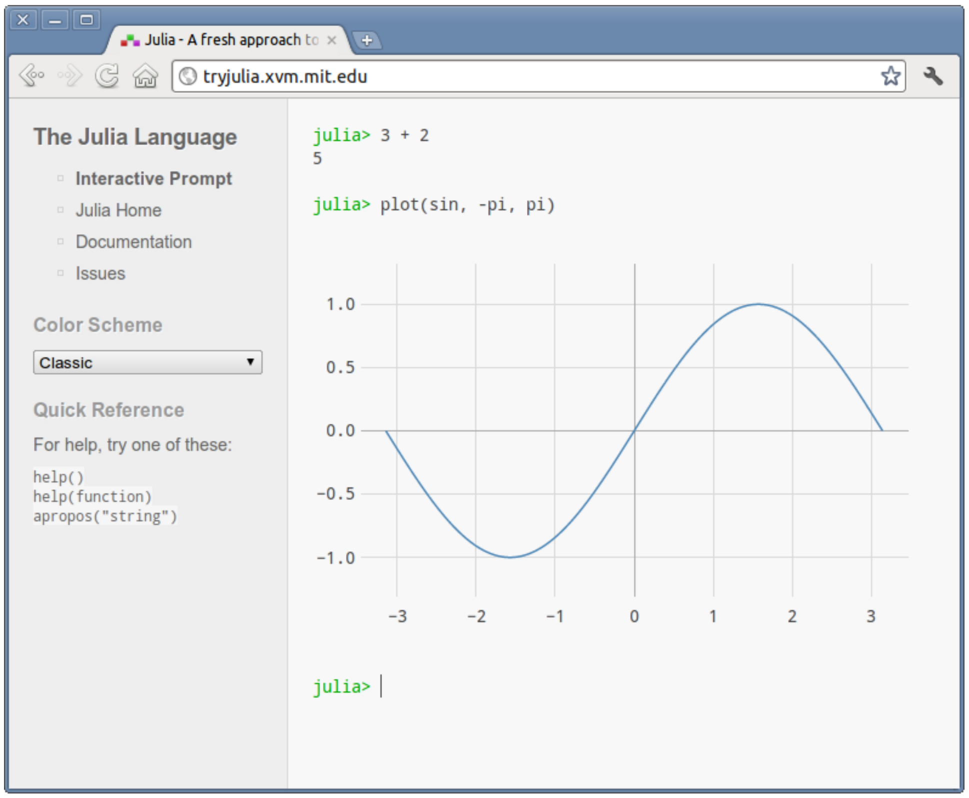
A Graphical Front End for the Julia Programming Language

Stephan Boyer

December 2011



Color Scheme: Classic

For help, try one of these:

```
help()
help(function)
apropos("string")
```

```
julia> 3 + 2
5

julia> plot(sin, -pi, pi)

julia> 
```

1 Introduction

In order to compete in the landscape of numerical and parallel computing, the Julia Programming Language[1] should be easy to use and include plotting functionality. Mathematicians and scientists who are not familiar with SSH and related utilities will find it difficult to command an array of nodes in a computing cluster in the traditional manner.

This paper introduces a web-based front end for Julia, which includes a read–evaluate–print loop (REPL) and basic plotting functionality. There are a number of reasons to use a web-based front end:

1. Users do not need to be familiar with UNIX utilities and shell scripting to use Julia on a computing cluster.

2. Users do not need to install special software to use Julia—a web browser is all that is needed.

3. Modern web technologies and standards, such as HTML5, Scalable Vector Graphics (SVG), and AJAX provide a rich platform for generating interactive plots and graphical user interfaces.

4. A front end based on web technologies does not require an Internet connection—the system can always be run locally.

5. A web interface allows for a new social component to numerical and parallel computing. For example, several users could share an interactive Julia session, or collaboratively edit a Julia program.

2 The Implementation

The current implementation consists of the *lighttpd* web server, a custom SCGI-based session manager, and a backend written in Julia. Together, these components provide a web-based read–evaluate–print loop (REPL), as shown on the title page. This figure also demonstrates the system's plotting capability. The following features are supported:

1. Can plot functions, 1-D arrays, or arrays of points for any numeric type.

[1]http://julialang.org/

2. Can automatically determine window dimensions, or you can specify that manually.
3. When domain and range are similar, window is chosen to preserve aspect ratio.
4. Picks tick marks intelligently.
5. When plotting a function with a limited domain (e.g. *sqrt*), undefined output is ignored.
6. Full error checking.
7. Extensible interface allows us to add more types of plots easily.
8. Supported browsers include Firefox, Chrome, Safari, and IE9.
9. Plots are ”print safe” and ”resolution independent” (thanks to SVG graphics).
10. More aesthetic than MATLAB and Mathematica plots (full anti-aliasing of text and lines).

3 Plotting Examples

1. plot(sin, -pi, pi)
2. plot(cos(-pi:0.01:pi), sin(-pi:0.01:pi))
3. plot([0.0, 0.1, 0.4, 0.3, 0.4])

4 Plotting Limitations

The plotting functionality is currently limited in the following ways:

1. No zooming or interacting with plots
2. No way to export plots
3. Only one type of plot (line graph)

4. Graphics only available in web interface
5. Can't plot multiple curves on same image
6. No titles/labels

Each of these limitations will be overcome in future versions.

5 Acknowledgement

I want to extend a special thanks to Professor Alan Edelman, Jeff Bezanson, Viral Shah, Stefan Karpinski , and everyone involved with the Julia project. Without them and their hard work, proprietary software might continue to dominate the field of numerical computing.