A Haskell Platform for Creating Progressive Web Applications

A midterm progress report

Jakub Zárybnický

2019-01-29

Outline

Motivation

- The trend of FP on the frontend, motivated by Elm, PureScript
- The full power of Haskell, sharing common code with the server
- Rapidly growing area in the Haskell ecosystem, many companies involved, OSS contributors:
  - IOHK (Cardano) - iohk-ops, distributed processing
  - Obsidian - Reflex, Obelisk, Rhyolite
  - Tweag - Asterius, Inline-js
  - QFPL - Reflex-workshop, UI components
- Growing, but still not established - missing tools and libraries
- More immediate need - I have clients with projects that are waiting on this thesis' results

Assignment

- My task:
    - *"A Haskell Platform for Creating Progressive Web Applications"*

Assignment

- My task:
  - *"A Haskell Platform for Creating Progressive Web Applications"*
- What's a PWA?
  - "Progressive Web Application"
  - new buzzword from Google
  - an *almost* native app
    - load a website
    - save to phone homepage like an app
    - available offline, perhaps with data sync
    - use push notifications, device APIs

Motivation

- Why this particular topic?
  - many missing tools in the ecosystem
  - I chose PWA - personal need, started a new project in August and September without necessary tools
  - The projects is mostly done, but it still isn't a PWA

Motivation

- Why this particular topic?
  - many missing tools in the ecosystem
  - I chose PWA - personal need, started a new project in August and September without necessary tools
  - The projects is mostly done, but it still isn't a PWA
- Since the start of the project, my real-world goals have drifted a bit:
  - PWA framework and a set of tools
  - full-stack framework, Meteor-alike
  - all the web-dev tools missing from Haskell
  - application patterns
    - static site, JAM (recompile-on-demand), real-time app, . . .

What have I been doing?

- How have I been working?
    - in part extracting relevant parts from my existing Haskell applications
    - in part research, type and API design

What have I been doing?

- How have I been working?
  - in part extracting relevant parts from my existing Haskell applications
  - in part research, type and API design
- What worked out?
  - survey of web frameworks all around the programming world
  - survey of the Haskell ecosystem around web development
  - proof-of-concept of a full-stack application
  - proof-of-concept of a JAM-stack-alike
  - proof-of-concept of an offline-capable client application
  - proof-of-concept of a frontend debugger toolbar (quite limited so far)

What haven't I been doing?

- What didn't work out?
  - attempts at a blog with regular updates - a failure this far
  - attempts at type design for the sync and transport part - I found a lot of ways that don't work or read/write well
  - a mile long to-do list and a backlog of ideas and research topics

What haven't I been doing?

- What didn't work out?
  - attempts at a blog with regular updates - a failure this far
  - attempts at type design for the sync and transport part - I found a lot of ways that don't work or read/write well
  - a mile long to-do list and a backlog of ideas and research topics
- Non-goals:
  - write and publish code
  - write the thesis itself

Technologies

The main ones:

- Haskell
- Nix

TODO: add logos!

Haskell

- since 1990, actively developed, rooted in academia and still the target of active research - "Dependent Haskell", "Linear Haskell"
- purely functional programming language with type inference and lazy evaluation
- expressive types, dialog with the compiler, types encoding effects
- language to write reliable software in - it eliminates entire classes of programming errors - usually the ones that remain are logic errors
- example of a nicely expressive type - servant

Haskell

```haskell
type HackageAPI =
  "users" :> Get '[JSON] [User] :<|>
  "user" :> Capture "login" Login :> Get '[JSON] User :<|>
  "packages" :> Get '[JSON] [Package]

getUsers :: Handler [User]
getUser :: Login -> Handler User
getPackages :: Handler [Package]

server :: Server HackageApi
server = getUsers :<|> getUser :<|> getPackages

getUsers :<|> getUser :<|> getPackages =
  client @HackageApi "http://hackage.haskell.org"
```

Nix

- since 2004, Eelco Dolstra's Ph.D. thesis in 2006
- purely functional, lazy evaluation
- one program consists of a closure that includes all dependencies including libc
- atomic upgrades, rollbacks
- complete isolation of dependencies, no DLL hell
- NixOS = OS built on top of Nix
- NixOps = a cloud deployment tool
    - conceptually: Terraform/CloudFormation + Puppet/Ansible
    - a network specification -> magic -> running set of servers on AWS, VPSs, VirtualBox, . . .

Nix

```
{
  network.description = "Web server";

  webserver = { config, pkgs, ... }: {
    services.httpd.enable = true;
    services.httpd.adminAddr = "alice@example.org";
    services.httpd.documentRoot =
      "${pkgs.valgrind.doc}/share/doc/valgrind/html";
    networking.firewall.allowedTCPPorts = [ 80 ];

    deployment.targetEnv = "virtualbox";
  };
}
```

Plans

Three tiers of plans:

- PWA - basic, as per assignment
- data sync - extra goal, would be quite an achievement
- pattern language - a vision for the future

TODO: add images (see board)

Next tasks

Wrapping up unfinished tasks:

- finish article drafts and publish them
- finish extracting useful patterns from my applications

Starting work on new areas:

- ServiceWorker wrapper or template
- push notifications
- pre-rendering (build- or runtime)
- CLI tool
- type design for data channel/synchronization

Finishing up

- four more months until the publication deadline
- tons of work left, mile-long lists of tasks and ideas
- basics are well underway
- many stretch goals