Software Tools – Web Technologies Tuesday 27 2021

Introduction

You are going to build on the final activities (3 and 4) from the last workshop on Friday 23 April. Recall that you were given data about pets in JSON format and your task was to write JavaScript (JS) code that parsed and displayed the data in an HTML file when a button was clicked. In this activity you will again write JS code to parse data about pets that is stored in JSON format. However, this time you will not be given the data file and you will instead have to write code to access the data that is stored on another website. You will use HTTP requests to access a **web service**, which is a server that listens for requests for data and provides an interface to a database server.

In this session you will be communicating with **Web APIs (Application Programming Interface)** where communication is based on representational state transfer or REST. Web APIs have public **endpoints** which specify where resources are located. RESTful web APIs are communicated with using HTTP requests and data is exchanged in JSON or XML format. Companies and governmental agencies often provide resources that software developers can interact with.

In the first activities we are going to get data from this Web API: https://petstore.swagger.io/ that contains data about pets. We'll refer to this Web API as the Petstore API. Ithas good documentation and will enable you to practice writing code to carry out a number of different operations: reading; creating; and deleting data on the web server. Please note that because this Web API is open to the public, the data can change quite frequently. It is also possible that some of the data could be in questionable taste. Please don't use obscenities if you are updating the data!

Activity 1

The first activity involves making a request to the Petstore API to get the names of all the pets that are currently available in the petstore. Pets can have three different status values: available; pending; or sold. We are interested in the available pets. Look in the documentation on the webpage for **GET /pet/findByStatus.** It gives an example Request URL:

```
https://petstore.swagger.io/v2/pet/findByStatus?status=sold
```

We will always use HTTPS for communication in these exercises (the secure version of HTTP).

The URL of the resource is: **petstore.swagger.io**

The path to the resource is: **/v2/pet/findByStatus**

The parameter value that specifies the availability of the pet is specified by : **?status=available**

Step One

On the petstore.swagger.io webpage, try out the request by clicking the 'Try it out' button in the documentation for the GET /pet/findByStatus documentation. You can set the parameter to any of the three different values. When you presss 'Execute' you can see the format of the data that the server returns as a response.

Step Two

Your first task is to build a simple webpage that contains a button. When this button is clicked it will run JS code that will make the request to the Petstore API that returns all of the data about available pets (as shown above). To help you get started you can download the following files from the Unit website in the Weeks 21 and 22 section(https://cs-uob.github.io/COMS10012/contents.html):

index.html, **style.css** and **script.js**

I recommend that you write your code using Visual Studio Code but feel free to use any IDE that you are familiar with.

Most of the code is written for you for Step One, but you need to add a button to index.html that has an onclick event that triggers the handleClick function that is defined in script.js. It is worth spending some time understanding how the code works as you will be adapting it in other activities.

You'll see that in index.html there is a table element that will be populated by the handleClick function. The table is constructed using the templating approach we used last week where elements are replaced by data values. In particular, 'PET_NAME' is replaced by the name of each pet. The script updates the table using 'getElementById', the id of the table and setting the 'innerHTML'.

You'll see that the script builds the path to the resource using the URL, path and parameter.

The main work in the handleClick function is done by fetch(). The documentation for this function is here: https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch. You might find the following documentation more user friendly: https://www.codeinwp.com/blog/fetch-api-tutorial-for-beginners/

fetch is a method for fetching resources asynchronously using HTTP. That means that the resources are requested on demand and don't require the whole HTML page to be reloaded. It also means that when fetch is called it returns a **promise**, which is an object to which you can attach code that will run when it completes fetching the resource. It is also possible to chain promises together, with each promise specified by the 'then' keyword.

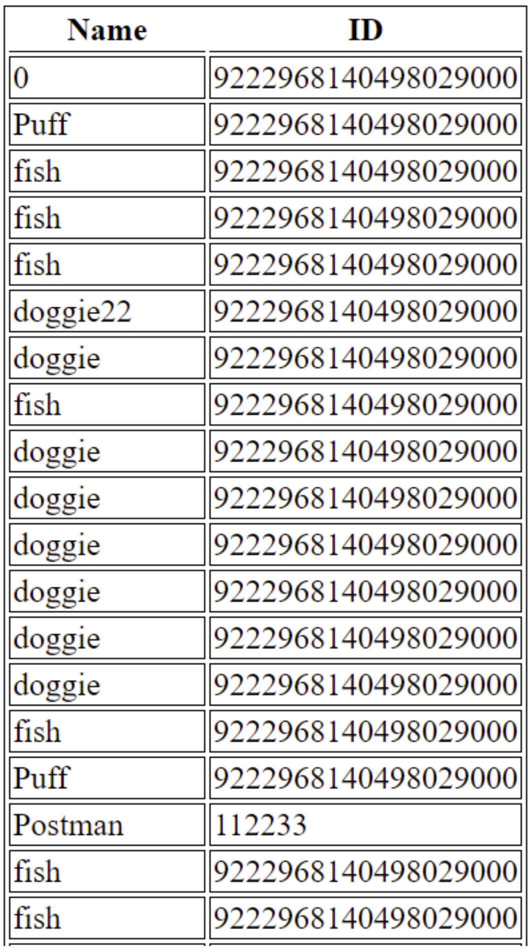
In handleClick the fetch function returns two chained promises:

```
fetch(path)
    .then(response => response.json())
    .then(data => {
      if(!Array.isArray(data)){
        throw new Error('The data is invalid!');
      }

      data.forEach(pet => {
        petsHTML += petRowHTML.replace('PET_NAME', pet.name);
      });

      document.getElementById('petsTable').innerHTML = petsHTML;
    })
    .catch((error) => {
      console.log('AN ERROR HAS OCCURRED: ', JSON.stringify(error))
    });
```

`() => x` is short for `() => { return x; }`.

Activity Two

Adapt the code in script.js so that the name **and** the id of the pets are displayed when the button is clicked. Display the data in a table which should look like this:



Name	ID
0	9222968140498029000
Puff	9222968140498029000
fish	9222968140498029000
fish	9222968140498029000
fish	9222968140498029000
doggie22	9222968140498029000
doggie	9222968140498029000
fish	9222968140498029000
doggie	9222968140498029000
doggie	9222968140498029000
doggie	9222968140498029000
doggie	9222968140498029000
doggie	9222968140498029000
doggie	9222968140498029000
fish	9222968140498029000
Puff	9222968140498029000
Postman	112233
fish	9222968140498029000
fish	9222968140498029000

Activity Three

Change the resource path that you pass to fetch so that you request one pet with a specific ID. For example, if you wanted to get the pet called ‘Postman’ from the above table, then you’d request the pet with an ID of 112233. As you can see from the above table, some of the pets ids have the same value (9222968140498029000) i.e they have values that have overflowed the maximum due to the way that the data have been manipulated by other developers. When you write this function you can hard code the id into the resource path but make sure you pick an id value that has not overflowed.

Activity Four

Use the POST method to add a pet to the petstore. You can do this by adding a second argument to fetch. You can check the documentation referred to above for the exact syntax. You will also need to pass in the data for the new pet in the appropriate JSON format. Here is a code snippet to help you:

```
{
    method: 'POST',
    headers: {
      'Content-Type': 'application/json',
    },
    body: JSON.stringify(data),
  })
```

Activity Five

Fetch a random dog image from the following Web API and display it on a webpage when a user clicks a button. Each time the button is pressed a different random image should be displayed.

```
'https://dog.ceo/api/breeds/image/random'
```

Activity Six

You can try out other free APIs available on this Github repo: https://github.com/public-apis/public-apis. Select one that doesn't require authentication (i.e. it has "No" under the "Auth" column). Build a webpage that displays the data when a button is clicked.