Aryan Rao

aryanrao@iastate.e

du

March 7th, 2023

Activity 10

Q3

```
> function sleep(t) {
      return new Promise((resolve, reject) => {
          console.log("running promise");
          setTimeout(() => {
              console.log("running timer");
              resolve()
          }, t)
      })
  }
  sleep(3000);
  running promise
  ▼Promise {<pending>} i
    ▶[[Prototype]]: Promise
     [[PromiseState]]: "fulfilled"
     [[PromiseResult]]: undefined
  running timer
>
```

We have a function called sleep with a parameter of 3000 milliseconds.

Running promise is the first statement to be executed once the Promise object is created, as seen in the console above. After that, a method named setTimeout is called. We first print "running timer" in this method before calling the resolve() function. T, which is essentially 3000, serves as the setTimeOut function's second parameter. As a result, after three seconds, the statement "running timer" is carried out, and the promise is fulfilled.

Q4

```
> function nosleep(t) {
      return new Promise((resolve, reject) => {
          console.log("running promise");
          setTimeout(() => {
              console.log("running timer");
              reject(new Error("Whoops!"));
          }, t);
      });
  }
  nosleep(3000);
running promise
Promise {<pending>} i
  ▶ [[Prototype]]: Promise
    [[PromiseState]]: "rejected"
  ▶ [[PromiseResult]]: Error: Whoops! at <anonymous>:6:20
running timer
⊗ ▶Uncaught (in promise) Error: Whoops!
      at <anonymous>:6:20
```

We have a function called nosleep that takes a parameter of 3000 milliseconds.

Running promise is the first statement to be executed once the Promise object is created, as seen in the console above. After that, a method named setTimeout is called. Printing "running timer" and calling the reject() function with the error object message "Whoops!" are both done within this procedure. T, which is essentially 3000, serves as the setTimeOut function's second parameter. As a result, the promise is denied after three seconds and the statement "running timer" is executed, displaying the message "Whoops" in the console.

Q5

5.1

A ".then" method is available on the promise object. Here, only this has changed.

In order to access the "Value" or "Error" directly depending on whether the promise is fulfilled or not, the ".then" method is utilized. When the promise is fulfilled and the value is received, or when the promise is denied and the error is received, the two parameters are functions that are called.

5.2

Calling the sleep function with a 3000ms argument

This function starts by creating a setTimeOut function with the arguments resolve, a function parameter, and t. The mypromise.then method, which receives the resolve value as a

result, is

then displayed. When we call sleep, the message "Success!!" appears since its parameter is set to t = 3000. After 3 seconds, the statement in the myResolve method, "Promise waited 3000ms," appears as seen above.

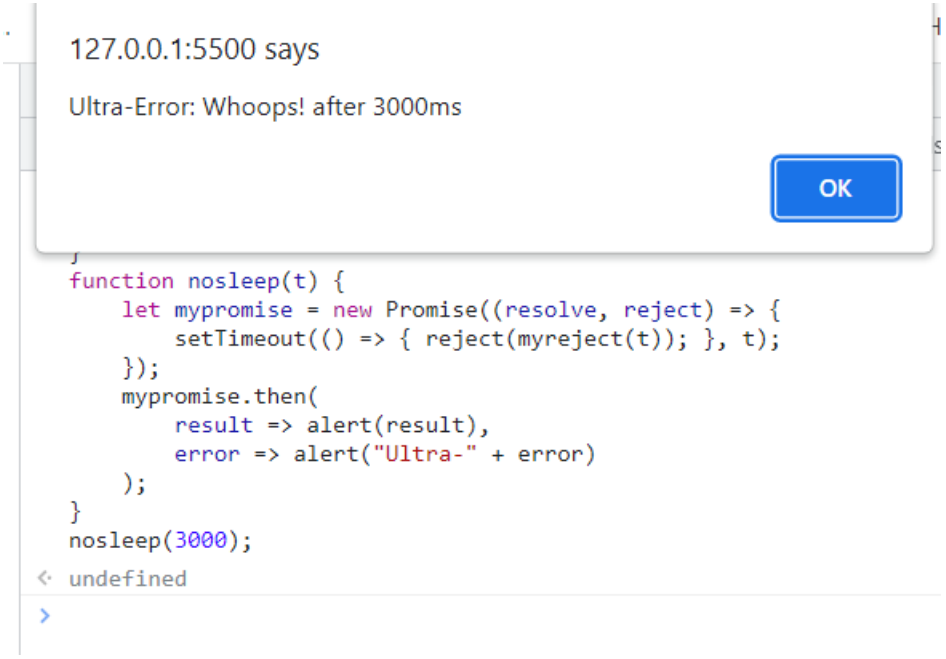
Q6

6.1

On the promise object, there is a ".then" method. Here, this is what has changed.

Depending on whether the promise is accepted or refused, we can access the "Value" or "Error" directly by using the ".then" function. The two arguments are actions that are taken when either the promise is accepted and the value is received, or when it is refused and the error is received.

6.2



Calling the sleep function with a 3000ms argument

The setTimeOut function is formed at the beginning of this function with the arguments as a reject with a function parameter and t. The mypromise.then method, which accepts the error message as an error, is then displayed. The warning "Ultra-Error: Whoops! after 3000ms"—which is the statement in the myReject function—pops up after 3 seconds since sleep contains a parameter t = 3000 when we use it, as can be seen above.

Q7

Code :

```
function myresolve(t){
   return "Sucess!! Promise waited ["+t+"] ms";
}

function sleep (t) {
   let mypromise = new Promise((resolve, reject) => {
     setTimeout(() => {resolve(myresolve(t))}, t)
   });
   mypromise.then(
      result => {
         let container = document.getElementById("promisehere");
         let div = document.createElement("div");
         div.innerHTML= result;
         container.appendChild(div);
      },
      error => alert(error)
   );
```

```
}


sleep(3000);
```

Output: (Shows up after 3 seconds.)