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 `lab09` and change to this directory with these comamnds:

```
$ mkdir lab09
$ cd lab09
```

Exercise: fetch_and_list

If you are working at CSE, explode the zip file containing the files for this activity:

```
$ unzip /web/cs2041/19T2/activities/js_fetch_and_list/fetch_and_list.zip
```

If you are working on your computer, download fetch_and_list.zip and unzip it .

Then in one window use Python to start a HTTP server for the exercises

```
$ cd fetch_and_list
$ python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

If you access the URL printed by Python you will see instructions for the exercise.
Remember to refresh the web page in your browser (`control-R` inchrome and firefox) when you make a change to the Javascript so it gets loaded. You can stop the server by pressing `control-c`

You do not change **index.html** but you should read its source:

```
$ cd fetch_and_list
$ more index.html
```

Notice **index.html** loads **fetch_and_list.js**. You need to edit **fetch_and_list.js**

```
$ vi fetch_and_list.js
```

Follow the instructions in the HTML provided. Your task is to modify the code in **fetch.js**

Remember to refresh the HTML page when you make a change to the JavaScript your page reflects your changes, 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(2041|9044) student before Tuesday 06 August 17:59:59. 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 lab09_js_fetch_and_list fetch.js
```

Sample solution for `fetch.js`

```
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);
}

export default function runApp() {
  output = document.getElementById('output');

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

Exercise: fetch_and_list_2

If you are working at CSE, explode the zip file containing the files for this activity:

```
$ unzip /web/cs2041/19T2/activities/js_fetch_and_list_2/fetch_and_list_2.zip
```

If you are working on your computer, download fetch_and_list_2.zip and unzip it .

Then in one window use Python to start a HTTP server for the exercises

```
$ cd fetch_and_list_2
$ python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

If you access the URL printed by Python you will see instructions for the exercise.
Remember to refresh the web page in your browser (`control-R` inchrome and firefox) when you make a change to the Javascript so it gets loaded. You can stop the server by pressing `control-c`

You do not change **index.html** but you should read its source:

```
$ cd fetch_and_list_2
$ more index.html
```

Notice **index.html** loads **fetch_and_list_2.js**. You need to edit **fetch_and_list_2.js**

```
$ vi fetch_and_list_2.js
```

Your task is to modify the code in **fetch.js**. Follow the instructions on the web page.

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(2041|9044) student before Tuesday 06 August 17:59:59. 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 lab09_js_fetch_and_list_2 fetch.js
```

Sample solution for `fetch.js`

```
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;
}

export default function runApp() {
  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

If you are working at CSE, explode the zip file containing the files for this activity:

```
$ unzip /web/cs2041/19T2/activities/js_pretty_pictures/pretty_pictures.zip
```

If you are working on your computer, download pretty_pictures.zip and unzip it .

Then in one window use Python to start a HTTP server for the exercises

```
$ cd pretty_pictures
$ python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

If you access the URL printed by Python you will see instructions for the exercise.
Remember to refresh the web page in your browser (`control-R` inchrome and firefox) when you make a change to the Javascript so it gets loaded. You can stop the server by pressing `control-c`

You do not change **index.html** but you should read its source:

```
$ cd pretty_pictures
$ more index.html
```

Notice **index.html** loads **pretty_pictures.js**. You need to edit **pretty_pictures.js**

```
$ vi pretty_pictures.js
```

Your task is to modify the code in **pretty_pictures.js**. Follow the instructions on the web page.

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

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(2041|9044) student before Tuesday 06 August 17:59:59. 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 lab09_js_pretty_pictures fetch.js
```

Sample solution for `fetch.js`

```
function addPost({ url }) {
  const output = document.getElementById('output');
  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 output = document.getElementById('output');
  const loading = document.getElementById('loading');
  const children = [];

  for (let child of output.childNodes)
    if (child.className === 'img-post') children.push(child);
  children.forEach(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.forEach(addPost);
  });
}

export default function runApp() {
  document
    .getElementById('more')
    .addEventListener('click', getMore);
}
```

Exercise: tabs

If you are working at CSE, explode the zip file containing the files for this activity:

```
$ unzip /web/cs2041/19T2/activities/js_tabs/tabs.zip
```

If you are working on your computer, download tabs.zip and unzip it .

Then in one window use Python to start a HTTP server for the exercises

```
$ cd tabs
$ python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

If you access the URL printed by Python you will see instructions for the exercise.
Remember to refresh the web page in your browser (`control-R` inchrome and firefox) when you make a change to the Javascript so it gets loaded. You can stop the server by pressing `control-c`

You do not change **index.html** but you should read its source:

```
$ cd tabs
$ more index.html
```

Notice **index.html** loads **tabs.js**. You need to edit **tabs.js**

```
$ vi tabs.js
```

Your task is to modify the code in **tabs.js**. Follow the instructions on the web page.

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(2041|9044) student before Tuesday 06 August 17:59:59. 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 lab09_js_tabs tabs.js
```

Sample solution for `tabs.js`

```
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 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);
}

export default function runApp() {
  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++;
  }
}
```

Challenge Exercise: webworker

If you are working at CSE, explode the zip file containing the files for this activity:

```
$ unzip /web/cs2041/19T2/activities/js_webworker/webworker.zip
```

If you are working on your computer, download webworker.zip and unzip it .

Then in one window use Python to start a HTTP server for the exercises

```
$ cd webworker
$ python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

If you access the URL printed by Python you will see instructions for the exercise.
Remember to refresh the web page in your browser (`control-R` inchrome and firefox) when you make a change to the Javascript so it gets loaded. You can stop the server by pressing `control-c`

You do not change **index.html** but you should read its source:

```
$ cd webworker
$ more index.html
```

Notice **index.html** loads **webworker.js**. You need to edit **webworker.js**

```
$ vi webworker.js
```

In this task you create a webworker to constantly get gifs of cats!

You will need to modify main.js and worker.js

Follow the instructions on the web page.

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(2041|9044) student before Tuesday 06 August 17:59:59. 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 lab09_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 fetchImageMetaData = url => fetch(url).then(res => res.json());

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

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

Sample solution for `main.js`

```
export default function runApp() {
  const worker = new Worker('worker.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 **Tuesday 06 August 17:59:59** 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 1.2 marks.

Usually each lab exercise will be worth the same - for example if there are 5 lab exercises each will be worth 0.4 marks.

Except challenge exercises (see below) will never total more than 20% of each week's lab mark.

All of your lab marks for weeks 1-10, will be summed to give you a mark out of 12.

If their sum exceeds 9 - your total mark will be capped at 9.

Running Autotests On your Own Computer

An experimental version of autotest exists which may allow you to run autotest on your own computer.
If you are running Linux, Windows Subsystem for Linux or OSX. These commands might let you run autotests at home.

```
$ sudo wget https://cgi.cse.unsw.edu.au/~cs2041/19T2/resources/home_autotest -O/usr/local/bin/2041_autotest
$ sudo chmod 755 /usr/local/bin/2041_autotest
$ 2041_autotest shell_snapshot
```

Autotest itself needs Python 3.6 (or later) installed.
Particular autotests may require other software install, e.g. autotests of perl programs require Perl installed (of course).

The legit autotests need python3.7, git & binfmt-support installed.

The program embeds the autotests themselves, so you'll need to re-download if autotests are changed, added, fixed, ...

If it breaks on your computer post on the class forum and we'll fix if we can, but this is very definitely experimental.