

# Web based teaching through real-time collaboration with ghcLiVE

Shae Erisson

Department of Computer Science and Information Systems  
University of North Alabama

April 9, 2013



# Google Summer of Code

- offers student developers stipends to write code for various open source projects.
- over 4,500 students with over 300 open source projects
- started in 2005, now in its eighth year.



# Haskell



Haskell is a general purpose purely functional programming language with non-strict semantics and strong static typing.



# Factorial in Haskell

```
factorial n = product [1..n]
```



# QuickSort in Haskell

```
quicksort :: Ord a => [a] -> [a]
quicksort []      = []
quicksort (p:xs) =
    (quicksort lesser) ++ [p] ++ (quicksort greater)
    where
        lesser = filter (< p) xs
        greater = filter (>= p) xs
```

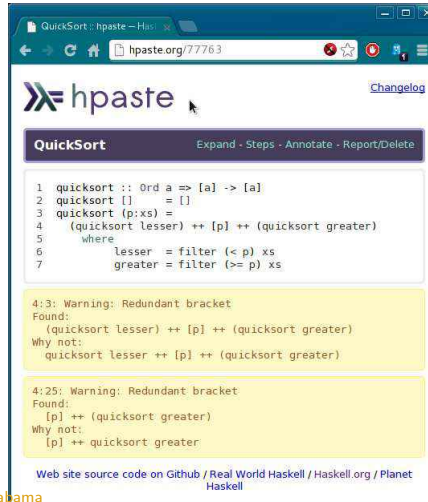


# Internet Relay Chat

```
[ "Haskell - the language of ICFP winners 3 years running", "Dist-tern
15:25 dons> ?wiki Monad
15:25 lambdabot> http://www.haskell.org/haskellwiki/Monad
15:25 dons> gives a few examples
15:26 sorear> So (with overloaded null operator) we can have e.g
        echo = putStrLn getStrLn
15:26 sorear> that desugars as echo = (ap putStrLn getStrLn)
15:26 sorear> the (I wish) bind instance triggers
15:26 sorear> echo = getStrLn >= putStrLn
15:27 sorear> If it can be made to work... all the niceties of a
        full-blooded strict language
15:28 sorear> getStrLn = case getChar of { '\n' -> [] ; x -> x ;
        getStrLn }
15:28 sorear> hmm, case is a problem
15:28 wy> I wonder why if (l<0) then a1+a2 else 2 work if a1 and a2
        are Natural type. There is a fromInteger in the instance.
        But why does if...then...else call that function?
15:29 sorear> sure... (l < 0) == (fromInteger l < fromInteger 0)
15:29 sorear> (l < 0) == (Num a, Ord a) => a
15:29 sorear> oops
15:30 wy> sorear: but that has nothing to do with the return type
15:30 sorear> (Num a, Ord a) => Bool
[15:30] [dons(+6ei)] [7:#haskell(+ns)] [Act: 2,3,10,14,16,20]
[#haskell]
```



# hpaste



The screenshot shows a web browser window with the address bar at `hpaste.org/77763`. The page title is "QuickSort : hpaste - Has...". The main content area displays Haskell code for a QuickSort function. Below the code, there are two yellow warning boxes from the Glasgow Haskell Compiler (GHC). The first warning is at line 4:3, indicating a "Redundant bracket" in the expression `(quicksort lesser) ++ [p] ++ (quicksort greater)`. The second warning is at line 4:25, indicating a "Redundant bracket" in the expression `[p] ++ (quicksort greater)`. The page also includes a "Changelog" link and a navigation bar with "Expand", "Steps", "Annotate", and "Report/Delete" options.

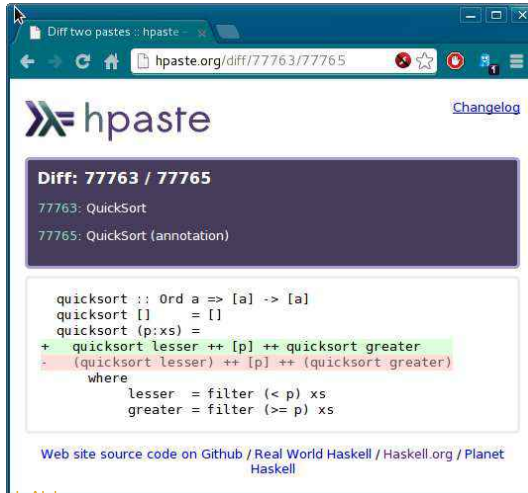
```
1 quicksort :: Ord a => [a] -> [a]
2 quicksort [] = []
3 quicksort (p:xs) =
4   (quicksort lesser) ++ [p] ++ (quicksort greater)
5   where
6     lesser = filter (< p) xs
7     greater = filter (>= p) xs
```

4:3: Warning: Redundant bracket  
Found:  
(quicksort lesser) ++ [p] ++ (quicksort greater)  
Why not:  
quicksort lesser ++ [p] ++ (quicksort greater)

4:25: Warning: Redundant bracket  
Found:  
[p] ++ (quicksort greater)  
Why not:  
[p] ++ quicksort greater

Web site source code on [Github](#) / [Real World Haskell](#) / [Haskell.org](#) / [Planet Haskell](#)

# hpaste



Diff two pastes :: hpaste - x

hpaste.org/diff/77763/77765

**Diff: 77763 / 77765**

77763: QuickSort

77765: QuickSort (annotation)

```
quicksort :: Ord a => [a] -> [a]
quicksort [] = []
quicksort (p:xs) =
+ quicksort lesser ++ [p] ++ quicksort greater
- (quicksort lesser) ++ [p] ++ (quicksort greater)
  where
    lesser = filter (< p) xs
    greater = filter (>= p) xs
```

Web site source code on Github / Real World Haskell / Haskell.org / Planet Haskell



# ghci

```
ghci Quicksort.hs
[Fri 12/11/16 11:23 EST][pts/0][x86_64/linux-gnu/3.2.0-3-amd64][4.3.17]
<shae@chaoyote:~/clients/gsoc>
zsh 5093 (darcs)-[gsoc]-% ghci Quicksort.hs
GHCi, version 7.4.1: http://www.haskell.org/ghc/ :? for help
Loading package ghc-prim ... linking ... done.
Loading package integer-gmp ... linking ... done.
Loading package base ... linking ... done.
[1 of 1] Compiling Main                ( Quicksort.hs, interpreted )
Ok, modules loaded: Main.
*Main> quicksort [7,0,6,5,1,9,8,2]
[0,1,2,5,6,7,8,9]
*Main> █
```





# How to debug source you can't see?

<novice>I still get those same errors I showed you before.

<expert>...but the errors you posted *\*still\** don't match with the file you posted.

<expert>Are you definitely hitting "save" in your text editor?

- It turned out that the editor used by the novice was not saving at all.



# Multi-user editor and compiler in a web browser

The screenshot displays the ghcLiVE web interface in a browser. The top navigation bar includes links for 'reader', 'hdiff', 'add to reader', 'programming', 'gmail', 'Comics', 'Manga', 'λx.comonad admin', 'Todo', and 'monoidal knuth ben'. The main content area is titled 'Editor' and contains a Haskell code snippet for generating a Hilbert curve. Below the editor is a 'Load shared document' button. The bottom section shows the REPL output, displaying a sequence of images representing the stages of a Hilbert curve's construction. A 'Refresh output' button is located below the output area.

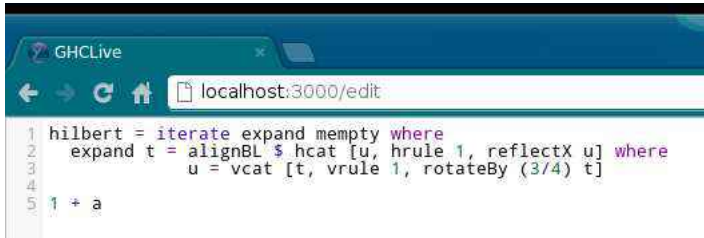
```
1 import Diagrams.Prelude
2 import Prelude
3 import Network.Web.GHCLive.Display
4
5
6 hilbert = iterate expand mempty where
7   expand t = alignBL $ hcat [u, hrule 1, reflectX u] where
8     u = vcat [t, vrule 1, rotateBy (3/4) t]
9
10 ex = pad 1.1 . centerXY . lw 0.05 $ hilbert!15
```

Load shared document

hint> map (pad 1.1 . centerXY . lw 0.05) \$ take 7 hilbert

Refresh output

## Source widget



The screenshot shows a web browser window titled 'GHCLive'. The address bar displays 'localhost:3000/edit'. The main content area shows Haskell code being edited in a text editor with line numbers 1 through 5 on the left. The code is as follows:

```
1 hilbert = iterate expand mempty where
2   expand t = alignBL $ hcat [u, hrule 1, reflectX u] where
3     u = vcat [t, vrule 1, rotateBy (3/4) t]
4
5 1 ÷ a
```

Haskell source code is edited in the top pane.

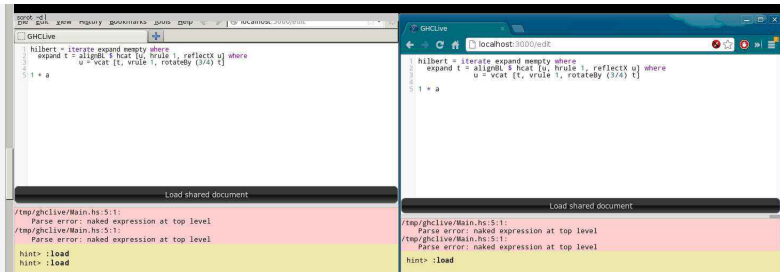
# Multi-user editor and compiler in a web browser

The screenshot displays the ghcLiVE web interface in a browser. The browser's address bar shows the URL `localhost:3000/edit`. The interface is divided into several sections:

- Editor:** A text area containing Haskell code for generating a Hilbert curve. The code is as follows:

```
1 import Diagrams.Prelude
2 import Prelude
3 import Network.Web.GHCLive.Display
4
5
6 hilbert = iterate expand mempty where
7   expand t = alignBL $ hcat [u, hrule 1, reflectX u] where
8     u = vcat [t, vrule 1, rotateBy (3/4) t]
9
10 ex = pad 1.1 . centerXY . lw 0.05 $ hilbert!15
```
- Load shared document:** A dark button located below the editor.
- REPL Output:** A large yellow area showing the output of the code. It includes a preview of the Hilbert curve and the command `hint> map (pad 1.1 . centerXY . lw 0.05) $ take 7 hilbert`. Below the command, seven small square images show the progression of the Hilbert curve from a single line to a complex fractal.
- Refresh output:** A dark button located below the REPL output.
- Input Field:** A text input field at the bottom containing the command `map (pad 1.1 . centerXY . lw 0.05) $ take 7 hilbert`.
- Evaluate:** A dark button to the right of the input field.

# synchronized source widget



As the source code changes in the shared editor,  
every attached client is updated in real-time.

# Multi-user editor and compiler in a web browser

The screenshot displays the ghcLiVE web interface in a browser. The browser's address bar shows the URL `localhost:3000/edit`. The interface is divided into two main sections: an "Editor" and a REPL output area.

The "Editor" section contains the following Haskell code:

```
1 import Diagrams.Prelude
2 import Prelude
3 import Network.Web.GHCLive.Display
4
5
6 hilbert = iterate expand mempty where
7   expand t = alignBL $ hcat [u, hrule 1, reflectX u] where
8     u = vcat [t, vrule 1, rotateBy (3/4) t]
9
10 ex = pad 1.1 . centerXY . lw 0.05 $ hilbert!15
```

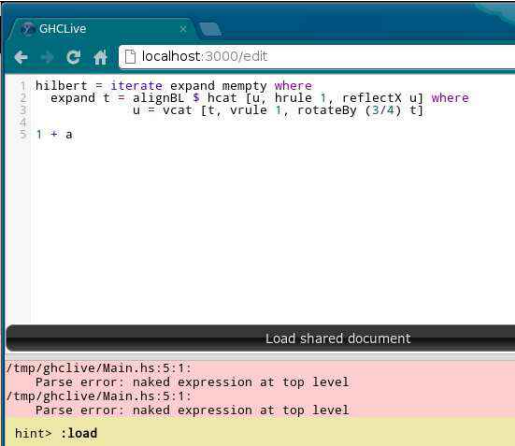
Below the editor is a dark bar with the text "Load shared document".

The REPL output area shows the command `hint> map (pad 1.1 . centerXY . lw 0.05) $ take 7 hilbert` and its corresponding visual output: a sequence of seven diagrams showing the iterative construction of the Hilbert curve. The first diagram is a small square, and subsequent diagrams show increasing levels of self-similarity and complexity.

Below the output is a dark bar with the text "Refresh output".



## compiler error



The screenshot shows the ghcLiVE web interface. The top part is a code editor with the following Haskell code:

```
1 hilbert = iterate expand mempty where
2   expand t = alignBl $ hcat [u, hrule 1, reflectX u] where
3     u = vcat [t, vrule 1, rotateBy (3/4) t]
4
5 1 + a
```

Below the editor is a button labeled "Load shared document". At the bottom is a REPL window showing two parse errors:

```
/tmp/ghclive/Main.hs:5:1:
  Parse error: naked expression at top level
/tmp/ghclive/Main.hs:5:1:
  Parse error: naked expression at top level
hint> :load
```

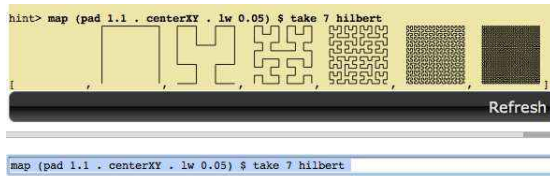
# Multi-user editor and compiler in a web browser

The screenshot displays the ghcLiVE web interface in a browser. The browser's address bar shows the URL `localhost:3000/edit`. The interface is divided into several sections:

- Editor:** A text area containing Haskell code for generating a Hilbert curve. The code is as follows:

```
1 import Diagrams.Prelude
2 import Prelude
3 import Network.Web.GHCLive.Display
4
5
6 hilbert = iterate expand mempty where
7   expand t = alignBL $ hcat [u, hrule 1, reflectX u] where
8     u = vcat [t, vrule 1, rotateBy (3/4) t]
9
10 ex = pad 1.1 . centerXY . lw 0.05 $ hilbert!15
```
- Load shared document:** A dark button located below the editor.
- REPL Output:** A large yellow area showing the output of the code. It includes a preview of the Hilbert curve and the command `hint> map (pad 1.1 . centerXY . lw 0.05) $ take 7 hilbert`. Below the command, seven small images show the progression of the Hilbert curve from a single square to a complex fractal.
- Refresh output:** A dark button located below the REPL output.
- Input Field:** A text input field at the bottom containing the command `map (pad 1.1 . centerXY . lw 0.05) $ take 7 hilbert`.
- Evaluate:** A dark button to the right of the input field.

## ghcLiVE REPL input widget



The input pane allows you to execute the loaded source code.

# Multi-user editor and compiler in a web browser

The screenshot displays the ghcLiVE web interface in a browser. The browser's address bar shows the URL `localhost:3000/edit`. The interface is divided into two main sections: an editor and a REPL.

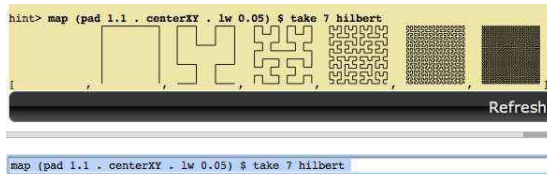
**Editor:** The top section is titled "Editor" and contains a Haskell code snippet:

```
1 import Diagrams.Prelude
2 import Prelude
3 import Network.Web.GHCLive.Display
4
5
6 hilbert = iterate expand mempty where
7   expand t = alignBL $ hcat [u, hrule 1, reflectX u] where
8     u = vcat [t, vrule 1, rotateBy (3/4) t]
9
10 ex = pad 1.1 . centerXY . lw 0.05 $ hilbert!15
```

Below the editor is a dark button labeled "Load shared document".

**REPL:** The bottom section is a REPL window. It shows the command `hint> map (pad 1.1 . centerXY . lw 0.05) $ take 7 hilbert` and its output, which is a sequence of seven fractal images (Hilbert curves) of increasing complexity. Below the output is a dark button labeled "Refresh output".

## ghcLiVE output widget



The output pane supports both text and Simple Vector Graphics output.

# Dependencies

- Glasgow Haskell Compiler has a complicated API (ghc-api)
- Hint is a user-friendly wrapping of the ghc-api
- Yesod web framework for Haskell (similar to Ruby on Rails)
- JavaScript source code editor from [codemirror.net](http://codemirror.net)
- WebSockets don't work with older browsers



# Architecture

- Three threads
  - webserver
  - compiler
  - editor synchronization
- just under 300 lines to sew together the compiler, webserver and synchronizer
- just under 200 lines for document state and synchronization



## Design choices

- Desktop only
- No security
- No accounts
- No state





## Current state

- Working prototype
- Several known bugs
- Interested academic users



## Future work

- Fix bugs
- Work with interested educational users
- Generalize to languages other than Haskell?
- Voice chat functionality with Google Talk?



**Thank you!**

- <http://github.com/shapr/ghclive>
- <http://ghclive.wordpress.com>

Shae Erisson

Dept. of Computer Science & Information Systems

University of North Alabama

<http://www.ScannedInAvian.com/>