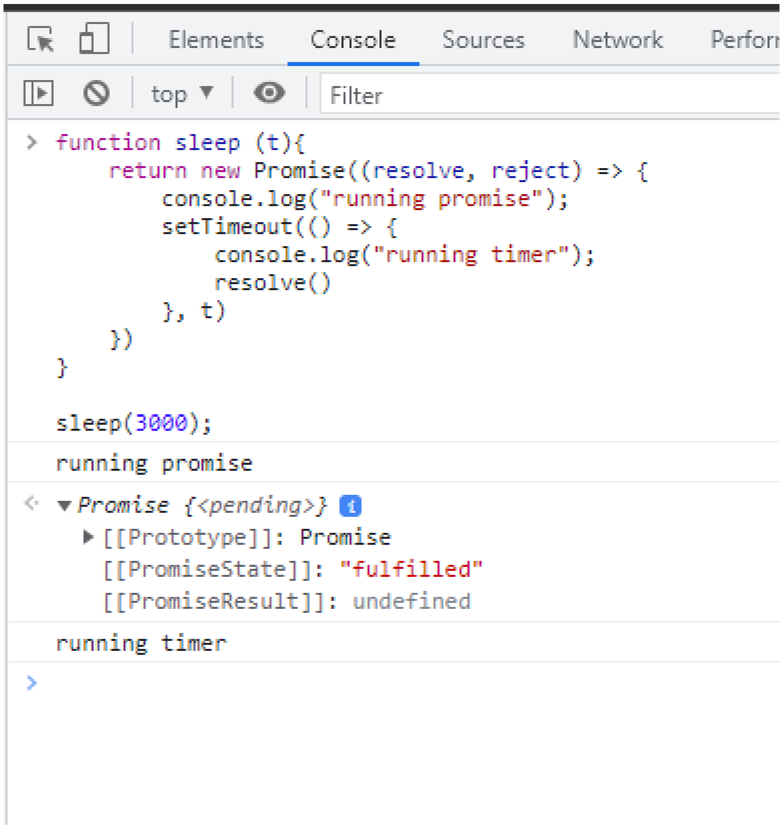
William Schulte

March 3, 2023

Activity 10

Exercise 3 –

The function is running our promise and testing whether or not it works, of which it does

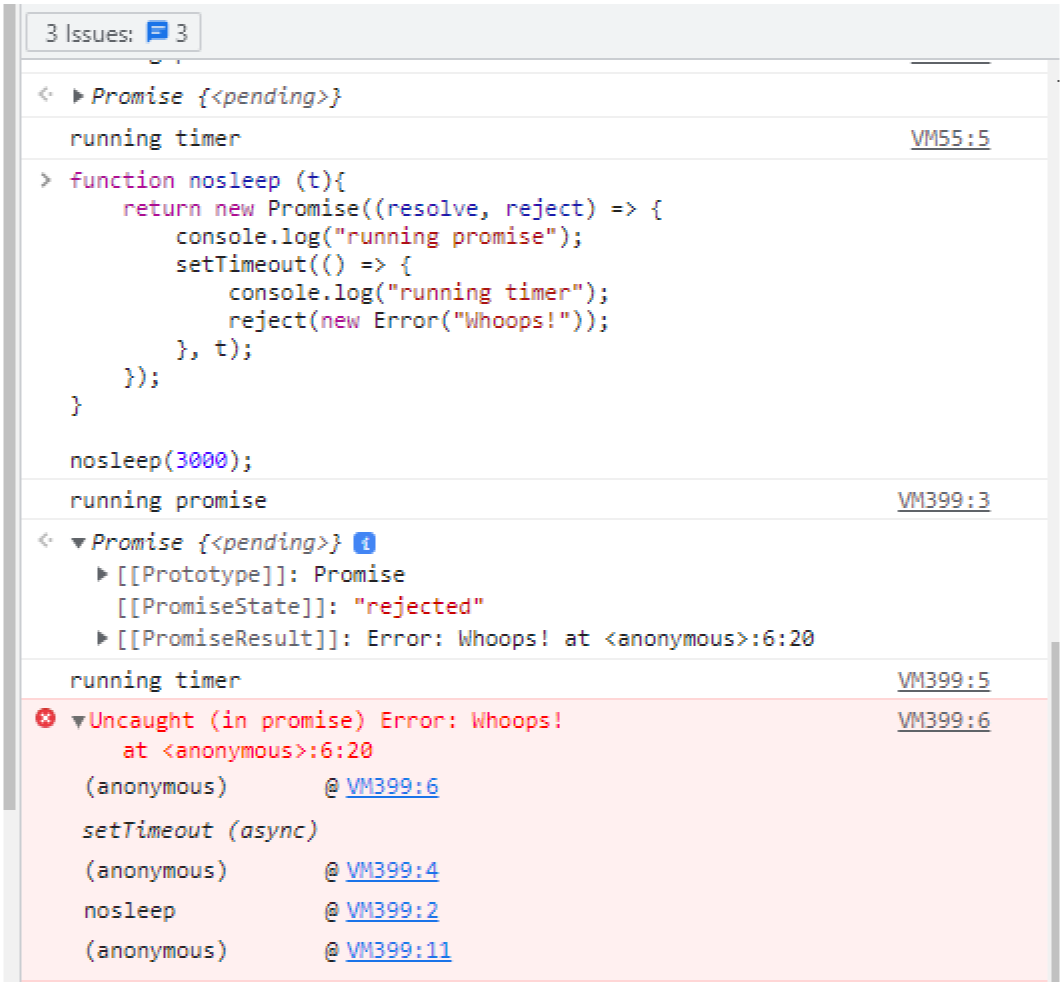


```
> function sleep (t){
      return new Promise((resolve, reject) => {
          console.log("running promise");
          setTimeout(() => {
              console.log("running timer");
              resolve()
          }, t)
      })
  }

  sleep(3000);
  running promise
<· ▼Promise {<pending>}
    ▶[[Prototype]]: Promise
     [[PromiseState]]: "fulfilled"
     [[PromiseResult]]: undefined
  running timer
>
```

Exercise 4 –

Similar as above, the program runs the promise only this time we throw an error throught, causing a runtime conflict

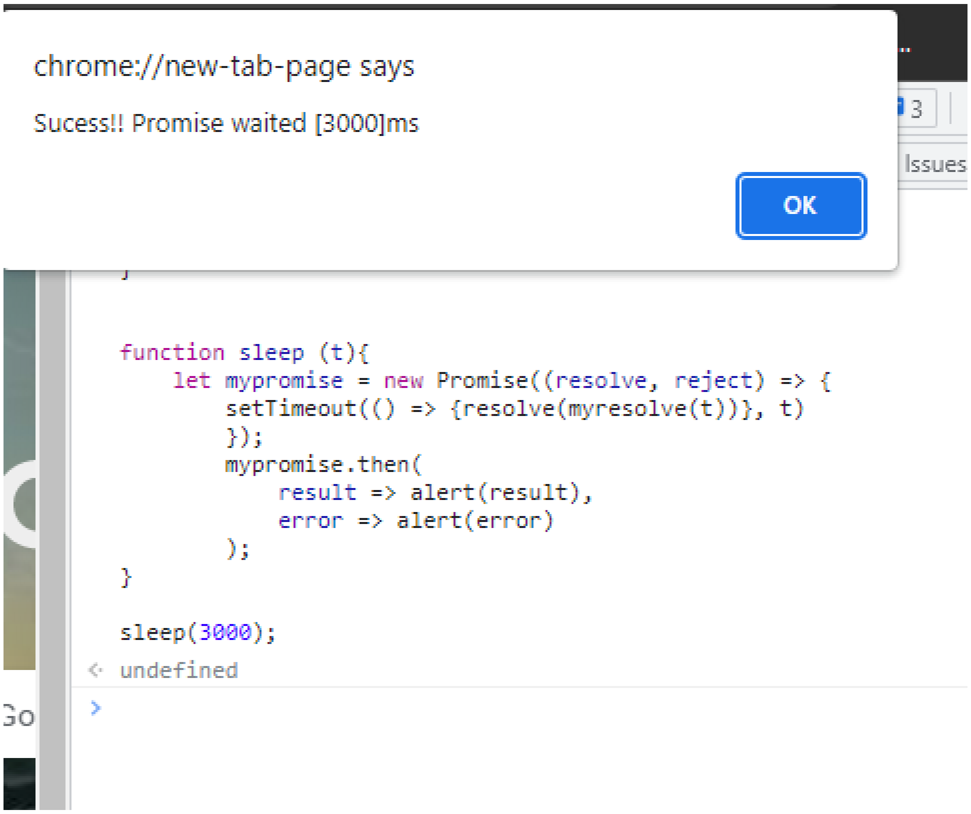


```
< ▶ Promise {<pending>}
  running timer                                                    VM55:5
> function nosleep (t){
      return new Promise((resolve, reject) => {
          console.log("running promise");
          setTimeout(() => {
              console.log("running timer");
              reject(new Error("Whoops!"));
          }, t);
      });
  }

  nosleep(3000);
  running promise                                                 VM399:3
< ▼ Promise {<pending>}
    ▶ [[Prototype]]: Promise
      [[PromiseState]]: "rejected"
    ▶ [[PromiseResult]]: Error: Whoops! at <anonymous>:6:20
  running timer                                                   VM399:5
⊗ ▼Uncaught (in promise) Error: Whoops!                           VM399:6
      at <anonymous>:6:20
   (anonymous)        @ VM399:6
   setTimeout (async)
   (anonymous)        @ VM399:4
   nosleep            @ VM399:2
   (anonymous)        @ VM399:11
```

Exercise 5 –

It gives us a popup notification for it, only using a then statement within giving it a delayed response to when it should appear.



chrome://new-tab-page says

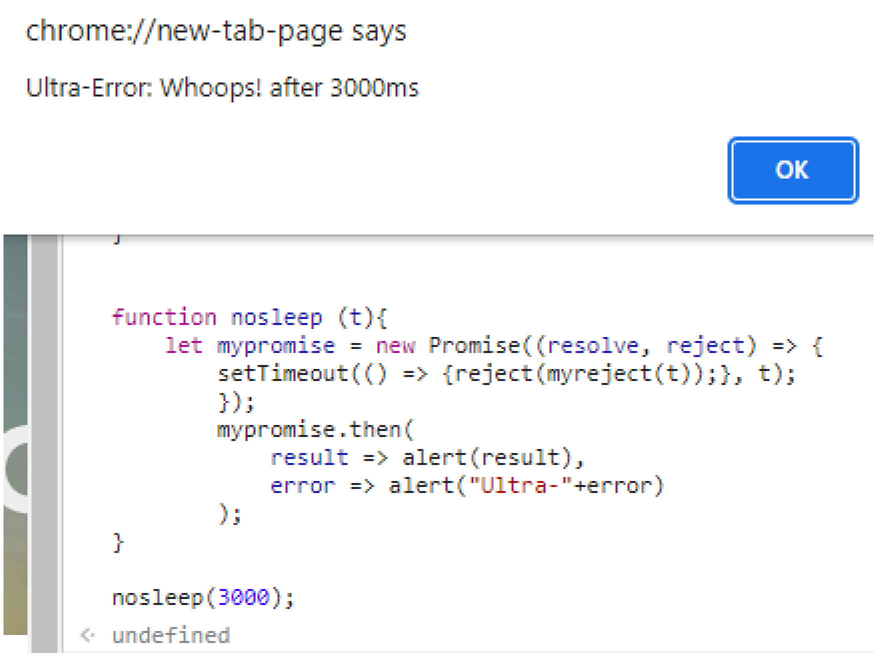
Sucess!! Promise waited [3000]ms

```
function sleep (t){
    let mypromise = new Promise((resolve, reject) => {
        setTimeout(() => {resolve(myresolve(t))}, t)
        });
        mypromise.then(
            result => alert(result),
            error => alert(error)
        );
}

sleep(3000);
<· undefined
>
```

Exercise 6 –

Same as above, only this time displaying an error message.



chrome://new-tab-page says

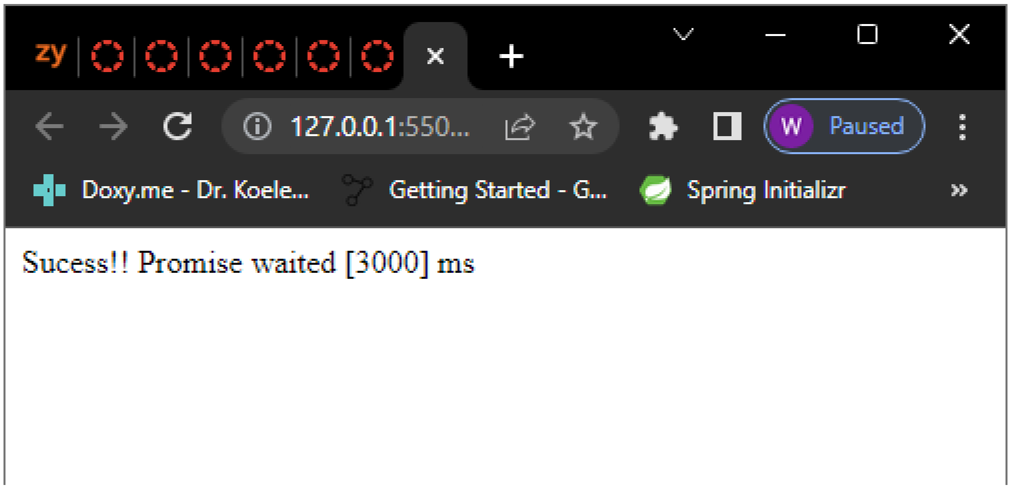
Ultra-Error: Whoops! after 3000ms

```
function nosleep (t){
    let mypromise = new Promise((resolve, reject) => {
        setTimeout(() => {reject(myreject(t));}, t);
        });
        mypromise.then(
            result => alert(result),
            error => alert("Ultra-"+error)
        );
}

nosleep(3000);
<· undefined
```

Exercise 7 –

We are essentially doing what we have been doing only this time we actually have out the code within a .js file.



Sucess!! Promise waited [3000] ms

(I also included my code for this as well, so that I have all my corners covered in regards to this exercise)