

A Web-based Environment for Developing and Utilizing Teaching Languages for Novice Computer Science Students

Benjamin J. Kruger & Richard Matzen, Northeastern State University

C Spot Run

Student Development Environment

- Runs completely in the browser.
- No compile server needed.
- Can be hosted from static server.
- Built with the full-feature Ace editor
- Custom syntax highlighting
- Multiple cursors
- RegExp find/replace
- Code Folding
- Et cetera

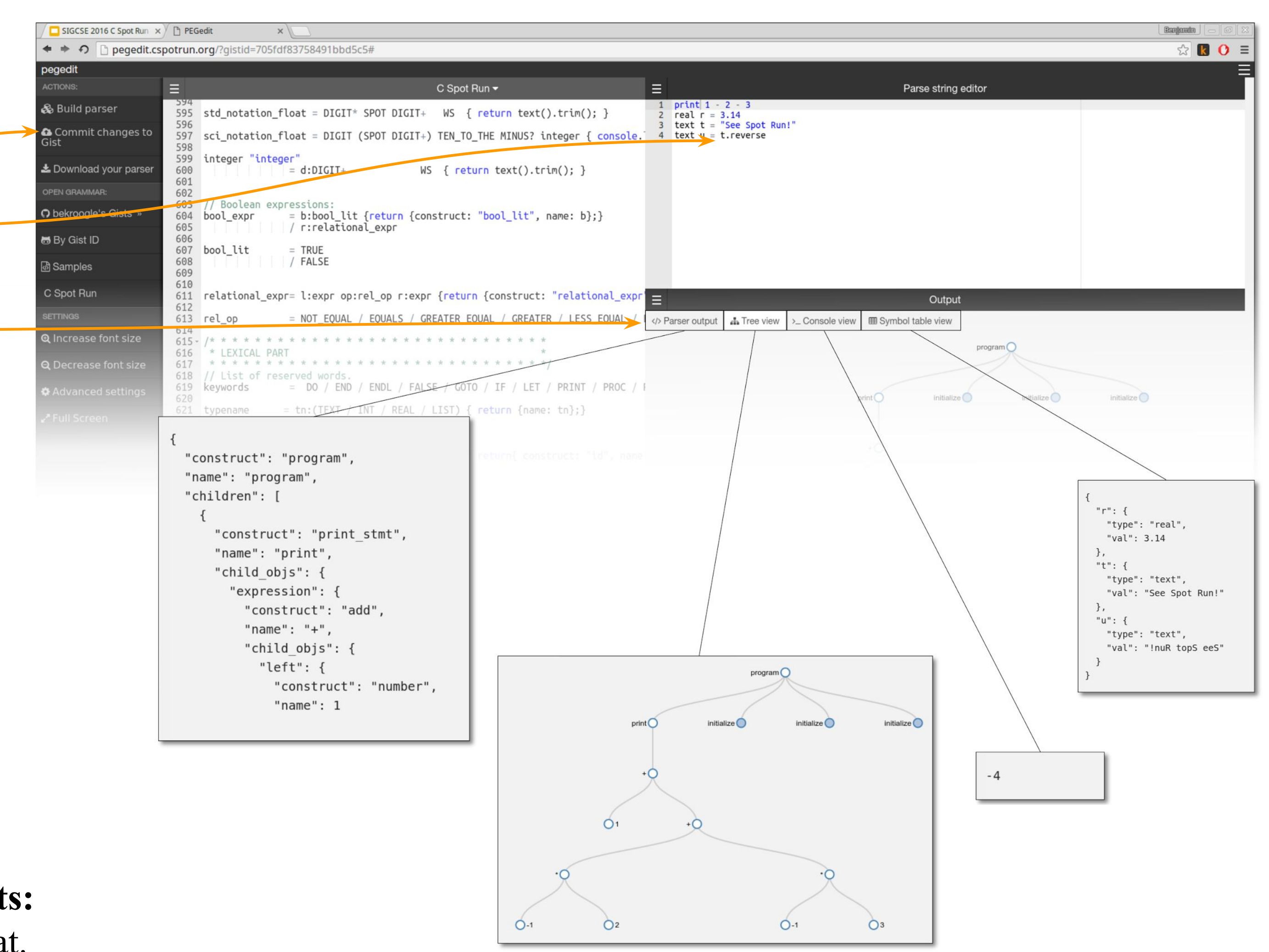


Language Development Toolkit

This site was built as an in-house tool to facilitate the development of the C Spot Run language and aid in debugging. The developer authors a PEG.js grammar (JavaScript implementation of Bryan Ford's Parsing Expression Grammar formalism) on the left side. Some test code goes on the right. Next, simply build and run the parser to see the console output, a visualized syntax tree, and an audit of the symbol table's final state.

Features:

- Integrated with GitHub's Gist service.
- In-place testing.
- Multiple views for debugging.



Several ways to view results:

- Syntax tree in JSON format.
- Graphical tree using the D3.js visualization library.
 - With multiple modes (e.g. zoom, collapse).
- Console view for text output.
- Symbol table shows final values for variables.

Syntax

- Intuitive
- Based on empirical research by Stefik and Siebert¹.
- Designed for teaching only (not for industry).
- Limits complexity.
- Overcomes objections to overloading = operator.

```
1 # Averages the number supplied by user interactively.
2
3 int iters = prompt "How many numbers?"
4 int nextNumber
5 int sum = 0
6
7 while i = 1 to iters
8   let nextNumber = prompt "Enter a number: "
9   let sum = sum + nextNumber
10 repeat
11
12 print "The average is " + (sum / iters)
13
```



<http://www.cspotrun.org>



<http://pegedit.cspotrun.org>

Roadmap

Student Development Environment

- Git-based data collecting to mine for common errors and issues with programming language syntax.
- Inform syntax choices in language design.
- Alert teachers when a change in pedagogy is needed.
- Assignment submission and file management.

Language Development Toolkit

- Add support for parser tracing (just added to PEG.js 0.9.0).
- Use web workers to allow infinite loops/recursion to be handled gracefully.
- Create a basic starter template for authoring a language for the system.
- Build automated language/library registry (à la NPM, Ruby GEM).
- Improve UI.
- Remove the need for the designer to use specific design patterns to access all features.

C Spot Run Language

- Add support for libraries (math, graphics web audio).
- Improve cryptic error messages.
- Give parser better awareness of the environment while generating tree.
- Implement proper type checking.

... and try the language in an actual classroom setting!

References & Acknowledgments

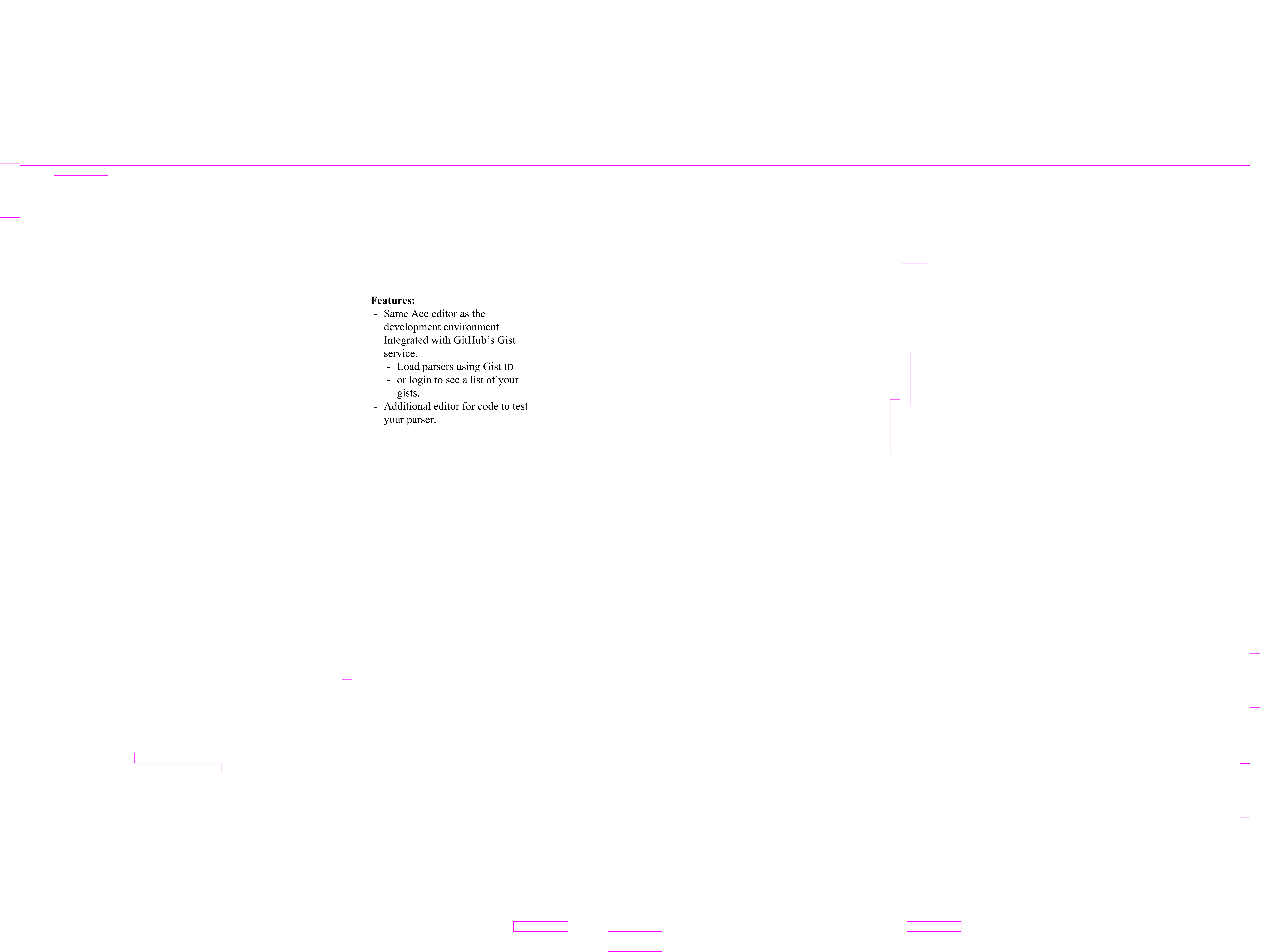


The following authors and their works informed and influenced this project:

1. Andreas Stefik and Susanna Siebert. 2013. An Empirical Investigation into Programming Language Syntax. Trans. Comput. Educ. 13, 4, Article 19 (November 2013).
2. Bryan Ford. 2004. Parsing expression grammars: a recognition-based syntactic foundation. In Proceedings of the 31st ACM SIGPLAN-SIGACT symposium on Principles of programming languages (POPL '04). ACM, New York, NY, USA, 111-122.

These open source tools made this project possible:

- Ace (Ajax.org Cloud9 Editor)
 - <https://ace.c9.io>
- PEG.js parser generator for JavaScript
 - <http://pegjs.org/>
- D3.js Data-Driven Documents
 - <https://d3js.org/>



Features:

- Same Ace editor as the development environment
- Integrated with GitHub's Gist service.
 - Load parsers using Gist ID
 - or login to see a list of your gists.
- Additional editor for code to test your parser.