Week 4 Practice Questions

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

index.html:

```html
<div id="app">{{message}}</div>
<script src="app.js"></script>
```

app.js:

```javascript
const dataObj = {
  message: 'Welcome',
}

const optObject = {
  el: '#app',
  data: dataObj,
}

const app = new Vue(optObject)
app.message = 'Welcome to iitm online degree'
```

What will be rendered by the browser?

A. Welcome
B. Welcome to iitm online degree
C. App will give a warning and will show a blank page.
D. None of the above

Answer: B

Solution: The HTML renders the value of “message” variable defined in the Vue instance. Since, its value has been changed using the object reference, the app will render the updated value of variable “message”.

Q2. Consider the following Vue application with markup index.html and JavaScript file app.js,

index.html:

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

app.js:

```javascript
const app = new Vue({
  el: '#app',
  data: {
    text: "<p style='color:red'> Hello IITM </p>",
  },
})
```

If the app renders “Hello IITM” (in red font color), what is the possible directive for “directive” mentioned in div with ID “app”?

A. v-bind= “text”
B. v-html= “text”
C. Both A and B
D. None of the above

Answer: B

Solution: The “v-html” Vue directive is used to update the inner HTML of an element. It updates the element’s inner HTML with the rendered HTML it gets as input.

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

index.html:

```html
    <div id="app">
      <p v-if="dark" v-bind:style="divStyle">{{message}}</p>
      <p v-else>{{message}}</p>
      <button @click="toggleTheme">change theme</button>
    </div>
  <script src="app.js"></script>
```

app.js:

```javascript
const app = new Vue({
  el: '#app',
  data: {
    message: 'IITM Online degree',
    dark: true,
    divStyle:{
        backgroundColor: "black",
        color: "white"
    }
  },
  methods: {
    toggleTheme() {
       // fill definition here
    },
  },
```

If the application toggles between dark theme (color:white, background-color: black) and normal theme(color: black, background-color:white) on the click of a button with text "change theme", what can be the possible definition of toggleTheme method? (Assume that the application is running on a common browser which displays black colored text on white background)

A. toggleTheme() {
        this.dark = !this.dark
    }

B. toggleTheme() {

```
        this.dark == !this.dark
      }
```

C. toggleTheme() {

```
        dark = !dark
      }
```

D. toggleTheme() {

```
        this.dark = divStyle
      }
```

Answer: A

Solution: The “toogleTheme” method requires toggling the value of Vue instance variable “dark” between “true” and “false”. Since, this variable is to be accessed inside a method defined in the Vue instance, “this” reference must be used to access it.

Q4. Which of the following is true regarding computed properties?

A. It is a function which is triggered when the data it refers to changes.
B. It is calculated each time it is used.
C. It is generally used when some property is used at multiple places in an application
D. None of the above

Answer: A and C

Solution: The computed properties cache the values and only get triggered when one of its reactive dependencies changes, else it serves the cached value each time a re-render happens.

Q5. Which of the following is correct regarding watchers in Vue?

A. It is a function which is triggered when the property it refers to changes.
B. Watchers are defined inside the watch object.
C. Both A and B are correct
D. None of the above

Answer: C

Solution: A watcher is a type of function which is defined inside the watch object of a Vue instance, and it gets triggered each time when the property it refers to changes.

Q6. 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',
 })
```

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: C

Solution: An object with the properties of component is created, but it is not registered with the main Vue instance. So, the browser will consider "<comp1> </comp1>" as the opening and closing of a tag, and it will be ignored while rendering. Thus, nothing will be shown on the browser screen.

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

index.html:

```
<div id="app">
  <ol>
    <comp v-for="message in messages" :message="message"></comp>
  </ol>
</div>
<script src="app.js"></script>
```

app.js:

```
const comp = {
  template: '<li> {{ message.message }} </li>',
  props: ['message'],
}

const app = new Vue({
  el: '#app',
  data: {
    messages: [
      { message: 'This is message1' },
      { message: 'This is message2' },
      { message: 'This is message3' },
    ],
  },
  components: {
    comp,
  },
})
```

What will be rendered by the browser?

A. 1. This is message1
   2. This is message2
   3. This is message3

B. 1. This is message1

1. This is message2
1. This is message3

C. 1. ‘This is message1’
2. ‘This is message2’
3. ‘This is message3’

D. 1. ‘This is message1’
1. ‘This is message2’
1. ‘This is message3’

Answer: A

Solution: A Vue directive “v-for” is being used for the Vue instance variable “messages”, which will iterate over each element in this list. Each iterated element will be sent as a prop to the component “comp”, which renders it as an element of an ordered list. The ordered list by default uses numbers as prefix to each element.

Q8. Which of the following are true?

A. Vue framework can be used for creating Single Page Applications.
B. VueJS can only be used in backend development.
C. A Vue application cannot be integrated with an API.
D. None of the above

Answer: A

Solution: VueJS is a JavaScript based frontend framework which can be used to develop much interactive single page applications, and API calls can also be made in the framework. Thus, it can be very well integrated with APIs.