Google Summer of Code 2017 with the Fluid Project

Overview

Fluid is an open source community of designers and developers who help improve the usability and accessibility of the open web. We contribute to a variety of open source projects (such as jQuery UI), and we work on a few projects of our own: the Design Handbook, a guidebook of techniques for improving usability, and Infusion, a JavaScript application framework for developing flexible user interfaces.

Fluid Infusion is built on top of jQuery, providing all the stuff you need to create user interfaces that are incredibly flexible, accessible, and easy-to-use. Infusion is an application framework and a suite of user interface components built with HTML, CSS, and JavaScript. In contrast to many other user interface toolkits, Infusion components aren't black boxes--they're built to be modified, adapted, and changed to suit your application or context. Taking a "one size fits one" approach, Infusion even lets end-users customize their experience with the UI Options component.

We're looking for students to collaborate with us on the Google Summer of Code 2017 program. Working with Fluid gives you a chance to learn more about accessibility and usability while writing code with cutting-edge open web technologies. Create cool stuff and make a real impact on users at the same time!

For information about the various ways we communicate with each other, see our Get Involved wiki page.

Getting Started

Make sure to read through the Getting Started on GSoC page for information on joining the Fluid community and preparing to work on a GSoC project.

Projects

Cross-Platform Audio Engine for Node.js

Node.js is a library for developing server-side and desktop applications using JavaScript. The Fluid Project community uses Node.js for (among other things) developing applications for small, resource-constrained devices such as the C.H.I.P. and the Raspberry Pi. In particular, we are developing sonification and sound processing applications to support inclusive learning and culture. However, there is currently no reliable, cross-platform, low latency solution for generating audio using Node.js. One popular library, node-speaker, is only sporadically maintained and has very high latency and CPU usage, making it unsuitable for real-time audio streams.

This project will entail the development of a new Node.js native module in C++ that uses an existing cross-platform audio API such as Port Audio, libsoundic , or OpenAL. This module should support macOS, Windows, and Linux. It should include the following features:

1. The ability to enumerate and list audio devices
2. The ability to specify the sample formats, including integer and floating point numbers at 8, 16, 24, and 32-bit resolution
3. Configurable buffer sizes/latency, including reliable support for < 10ms latency
4. A callback-based API that allows JavaScript code to provide blocks of raw audio samples

Difficulty: Medium

Mentor: Colin Clark

IRC: colinclark

Skills: Knowledge of C++; basic familiarity with Node.js and how native modules are written for Node.js

How to get started/getting to know us: Prospective students who are interested in this project should spend some time familiarizing themselves with writing native Node.js modules in C++ using Native Abstractions for Node.js (NAN). The NAN project provides excellent tutorials and background information, as does the Node.js documentation. To familiarize yourself with the requirements of the project, you may want to consider developing a very simple native module for Node.js. You should also familiarize yourself with various cross-platform audio libraries, and write a basic example client that generates some simple audio output using Port Audio, libsoundio, or a similar library.

JavaScript SoundFont 2 Parser and Synthesizer Engine

SoundFonts provide a means for packaging and distributing audio samples for use in wavetable synthesizers and samplers. They typically provide a variety of instrument sounds sampled at different pitches and octaves, making it easy to create realistic-sounding digital instruments. SoundFonts are particularly useful for data sonification, since they provide a simple and low-cost way to give users the ability to choose from a variety of instrumental sounds when creating their sound designs.

The Floe Project (Flexible Learning for Open Education) is developing new tools for sonification and data presentation using audio. These tools are based on Flocking, a framework for audio signal processing, synthesis, and music composition, which uses the Web Audio APIs now built into most modern web browsers.

However, neither Flocking nor the Web Audio API currently provides any support for parsing or playing back SoundFont-based instruments. While there are JavaScript libraries (such as MIDI.js and the soundfont-player library) that claim to provide SoundFont support, these are all based on an approach that extracts the sound files from a SoundFont with a fixed duration and envelope, significantly limiting the usefulness and expressiveness of the SoundFonts.

This project will entail the development of a robust, pure JavaScript SoundFont 2 parser as well as a Flocking unit generator that is capable of expressively playing a SoundFont. The student may choose to take the sf2-parser library written by Gree and modified by Colin Clark as the basis of the project. The parser should support all the major features of the SoundFont 2 specification, and should be able to parse standard .sf2 files and expose their contents in a data-oriented, JSON-style data structure. The playback unit generator should provide inputs that can control and modulate all of the essential parameters of a SoundFont. All code written should be accompanied by unit tests. Along the way, the student is encouraged to create an awesome musical demo of their work.

Difficulty: Medium

Mentor: Colin Clark

IRC: colinclark

Skills required: In-depth knowledge of JavaScript and web development. Familiarity with digital audio and MIDI.

How to get started/getting to know us: Prospective students who are interested in this project should spend some time familiarizing themselves with the SoundFont 2 specification and the sf2-parser library.

Add Rate Limiting Control for AChecker Web Service API

Project Description: AChecker is an online accessibility validator that provides a web service API for users to validate the given URL or content via http requests. Each user is given a web service ID that can be sent along with http requests to use the API. To avoid the abuse of AChecker web service API, rate limiting needs to be added to control the number of requests can be made at a certain period by one web service ID. Adding this feature requires:

1. Implement the rate limiting;
2. Add HTTP endpoints for querying rate limits, revoking and regenerating IDs etc;
3. Change the user profile UI to allow users to revoke/delete the compromised web service IDs and regenerate new IDs;
4. Change the AChecker Administrator's user management UI to allow administrators to revoke the compromised web service IDs.

AChecker Issues to start with:

- Empty exported reports
- Support HTM5 "placeholder" attribute
- CSS validation section shows the number of results without showing errors

More issues can be found at AChecker bug tracker.

Difficulty: Medium

Mentor: Cindy Li

IRC: cindyli

Skills required: PHP, Javascript, HTML, CSS. The experience of use and the understanding of the API rate limiting would be a bonus.

Automated Test Suite for AChecker

Project Description: AChecker is an online accessibility validator. The goal of this project is to add an automated test suite for AChecker. This test suite provides:

1. Backend PHP tests to verify the correctness of validation rules, web service API etc;
2. Front-end Javascript tests to ensure the integrity of the user interface and user interactions.

Both type of tests include unit tests, if time allows, integration tests.

This project requires the research on what PHP testing framework to use. We suggest to use QUnit as the javascript testing framework to be in sync with our other projects. The AChecker source code may need to be refactored to improve its testability.

Note: AChecker works with PHP 5.4.x. It hasn't been upgraded to work with new PHP versions.

AChecker Issues to start with:

- Empty exported reports
- Support HTM5 "placeholder" attribute
- CSS validation section shows the number of results without showing errors

More issues can be found at AChecker bug tracker.

Difficulty: Medium

Mentor: Cindy Li

IRC: cindyli

Skills required: PHP, Javascript, HTML, CSS.

Infusion Documentation

Most of the work we do here either uses or directly involves the Infusion Framework and Component Library. These links should get you started learning about Infusion, and should lead you to many more pages.

Contributing Code To Infusion
Infusion Documentation
Tutorial - Getting started with Infusion