Background

Author: Paul Jolly

For those who were unable to make GopherCon 2018, I'll provide a bit of background as to how these sessions came about. Apologies for the length and format of this document; it's tricky trying to cater for the varying levels of context that people will have, from those here at GopherCon that I've spoken to at length, to those who unfortunately couldn't be here and might be seeing this for the first time.

I went along to Monday's Go Contributor Summit with the hope of having a session where we could discuss go modules tooling and editor/IDE support. Then Marcel, Ian and Robert rained on my parade :)

But I got chatting to Ian Cottrell about go/packages, tooling, editor/IDE support for Go amongst other things, and thought it would be useful to gather interested parties at GopherCon to talk about all of these topics in the context of Go modules, but also generally where things are heading.

I did a best-efforts job of gathering a list of folks I thought would be interested in such a session. I also augmented that list with folks I know would also be interested but were unfortunately not at GopherCon. These notes are being shared with that wider, all-encompassing list, a list that is now defined by membership of https://groups.google.com/forum/#!forum/golang-tools. All are welcome and should just join that list.

Any errors/omissions/etc with anything below, please comment as appropriate and I'll fix things up.

"Kick off" session - Tue 28 Aug

- Present
    - Ramya Rao (vscode-go)
    - Joe Fitzgerald (Atom go-plus)
    - Zac Bergquist (Atom go-plus)
    - Alexander Zolotov (GoLand)
    - Keegan Carruthers-Smith (SourceGraph)
    - Josh Bleecher Snyder
    - Caleb Sapre
    - Ian Cottrell
    - Daniel Martí
    - Paul Jolly
- We went round the table to do brief intros with everyone explaining their interest in the topic
- Current state of the nation:
    - A number of tools "know" nothing about (and hence are broken from a user's perspective) Go 1.11 module mode; e.g. goimports, gocode, guru etc
    - The editor plugins (vscode-go, go-plus, vim-go, less so GoLand because of their more native support for tooling/modules) are starting to see more questions about strange/random behaviour of these tools with Go 1.11; most likely because sometimes things are working by accident of relying on code pre-existing in GOPATH
    - go/packages is ready to be used by tool authors and switching tools to use it will fix the modules issues whilst retaining backwards compatibility
- Ian gave a bit of background:
    - Ian explained that he heads up the Go tooling team in NY
    - Current efforts to help fix, in the short term, tools to enable Go 1.11 module support
        - go/packages - a brief primer on what it is
        - The work his team are doing to try and help with the migration of tools to go/packages; e.g. Rebecca Stambler will shortly (next couple of weeks) be working on fixing github.com/mdempsky/gocode (it was noted that github.com/nsf/gocode has been officially retired)

        - How their team used vscode-go's tool dependencies as an initial reference point for the tooling migration TODO list; ~~Ian to share snapshot of progress~~. (https://groups.google.com/d/msg/golang-tools/ZrjgsgTyR6Y/aCxqm4Z8AgAJ). Includes, by definition, a list of who/which tools the team are in contact with.
        - Ian stressed that neither he nor anyone in his team wants to tread on anyone's toes as part of this process; rather they want to volunteer their help/support in fixing things and facilitate where that's helpful
    - One of the reasons Go was so readily adopted in the early days was the initial editor support (e.g. Emacs, Vim etc). That editor support blossomed in the form of vscode-go, go-plus, vim-go, GoLand etc, but in Ian's opinion the Go team stepped too far back on the *tooling* front
    - A number of tools *not* used by vscode-go, go-plus, vim-go, GoLand etc are totally unknown, precisely because those editor/IDE plugins don't reference them (e.g. https://godoc.org/golang.org/x/tools/cmd/eg). By no means the fault of any of these editors/IDE, rather that it's impossible to keep track of the patchwork of tools out there, there's no good directory or equivalent
- Ian therefore shared a bit of his vision for where things should be heading:
    - An LSP is the best thing long-term; would be shipped and fully supported by his team
    - Becomes the unification point for all these good tools
    - And helps to simplify the "interface" that editors/IDEs talk to for that common and consistent set of tooling/features
- There are however a number of important open questions about LSP
    - What would the work split be?
    - Reuse of existing code - to what extend would this be possible? Particularly in the context of a number of tools needing to be rewritten to support Go 1.11 modules
    - … plus many others not even covered
- Alexander shared some stats from GoLand in terms of the dependency management system being used for new projects in GoLand over the last month (note these are opt-in stats):
    - 27% dep
    - 9% glide
    - 3% godep
    - 3% vgo/go modules
- Folks present wanted to ensure a couple of tools/issues were definitely on Ian's TODO list
    - gopkgs
    - godoc
    - (per Josh) a tool to help suggest ways to shrink your module dependencies
        - Per Ian, Alan Donovan has ideas in this space
    - Sublime text support
    - Question on whether a go list fix made it into Go 1.10.4 - ~~Ian to check~~
        - Michael confirmed it got missed somehow. Michael and Bryan to introduce in 1.10.5
- In the context of go/packages (and any higher level packages that build atop it) it was also felt that issuing guidance for other tool creators would be useful
    - How to use go/packages
    - Making contact with Ian's team so that both sides are in touch. Ian wants to be open and help with all tools and editors wherever possible
- Ian mentioned speed problems that become obvious in tools like gocode and godef
    - Ian has a rewrite of Roger Peppé's godef, that starts to feel a bit sluggish on a good size go.mod
    - Someone in the core Go team will be working on go list performance in the next two months to help alleviate this
- The question was raised, can we have experimental versions of the editor plugins that people opt in to?
    - This would be an *editor* level setting to say "yes I want modules support" atop a using developing outside of GOPATH or setting GO111MODULE=on
    - Ramya (vscode-go) and Joe (go-plus) were in favour of this
    - A number of open questions about what exact form this would take, how multiple copies of tools would work etc… but broad agreement this would be a good thing, in the spirit of the initial vgo experiment and the current opting in to module support in Go 1.11

- Joe raised the fact that many people are asking the question “how do I go get *outside* of a module, to globally install tool X?”
    - This is covered by https://github.com/golang/go/issues/24250
- Discussed how best to structure projects with tool dependencies
    - For now this is covered by https://github.com/golang/go/issues/25922
- Joe suggested a more regular meeting with the wider group (including of course those not present at GopherCon)
    - A good mailing list would be a start; that can serve as the record of “interested parties”
    - Hangouts might be useful on occasion?
    - Likely there will be more sessions required in the short term whilst the current tool issues are being fixed
    - Conversation will then shift more towards the medium/longer term LSP etc
- Proposal for a second session for Wed at GopherCon (again, notes to be distributed):
    - Editor changes required for module support: current module concept
        - How we handle multiple editors windows, splits etc
        - How do we maintain context
        - Formalising multiple contexts accessing tools
    - Speed changes required for go list to make things usable
    - Editor opt-in mode
    - Create a mega umbrella issue for all “tooling” then tool/editor-specific issues linking to/linked from that (so people can track what’s happening in, for example, vscode-go)
    - Timings for regular meetings
    - More discussion/thinking about LSP
        - Open questions
        - Concerns etc
    - ~~Paul to arrange~~
- Ramya also noted that at previous GopherCons the community had been a great time to talk through ideas more - should do the same again this year for whoever remains in Denver come Thursday
- Paul to share wider list of people who would have been invited but were unable to make GopherCon (again, this is a WIP list, please let me know of omissions) - see below. These are either people directly interested (e.g. Ramya, Joe, Alexander, Fatih etc) or who have a passing interest because of dependencies (e.g. Bryan because of his involvement with go modules)
- Ramya made the point that with a mega tracking umbrella issue and linked “sub” issues it would help vscode-go, go-plus, atom, GoLand etc be a bit more synchronised and coordinated in their communication with users. E.g. for progress reports on module tooling, best practices for doing X

“Follow up” session - Wed 29 Aug

- Present
    - Bryan Mills
    - Ian Cottrell
    - Jay Conrod
    - Michael Matloob
    - Caleb Spare
    - Daniel Martí
    - Marwan Sulaiman
    - Joe Fitzgerald
    - Zac Bergquist (apologies, everywhere above/below that I’ve written “Joe” I guess I should have written Joe/Zac?)
    - Keegan Carruthers-Smith
    - Alexander Zolotov
    - Paul Jolly
    - Ivan Daniluk
    - Ramya Rao

- Agenda
    - Editor changes required for module support: current module concept
        - How we handle multiple editors windows, splits etc
        - How do we maintain context
        - Formalising multiple contexts accessing tools
    - Speed changes required for go list to make things usable
    - Editor opt-in mode
    - Create a mega umbrella issue for all “tooling” then tool/editor-specific issues linking to/linked from that (so people can track what’s happening in, for example, vscode-go)
    - Timings for regular meetings
    - More discussion/thinking about LSP
        - Open questions
        - Concerns etc
- Ramya and Joe brought up the question of go.mod (TextMate) grammar files
    - Useful in case someone opens/edits a go.mod file
    - To take discussion offline and include Fatih too
- Ramya and Joe mentioned there could be gocode issues with Go 1.11 inside GOPATH
    - Follow up offline
- The issue of tooling speed came up again in the context of goimports
- Follow up post-GopherCon
    - Bryan created https://groups.google.com/forum/#!forum/golang-tools as a mailing list for anyone interested in tooling
    - We can therefore arrange online hangouts/sessions etc at a later date via the mailing list
    - #tools on Slack is also a place to hangout… but important to surface key points/decisions etc via the mailing list for a good, public record
- Editor opt-in mode and context discussion
    - Ramya and Joe comfortable with idea of a tool split for the opt-in mode; they will work out more details offline (and we should also loop in Fatih to the discussion)
    - Having an editor opt-in will help to soften the change in experience if any of the module-aware tools are slower (or faster)
    - go env was mentioned as the way to determine the module for a current file (via GOMOD)
    - But this doesn’t necessarily correspond to the context required in editors (in fact it’s almost certainly *not* the correct context in 99% of cases in an editor)
    - Rather “open project” or “open a directory” *are* the context that is required for go/packages-based tools; we noted however that Vim doesn’t have an analogous concept… nor a consistently rich interface for surfacing what the current context is
    - A question came up (similar to the discussion in https://github.com/golang/go/issues/27227) about the context/mode that should be assumed If vendor directory is present
        - General feeling was that https://github.com/golang/go/issues/27227 isn’t a good idea… and it further complicates the questions of context/mode
    - So assuming editors have a “open directory/project” concept or (as in the case of Vim) gain the concept of “current context (read directory)”, the workflow around working on a fork of a project where someone will be contributing a fix upstream can be considered separately, because in that situation they *will* need to switch contexts. Tooling like https://github.com/rogpeppe/gohack will help… but more details to be worked out here
    - Joe re-emphasised the problem of multiple windows and the question of context (specifically when launching a new window from a terminal, and the lengths Atom goes to in order to “resolve” a context)
    - Discussed the fact that the context for a given editor window/pane could extend to include
        - GOFLAGS settable flags
        - GOPROXY
    - Opening random files unconnected via the transitive dependency graph to the current module was also parked
- Discussion about multi-module repositories was also parked for now
- There was a discussion about the module cache and files in the cache that are reached via godef/equivalent

        - Bryan re-emphasised why the cache is read-only by default (and shouldn't be changed, Marwan :)
        - We referenced the earlier discussion about gohack and how somebody *wanting* to make a change would use such a tool/workflow to flip out of the module cache
        - Ivan raised the question of whether obfuscating the file names in the cache would help to discourage people from editing them, hence signposting them towards a gohack-esque workflow
            - Felt like there were more costs (particularly for editors) to benefits to this
- Next steps
    - ~~Setup mailing list~~: done
        - https://groups.google.com/forum/#!forum/golang-tools
    - ~~Agree on Slack group~~: done
        - #tools (for now)
    - Discuss via mailing list regular follow up sessions: Ian C
    - Editor opt-in mode: Ian, Ramya, Joe, Fatih
    - Share these notes with golang-tools: Paul
    - Various smaller points above

~~Current WIP wider list of interested parties~~

This has now been replaced by https://groups.google.com/forum/#!forum/golang-tools

~~Ideally this becomes a mailing list or similar to avoid the hopeless admin of maintaining it in a Google Doc; this is just a start.~~

~~*Direct interest:*~~

~~*Ramya Rao*~~
~~*Joe Fitzgerald*~~
~~*Zac Bergquist*~~
~~*Fatih Arslan*~~
~~*Florin Pățan*~~
~~*Alexander Zolotov*~~
~~*Keegan Carruthers-Smith*~~
~~*Dominik Honnef*~~
~~*Daniel Martí*~~
~~*Roger Peppé*~~
~~*Ian Cottrell and team*~~
~~*Matthew Dempsky*~~
~~*Josh Bleecher Snyder*~~
~~*Jimmy Frasche*~~
~~*Emmanuel T Odeke*~~
~~*Ivan Daniluk*~~
~~*Marwan Sulaiman*~~
~~*Caleb Sapre*~~
~~*Paul Jolly*~~

~~*Passing interest:*~~

~~*Marcel van Lohuizen*~~
~~*Kevin Burke*~~
~~*Bryan Mills*~~
~~*Richard Musiol*~~
~~*Dmitri Shuralyov*~~
~~*Joe Tsai*~~
~~*Robert Griesemer*~~