# jQuery Programming with Haskell

## Dutch FP Day 2011

Jurriën Stutterheim

January 7, 2011

# Overview

jQuery Programming with Haskell

- Introduction
- Programming jQuery
- Demo Time!
- Current Challenges
- Concluding
- Questions?

# 1. jQuery Programming with Haskell

# 1.1 Introduction

# The Idea



- JavaScript is great for client-side web programming, but it lacks type safety
- Idea: write Haskell, compile to JavaScript
- Laziness $\wedge$ import/export $\wedge$ compiler $\Rightarrow$ Haskell web scripting
- Part of the Utrecht Haskell Compiler (UHC)
  - By Atze Dijkstra
- Type safe JavaScript coding!

# Related Work



Cross-compiling to JavaScript is not new:

- GWT (Java)
- CoffeeScript (New language)
- Pyjamas (Python)
- RubyJS (Ruby)
- Java2Script (Java)

# Great! But...



- Compiling Haskell to JavaScript is great, but reinventing the wheel is not
- Dozens of useful JavaScript libraries already exist
- It is essential that these can be reused in the Haskell code

# Planning



Today:

- Program jQuery using Haskell and use it for DOM manipulation
- See Haskell-programmed jQuery in action
- Discuss some of the current challenges in implementing support for jQuery and other libraries

Another time:

- Challenges in cross-compiling a lazy language to a non-lazy language
- In-depth discussion of how UHC's JavaScript backend works

# 1.2 Programming jQuery

# Hello, jQuery! – JavaScript



Short version:

$$showNeat = \$(\text{"}p.neat\text{"}).addClass(\text{"}ohmy\text{"}).show(\text{"}slow\text{"});$$

Expanded version:

$$\begin{aligned}
&showNeat = function(){\\
&\quad j = \$(\text{"}p.neat\text{"});\\
&\quad j.addClass(\text{"}ohmy\text{"});\\
&\quad j.show(\text{"}slow\text{"});\\
&\}
\end{aligned}$$

# Hello, jQuery! – Haskell



Very much like the expanded JavaScript version:

```
foreign export jscript "showNeat" showNeat :: IO ()
showNeat :: IO ()
showNeat =
  do j ← select  $ s2js "p.neat"
     addClass j  $ s2js "ohmy"
     jqshow   j (Just slow) Nothing Nothing
```

# Bootstrapping



- $main :: IO()$ is executed the moment the JavaScript is loaded
- However, jQuery code is bootstrapped using $\$(document).ready();$
- Solution: create a separate $jQueryMain :: IO()$ function and add the $\$(document).ready(jQueryMain);$ to the document manually
- $main :: IO()$ should $return\,()$ to prevent interfering with jQuery
  - Or should be omitted, if UHC would allow that...

# 1.3 Demo Time!

# 1.4 Current Challenges

# Exposing jQuery Functions to Haskell



Using $Maybe$s for dealing with optional arguments is clumsy and requires multiple FFI definitions for one function and a lot of pattern match cases.

```
foreign import jscript "%1.show()"   jqshow0  :: JQuery -> IO ()
foreign import jscript "%1.show(%*)" jqshow1  :: JQuery -> Int -> IO ()
foreign import jscript "%1.show(%*)" jqshow2  :: JQuery -> Int -> JSString -> IO ()
foreign import jscript "%1.show(%*)" jqshow2' :: JQuery -> Int -> IO () -> IO ()
foreign import jscript "%1.show(%*)" jqshow3  :: JQuery -> Int -> JSString -> IO () -> IO ()

jqshow :: JQuery -> Maybe Int -> Maybe String -> Maybe (IO ()) -> IO ()
jqshow j Nothing  Nothing  Nothing  = jqshow0  j
jqshow j (Just n) Nothing  Nothing  = jqshow1  j n
jqshow j (Just n) (Just e) Nothing  = jqshow2  j n (s2js e)
jqshow j (Just n) Nothing  (Just c) = jqshow2' j n c
jqshow j (Just n) (Just e) (Just c) = jqshow3  j n (s2js e) c
```

# Lots of IO...



There seems to be no escaping the $IO$ monad...

$$
\begin{aligned}
&\mathit{select} && :: \mathit{JSString} \to \mathit{IO}\ \mathit{JQuery} \\
&\mathit{getHTML} && :: \mathit{JQuery} \to \mathit{IO}\ \mathit{JSString} \\
&\mathit{setHTML} && :: \mathit{JQuery} \to \mathit{JSString} \to \mathit{IO}\ () \\
&\mathit{hide} && :: \mathit{JQuery} \to \mathit{IO}\ () \\
&\mathit{addClass} && :: \mathit{JQuery} \to \mathit{JSString} \to \mathit{IO}\ () \\
&\mathit{jqshow0} && :: \mathit{JQuery} \to \mathit{IO}\ () \\
&\mathit{jqshow1} && :: \mathit{JQuery} \to \mathit{Int} \to \mathit{IO}\ () \\
&\mathit{jqshow2} && :: \mathit{JQuery} \to \mathit{Int} \to \mathit{JSString} \to \mathit{IO}\ () \\
&\mathit{jqshow2'} && :: \mathit{JQuery} \to \mathit{Int} \to \mathit{IO}\ () \to \mathit{IO}\ () \\
&\mathit{jqshow3} && :: \mathit{JQuery} \to \mathit{Int} \to \mathit{JSString} \to \mathit{IO}\ () \to \mathit{IO}\ ()
\end{aligned}
$$

# Current Challenges



- Can the big block of imports, pattern matches and $Maybe$s be replaced by something more elegant?
- Can the use of $IO$ be reduced? (Would STM provide some sort of compromise?)

# 1.5 Concluding

# Concluding



- Haskell can be compiled to JavaScript
  - JavaScript? HaskellScript!
- Existing libraries can be used from the Haskell code
  - E.g. use jQuery for DOM manipulation
- There are still some problems left to solve
  - $Maybe$
  - $IO$

# Start Playing!



- UHC on GitHub:
  - https://github.com/uhc/uhc
- jQuery support on GitHub:
  - https://github.com/norm2782/haskell-jscript-libraries
- UHC blog
  - http://utrechthaskellcompiler.wordpress.com/

# 1.6 Questions?