New JavaScript console in p5.js web editor

Liang Tang, shinytang6@gmail.com

Project Abstract

p5.js web editor is an environment to write p5.js sketches with no need for downloading or installing any dependencies. With the purpose of making the web editor more user-friendly and effective to the potential users, the project will integrate a new JavaScript console into the p5.js web editor.

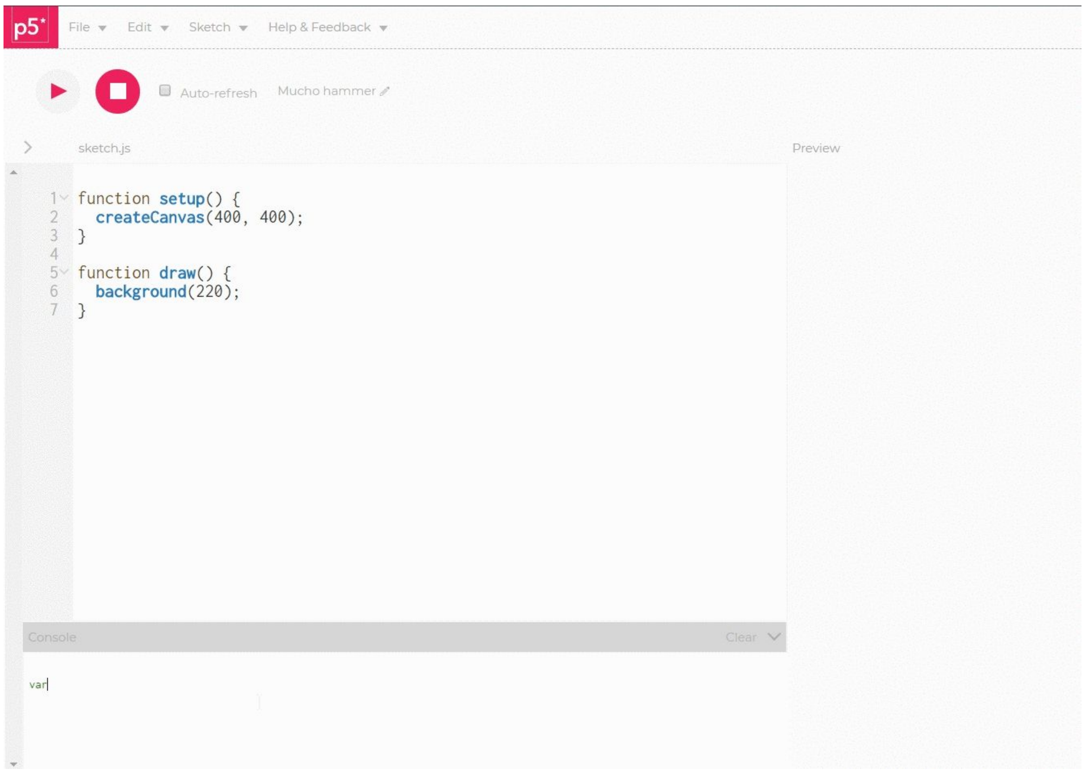
Project Description

The goal of the Processing is always to empower those who are less familiar with code to make creative work easily. Considering the setup of p5.js/processing still requires some knowledge of code, the most effective and useful way to achieve this goal is writing code in p5.js web editor. However, the console of the current version is not as user-friendly and effective as it can be. Hence, what I want to do is improving it, getting it accessible to more people who may need it.

These developments have two main target audiences. One of them is beginner coder who wants to get some useful and friendly logs information associated with the web page. And the other is developer who wants to interact with the web page by executing JavaScript expressions in the p5.js web editor.

Prototype

I have created a prototype when I am playing around the CodeMirror. It has the basic function of executing code from console. But its robustness, functionality as well as the style still need to be enhanced.



```
function setup() {
  createCanvas(400, 400);
}

function draw() {
  background(220);
}
```

```
var
```

You can see the prototype here:
https://github.com/shinytang6/p5.js-web-editor/tree/interactive-console

Implementation

According to the GSoC timeline, my work will be divided into three stages.

The first stage is to improve the current console. There are a few features similar to the chrome developer tool that we need to implement.

1. Display errors with linked line numbers and resource errors. Issue #468 (partly solved in my PR #579)
2. Fold/unfold console logged object/array.
3. Show repeated console logs only once (with a number displaying times logged).
4. Syntax highlighting of console logs.
5. Support some other console methods.

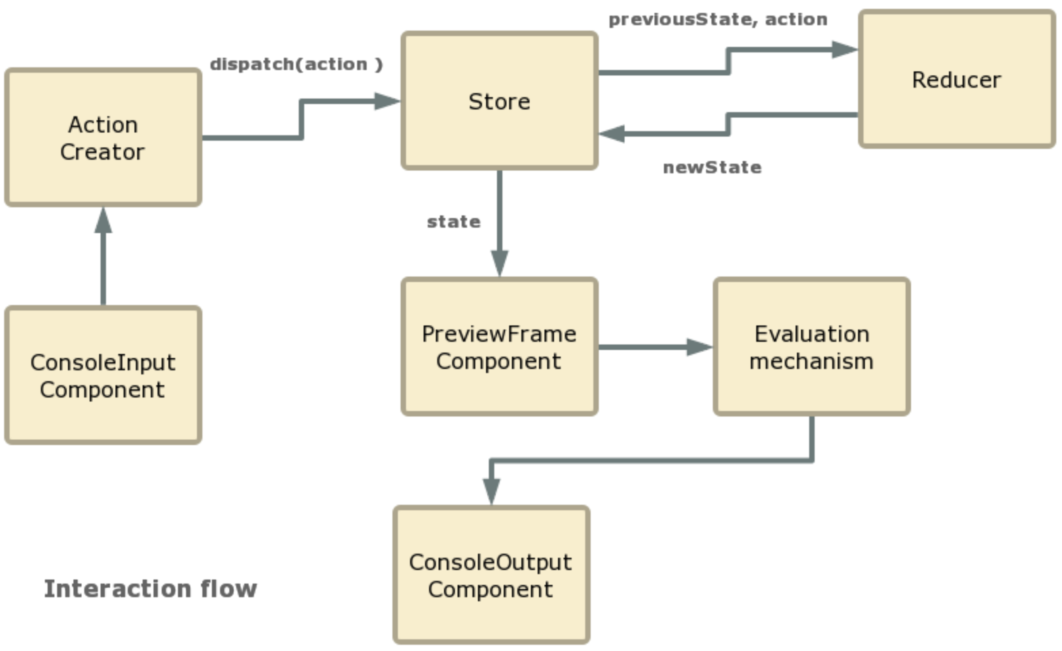
The second stage is to redesign a new Console component in the code base. Without processing, the current one only displays the error or console information hijacked from window console/injected script. To make the console interactive, adding two subcomponents to Console will make sense. One of them is ConsoleInput, the other is ConsoleOutput.

Steps related to ConsoleInput component

1. Contain another code editor (e.g CodeMirror) that allow users to type in.
2. Send the message to an Action once users enter a line.
3. Create a new Reducer that can save the typed message to the Store.
4. Add another life-cycle (componentWillReceiveProps) to PreviewFrame.jsx, to detect the change of typed message and send it to an evaluation function.
5. Save the evaluated result in the Store same as now (the state called consoleEvents).

Steps related to ConsoleOutput component

1. Load the consoleEvents state.
2. Render it to a new line of the ConsoleInput.



It would be ideal if the console has history which enables users to trace record. Therefore, I will also append history function to console during the second stage. Obviously it would be accessible if we persist the history in database. The React app would make a request to the backend to retrieve the history once started. But in my opinion, another sufficient way could be using the localStorage of browser. What we need to do is configuring the code editor properly and then we can get the cached data whenever keyboard event is triggered.

Supposing the first two stages are accomplished, the new interactive console can function now. Then l will spend some time refactoring over the code to satisfy the check, so it can be applied to the code base. After that, we come to the third stage and l will set about improving the console to make it more user-friendly. And these steps will be included in the third stage:

1. Autocompletion.
2. Separation of the console and error information.

Development Process

Preparation (Prior - April 24)

During this period, l will keep familiarizing myself with the code base by fixing issues.

Community Bonding (April 24 - May 14)

Over this period, l am going to discuss with community regarding my proposal and investigate how the existing tools work.

Discussion (April 24 - April 30)

- Discuss with my mentor on the feasibility of my proposal and consider if there are some other potential enhancements to my plan.

Investigation (May 1 - May 14)

- Analyze some existing tools and think over how to integrate these features into code base better.

Coding Phase 1 (May 15 - June 11)

My goal over this period of time is to realize the improvement of the current console.

Week 1 (May 15 - May 21)

- Solve the existing issues here #468.
- Implement the folding and unfolding of console logged object/array.

Week 2 (May 22 - May 28)

- Display the duplicated console logs once with a number (One possible solution is to add a new state to indicate the number of repetitions).

Week 3 (May 29 - June 4)

- Support syntax highlighting.

Week 4 (June 5 - June 11)

- Support more console methods (e.g. console.table, console.time).

Coding Phase 2 (June 12 - July 9)

Now, the major task is to support a prototype of interactive console.

Week 5 (June 12 - June 18)

- Build a new code editor(CodeMirror) into ConsoleInput component which allow users to type in.
- Design the style of the new console.

Week 6 (June 19 - June 25)

- Create an Action and Reducer to save the typed message to Store.
- Establish another postMessage communication to connect evaluated result with the state consoleEvents (I believe it will function more effectively by distinguishing error events from console events).

Week 7 (June 26 - July 2)

- Realize the evaluation function which can evaluate the next statement in the context created by the last one.
- Create a ConsoleOutput component to display the console message.

Week 8 (July 3 - July 9)

- Save the typed message in localStorage and bind it with the CodeMirror.
- Enhancement: the current postMessage communication is just one-way, it will make sense by introducing some external library like postmate or jschannel to support promise apis.

Coding Phase 3 (July 10 - August 6)

In the phase, my goal is to make the interactive console more user-friendly.

Week 9 (July 10 - July 16)

- Improve the code quality and refactor code structure, to make it more robust and readable.
- Test the new console thoroughly and fix bugs of it.
- Discuss with mentor and apply the prototype to the code base if possible.

Week 10 (July 17 - July 23)

- Support autocompletion of the new console (CodeMirror supports basic javascript autocompletion itself).

Week 11 (July 24 - July 30)

- Extend the test cases if necessary.
- Separate the interactive console from error messages (TBD: I believe this feature will meet both experienced coders and newbies' requirements).

Week 12 (July 31 - August 6)

- Clean up project.
- Serving as buffer time (The second coding phase may be more challenging than expected and will take more time than allocated).

More about you

My name is Liang Tang, a third year student in Shanghai Jiao Tong University. My major is Information Security. l am a programmer who devotes to simplification of the process to get things done, providing convenience to users. l am also a big enthusiast of the latest technologies in programming languages. l like to spend most of my current time exploring more of JavaScript along with its recent technologies. Also l am very interested in data visualization and have some experience on it. My real world experience in web development, through several own projects and internships, have given me a great capability in projects written in JavaScript.

l first came across p5.js a while ago and started to play around by doing some small visual work with it. It really fascinated me but soon l found it a bit tedious to refresh the page when changes were made. Then I realized that p5.js web editor was a powerful tool without worries of setup, so it would enable us to focus on creating. That's why l am interested in it.

Why me?

GSoC provides me a great chance to contribute to open source projects, so l want to seize this opportunity to improve myself. l am now very proficient in JavaScript and some related web technologies. l have earned the full-stack certificate on FreeCodeCamp and worked as a full-stack intern in a startup company. l have been programming in JavaScript for almost two years and have built many web-based projects. Also l am familiar with MEAN stack which the p5.js web editor uses.

Some of my projects

- Tango - A high-precision rights control system. Its goal is to be used as an internal document management system. It was used in a incubation platform called neoBay. Specific technologies: Vue.js, Golang
- Yichen - an enterprise website. It was my first website in which I handled a large amount of user data. During the development, I used the Object-oriented programming method and realized that it was important to have a good code structure. Specific technologies: React, Java

More work can be seen here: cv

In the past few weeks, I have dived into the codebase and made some contributions to the p5.js web editor. Although this is my first time contributing to open source, I begin to appreciate the fun of contributing to open source. By now, I have got familiar with how it works and I am hoping to push it forward by lowering the threshold for newbies and making it more user-friendly to novices.

Contributions

Issues fixed

- Cross-platform failure building.
  https://github.com/processing/p5.js-web-editor/issues/513
- Implementation of auto-save when downloading.
  https://github.com/processing/p5.js-web-editor/issues/517
- Console errors causing the editor to crash.
  https://github.com/processing/p5.js-web-editor/issues/527
- Error route `/<username>/sketches/<random_string>`.
  https://github.com/processing/p5.js-web-editor/issues/520
- Unexpected reversion to previous version of file.
  https://github.com/processing/p5.js-web-editor/issues/470
  https://github.com/processing/p5.js-web-editor/issues/419
  https://github.com/processing/p5.js-web-editor/issues/437
- Changing sketch view page title.
  https://github.com/processing/p5.js-web-editor/issues/554

- Bug report - failure in closing the ErrorModal.
  https://github.com/processing/p5.js-web-editor/issues/567
- Implementation of more user-friendly saving.
  https://github.com/processing/p5.js-web-editor/issues/560
- Bug report - error keyboard triggering.
  https://github.com/processing/p5.js-web-editor/issues/587
- Missing console errors.
  https://github.com/processing/p5.js-web-editor/issues/468
- Detailed git workflow
  https://github.com/processing/p5.js-web-editor/issues/602

Pull requests

View merged PRs on Github.

Currently, the p5.js web editor is under active development and will be released to the public soon. That is unbelievably cool for me to work with such a fantastic tool which is accessible to lots of users, and I really treasure this chance provided by GSoC. If I get the opportunity to work with p5.js web editor this summer, I will exert all my effort to improve it and also I am looking forward to working as a long-term maintainer for it even after the summer!