Make sure the directory is called `joyofcoding`!

```
npm init -y
```

```
npm link jspm
npm install live-server
```

Start live-server and load up in the browser.

Initialise jspm

```
jspm init .
```

Quick set up

Set up SystemJS package main as `index.js`

```
mkdir src
```

```
vim index.html
<!DOCTYPE html>
<html>
  <head>
    <title>Open Sauce</title>
    <script src="jspm_packages/system.js"></script>
    <script src="jspm.config.js"></script>
    <script>
      System.import('joyofcoding');
    </script>
  </head>
  <body>
  </body>
</html>
```

```
vim src/index.js
console.log('hello world');
```

ES2015 module support

```
vim src/make-red.js
export default function makeRed() {
  document.body.style.backgroundColor = 'red';
}
```

```
vim src/index.js
import makeRed from './make-red';
makeRed();
```

Fetching data

```
jspm install npm:whatwg-fetch
```

```
vim src/github-api.js
import 'whatwg-fetch';

export const fetchUserInfo = (username) => {
  return fetch(`https://api.github.com/users/${username}`)
    .then(d => d.json());
}
```

```
vim src/index.js
import makeRed from './make-red';
import { fetchUserInfo } from './github-api';

makeRed();

console.log(`2 + 2 is ${2 + 2}`);

fetchUserInfo('jackfranklin').then(d => {
  document.body.innerHTML = JSON.stringify(d, null, 4);
});
```

Bundling files up into one

Once you get past a certain number of requests this will get slow, so we can rebuild our browser build every time a file changes.

This is cached so it stays mega performant even at large scale applications.

```
jspm bundle joyofcoding -wid
```

Now refresh and view the network requests.

Now make a change and see that the build gets rebuilt (change ‘red’ to ‘blue’ or something).

This even works if you install things (pending my PR) - install jQuery and prove it.

```
vim src/make-red.js
import $ from 'jquery';

export default function makeRed() {
  $('body').css('backgroundColor', 'red');
}
```

Building something “proper”

```
jspm install npm:yo-yo
```

```
import makeRed from './make-red';
import { fetchUserInfo } from './github-api';
import yo from 'yo-yo';

makeRed();

const template = user => {
  return yo`<div>
    User: ${user.name} works for ${user.company}
  </div>`;
}

fetchUserInfo('jackfranklin').then(user => {
  const outputElement = template(user);
  document.body.appendChild(outputElement);
});
```

Note that build.js is a bit larger now!

Interactive App

```
import makeRed from './make-red';
import { fetchUserInfo } from './github-api';
import $ from 'jquery';
import yo from 'yo-yo';

makeRed();

const renderUser = user => {
  if (user) {
    return yo`<p>
      User: ${user.name} works for ${user.company}
    </p>`;
  } else {
    return yo`<p>No user!</p>`;
  }
}

const template = (user, buttonClick) => {
  return yo`<div>
    <input type="text" value="${user && user.login || ''}" data-user-input />
    <button onclick=${buttonClick}>Update!</button>
    ${ renderUser(user) }
  </div>`;
}

const update = () => {
  const username = $('[data-user-input]').val();
  fetchUserInfo(username).then(d => {
    const newOutput = template(d, update);
    yo.update(outputElement, newOutput);
  });
}

const outputElement = template(undefined, update);
document.body.appendChild(outputElement);
```

Building for production

```
jspm bundle joyofcoding dist/bundle.js --minify
```

```
vim dist/index.html
<!DOCTYPE html>
<html>
  <head>
    <title>Open Sauce</title>
    <script src="jspm_packages/system.js"></script>
    <script src="jspm.config.js"></script>
    <script src="dist/bundle.js"></script>
  </head>
  <body>
    <script>
      System.import('joyofcoding');
    </script>
  </body>
</html>
```

Caching vendor files

Vendor libraries won’t change very much: for us we could keep Yo-Yo and whatwg-fetch in a separate bundle which will change infrequently and can be cached.

First bundle our vendor files:

```
jspm bundle yo-yo + whatwg-fetch + jquery dist/vendor.js --minify
```

Now build the main file:

```
jspm bundle joyofcoding - yo-yo - whatwg-fetch - jquery dist/app.js --minify
```

And update the prod HTML file:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Open Sauce</title>
    <script src="../jspm_packages/system.js"></script>
    <script src="../jspm.config.js"></script>
    <script src="vendor.js"></script>
    <script src="app.js"></script>
  </head>
  <body>
    <script>
      System.import('joyofcoding');
    </script>
  </body>
</html>
```

Self executing build

We can improve further by building a bundle that includes SystemJS, the config and everything required:

```
jspm build joyofcoding dist/sfx.js --minify
```

And update our HTML file:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Open Sauce</title>
  </head>
  <body>
    <script src="sfx.js"></script>
  </body>
</html>
```

And now we have a smaller footprint and fewer scripts to run.