



Asynchronous Control Flow

Module 1 Week 4

Notes Repo: <https://github.com/C-Shi/lhl-flex-lecture>



Learning Objectives

Recap: Higher order function / Callbacks

Synchronous and Asynchronous programming

Async control flow with `setTimeout` and `setInterval`

Node.js File System

JavaScript Events



Higher Order Func / Callback

- A function that is passed to another function is a **Callback**
- The function that **takes in a callback**, or returning a function is called higher-order function
- Higher order functions allow us to write reusable code that **operates on action**

```
const first = () => { /* do something */ };  
  
const second = (func) => { func() };  
  
[1, -2, 3, -4].filter((x) => x > 0);
```

Synchronous Programming

JavaScript can only do one thing at a time. No exception

Synchronous JavaScript code will execute in the same sequence as you write

An operation that takes long time to finish can potentially block the other code

Even worse if an operation may or may not happens, which can block forever



So what is the solution?

Asynchronous JavaScript



Call Stack



Web API



Task Queue



Event Loop

1. JavaScript provide asynchronous programming ability
2. JavaScript can queue up asynchronous code to allow non-blocking code to finish
3. Async codes are usually written in the form of Callback or Promise
4. Not all Callback are asynchronous

```
loadPageHeader();  
  
getVancouverWeather(showVancouverWeather);  
  
getTorontoWeather(showTorontoWeather);  
  
loadPageFooter();
```

Call Stack

```
showTorontoWeather()
```

Task Queue

```
showVancouverWeather
```

```
showTorontoWeather
```

Terminal

Show Page Header

Getting Vancouver Weather...

Getting Toronto Weather...

Show Page Footer

Vancouver: Sunny 20C

Toronto: Cloudy 15C



Quizzes

Are these operation synchronous or asynchronous in JavaScript ?

Read data from Database	Asynchronous
Toggle an list by clicking on a button on the page	Asynchronous
Sorting a long list using Array.sort() method	Synchronous
Build a browser timer/stopwatch	Asynchronous
Deep search for an value in a nested object	Synchronous



setTimeout and setInterval

setTimeout is an asynchronous functions that execute a callback after timer expires

setInterval is an asynchronous functions that repeatedly executes a callback with a fix interval

Timer is the minimum guaranteed time

```
setTimeout(() => {  
  console.log('After 3 second')  
}, 3000);  
  
setInterval(() => {  
  console.log('Nice to see you again');  
}, 2000)
```


File system and **fs** module

Built in fs module to interact with file

Both synchronous and asynchronous

Reference to official [Doc](#)

```
const fs = require('fs');

fs.readFile('./file.txt', 'utf-8', (err, data) => {
  console.log('Reading the file...');
  console.log(data)
})

fs.writeFile('./file.txt', 'Write something to file', (err) => {
  console.log('Finish Writing')
})
```



JavaScript Event

Event is an action that registered ahead of time, and can be called later

Event can called synchronously or **asynchronously**

Async Event is very powerful in DOM

```
const button = document.querySelector('button');  
// event can be asynchronously  
button.addEventListener('click', function() {  
  console.log('button click')  
})  
  
// event can also be trigger synchronously  
button.dispatchEvent(new Event('click'))
```