Week 5 Practice Questions

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

A. A non async function cannot use fetch API in its body.
B. An async function always returns a promise.
C. Fetch API can be used to make HTTP DELETE requests.
D. Fetch API does not allow modifications in the request headers.

Answer: B and C

Solution: An async function always returns a promise, and a fetch API call can be used inside a non async function with causing any errors. The fetch API supports various HTTP request methods, which includes GET, POST, PUT, DELETE etc.

Q2: Which of the following statement(s) is/are true regarding Vue lifecycle hooks?

A. The lifcycle hooks are triggered implicitly, depending on the state of the component.
B. The created() hook is invoked when the reactive data properties have been set up.
C. The lifecycle hooks have to be triggered by the developer explicitly.
D. The lifecycle hooks cannot be async.

Answer: A and B

Solution: The VueJS framework provides multiple lifecycle hooks, which are invoked implicitly (by the framework), at certain points, depending upon what state the component is at. The lifecycle hooks can be made async. The reactive properties of the Vue are set up by the time, the created() hook is invoked. Hence, it has the access to the reactive properties.

Q3: Which of the following statement(s) is/are correct regarding APIs?

A. An API is designed to separate frontend and backend.
B. All the APIs must comply with the REST specifications.
C. Multiple different interfaces (desktop, mobile etc.) can use the same API implementation.
D. All of the above

Answer: A and C

Solution: An API (Application Programming Interface) is used to decouple the frontend from the backend so that both can be built independently. This way, a single API implementation can be used for various UIs. REST is an architectural style, but all APIs need not comply with its specifications.

Q4: Which of the following statement(s) is/are true regarding fetch API?

A. The promise returned by fetch does not reject for HTTP status code 400.
B. The promise returned by fetch only reject on network failure or if anything prevented the request from completing.
C. The option “credentials : ‘omit’” ensures that no cookies are sent with the request.
D. All of the above

Answer: D

Solution: Reference:[Using the Fetch API - Web APIs | MDN (mozilla.org)]

Q5: Consider the following JavaScript program, and predict the output if executed.

```
new Promise((resolve, reject) => {
   function some(a, b){
       if (a == b){
           resolve("Matched");
       }
       else if (a.toString() === b){
           reject("Partial Match");
       }
       else{
           reject("Unmatched");
       }
   }

   some("abhi", "abhi");
}).then(pass => console.log("Pass Summary:", pass),
fail => console.log("Fail Summary:", fail));
```

A. Pass Summary: Matched
B. Pass Summary: Partial Match

C. Fail Summary: Partial Match
D. Fail Summary: Matched

Answer: A

Solution: The promise accepts two callbacks (resolve and reject), and the resolve callback is invoked as the condition specified is met. Thus, the first function in the then block is executed.

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

```
function foo(x) {
  const pr = new Promise((resolve, reject) => {
    if (x) {
      resolve(2)
    } else {
      reject(3)
    }
  })
  return pr
}
foo(false).then(
  (p) => {
    console.log(p)
  },
  (q) => {
    console.log(q)
  }
)
```

A. 2
B. 3
C. undefined
D. Null

Answer: B

Solution: Promise constructor takes two callback functions as argument, first argument is called when the promise is resolved and second argument is called when the promise is rejected. Because we are passing false as an argument, then the promise will be rejected. So will be handled by the second callback function of then method so, and option B is correct.

Q7: Which of the following is correct regarding catch() method in promise?

A. The callback function passed as argument in the catch is used to handle the rejection.
B. The callback function passed as argument in the catch is used to handle the resolution of the promise.
C. The catch method takes two callback functions as parameters.
D. None of the above

Answer: A

Solution: The call back function passed as parameter is catch method will be invoked when the promise is rejected.

Q8: Which of the following is/are true regarding then(handler1, handler2) method in promise?

A. handler1 is called when a promise is rejected.
B. handler2 is called when promise is rejected.
C. If there is only one handler in then method, then it will be invoked when promise is rejected.
D. If there is only one handler in then method, then it will be invoked when promise is resolved.

Answer: B and D

Then the method takes two callback functions as an argument, the first callback function handles the success and the second handles the failure. So, option B is correct. If only one callback function is passed to the then method, then it will be invoked when the promise is resolved