```
npm init -y
```

```
npm link jspm
```

Start live-server and load up in the browser.

Initialise jspm

```
jspm init .
```

Pick quick setup, say no to dev version, set local 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('client-side-jspm');
    </script>
  </head>
  <body>
  </body>
</html>
```

```
vim src/index.js
console.log('hello world');
// change it to some ES2015
```

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 client-side-jspm -wid
```

Now create a new file

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

```
vim src/index.js
import makeRed from './make-red';
makeRed();
console.log(`2 + 2 is ${2 + 2}`);
```

Note how there's no extra requests in the browser but now stuff just works.

Building something "proper"

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

```
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);
});
```

Note that build.js is a bit larger now!

Interactive App

```
import makeRed from './make-red';
import { fetchUserInfo } from './github-api';
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 = document.querySelector('[data-user-input]').value;
  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 client-side-jspm dist/bundle.js --minify
```

```
vim prod-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('client-side-jspm');
    </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 dist/vendor.js --minify
```

Now build the main file:

```
jspm bundle client-side-jspm - yo-yo - whatwg-fetch 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="dist/vendor.js"></script>
    <script src="dist/app.js"></script>
  </head>
  <body>
    <script>
      System.import('client-side-jspm');
    </script>
  </body>
</html>
```