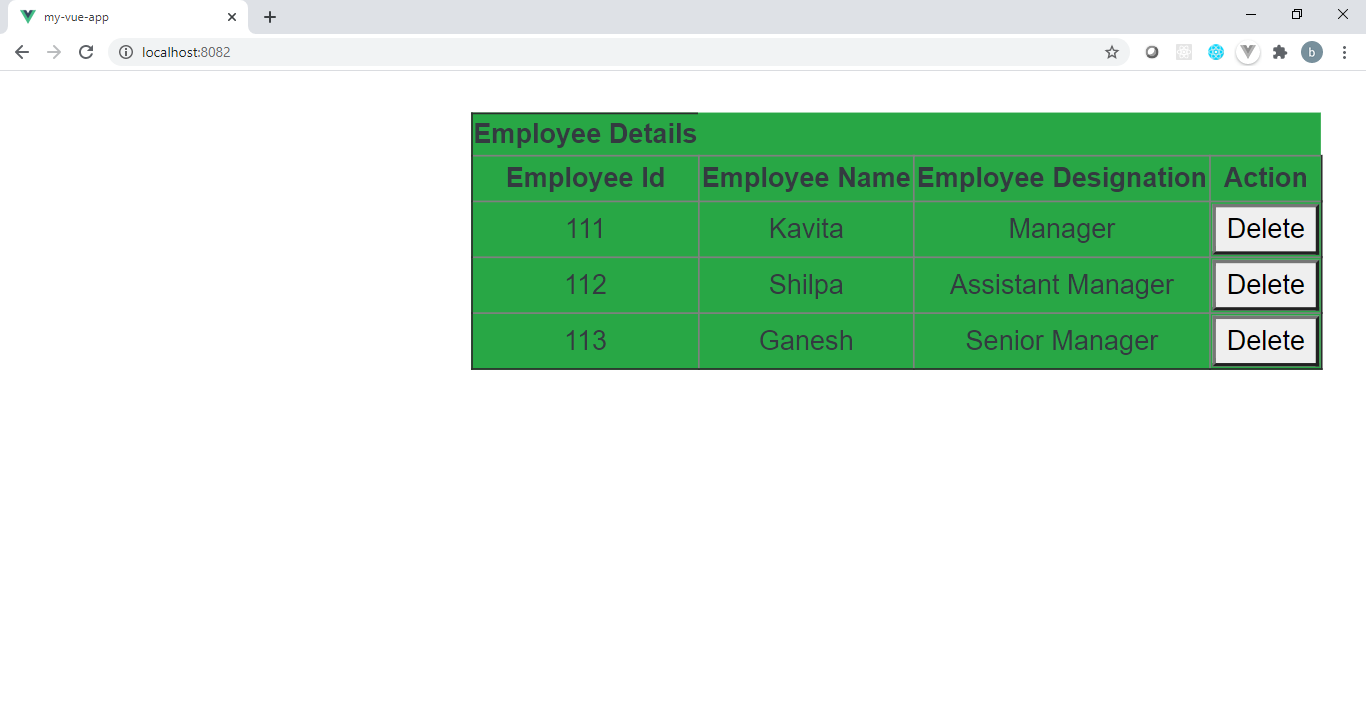
Occupation: 'Service'

}

and will generate a view as given below.

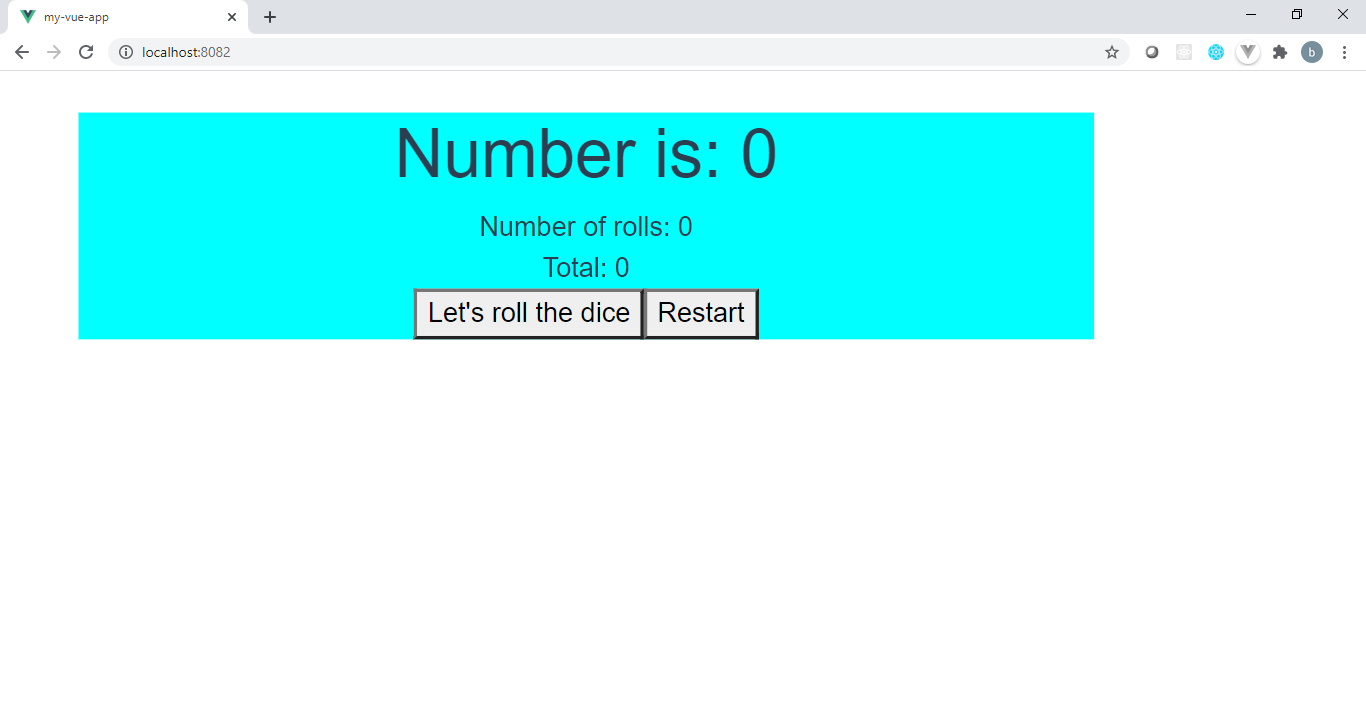


5.3 Create a Vue instance to display the list of employees with delete button on each list item as given below. On click of delete respective item should be removed from list.

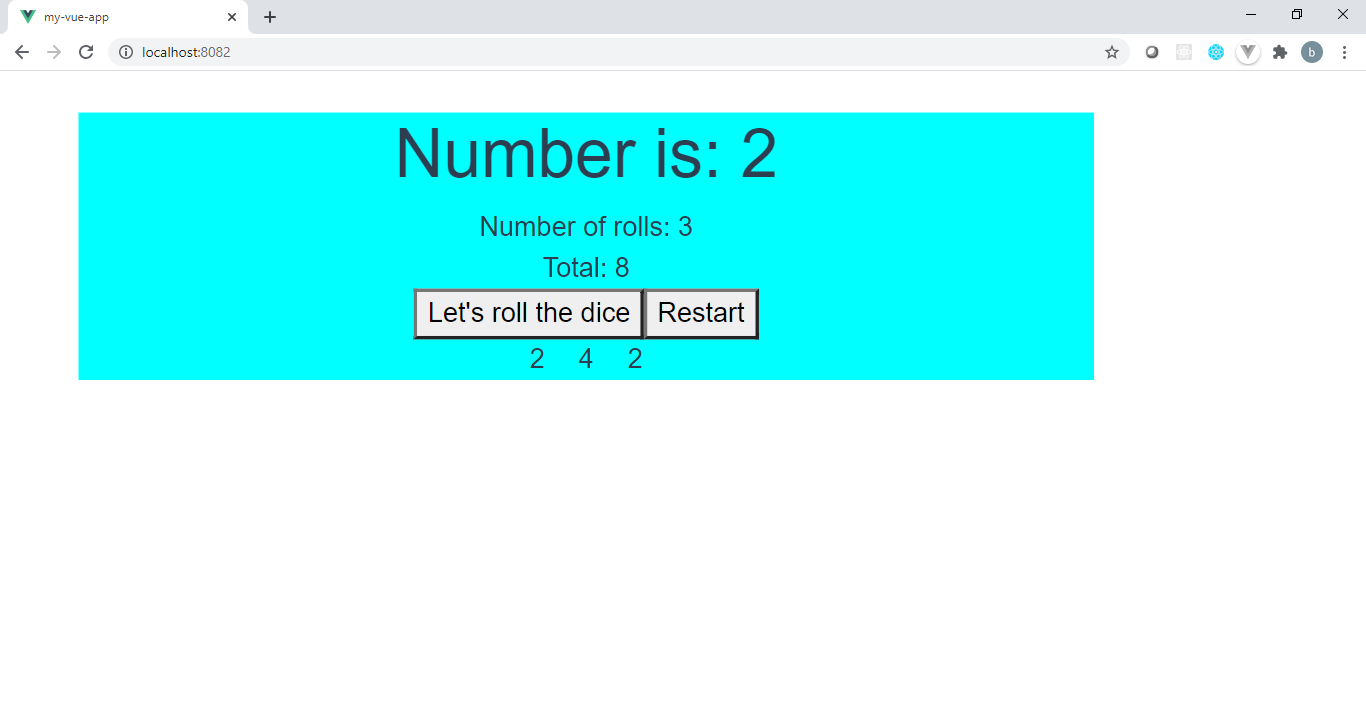


**Module 6 :** **Reactivity and Vue Lifecycle**

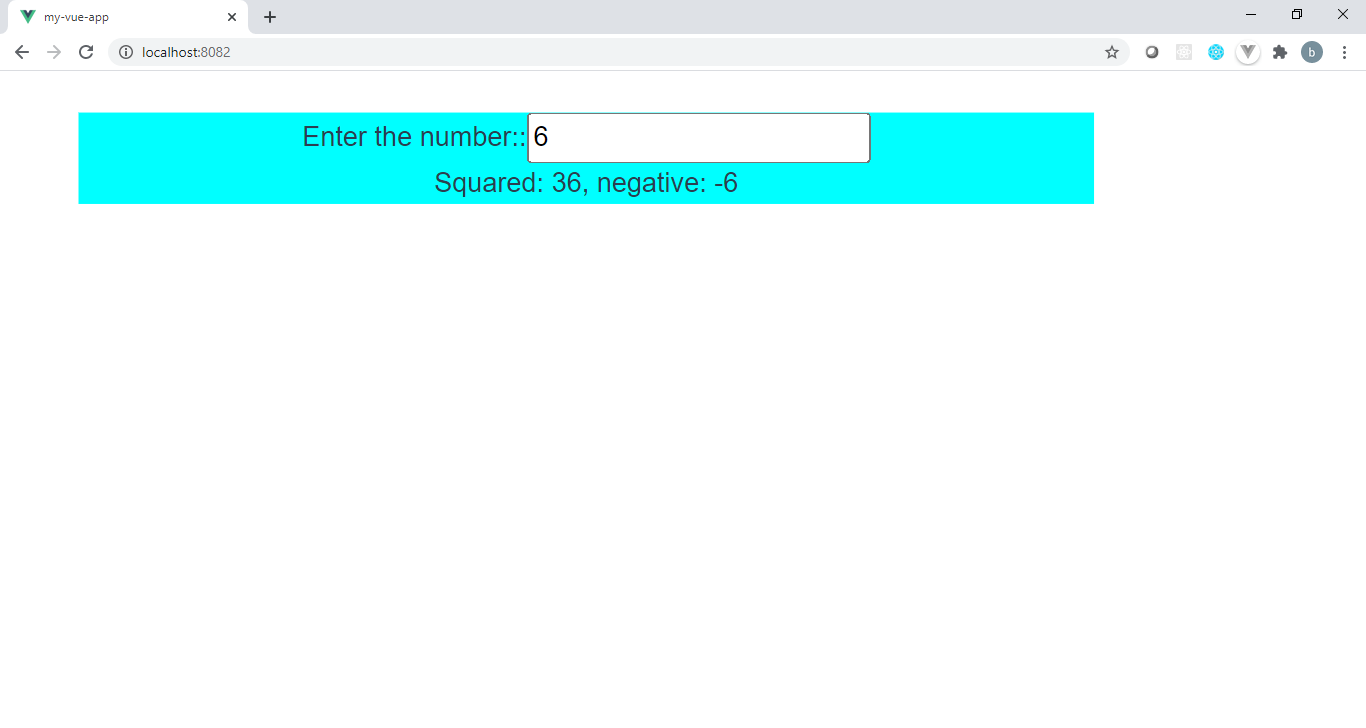
6.1 Implement Roll the Dice case study using composition API and reactivity from Vue 3. Create a “div” to show the result and a button to trigger the roll and generate a random number from 1 to 6.



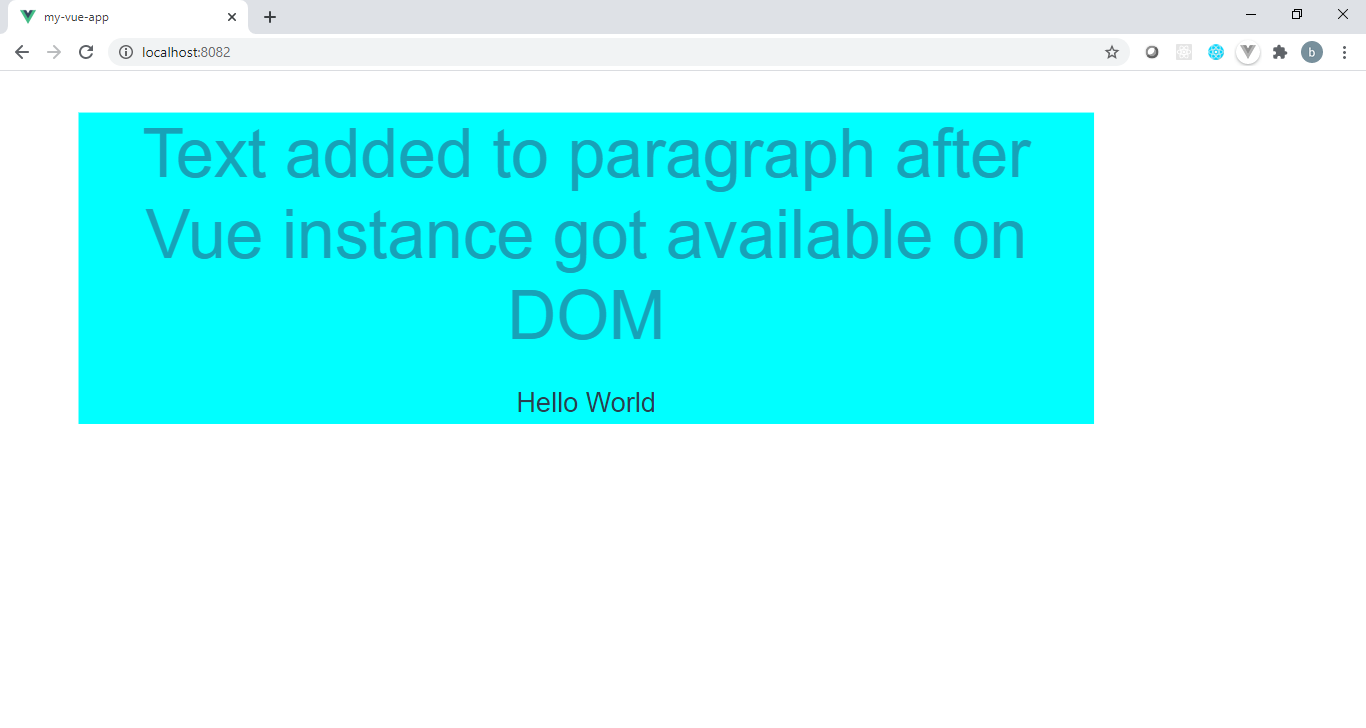
Once you start rolling the dice, view updates as given below.



6.2 Generate the below given UI using Vue composition API and compute the square and negative value of input number.

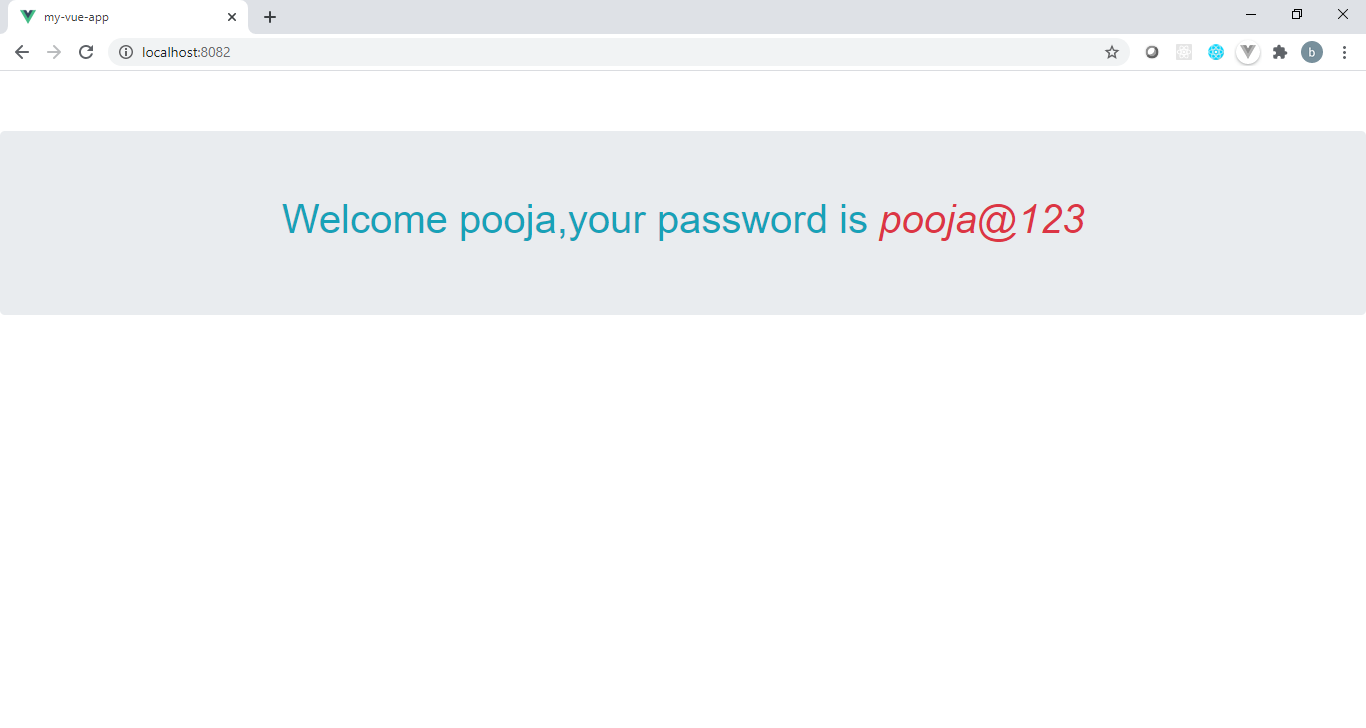


6.3 Create Vue instance using composition API that will display below output. Create a division with message HelloWorld and a paragraph with no value. Add value to paragraph once the vue instance get available on DOM as given below.



**Module 7 :** **Components & Component Communication**

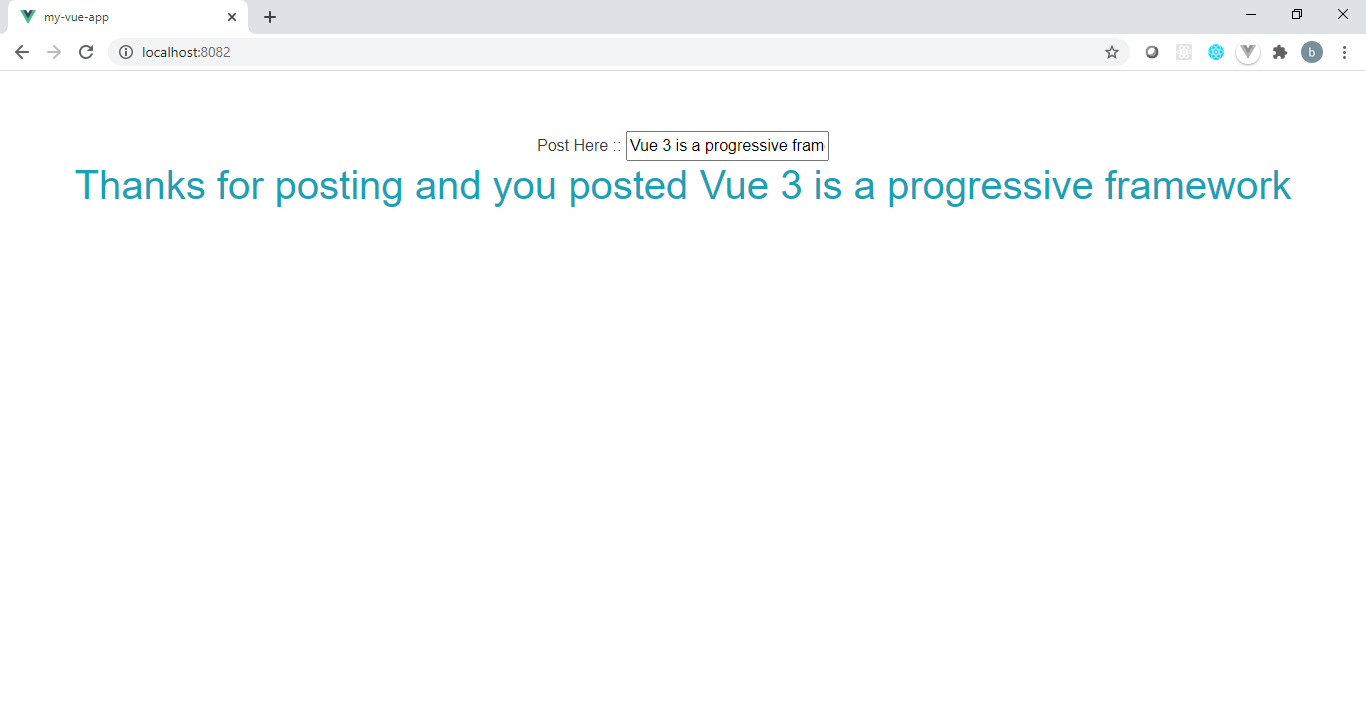
7.1 Create Vue components ProfileComp and AccountInfo. When ProfileComp will render AccountInfo it should pass username and password to AccountInfo and AccountInfo will generate below given view. Also take care of validation of username and password to type string.



7.2 Create a Vue component PostComponent with data originalPost of type string. If originalPost value will be null, it should generate below UI.



Once the post value will be entered, originalPost from this component should be updated to entered value and that will be passed to component BlogPost and it will generate below view.



7.3 Consider problem statement 7.2 and generate below given view using PostComponent



Enter the post here and pass it to BlogPost, if post is not null then BlogPost will get rendered and will generate below given view.



After editing the post in the above UI, everywhere the post value should be changed as given below

