

# CSC20005: Lab 3

JS in more depth

Useful design patterns / things to know

# Objective

This is a quick primer into useful javascript design patterns.  
These concepts will be useful because you will see them  
everywhere in frontend work.

# Function chaining

Or why JQuery became popular

You could write this in plain vanilla JS:

```
var node = document.createElement("p");  
var textnode = document.createTextNode("Test");  
node.appendChild(textnode);  
document.getElementById("main").appendChild(node);
```

OR in JQuery

```
$("#main").append("<p>Test</p>");
```

Function chaining makes reading / writing code easier.

# Adding JQuery

Easiest is to import via script via a CDN (content delivery network)

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
```

Other methods: downloading and importing yourself

Install as a package, using some module bundler to manage (e.g. webpack)

# JSON

JSON (Javascript Object Notation), sample code snippet:

```
let classroom = [  
  {name: "Alfred", math: 90, english: 70 },  
  {name: "Betty", math: 85, english: 30 },  
  {name: "Cindy", math: 35, english: 60 }  
];
```

JSON — a collection of key / value pairs — has overtaken XML as a data structure for backend to communicate with frontend. You will see this everywhere.

Know how to access JSON **key / value pairs**.

# Anonymous functions

**Anonymous functions** are functions without names.

It is usually not accessible after its initial creation. Often seen as arguments to other functions.

```
classroom.forEach(function(ele) {  
    console.log(ele);  
});
```

OR you can use **arrow notation** (ES6):

```
classroom.forEach(ele => console.log(ele));
```

Arrow notation is a shorthand for anonymous functions. Note that the two methods are not exactly synonymous (e.g. function scope). Most use cases it does not matter though.

# AJAX, fetch, Axios

Different methods / design patterns of accessing HTTP resources, e.g. reading and writing (GET/POST) to a web API.

More detail: [jQuery AJAX](#), [fetch](#), [Axios](#)

For AJAX you can also use vanilla JS to write. Fetch is newer JS specification (ES6), and Axios is a popular promise-based HTTP client.

Simple code snippet for fetch:

```
fetch('https://api.data.gov.sg/v1/environment/psi')  
  .then(response => response.json())  
  .then(data => console.log(data));
```

# Callbacks, promises, async / await

Design patterns for handling the asynchronous nature of JS.

More detail: [Callbacks](#), [Promises](#)(ES6), [Async / Await](#) (ES6)

You will mainly encounter asynchronous errors typically when you do get / post operations.

For example:

```
let myData = {};  
fetch('https://api.data.gov.sg/v1/environment/psi')  
  .then(response => response.json())  
  .then(data => {myData = data});  
console.log(myData);
```

What is wrong with this code?



# Data manipulation - map, reduce, filter, etc.

Array functions for **handling data**(ES6). Or use a library like **lodash** for more use cases.

```
const words = ['spray', 'limit', 'elite', 'exuberant', 'destruction', 'present'];  
console.log(words.filter(word => word.length > 6));
```

# Assignment

## Lab 3: Carpark utilization table

# Assignment Lab 3: Setup

To be completed by **05-10-2020**(Mon) 2359hrs

- Fork the repo <https://github.com/csc2005/csc2005-lab03-2020> for lab3
- This repo link will be of the format <https://<username>.github.io/csc2005-lab03-2020>
- In the forked repo create a basic **index.html** file, and make it visible on GitHub repo pages (it's under the settings tab, under GitHub pages).
- You can put your CSS style tags, HTML and JS script code into one file, or organize them into files/directories.

# Assignment Lab 3: Problem

- Connect to the data.gov.sg carpark API and get the realtime dataset. It is detailed [here](#).
- The URI for the API is:  
<https://api.data.gov.sg/v1/transport/carpark-availability>
- Print out all the data into a table - carpark number, timestamp, lot type, total lots, lot availability, and utilization.
- Utilization is calculated - lot availability / total lots.

## Assignment Lab 3: Extra Challenge

- Theme the layout of the table nicely - centralize text, use font classes, etc.
- If the utilization of the carpark is more than 80%, highlight it in red.
- Note that the dataset may be full of data that does not make sense - carpark lots with 0 total lots, more lot availability than carpark lots.
- You can ignore these, or you can write code to weed out / highlight these rows!

# Assignment Lab 3: Screenshot

To prevent copying code, here's a screenshot of my solution:

Singapore carpark availability table					
Carpark Number	Timestamp	Lot Type	Total Lots	Lots available	Utilization
HE12	2020-09-16T03:06:14	C	91	34	37%
HLM	2020-09-16T03:06:18	C	583	478	82%
RHM	2020-09-16T03:06:15	C	322	114	35%
BM29	2020-09-16T03:06:16	C	97	88	91%
Q81	2020-09-16T03:06:18	C	96	70	73%
C20	2020-09-16T03:06:41	C	173	120	69%
FR3M	2020-09-16T03:06:31	C	228	166	73%
C32	2020-09-16T03:06:41	C	289	87	30%
C6	2020-09-16T03:06:24	C	332	112	34%
TG2	2020-09-16T03:06:22	C	273	0	0%
BP1	2020-09-16T03:06:21	C	543	178	33%
TG1	2020-09-16T03:06:37	C	133	26	20%
TGM2	2020-09-16T03:06:23	C	189	115	61%
TE14	2020-09-16T03:06:37	C	138	28	20%
BM3	2020-09-16T03:06:28	C	48	0	0%
BM9	2018-12-17T07:42:34	C	612	121	20%
HG44	2020-09-16T03:04:37	C	143	17	12%
HG64	2020-09-16T03:04:39	C	81	39	48%
PM27	2020-09-16T03:04:35	C	308	75	24%
PM28	2020-09-16T03:04:26	C	302	71	24%
TM36	2020-09-16T03:04:26	C	221	29	13%
TM37	2020-09-16T03:04:40	C	238	53	22%

Lots more rows below. Over 2000, in fact.

I also added quite a bit of CSS theming to make the table look nicer.

# Assignment Lab 3: Hints

- This is not a design challenge - you will not be graded on design.
- More important is to show you can connect to an API, manipulate the data, and construct the table.
- Be sure you know what you are manipulating in the JSON schema - the key-value pairs you need are nested in deeper array / objects. **Console.log** it to inspect the JSON structure.
- Take note that some carpark numbers have more than 1 lot type!
- The total lots and lot availability values are strings - you need to convert them to integers. Round off divided numbers.
- You can use JQuery to append the HTML elements you need.

# Assignment Lab 3: More help please!

- You sure?
- It's better if you challenge yourself to write from scratch. 😊
- All right, here's a basic starter **template**.



# Questions?



Chi-Loong | V/R