COMP3322 MODERN TECHNOLOGIES ON WORLD WIDE WEB

Workshop – JavaScript, JSON, and fetch()

Overview

In this workshop, you will develop a simple Web app, which retrieves real-time MTR train data via the MTR Next Train Open Data API. As shown in Fig. 1, the app provides two drop-down selection lists for selecting an MTR line and the target station on that MTR line.

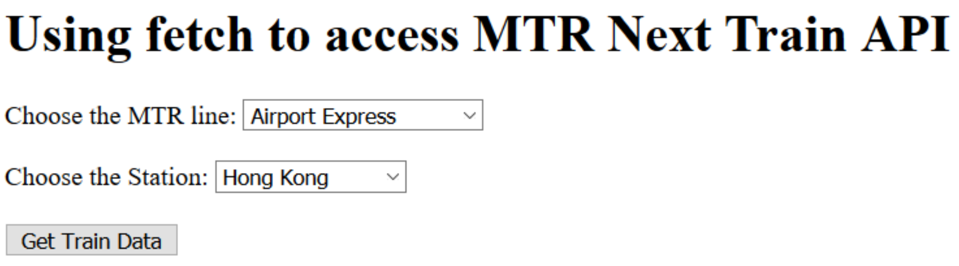
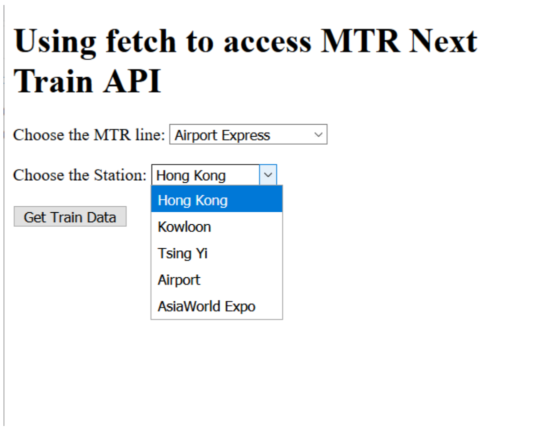

Using fetch to access MTR Next Train API
Choose the MTR line: Airport Express
Choose the Station: Hong Kong
Get Train Data

Fig. 1

Instead of listing all stations, you will design a selection menu that lists only the stations on that selected MTR line. For example, Fig. 2. and Fig. 3 show the lists of stations on the Airport Express line and Tseung Kwan O line respectively.


Using fetch to access MTR Next Train API
Choose the MTR line: Airport Express
Choose the Station: Hong Kong
Get Train Data
Hong Kong
Kowloon
Tsing Yi
Airport
AsiaWorld Expo

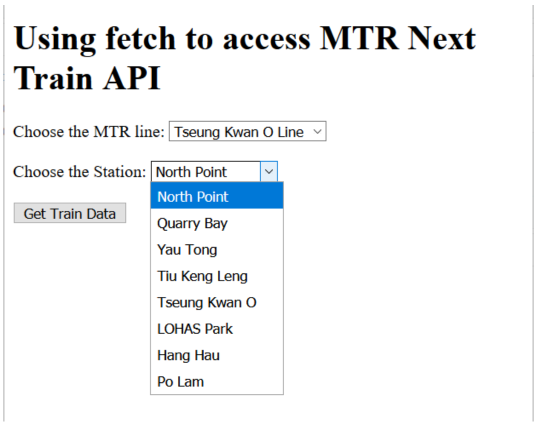
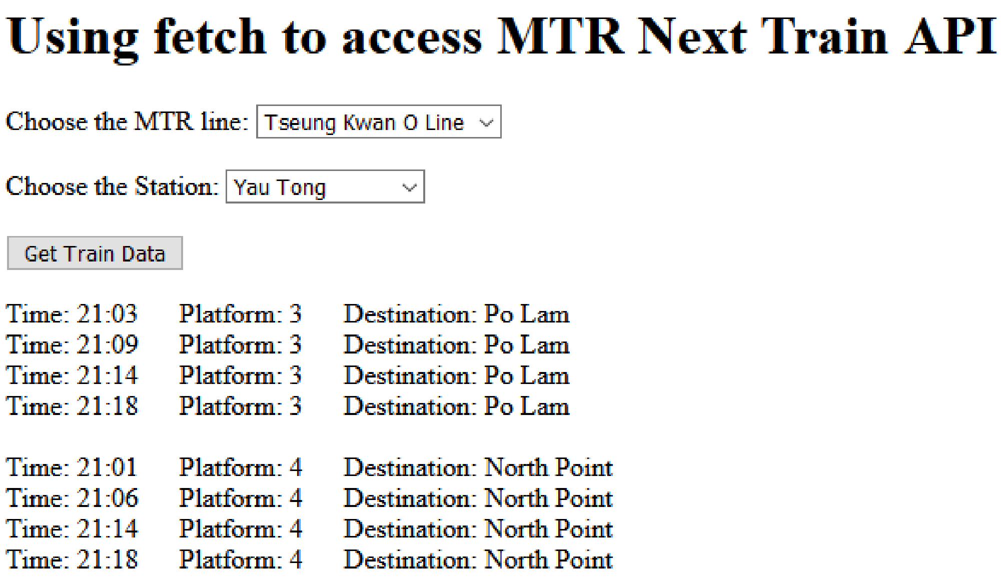

Using fetch to access MTR Next Train API
Choose the MTR line: Tseung Kwan O Line
Choose the Station: North Point
Get Train Data
North Point
Quarry Bay
Yau Tong
Tiu Keng Leng
Tseung Kwan O
LOHAS Park
Hang Hau
Po Lam

Fig. 2 **Fig. 3**

You will practice retrieving the JSON data using the fetch() promise call and extract the data from the returned JSON data file. Then present the data at the space beneath the "Get Train Data" button as shown in Fig. 4.


Using fetch to access MTR Next Train API
Choose the MTR line: Tseung Kwan O Line
Choose the Station: Yau Tong
Get Train Data

```
Time: 21:03    Platform: 3    Destination: Po Lam
Time: 21:09    Platform: 3    Destination: Po Lam
Time: 21:14    Platform: 3    Destination: Po Lam
Time: 21:18    Platform: 3    Destination: Po Lam

Time: 21:01    Platform: 4    Destination: North Point
Time: 21:06    Platform: 4    Destination: North Point
Time: 21:14    Platform: 4    Destination: North Point
Time: 21:18    Platform: 4    Destination: North Point
```

Fig. 4

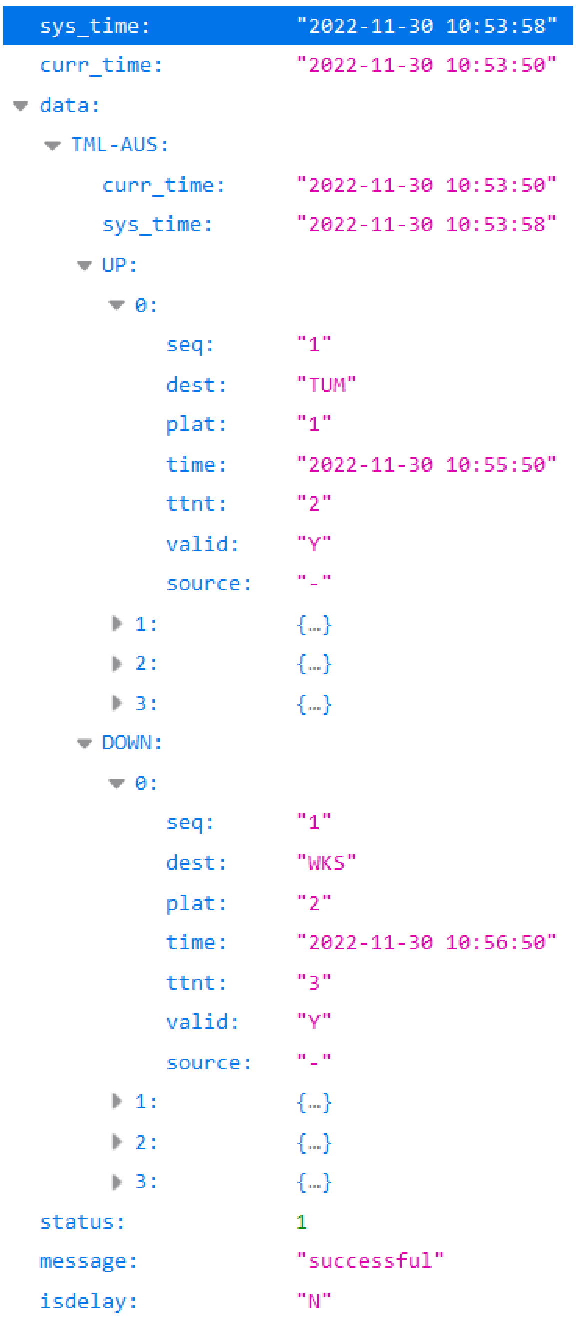
Task 1: Examine MTR Next Train Open Data

Step1: Visit the following Webpage under the DATA.GOV.HK Website.
https://data.gov.hk/en-data/dataset/mtr-data2-nexttrain-data

Before using the API to access real-time MTR data, please download the two documents: Data dictionary for Next Train API and Next Train API Specification.
https://opendata.mtr.com.hk/doc/Next_Train_DataDictionary_v1.2.pdf
https://opendata.mtr.com.hk/doc/Next_Train_API_Spec_v1.2.pdf

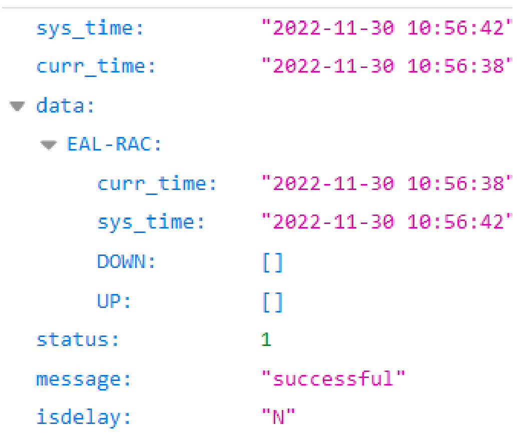
These two documents contain necessary information for you to understand the structure of the returned JSON string and the meanings of various data objects (and terms) carried in the JSON string.

For example, in the API specification document, it gives an example of the response JSON string when the HTTP request is successful (as shown in Fig. 5) and another example of the response when no data is available for that line/station (as shown in Fig. 6).



```
sys_time:           "2022-11-30 10:53:58"
curr_time:          "2022-11-30 10:53:50"
▼ data:
  ▼ TML-AUS:
      curr_time:    "2022-11-30 10:53:50"
      sys_time:     "2022-11-30 10:53:58"
    ▼ UP:
      ▼ 0:
          seq:      "1"
          dest:     "TUM"
          plat:     "1"
          time:     "2022-11-30 10:55:50"
          ttnt:     "2"
          valid:    "Y"
          source:   "-"
      ▶ 1:          {…}
      ▶ 2:          {…}
      ▶ 3:          {…}
    ▼ DOWN:
      ▼ 0:
          seq:      "1"
          dest:     "WKS"
          plat:     "2"
          time:     "2022-11-30 10:56:50"
          ttnt:     "3"
          valid:    "Y"
          source:   "-"
      ▶ 1:          {…}
      ▶ 2:          {…}
      ▶ 3:          {…}
status:             1
message:            "successful"
isdelay:            "N"
```

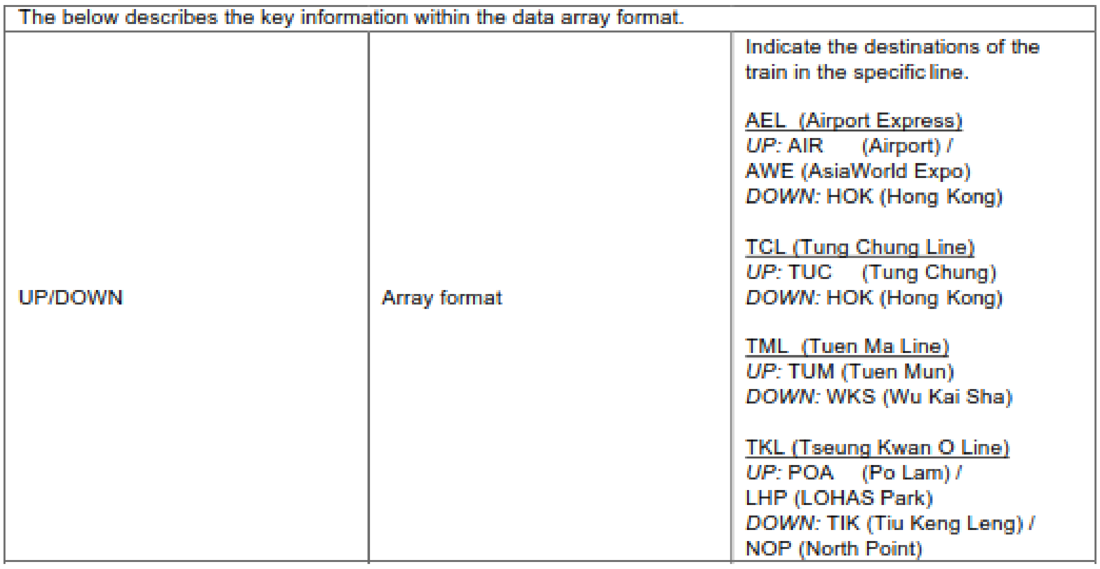
Fig. 5



```
sys_time:           "2022-11-30 10:56:42"
curr_time:          "2022-11-30 10:56:38"
▼ data:
  ▼ EAL-RAC:
      curr_time:    "2022-11-30 10:56:38"
      sys_time:     "2022-11-30 10:56:42"
      DOWN:         []
      UP:           []
status:             1
message:            "successful"
isdelay:            "N"
```

Fig. 6

In the data dictionary document, we can find the required variables for using the API and the meanings of various terms and objects in the returned JSON data. Fig. 7 gives you the screen capture of the description of the UP/DOWN fields.



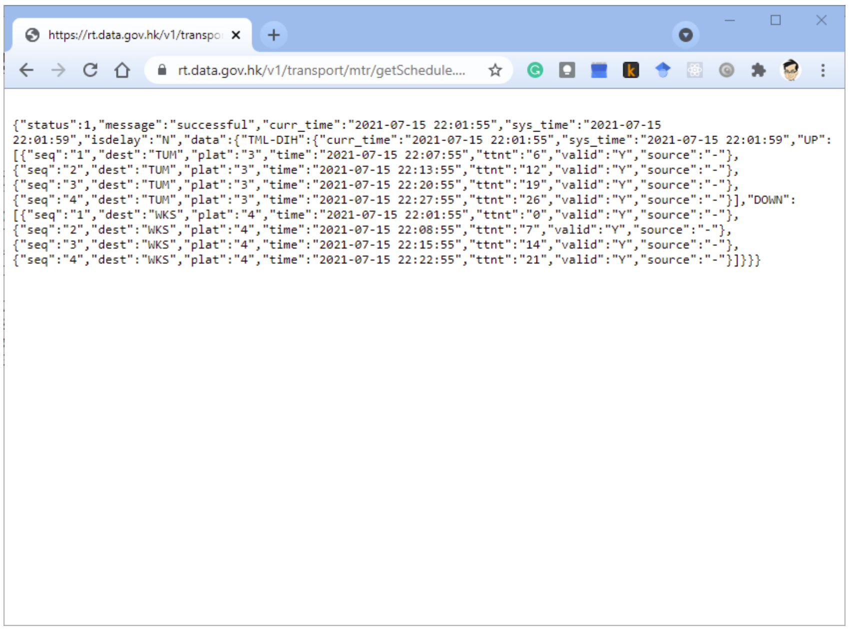
The below describes the key information within the data array format.		
UP/DOWN	Array format	Indicate the destinations of the train in the specific line. <u>AEL (Airport Express)</u> *UP*: AIR (Airport) / AWE (AsiaWorld Expo) *DOWN*: HOK (Hong Kong) <u>TCL (Tung Chung Line)</u> *UP*: TUC (Tung Chung) *DOWN*: HOK (Hong Kong) <u>TML (Tuen Ma Line)</u> *UP*: TUM (Tuen Mun) *DOWN*: WKS (Wu Kai Sha) <u>TKL (Tseung Kwan O Line)</u> *UP*: POA (Po Lam) / LHP (LOHAS Park) *DOWN*: TIK (Tiu Keng Leng) / NOP (North Point)

Fig. 7

Step2: Retrieve MTR data by manually composing the request URL.
Here is the resource URL: https://rt.data.gov.hk/v1/transport/mtr/getSchedule.php
To get the data, you need to provide a query string with two name/value pairs; they are the line parameter and station parameter.

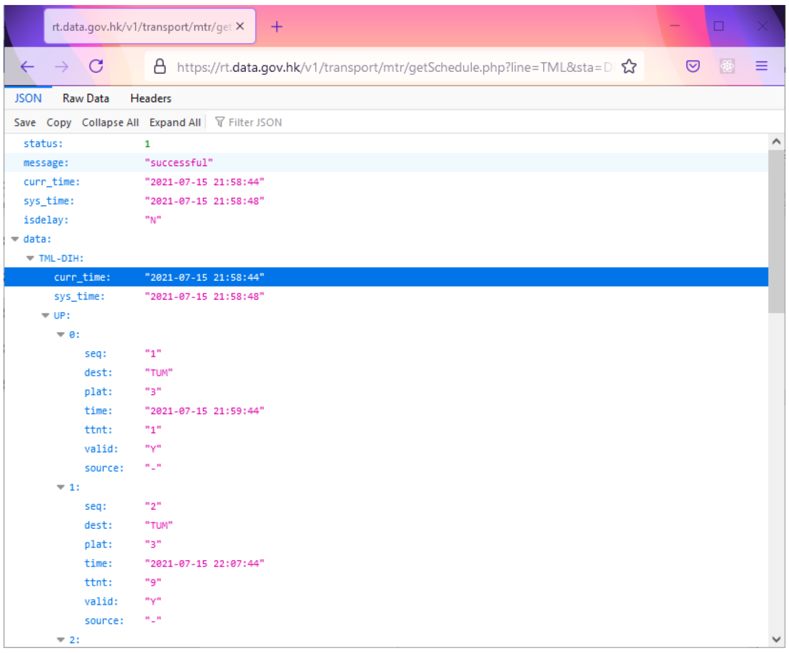
For example, to query the next train arrival schedule of the Diamond station on the Tuen Ma line, we need this URL:
https://rt.data.gov.hk/v1/transport/mtr/getSchedule.php?line=TML&sta=DIH

Copy this URL and paste it to the Chrome browser and then the Firefox browser; you should find that Firefox presents the returned JSON data in a more visually useful way.


https://rt.data.gov.hk/v1/transpo
rt.data.gov.hk/v1/transport/mtr/getSchedule....

```
{"status":1,"message":"successful","curr_time":"2021-07-15 22:01:55","sys_time":"2021-07-15
22:01:59","isdelay":"N","data":{"TML-DIH":{"curr_time":"2021-07-15 22:01:55","sys_time":"2021-07-15 22:01:59","UP":
[{"seq":"1","dest":"TUM","plat":"3","time":"2021-07-15 22:07:55","ttnt":"6","valid":"Y","source":"-"},
{"seq":"2","dest":"TUM","plat":"3","time":"2021-07-15 22:13:55","ttnt":"12","valid":"Y","source":"-"},
{"seq":"3","dest":"TUM","plat":"3","time":"2021-07-15 22:20:55","ttnt":"19","valid":"Y","source":"-"},
{"seq":"4","dest":"TUM","plat":"3","time":"2021-07-15 22:27:55","ttnt":"26","valid":"Y","source":"-"}],"DOWN":
[{"seq":"1","dest":"WKS","plat":"4","time":"2021-07-15 22:01:55","ttnt":"0","valid":"Y","source":"-"},
{"seq":"2","dest":"WKS","plat":"4","time":"2021-07-15 22:08:55","ttnt":"7","valid":"Y","source":"-"},
{"seq":"3","dest":"WKS","plat":"4","time":"2021-07-15 22:15:55","ttnt":"14","valid":"Y","source":"-"},
{"seq":"4","dest":"WKS","plat":"4","time":"2021-07-15 22:22:55","ttnt":"21","valid":"Y","source":"-"}]}}}
```

Fig. 8


rt.data.gov.hk/v1/transport/mtr/get
https://rt.data.gov.hk/v1/transport/mtr/getSchedule.php?line=TML&sta=D
JSON
Raw Data
Headers
Save
Copy
Collapse All
Expand All
Filter JSON

```
status:          1
message:         "successful"
curr_time:       "2021-07-15 21:58:44"
sys_time:        "2021-07-15 21:58:48"
isdelay:         "N"
▾ data:
  ▾ TML-DIH:
      curr_time:   "2021-07-15 21:58:44"
      sys_time:    "2021-07-15 21:58:48"
    ▾ UP:
      ▾ 0:
          seq:     "1"
          dest:    "TUM"
          plat:    "3"
          time:    "2021-07-15 21:59:44"
          ttnt:    "1"
          valid:   "Y"
          source:  "-"
      ▾ 1:
          seq:     "2"
          dest:    "TUM"
          plat:    "3"
          time:    "2021-07-15 22:07:44"
          ttnt:    "9"
          valid:   "Y"
          source:  "-"
      ▾ 2:
```

Fig. 9

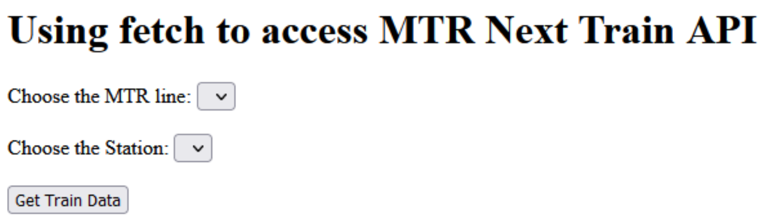
Use the same mechanism to create more requests to practice getting data on different lines and stations.

Task 2: Implement the program

We expect you use the course LAMP container set to work on the workshop. Create a folder named "WK3" in the public_html folder of the c3322-WWW container. Download the "WK3-files.zip" file from our course's Moodle site to the WK3 folder and extract all files. You will find two files – index.html and lines.json. Open the index.html with a code editor. You will see it contains the following HTML content:

```html
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width">
  <title>MTR Next Train</title>
  <style>
  span {
    margin-right: 1.5em;
  }
  </style>
</head>
<body>
  <h1>Using fetch to access MTR Next Train API</h1>
  <label for="line">Choose the MTR line:</label>
  <select name="line" id="line">

  </select>
  <br><br>
  <label for="station">Choose the Station:</label>
  <select name="station" id="station">

  </select>
  <br><br>
  <button id="bttn">Get Train Data</button>
  <div id="output" style="margin-top: 1rem"></div>

  <script>

  </script>
</body>
</html>
```

When the page (http://localhost:9080/WK3/) is loaded, both the selection lists are empty.


Using fetch to access MTR Next Train API

Choose the MTR line:

Choose the Station:

Fig. 10

Step 1: Fill in the selection options for selecting a line out of the five MTR lines. For example, the following option is for selecting the Airport Express line and it is the default option.

```
<option value="AEL" selected>Airport Express</option>
```

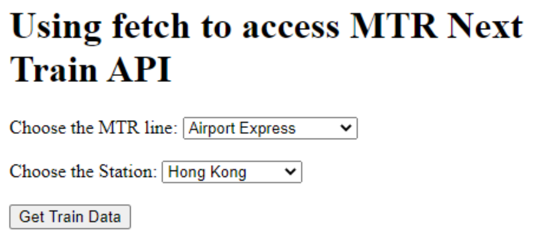
The other options are: ‘TCL’ for the Tung Chung line, ‘TML’ for the Tuen Ma line, ‘TKL’ for the Tseung Kwan O line, and ‘EAL’ for the East Rail line.

Step 2: Fill in the selection options for selecting a station out of the five stations on the **Airport Express line**. For example, the following option is for selecting the Hong Kong station on the Airport Express line.

```
<option class="AEL" value="HOK">Hong Kong</option>
```

To differentiate different stations on different lines, we add the class attribute and set its value to “AEL” for all stations on the Airport Express line.

When done, the two selection lists will look as follows:



Using fetch to access MTR Next Train API

Choose the MTR line: Airport Express

Choose the Station: Hong Kong

Get Train Data

```html
<label for="line">Choose the MTR line:</label>
<select name="line" id="line">
  <option value="AEL" selected>Airport Express</option>
  <option value="TCL">Tung Chung Line</option>
  <option value="TML">Tuen Ma Line</option>
  <option value="TKL">Tseung Kwan O Line</option>
  <option value="EAL">East Rail Line</option>
</select>
<br><br>
<label for="station">Choose the Station:</label>
<select name="station" id="station">
  <option class="AEL" value="HOK">Hong Kong</option>
  <option class="AEL" value="KOW">Kowloon</option>
  <option class="AEL" value="TSY">Tsing Yi</option>
  <option class="AEL" value="AIR">Airport</option>
  <option class="AEL" value="AWE">AsiaWorld Expo</option>
</select>
```

Fig. 11

Currently, we only have the stations on the Airport Express line, we need to use JavaScript to change the set of stations from the AEL set to another set if the user selects another MTR line.

Step 3: We have five MTR lines and each line has its own set of stations. We are going to use JavaScript to dynamically load the set of stations and set up the selection options by fetching a "local" JSON file – lines.json, which contains the set of stations in all five MTR lines.

```json
[{
    "AEL": {
      "HOK": "Hong Kong",
      "KOW": "Kowloon",
      "TSY": "Tsing Yi",
      "AIR": "Airport",
      "AWE": "AsiaWorld Expo"
    }
  },
  {
    "TCL": {
      "HOK": "Hong Kong",
      "KOW": "Kowloon",
      "OLY": "Olympic",
      "NAC": "Nam Cheong",
      "LAK": "Lai King",
      "TSY": "Tsing Yi",
      "SUN": "Sunny Bay",
      "TUC": "Tung Chung"
    }
  },
  {
    "TML": {
      "WKS": "Wu Kai Sha",
      "MOS": "Ma On Shan",
      "HEO": "Heng On",
      "TSH": "Tai Shui Hang",
      "SHM": "Shek Mun",
      "CIO": "City One",
      "STW": "Sha Tin Wai",
      "CKT": "Che Kung Temple",
      "TAW": "Tai Wai",
      "HIK": "Hin Keng",
      "DIH": "Diamond Hill",
      "KAT": "Kai Tak",
      "SUW": "Sung Wong Toi",
```

Use fetch() API to retrieve the lines.json file from the local server, i.e., localhost:9080, then convert it to a JavaScript object. Enter the following code between the <script> tags.

```javascript
<script>

    var Stations = {};  //for the option list
```

```
     var stnList = {};    //for the dictionary of station names

     //Get the lines information
     fetch("lines.json")
     .then(response => {
       response.json().then(lines => {

         //Build the option selection lists for each MTR line

         STEP 4


         //Install an event handler for changing the station list after
selecting the MTR line

         STEP 5


         //Install an event handler for handling the "Get Train Data"
button
         STEP 6

       })
     })

</script>
```

Step 4: Use JavaScript to build a JavaScript object (**Stations**) that contains sets of selection options for all MTR lines. The object contains **five** key/value pairs. Each key maps to the set of station options for one MTR line. The result object would like as follows:

```
 {
   'AEL':
   '<option class="AEL" value="HOK">Hong Kong</option>
         :
   <option class="AEL" value="AWE">AsiaWorld Expo</option>',
   'TCL':
   '<option class="TCL" value="HOK">Hong Kong</option>
         :
   <option class="TCL" value="TUC">Tung Chung</option>',
   'TML':
   '<option class="TML" value="WKS">Wu Kai Sha</option>
         :
   <option class="TML" value="TUM">Tuen Mun</option>',
   'TKL':
   '<option class="TKL" value="NOP">North Point</option>
         :
   <option class="TKL" value="POA">Po Lam</option>',
   'EAL':
   '<option class="EAL" value="ADM">Admiralty</option>
         :
   <option class="EAL" value="LMC">Lok Ma Chau</option>'
 };
```

In addition, construct an associative array (**stnList**) for mapping a station's acronym to the station's full name while parsing the lines.json object.

```
  { "HOK": "Hong Kong", "KOW": "Kowloon", "TSY": "Tsing Yi", "AIR": "Airport", "AWE": "AsiaWorld Expo", "HOK": "Hong Kong", "KOW": "Kowloon", . . . ., "LHP": "LOHAS Park", "HAH": "Hang Hau", "POA": "Po Lam" };
```

```
        //Build the option selection lists for each MTR line
        lines.forEach(line => {
          let station_opt_list = "";
          let linename = Object.keys(line)[0];
          let linelist = Object.values(line)[0];
          for (const station in linelist) {
            station_opt_list += `<option class="${linename}"
 value="${station}">${linelist[station]}</option> `;
            //create the dictionary of station names
            stnList[station] = linelist[station];
          }
          Stations[linename] = station_opt_list;
        });
```

Step 5: Use JavaScript to check whether the user has selected a different MTR line. When detecting there is a change of the selection option, an event handler will be executed to switch the list of stations from the current set to the target set of stations according to the change.

```
    let currentClass="AEL"; //assume Airport Express line initially
    let line = document.getElementById('line');
    line.addEventListener('change', (evt) => {
      let select = line.value;
      if (select != currentClass) { //there is a change
        let station = document.querySelector('#station');
        station.innerHTML = Stations[select];
        currentClass=select;
      }
    });
```

Based on the selected MTR line, we access the Stations object to get the new set of station options and replace the previous set. Be remembered to keep track of the selected MTR line.

Load the index.html file to the browser, you should see that the two selection lists working as shown in Fig. 2 and Fig. 3.

Step 6: Use fetch() API to retrieve the next train information

Register an event handler for the “Get Train Data” button. Upon clicking on the button, a fetch() request will be sent to the server with the correct ‘line’ and ‘sta’ parameters. When the response returns, check whether the response status is successful; if it is successful, extract the data from the returned JSON string and output the data to the <div> block with id=”output”.

The basic idea is to first check the return **status** of that line&station. If that line&station is operating, the status is 1; otherwise, the status is 0. However, even the status is 1, the returned data may not contain any train information, this would be reflected by the **isdelay** key. For normal cases, the returned data may contain a set of train data for both the UP and DOWN routes, or just UP/DOWN route as it depends on the station.

```
    let bttn = document.getElementById('bttn');
    bttn.addEventListener("click", fRequest); //register the handler fRequest

    function fRequest() {
      let line = document.getElementById('line').value; //get the MTR line
      let station = document.getElementById('station').value; //get the station

fetch(`https://rt.data.gov.hk/v1/transport/mtr/getSchedule.php?line=${line}&sta=${station}`)
      .then( response => {
        if (response.status == 200) {  //receive response successfully
          response.json().then( schedule => {
            let output = "";

            if (schedule.status == 0) {  //Special Train Service
              //will handle this later
                  STEP 7

            } else {
              if (schedule.isdelay == 'Y') {  //Data Absence
                //will handle this later
                  STEP 7

              } else {  //Normal response
                let dataUP = schedule.data[line+'-'+station].UP;
                let dataDN = schedule.data[line+'-'+station].DOWN;

                if (dataUP) {  //has the UP data
```

```
                  for (let train of dataUP) {
                    let time = train.time.substr(11,5);
                    output += '<span>Time: '+time+'</span>';
                    output += '<span>Platform: '+train.plat+'</span>';
                    output += '<span>Destination: '+stnList[train.dest]+'<br></span>';
                  }
                  output += '<br>';
                }
                if (dataDN) {  //has the DOWN data
                  for (let train of dataDN) {
                    if (Object.keys(train).length) {  //May not have data – Last Train
                      let time = train.time.substr(11,5);
                      output += '<span>Time: '+time+'</span>';
                      output += '<span>Platform: '+train.plat+'</span>';
                      output += '<span>Destination: '+stnList[train.dest]+'<br></span>';
                    }
                  }
                }
              }
            }
            //Write the data beneath the button
            document.getElementById('output').innerHTML = output;
          });
        } else {
          console.log("HTTP return status: "+response.status);
        }
      });
    }
```

Load the index.html file to the browser and test the program by selecting an MTR line and a station on that line, then click on the “Get Train Data” button to retrieve the data.

Step 7: Complete the program by handling those special cases.

When received a successful response but with the returned JSON status code equals zero, the MTR line is in special arrangement status or your query parameter(s) is/are not correct. Then output the returned message and URL (if available).

```
    output += schedule.message;
    if (schedule.url)
      output += `<br><a href='${schedule.url}'>${schedule.url}</a>`;
```

When received a successful response but with the returned JSON data ‘isdelay’ field equals “Y”, this indicates the absence of data. Then output the following message.

```
    output = "No data is available";
```

Now the program should be able to retrieve the next train arrival timings of the stations along the five MTR lines. Also, it should be able to handle special cases.

Test your page

Browse and test your Web page with different MTR lines and stations at:
http://localhost:9080/WK3/