香港中文大學
The Chinese University of Hong Kong

MORE ON PROMISE AND FETCH

ESTR2106 2022–23 Term 1

Building Web Applications

Dr. Chuck-jee Chau
chuckjee@cse.cuhk.edu.hk

OUTLINE

- Promise with async/await
- Waiting for variables
- Try/catch/finally
- More on fetch()
- Waiting for some Promises together

ASYNC/AWAIT

- The Promise chain of **`then()`** is a bit confusing in syntax
- The new keywords **`async`** and **`await`** deals with promises
  - **`async`** makes a function return a **`Promise`**
  - **`await`** makes a function wait for a **`Promise`**
    - *Execution not going on until Promise is ready!*
- *See: https://www.w3schools.com/js/js_async.asp*

```
async function myFunction() {
  return "Hello";
}
```

the same as

```
function myFunction() {
  return Promise.resolve("Hello");
}
```

THE ORIGINAL PROMISE

https://codepen.io/chuckjee/pen/eYNwbBK

```
function waitnprint(str) {
  return new Promise((resolve, reject) => {
    setTimeout( function() { // ...wait for a while...
      console.log(str);
      resolve();
    }, 1000);
  })
}
```

```
waitnprint("Hello")
  .then(()=>waitnprint("World"))
  .then(()=>waitnprint("!"))
  .then(()=>waitnprint("END"))
  .catch((err)=>{...});
```

PROMISE WITH ASYNC/AWAIT

https://codepen.io/chuckjee/pen/ZEGdVLo

```
// same Promise from before
function waitnprint(str) {
  return new Promise((resolve, reject) => {
    setTimeout( function() { // ...wait for a while...
      console.log(str);
      resolve();
    }, 1000);
  })
}
```

```
async function longwait() {
  await waitnprint("Hello");
  await waitnprint("World");
  await waitnprint("!");
  await waitnprint("END");
}
longwait();
```

Note: **await** *must be inside an* **async** *function*

WAITING FOR VARIABLES

https://codepen.io/chuckjee/pen/gOXWMmq

```
function waitfornum(x) {
  return new Promise((resolve,reject) => {
    setTimeout(() => {
      console.log("Waiting for 2 seconds, with x="+x);
      resolve(x);
    }, 2000);
  });
};

let add = async x => {
  let a = waitfornum(10); // OR await waitfornum(10)
  let b = waitfornum(20); // OR await waitfornum(20)
  return x + await a + await b; // then no await here
}; // the return is a Promise

add(5).then(sum => console.log(sum));
```

- Besides waiting for function returns, it is also possible to wait for a variable to be ready
- Whether **await** is at the ***function return*** or the ***variable*** shows a small difference

TRY-CATCH-FINALLY IN ASYNC FUNCTIONS

- In an **async** function, the try/catch/finally in a Promise chain can be defined this way

```
async function f() {
  try {
    /* some actions */
    await something; // waiting for something
    /* some actions */
  } catch(error) {
    /* error handling */
  } finally {
    /* some actions regardless successful or not */
  }
}
```

FETCH() WITH ASYNC/AWAIT

- Fetch returns a Promise, that can be handled using async/await

```
fetch('https://www.cse.cuhk.edu.hk/~chuckjee/csci2720.json')
// parsing retrieved data as JSON
.then(res=>res.json())
// displaying data
.then(data=>console.log(data))
```

the same as

```
let getjson = async () => {
    let response = await fetch(
        'https://www.cse.cuhk.edu.hk/~chuckjee/csci2720.json');
    let data = await response.json();
    return data;
}
getjson().then(text=>console.log(text));
```

FETCH REQUEST AND RESPONSE

- The `fetch()` request and its headers can be manually built
  - For customizing HTTP method, headers, etc.
  - *See: https://developer.mozilla.org/en-US/docs/Web/API/Request/Request*
- The `fetch()` response is an object which has many properties and methods, e.g.:
  - ***`Response`***`.ok`: true if the request was successful (with status 200–299)
  - ***`Response`***`.status`: the HTTP status code
  - ***`Response`***`.statusText`: the status message
  - *See: https://developer.mozilla.org/en-US/docs/Web/API/Response*

WAITING FOR SOME PROMISES TOGETHER

- Sometimes, multiple resources are required, and the order of obtaining them is not important
  - It waits for all fulfilments, or the first rejection

```
const promise1 = Promise.resolve(3);
const promise2 = 42;
const promise3 = new Promise((resolve, reject) => {
  setTimeout(resolve, 100, 'foo');
});

Promise.all([promise1, promise2, promise3]).then((values) => {
  console.log(values);
}); // [3, 42, "foo"]
```

- *See: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise/all*

READ FURTHER…

async/await on javascript.info

https://javascript.info/async-await

How to Use Fetch with async/await

https://dmitripavlutin.com/javascript-fetch-async-await/