An Exploration of Go

Starting Webassembly

Admin

- Questions?
- exam coming soon (or perhaps we just had it)
  - Open book, open note closed neighbor

Web assembly

- So most of you told me you really hadn’t done much with web assembly – so lets start from the beginning
-

Web assembly

- So most of you told me you really hadn't done much with web assembly – so lets start from the beginning
- Once upon a time long long ago….
  - The fine people at netscape (yes **long** ago when I was young and many of you were newborns) needed a scripting language to make the web more dynamic
  -

Web assembly

- So most of you told me you really hadn’t done much with web assembly – so lets start from the beginning
- Once upon a time long long ago….
  - The fine people at netscape (yes **long** ago when I was young and many of you were newborns) needed a scripting language to make the web more dynamic
  - And there was javascript
  -

Web assembly

- So most of you told me you really hadn’t done much with web assembly – so lets start from the beginning
- Once upon a time long long ago….
  - The fine people at netscape (yes **long** ago when I was young and many of you were toddlers) needed a scripting language to make the web more dynamic
  - And there was javascript
  - And devs look on it and saw that it was bad

Web assembly

- So most of you told me you really hadn't done much with web assembly – so lets start from the beginning
- Once upon a time long long ago….
  - The fine people at netscape (yes **long** ago when I was young and many of you were toddlers) needed a scripting language to make the web more dynamic
  - And there was javascript
  - And devs look on it and saw that it was bad
  - But it was the only choice

Web assembly

- So most of you told me you really hadn’t done much with web assembly – so lets start from the beginning
- Once upon a time long long ago….
  - The fine people at netscape (yes **long** ago when I was young and many of you were toddlers) needed a scripting language to make the web more dynamic
  - And there was javascript
  - And devs look on it and saw that it was bad
  - But it was the only choice
  - So javascript became a big deal (And Node was born etc)

Web Assembly history

- But but javascript
  - What are some of its perceived limitations?
  - Lucky volunteer$^{(TM)}$
  -

Web Assembly history

- But but javascript
  - What are some of its perceived limitations?
    - Dynamically typed
      - So popular a decade or so ago, but turns out not to scale
      - Typescript is one of the most popular ecmascript standard languages
    - Slow
      - Lets discuss “slow” in terms of modern cloud based software
    - Lucky volunteer$^{(TM)}$?

Web Assembly history

- But but javascript
  - What are some of its perceived limitations?
    - Dynamically typed
      - So popular a decade or so ago, but turns out not to scale
      - Typescript is one of the most popular ecmascript standard languages
    - Slow
      - Lets discuss “slow” in terms of modern cloud based software
      - For lots of stuff this doesn’t matter – latency for the Boston-Singapore database query is higher than the interpreted language slowdown.
    - If we haven’t covered it yet – when does it matter?

Web Assembly history

- But but javascript
  - What are some of its perceived limitations?
    - Dynamically typed
      - So popular a decade or so ago, but turns out not to scale
      - Typescript is one of the most popular ecmascript standard languages
    - Slow
      - Lets discuss “slow” in terms of modern cloud based software
      - For lots of stuff this doesn’t matter – latency for the Boston-Singapore database query is higher than the interpreted language slowdown.
    - If we haven’t covered it yet – when does it matter?
      - Games
      - Local computation-heavy work
      - Anything else?

Web Assembly History

- So in 2013 – asm.js
  - A strict subset of javascript
  - Designed to allow statically typed languages with manual memory management to be cross compiled to a subset of the javascript vm
    - So give me an example of the target language
    - Lucky volunteer$^{(TM)}$?
  - Provided really good performance.

WebAssembly

- In 2015 the powers that be decided to evolve asm.js into webassemby.
- By 2017 the MVP was out and implemented in “all major browsers”
  - Firefox, chrome, safari, and IE (later also the chrome based edge)

WebAssembly

- Web assembly is a compile target
  - Cross compile – what do we mean?
    - Lucky volunteer(TM)?
- Runs on virtual machine in browser
  - Model similar to java
-

Webassembly

- So my web assembly runs in your browser on your machine
- What are you thinking?

Webassembly

- So my web assembly runs in your browser on your machine
- What are you thinking?
- What could ***possibly*** go wrong?

Webassembly

- So my web assembly runs in your browser on your machine
- What are you thinking?
- What could ***possibly*** go wrong?
  - So hopefully you have deep worries about security
  - Webassembly incorporates lots of lessons learned from earlier technologies and prevents webassembly programs from getting any resources on the local machine
    - Get resources from program itself
    - Or internet.

Go and web assembly

- Go was not the language Mozilla had in mind
  - Mozilla created rust and thought it was an ideal web assembly based language
  - Why?
    - Lucky volunteer$^{(TM)}$?
  -

Go and web assembly

- Go was not the language Mozilla had in mind
  - Mozilla created rust and thought it was an ideal web assembly based language
  - Why?
    - Lucky volunteer^(TM)?
  - Manual memory management
  - When go is compiled to webassembly so is its runtime
    - So application is minimum of 2mb
    - Hows that?
    - Lucky volunteer^(TM)?

Go and web assembly

- Go was not the language Mozilla had in mind
  - Mozilla created rust and thought it was an ideal web assembly based language
  - Why?
    - Lucky volunteer(TM)?
  - Manual memory management
  - When go is compiled to webassembly so is its runtime
    - So application is minimum of 2mb
    - Hows that?
    - Kinda crappy for a web commerce store – pretty ok for a 3d game

Webassembly and go

- Pure go translates to web assembly pretty easily.
  - Go with foreign dependencies is harder (qt gtk etc)
- In fact go has provided us with tools to make go apps web assembly even easier for a while
  - In your go install, there is a misc folder, and in there is a wasm_exec.html, we’ll use this as your index.html
- Make a new folder for your running web project
  - Copy <path to go>/misc/wasm/wasm_exec.html to your project
  - Replace test.wasm with your wasm file.

Go 1.21 (August 2023)

- Go 1.21 increase support for wasi and wasm
  - WebAssembly
    - The new go:wasmimport directive can now be used in Go programs to import functions from the WebAssembly host.
    - The Go scheduler now interacts much more efficiently with the JavaScript event loop, especially in applications that block frequently on asynchronous events.
  - WebAssembly System Interface
    - Go 1.21 adds an experimental port to the WebAssembly System Interface (WASI), Preview 1 (GOOS=wasip1, GOARCH=wasm).
    - As a result of the addition of the new GOOS value “wasip1”, Go files named *_wasip1.go will now be ignored by Go tools except when that GOOS value is being used. If you have existing filenames matching that pattern, you will need to rename them.

Now you need to web serve

- At this point you could just run a heavy duty webserver like ngix
  - And if we were deploying this to lots of people we would
  - But for development lets use a simple webserver that I pulled from a tutorial (reference link included)
- See next slide
  - Put it into a single file, compile it to an executable
  - And then when we have our wasm ready we will run it

```go
package main
//https://itnext.io/webassemply-with-golang-by-scratch-e05ec5230558
import (
    "log"
    "net/http"
)

const(
    AddSrv = ":8080"
    TemplatesDir = "."
)

func main() {
    log.Printf("listening on %q...", AddSrv)
    fileSrv := http.FileServer(http.Dir(TemplatesDir))
    if err := http.ListenAndServe(AddSrv, fileSrv); err!=nil{
        log.Fatal(err)
    }
}
```

Lets have a look

- Lets try this now with the version of the project from the last class
  -

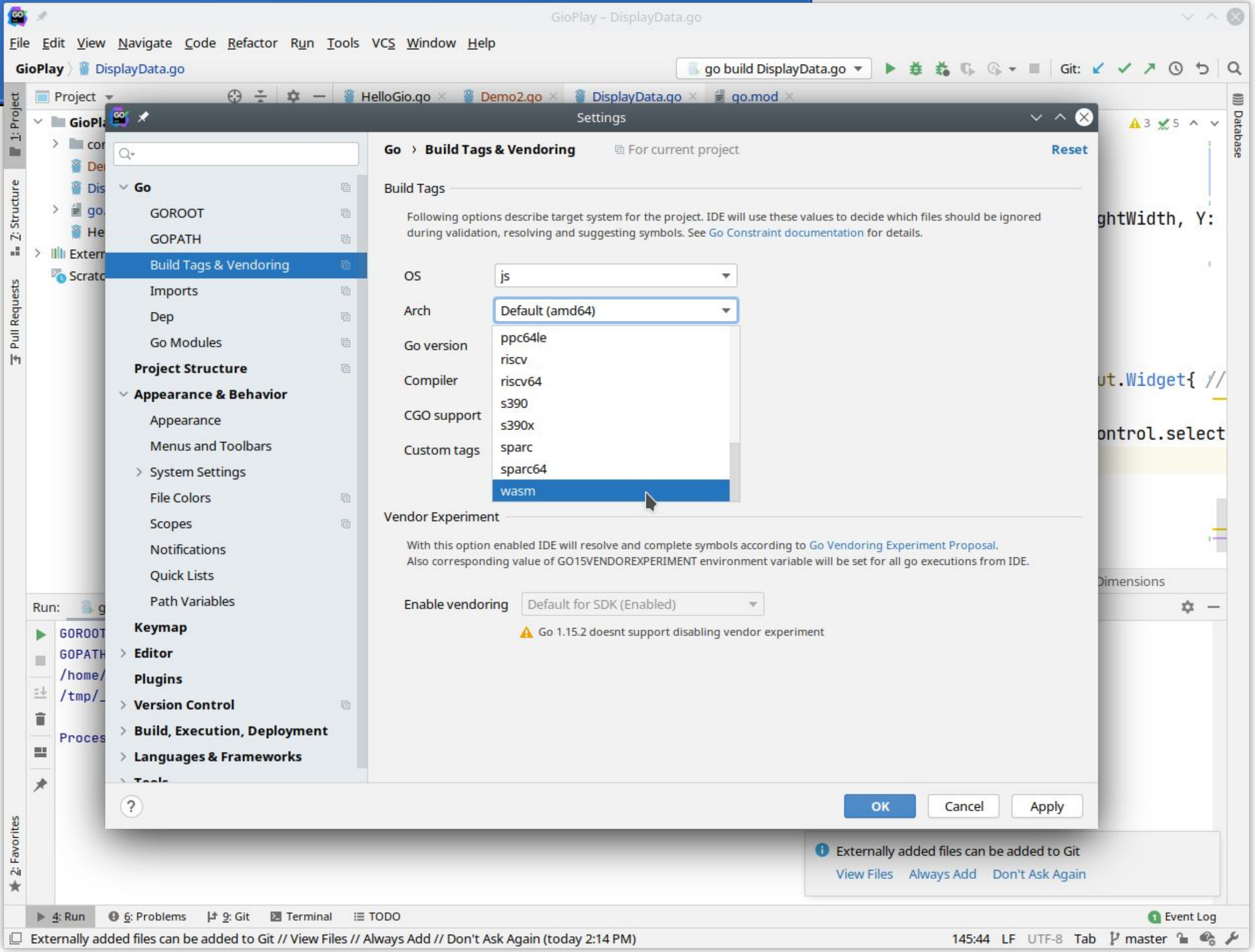
What about non-goland?

- So if we are using goland, then web assembly is easy
  - But go and webassembly without goland is still not bad.
  - At the command line
    - Set the GOOS and GOARCH environment variables
    - For example from the doc I put in the news:
    - GOOS=js GOARCH=wasm go build -o ../../assets/json.wasm <your file name>
  - But the command line is scary??!?!?
    - No but….

From goland

- You can do this painlessly in modern goland
  - Open up the settings/preferences dialog
    - <file> <settings>Or <goland><preferences> in mac land
  - Open the <go> option and select the <build tags & vendoring> option
  - In the OS option choose ‘js’
  - In the Arch option choose wasm
    - Now you are building for web assembly
    - And your executable will no longer run on your local system as before

1000 words:



Go › Build Tags & Vendoring — For current project — Reset

Build Tags

Following options describe target system for the project. IDE will use these values to decide which files should be ignored during validation, resolving and suggesting symbols. See Go Constraint documentation for details.

OS: js

Arch: Default (amd64)

ppc64le
riscv
riscv64
s390
s390x
sparc
sparc64
wasm

Go version

Compiler

CGO support

Custom tags

Vendor Experiment

With this option enabled IDE will resolve and complete symbols according to Go Vendoring Experiment Proposal.
Also corresponding value of GO15VENDOREXPERIMENT environment variable will be set for all go executions from IDE.

Enable vendoring: Default for SDK (Enabled)

Go 1.15.2 doesnt support disabling vendor experiment

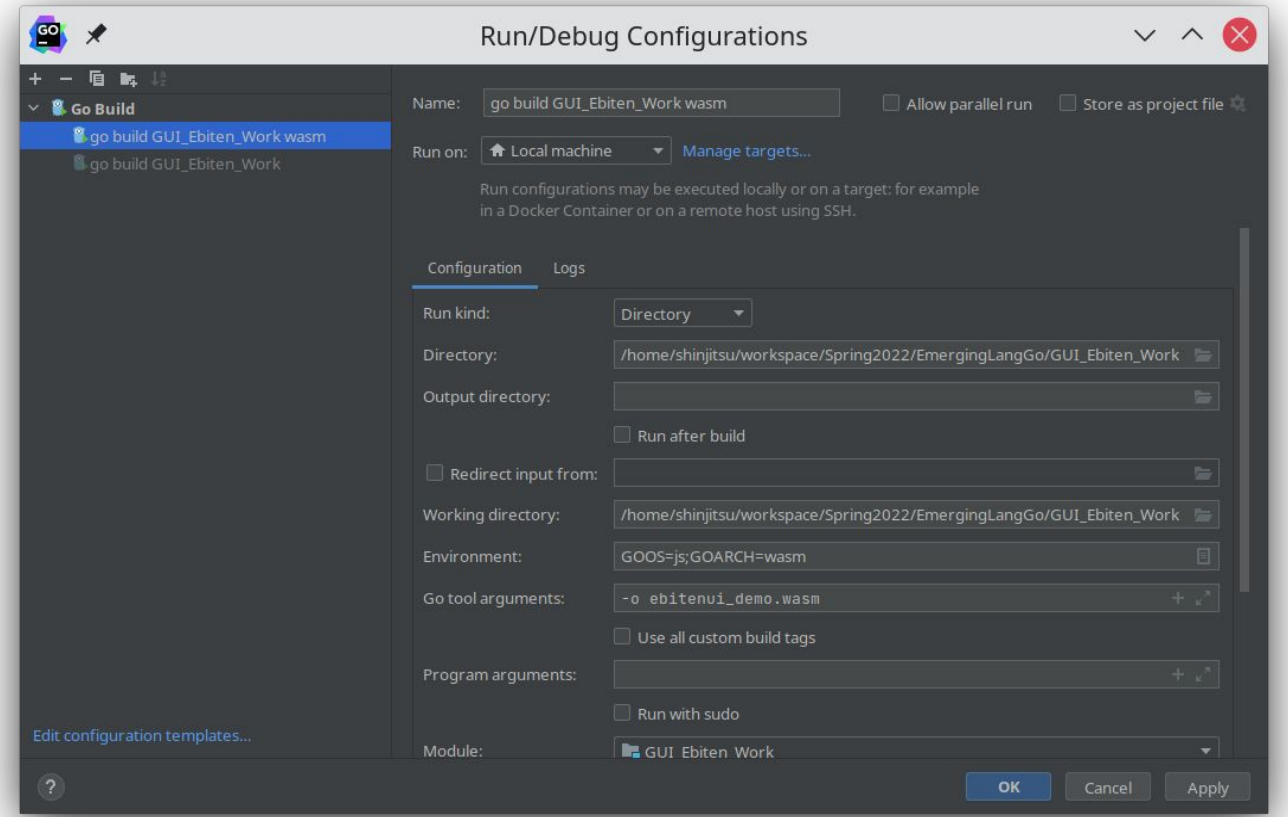
Externally added files can be added to Git
View Files   Always Add   Don't Ask Again

Externally added files can be added to Git // View Files // Always Add // Don't Ask Again (today 2:14 PM)

Edit run configurations

- One last thing to do, go to <run> menu and <edit configurations>
  - Hit the ‘+’ button to add a new configuration
  - Give it a new name (I like to put wasm in the name)
  - Make run kind *directory*
  - Uncheck ‘run after build’
  - Add two new environment variables GOOS=js;GOARCH=wasm
  - Change go tool arguments to -o <your program>.wasm
  - And hit ok
  - Then run your new configuration, nothing run this time, but you should have a wasm file in your folder, move it to your web project folder

Another 1000 words



Name: go build GUI_Ebiten_Work wasm
Allow parallel run
Store as project file
Run on: Local machine
Run configurations may be executed locally or on a target: for example in a Docker Container or on a remote host using SSH.
Run kind: Directory
Directory: /home/shinjitsu/workspace/Spring2022/EmergingLangGo/GUI_Ebiten_Work
Output directory:
Run after build
Redirect input from:
Working directory: /home/shinjitsu/workspace/Spring2022/EmergingLangGo/GUI_Ebiten_Work
Environment: GOOS=js;GOARCH=wasm
Go tool arguments: `-o ebitenui_demo.wasm`
Use all custom build tags
Program arguments:
Run with sudo
Module: GUI_Ebiten_Work

Run it

- Run your compiled web server in the folder with your index.html and wasm program
- Then go to http://localhost:8080
  - And lets see your program in the web browser.

Lets have a look

- Lets try this now with the version of the project from the beginning of class
  - It doesn’t look beautiful, but it works
  - That same project that we just saw on desktop is now on the web browser with minimal extra work.
  -

Index.html

- Super simple one from article in the news:

```
• <html>
•     <head>
•         <meta charset="utf-8"/>
•         <script src="wasm_exec.js"></script>
•         <script>
•             const go = new Go();
•             WebAssembly.instantiateStreaming(fetch("test.wasm"), go.importObject).then((result) => {
•                 go.run(result.instance);
•             });
•         </script>
•     </head>
•     <body></body>
• </html>
```

Lets stop here

- Lets look at the next project
- exam week. (or last week)
-