Melchior

Web Programming in Haskell

Kieran Gorman, Timothy Jones and Lindsay Groves

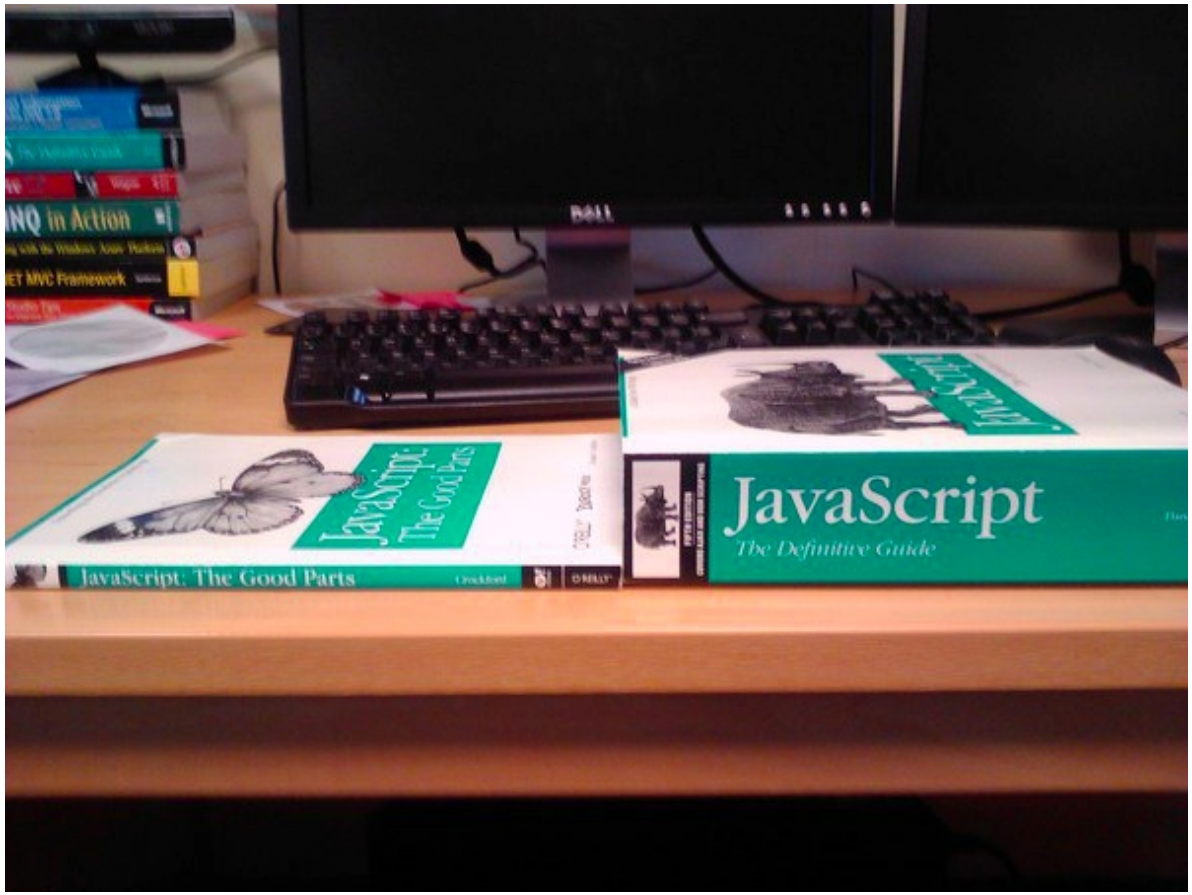
Victoria University of Wellington

{Kieran.Gorman,tim,lindsay}@ecs.vuw.ac.nz

December 16, 2013

Web Programming

Everyone wants to program for the web! But… JavaScript



Transpiling

Aiming for JavaScript as a compile target

Decent VMs, Emscripten and ASM.js

> *The point is, JS is as low as we can go.*
>
> — Brendan Eich

Event-Driven Programming

JavaScript is tailored to event-driven programming

Except it's not very good at it

```
var read = 0, total = files.length;
for (var i = 0; i < total; i++) {
  var file = files[i];
  read(file, function(err) {
    if (err) console.error("failed to read a file");
    if (++read == total)
      console.log(file + " was last");
  });
}
```

Reactive Programming

Essentially about defining a directed graph of values

Spreadsheets!

What we want to express:

$$x = y + z$$

What we have to work with:

```
function updateX() { x.set(y + z); }
y.onchange = updateX;
z.onchange = updateX;
updateX();
```

Functional Reactive Programming

Representing change in time as a pure function

$$\textbf{type } \mathit{Signal}\ a = \mathit{Time} \to a$$
$$\textbf{type } \mathit{Animation} = \mathit{Signal}\ \mathit{Image}$$

Build new signals from existing ones

FRP on the Web

Directing JavaScript with Arrows — FRP possible in JavaScript, but…

Need a strong static type system to enforce crucial properties

Haskell to the rescue!

Getting Haskell in a browser

- Fay

Getting Haskell in a browser

- Fay
- GHCJS

Getting Haskell in a browser

- Fay
- GHCJS
- UHC

Getting Haskell in a browser

- Fay
- GHCJS
- UHC

Elm?

Melchior

Features

Event-driven ‘Push’ Signals

JavaScript runtime exceeds feature parity with Flapjax/Arrowlets

Haskell framework meets feature parity with Elm

Type-safe DOM selection API

$$inputs \circ byClass\ \texttt{"big"}$$

Timer — In JavaScript

```
// Select an element from the page
var content = document.getElementById("timer");
var elapsed = 0;

// Every second, increment the elapsed time and update the label
setInterval(function () {
    content.innerHTML = elapsed++;
}, 1000);

// When we click the label, reset to zero
content.onclick = function () {
    elapsed = 0;
};
```

Timer — In Melchior

Select an element from the page and bind the label

```
bindTimer :: [Element] → Dom ()
bindTimer html = do
  timer ← select (byId "timer") html
  bind timer (elapsed timer)
```

Every second, increment the elapsed time or reset on click

```
elapsed :: Element → Signal Int
elapsed elem = foldp (λc e → either (e + 1) 0 c) 0
  (every second & clicks elem)
```

Ajax

Opaquely combine asynchronous actions

$$request\ POST\ \texttt{"/courses"}\ (const\ courses\ \textcircled{\$}\ clicks\ submit)$$

Share datatypes and functions between client and server!

$$\textbf{data}\ Course\ \{title :: String, code :: Int, points :: Int\}$$

instance *Json Course* **where**

...

$$request :: (Json\ a, Json\ b) \Rightarrow Method \rightarrow Path \rightarrow a \rightarrow Signal\ b$$

A Haskell Nicety

A little type-class trick:

$$\begin{aligned}&\textbf{instance } \mathit{Num}\ a \Rightarrow \mathit{Num}\ (\mathit{Signal}\ a) \textbf{ where}\\ &\quad \mathit{fromInteger} = \mathit{constant}\\ &\quad a + b = (+) \circledast a \circledast b\\ &\quad a - b = (-) \circledast a \circledast b\end{aligned}$$

And now:

$$\begin{aligned}&x :: \mathit{Signal}\ \mathit{Int}\\ &x = y + z\end{aligned}$$

Future Work

- Tidy up the API
- Lens selectors?
- Deal with browser inconsistencies
- Jump on board the GHCJS bandwagon

Source

`github.com/kjgorman/489-uhc`