CSC2005: 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.

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)

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.

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 synonomous (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 loadash 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

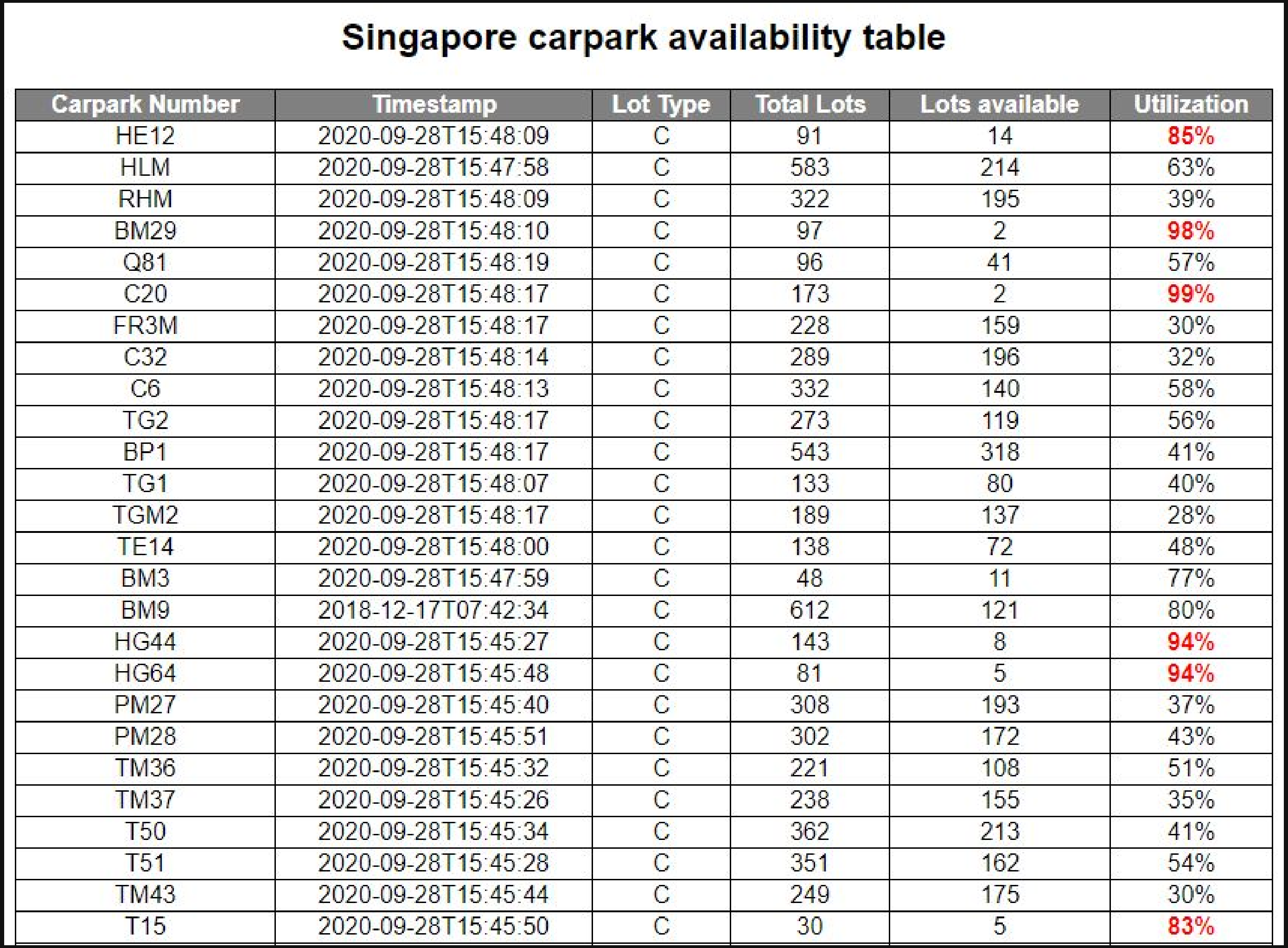
- Connect to the data.gov.sg carpark API and get the realtime dataset. It is detailed here.
- The URI for the API is: https://api.data.gov.sg/v1/transport/carpark-availability
- Print out all the data into a table - carpark number, timestamp, lot type, total lots, lot availability, and utilization.
- Utilization is calculated. It is (total lots - available lots) / 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	2020-09-28T15:48:09	C	91	14	**85%**
HLM	2020-09-28T15:47:58	C	583	214	63%
RHM	2020-09-28T15:48:09	C	322	195	39%
BM29	2020-09-28T15:48:10	C	97	2	**98%**
Q81	2020-09-28T15:48:19	C	96	41	57%
C20	2020-09-28T15:48:17	C	173	2	**99%**
FR3M	2020-09-28T15:48:17	C	228	159	30%
C32	2020-09-28T15:48:14	C	289	196	32%
C6	2020-09-28T15:48:13	C	332	140	58%
TG2	2020-09-28T15:48:17	C	273	119	56%
BP1	2020-09-28T15:48:17	C	543	318	41%
TG1	2020-09-28T15:48:07	C	133	80	40%
TGM2	2020-09-28T15:48:17	C	189	137	28%
TE14	2020-09-28T15:48:00	C	138	72	48%
BM3	2020-09-28T15:47:59	C	48	11	77%
BM9	2018-12-17T07:42:34	C	612	121	80%
HG44	2020-09-28T15:45:27	C	143	8	**94%**
HG64	2020-09-28T15:45:48	C	81	5	**94%**
PM27	2020-09-28T15:45:40	C	308	193	37%
PM28	2020-09-28T15:45:51	C	302	172	43%
TM36	2020-09-28T15:45:32	C	221	108	51%
TM37	2020-09-28T15:45:26	C	238	155	35%
T50	2020-09-28T15:45:34	C	362	213	41%
T51	2020-09-28T15:45:28	C	351	162	54%
TM43	2020-09-28T15:45:44	C	249	175	30%
T15	2020-09-28T15:45:50	C	30	5	**83%**

Lots more rows below. Over 2000, in fact.

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