Objectives

- Return of Javascript

Preparation

Before the lab you should re-read the relevant lecture slides and their accompanying examples.

Getting Started

Create a new directory for this lab called `lab12` by typing:

```
$ mkdir lab12
```

Change to this directory by typing:

```
$ cd lab12
```

Exercise: fetch_and_list

Similar to last week clone the repo again. This time it'll have the week 12 materials. You can also just do a `git pull` in last weeks repo if you are feeling comfertable with git.

```
$ git clone https://github.com/COMP2041UNSW/js
...
```

Now run the Node Package Manager (NPM)

```
$ cd js
$ 2041 npm install
...
```

The below you will start a simple server which will serve all the files in the repo.

```
$ 2041 npm start
...
```

The command will make the web-server available at http://127.0.0.1:8080
You can stop the server by pressing `control-c`

Working from Home

You can complete all the JavaScript exercises in your own computer.

You'll first need to install **node** on your computer. You can find installation instructions here: https://nodejs.org/en/download/

If you have problem installing **node** ask in the course forum.

You run the same commands on your own computer but without the 2041 prefix

```
$ cd js
$ npm install
....
$ npm start
```

The Lab

Now that you are all set up you can see the first lab exercise by going to http://127.0.0.1:8080/labs/lab12/fetch_and_list

Go to the code and edit it so you fetch some users from the outlined API and render them on screen

More information can be seen at http://127.0.0.1:8080/labs/lab12/fetch_and_list

```
$ cd labs
$ cd lab12
$ cd fetch_and_list
$ vi fetch.js # or any text editor that is not notepad++
```

Remember to refresh the html page when you make a change to the js so it gets loaded, good luck!

There is no autotest and no automarking of this question.
When you have completed demonstrate your work to another student in your lab and ask them to enter a peer assessment here

It is prefered you do this during your lab, but if this is not possible you may demonstrate your work to any other COMP[29]041 student before Sunday midnight. Note, you must also submit the work with give.

When you are finished working on this exercise you must submit your work by running **give**:

```
$ give cs2041 lab12_js_fetch_and_list fetch.js
```

Sample solution for `fetch.js`

```
(function () {
    'use strict';

    window.onload = init;
    let output;

    function createUserDiv(user) {

        const div = document.createElement('div');
        div.className = 'user';

        const h2 = document.createElement('h2');
        h2.innerText = user.name;

        const p = document.createElement('p');
        p.innerText = user.company.catchPhrase;

        div.appendChild(h2);
        div.appendChild(p);

        return div;
    }

    function append(element) {
        output.appendChild(element);
    }

    function init() {

        output = document.getElementById('output');

        fetch('https://jsonplaceholder.typicode.com/users')
            .then(res => res.json())
            .then(data => data.map(createUserDiv))
            .then(elements => elements.map(append));
    }

}());
```

Exercise: fetch_and_list_2

Navigate to http://127.0.0.1:8080/labs/lab12/fetch_and_list_2/

Code is available at `labs/lab12/fetch_and_list_2/fetch.js`

Extend your fetch and list code to combine information from both the users and posts endpoints from the supplied test api.

see http://127.0.0.1:8080/labs/lab12/fetch_and_list_2/ for more details

There is no autotest and no automarking of this question.
When you have completed demonstrate your work to another student in your lab and ask them to enter a peer assessment here

It is prefered you do this during your lab, but if this is not possible you may demonstrate your work to any other COMP[29]041 student before Sunday midnight. Note, you must also submit the work with give.

When you are finished working on this exercise you must submit your work by running **give**:

```
$ give cs2041 lab12_js_fetch_and_list_2 fetch.js
```

Sample solution for `fetch.js`

```
(function() {
    'use strict';

    window.addEventListener('load', init);

    function getJSON(path) {
        return fetch(path).then(res => res.json());
    }

    function append(element, parent) {
        parent.appendChild(element);
    }

    const mapAppend = parent => element => append(element, parent);

    function createUserDiv(user) {

        const div = document.createElement('div');
        div.className = 'user';

        const h2 = document.createElement('h2');
        h2.innerText = user.name;

        const p = document.createElement('p');
        p.innerText = user.company.catchPhrase;

        div.appendChild(h2);
        div.appendChild(p);

        return div;
    }

    function createPostsList(posts) {
        const ul = document.createElement('ul');
        ul.className = 'posts';

        void posts.map(post => {
            const li = document.createElement('li');
            li.innerText = post.title;
            li.className = 'post';
            ul.appendChild(li);
            return li;
        });

        return ul;
    }

    function init() {

        const output = document.getElementById('output');

        Promise.all([
            getJSON('https://jsonplaceholder.typicode.com/users'),
            getJSON('https://jsonplaceholder.typicode.com/posts')
        ])
        .then(([users, posts]) => {
            const postsMap = posts.reduce((postsMap, post) => {
                if (!postsMap[post.userId])
                    postsMap[post.userId] = [];

                postsMap[post.userId].push(post);
                return postsMap;

            }, {});

            return users.map(user => {
                user.posts = postsMap[user.id];
                return user;
            });

        })
        .then(users => {
            return users.map(user => {
                const userDiv = createUserDiv(user);
```

```
                userDiv.appendChild(createPostsList(user.posts));
                return userDiv;
            });
        })
        .then(elements => elements.map(mapAppend(output)));

    }
}());
```

Exercise: pretty_pictures

Navigate to http://127.0.0.1:8080/labs/lab12/pretty_pictures/

Code is available at `labs/lab12/pretty-pictures/fetch.js`

Write some code to fetch a set of nice images and display them in a list

Optional Challange Add a nice loading animation while the page waits for all the images to fetch

see http://127.0.0.1:8080/labs/lab12/pretty_pictures/ for more details

There is no autotest and no automarking of this question.
When you have completed demonstrate your work to another student in your lab and ask them to enter a peer assessment here

It is prefered you do this during your lab, but if this is not possible you may demonstrate your work to any other COMP[29]041 student before Sunday midnight. Note, you must also submit the work with give.

When you are finished working on this exercise you must submit your work by running **give**:

```
$ give cs2041 lab12_js_pretty_pictures fetch.js
```

Sample solution for `fetch.js`

```
(function () {
    'use strict';

    window.onload = init;
    let output;
    let loading;

    function addPost(url) {
      const elem = document.createElement("div");
      elem.className = "img-post";
      const img = document.createElement("img");
      img.src = url;
      const t = document.createElement("p");
      const d = new Date();
      t.innerText = `Fetched at ${d.getHours()}:${d.getMinutes()}`;
      elem.appendChild(img);
      elem.appendChild(t);
      output.appendChild(elem);
    }

    function getMore() {
      const children = [];
      for(let child of output.childNodes)
        if(child.className === "img-post")
          children.push(child);
      children.map(child=>output.removeChild(child));
      let i = 0;
      const API_URL = "https://picsum.photos/300/300/?random";
      const promises = [];
      while(i < 5) {
        promises.push(fetch(API_URL));
        i++;
      }
      loading.style.display = "block";
      Promise.all(promises).then((responses)=>{
        loading.style.display = "none";
        responses.map((x)=>addPost(x.url));
      })
    }

    function init() {
        output = document.getElementById('output');
        loading = document.getElementById('loading');
        document.getElementById("more").addEventListener("click",getMore);
    }
}());
```

Exercise: tabs

Navigate to http://127.0.0.1:8080/labs/lab12/tabs/

Code is available at `labs/lab12/tabs/tabs.js`

Write some code to let you tab through a set of information on planets.

see http://127.0.0.1:8080/labs/lab12/tabs/ for more details

There is no autotest and no automarking of this question.
When you have completed demonstrate your work to another student in your lab and ask them to enter a peer assessment here

It is prefered you do this during your lab, but if this is not possible you may demonstrate your work to any other COMP[29]041 student before Sunday midnight. Note, you must also submit the work with give.

When you are finished working on this exercise you must submit your work by running **give**:

```
$ give cs2041 lab12_js_tabs tabs.js
```

Sample solution for `tabs.js`

```
(function () {
    'use strict';
    window.onload = init;
    let planets;
    let info;
    let mapping = {
      "tab-1": "Saturn",
      "tab-2": "Earth",
      "tab-3": "Jupiter",
      "tab-4": "Mercury",
      "tab-5": "Uranus",
      "tab-6": "Venus",
      "tab-7": "Mars",
      "tab-8": "Neptune"
    }
    function init() {
      info = document.getElementById("information");
      fetch("planets.json").then(r=>r.json()).then(r=>planets = r);
      let i = 1;
      while(i <= 8) {
        document.getElementById(`tab-${i}`).addEventListener("click",changeTab);
        i++;
      }
    }

    function changeTab(e) {
      let i = 1;
      while(i <= 8) {
        document.getElementById(`tab-${i}`).classList.remove("active");
        i++;
      }
      document.getElementById(e.target.id).classList.add("active");
      let p = mapping[e.target.id];
      p = planets.filter(x=>x.name == p)[0];
      let heading = document.createElement("h2");
      heading.innerText = p.name;
      let summary = document.createElement("ul");
      for(let k of Object.keys(p.summary)) {
        // not the most elegant solution but easy
        summary.innerHTML += `<li><b>${k}:</b> ${p.summary[k]}</li>\n`
      }
      let details = document.createElement("p");
      details.innerText = p.details;
      let ul = document.createElement("hr");
      info.innerHTML = "";
      info.appendChild(heading);
      info.appendChild(ul);
      info.appendChild(details);
      info.appendChild(summary);
    }
}());
```

Challenge Exercise: webworker

Navigate to http://127.0.0.1:8080/labs/lab12/webworker/

Code is available at `labs/lab12/webworker/main.js` and `labs/lab12/webworker/worker.js`

Use a webworker to constantly get gifs of cats!

see http://127.0.0.1:8080/labs/lab12/webworker/ for more details

There is no autotest and no automarking of this question.
When you have completed demonstrate your work to another student in your lab and ask them to enter a peer assessment here

It is prefered you do this during your lab, but if this is not possible you may demonstrate your work to any other COMP[29]041 student before Sunday midnight. Note, you must also submit the work with give.

When you are finished working on this exercise you must submit your work by running **give**:

```
$ give cs2041 lab12_js_webworker worker.js main.js
```

Sample solution for `worker.js`

```
// your web worker goes here.

const API_URL = 'https://api.thecatapi.com/v1/images/search?&mime_types=image/gif';

const fetchImage = (url) => fetch(url).then(res => res.json());

const ready = (data) => {
    console.log(data);
    self.postMessage(data);
};

self.addEventListener('message', () => {
    fetchImage(API_URL)
    .then(([data]) => data)
    .then(ready);
});
```

Sample solution for `main.js`

```
(function() {
    'use strict';

    const worker = new Worker('worker_soln.js');
    const image = document.getElementById('cat');

    function getImage() {
        worker.postMessage('getImg');
    }

    function postImage({data: { url }}) {
        image.src = url;
    }

    worker.addEventListener('message', postImage);

    setInterval(getImage, 10000);

 }());
```

Submission

When you are finished each exercises make sure you submit your work by running **give**.
You can run **give** multiple times. Only your last submission will be marked.

Don't submit any exercises you haven't attempted.

If you are working at home, you may find it more convenient to upload your work via give's web interface.

Remember you have until **Monday 22 October 23:59:59** to submit your work.

You cannot obtain marks by e-mailing lab work to tutors or lecturers.

You check the files you have submitted here

Automarking will be run by the lecturer several days after the submission deadline for the test, using test cases that you haven't seen: different to the test cases `autotest` runs for you.

(Hint: do your own testing as well as running `autotest`)

After automarking is run by the lecturer you can view it here the resulting mark will also be available via via give's web interface

Lab Marks

When all components of a lab are automarked you should be able to view the the marks via give's web interface or by running this command on a CSE machine:

```
$ 2041 classrun -sturec
```

The lab exercises for each week are worth in total 1 mark.
The best 10 of your 11 lab marks will be summed to give you a mark out of 9. If their sum exceeds 9 - your mark will be capped at 9.

- You can obtain full marks for the labs without completing challenge exercises
- You can miss 1 lab without affecting your mark.