Software Tools – Web Technologies Friday 23 April 2021

Activity 1

Step One

You are going to try running the JS code snippets in jsbin, which is an online, collaborative coding environment that is primarily used to test HTML, CSS and JS code. There is also a JS console built in to this tool. Go to: jsbin.com, click on the HTML tab – what attribute and value are missing from one of the tags?

Step Two

Type in the JS code snippets that were introduced in the lecture – there are 7 different example areas - into the JavaScript window. Then click the Run button which is situated in the top right half of the window. When you enter the first snippet there will be an error message in red that states "Assignment to constant variable". Try modifying the JS and see what effect, if any, that has on the output of the code snippet. If the snippet does not contain a call to console.log then add one and test the outcome of your modifications. The goal of this initial activity is to get you more familiar with JS syntax.

Activity 2

You are now going to download Visual Studio Code, which is a free and open source code editor that is available for Windows, Mac and Linux. Download the correct version for your machine from here: https://code.visualstudio.com/

This editor is really good for doing client-side development with HTML, CSS and JavaScript, particularly when you configure it with some extensions. VS Code is also a good editor for lots of other languages. It also integrates well with Git.

When you're installing Visual Studio Code I'd recommend ticking all of the additional tasks when you get to the "Select additional tasks" part of the installation.

Once installed you can might want to configure some of the commonly used settings: click the cog in the bottom left corner and select settings. In particular, you might want to increase the font size (tor reduce eye strain), reduce the tab size (from 4 to 2, otherwise when there are lots of levels in your code it can shift too far to the right) and you can turn word wrap on to prevent lines of code going off the right hand side of the window.

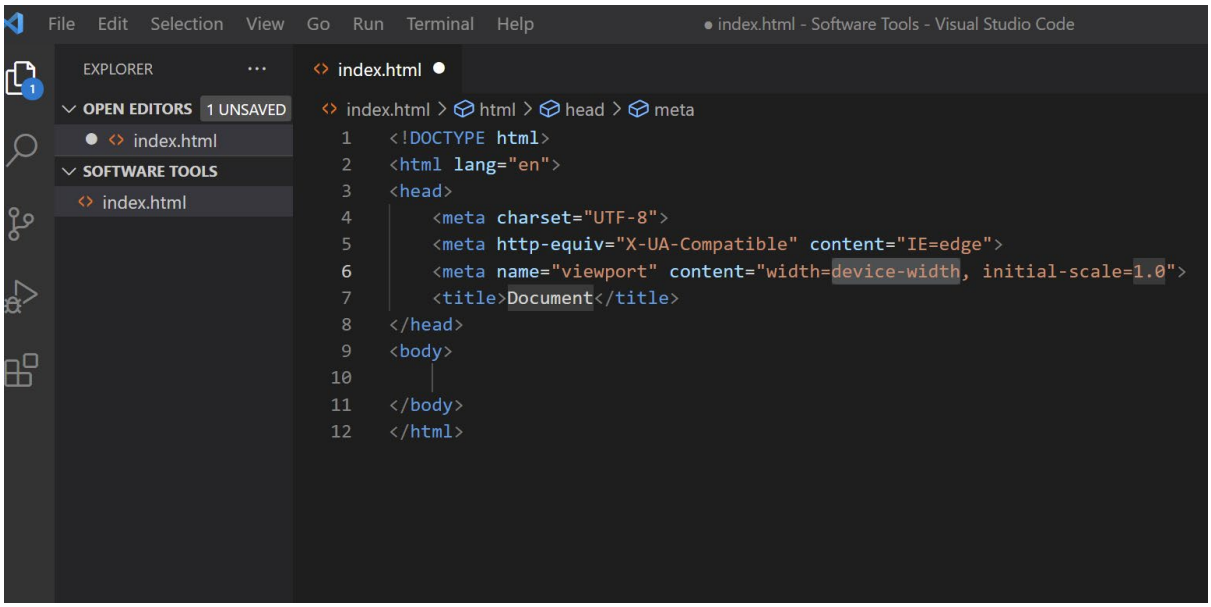
Click on the extensions icon on the left hand side navigation bar (typically the fifth icon down). Select these two as they will support today's activities:

Live Server, which sets up a local web server on your machine

JavaScript ES6 snippets, which will help you write JS code

Then click 'File' and 'Add Folder to Workspace' and this will open your File Explorer application to the Visual Studio Code workspace folder. You can then right click and add a folder to this workspace folder. Call it something like 'Software Tools'. Then if you hover over the folder name in the Explorer pane you will see the 'New File' icon – click this and create a file called 'index.html'. Click in the file

window pane and type '!' – this will open an Emmet that will rapidly create an empty HTML file for you:



VS Code has IntelliSense that will make suggestions to help you write code. This can save you a lot of time. For example, it will automatically create closing tags.

Activity 3

We are now going to parse a JSON file with JavaScript and display it in an HTML file. This activity was developed by Tom and Marceli for the MSc version of the Software Tools unit.

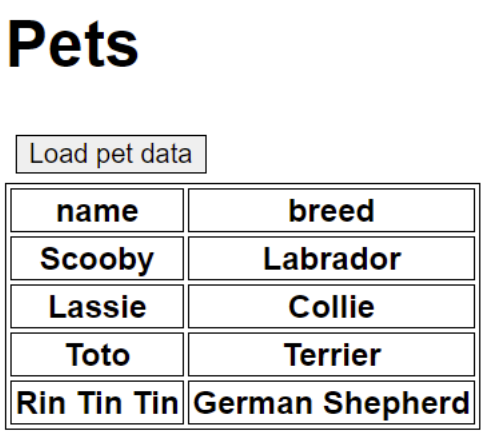
The JSON file can be accessed at:

https://cs-uob.github.io/COMSM0085/resources/web/activity1/data.json

It has the following structure:

```
{
  "pets": [
    {
      "name": "Scooby",
      "breed": "Labrador"
    },
    {
      "name": "Lassie",
      "breed": "Collie"
    },
    {
      "name": "Toto",
      "breed": "Terrier"
    },
    {
      "name": "Rin Tin Tin",
      "breed": "German Shepherd"
    }
  ]
}
```

As you can see, the file contains an array of pets, specifically dogs, each of which has a name and a breed. The task is to parse this JSON file and display it in a table in the index.html file when a person clicks a button. It should look like this, although I'm sure you can style it better than I have done:



Create another file in your workspace folder called "javascript.js" – we'll keep all of the JS code in this file and also the JSON file. Also create a CSS file called "stylesheet.css" in your workspace folder – this will contain your CSS code.

You need to link to the CSS and JS files from index.html. Add the link to the CSS file in the head. You might need to refer to the lecture slides from Tuesday or refer to www.w3schools.com

Add the link to the JS file at the bottom of the body

```
<script src="javascript.js" defer></script>
```

Note that I've added the defer keyword to the script tag. This is not completely necessary as we are going to call the JS by attaching it to the button onclick event.

We are now going to write some JS code in "javascript.js" that will parse the JSON file and dynamically create the table containing the pet data in index.html. To do this add the following code to the body:

```
<body>
    <h1>Pets</h1>
    <button onclick="loadPetData()">Load pet data</button>
    <table id="pets"></table>
    <script src="javascript.js" defer></script>
</body>
```

I'm now going to guide you through writing the JS code. After we've finished parsing and displaying this JSON file, I will then give you a link to a more complicated JSON file and it will be your task to parse and display those data.

At the top of javascript.js, declare a constant "pet_data" and assign the JSON data to it:

```
const pet_data =
{
    "pets": [
      {
        "name": "Scooby",
        "breed": "Labrador"
      },
      {
        "name": "Lassie",
        "breed": "Collie"
      },
      {
        "name": "Toto",
        "breed": "Terrier"
      },
      {
        "name": "Rin Tin Tin",
        "breed": "German Shepherd"
      }
    ]
  };
```

Next, create the table header that is the first row of the table and contains the words “name” and “breed”:

```
const tableHeaders =
  '<tr><th>name</th><th>breed</th></tr>';
```

We now need to parse the JSON data and create a table that contains the pet data. This will be stored as a string in a variable called petsHTML. We declare this and add the tableHeaders to the string:

```
 let petsHTML = '';
  petsHTML += tableHeaders;
```

We next need to think about what structure every row of the table below the headers has. This is going to be our *template* and we will use upper case names in each of the table cells to indicate what data will be placed there:

```
  const petRow =
  '<tr><td>PET_NAME</td><td>PET_BREED</td></tr>';
```

We are now going to write a JS function that parses the JSON file and uses the template structure to build the table a row at a time.

```
  function loadPetData(){
  pet_data.pets.forEach(pet=>{
    let thisPetHTML = petRow;
    thisPetHTML = thisPetHTML.replace('PET_NAME', pet.name);
    thisPetHTML = thisPetHTML.replace('PET_BREED', pet.breed);
    petsHTML += thisPetHTML;

  });

  document.getElementById //this line needs to be completed
}
```

This function iterates through each of the objects in the JSON pets array using 'forEach' and assigns the object to the variable 'pet'. Each iteration it assigns the template held in the variable 'petRow' to the variable 'thisPetHTML'. It then calls the replace function on the template to replace the 'PET_NAME' and 'PET_BREED' placeholders with the actual JSON data, held in 'pet.name' and 'pet.breed' respectively. It then updates petsHTML, which is holding all of the table HTML.

When all of the JSON has been parsed, the function calls the DOM to update the HTML, specifically the table which has a particular ID in 'index.html'. Your job is to complete this line so that when the button is pressed the table appears below the button.

You'll want to do some styling of the table. Here's something to get you started which will add borders around each cell in the table:

```
table, th, td {
    border: 1px solid black;
  }
```

You should also add some space between the button and the table and use a consistent font in the page.

To test whether your code works, right click 'index.html' in the Explorer pane and select "Open With Live Server". This will open index.html in a browser window. You can then click the button and see if the table correctly displays the pet data.

Activity 4

Activity 3 demonstrated how to parse a JSON file and dynamically create an HTML table that displays the data from the JSON file.

In this activity your challenge is to parse a more complex JSON file (also containing data about different pets) using the same techniques as we used in Activity 3. The JSON file for this activity is here: https://cs-uob.github.io/COMSM0085/resources/web/activity2/activity_2_data.json

Your first step is to look at the JSON data and think what it would look like in a table. Then create a template so that you can use the same replace technique as in Activity 3.

Hint: if you hold the cursor over an element then it will indicate what JSON is contained in it. This can help you develop your algorithm.

Hint #2: given the increased complexity of the JSON file you will need to iterate through different parts of the JSON data.

The JSON data for this final activity looks like this:

```
{
  "pets": [
    {
      "type": "fish",
      "pets": [
        {
          "name": {
            "first": "Nemo",
            "last": "Finding"
          },
          "colours": ["orange", "white"]
        },
        {
          "name": {
            "first": "Michael",
            "last": "Thomas"
          },
          "colours": ["yellow"]
        },
        {
          "name": {
            "first": "Dory",
            "last": "Grubber"
          },
          "colours": ["blue", "yellow", "black"]
        }
      ]
    },
    {
      "type": "cat",
      "pets": [
        {
          "name": {
            "first": "Squid",
            "last": "Johnson"
          },
          "colours": ["black"]
        },
        {
          "name": {
            "first": "Peanut",
            "last": "Day"
          },
          "colours": ["black", "brown"]
        },
        {
          "name": {
            "first": "Marley",
            "last": "Day"
          },
          "colours": ["black", "brown"]
        },
        {
          "name": {
            "first": "Bear",
            "last": "Blue"
```

```
          },
          "colours": ["gray"]
        }
      ]
    },
    {
      "type": "dog",
      "pets": [
        {
          "name": {
            "first": "Rin",
            "last": "Tin Tin"
          },
          "colours": ["black"]
        },
        {
          "name": {
            "first": "Sharik",
            "last": "Gajos"
          },
          "colours": ["black", "brown"]
        }
      ]
    }
  ]
}
```