Week 1 Graded Questions

1. Which of the following keywords can be used to declare a variable with global scope? [MCQ : 1 point]

A. let
B. var
C. const
D. All of the above

Answer: D

Solution: All of these can be used to create variables with global scope, if a variable is declared with them outside a function, which is not limited to any code block.

2. Which of the following statements is true regarding JavaScript language? [MSQ : 1 point]

A. JavaScript is a statically typed language.
B. JavaScript is a dynamically typed language.
C. JavaScript is a weakly typed language.
D. JavaScript is a strongly typed language.

Answer: B and C

Solution: JavaScript language is a dynamically typed language because it doesn't require the data type of the variable is to be specified, when it is declared. It is also a weakly typed language, as it allows various implicit conversions, and provides a lot of flexibility.

3. Which of the following statement(s) is/are correct regarding JavaScript? [MSQ : 1 point]

A. It is a forgiving language like HTML.
B. It inserts a semicolon after each statement automatically.
C. Blocks and functions both use { } for body contents.
D. Blocks use { } but functions use indentation for body contents.

Answer: A, B and C

Solution: JavaScript is considered a forgiving programming language, as it gives a lot of flexibility to the user like coercion etc. JavaScript doesn't require a semicolon to be put at the end of each statement explicitly, as it does it on its own. Both the blocks and functions make use of curly braces ({ }) to define their body or scope, unlike Python, which uses indentation.

4. Consider the following program, and predict the last value shown on the console? [MCQ : 3 points]

```
const timer = (time) => {
  let x = 1
  let handlar = setInterval(() => {
    console.log(2 * x)
    x++
  }, 1000)

  setTimeout(() => {
    clearInterval(handlar)
  }, time)
}

timer(5000)
```

A. 4
B. 8
C. 5
D. 10

Answer: B

Solution: In the program given above, setInterval() function is triggered after every 1 second (1000 milliseconds), and will terminate after waiting for 5 seconds. The first value that will be shown on the console is 2 (after waiting for 1 second), then 4 (after waiting for 1 more second), then 6 (after waiting for 1 more second), then 8 (after waiting for 1 more second). The setTimeout() function will be executed after waiting for 5 seconds. Thus, the last value shown on the console is 8.

5. Consider the following code snippet,

```
const x = { name: 'rohit' }
x.name = 'Mohit'
```

What will be the output of console.log(x.name)? [MCQ : 2 points]

A. It will throw a TypeError.
B. Rohit
C. Mohit
D. undefined

Answer: C

Solution: Using a const keyword while creating or declaring an object will make the reference constant. However, the properties of the object (declared with const) can still be changed.

6. What will be the output of the following program? [MCQ : 2 points]

```
const obj = {
  name: 'Rohit',
  changeName: (name) => {
    this.name = name
  },
}

obj.changeName('Mohit')
console.log(obj.name)
```

A. undefined
B. Rohit
C. Mohit
D. null

Answer: B

Solution: Arrow function does not have “this” of its own. Arrow functions establish "this" based on the scope the Arrow function is defined within. So, the “name” property of the “obj” object will not change. It will remain “Rohit”.

7. What will be the output of the following program? [MCQ : 3 points]

```
const obj = {
  name: 'Rohit',
  arrowFunction: null,
  normalfunction: function () {
    this.arrowFunction = () => {
      console.log(this.name)
    }
  },
}

obj.normalfunction()
obj.arrowFunction()
```

A. Rohit
B. Mohit
C. undefined
D. Syntax Error

Answer: A

Solution: Arrow function does not have “this” of its own. Arrow functions establish "this" based on the scope the Arrow function is defined within. So, the value of “this” will be the “obj” object. Because arrowFunction is defined inside the normalFunction so it will inherit the value of “this” from normalFunction.

8. Which of the following is correct output for the following code snippet? [MCQ : 2 points]

```
setTimeout(() => console.log('hello from setTimeOut one'), 0)
console.log('hello for main one')
setTimeout(() => console.log('hello from setTimeOut two'), 0)
console.log('hello from main two')
```

A. 'hello from setTimeOut one'
'hello for main one'
'hello from setTimeOut two'
'hello from main two'

B. 'hello for main one'
'hello from setTimeOut one'

'hello from main two'
'hello from setTimeOut two'

C. 'hello for main one'
'hello from main two'
‘hello from setTimeOut two'
'hello from setTimeOut one'

D. 'hello for main one'
'hello from main two'
'hello from setTimeOut one'
‘hello from setTimeOut two'

Answer: D

Solution: setTimeout() is an asynchronous function. So, the callback function will be called once the call stack is empty, i.e. first the synchronous codes will be executed and then asynchronous codes.

9. Consider the following HTML document,

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <script>
      setTimeout(() => {
        document.getElementById('div1').style.backgroundColor = 'yellow'
      }, 0)
    </script>
    <title>Document</title>
  </head>
  <body>
    <div
      id="div1"
      style="height: 100px; width: 100px; background-color: red"
    ></div>
  </body>
</html>
```

What will be the background color of an element having ID ‘div1’ in the rendered webpage corresponding to this document? [MCQ : 2 points]

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

Answer: C

Solution: setTimeout() is an asynchronous function. So, the callback function will be called once the call stack is empty, i.e. first the synchronous codes will be executed and then asynchronous codes. Till the time, the statement document.getElementById('div1').style.backgroundColor = 'yellow' will be executed the DOM is already built.

10. Consider the following JavaScript program,

```
let startNamePrinter = (name) => {
  let x = name.split('').reverse()
  let handler = setInterval(() => {
    let y = x.pop()
    console.log(y)
  }, 1000)

  setTimeout(() => {
    clearInterval(handler)
  }, (name.length + 1) * 1000)
}

startNamePrinter('orange')
```

What will be the last value shown on the console after 4 seconds? [MCQ : 3 points]

A. r
B. a
C. n
D. o

Answer: C

Solution: Callback function defined as the parameter of setIntervel will be executed after each 1 second. After every 1 second, it will pop one character from the string and logs it on the console. So, after 4 seconds, it will log character ‘n’.

Week 2 Graded Questions

Q1: Which of the following shows the correct output, if the program written below is executed?

let a = [75, 55, 22, 5];
console.log(a.sort());

A. []
B. Error
C. [5, 22, 55, 75]
D. [22, 5, 55, 75]

Answer: D

Solution: The sort() method sorts the elements in the alphabetical order by default. A user defined function can be passed as an argument to this method in order to customize the default sorting logic.

Q2: Which of the following shows the correct output, if the program written below is executed?

```
a = {
  name: 'Abhi',
  age: 22
};

b = [];
for (let i = 0; i < 3; i++){
    b.push(a)
}

b[1].name = 'Akshay';
console.log(b[0].name);
```

A. Abhi
B. Akshay
C. undefined
D. ReferenceError

Answer: B

Solution: Here, the same object reference “a” is pushed thrice to an array. Hence, a change in one element of the array will affect all other elements of the array, as all the elements are referencing to the same object.

Q3: Which of the following shows the correct output, if the program written below is executed?

```
const a = {
  name: 'Abhi',
  age: 22
};

b = [];
for (let i = 0; i < 3; i++){
    b.push({...a})
}
b[1].name = 'Akshay';
console.log(b[0].name);
```

A. Abhi
B. Akshay
C. Undefined
D. ReferenceError

Answer: A

Solution: Here, a destructured version of the object reference “a” is pushed thrice to an array. Hence, a change in one element of the array will not impact any other element of this array, as all the elements are referencing to the different objects.

Q4: Consider the following program,

```
x = [1, 2, 3, 4, 5, 6]
y = [...x, 7, 8, 9]
z = y.filter((x) => x % 2 == 0)
     .map((i) => i * i)
     .reduce((a, i) => a + i, (a = 0))
```

```
console.log(z)
```

What will be the output of the program?

A. 100
B. 120
C. 140
D. 160

Answer: B

Solution: Here, the filter() method returns an array of the even elements from the array “y”. The result of which is given to the map() method, which returns an array of square of elements from the array it got from the filter() method as the result. The resultant array is then passed to the reduce() method, which computes the final answer as 120, which further gets displayed on the console.

Q5: What will be the output of the following program?

```
const obj = {
  x: 1,
  y: 1,
  set nums(xCommay) {
    numbers = xCommay.split(',')
    this.x = numbers[0]
    this.y = numbers[1]
  },
  get nums() {
    return this.x + ',' + this.y
  },
  xPowery: function () {
    let result = 1
    for (let i = 1; i <= this.y; i++) {
      result = result * this.x
    }
    return result
  },
}
obj.nums = '2,3'
console.log(obj.xPowery())
```

A. 6
B. 5
C. 8
D. None of the above

Answer: C

Solution: The example uses getter and setter properties of an object. The setter modifies the properties of the object, whereas the getter returns the properties. Then, it invokes a method, which does some computations and returns the result.

Q6: What will be the output of the following program?

```
const ePowerx = {
  x: 1,
  n: 1,
  set parameters(param) {
    param = param.split(' ')
    this.x = param[0]
    this.n = param[1]
  },
  get uptoNterms() {
    let result = 1
    let y = 1
    for (let i = 1; i < this.n; i++) {
      y = y * this.x
      result = result + y
    }
    return result
  },
}
ePowerx.parameters = '2 3'
console.log(ePowerx.uptoNterms)
```

A. 3
B. 5
C. 7
D. 15

Answer: C

Solution Above is the logic to compute 1+x+x^{2}+x^{2}+... upto n terms and in this problem x=2 and n=3 so, the answer is 1+2+4=7.

Q7: What will be the output of the following program?

```
const course = {
  courseName: 'Modern Application Development 2',
  courseCode: 'mad2',
}

const student = {
  __proto__: course,
  studentName: 'Rakesh',
  studentCity: 'Delhi',
}

const { courseName } = student
console.log(courseName)
```

A. Undefined
B. Modern Application Development 2
C. Will throw syntax error
D. None of the above

Answer: B

Solution: Student object inherit from the course object so, it will have access to all the properties of course object so, Option B is correct.

Q8: Consider the following,

Index.html:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>Document</title>
  </head>
  <body>
```

```
    <div id="div1" style="background-color: red"></div>
  </body>
  <script src="script.js"></script>
</html>
```

script.js:

```
var height = 0
var width = 0
var divStyle = document.getElementById('div1').style
let res = setInterval(() => {
  height += 50
  width += 50
  divStyle.height = height + 'px'
  divStyle.width = width + 'px'
  console.log(height)
}, 1000)

setTimeout(() => {
  clearInterval(res)
}, 10050)
```

Assuming that the index.html and script.js are kept inside the same directory, then what will be the height of the HTML element having ID ‘div1’ after 20 seconds, if rendered using a browser?

A. 500px
B. 1000px
C. 0px
D. None of the above

Answer: A

Solution: The callback function inside the setInterval will get executed every 1 second but because after 10050 milliseconds callback function inside the setTimeout will get executed and will clear the interval. And, the callback function inside setInterval increases the height of div by 50px, and it will run 10 times so, its final height will be 500px.

Q9: Consider the following,

Index.html:

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>Document</title>
    <script src="module1.js" type="module"></script>
  </head>
  <body>
    <div id="div1" style="border: 2px solid black"></div>
  </body>
</html>
```

module1.js:

```javascript
import { divStyle1, divStyle2 } from './module2.js'
divStyle1.width = "50px"
const div1Style = document.getElementById('div1').style
div1Style.backgroundColor = divStyle1.color
div1Style.height = divStyle2.height
div1Style.width = divStyle1.width
```

module2.js:

```javascript
export const divStyle1 = {
  color: 'red',
  height: '100px',
  width: '100px',
}

export const divStyle2 = {
  color: 'green',
  height: '50px',
}
```

Assuming that the index.html, module1.js, and module2.js are kept inside the same directory, then what will be the background color, height and width of the HTML element having ID ‘div1, if rendered using a browser?

A. Red, 100px, 100px
B. Red, 50px, 50px
C. Green, 50px, 100px
D. Green, 50px, 50px

Answer: B

Solution: width property of divStyle1 has been override inside the module 1 so, width of the div1 is 50px, and backgroundColor comes from divStyle1 so, it will be red and height comes from divStyle2 so, it will be 50px.

Q10: What will be the output of the following program?

```
api = { A: 'Application', P: 'Programming', I: 'Interface' }
const standsFor = (x) => {
  for (const [k, v] of Object.entries(api)) {
    if (k != x) {
      return v
    }
  }
}

console.log(standsFor('A'))
```

A. Application
B. Programming
C. Interface
D. A

Answer: B

Solution: Program will log the first value whose key is not ‘A’. So, it will be “Programming”.

Week 3 Graded Questions

Q1: Which of the following are examples of system level state that needs to be maintained for the entire app?

A. User database of NPTEL
B. List of emails selected for archiving in Gmail
C. Google search index
D. User shopping cart content on Amazon.in

Ans: A and C

Explanation: User preference is frontend state, shopping cart is user specific application state.

Q2: Which of the following could be considered user specific application state?

A. User database of NPTEL
B. List of emails selected for archiving in Gmail
C. Google search index
D. User shopping cart content on Amazon.in

Ans: D

Explanation: User shopping cart content is specific to the user. So, it can be considered as user specific application state.

Q3: When you view the online discourse forum, there is a line indicating “last visit” - posts above this have been updated since the last time you visited the forum. What kind of state information is this indicator conveying?

A. Ephemeral UI state
B. Application state
C. System state
D. None of the above

Ans: B

Explanation: Although this is temporary and changes each time the user loads the page, it still needs to be stored at the server to keep it up to date.

Q4: Which of the following methods can be used to ensure that the displayed state and system state are kept in sync at all times?

A. Ajax requests on each UI change
B. Periodic reloading of web-page
C. Vue bindings to update data reactively
D. Pure static pages with all updates rendered from server

Ans: A, D

Explanation: Vue bindings by themselves do not synchronize state, and reloading a page will most likely cause loss of any ephemeral information.

Q5: The M in MVC stores _____.

A. Ephemeral UI state
B. Application state
C. System state
D. None of the above

Ans: C

Explanation: The main idea of persistent storage is that you can reconstruct system state from the stored data

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

index.html:

```
<div id="app">{{newMessage}}</div>
```

app.js:

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

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

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

What will be rendered by the browser?

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

Answer: C

Explanation: properties in data are only reactive when they are present when instance was created and “newMessage” is not present in data object it was assigned after the instance was created so, app will not be able to identify the property.

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

index.html:

```
<div id="app">{{count+' second remaining'}}</div>
```

app.js:

```
const dataObj = {
  count: 10000,
}

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

```
const app = new Vue(optObject)

setInterval(() => {
  app.count -= 1
}, 1000)
```

What will be rendered by the browser after 1 minutes?

A. 10000
B. 9999
C. 9940
D. None of the above

Answer: C

Explanation: After each 1 second callback function of setInterval will reduce the count property of data object by 1 so after 1 minute it well be reduced by 60 so, answer is 9940.

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

index.html:

```
<div id="app" style="color: white"
v-bind:style="divStyle">{{message}}</div>
```

app.js:

```
const dataObj = {
  message: 'IITM online',
  divStyle: {
    backgroundColor: 'red',
    padding: '20px',
    fontSize: '2em',
  },
}

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

const app = new Vue(optObject)
```

What will be the color, background color and font-size of the div with ID “app”?

A. White, red, 2em
B. Black, white, 1em
C. white, white, 1 em
D. Black, red, 2em

Answer: A

Explanation: v-bind will bind the style property of div element with divStyle property of data object, and it already has color white assigned so, option A is correct.

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

index.html:

```
<div id="app" v-bind:style="{fontSize:fontSize+'em'}">
   <p>{{message}}</p>
   <button v-on:click="zoom">click me</button>
</div>
```

app.js:

```
const dataObj = {
  message: 'IITM online',
  fontSize: 2,
}

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

```
const app = new Vue(optObject)
```

If the font size of the text ‘IITM Online’ increases by 1 em each time the user clicks on the button, which one of the following is the definition of objMethod?

A. const objMethods = {
```
  zoom() {
    this.fontSize += 1em
  },
}
```

B. const objMethods = {
```
  zoom() {
    fontSize += 1
  },
}
```

C. const objMethods = {
```
  zoom() {
    this.fontSize += 1
  },
}
```

D. const objMethods = {
```
  zoom() {
    fontSize += 1em
  },
}
```

Answer: C

Explanation: Each time user clicks on the button, zoom method will be triggered, and style property is bind with the object {fontSize:fontSize+'em'} so, in order to change the font size we zoom need to change the fontSize property of data object. So, Option C is correct.

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

index.html:

```html
<div id="app">
     <p directive1>{{message}}</p>
     <button directive2>click</button>
 </div>
```

app.js:

```javascript
const app = new Vue({
  el: '#app',
  data: {
    message: 'Hello',
    seen: true,
  },
  methods: {
    toggleSeen() {
      this.seen = !this.seen
    },
  },
})
```

If you want to toggle between the presence of p element on click of the button element, what are the possible value of directive1 and directive2 and their respective values?

A. v-if=”seen”, v-on:click=”toggleSeen”
B. v-if:”seen”, v-on:click:”toggleSeen”
C. v-if=”seen”, @click=”toggleSeen”
D. v-if:”seen”, @:click:”toggleSeen”

Answer: A and C

Explanation: v-on will bind the click event to the method “toggleSeen” which will change the value of seen property according to the previous set property. And v-if will control the display of p element based on truth value of seen property. And @ is shorthand for v-on. So, A and C are correct.

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:

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

app.js:

```javascript
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:

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

app.js:

```javascript
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:

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

app.js:

```javascript
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.

Week 5 Graded Questions

Q1: Consider the following JavaScript program.

```
const a = fetch("https://httpbin.org/get")

a.then(r => {
    if (!r.ok){
       throw new Error("HTTP status code: " + r.status);
   }
   return r.json();
}).then(d => {
   console.log("Got the data !!");
}).catch(e => {
   console.log(e);
})
```

What will be shown on the browser console if the above program is executed?

A. Error: HTTP status code: 200
B. Error: HTTP status code: 404
C. Got the data !!
D. Error: HTTP status code: 500

Answer: C

Solution: The fetch() method will return the first promise, which will resolve into the response object (r), since the URL given was valid. After that, a second promise will be returned by the statement “r.json()”, which will further resolve into data (d), since there is no error.

Q2: Which of the following statements is true regarding fetch method?

A. Fetch method works asynchronously by default.
B. Fetch method cannot work synchronously.
C. Fetch method always returns a promise.
D. Fetch method cannot be used for making a POST request.

Answer: A and C

Solution: The fetch() method works asynchronously by default, but can be made work in a synchronous manner by using async and await keywords. Fetch method always returns a promise which resolves, reject or generate an error. Fetch method can be used to create almost all types of HTTP requests.

Q3: Consider the following files are present inside the same directory.

index.html:

```html
<!DOCTYPE html>
<html>
 <head>
   <meta charset="utf-8">
   <meta name="viewport" content="width=device-width">
   <title>repl.it</title>
   <link href="style.css" rel="stylesheet" type="text/css" />
 </head>
 <body>
 </body>
<script src="app.js"></script>
</html>
```

app.js:

```javascript
const a = fetch("sample.txt")

a.then(r => {
    if (!r.ok){
      throw new Error("HTTP status code: " + r.status);
   }
   return r.text();
}).then(d => {
   console.log("Got the data !!", d);
}).catch(e => {
   console.log(e);
})
```

sample.txt:

```
Hello World !!
```

What will be shown on the browser console if index.html is rendered with the help of a browser?

A. Got the data !! Hello World !!
B. Hello World !!
C. Got the data !!
D. Error: HTTP status code: 404

Answer: A

Solution: The fetch() method will return the first promise, which will resolve into the response object (r), since the URL given was valid. After that, a second promise will be returned by the statement “r.json()”, which will further resolve into data (d), since there is no error, and print the message “Got the data !! Hello World !!” on the console.

Q4: Place the correct code in places marked “code1”, “code2”.

Say the files shown below are present inside the same directory. What will be the correct way to retrieve the attendees names and IDs out of the response?

data.json:

```
{
    "id":1,
    "course":"MAD-II",
    "date":"Jan 19 2022",
    "time":"18:00",
    "description":"Live Session Vuejs",
    "attendees":[
        {
            "id":"mada123",
            "name":"Ravi"
        },
        {
           "id":"mad124",
            "name":"Raju"
```

```
         },
         {
            "id":"mad125",
            "name":"Manju"
         },
         {
            "id":"mad127",
            "name":"Ck"
         },
         {
            "id":"mad130",
            "name":"Rahim"
         }
     ]
 }
```

index.html:

```
    <div id="app">
        <table border="1px">
            <tr>
                <td>AttendeeId:</td>
                <td>AttendeeName:</td>
            </tr>
            <tr v-for ="code1">
                <td>{{item.id}}</td>
                <td>{{item.name}}</td>

            </tr>
        </table>
    </div>

    <script src="app.js"></script>
  </body>
```

app.js:

```
new Vue({
    el: '#app',
    data(){
    return{
        liveSessions:[]
    }
    },
    async created(){
      return fetch('data.json')
      .then(response => response.json())
      .then(responseJson => {
        code2

      })
      .catch(error => {
        console.error(error);
      });

    }
})
```

A. code1:this.liveSessions=responseJson.data(),
   code2 : item in liveSessions
B. code1:this.liveSessions=responseJson,
   code2 : item in liveSessions['attendees']
C. code1:this.liveSessions=responseJson.data.attendees,
   code2 : item in liveSessions['attendees']
D. code1:this.liveSessions=responseJson,
   code2 : item in liveSessions.attendees

Answer: B and D

Solution: The promise returned by the fetch call resolves into a response object, which further resolves into the data using json() method. The data obtained from the fetch call is an object having attributes “id”, “course”, “attendees” etc., where “attendees” is an array of objects which should be iterated through, to display the needed information on the web page.

Q5: How to extract “temp” from the given API call and display the information in the index.html?

To get output in a format like Temperature:26.5°C, fill Code1

index.html:

```
<div id="app">
       <p>Temperature:Code1</p>
</div>
<script src="app.js"></script>
```

app.js:

```
new Vue({
    el: '#app',
    data(){
    return{
        weather:[]
    }
    },

    async created(){
      return
fetch('https://fcc-weather-api.glitch.me/api/current?lat=12.97&lon=77.59')
      .then(response => response.json())
      .then(responseJson => {
        this.weather=responseJson
        console.log(this.weather)
      })
      .catch(error => {
        console.error(error);
      });

    }
})
```

A. {{weather.main.temp}}degreeC
B. {{weather.main.temp}}&deg;C
C. {{weather.main.temp}}C
D. None of the above

Answer: B

Solution : The entire json data from the url is first stored in weather[] .
To get the temperature from the given we can use weather.main.temp .
(check the data format of the api
https://fcc-weather-api.glitch.me/api/current?lat=12.97&lon=77.59)
To get the degree symbol we can use the unicode use
&deg; or &#176;

Q6: Suppose the server at “worldtimeapi.org” is down. What will be shown on the console?

app.js:

```
new Vue({
    el: '#app',
    data(){
    return{
        time:[]
    }
    },

    created(){
      fetch('http://worldtimeapi.org/api/timezone/Asia/Kolkata')
      .then(response => response.json())
      .then(responseJson => {
        this.time=responseJson
        console.log(this.time)
      })
      .catch(error => {
        console.log(“Failed to Fetch”);
      });

    }
})
```

A. Failed to fetch
B. Reference error

C. Server Down
D. None of the above

Answer: A
Solution: If a fetch to the url fails, it will enter the catch and print the error message.

Q7: Consider the following JavaScript code,

```javascript
let x = 2,
  n = 3,
  k = 4

const Promise1 = new Promise((resolved, rejected) => {
  if (k < n) {
    resolved(x)
  } else {
    rejected('Bad Promise')
  }
})

const promise2 = Promise1.then((x) => {
  return x * x
})
promise2
  .then((x) => {
    console.log(x)
  })
  .catch((err) => {
    console.log(err)
  })
```

What will be logged on the console?

A. 2
B. 4
C. 8
D. Bad Promise

Answer: D

Solution: As the condition k < n fails, the “Promise1: gets rejected with a message “Bad Promise”. Since, the rejection of this promise is not handled, it returns an unhandled or rejected promise to “promise2”. Now, “promise2” has a rejected promise, which gets caught in the catch block, and shows the message “Bad Promise”.

Q8: Consider the following JavaScript Code.

```
function promiseGenerator(s) {
  return new Promise((resolve, reject) => {
    setTimeout(() => {
      resolve(`Will resolve after ${s} seconds`)
    }, s * 1000)
  })
}
async function asyncFunc() {
  console.log('Initial statement') // statement 1
  let prom1Val = await promiseGenerator(20)
  console.log(prom1Val) // statement 2
  let prom2Val = await promiseGenerator(30)
  console.log(prom2Val) // statement 3
}

asyncFunc()
```

What is the approximate time (in seconds) taken by statement 2 to run after statement 1 and time taken by statement 3 to run after statement 1, respectively?

A. 20, 30
B. 30, 30
C. 20, 50
D. 50, 50

Answer: C

Note: This question is inspired by the example given under the heading "Async functions and execution order" in the document https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/async_function#async_functions_and_execution_order

Q9: Consider the following JavaScript program, and predict the output if executed.

```
let start = 5;
function check() {
   return new Promise((res, rej) => {
   let a1 = setInterval(() => {
         start++;
         if (start === 7) {
             console.log("Reached");
             clearInterval(a1);
             res("Pass");
             rej("Fail");
         }
         else {
             console.log("Yet to Reach");
         }

     }, 500);
 })
}
check().then(
   pass => console.log(pass)
   ).catch(
       fail => console.log(fail)
   );
```

A. Yet to Reach
   Reached
   Pass

B. Yet to Reach
   Reached
   Pass
   Fail

C. Yet to Reach
   Reached
   Fail

D. Yet to Reach
   Yet to Reach
   Reached
   Pass
   Fail

Answer: A

Solution: The function named “check” returns a promise. This promise resolves when the value of variable start becomes 7 (a number). It will take one iteration for the value to become 7, since it is initialized with the value “5”. So, it will log “Yet to Reach” firstly. Once, the value of the variable named “start” becomes 7, it logs the message “Reached” on the console, and resolves the promise with the message “Pass”, which will be logged on the console as well, as the “then” simply logs this value on the console.

Q10: Consider the following JavaScript program.

```
async function func1(){
   new Promise(rej => setTimeout(rej, 4000));

   async function func2(){
       await new Promise(rej => setTimeout(rej, 2000));

       console.log("Finished");
   }
   func2();
}
func1();
```

What will be the minimum time taken by the above program to log the message “Finished” on the console?

A. 4 seconds
B. 6 seconds
C. 0 seconds
D. 2 seconds

Answer: D

Solution: The function named “func1” is an async function, and the keyword “await” can be used inside it. The function defined a promise asynchronously, also this function contains a nested function with name “func2”, which again defined a promise, but it will be executed synchronously, and there will be a wait of minimum 2000 milliseconds (2 second) to execute the console.log(“Finished”) statement, since the promise will take 2 seconds to get resolved. After waiting for 2 more second, the promise in the outer function will also get resolve

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.

Week 7 Graded Questions

Q1: Consider the following markup index.html and JavaScript app.js for a Vue application.

index.html:

```
<div id="app">
      <div>
        <router-link :to="{name:'home'}"> Home </router-link>
        <router-link :to="{name:'profile', params:{name:'Abhisek'}}">
          Profile</router-link
        >
      </div>
      <router-view />
    </div>
<script src="app.js"></script>
```

app.js:

```
const home = {
  template: `<h1> Home </h1>`,
}

const profile = {
  template: `<h1> Hello {{ name }}</h1>`,
  props: ['name'],
}

const routes = [
  { path: '/', name: 'home', component: home },
  { path: '/profile/:name', name: 'profile', props: true, component:
profile },
]

const router = new VueRouter({
  routes,
  base: '/',
})

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

What will be rendered inside the router-view when user loads the application for the first time and when user clicks on the link “Profile", respectively?

A. Blank Page, Hello Abhisek
B. Hello Abhisek, Home
C. Home, Hello Abhisek
D. None of the above

Answer: C

Solution: Because base property of router instance is ‘/’. A component associated with ‘/’ which is home will be rendered inside the router-view component. Once the user clicks on the Profile, the user will be redirected to the route with the name ‘profile’ along with the parameter name = Abhisek. props = true so, parameters will be passed as props. So, answer C is correct.

Q2: Consider the following markup index.html and JavaScript app.js for a Vue application.

app.js:

```
const player = {
  template: `<div>
    <h1> Name: {{ name }}</h1>
    <router-view />
  </div>`,
  props: ['name'],
}

const test = {
  template: '<div><h1> Test Runs: {{ runs }} </h1></div>',
  data() {
    return { runs: 5000 }
  },
}

const oneDay = {
  template: '<div><h1> Oneday Runs: {{ runs }} </h1></div>',
  data() {
    return { runs: 10000 }
  },
}

const routes = [
  {
    path: '/player/:name',
    component: player,
    children: [
      { path: '', component: oneDay },
      { path: 'test', component: test },
      { path: 'oneday', component: oneDay },
    ],
    props: true,
  },
  { path: '*', component: test },
]
const router = new VueRouter({
  routes,
  base: '/',
})

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

index.html:

```
<div id="app">
   <router-view></router-view>
</div>
<script src="app.js"></script>
```

Suppose the application is running on http://localhost:8080, and user visits the URL "http://localhost:8080/#/player/rohit". What will be rendered inside the router-view?

A. Name: Rohit
   Oneday Runs: 10000

B. Test Runs: 5000

C. Name: Rohit
   Test Runs: 5000

D. Oneday Runs: 10000

Answer: A

Solution: Because nothing has been mentioned after the "http://localhost:8080/#/player/rohit". The component associated with path ' ' inside the children list will be rendered inside the router-view of player component template. Which is oneDay so, Option A.

Q3: Consider the app.js and index.html given in the Question No.2.

Suppose the application is running on http://localhost:8080, and the user visits the URL "http://localhost:8080/#/player/rohit/test". What will be rendered by the browser?

A. Name: Rohit
   Oneday Runs: 10000

B. Test Runs: 5000

C. Name: Rohit
   Test Runs: 5000

D. Oneday Runs: 10000

Answer: C

Solution: Because ‘/test’ is after the “http://localhost:8080/#/player/rohit”. The component associated with path ‘test’ inside the children list will be rendered inside the router-view of player component template, which is “test” so, Option C.

Q4: Consider the app.js and index.html given in the Question No.2.

Suppose the application is running on http://localhost:8080, and the user visit the URL “http://localhost:8080/#/player/rohit/t20”. What will be rendered by the browser?

A. Name: Rohit
   Oneday Runs: 10000

B. Test Runs: 5000

C. Name: Rohit
   Test Runs: 5000

D. Oneday Runs: 10000

Answer: B

Solution: Because ‘/t20’ is after the “http://localhost:8080/#/player/rohit”. So, router will not be able to find any match. So, the component associated with ‘*’ will be rendered inside the router-view of div with ID “app”.

Q5: Consider the following markup index.html and JavaScript app.js for a Vue application.

index.html:

```
<div id="app">
    <div>
        <router-link to="/profile" replace> Profile </router-link>
        <router-link to="/post"> Post </router-link>
    </div>
    <div>
```

```
        <router-view />
    </div>
</div>
```

app.js:

```
const profile = {
  template: '<div> <p> Profile Page </p> </div>',
}

const post = {
  template: '<div><p> Users Post </p></div>',
}

const routes = [
  { path: '/profile', component: profile },
  { path: '/post', component: post },
  { path: '*', component: post },
]
const router = new VueRouter({
  routes,
  base: '/',
})

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

Suppose the application is running on http://localhost:8080. If after running the application in new tab for the first time (browser's history stack is empty), and the user clicks on the link 'Profile' then which of the following statement is true?

A. 'Profile Page' will be rendered inside router-view, the user will not be able to navigate back to the default post page using browser's back button.
B. 'Profile Page' will be rendered inside router-view, the user will be able to navigate back to the default post page using browser's back button.

C. ‘Users Post’ will be rendered inside router-view, the user will not be able to navigate back to the default post page using browser’s back button.
D. ‘Users Post’ will be rendered inside router-view, the user will be able to navigate back to the default post page using browser’s back button.

Answer: A

Solution: Once the user visit the app for first time, post component will be rendered inside the router view and ‘/post’ will be pushed inside the browser’s history stack. When the user click on ‘Profile’, profile component will be rendered inside the router-view and ‘/post’ will be replaced with the ‘/profile’ because of replace props so, the browser’s history stack will look something like [‘/profile’] so, user will not be able to go back to default post page using browsers back button. So, option A.

Q6: Consider app.js and index.html from the Question No.5.

Suppose the application is running on http://localhost:8080. If after running the application in new tab for the first time (browser’s history stack is empty), and the user clicks on the link “Profile” and then ‘Post’, then which of the following statement is true?

A. ‘Profile Page’ will be rendered inside router-view, the user will not be able to navigate back to the default post page using browser’s back button.
B. ‘Profile Page’ will be rendered inside router-view, the user will be able to navigate back to the default post page using browser’s back button.
C. ‘Users Post’ will be rendered inside router-view, the user will not be able to navigate back to the default post page using browser’s back button.
D. ‘Users Post’ will be rendered inside router-view, the user will be able to navigate back to the default post page using browser’s back button.

Answer: C

Solution: When the user visits the app for the first time, the default ‘/post’ will be pushed to the browser’s history stack. And then when the user clicks on the ‘profile’ link, ‘/post’ will be replaced with the ‘/profile’ because of the ‘replace’ prop. Again, when the user visits the post page, ‘/post’ will be pushed to the history stack. So at the end stack will look something like [‘/profile’, ‘/post’] so, user will not be able to navigate back to the default ‘post’ page using browsers back button.

Q7: In the code snippet below, the Vuex store has a state called product.

Fill in the code in ***code1*** inside the “products” computed property in order to render the name and prices of the different products.

index.html:

```html
<div id = "app">
         <table border="1px">
           <tr>
                <td>Product</td>
                <td>Price</td>
           </tr>
           <tr v-for="product in products">
               <td>{{product.name}}</td>
               <td>{{product.price}}</td>
           </tr>
       </table>

    </div>
```

app.js:

```js

const store = new Vuex.Store({
    state: {
      count:0,
      products:[
        {name:'apple',
        price:120
        },
        {
          name:'banana',
          price:60
        }
      ]
    },


  })
   new Vue({
    el:"#app",
    store:store,

    computed:{
        products(){
            return code1
          }
    }
})
```

A. store.commit.products
B. store.state.products
C. store.dispatch.products
D. None of the above

Answer: B

Solution: One can retrieve the values in the central store by returning the store state within a computed property. In the above example, it should be store.state.products

Q8: Fill in ***code1***, inside a mutation handler, which can be used to reduce the price of each of the products in the state by a certain discount.

app.js:

```
const store = new Vuex.Store({
    state: {
      discount:10,
      products:[
        {name:'apple',
        price:120
        },
        {
          name:'banana',
          price:60
        }
      ]
    },
    mutations: {
        reducePrice:state=>{
            Code 1

        }
    }
})
```

A. this.$store.state.products.forEach(product => {
   product.price -= (state.discount)/100*product.price;})

B. state.products.forEach(product => {
   product.price -= (state.discount)/100*product.price;})

C. commit.products.forEach(product => {
   product.price -= (state.discount)/100*product.price;})

D. dispatch.products.forEach(product => {
   product.price -= (state.discount)/100*product.price;})

Answer: B

Solution: Correct option is B. The mutation handler receives state as its first argument. The discount needs to be applied on each of the products.

Q9: Which of the following statement(s) is are correct?

A. Web Manifests, service workers are some characteristics of a PWA.
B. All SPAs are PWAs.
C. A Web worker is a script started by a web content that runs in the background, which can perform computations and fetch requests and communicate back messages to the origin web content.
D. Search engine optimization, managing browser history are some challenges of SPAs.

Answer A, C and D

Solution: Reference :
https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps/Introduction

Q10: Which of the following statements is false regarding Vuex?

A. The mutations must be synchronous.
B. The state provided by Vuex store is shared among all the components of the application.
C. An action should not directly change the state of a Vuex store, and must call a mutation to change the state.
D. Only the direct child components (not descendants) of the app have the access to the "this.$store" variable.

Answer: D

Solution: All components of an application have access to the centralized store.

Week 8 Graded Questions

Q1: Which of the following statements is false regarding REST and GraphQL?

A. In RESTful APIs, there are multiple endpoints to access different resources.
B. In GraphQL, there is only a single entry point in general.
C. GraphQL fixes the issue of over fetching the data.
D. Both REST API and GraphQL can request only a single resource with a single request.

Answer: D

Solution: GraphQL is a query language for our API, which helps to make complex queries and helps to retrieve only the data that is required.
Ref: https://graphql.org/

Q2: Which of the following statement(s) is/are false regarding REST and GraphQL?

A. GraphQL can fetch the data from multiple sources and return the fused response.
B. In REST, the GET requests are not cacheable in general.
C. In GraphQL, the POST requests are cacheable in general.
D. GraphQL can execute a complex query to return a complex data set with a single request, whereas, REST APIs may require to make multiple requests for the same.

Answer: B and C

Solution: A GraphQL query can be constructed, requesting data from multiple resources. A request is generally cacheable, when the response can be constructed with just the URL, and the reason for GET requests being cacheable is that a GET request has all the required data in the URL. However, a POST request has data in the request body, which is not cacheable in general.

Q3: Which of the following HTTP request methods is cacheable in general?

A. GET
B. POST
C. PUT
D. All of the above

Answer: A

Solution: A request is generally cacheable, when the response can be constructed with just seeing the URL, and the reason for GET requests being cacheable is that a GET request has all the required data in the URL. However, a POST or PUT request has data in the request body, which is not cacheable in general.

Q4: Which of the following statements is true?

A. A POST GraphQL request with Content-Type header as “application/graphql” need not always have the request body content as GraphQL query string.
B. A POST GraphQL request can also be made with a JSON body content.
C. The response to a GraphQL request is a JSON response, with the result stored in the “data” field.
D. All of the above

Answer: B and C

Solution: A GraphQL request can be constructed with the “Content-Type” header value as both “application/graphql” and “application/json”, but the body content should match with the header value. The response to a GraphQL request is generally a JSON response, with the data stored in “data” field.

Q5: Which of the following statements is/are false?

A. All the responses to the GraphQL requests should return 200 as the status code in general.
B. A GET request cannot be made in GraphQL, and it must always send POST requests.
C. All the endpoints in a RESTful API should return 200 as the status code.
D. None of the above

Answer: B and C

Solution: An endpoint in an REST API should return an appropriate status code with the response, and it need not be 200 always. You can read more about GraphQL at GraphQL - GraphQL (dgraph.io)

Q6: Which of the following statement(s) is/are true?

A. Web Components allow us to create custom HTML elements.
B. HTML markup is more verbose than Markdown language.
C. Markdown content has inline styling in general.
D. A Web Component can have scoped styling.

Answer: A, B, C and D

Solution: A web component can be used to create custom HTML tags or elements (just the way components in Vue work), and each of the web component can have styling specific to them, which is not applied to other parts of the web page.
HTML is considered to be a more verbose language, if compared with Markdown, which has inline styling in general.

Q7: Which of the following statement(s) is/are true about GraphQL?

A. Mutations in GraphQL can be used to change the underlying data store.
B. GraphQL services are created by defining types and fields of those types, then providing functions for each field.
C. GraphQL can only fetch data from a single source.
D. GraphQL allows only GET operations.

Answer: A and B

Solution : In GraphQL query is used to retrieve information and mutations can be used to change the underlying data store. The API's in GraphQL are organized through types (entities) and corresponding fields.
Reference : https://graphql.org/

Q8: Which of the following statement(s) is/are true regarding GraphQL?

A. It is a query language for APIs.
B. It is a client side runtime.
C. It is not tied to any specific database or storage engine.
D. It usually sends complex queries over the POST body of the HTTP request.

Answer: A, C and D

Solution: GraphQL is a query language for API. It can be thought of as a layer on the server side through which we can make complex queries, usually over the POST body of the request. It can connect to multiple data sources.

Q9: What is the main idea behind headless CMS?

A. The content repository is separated/decoupled from the presentation layer.
B. Content is delivered via APIs.
C. It helps in easier scalability.
D. A headless CMS cannot use REST APIs.

Answer: A, B and C

Solution: Headless CMS the content is decoupled from the front end. Content API can then be used to access content where needed - website/mobile app.

Q10: Which of the following is the most commonly used representation for resources in web APIs?

A. xml
B. json
C. webp
D. woff

Answer: B

Reference: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/JSON

Week 9 Graded Questions

Q1: Consider the flask application app.py and an HTML file index.html,

app.py:

```
from flask import Flask, jsonify
from datetime import datetime
import time

app = Flask(__name__)


@app.route('/')
def home():
    time.sleep(10)
    return jsonify(datetime.now().second), 200,
{'Access-Control-Allow-Origin': '*'}


@app.route('/profile')
def profile():
    time.sleep(30)
    return jsonify(datetime.now().second), 200,
{'Access-Control-Allow-Origin': '*'}


if __name__ == "__main__":
    app.run(debug=True)
```

index.html:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>Document</title>
  </head>
  <body>
    <script>
      function test() {
        const res1 = fetch('http://127.0.0.1:5000')
        const res2 = fetch('http://127.0.0.1:5000/profile')
        res1
          .then((res) => {
            return res.json()
          })
          .then((data) => {
            console.log(data)
          })

        res2
          .then((res) => {
            return res.json()
          })
          .then((data) => {
            console.log(data)
          })
      }
      test()
    </script>
  </body>
</html>
```

If a user visits index.html at 10:10:10 AM (in format HH:MM:SS), what will be logged in the console (approximately)?

A. 20, 50
B. 20, 40
C. 40, 40
D. 50, 50

Answer: B

Solution: The flask application is being run in the threaded mode (default). This means that the application will be able to accept and process multiple requests simultaneously. Thus, if 2 requests are sent at the same time (10:10:10), the route “/” will take a minimum of 10 seconds to respond, and route “/profile” will take a minimum of 30 seconds, but both will be processed simultaneously in different threads.

Q2: Consider the flask application app.py and an HTML file index.html,

app.py:

```
from flask import Flask, jsonify
from datetime import datetime
import time

app = Flask(__name__)


@app.route('/')
def home():
    time.sleep(10)
    return jsonify(datetime.now().second), 200,
{'Access-Control-Allow-Origin': '*'}


@app.route('/profile')
def profile():
    time.sleep(30)
    return jsonify(datetime.now().second), 200,
{'Access-Control-Allow-Origin': '*'}


if __name__ == "__main__":
    app.run(threaded=False, debug=True)
```

index.html:

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>Document</title>
  </head>
  <body>
    <script>
      function test() {
        const res1 = fetch('http://127.0.0.1:5000')
        const res2 = fetch('http://127.0.0.1:5000/profile')
        res1
          .then((res) => {
            return res.json()
          })
          .then((data) => {
            console.log(data)
          })

        res2
          .then((res) => {
            return res.json()
          })
          .then((data) => {
            console.log(data)
          })
      }
      test()
    </script>
  </body>
</html>
```

If a user visits index.html at 10:10:10 AM (in format HH:MM:SS), what will be logged in the console (approximately)?

A. 20, 50

B. 20, 40
C. 40, 40
D. 50, 50

Answer: A

Solution: The flask application is being run in the non-threaded mode. This means that the application will not accept any new request before it finishes processing the previous request. Thus, if 2 requests are sent at the same time (10:10:10), firstly, the route “/” will take a minimum of 10 seconds to respond, and then the next request will be accepted, and the route “/profile” will take a minimum of 30 seconds before responding.

Q3: Which of the following statements is/are true about Redis?

A. Data is always manipulated in the main computer memory (RAM).
B. It stores data on the disk by default.
C. Data cannot be reconstructed back.
D. Data is stored as key value pairs.

Answer: A and D

Answer: Redis is an in-memory database, which can be made to store data in the disk, and it stores the data in the form of key-value pairs, but it stores the data in main-memory by default. Even, if the data is stored in the disk, the data needs to be brought to the main memory for manipulating it.

Q4: Which of the following is/are true for Redis?

A. Redis supports multiple data types.
B. It is possible to set expiration time of data.
C. Authentication features are available.
D. TTL value of 1 implies that data will not expire.

Answer: A, B and C

Solution: Redis is an in-memory database, which supports various data types. It is always possible to store data with timeouts. The TTL value of -1 indicates that the data is not meant to be expired. It is also possible to have an authentication layer with Redis.

Q5: Which of the below is true for Celery?

A. Celery systems can consist of multiple brokers.
B. Workers and clients will automatically retry if connection is lost.
C. Redis and RabbitMQ are the only supported brokers with Celery.
D. It has features to monitor and schedule tasks.

Answer: A, B and D

Solution: Celery is a python based task queue, which provides multiples workers to perform heavy computed job asynchronously. It generally retries certain times in case of a connection loss. Celery supports brokers other than Redis and RabbitMQ, like Amazon SQS. It is always helpful for scheduling tasks or jobs.

Q6: Which of the following statements is false regarding concurrency?

A. The concurrency is achieved by executing multiple tasks overlapping in time periods, but not always simultaneously.
B. The concurrency can be achieved in both “single-core” and “multi-core” environments.
C. The concurrency is always achieved via “context switching” in a multicore environment.
D. Concurrency and Parallelism are not mutually exclusive.

Answer: C

Solution: Concurrency is the notion of carrying out multiple tasks or threads overlapping in time periods or even simultaneously (at the same time), not necessarily always. Concurrency is achieved via “context-switching” in a single core environment, whereas, the same can be achieved using “parallelism” in a multicore environment.

Q7: Which of the following statements is false regarding concurrency and parallelism?

A. The parallelism can be achieved in a “multi-core” environment.
B. It is possible to achieve parallelism without concurrency.
C. The parallelism is achieved by executing multiple tasks simultaneously.
D. It is possible to achieve concurrency without parallelism.

Answer: B

Solution: Kindly refer to this thread on discourse: [W9 Graded Assignment : Q7 - Courses / Modern Application Development - II - IITM-PDS]

Q8: Which of the following statement(s) is/are correct regarding message queue?

A. Message queue provides communication between components in a distributed architecture.
B. It provides a buffer which temporarily store messages.
C. In general, a message in message queue is processed more than once in point-to-point messaging.
D. In general, a message in message queue can be processed only once in point-to-point messaging.

Answer: A, B, D

Solution: Message queue provides a means of communication between different components in a distributed network. It does provide a buffer to store temporary messages. A message in a message queue is processed or consumed by only one consumer (in point-to-point messaging).

Q9: Which of the following is/are the true regarding message broker?

A. A message broker helps in reducing the number of connections between servers, if compared with many-to-many connections between servers.
B. It does not allow easier scalability, if compared with many-to-many connections between servers.
C. It can be used for batch processing of many similar types of requests.
D. All of the above

Answer: A and C

Solution: A message brokers provides easier scalability, if compared with many-to-many connections. The simple reason is that the addition of a new component or a server need not be connected to all other servers in the network, and it just needs to connected with the message broker. It is also useful for batch processing, where multiple requests of similar types and collected and processed together.

Q10: Which of the following is/are true regarding message broker?

A. They serve as intermediary between applications, which allow senders to issue a message without knowing the state of the receiver.
B. They serve as intermediary between applications, which allow senders to issue a message only after knowing the state of the receiver.
C. Inter application communication is often asynchronous.
D. Inter application communication is often synchronous.

Answer: A and C

Solution: A sender sending a message need not worry about the present state of the receiver, if the communication is happening using a message broker, as the message broker receives the message from the sender and inform the receiver once it is available to take new requests. This communication is generally asynchronous, as the sender need not wait for the receiver to accept the previous request before sending anew request.

Week 10 Graded Questions

Q1: Which of the following statement(s) is/are true regarding Webhooks?

A. A Webhook should give an immediate response.
B. A Webhook cannot post form encoded data.
C. A Webhook cannot be secured, if the receiver URL to be accessed is publicly available.
D. A Webhook receiver can be a trivial flask application.

Answer: A and D

Solution: A Webhook is meant to work synchronously and should result in an instant response. Webhooks can post JSON as well as form-encoded data.
A public receiver URL can be secured using security tokens to authenticate the request. A trivial flask application (with at least 1 endpoint) can easily act as webhook receiver.

Q2: Which of the following statement(s) is/are correct, regarding Webhooks and Web sockets?

A. A Webhook uses HTTP protocol in general.
B. A web socket is used to achieve 2-way communication.
C. A Webhook is primarily used for server-to-server communication.
D. A Webhook generally keeps the connection open.

Answer: A, B and C

Solution: Webhooks use HTTP protocol to post messages. A web socket is meant to be used to achieve 2-way communication between the client and the server, where the connection between the two is left open, unlike webhooks.
A webhook is primarily used for server-to-server communication because the receiver URL also belongs to an application, which is responding to the requests it gets.

Q3: Which of the following statement(s) is/are true regarding polling and long poll?

A. In Polling, a client make repetitive calls to the server at fixed time intervals.
B. Polling requires a persistent connection to the server.
C. In long polling, the connection is kept open till the response is sent.
D. All of the above

Answer: A and C

Solution: Polling (i.e., short polling) is a process in which a client repeatedly requests the information from the server at a fixed time interval. It doesn't require a persistent connection, as the client makes request to the server again and again.
While, in long polling, the client makes a request to the server, and waits or keeps the connection open till the server responds.

Q4: Which of the following statements is false regarding server sent events?

A. It is a mechanism at the server to push events to the client.
B. It requires a task queue to be maintained.
C. It requires the service workers on the client.
D. The service workers should run in the background.

Answer: B

Solution: Server sent events are used to push information to the clients, which are currently connected via a sse stream, which is meant to be an infinite stream. It requires the service workers to be running in the background to ensure a smooth flow of information from the server to the client.

Q5: Which of the following statement(s) is/are true regarding short polling?

A. The client sends a request to the server, and the server responds to the client immediately, even if it does not have data.
B. The client sends a request to the server, and the server waits to respond to the client if it does not have data and keep the connection open.
C. Request is sent after fixed delay periodically.
D. All of the above

Answer: A, C

Solution: Polling (i.e., short polling) is a process in which a client repeatedly requests the information from the server at a fixed time interval.

Q6: Which of the following is true regarding long polling?

A. The client sends a request to the server, and the server responds to the client even if it does not have data.
B. The client sends a request to the server, and the server waits to respond to the client if it does not have data and keep the connection open.
C. Request is sent after fixed delay periodically.
D. The client must send the next request immediately after it receives the response.

Answer: B

Solution: In long polling, the client makes a request to the server, and waits or keeps the connection open till the server responds.
It is not at all must or required to send the request new request to the server as soon as it responds, and depends on the application.

Q7: Which of the following statement(s) is/are correct?

A. A Webhook pushes the messages to an application.
B. In general, webhook pulls the messages from an application, but an API pushes the messages to an application.
C. Webhooks are generally synchronous in nature.
D. An API can pull and push the messages from/to an application.

Answer: A, C and D

Solution: An API is capable of both pushing (POST, PUT etc.) and pulling (GET) messages from/to an application. A Webhook is meant to work synchronously, and it usually pushes (POST) messages to an application.

Q8: Which of the following is correct regarding pub/sub messaging?

A. It can be used to enable event driven architecture.
B. Any message published to a topic should be immediately received by the subscriber(s) of the topic.
C. The communication is synchronous in nature.
D. The communication is asynchronous in nature.

Answer: A, B and D

Solution: Reference to read: [What is Pub/Sub Messaging? (amazon.com)]

Q9: What is the role of the message broker?

A. It ensures that the message gets delivered to the correct recipient.
B. If the message is undelivered in the first try, it fails and doesn’t retry.
C. It ensures that the messages are processed in LIFO manner.
D. It can send messages to multiple recipients, if required.

Answer A and D

Solution: A message broker performs various functions. It makes sure that the message is delivered only to that recipient, it is meant for, and can be configured to deliver messages to multiple recipients. It generally processes the messages in FIFO (First-In-First-Out) order.

Q10: Which of the statement(s) is/are true for webhook?

A. The message content is sent as part of the request body.
B. The message content in the response body should be detailed.
C. Webhooks are usually invoked by the human clients.
D. Webhooks are one way messages.

Answer: A and D

Solution: A Webhook typically pushes (POST) the messages to an application, and the content is sent as part of the request body. A webhook response should indicate the status code of the request, and should have minimal or no data in the response body.
Webhooks are meant for one way messaging (i.e., the connection is not kept open), and they are typically invoked by machine clients, when the event a webhook is configured for, is triggered.

Week 11 Graded Questions

Q1: Which of the following statements(s) is/are true regarding flask_caching?

A. The cache decorator includes the function parameters in the cache key.
B. The memoize decorator includes the function parameters in the cache key.
C. Both the cache and memoize decorators work the same way for functions not having any parameters.
D. All of the above

Answer: B and C

Solution: Reference to read: Flask-Caching — Flask-Caching 1.0.0 documentation

Q2: Which of the following statement(s) is/are correct regarding caching?

A. Caching can be done at browser level.
B. Caching cannot be done at server (proxy) level.
C. Hard refreshing a web page clears the browser cache for that specific web page.
D. All of the above

Answer: A and C

Solution: Caching can be done at various levels, like browser level, proxy server level, database level etc. A hard refresh clears the browser's cache, and forces the browser to load or get the most recent version of the web page from the server.

Q3: Which of the following statement(s) is true?

A. A cached resource stored at a proxy server serves its copy to multiple users.
B. Hard refreshing a web page can still result in a cached response being served from a proxy server.
C. Caching at browser level provides the least latency, if compared with proxy level.
D. All of the above

Answer: D

Solution: A resource being cached on the proxy server is meant to be served to multiple users, without hitting the origin or the database server. A hard refresh forces the browser to get the most recent version of the web page, but it can still be a cached version available at the proxy server.
The cached resources that are available on the client's browser provides the least latency, as the browser need not even make a network call to fetch the resource.

Q4: Which of the following criterions should be directly considered while measuring the performance of an application?

A. The number of requests an application can handle for a given time interval with as much as little latency.
B. How does the application scale over load.
C. The application should be written in Python based flask or Django framework.
D. All of the above

Answer: A and B

Solution: It is not at all mandatory for an application to be written in flask or Django to provide a good performance.

Q5: Which of the following tools can be used to measure or analyze the performance of a web application?

A. GTmetrix
B. Postman
C. Lighthouse
D. All of the above

Answer: A and C

Solution: Both Lighthouse and GTmetrix are the tools that can be used for performance measurement of a website or a web application, whereas, Postman is a web client, majorly used for testing the APIs.

Q6: Which of the following is the correct statement?

A. Cookie size can impact the website performance.
B. Cookie size does not affect the website performance.
C. A request to static.example.com will include cookies set for example.com (assuming domain attribute is specified with “example.com”).
D. A request to static.example.com will not include cookies set for example.com (assuming domain attribute is specified with “example.com”).

Answer: A and C

Solution: A cookie set with the domain will be sent to the server with each request. So, unnecessary cookie will create unnecessary traffic. And cookie with large size may slow down the website because of the requirements of higher bandwidth.

Q7: Suppose you have a web page which includes a paragraph with some content, images and a stylesheet. You want to update the content of the paragraph with JavaScript. Which of the event, among load and DOMContentLoaded, you should wait to fire before updating the content for better performance?

A. load
B. DOMContentLoaded
C. Both of them will result in same performance.
D. DOM is not required to be loaded fully before updating the content.

Answer: B

Solution: The load event is fired when every thing including images and stylesheets are fully loaded, but DOMContentLaded is fired as soon as the HTML is loaded without waiting for stylesheet and images to load. In order to change the content of paragraph only DOM is required so, DOMContentLoaded will result in better performance.

Q8: Suppose you want to embed an image of 100*100 px in your webpage. Which of the following statement is correct?

A. Using an image with dimension 200*200 px and scaling it using HTML will perform better than using a 100*100 px image directly.
B. Using an image with dimension 200*200 px and scaling it using HTML will perform worse than using a 100*100 px image directly.

C. Scaling using HTML does not impact the performance.
D. None of the above

Answer: B

Solution: Using the exact size image as required gives the better performance, as the resize of the image or scaling may affect the performance of a web page by some margin.

Q9: Which of the following is/are correct regarding the E-Tag and If-Match?

A. E-Tag is a response header to validate the current version of resource.
B. E-Tag is a string of ASCII characters placed between double quotes.
C. For get and head requests, If-Match request header is sent to get the resource only if E-Tag value matches.
D. For get and head request, If-Match request header is sent to get the resource only if E-Tag value does not match.

Answer: A, B and C

Solution: E-Tag is an HTTP response header, which acts as an identifier for a particular version of the resource. A new E-Tag must be generated if the resource at a specified URL changes. You can read more about it here: ETag - HTTP | MDN (mozilla.org)

Q10: Which of the following is/are true regarding CDN?

A. CDN is an acronym for content delivery network.
B. It is the network of servers to provide fast delivery of web content.
C. It reduces the distance between client and the resource, which results in faster delivery of content.
D. None of the above

Answer: A, B and C

Solution: CDN (Content Delivery Network) is mainly used to deliver the static content with a high speed. The content is served from the nearest CDN server to the user to provide high performance.