User interfaces

Running Haskell in the browser

This lecture

- Preparation for lab 4
- Simple GUIs in Haskell

- **Main point:** How to “think functionally” when designing interactive programs
  - Separate logic (pure functions) from interaction (IO)

Separating logic from interaction

Example: printSudoku

- **Bad:**

```
printSudoku (Sudoku [])      = return ()
printSudoku (Sudoku (r:rs)) = do  printRow r
                                  printSudoku (Sudoku rs)

printRow :: [Maybe Int] -> IO ()
printRow []       = putStr ”\n”
printRow (c:cs) = do  putStr ...
                      printRow cs
```

- **Good:**

```
printSudoku sud = putStrLn $ sudToString sud

sudToString :: Sudoku -> String
sudToString (Sudoku rs) = unlines … map ...
```

User interfaces

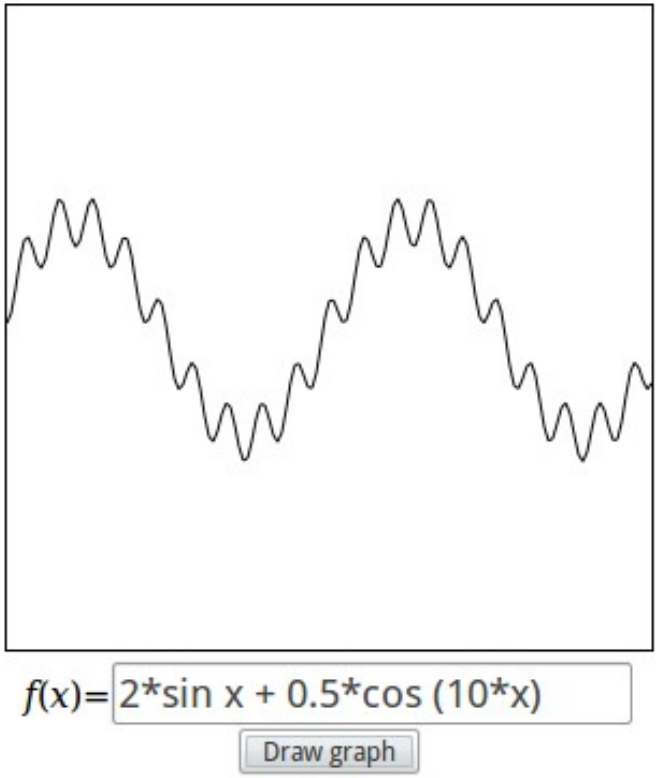
- Programs in the terminal don't scale well
- How to make windows, buttons, graphics, etc. ?
- Several GUI libraries in Haskell:
  - wxHaskell, Gtk2Hs, etc.
- Status 2013: Hard to install reliably across platforms
  http://www.haskell.org/pipermail/haskell-cafe/2013-September/110440.html
- But there is active development. Maybe things are better today.

Alternative: Haskell in the web browser

- Haste (the choice in this course)
  - Compile Haskell to Javascript
  - Run Haskell *directly in the browser*
  - Developed by former Chalmers/GU student Anton Ekblad
- Other Haskell-to-Javascript options
  - Fay (only supports a limited subset of Haskell)
  - GHCJS
  - PureScript
- Threepenny-gui
  - A small graphical web-server that interacts with Haskell programs

Lab 4

- Part I: Pretty printer and parser for mathematical expressions
- Part II: Draw function graphs in the browser
  - Using Haste



Haste

See examples on the Haste page