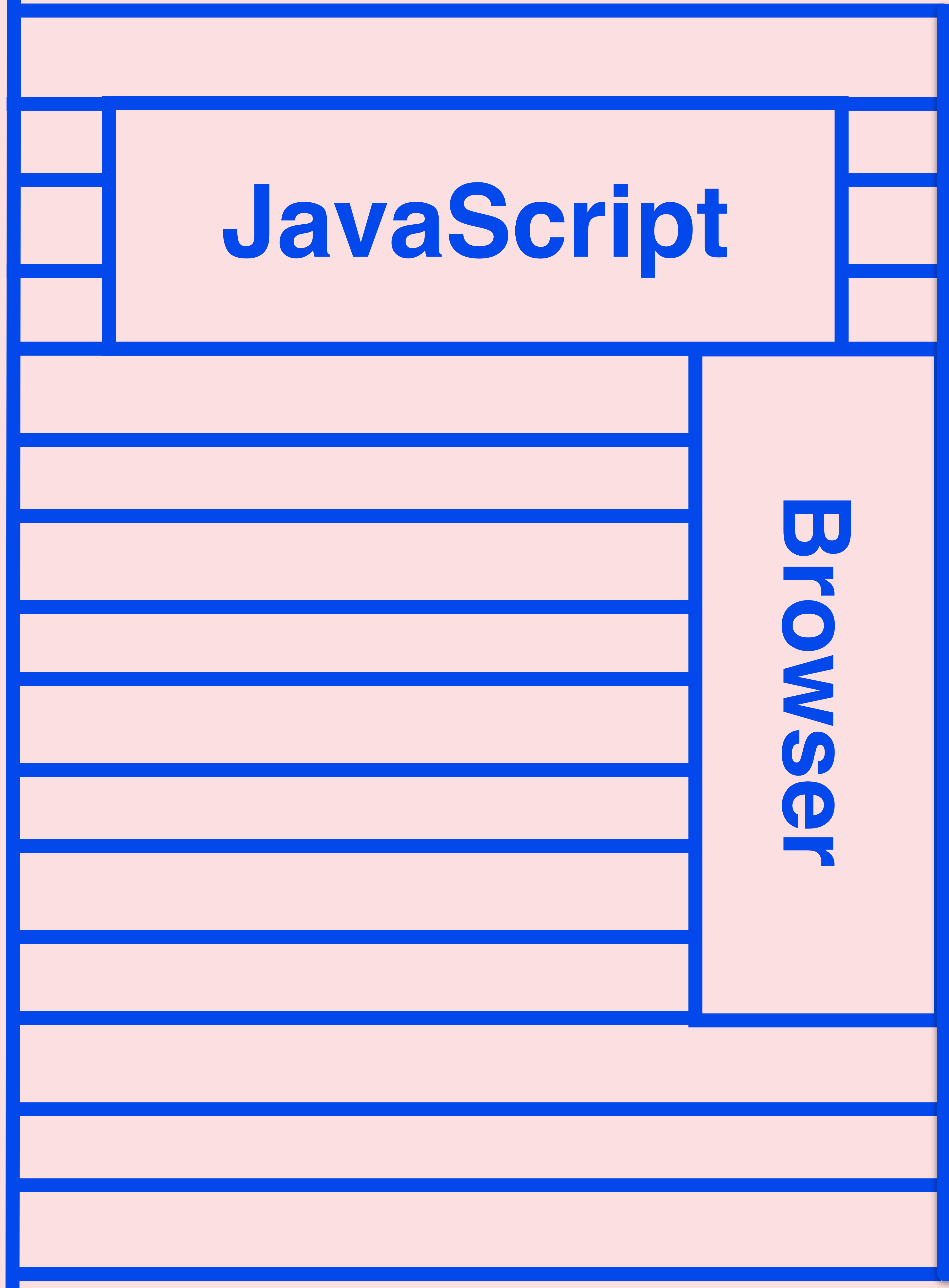
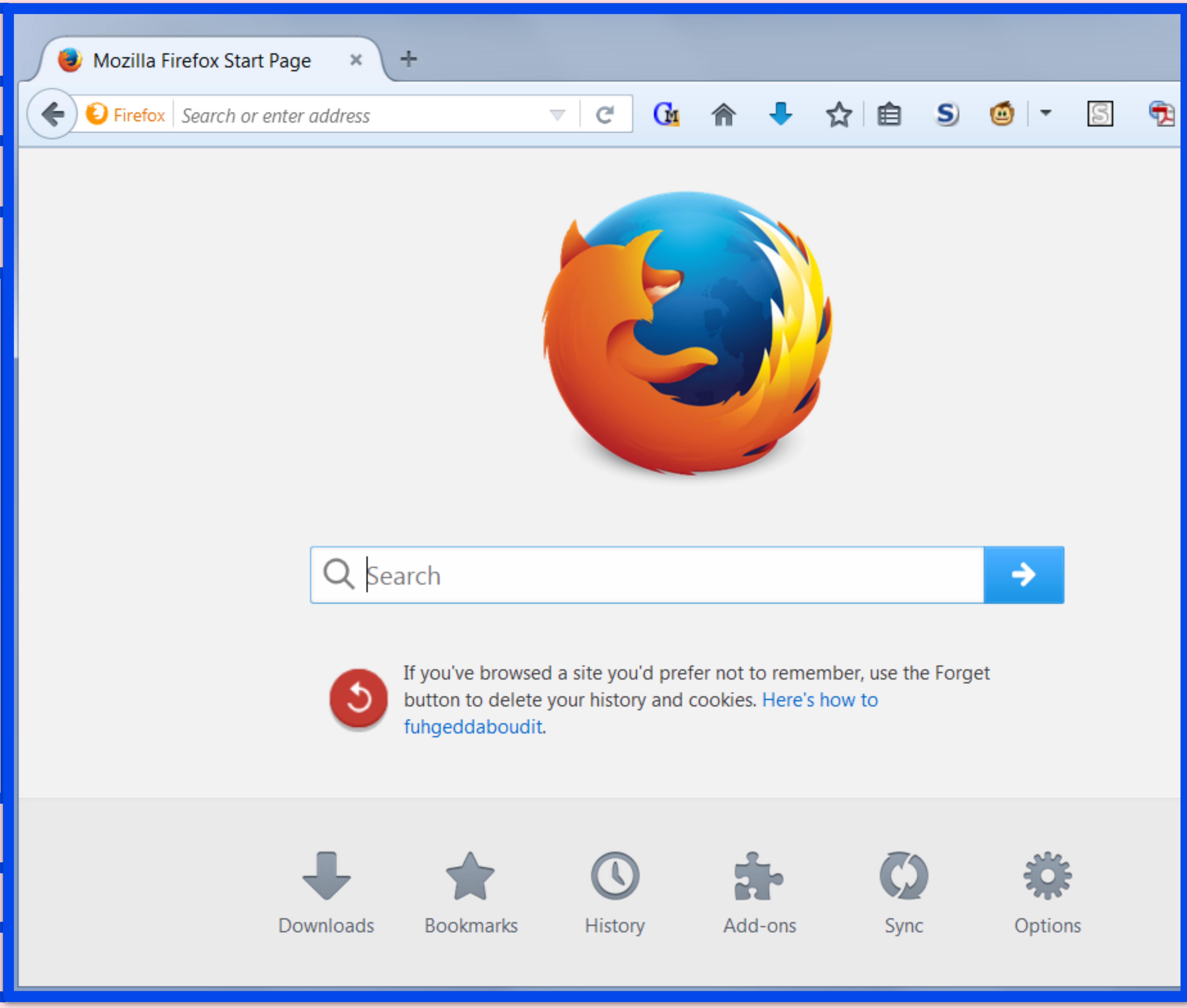
Week 5

Making Desktop Apps



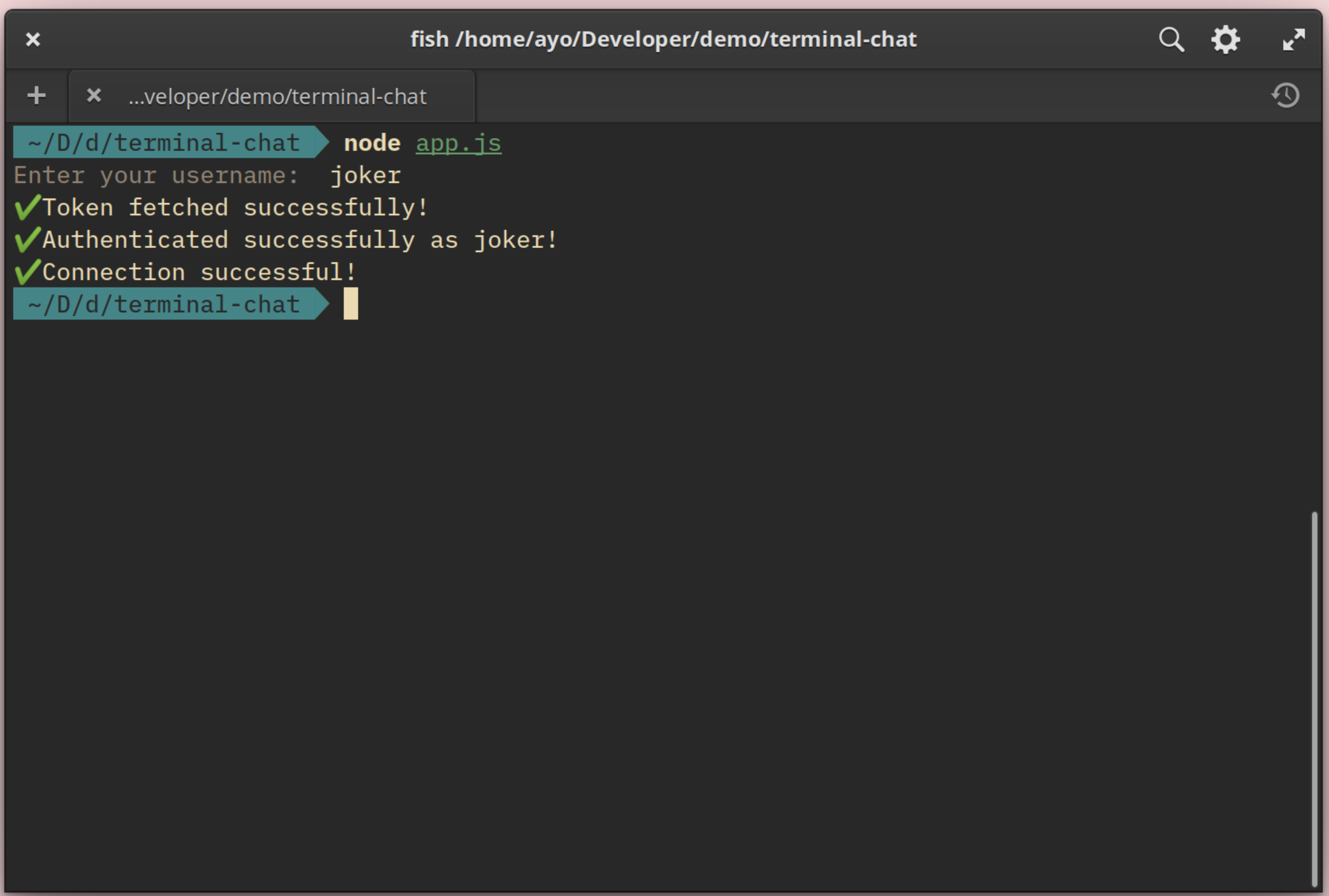


If you've browsed a site you'd prefer not to remember, use the Forget button to delete your history and cookies. Here's how to fuhgeddaboudit.

Electron

JavaScript

Node.js



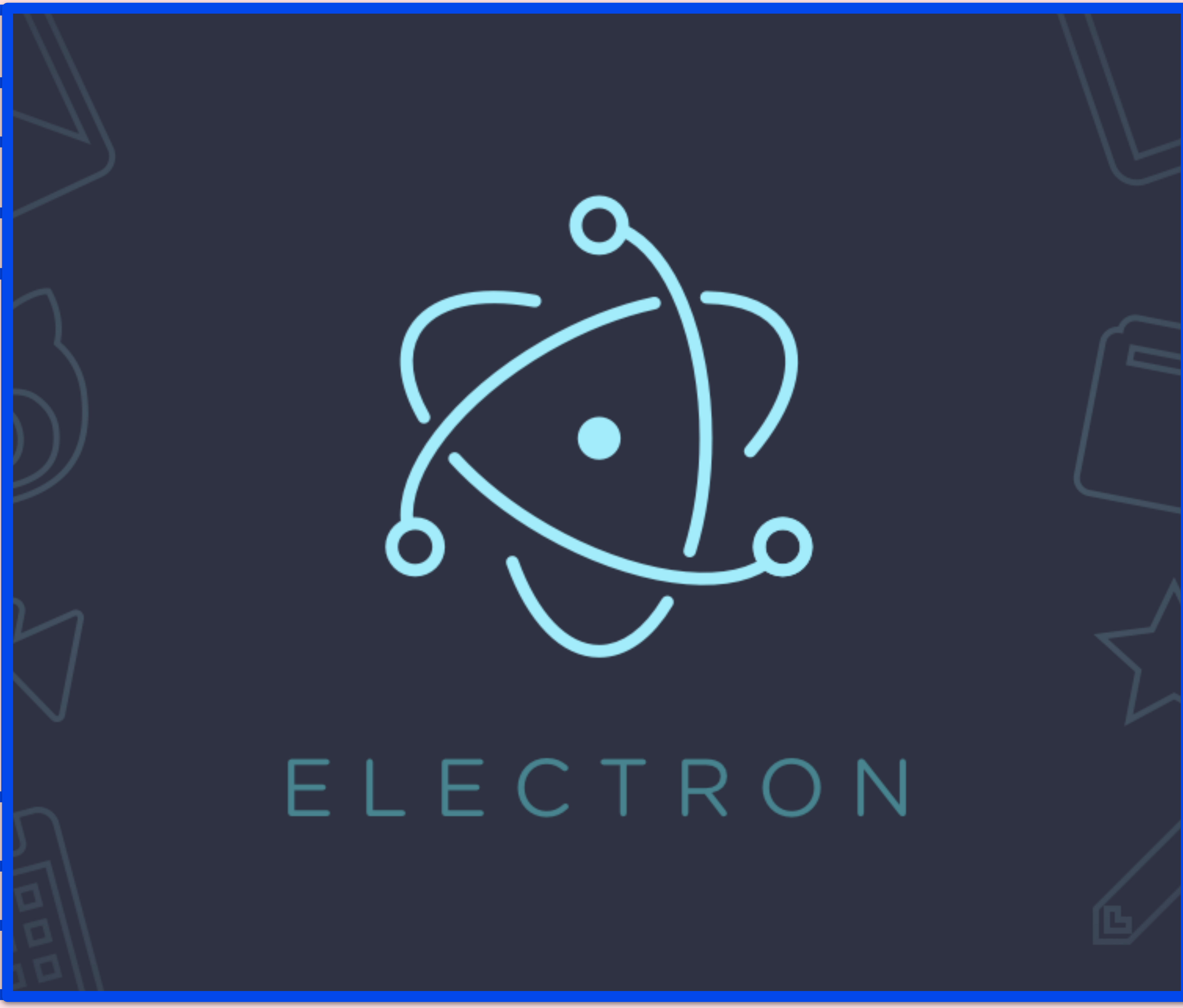
fish /home/ayo/Developer/demo/terminal-chat

...veloper/demo/terminal-chat

```
~/D/d/terminal-chat  node app.js
Enter your username:  joker
✓Token fetched successfully!
✓Authenticated successfully as joker!
✓Connection successful!
~/D/d/terminal-chat
```

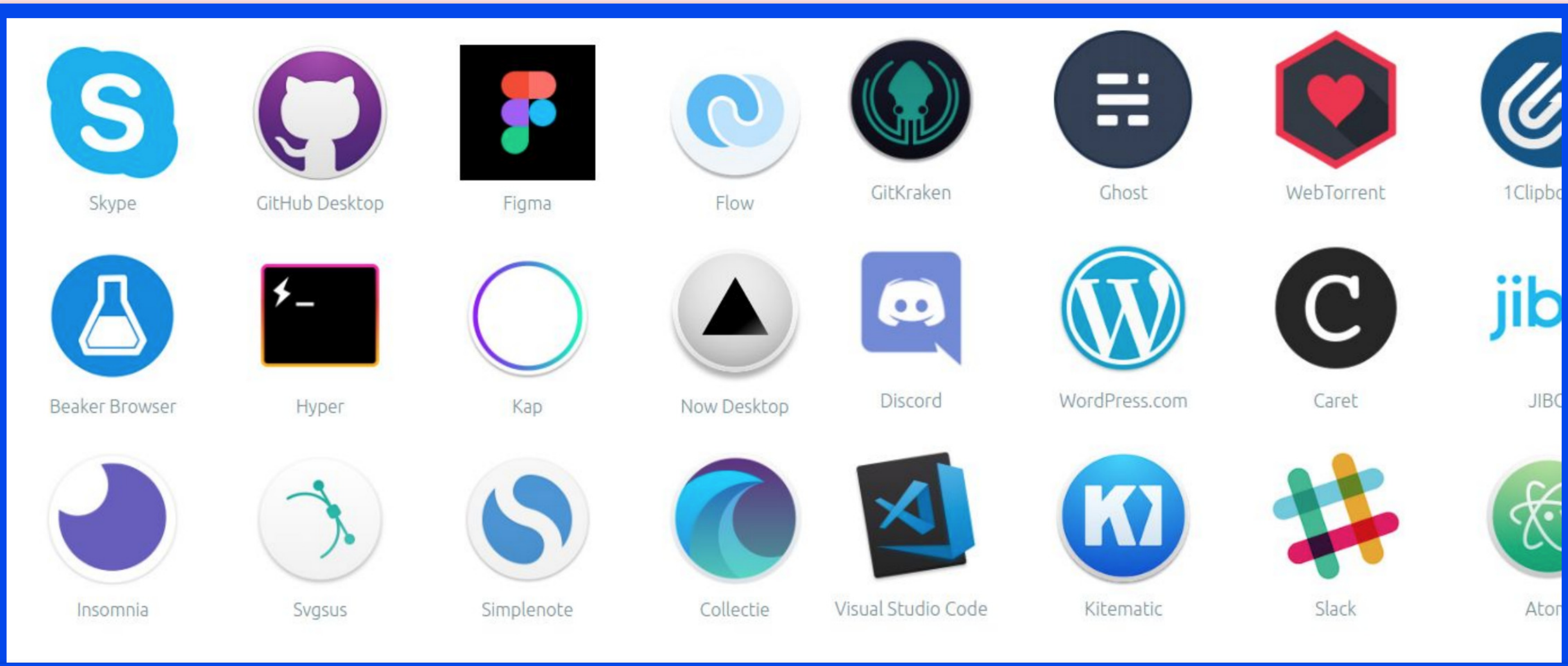
JavaScript

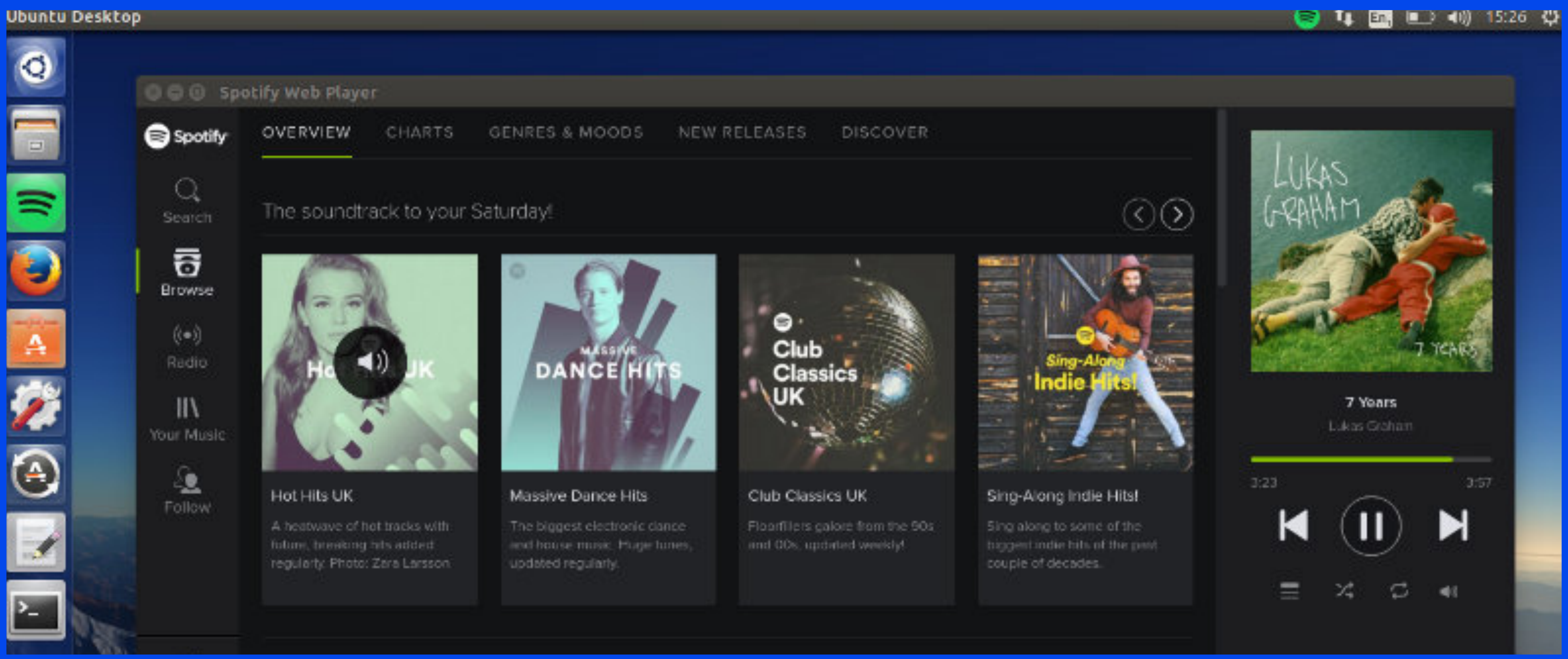
in BOTH



JavaScript

in BOTH





The soundtrack to your Saturday!

Hot Hits UK

A heatwave of hot tracks with future, breaking hits added regularly. Photo: Zara Larsson

Massive Dance Hits

The biggest electronic dance and house music. Huge tunes, updated regularly.

Club Classics UK

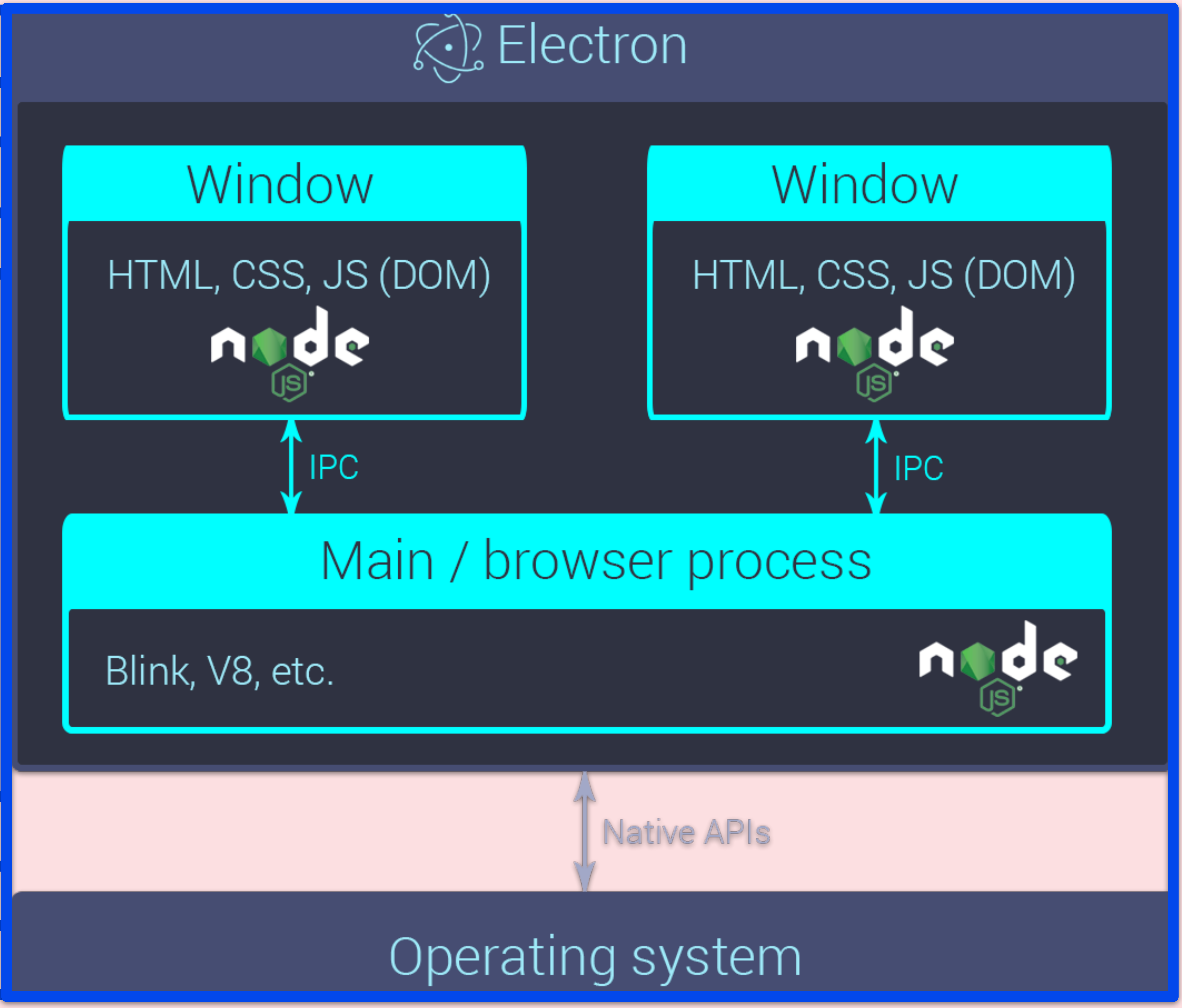
Floorfillers galore from the 90s and 00s, updated weekly!

Sing-Along Indie Hits!

Sing along to some of the biggest indie hits of the past couple of decades.

Setup

Electron



Using the Quickstart Repo

We can skip a lot of the tedious steps of the setup process with

```
> cd [path-to-cloned-repo]
> npm i (or npm install)

> npm run start
```

//instructions are also in the readme.md

Main.ts

This is the main Node.js process which hosts the rest of the application– creating the application window and responding to and OS events or apis

```
app.on("ready", () => {
  createWindow();

  app.on("activate", function () {
    // On macOS it's common to re-create a window in the app when the
     // dock icon is clicked and there are no other windows open.
     if (BrowserWindow.getAllWindows().length === 0) createWindow();
  });
});
```

Renderer.ts

This code is run within the index.html file and runs the same way our existing code has run before

```
> cd [path-to-cloned-repo]
> npm i (or npm install)
> npm start
```

Inter-Process Communication

We can also communicate between the main and render processes to let the view of our app react to changes based on desktop actions or apis

https://www.electronjs.org/docs/latest/tutorial/ipc

```
const { contextBridge, ipcRenderer } = require('electron')


contextBridge.exposeInMainWorld('electronAPI', {

  handleCounter:(callback)=>

    ipcRenderer.on('update-counter', callback)

})
```

Build for distribution

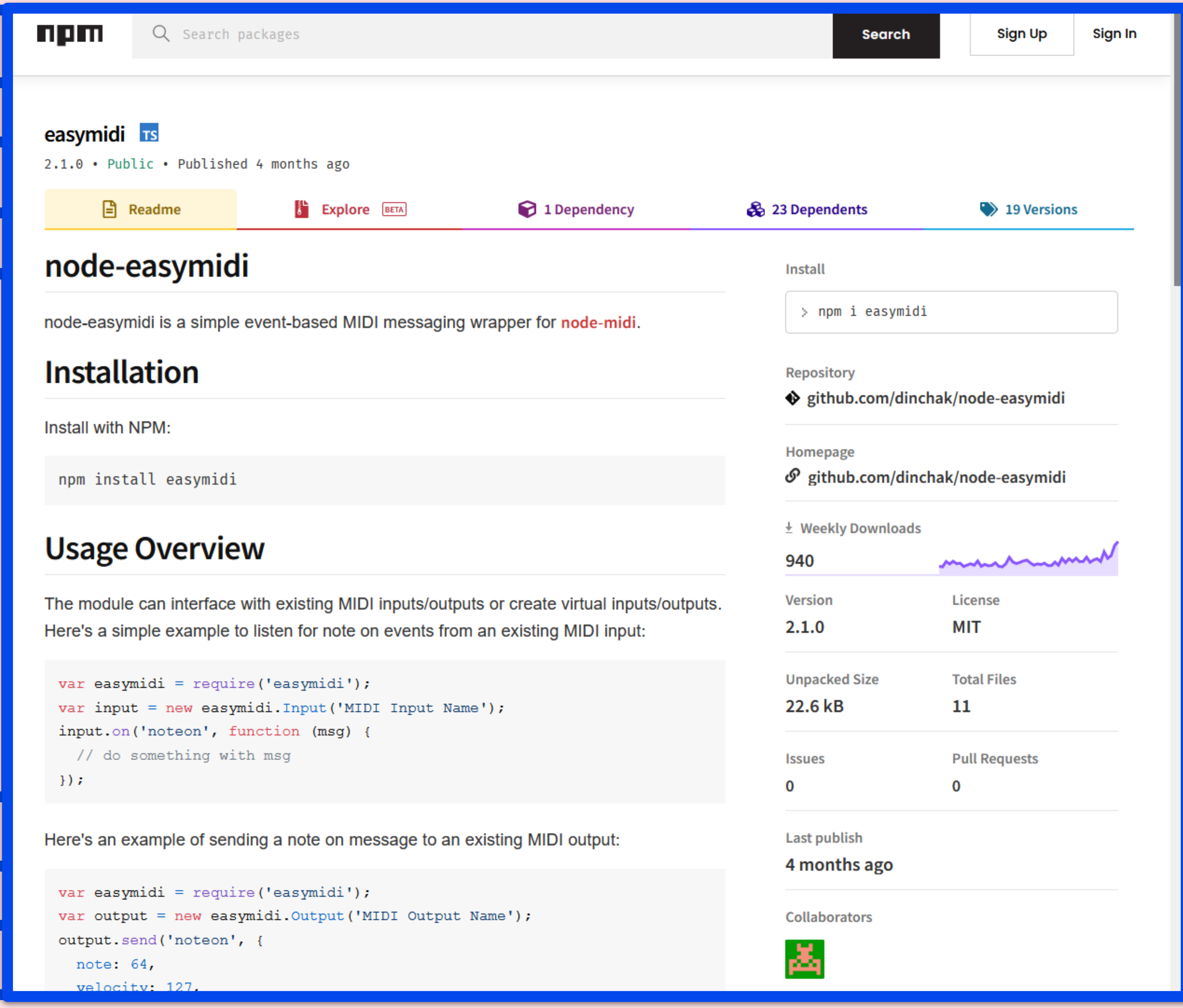
We can then take our code, and fully compile it into a desktop app for many different OSes

https://www.electronforge.io/

```
> npm run build
```

Next Week

via MIDI



easymidi TS

2.1.0 • Public • Published 4 months ago

node-easymidi

node-easymidi is a simple event-based MIDI messaging wrapper for node-midi.

Installation

Install with NPM:

```
npm install easymidi
```

Usage Overview

The module can interface with existing MIDI inputs/outputs or create virtual inputs/outputs. Here's a simple example to listen for note on events from an existing MIDI input:

```
var easymidi = require('easymidi');
var input = new easymidi.Input('MIDI Input Name');
input.on('noteon', function (msg) {
  // do something with msg
});
```

Here's an example of sending a note on message to an existing MIDI output:

```
var easymidi = require('easymidi');
var output = new easymidi.Output('MIDI Output Name');
output.send('noteon', {
  note: 64,
  velocity: 127,
```

Install

```
> npm i easymidi
```

Repository
github.com/dinchak/node-easymidi

Homepage
github.com/dinchak/node-easymidi

Weekly Downloads
940

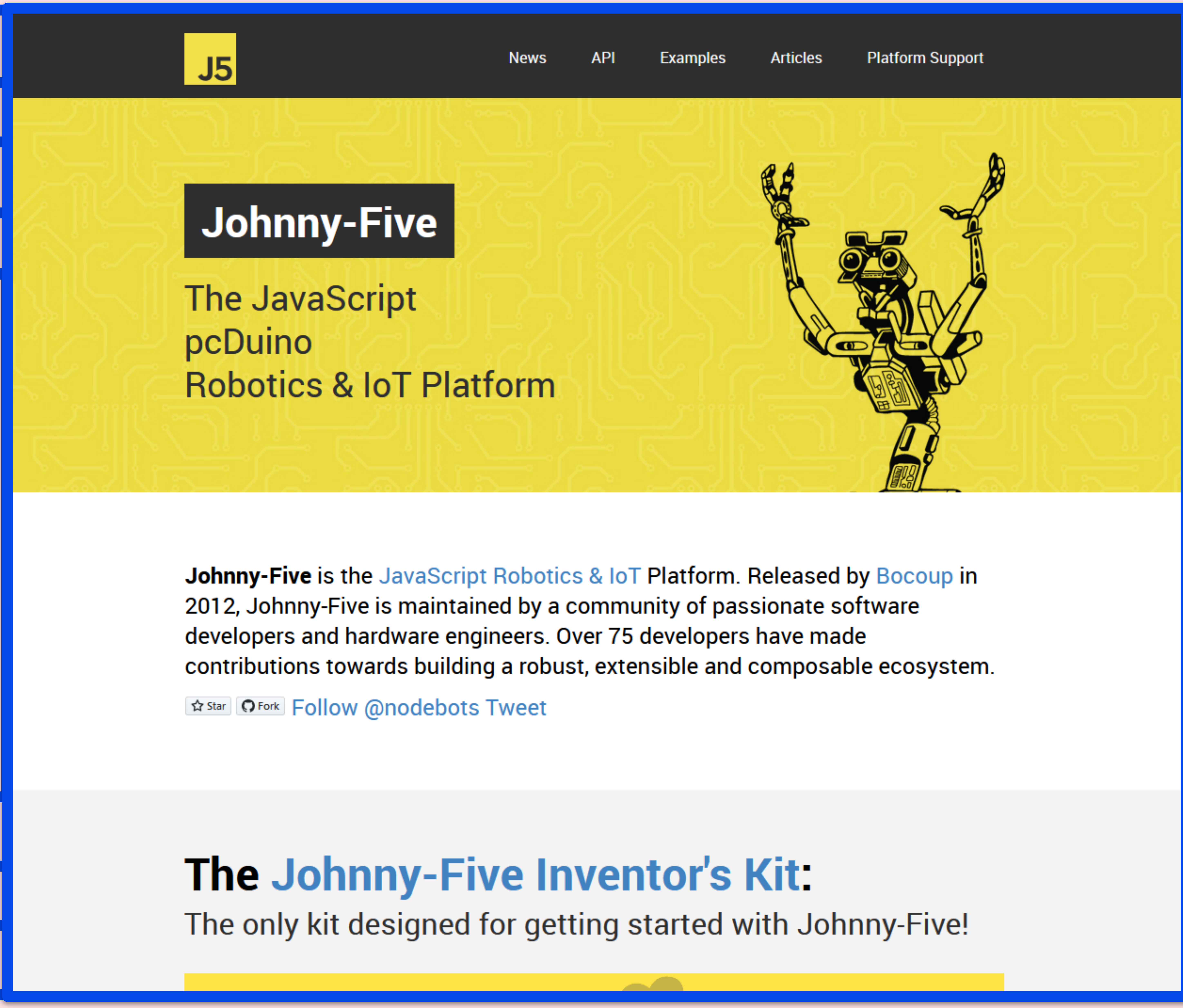
Version	License
2.1.0	MIT
Unpacked Size	Total Files
22.6 kB	11
Issues	Pull Requests
0	0

Last publish
4 months ago

Collaborators

Next Week

PComp



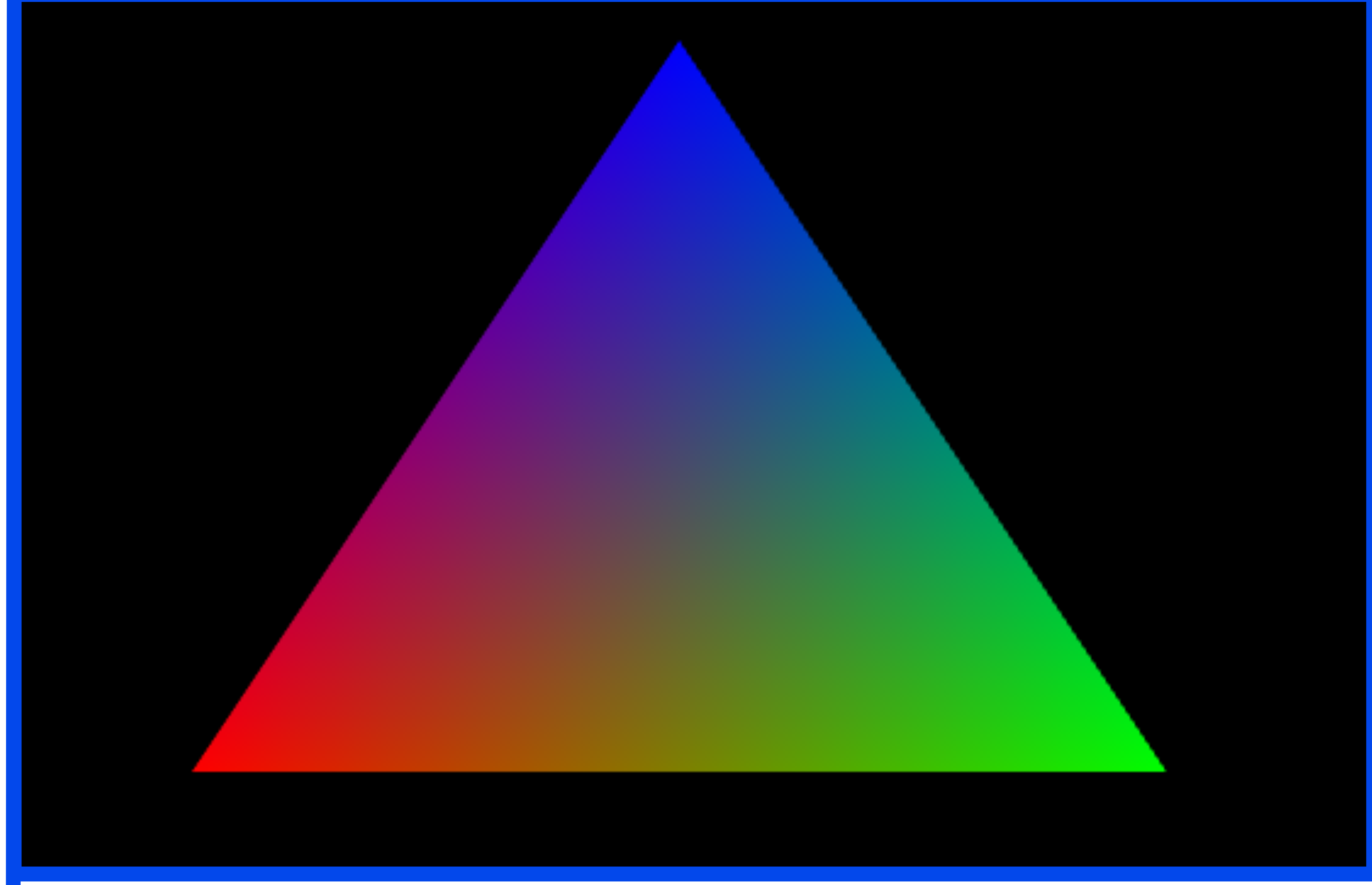
Johnny-Five

The JavaScript
pcDuino
Robotics & IoT Platform

Johnny-Five is the JavaScript Robotics & IoT Platform. Released by Bocoup in 2012, Johnny-Five is maintained by a community of passionate software developers and hardware engineers. Over 75 developers have made contributions towards building a robust, extensible and composable ecosystem.

The Johnny-Five Inventor's Kit:

The only kit designed for getting started with Johnny-Five!



Homework for 3/6

Midterm

Your midterm assignment is designed to demonstrate competency in the fundamentals of WebGL programming we have been learning with Pixi.js. You will make your own interactive, screen-based artwork. Your sketch needs to contain at least a few objects (they can be from the same class) that behave independently from one another or compose together to facilitate a broader intended experience.

Taking into consideration responsiveness, accessibility, and aesthetics there should be some form of interaction for your audience. This could be through, mouse movements, UI objects, external inputs etc. Your code must be clearly and thoroughly commented throughout - indicating what is your original contribution and where you have derived inspiration from somewhere else.

You'll have 10 minutes to present and collect feedback in class. Be sure to include documentation of your process and exploration when presenting. Keep in mind that giving feedback to classmates also contributes to your participation grade.