Week 4 Graded Questions

Q1. Consider the following Vue application with markup index.html and javascript file app.js.

index.html:

```
<div id="app">
    <ol>
        <li v-for="item in items">{{item}}</li>
    </ol>
    <button @click="addItem">Add Item</button>
</div>
<script src = "app.js"> </script>
```

app.js:

```
const app = new Vue({
  el: '#app',
  data: {
    items: [],
  },
  methods: {
     // add methods here
      },
})
```

Suppose the application adds a “New item” in the list every time you click on the button “Add Item”, what can be the possible definition of the method “addItem”?

A. addItem() {
items.push('New item')
},

B. addItem() {
this.items.push('New item')
},

C. addItem() {
this.items = 'New item'

},

D. All of the above

Answer: B

Solution: addItem() {
this.items.push('New item')
},
Will add ‘New item’ in items array.

Q2. Which of the following statement(s) is/are correct?

A. v-bind directive is used for one way data binding.
B. v-model directive is used for one way data binding.
C. v-bind directive is used for two-way data binding.
D. v-model directive is used for two-way data binding.

Answer: A and D

Solution: V-bind directive is used for one way binding, and v-model is used for 2-way binding.

Q3. Consider the following Vue application with markup index.html and javascript file app.js.

index.html:

```
<div id="app">
      <h4>total payable amount is {{totalPayableAmount}}</h4>
</div>
<script src = "app.js"> </script>
```

app.js:

```
const app = new Vue({
  el: '#app',
  data: {
```

```
    principal: 0,
    annualInterestRate: 0,
    duration: 0,
    totalPayableAmount: 0,
  },
  computed: {
    simpleInterest() {
      return (this.principal * this.annualInterestRate * this.duration) /
100
    },
  },
})


appData = [
  [2000, 10, 2],
  [3000, 20, 3],
  [5000, 30, 4],
]

let handler = setInterval(() => {
  data = appData.pop()
  app.principal = data[0]
  app.annualInterestRate = data[1]
  app.duration = data[2]
  app.totalPayableAmount += app.simpleInterest
}, 1000)
```

What will be rendered by the browser after 4 seconds?

A. total payable amount is 6000
B. total payable amount is 8200
C. total payable amount is 1800
D. total payable amount is 7800

Answer: B

Solution: After every 1 second, callback function will add simpleInterest in totalPayableAmount. So, after 1 second it will be: 0+6000=400, after 2 seconds it will be 6000+1800=7800 and after 3 seconds it will be 7800+400 = 8200 and after that stack will be empty. Thus, the total amount will not be modified further.

Q4. Consider the following Vue application with markup index.html and javascript file app.js.

index.html:

```
<div id="app">y: {{y}}</div>
<script src = "app.js"> </script>
```

app.js:

```
const app = new Vue({
  el: '#app',
  data: {
    x: 0,
    y: 0,
  },
  watch: {
    x(p, q) {
      if (p > q && p > 10) {
        this.y = 1
      }
    },
  },
})

for (let i = 0; i < 20; i++) {
  app.x++
}
```

What will be rendered by the browser (for the mustache expression 'y')?

A. 0
B. 1
C. 20
D. None of the above

Answer: B

Solution: For loop will increase the data x by 1 at each iteration. So, watcher x will be called, and it will modify the value of y only if new value is greater than old value and also new value is greater than 10. Thus, the value of y will be modified to 1.

Q5. Consider the following Vue application with markup index.html and javascript file app.js.

index.html:

```
<div id="app">y: {{y}}</div>
<script src = "app.js"> </script>
```

app.js:

```
const app = new Vue({
  el: '#app',
  data: {
    x: 20,
    y: 40,
  },
  beforeCreate() {
    console.log(this.x)
  },
})
```

What will be logged on console and rendered on screen (for the mustache expression 'y'), respectively?

A. undefined, 40
B. undefined, 20
C. 20, 40
D. 40, 40

Answer: A

Solution: beforeCreate hook will be called synchronously immediately after the instance has been initialized, before data observation and event/watcher setup (Reference: Vue documentation).
So, function will not have access to data x. Thus, it will log undefined.

Q6. Consider the following Vue application with markup index.html and javascript file app.js.

index.html:

```
<div id="app">y: {{y}}</div>
<script src = "app.js"> </script>
```

app.js:

```
const app = new Vue({
  el: '#app',
  data: {
    x: 20,
    y: 40,
  },

  created(){
   console.log(this.x)
  },
})
```

What will be logged on console and rendered on screen (for the mustache expression 'y'), respectively?

A. undefined, 40
B. undefined, 20
C. 20, 40
D. 40, 40

Answer: C

Solution: Called synchronously after the instance is created. At this stage, the instance has finished processing the options which means the following have been set up: data observation, computed properties, methods, watch/event callbacks. However, the mounting phase has not been started, and the `$el` property will not be available yet (Reference: Vue documentation)

So, created() function will have access to the data x. Thus, it will log 20.

Q7. Consider the following Vue application with markup index.html and javascript file app.js.

index.html:

```
 <div id="app">
      <p directive> data </p>
 </div>
```

app.js:

```
const players = [
  { name: 'Rohit Sharma', role: 'Batsman', team: 'MI' },
  { name: 'Virat Kohli', role: 'Batsman', team: 'RCB' },
  { name: 'Jaspreet Bumrah', role: 'Bowler', team: 'MI' },
]

const app = new Vue({
  el: '#app',
  data: {
    players: players,
  },
})
```

If your app displays the name of the players who play for “MI”. What can be the possible value(s) for directive and data respectively in the above code (mentioned in the element with ID “app”)?

A. v-if="player.team=='MI'" v-for="player in players", {{player.name}}
B. v-for="player in players" v-if="player.team=='MI'", {{players.name}}
C. v-for="player in players" v-if="player.team=='MI'", {{player.name}}
D. v-if="player.team=='MI'" v-for="player in players", {{players.name}}

Answer: A and C

Solution: Because app has to display the player's name who play for MI we need v-if = “player.team == ‘MI’, and it has to display all the players so will need v-for="player in players" to iterate over all the players. {{player.name}} will display the name property of player.

Q8. Consider the following Vue application with markup index.html and javascript file app.js.

index.html:

```
<div id="app">
      <comp1></comp1>
</div>
<script src = "app.js"> </script>
```

app.js:

```
const comp1 = {
  template: '<h4> This is {{name}}</h4>',
  data: function () {
    return {
      name: 'component 1',
    }
  },
}

const app = new Vue({
  el: '#app',
  components: {
    comp1,
  },
})
```

What will be rendered by the browser?

A. This is
B. This is component 1
C. No text will be shown on the browser
D. None of the above

Answer: B

Solution: comp1 is registered in app and template of component is <h4>This is {{name}}</h4> so, name will be replaced by data property name of component. Which is component 1. So, “this is component 1” will be rendered.

Q9. Consider the following Vue application with markup index.html and javascript file app.js.

index.html:

```
<div id="app">
      <comp1></comp1>
 </div>
<script src = "app.js"> </script>
```

app.js:

```
const comp1 = {
  template: '<h4> This is {{name}}</h4>',
  data: {
    name: 'component 1',
  },
}


const app = new Vue({
  el: '#app',
  components: {
    comp1,
  },
})
```

What will be rendered by the browser?

A. This is
B. This is component 1
C. No text will be shown on the browser
D. None of the above

Answer: A

Solution: data option inside the components comp should be a function. So, interpolation will not work here because name property will not be recognized. The browser will only manage to render “This is “. So, option A is correct.

Q10. Consider the following Vue application with markup index.html and javascript file app.js.

index.html:

```
<div id="app">
      <comp-a></comp-a>
</div>
<script src = "app.js"> </script>
```

app.js:

```
const compA = Vue.component('comp-a', {
  template: '<h4> Hello:  {{ message }}</h4>',
  data: function () {
    return {
      message: 'Welcome to IITM',
    }
  },
})

const app = new Vue({
  el: '#app',
})
```

What will be rendered by the browser?

A. Hello:
B. Hello: Welcome to IITM
C. No text will be shown on the browser
D. None of the above

Answer: B

Solution: compA is global component so, we don't need to register it inside the app we can use it directly. So, ‘Hello: Welcome to IITM’ will be rendered.