

# Building a Godot Addon

by Johannes Ebner  
Structed.me



# whoami

Sr. Production Partner Manager | Xbox

Founder & Gaming Solution Architect | Structed.me



# Past Experiences

- Game Developer & Programmer by trade
- C# Developer, embraced GDScript
- Huge Rider/JetBrains fan
- Worked as a Gaming Solution Architect for Microsoft



# Success Stories & References

## Valheim (Coffee Stain/Piktiv)

- Consulting on how to migrate from Steam P2P to PlayFab Party
- Direct connection with PlayFab Party Team
- Consulting on economics
- Consulting on best best practices



*Valheim* Case Study:  
Piktiv Shares Learnings from  
Integrating Azure PlayFab  
Multiplayer Services

Johannes Ebner  
@struced

Kenneth Jonsson (He/Him)  
CTO, Piktiv



# Success Stories & References

## Warhammer 40.000: Speed Freeks (Caged Element/PLAION)

- Establishing direct connection with PlayFab Party Team
- Review of Multiplayer Architecture
- General online services & multiplayer consulting



# Success Stories & References

## godot-playfab

- Premier PlayFab SDK for Godot Engine
- Used by hit title Dome Keeper, (Bippinbits/Raw Fury)





# GodotCon 2023

- Co-Host
- Responsible for Location

[GodotCon 2023](https://godotengine.org)  
 [\(godotengine.org\)](https://godotengine.org)



Image courtesy of Jackie Codes (<https://linktr.ee/jackiecodes>)



# Building a Godot Addon

And why every game needs one





# What is an Addon?

- Adds on! (duh!)
- Re-usable components:
- Assets
- Code
- Applications
- Libraries

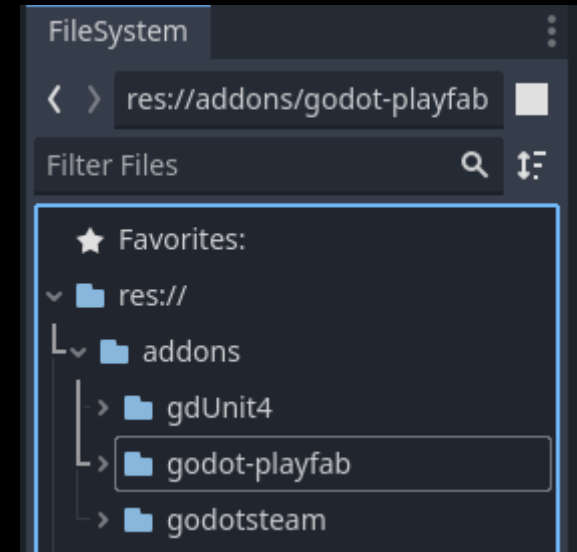


# Types of Addons



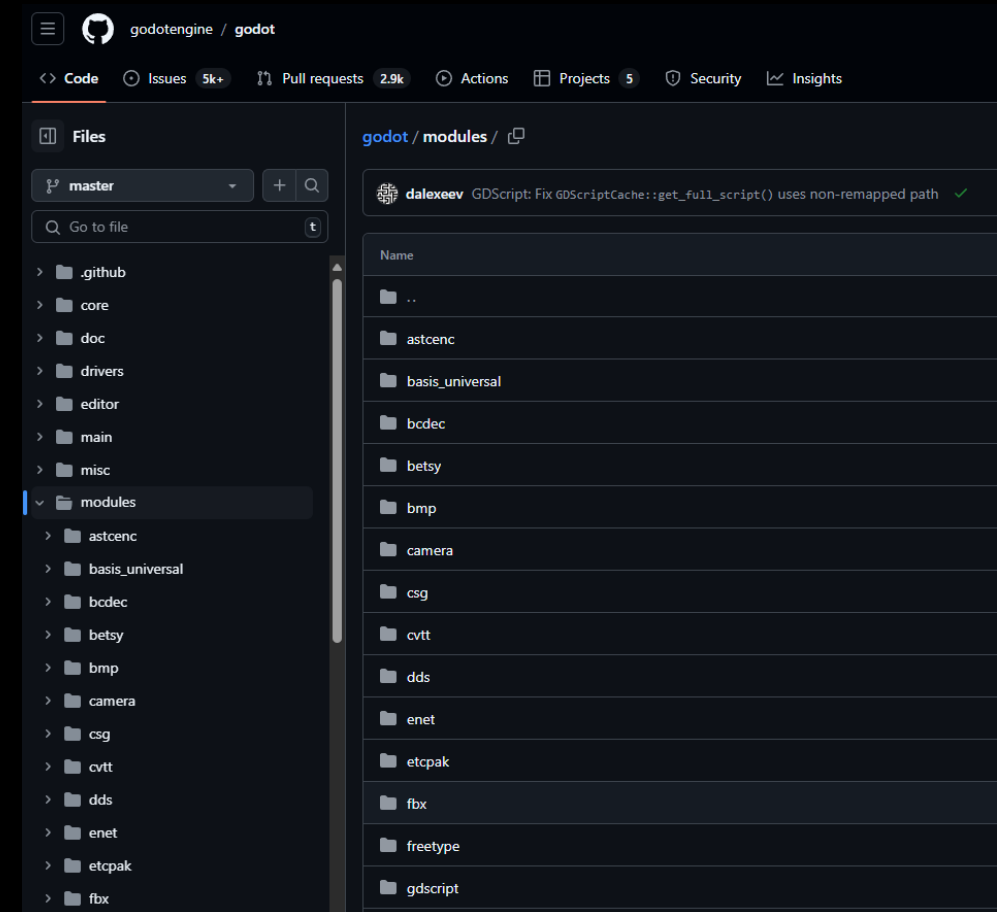
# Plugins

- Regular code files
- Editor & Gameplay extension
- Easiest
- In-Editor



# Modules

- Direct extension of Godot
- Requires source build of Godot
- Can do everything
- Used by Godot itself
- Distribution challenge



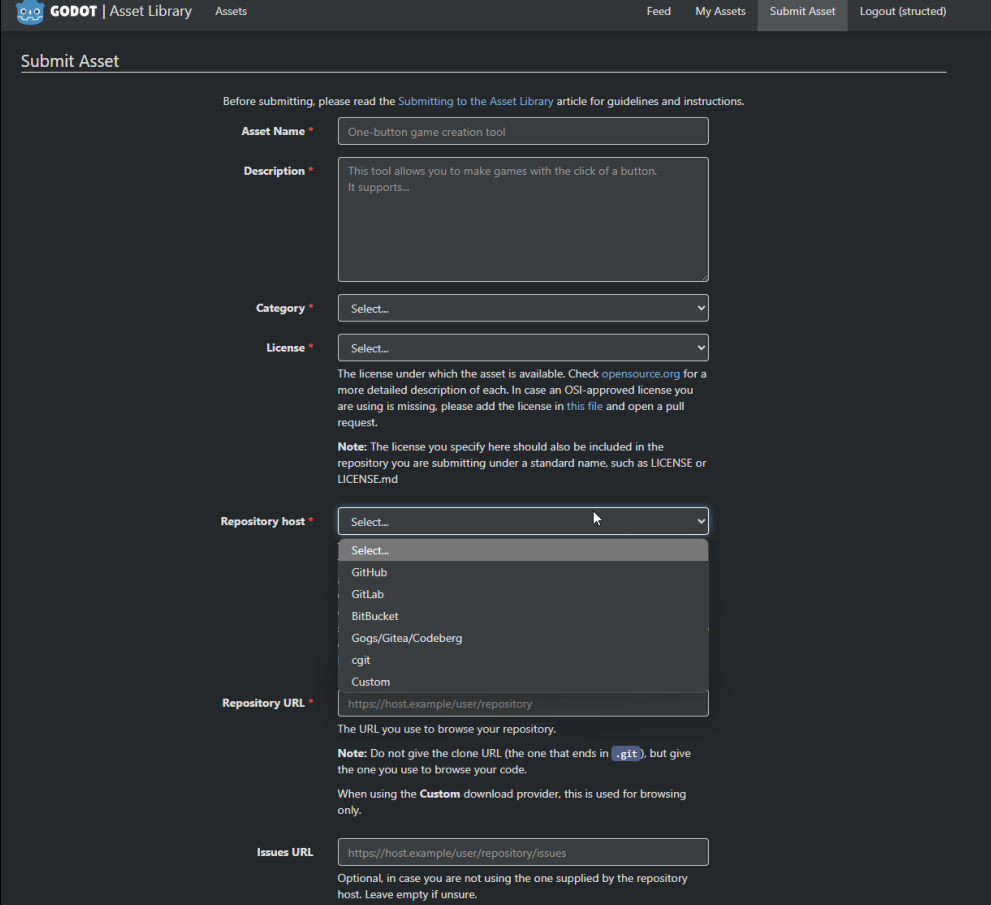
# GDExtension

- Direct extension of Godot
- C++ bindings (official)
- Additional community bindings
- Full API available
- Can do everything
- Always “@tool” mode
- Easy distribution



# Distribution

- AssetLib
- Direct download
- Itch.io etc



The screenshot shows the 'Submit Asset' page on the Godot Asset Library. The page has a dark theme with a top navigation bar containing the Godot logo, 'Asset Library', 'Assets', 'Feed', 'My Assets', 'Submit Asset' (active), and 'Logout (strusted)'. Below the navigation bar, the title 'Submit Asset' is followed by a horizontal line. The main content area contains a form with the following fields and instructions:

- Asset Name \***: A text input field containing 'One-button game creation tool'.
- Description \***: A text area containing 'This tool allows you to make games with the click of a button. It supports...'.
- Category \***: A dropdown menu with 'Select...' selected.
- License \***: A dropdown menu with 'Select...' selected.
- Repository host \***: A dropdown menu with 'Select...' selected. The dropdown list is open, showing options: 'Select...', 'GitHub', 'GitLab', 'BitBucket', 'Gogs/Gitea/Codeberg', 'cgit', and 'Custom'.
- Repository URL \***: A text input field containing 'https://host.example/user/repository'.
- Issues URL**: A text input field containing 'https://host.example/user/repository/issues'.

Instructions and notes on the page include:

- 'Before submitting, please read the Submitting to the Asset Library article for guidelines and instructions.'
- 'The license under which the asset is available. Check [opensource.org](https://opensource.org) for a more detailed description of each. In case an OSI-approved license you are using is missing, please add the license in this file and open a pull request.'
- 'Note: The license you specify here should also be included in the repository you are submitting under a standard name, such as LICENSE or LICENSE.md'
- 'The URL you use to browse your repository.'
- 'Note: Do not give the clone URL (the one that ends in `.git`), but give the one you use to browse your code.'
- 'When using the **Custom** download provider, this is used for browsing only.'
- 'Optional, in case you are not using the one supplied by the repository host. Leave empty if unsure.'



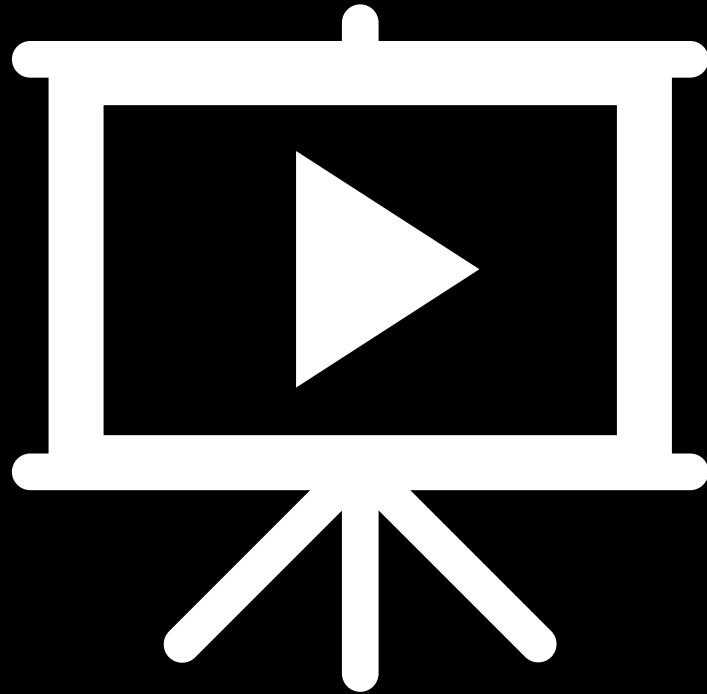
# Creating a Plugin

With GDScript





# Demo Time!



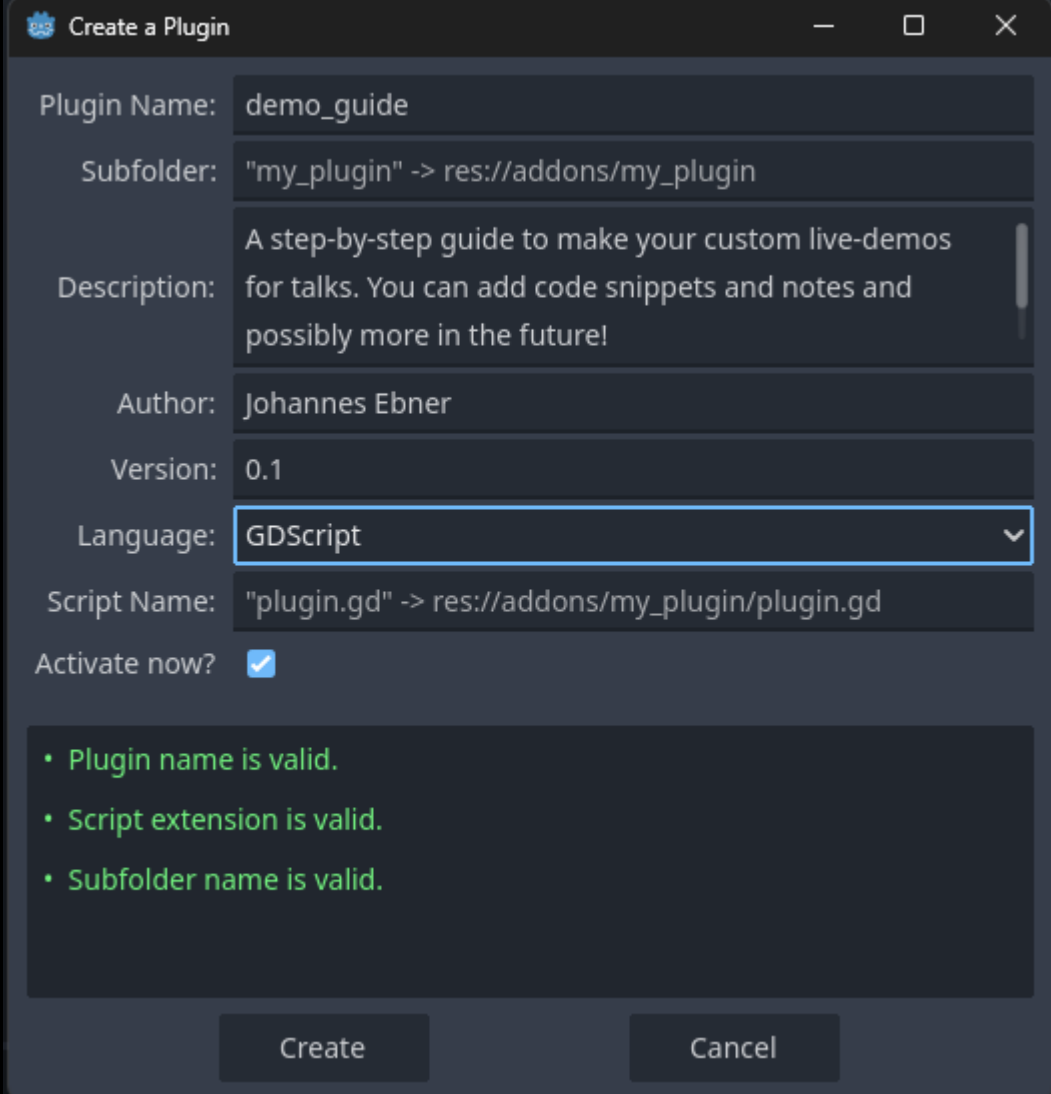
# Recap

Creating a Plugin



# Creating a new Plugin

- Create new Project
- Create new Plugin in Settings



The screenshot shows the 'Create a Plugin' dialog box in Godot Engine. The dialog has a title bar with the Godot logo and the text 'Create a Plugin'. It contains several input fields and a checkbox. The 'Plugin Name' field is filled with 'demo\_guide'. The 'Subfolder' field is filled with '"my\_plugin" -> res://addons/my\_plugin'. The 'Description' field is filled with 'A step-by-step guide to make your custom live-demos for talks. You can add code snippets and notes and possibly more in the future!'. The 'Author' field is filled with 'Johannes Ebner'. The 'Version' field is filled with '0.1'. The 'Language' field is a dropdown menu with 'GDScript' selected. The 'Script Name' field is filled with '"plugin.gd" -> res://addons/my\_plugin/plugin.gd'. The 'Activate now?' checkbox is checked. At the bottom, there are 'Create' and 'Cancel' buttons. Below the input fields, there is a list of validation messages in green text: 'Plugin name is valid.', 'Script extension is valid.', and 'Subfolder name is valid.'

Create a Plugin

Plugin Name: demo\_guide

Subfolder: "my\_plugin" -> res://addons/my\_plugin

Description: A step-by-step guide to make your custom live-demos for talks. You can add code snippets and notes and possibly more in the future!

Author: Johannes Ebner

Version: 0.1

Language: GDScript

Script Name: "plugin.gd" -> res://addons/my\_plugin/plugin.gd

Activate now? ☒

- Plugin name is valid.
- Script extension is valid.
- Subfolder name is valid.

Create Cancel



# plugin.cfg

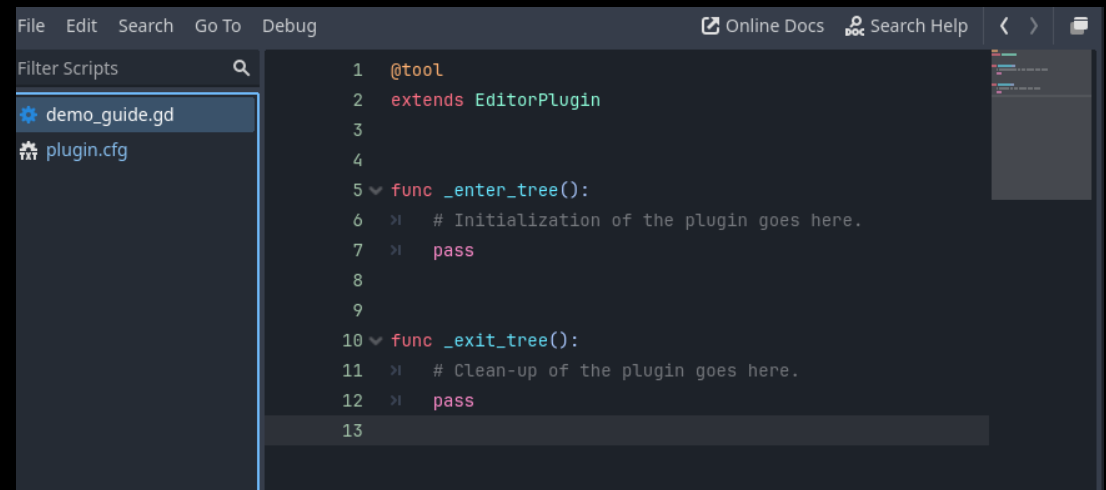
- Metadata
- For AssetLib + Editor

```
1  [plugin]
2
3  name="demo_guide"
4  description="A step-by-step guide to make your custom live-
5  demos for talks. You can add code snippets and notes and
6  possibly more in the future!"
7  author="Johannes Ebner"
8  version="0.1"
9  script="demo_guide.gd"
10
```



# Entry Point

- File name arbitrary
- Defined in plugin.cfg
- Has @tool attribute



The screenshot shows the Godot Engine's code editor interface. On the left, a sidebar titled 'Filter Scripts' contains two entries: 'demo\_guide.gd' with a gear icon and 'plugin.cfg' with a document icon. The main editor area displays the contents of 'plugin.cfg'. The code is as follows:

```
1 @tool
2 extends EditorPlugin
3
4
5 func _enter_tree():
6     # Initialization of the plugin goes here.
7     pass
8
9
10 func _exit_tree():
11     # Clean-up of the plugin goes here.
12     pass
13
```



# Extending the Editor

- The `@tool` attribute
  - Runs in Editor
  - Has access to editor scope
  - Does not inherit
  - Referenced scripts need `@tool`
- Be aware of “quirky” behaviour
  - Refer to [Docs](#)

```
@tool
class_name MyTool
extends Node

@export var resource: MyResource:
    set(new_resource):
        resource = new_resource
        _on_resource_set()

# This will only be called when you create, delete, or paste a resource.
# You will not get an update when tweaking properties of it.
func _on_resource_set():
    print("My resource was set!")
```

```
# Make Your Resource a tool.
@tool
class_name MyResource
extends Resource

@export var property = 1:
    set(new_setting):
        property = new_setting
        # Emit a signal when the property is changed.
        changed.emit()
```



# Extending a Game

- Make sure to not use @tool
- Or check: `Engine.is_editor_hint()`





# Key Takeaways

## Addon Development

- No debugging for @tool
- Clean up! 🧹
- Figuring out versioning
- Testing manually



# Publishing



# Publishing on the Asset Library

- Name
- Description
- License
- Author
- Repo host
- Specific commit hash
- Logo can/should live with code
- Manual review
  - Takes time
  - No comms



Submit Asset

Before submitting, please read the [Submitting to the Asset Library](#) article for guidelines and instructions.

Asset Name \*

Description \*

Category \*

License \*

The license under which the asset is available. Check [opensource.org](#) for a more detailed description of each. In case an OSI-approved license you are using is missing, please add the license in this file and open a pull request.

**Note:** The license you specify here should also be included in the repository you are submitting under a standard name, such as LICENSE or LICENSE.txt

Repository host \*

The site or service hosting your repository.  
The **Custom** download provider can be used to link to premade ZIP archives, such as those uploaded on GitHub Releases. This is useful for **GoNative/GCExtension** add-ons which contain precompiled binaries. This can also be useful when **including submodules** in the ZIP is desired (as submodules are not included in automatically generated ZIPs from GitHub or GitLab).  
If your repository host is missing, you might like to open an [Issue](#) about it.

Repository URL \*

The URL you use to browse your repository.  
**Note:** Do not give the clone URL (the one that ends in `.git`), but give the one you use to browse your code.  
When using the **Custom** download provider, this is used for browsing only.

Issues URL

Optional, in case you are not using the one supplied by the repository host. Leave empty if unsure.

Minimum Godot version \*

The oldest version of Godot the asset works with.  
In the Godot editor and project manager, the asset will be displayed for all versions that are compatible (within the same major version).  
For example:  
• If your asset works with Godot 4.1 and later, use 4.1 as the minimum Godot version.  
• If your asset works with Godot 3.5 but not Godot 4.0 or later, use 3.5 as the minimum Godot version.

Asset Version \*

Download Commit/URL \*

The commit hash that should be downloaded. Expects 40 or 64 hexadecimal digits fully specifying a Git commit.  
When using the **Custom** download provider, this must be set to the full download URL instead of a commit hash.

Icon URL \*

# .gitattributes

- Define shared git settings
- Also affects archive download
- Removes “junk” from distribution

```
# Normalize EOL for all files that Git considers text files.
* text=auto eol=lf

# IDE
/.idea                export-ignore
/.vscode              export-ignore

# Exclude other addons
/addons/another-addon export-ignore

# GDUnit - Godot Unit Test Framework
/addons/gdUnit4        export-ignore

# Project items
/addons/demo_guide/test export-ignore # Test project
/raw_assets            export-ignore # This is where I keep my raw assets
demo-scene.gif         export-ignore # Showing off the demo scene.

# CI/CD
GitVersion.yml         export-ignore
/.github               export-ignore
.asset-template.json.hb export-ignore # Template for AssetLib publishing
```



# Automating Publishing

```
prerelease:
  if: "github.event.head_commit.message != 'Release preparation: storing Version an
  name: Preparing release
  runs-on: ubuntu-latest
  outputs: <3 keys>
  steps:
    - uses: actions/checkout@v3
      with: <2 keys>

    - name: Calculate version
      id: calculate_version
      uses: mathieudutour/github-tag-action@v6.1
      with: <5 keys>

    - name: "
      run: |
        CHAN
      then
        fi
```

## Prepare

- Calculate version
- Update version in plugin.cfg
- Write changelog
- Commit



```
release:
  # Create release only after the version was bumped and release notes added:
  if: "github.event.head_commit.message = 'Release preparation: storing Version an
  runs-on: ubuntu-latest

  steps:
    - uses: actions/checkout@v3
      with: <2 keys>

    - name: Create version
      id: create_version
      uses: mathieudutour/github-tag-action@v6.1
      with: <5 keys>
      if: st

    - name:
      uses:
      with:
      if: st
```

## Release

- Calculate version
- Tag
- Upload artifact
- Create release
- Submit AssetLib update



# AssetLib Update

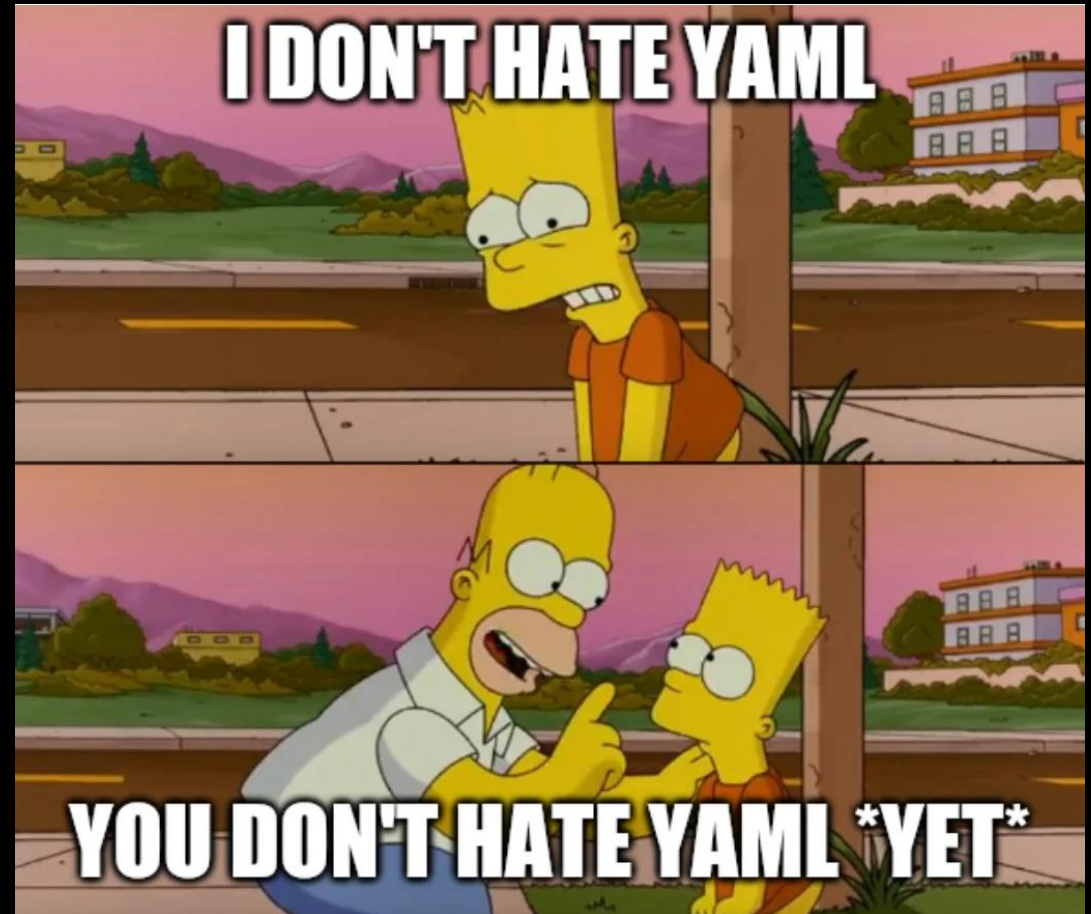
- GH Action
- Made by Dennis Ploeger (deep-entertainment)
- Updates AssetLib via HTTP

```
.asset-template.json.hb x
1
2 // This is a Handlebars template that will be used to generate the final asset information.
3 // Replace the 'icon_url' with the static URL of your own icon.
4
5 {
6   "version_string": "{{ env.version }}",
7   "download_commit": "{{ env.GITHUB_SHA }}",
8   "browse_url": "{{ context.repository.html_url }}",
9   "issues_url": "{{ context.repository.html_url }}/issues",
10  "icon_url": "https://raw.githubusercontent.com/Structed/godot-playfab/main/addons/godot-playfab/icon.png"
11 }
```



# Key Takeaways Publishing

- YAML sucks
- Iterating takes a lot of time
- Manual approval required
- Can be very slow
- Coordinate for bigger release
- Latest approved always atop





# GDExtension



# Why

- Binding external libs
- Performance
- Use another language

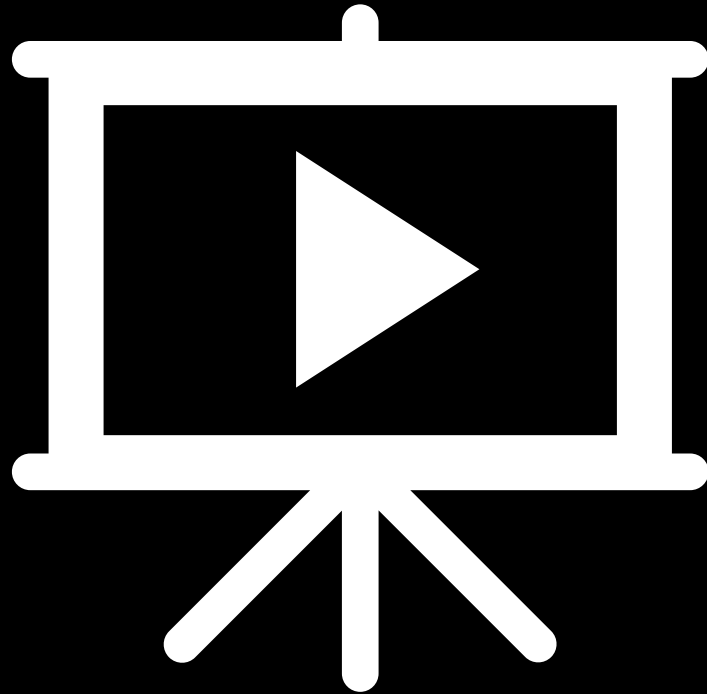


# Prerequisites

- godot-cpp
- Compiler toolchain
- Python & pip
- Scons
- Godot > 4.1



# Demo Time!



# Recap

GDExtension



# gdexample

- Your extension code
- Access Godot API

```
#include "gdexample.h"
#include <godot_cpp/core/class_db.hpp>

using namespace godot;

void GDExample::_bind_methods() {
}

GDExample::GDExample() {
    // Initialize any variables here.
    time_passed = 0.0;
}

GDExample::~GDExample() {
    // Add your cleanup here.
}

void GDExample::_process(double delta) {
    time_passed += delta;



    Vector2 new_position = Vector2(10.0 + (10.0 * sin(time_passed * 2.0)), 10.0 + (10.0 * cos(time_passed * 1.5)));

    set_position(new_position);
}
```



# register\_types

- Defines

-  Entry Point
-  Initialization
-  De-Initialisation

```
#include "register_types.h"

#include "gdexample.h"

#include <gdeextension_interface.h>
#include <godot_cpp/core/defs.hpp>
#include <godot_cpp/godot.hpp>

using namespace godot;

void initialize_example_module(ModuleInitializationLevel p_level) {
    if (p_level != MODULE_INITIALIZATION_LEVEL_SCENE) {
        return;
    }

    GDREGISTER_CLASS(GDExample);
}

void uninitialize_example_module(ModuleInitializationLevel p_level) {
    if (p_level != MODULE_INITIALIZATION_LEVEL_SCENE) {
        return;
    }
}

extern "C" {
    // Initialization.
    GDEExtensionBool GDE_EXPORT example_library_init(
        GDEExtensionInterfaceGetProcAddress p_get_proc_address,
        const GDEExtensionClassLibraryPtr p_library,
        GDEExtensionInitialization *r_initialization) {

        godot::GDEExtensionBinding::InitObject init_obj(p_get_proc_address, p_library, r_initialization);

        init_obj.register_initializer(initialize_example_module);
        init_obj.register_terminator(uninitialize_example_module);
        init_obj.set_minimum_library_initialization_level(MODULE_INITIALIZATION_LEVEL_SCENE);

        return init_obj.init();
    }
}
```





# gdexample.gdextension

- Metadata for Loading in Editor
- Defines
  - Entry Point
  - Minimum Godot version
  - Reloadable?
  - Library paths (per platform & env)
  - Extra dependencies

```
[configuration]

entry_symbol = "example_library_init"
compatibility_minimum = "4.3"
reloadable = true


[libraries]

macos.debug = "res://addons/demo_gdextension/addons/demo_gdextension/bin/libgdexample.macos.template_debug.framework"
macos.release = "res://addons/demo_gdextension/addons/demo_gdextension/bin/libgdexample.macos.template_release.framework"
ios.debug = "res://addons/demo_gdextension/addons/demo_gdextension/bin/libgdexample.ios.template_debug.xcframework"
ios.release = "res://addons/demo_gdextension/bin/libgdexample.ios.template_release.xcframework"
windows.debug.x86_32 = "res://addons/demo_gdextension/bin/libgdexample.windows.template_debug.x86_32.dll"
windows.release.x86_32 = "res://addons/demo_gdextension/bin/libgdexample.windows.template_release.x86_32.dll"
windows.debug.x86_64 = "res://addons/demo_gdextension/bin/libgdexample.windows.template_debug.x86_64.dll"
windows.release.x86_64 = "res://addons/demo_gdextension/bin/libgdexample.windows.template_release.x86_64.dll"
linux.debug.x86_64 = "res://addons/demo_gdextension/bin/libgdexample.linux.template_debug.x86_64.so"
linux.release.x86_64 = "res://addons/demo_gdextension/bin/libgdexample.linux.template_release.x86_64.so"
linux.debug.arm64 = "res://addons/demo_gdextension/bin/libgdexample.linux.template_debug.arm64.so"
linux.release.arm64 = "res://addons/demo_gdextension/bin/libgdexample.linux.template_release.arm64.so"
linux.debug.rv64 = "res://addons/demo_gdextension/bin/libgdexample.linux.template_debug.rv64.so"
linux.release.rv64 = "res://addons/demo_gdextension/bin/libgdexample.linux.template_release.rv64.so"
android.debug.x86_64 = "res://addons/demo_gdextension/bin/libgdexample.android.template_debug.x86_64.so"
android.release.x86_64 = "res://addons/demo_gdextension/bin/libgdexample.android.template_release.x86_64.so"
android.debug.arm64 = "res://addons/demo_gdextension/bin/libgdexample.android.template_debug.arm64.so"
android.release.arm64 = "res://addons/demo_gdextension/bin/libgdexample.android.template_release.arm64.so"

[dependencies]
ios.debug = {
    "res://addons/demo_gdextension/bin/libgodot-cpp.ios.template_debug.xcframework": ""
}
ios.release = {
    "res://addons/demo_gdextension/bin/libgodot-cpp.ios.template_release.xcframework": ""
}
```



# Compiling

- `scons platform=<platform>`
  - Builds binaries
  - Drops binaries in addon dir
- SConstruct: Build script
-  With Debugging:
  - `scons dev_build=yes debug_symbols=yes`



# GDExtension

New in 4.4

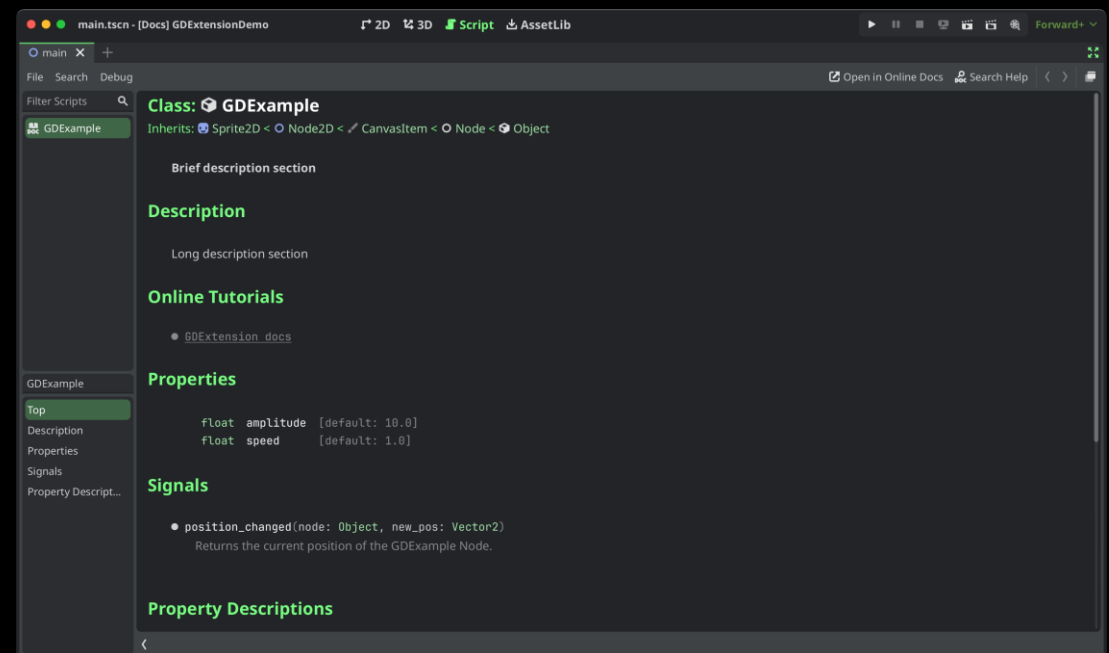
- Expose runtime methods
- Can declare virtual methods



# Key Takeaways GDExtension

- Full access to Godot API
  - i.e. `_process()`
- Do not use Visual Studio Preview!
  - Behaves as if no compiler present
- Create Docs!
  - Create stub XML files
  - Fill out with your docs
  - Include in build via Scons
  - [GDExtension documentation system](#)
- Debugging:
  - `scons dev_build=yes debug_symbols=yes`
  - [Debug c++ GDExtension](#)

```
scratch.log x
1 scons
2 Auto-detected 16 CPU cores available for build parallelism. Using 15 cores by default. You can override it
  with the -j argument.
3 Building for architecture x86_64 on platform windows
4 ['src\\Log.cpp', 'src\\NetworkStateManager.cpp', 'src\\PlayFabHelper.cpp', 'src\\my_node.cpp',
  'src\\PartySampleNetworkCommon\\lib\\PlayFabManager.cpp', 'src\\PartySampleNetworkCommon\\lib\\pch.cpp']
5 Generating godot-cpp/gen/include/godot_cpp/core/ext_wrappers.gen.inc ...
6 Built-in type config: float_64
7 Compiling godot-cpp/src/core/memory.cpp ...
8 Compiling shared src\\Log.cpp ...
9 scons: ** [godot-cpp/src/core/memory.windows.template_debug.x86_64.o] The system cannot find the file
  specified
```



# Thanks

- Dennis Ploeger (deep-entertainment) for the AssetLib action and general help
- Bitbrain for help with GitHub Actions
- Enrico Barbieri Cavallini (EnricoBC) for his support on GDExtension
- Patrick Exner (paddy\_exe) for his work with docs on addons and GDExtension
- And of course, all the maintainers of Godot!



# Links

- @tool: [Running code in the editor](#)
  - [Plugin Demos](#)
  - [My Workflow Examples](#)
  - [Create Release Archive](#)
- [Structed/jb-gamedev-days-24: Supporting project for this talk](#)



# Contact

✉ [Johannes@structed.me](mailto:Johannes@structed.me)



[Structed](#)



[johannesebner](#)



[Structed](#)



[Structed](#)

