Contribution Plan - PureScript compiler caching, filha225

Technical difficulty

Build and Test Environment

Build and test environment is already set up with a CI job that runs some tests and builds. The entire project is built using `stack` (one of two popular Haskell package managers). `stack` automatically installs dependencies and the like.

There are custom tools for setting up a purescript project, running the compiler on it, fetching the resulting build files and compiler output, comparing it against known good values, and tearing it down again before running the next test. This is quite a complex machine, but it's easy to use as long as it works but likely an incredible mess if it needs to be extended or changed.

The build system is **medium**, iff no changes to the custom compiler test suite are needed, otherwise hard.

Domain knowledge

The project is a compiler. Compilers are one of the most studied areas in computer science. PureScript would require knowledge of compilers and strong fundamentals in computer science to work on. Pure functional languages also differ from what's usually taught in compiler courses at university. Luckily, I've worked professionally with this for a couple of years, so I already have most of the knowledge needed.

I will classify the project as **hard** in domain knowledge.

Project size

Running `cloc` on the main PureScript compiler repository give this:

Language	files	blank	comment	code
Haskell	220	5391	6901	34574
PureScript	965	3874	877	9170
Markdown	22	1648	0	4271
CSS	2	85	283	799
yacc	1	122	0	689
LESS	1	166	116	624
JavaScript	38	25	20	512
Bourne Shell	6	63	95	244
YAML	4	47	109	221
JSON	14	0	0	194
make	1	21	11	47

```
dhall                1          5         10         19
Text                 1          6          0          9
XML                  1          0          0          2
-------------------------------------------------------
SUM:              1277      11453       8422      51375
-------------------------------------------------------
```

The naive measurement here is 51ksloc, but I would argue that this much too low an estimate. - There are many dependencies, internal and external - The project is spread out among many repositories, where the 79 repositories under the purescript organization on github total about 261ksloc. - The main purescript organization only contains one compiler frontend-backend pair; each backend lives under its own organzation, and there are multiple frontends too. For example, the github organization for the erlang backend for the main purescript compiler frontend has 63 repos. - Pure functional programs are often much shorter (5-10x) than the equivalent program in an imperative language https://youtu.be/BXmOlCy0oBM?t=69 - One needs to match up the versions of the various components to make things work. Each component is tested against a couple of versions of each other component, and they're mixed and matched quite freely.

Sloc per repository in the purescript organization

```
docker-haskell 985
documentation 6329
governance 184
gsoc infrastructure 49
logo 468
npm-installer 2707
package-sets 11245
package-sets-preview 8890
psc-package 1140
purescript 51366
purescript-arrays 3025
purescript-assert 252
purescript-bifunctors 226
purescript-catenable-lists 705
purescript-console 286
purescript-const 177
purescript-contravariant 309
purescript-control 337
purescript-datetime 1408
purescript-distributive 179
purescript-docs-search 7958
purescript-effect 664
purescript-either 531
```

```
purescript-enums 762
purescript-exceptions 256
purescript-exists 180
purescript-filterable 742
purescript-foldable-traversable 2102
purescript-foreign 694
purescript-foreign-object 867
purescript-free 1369
purescript-functions 381
purescript-functors 880
purescript-gen 416
purescript-graphs 290
purescript-identity 190
purescript-in-purescript 8260
purescript-integers 540
purescript-invariant 155
purescript-lazy 275
purescript-lcg 167
purescript-lists 2899
purescript-maybe 310
purescript-metadata 21
purescript-minibench 224
purescript-newtype 276
purescript-nonempty 256
purescript-numbers 719
purescript-ordered-collections 2048
purescript-orders 190
purescript-parallel 347
purescript-partial 335
purescript-prelude 2749
purescript-profunctor 350
purescript-psci-support 170
purescript-quickcheck 1123
purescript-random 171
purescript-record 575
purescript-refs 275
purescript-safe-coerce 156
purescript-semirings 176
purescript-st 351
purescript-strings 3593
purescript-tailrec 376
purescript-transformers 1807
purescript-tuples 355
purescript-type-equality 148
purescript-typelevel-prelude 391
purescript-unfoldable 383
```

```
purescript-unsafe-coerce 164
purescript-validation 383
purescript.github.io 297
pursuit 3689
pursuit-backups 4778
registry 86080
registry-dev 10765
registry-index roadmap spago 8524
survey 4406
trypurescript 3918
```

261724 sloc

I'd classify the project size as **hard**.

Process difficulty

Issue tracker

The issues have no difficulty markers whatsoever. There are multiple issue trackers. It's easy to ask in Discord but most compiler devs are in the US, and mostly on the west coast, so time zones might be an issue.

I'd classify it as **medium** difficulty, even though it seems to be hard from the above points. It's fairly easy to see that an issue is way out of your league, from not understanding half the words in the issue description.

Documentation

No clear introduction for new developers - some documented code and a lot of `todo`s for various errors. There is a documentation repo, but it looks much better than it is, since many of the markdown files are placeholders, like https://github.com/purescript/documentation/blob/master/errors/CannotUseBindWithDo.md

The project is **hard** in this respect.

Communications

PureScript has a Discord server with a specific channel for compiler development.

It'd therefore classify the project as **medium**.

Development process

Group contributions

I plan on actively helping out the team, since I've worked with this before.

I'm aiming for **hard**.

Planning and reflection

The work I'll be doing is very very hard to estimate, so I'll be struggling in this area but I'll do my best to plan things out. The main time sink is that for each test release I have to get the compiler out to people and wait for them to use it and find bugs which I can fix. Sometimes I know things to fix, and I have a backlog, and sometimes I'll find things myself, but more often than not that backlog will be empty and I'll be waiting for input and trying things out myself to find bugs myself. One real risk with this project is that the above iteration cycle doesn't result in a finished product before the course ends, but my plan B for that is to merge some of the finished parts just to get a contribution merged, and fix the rest later.

I'll do the **easy**.

Engineering practice

I'm going to have to update and significantly expand the test suite to handle all new cases. This will be a large part of the work needed to get something merged; basically convincing others that it should be merged. I'm guessing this means that medium is the bare minimum to get things merged, but it might end up higher too.

I'll do the **medium**.

Individual Contributions

Quantity of code contributions

There will be several types of contributions as part of one big contribution. I've got plenty of stretch goals too which I'll do as part of this course or after this course.

I'll do the **hard**.

Complexity of code contributions

Writing a proper state-of-the-art caching system for the compiler, something that nobody else has properly done in the last 10 years, and which I've only seen in one langauge in this family, is definitely a contribution that solves a challenging problem. There have been a couple of failed attempts at this before too, in the same code base.

I'll do the **hard**.

External communications

I'm proposing a new feature to the project; there's no need to rebuild a module because its dependency changed, if that dependency's public api didn't change.

I'll do the **hard**.

Points summary:

- Technical difficulty: medium+hard+hard=hard
- Process difficulty: medium+hard+medium=medium
- Development process: hard+???+medium=medium
- Individual contributions: hard+hard+hard=hard
- Sum: 18+6+8+24=56, i.e. grade 5, which is what I'm aiming for

Tooling

- Stack / stackage (haskell build system and package manager)
- Cabal (if needed, the other big haskell build system and package manager, most projects use both but only interact via one of them)
- Github (git commits, github issue, github pull request)
- Not sure how we'll track progress within the team; I bet we'll figure something out together. For me, tracking things via github issues and pull requests are enough.
- The proof that we're actively supporting each other comes from asking each team member if they've gotten help from any other team member. I don't see a simpler way to do that. Edvard, Erik and I all work in the same team at the same company, and we're friends outside work, so we've got plenty of communication between us. I bet we'll set up a discord or similar for the project so that others can join too.
- Editor
- Hspec
- Quickcheck