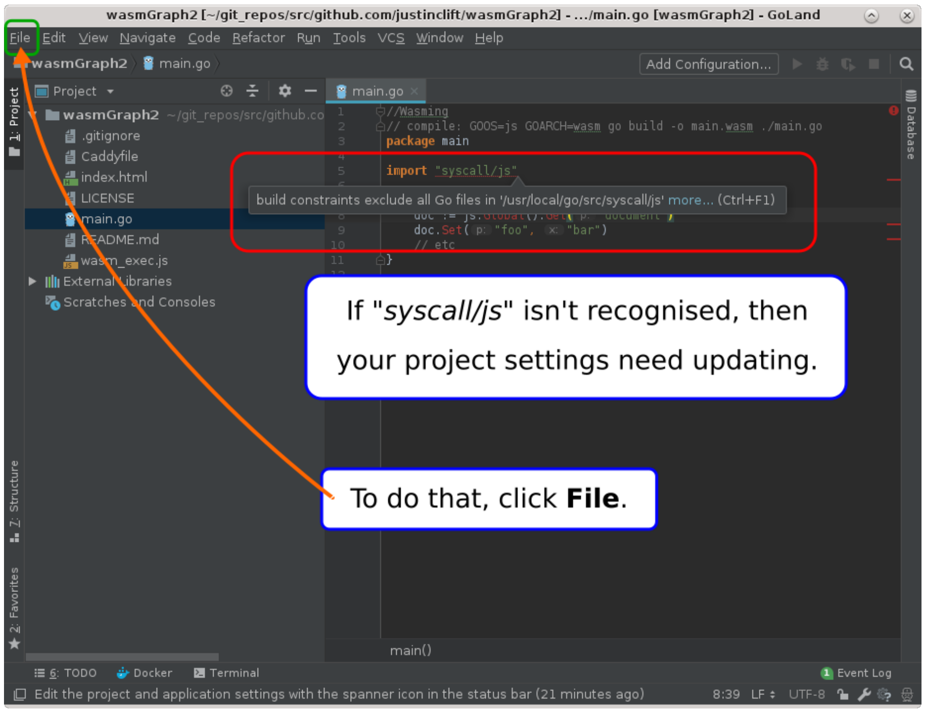
← **Back to the main WebAssembly page.**

Configuring GoLand for WebAssembly (Wasm) projects

Initial project configuration

When you first open or start a WebAssembly project in GoLand, it won't understand the "*syscall/js*" package.

That's easily fixable, by changing the **GOOS** and **GOARCH** values in the project settings, as per the screenshots below.



```go
//Wasming
// compile: GOOS=js GOARCH=wasm go build -o main.wasm ./main.go
package main

import "syscall/js"

    doc := js.Global().Get( p: "document")
    doc.Set( p: "foo",  x: "bar")
    // etc
}
```

build constraints exclude all Go files in '/usr/local/go/src/syscall/js' more... (Ctrl+F1)

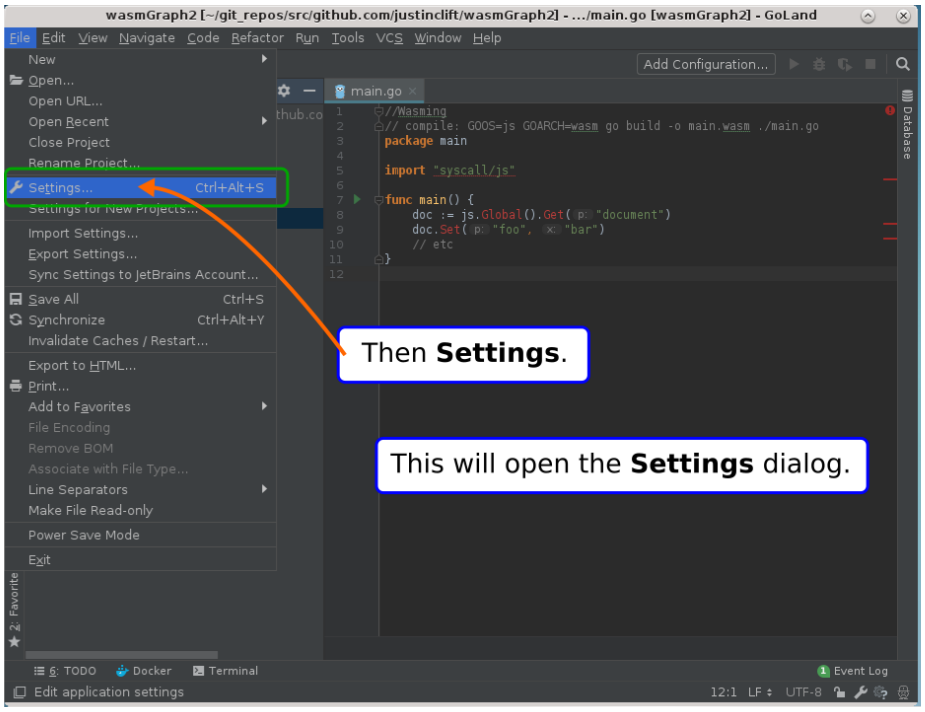
If "*syscall/js*" isn't recognised, then your project settings need updating.

To do that, click **File**.

Edit the project and application settings with the spanner icon in the status bar (21 minutes ago)

Figure 1: GoLand Wasm Setup pic1

Note - The screenshot below shows how to access **Settings** on a Linux desktop. If you're using macOS, you'll probably need to access them through the standard ma-


wasmGraph2 [~/git_repos/src/github.com/justinclift/wasmGraph2] - .../main.go [wasmGraph2] - GoLand

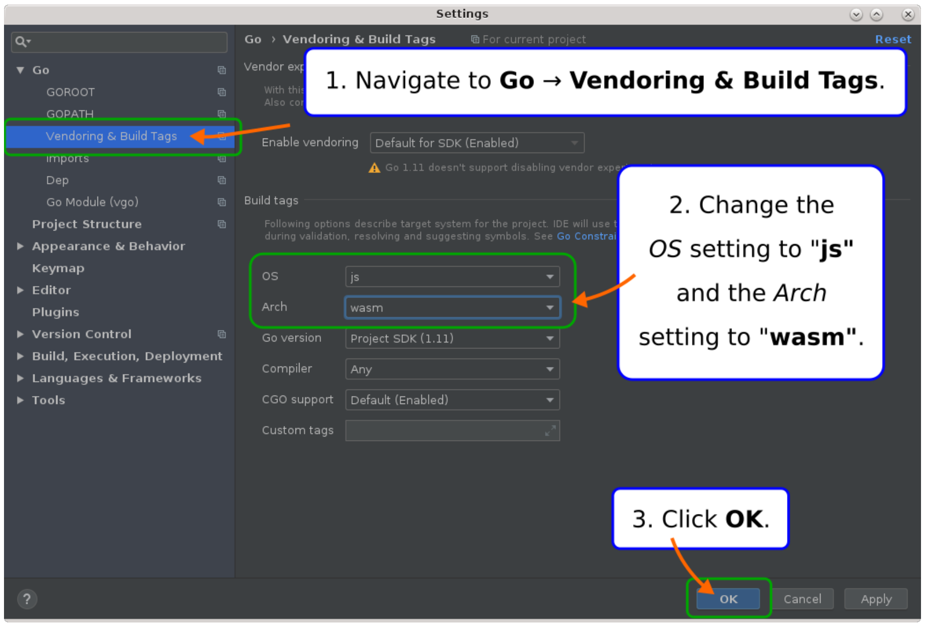
main.go

```go
//Wasming
// compile: GOOS=js GOARCH=wasm go build -o main.wasm ./main.go
package main

import "syscall/js"

func main() {
	doc := js.Global().Get( p: "document")
	doc.Set( p: "foo",  x: "bar")
	// etc
}
```

Then **Settings**.

This will open the **Settings** dialog.

cOS **Preferences** option.


Settings

Go › Vendoring & Build Tags — For current project

Enable vendoring: Default for SDK (Enabled)

Go 1.11 doesn't support disabling vendor expe

Build tags

Following options describe target system for the project. IDE will use t during validation, resolving and suggesting symbols. See Go Constrai

OS: js

Arch: wasm

Go version: Project SDK (1.11)

Compiler: Any

CGO support: Default (Enabled)

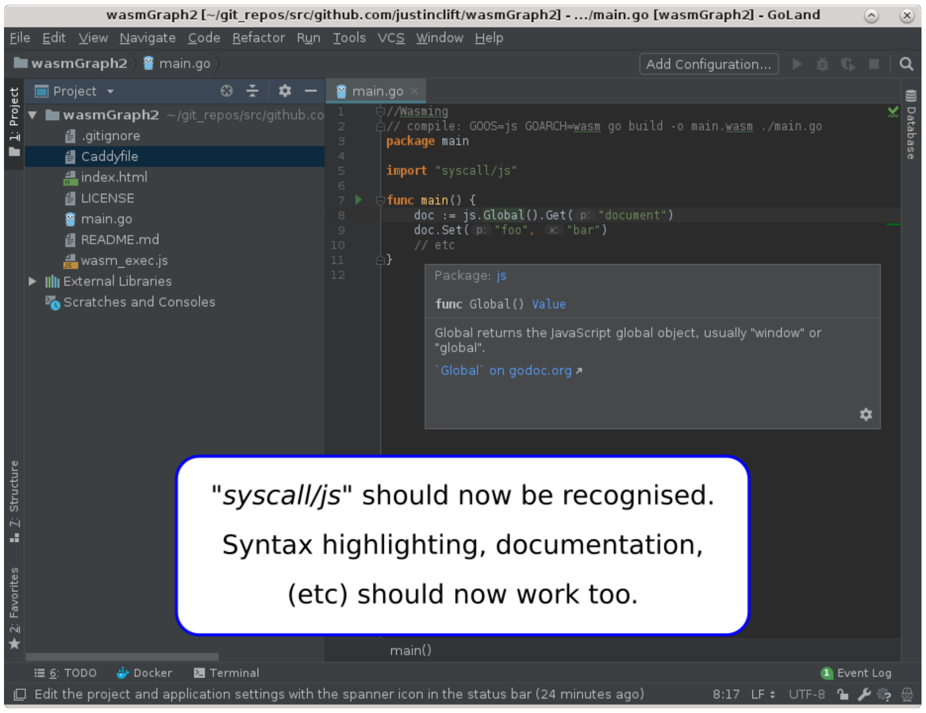
Custom tags:

1. Navigate to **Go** → **Vendoring & Build Tags**.

2. Change the *OS* setting to "**js**" and the *Arch* setting to "**wasm**".

3. Click **OK**.

Figure 2: GoLand Wasm Setup pic3


wasmGraph2 [~/git_repos/src/github.com/justinclift/wasmGraph2] - .../main.go [wasmGraph2] - GoLand
File Edit View Navigate Code Refactor Run Tools VCS Window Help
wasmGraph2 main.go
Add Configuration...
Project
wasmGraph2 ~/git_repos/src/github.co
.gitignore
Caddyfile
index.html
LICENSE
main.go
README.md
wasm_exec.js
External Libraries
Scratches and Consoles
main.go
6: TODO
Docker
Terminal
Event Log

```go
//Wasming
// compile: GOOS=js GOARCH=wasm go build -o main.wasm ./main.go
package main

import "syscall/js"

func main() {
	doc := js.Global().Get("document")
	doc.Set("foo", "bar")
	// etc
}
```

Package: js

func Global() Value

Global returns the JavaScript global object, usually "window" or "global".

`Global` on godoc.org

"*syscall/js*" should now be recognised.
Syntax highlighting, documentation,
(etc) should now work too.

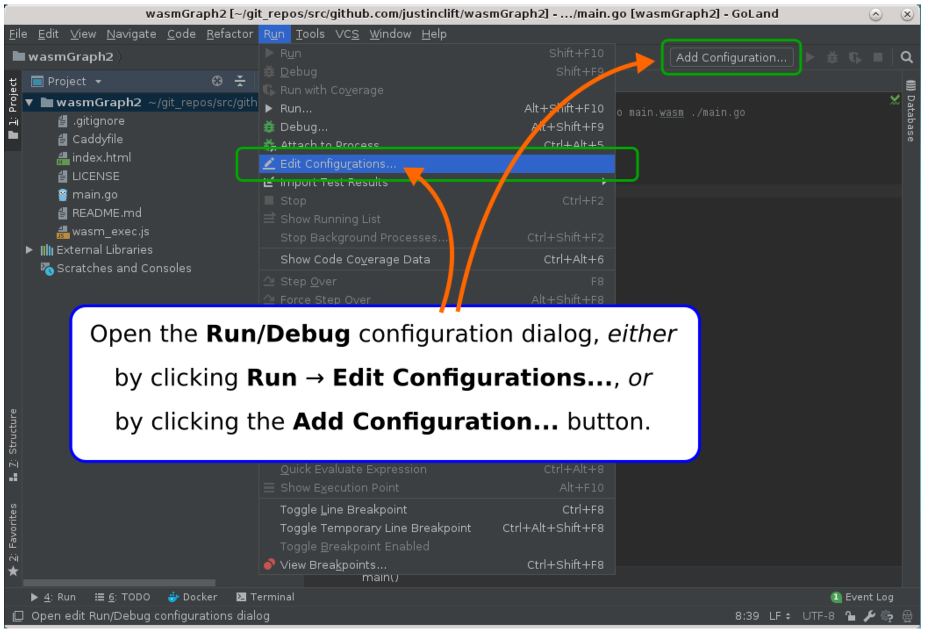
Edit the project and application settings with the spanner icon in the status bar (24 minutes ago)   8:17 LF UTF-8

Figure 3: GoLand Wasm Setup pic4

Configuring Run/Debug settings

With the initial project settings changed, you'll probably want to configure the Run/Debug settings next.

That will let you recompile the .wasm file by just launching `Run` (Shift+F10 on Linux).


wasmGraph2 [~/git_repos/src/github.com/justinclift/wasmGraph2] - .../main.go [wasmGraph2] - GoLand
File Edit View Navigate Code Refactor Run Tools VCS Window Help
wasmGraph2
Project
wasmGraph2 ~/git_repos/src/gith
.gitignore
Caddyfile
index.html
LICENSE
main.go
README.md
wasm_exec.js
External Libraries
Scratches and Consoles
Run Shift+F10
Debug Shift+F9
Run with Coverage
Run... Alt+Shift+F10
Debug... Alt+Shift+F9
Attach to Process... Ctrl+Alt+5
Edit Configurations...
Import Test Results
Stop Ctrl+F2
Show Running List
Stop Background Processes... Ctrl+Shift+F2
Show Code Coverage Data Ctrl+Alt+6
Step Over F8
Force Step Over Alt+Shift+F8
Quick Evaluate Expression Ctrl+Alt+8
Show Execution Point Alt+F10
Toggle Line Breakpoint Ctrl+F8
Toggle Temporary Line Breakpoint Ctrl+Alt+Shift+F8
Toggle Breakpoint Enabled
View Breakpoints... Ctrl+Shift+F8
Add Configuration...
Database
1: Project
7: Structure
2: Favorites
4: Run
6: TODO
Docker
Terminal
Event Log
8:39 LF UTF-8

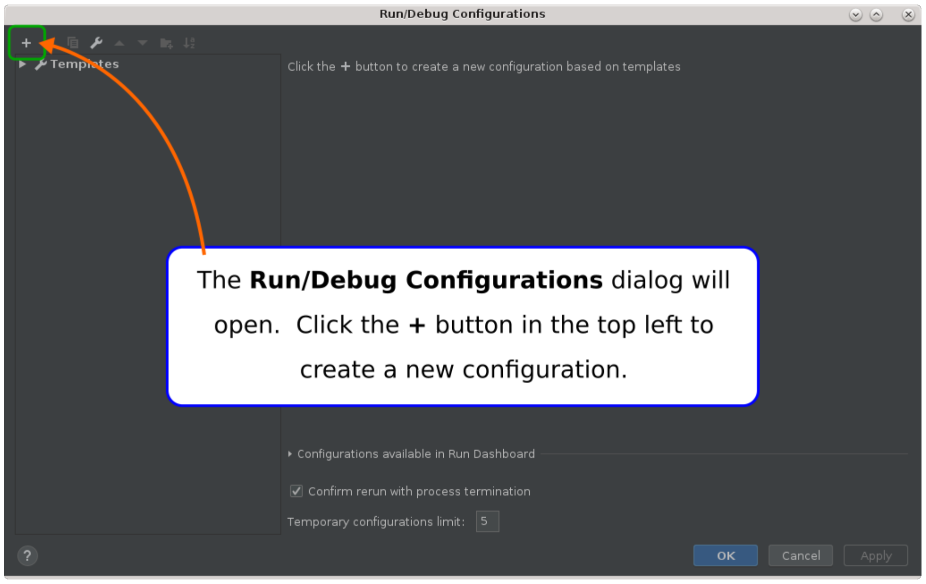
Open the **Run/Debug** configuration dialog, *either* by clicking **Run → Edit Configurations...**, *or* by clicking the **Add Configuration...** button.

Open edit Run/Debug configurations dialog

Figure 4: GoLand Wasm Build pic2

Finished, your GoLand setup should now be complete.

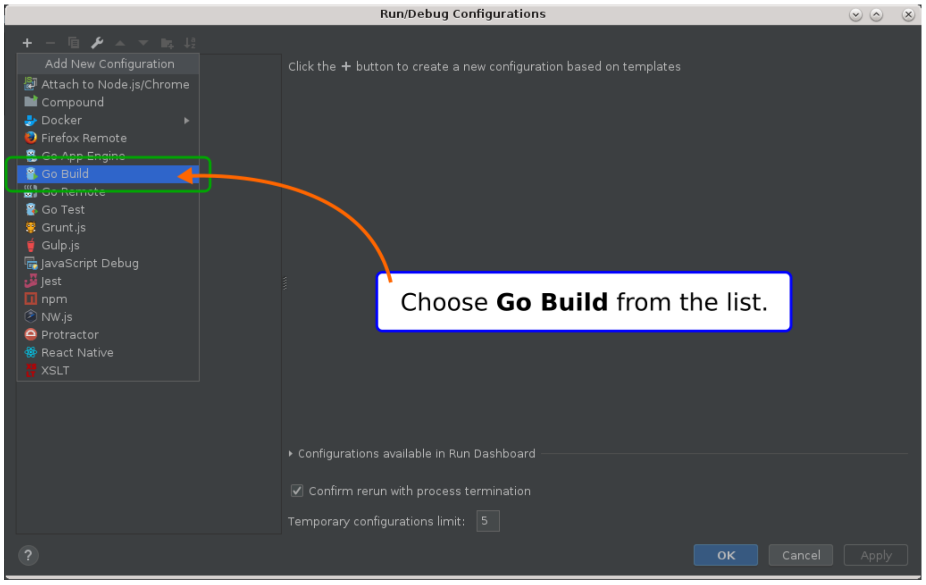
← Back to the main WebAssembly page.


Run/Debug Configurations
Templates
Configurations available in Run Dashboard
Confirm rerun with process termination
Temporary configurations limit: 5
OK
Cancel
Apply

Click the + button to create a new configuration based on templates

The **Run/Debug Configurations** dialog will open. Click the **+** button in the top left to create a new configuration.

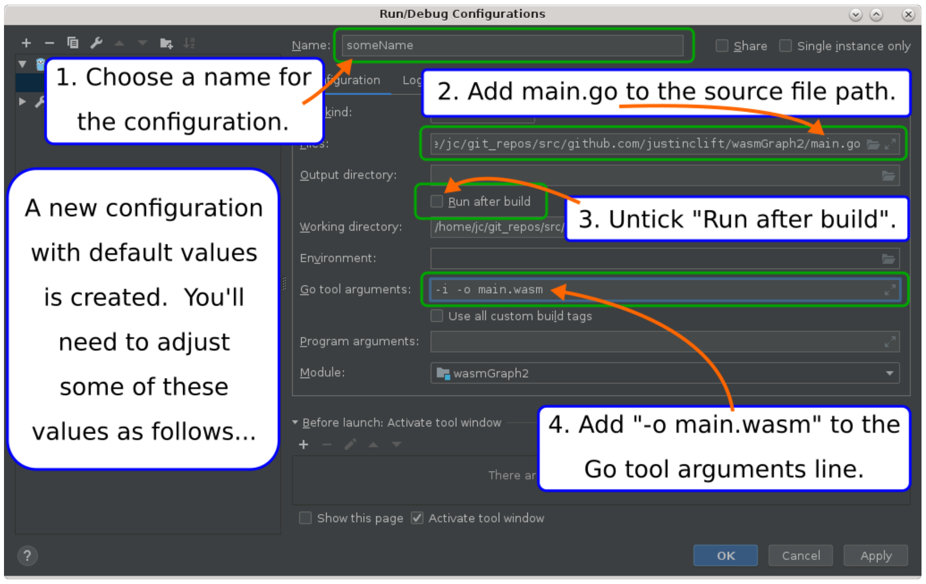
Figure 5: GoLand Wasm Build pic3



Click the + button to create a new configuration based on templates

Choose **Go Build** from the list.

Figure 6: GoLand Wasm Build pic4



1. Choose a name for the configuration.

2. Add main.go to the source file path.

3. Untick "Run after build".

4. Add "-o main.wasm" to the Go tool arguments line.

A new configuration with default values is created. You'll need to adjust some of these values as follows...

Name: someName

Files: ɜ/jc/git_repos/src/github.com/justinclift/wasmGraph2/main.go

Output directory:

Working directory: /home/jc/git_repos/src/

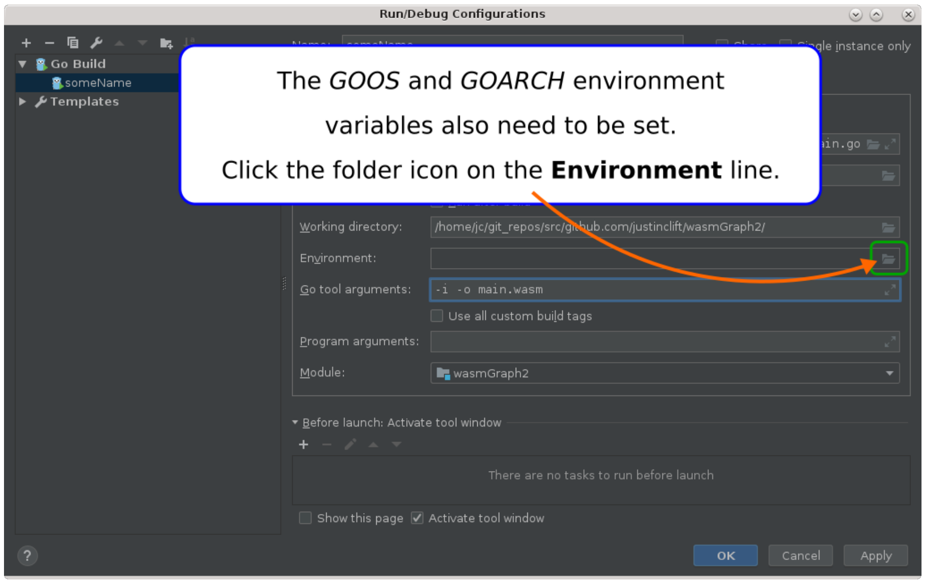
Environment:

Go tool arguments: `-i -o main.wasm`

Program arguments:

Module: wasmGraph2

Figure 7: GoLand Wasm Build pic5



The *GOOS* and *GOARCH* environment variables also need to be set.
Click the folder icon on the **Environment** line.

Working directory: /home/jc/git_repos/src/github.com/justinclift/wasmGraph2/

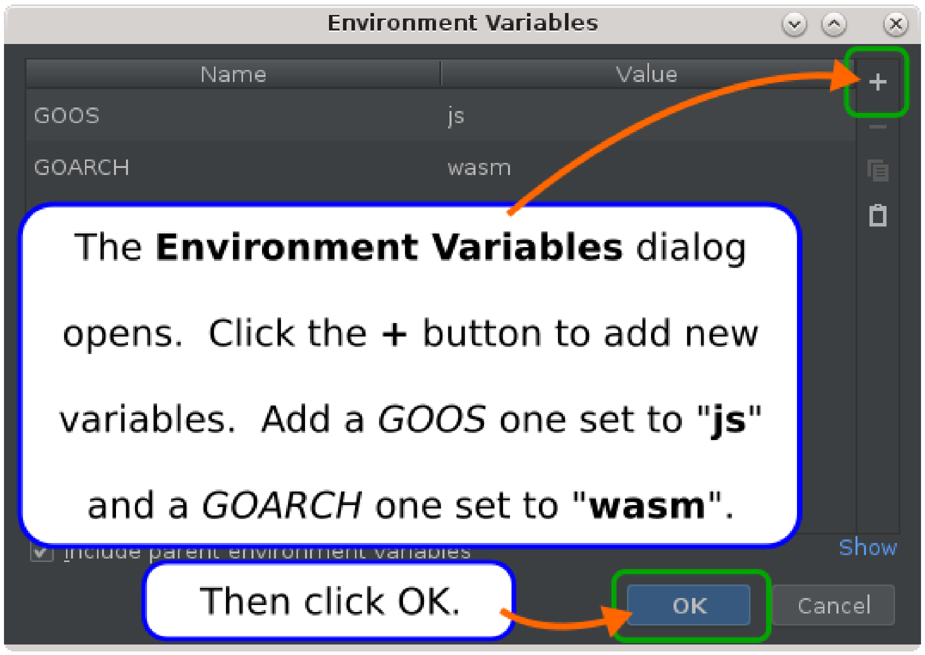
Environment:

Go tool arguments: `-i -o main.wasm`

Program arguments:

Module: wasmGraph2

Figure 8: GoLand Wasm Build pic6



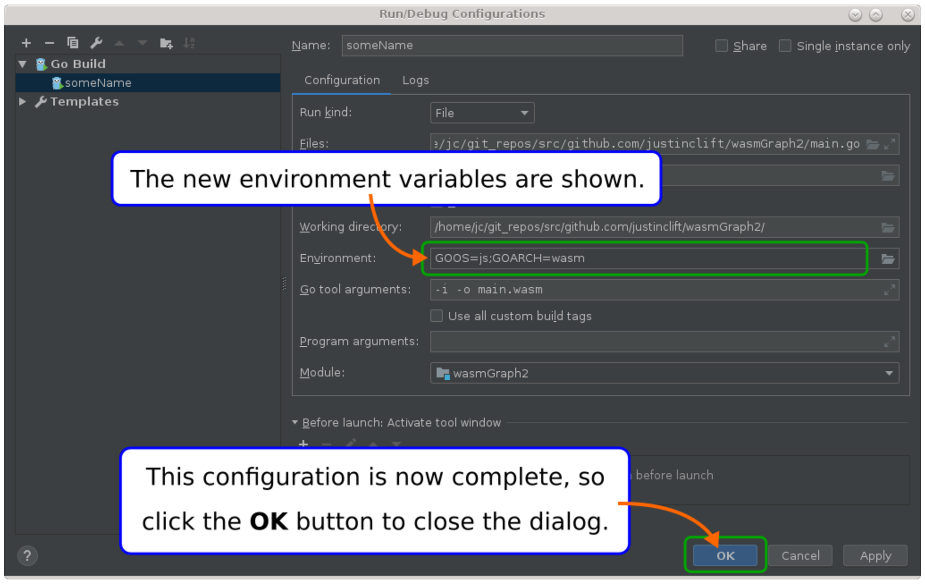
Name	Value
GOOS	js
GOARCH	wasm

The **Environment Variables** dialog opens. Click the **+** button to add new variables. Add a *GOOS* one set to "**js**" and a *GOARCH* one set to "**wasm**".

Include parent environment variables

Then click OK.

Figure 9: GoLand Wasm Build pic7



Name: someName

Share Single instance only

Run kind: File

Files: ɜ/jc/git_repos/src/github.com/justinclift/wasmGraph2/main.go

The new environment variables are shown.

Working directory: /home/jc/git_repos/src/github.com/justinclift/wasmGraph2/

Environment: GOOS=js;GOARCH=wasm

Go tool arguments: -i -o main.wasm

Use all custom build tags

Program arguments:

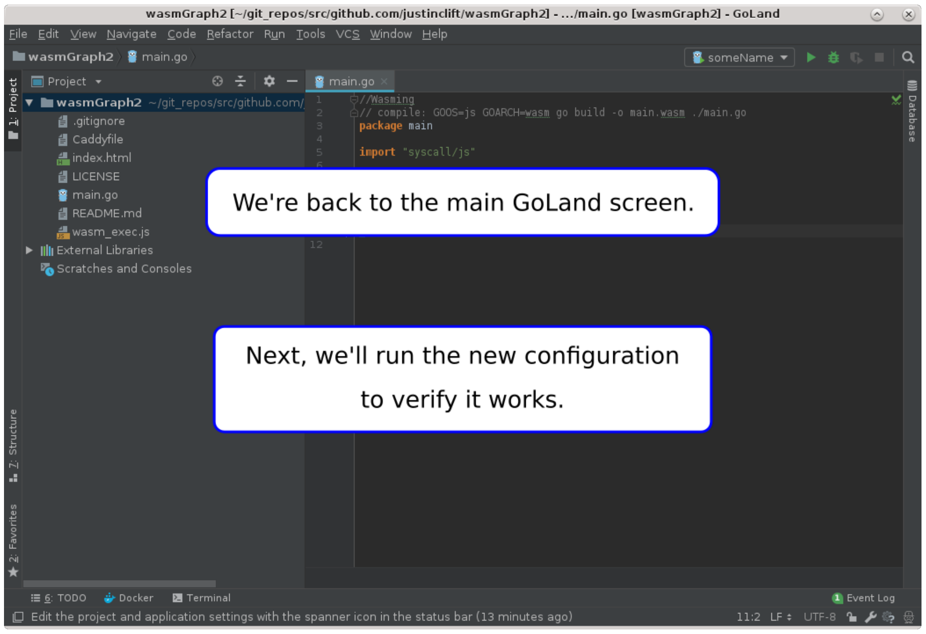
Module: wasmGraph2

Before launch: Activate tool window

before launch

This configuration is now complete, so click the **OK** button to close the dialog.

Figure 10: GoLand Wasm Build pic8


wasmGraph2 [~/git_repos/src/github.com/justinclift/wasmGraph2] - .../main.go [wasmGraph2] - GoLand
File Edit View Navigate Code Refactor Run Tools VCS Window Help
wasmGraph2 main.go
someName
Project
wasmGraph2 ~/git_repos/src/github.com/
.gitignore
Caddyfile
index.html
LICENSE
main.go
README.md
wasm_exec.js
External Libraries
Scratches and Consoles
6: TODO Docker Terminal
Event Log
11:2 LF UTF-8

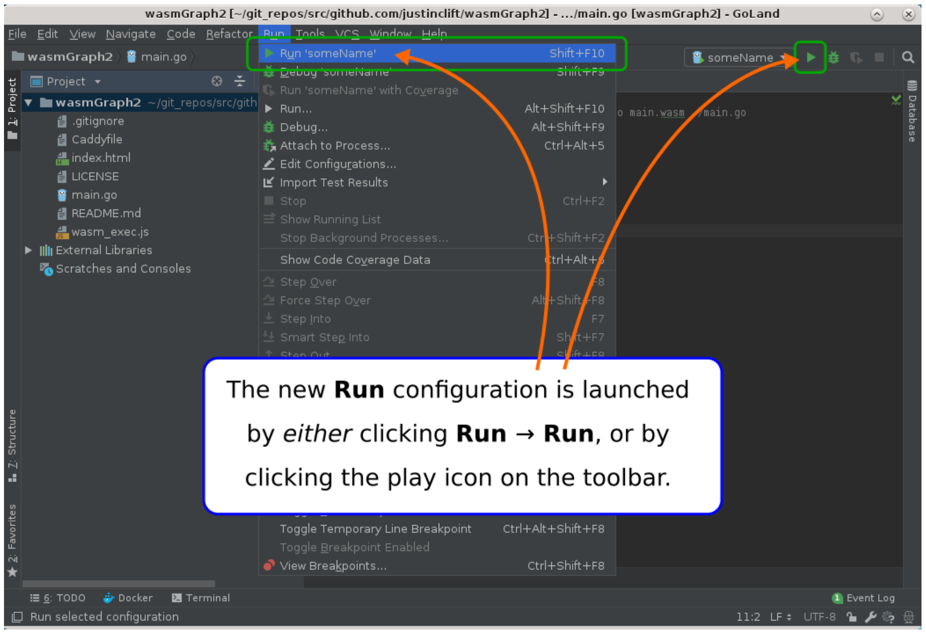
```
//Wasming
// compile: GOOS=js GOARCH=wasm go build -o main.wasm ./main.go
package main

import "syscall/js"
```

We're back to the main GoLand screen.

Next, we'll run the new configuration to verify it works.

Edit the project and application settings with the spanner icon in the status bar (13 minutes ago)

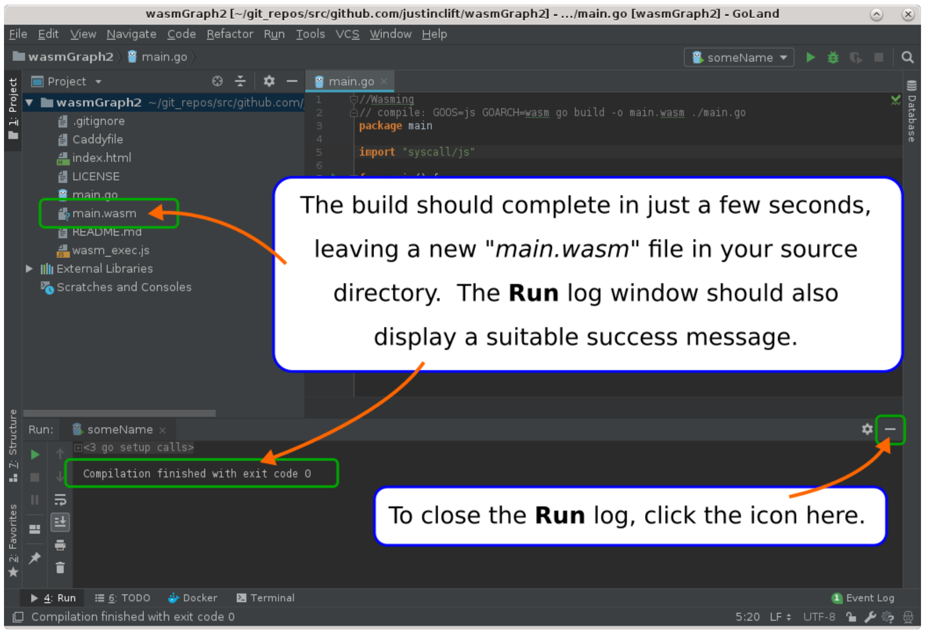
Figure 11: GoLand Wasm Build pic9


wasmGraph2 [~/git_repos/src/github.com/justinclift/wasmGraph2] - .../main.go [wasmGraph2] - GoLand
File Edit View Navigate Code Refactor Run Tools VCS Window Help
Run 'someName' Shift+F10
Debug 'someName' Shift+F9
Run 'someName' with Coverage
Run... Alt+Shift+F10
Debug... Alt+Shift+F9
Attach to Process... Ctrl+Alt+5
Edit Configurations...
Import Test Results
Stop Ctrl+F2
Show Running List
Stop Background Processes... Ctrl+Shift+F2
Show Code Coverage Data Ctrl+Alt+6
Step Over F8
Force Step Over Alt+Shift+F8
Step Into F7
Smart Step Into Shift+F7
Toggle Temporary Line Breakpoint Ctrl+Alt+Shift+F8
Toggle Breakpoint Enabled
View Breakpoints... Ctrl+Shift+F8
someName
6: TODO Docker Terminal
Event Log
11:2 LF UTF-8

The new **Run** configuration is launched by *either* clicking **Run** → **Run**, or by clicking the play icon on the toolbar.

Run selected configuration

Figure 12: GoLand Wasm Build pic10



```
//Wasming
// compile: GOOS=js GOARCH=wasm go build -o main.wasm ./main.go
package main

import "syscall/js"
```

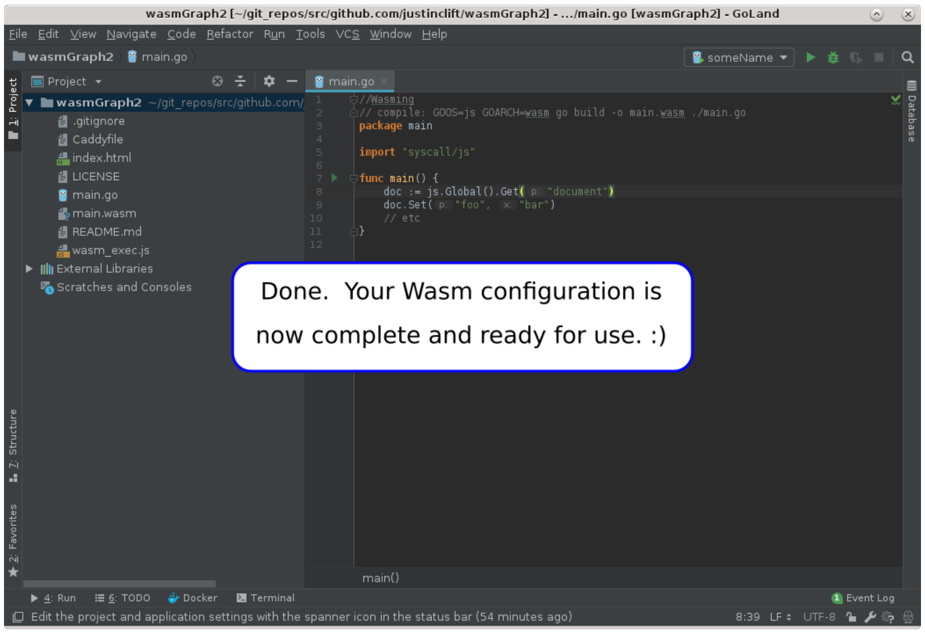
```
<3 go setup calls>
Compilation finished with exit code 0
```

Compilation finished with exit code 0

The build should complete in just a few seconds, leaving a new "*main.wasm*" file in your source directory. The **Run** log window should also display a suitable success message.

To close the **Run** log, click the icon here.

Figure 13: GoLand Wasm Build pic11



```
//Wasming
// compile: GOOS=js GOARCH=wasm go build -o main.wasm ./main.go
package main

import "syscall/js"

func main() {
    doc := js.Global().Get( p: "document")
    doc.Set( p: "foo",  x: "bar")
    // etc
}
```

Edit the project and application settings with the spanner icon in the status bar (54 minutes ago)

Done. Your Wasm configuration is now complete and ready for use. :)

Figure 14: GoLand Wasm Build pic12