Week 6 Graded Questions

Q1: In the below code snippet, which takes a number as input and stores it in localStorage so that on refreshing the page, the previous number stored should show up in the input area. Fill in code 1 and code 2.

index.html:

```
    <div id="app">
        My lucky Number is
        <input type="number" v-model=code1>
        <button v-on:click="addNumber">ADD</button>

    </div>
```

app.js:

```
const app = new Vue({
    el: '#app',
    data: {
      number: '',
      newNumber:''
    },
    mounted() {
      if (localStorage.number) {
        this.number = localStorage.number;
      }
    },
    methods:{
    addNumber() {

        code2     }
}
  });
```

A. code1: “number”
code 2: localStorage.number = this.number;

B. code1: “newNumber”
code 2: localStorage.number = this.newNumber;

C. code1: “newNumber”
code 2: localStorage.number = this.number;

D. code1: “number”
code 2: localStorage.number = this.newNumber;

Answer: A

Solution: To be able to see the previous number entered in the input box upon refresh, the number taken as input should be stored somewhere and fetched back when the application reloads.
In the above question, local storage is used. In the mounted(), the number from localStorage Is being placed in this.number. So, this is the number that needs to reflect in the input box. Thus, `<input type="number" v-model=number>.`
And when the Add button is clicked , the number in the input box should update the localStorage,with `localStorage.number = this.number;`

Q2: In order to ensure that items entered to the toDoList persist after screen refresh by adding it to localstorage, choose the code that can be placed in *code1* as marked below.

index.html:

```
<input  v-model="newData"
              placeholder="Enter your List"/>
      <button @click="addTodoList">Add</button>
```

app.js:

```
data:{
        title:"To Do List",
        newData:'',
```

```
        todoList:new Array(),
      },
 methods:{
        addTodoList(){
          this.todoList.push(this.newData)
          code1
        }
      }
```

A. localStorage.setItem(STORAGE_KEY,JSON.stringify(this.todoList))
B. localStorage.getItem(STORAGE_KEY,JSON.stringify(this.todoList))
C. localStorage.setItem(STORAGE_KEY,JSON.parse(this.todoList))
D. localStorage.getItem(STORAGE_KEY,JSON.parse(this.todoList))

Answer A

Solution: The key value pairs stored in localStorage are in string format. So when handling arrays and objects, JSON.stringify() can be used when placing data into the local storage with the setItem().
(https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage)

Q3: Consider the following markup index.html and JavaScript file app.js for an application,

index.html:

```
<div id="app">
      <named-slot v-slot:header="slotProp">
        Name: {{slotProp.user.name}}, District:
        {{slotProp.user.dist}}</named-slot
      >
</div>
<script src="app.js"></script>
```

app.js:

```
const namedSlot = {
  template: `
```

```
    <div>
        <slot name = "header" :user="currentUser"></slot>
    </div>
    `,
  data() {
    return {
      users: [
        {
          user_id: 1,
          name: 'Narendra',
          dist: 'Ballia',
        },
        {
          user_id: 2,
          name: 'Abhisek',
          dist: 'Delhi',
        },
      ],
      currentUser: null,
    }
  },
  props: ['current_user_id'],
  created() {
    current_user = sessionStorage.getItem('userId')
    current_user_id = current_user ? current_user : 2
    this.currentUser = this.users.find(
      (user) => user.user_id == current_user_id
    )
    sessionStorage.setItem('userId', current_user == 1 ? 2 : 1)
  },
}

const app = new Vue({
  el: '#app',
  components: {
    'named-slot': namedSlot,
  },
})
```

Suppose the application is running on port 8080 what will be rendered by the browser when the page index.html loads on the screen for the first time (without refreshing the page)?

A. Name: Narendra, District: Ballia
B. Name: Abhisek, District: Delhi
C. Nothing will be shown on the screen
D. None of the above

Answer: B

Solution: If the value is not stored in sessionStorage 2 will be stored in the current_user and if already stored then it will store 2 if its value is 1 and 2 if value stored is 2. So if you will run the application for first time then 2 will be stored in sessionStorage and information related to id 2 will be rendered so, Option B is correct.

Q4. Consider the app.js and index.html from the Question number 3.

What will be rendered by the browser after refreshing the page 2021 times?

A. Name: Narendra, District: Ballia
B. Name: Abhisek, District: Delhi
C. Nothing will be shown on the screen
D. None of the above

Answer: A

Solution: If the value is not stored in sessionStorage 2 will be stored in the current_user and if already stored then it will store 2 if its value is 1 and 2 if value stored is 2. So if you will refresh the page an odd number of times information related to id 1 will be rendered, Option A is correct.

Q5. Consider the following markup index.html and JavaScript file app.js for an application,

index.html:

```
<div id="app" v-bind:style="mode === 'dark' ? dark: normal">
      <button @click="changeMode('dark')">Dark</button>
      <button @click="changeMode('normal')">Normal</button>
```

```
      <div style="margin-top: 20px">Hello IITM</div>
    </div>
<script src="app.js"></script>
```

app.js:

```
const app = new Vue({
  el: '#app',
  data: {
    dark: {
      backgroundColor: 'black',
      color: 'white',
    },
    normal: {
      backgroundColor: 'red',
      color: 'white',
    },
    mode: null,
  },
  methods: {
    changeMode(mode) {
      sessionStorage.setItem('mode', mode)
      this.mode = mode
    },
  },
  created() {
    websiteMode = sessionStorage.getItem('mode')
    this.mode = websiteMode ? websiteMode : 'dark'
  },
})
```

Suppose the application is running on http://localhost:8080. If a user visit the page “index.html”, and clicks on the button normal, and then close the tab in which the application is open, and then again visit the website, what will be the background color of div with ID “app”?

A. black
B. red
C. white

D. None of the above

Answer: A

Solution: Because we are using session storage, once you close the session data stored inside the session storage is lost. And if you will again open the application, dark mode will be stored inside the session storage. So, option A is correct.

Q6. Consider the following markup index.html and JavaScript file app.js for an application,

index.html:

```
<div id="app" v-bind:style="mode === 'dark' ?dark: normal">
      <button @click="changeMode('dark')">Dark</button>
      <button @click="changeMode('normal')">Normal</button>
      <div style="margin-top: 20px">Hello IITM</div>
</div>
<script src="app.js"></script>
```

app.js

```
const app = new Vue({
  el: '#app',
  data: {
    dark: {
      backgroundColor: 'black',
      color: 'white',
    },
    normal: {
      backgroundColor: 'red',
      color: 'white',
    },
    mode: 'dark',
  },
  methods: {
    changeMode(mode) {
      localStorage.setItem('mode', mode)
```

```
      this.mode = mode
    },
  },
  created() {
    websiteMode = localStorage.getItem('mode')
    if (websiteMode) {
      this.mode = localStorage.getItem('mode')
    }
  },
})
```

Suppose the application is running on http://localhost:8080. If a user visits the page “index.html”, and click on the button normal, and then close the tab in which the application is open, and then again visit the website, what will be the background color of div with ID “app”?

A. black
B. red
C. white
D. None of the above

Answer: B

Solution: Because localStoreage is being used even if you will close the tab, the data stored inside the local storage will persist. And mode stored inside the local storage is normal so, background color will be red.

Q7: Consider the following markup index.html and JavaScript file app.js for an application.

index.html:

```
<div id="cards">
      <p>{{ currentPlayer}}</p>
      <select name="cards" @change="validate($event)">
        <option value="black-Diamond">black Diamond</option>
        <option value="red-Diamond">red Diamond</option>
        <option value="black-Heart">black Heart</option>
        <option value="red-Heart">red Heart</option>
      </select>
```

```
      <p v-if="error">{{error}}</p>
    <script src="app.js"></script>
</div>
```

app.js:

```
const app = new Vue({
  el: '#cards',
  data: {
    players: ['player2', 'player1', 'player3', 'player4'],
    cards: {
      player1: null,
      player2: null,
      player3: null,
      player4: null,
    },
    currentPlayer: null,
    previousPlayer: null,
    error: null,
  },
  methods: {
    validate() {
      card = event.target.value.split('-')[0]
      this.cards[this.currentPlayer] = card
      this.error = null
      if (this.cards[this.currentPlayer] ==
this.cards[this.previousPlayer]) {
        this.error = 'You cannot pick this card'
        this.cards[this.currentPlayer] = null
      } else {
        this.previousPlayer = this.currentPlayer
        this.currentPlayer = this.players.pop()
      }
    },
  },

  created() {
```

```
    this.currentPlayer = this.players.pop()
    this.previousPlayer = this.players[0]
  },
})
```

Suppose the application is running on http://localhost:8080. If the user selects the options in the order “Red Diamond”, “Red Heart”, “Black Heart” and “Black Diamond”. Which of the following is true?

A. The application will show a message with text “You cannot pick this card”.
B. The application will not show any message with text “You cannot pick this card”.
C. The message with text “You cannot pick this card” is not dependent on the order of the options picked.
D. None of the above

Answer: A

Solution:

Q8: Which of the following statement(s) is/are false regarding localStorage API?

A. Local Storage is a client-side storage mechanism.
B. localStorage API provides a “removeElement” method to remove a key-value pair.
C. localStorage API provides a “length” method to get the number of key-value pairs stored.
D. The data stored by a specific domain in local storage of the browser can be directly accessed by its subdomain.

Answer: B and D

Solution: Local Storage is a client side storage mechanism which allows storing key value pairs in the browser for a specific origin. It provides several methods like “removeItem()” to delete a key value pair from the local storage, and “length” that returns the number of key-value pairs stored in it currently. The local storage data for a specific domain is not accessible to its subdomains.

Q9: In order to test that the app is correctly validating the username, what should be the expected assertions below in C1,C2,C3,C4?

HTML:

```
<template>
<div>
    <input v-model="username">
    <div
      v-if="error"
      class="error"
    >
      Validation Failed
    </div>
</div>
</template>

<script>
export default {
  name: 'Hello',
  data () {
    return {
      username: ''
    }
  },
computed: {
    error () {
      if(this.username.trim().length < 7){
        return true
        }
        else{
            let charCode=username.charCodeAt(0)
            if((charCode >= 33 && charCode <= 47) || (charCode >= 58 &&
charCode <= 64)){
                console.log("special char not allowed");
                return true
            }
            else{
                return false
            }
        }

  }
}
</script>
```

JS:

```
test('Login', () => {
  // render the component
  const wrapper = shallowMount(Hello)

   wrapper.setData({ username: ' '.repeat(7) })

   expect(wrapper.find('.error').exists()).toBe(C1)

    wrapper.setData({ username: 'Ramachandra' })

    expect(wrapper.find('.error').exists()).toBe(C2)

    wrapper.setData({ username: '#Radha$' })

    expect(wrapper.find('.error').exists()).toBe(C3)

   wrapper.setData({ username: '__Radha' })

  expect(wrapper.find('.error').exists()).toBe(C4)
})
```

A. true, false, true, true
B. false, true, false, false
C. true, false, true, false
D. false, true, false, true

Answer: C

Solution: The test case here is trying to check if the Vue component is validating the username properly, i.e., if the username is less than 7 characters or if the first character entered is a special character (except underscore). So, if an invalid username is passed using the wrapper.setData(), the test expects a true value, i.e., the computed method should return a true value.