# 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:

```
<!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:

```
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.

## 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}}°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
° or °

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,

```
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

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?

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