



A New **Developer** **Documentation** Platform





Why Change? (1/4)

Let's talk about the Wiki...

- **Not the best experience**
Developers are not exactly excited about it.
- **Navigation & structure needs rethinking**
Important content is elusive, we're not guiding people to it
- **Disconnected from the code**
Context switch required



Why Change? (2/4)

Let's talk about the Wiki...

- **Closed platform**
No real community involvement, no feedback loops.
- **Better writing experiences/tools available**
WYSIWYG, collaborative editing, review changes, ... – HackMD :)



Why Change? (3/4)

We're not good at high level technical documentation.





Why Change? (4/4)

Developers don't really use the documentation either.



Can we establish a new **developer**
documentation **culture**?

Well, others did!

[← All posts](#)

How Google, Twitter, and Spotify built a culture of documentation



Niklas Begley



Sep 7, 2021

Many technical problems ultimately turn out to be people problems, and a lack of good documentation is no exception. Writing and maintaining documentation is a habit that



Twitter, Google, Spotify

Before

- Technical documentation identified as **big problem**
- **Everybody's problem, but nobody's job**
- **Technical writers didn't solve it**
Jumped from project to project, docs outdated quickly.



Twitter, Google, Spotify

After

- **Greatly improved documentation culture**
Technical documentation **used and updated all the time** - by engineers!
- **Technical writers help & empower**
Maintain docs infrastructure, make strategic decisions.



How did they do it?

- **Culture of docs**
Documentation sprints, education, lead by example
- **Standardize & centralize**
Common platforms, templates



How did they do it?

- **Feedback loops**
Easy bug reporting for docs, "Was this page useful?", ...
- **Keep it simple**





How did they do it?

Empower developers:
"Fiercely optimize for the engineer!"

Docs as Code



Docs as Code



Docs as Code


Treat documentation like code

Version control, collaboration, and automation

- Simple markup language
- Close to the code
- Pull requests, versioning, branching
- Forge integration (Gitea, Github, etc.)
- Continuous delivery, automated checks

Sounds Familiar?

Blender 4.0 Manual



Search docs

GETTING STARTED

- About Blender
- Installing Blender
- Configuring Blender
- Help System

SECTIONS

- User Interface
- Editors
- Scenes & Objects
- Modeling

Sculpting & Painting

- Introduction

🏠 / [Sculpting & Painting](#) / [Brush](#) / Brushes

Brushes

Reference

Mode: `Sculpt Mode`

Panel: `Sidebar ▶ Tools ▶ Brushes`

For painting/sculpting modes each brush type is exposed as a tool in the toolbar. The brush on the other hand is a saved preset of all the brush settings, including a name and thumbnail.

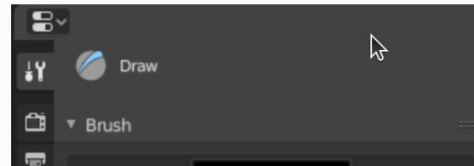
All these settings can be found and changed here in the tool setting (brush, texture, stroke, falloff & cursor).

Brushes

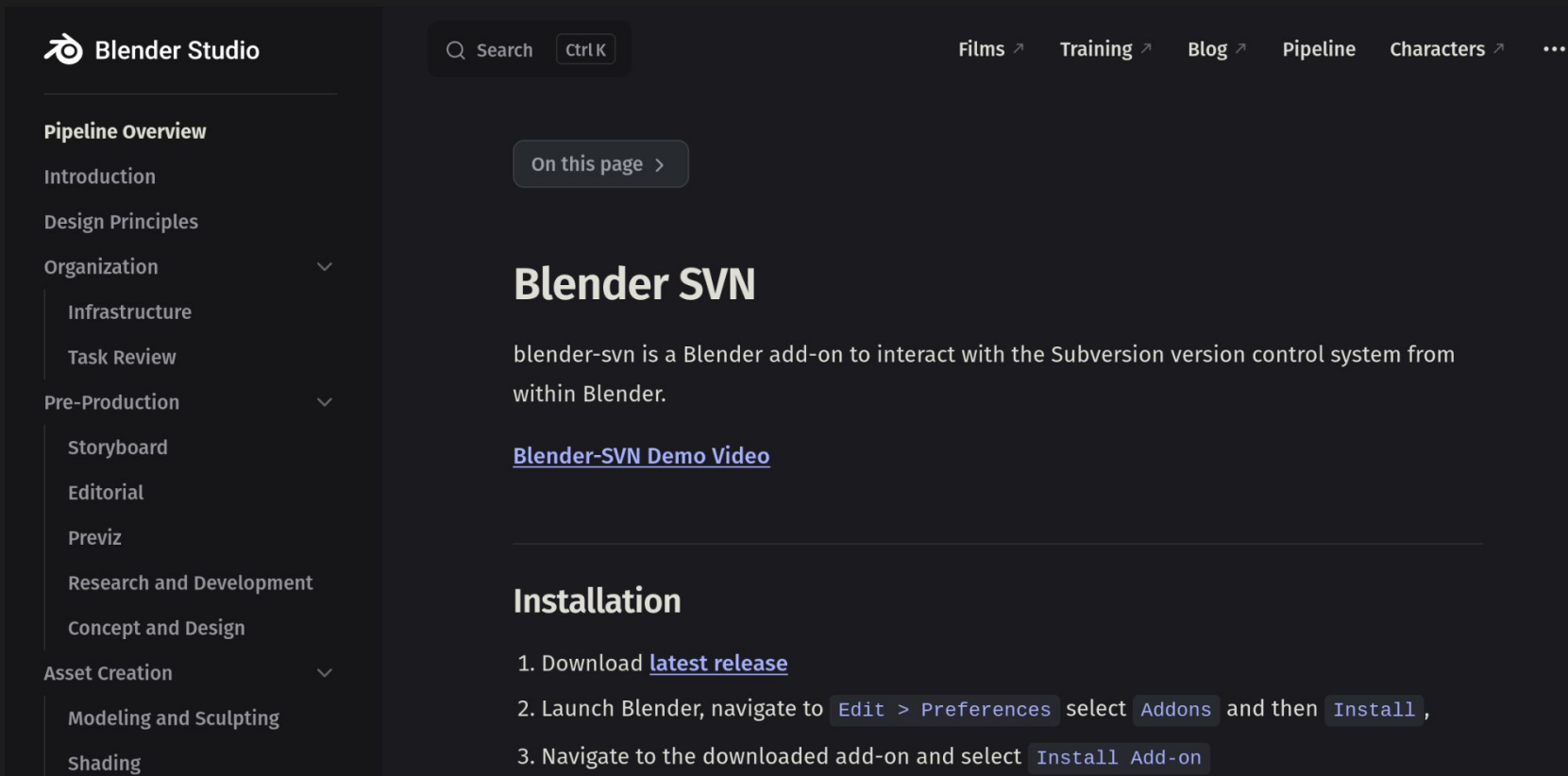
Clicking on the brush thumbnail will open the [Data-Block Menu](#) to select a brush.

Add Brush (Duplicate icon)

When you add a brush, the new brush is



Sounds Familiar?



The screenshot shows the Blender Studio website interface. At the top left is the Blender Studio logo. To its right is a search bar with the text "Search" and a "Ctrl K" button. Further right are navigation links for "Films", "Training", "Blog", "Pipeline", "Characters", and a menu icon. The left sidebar contains a "Pipeline Overview" section with a list of sub-sections: "Introduction", "Design Principles", "Organization" (with a dropdown arrow), "Pre-Production" (with a dropdown arrow), and "Asset Creation" (with a dropdown arrow). Under "Organization" are "Infrastructure" and "Task Review". Under "Pre-Production" are "Storyboard", "Editorial", and "Previz". Under "Asset Creation" are "Modeling and Sculpting" and "Shading". A button labeled "On this page >" is positioned above the main content area. The main content area features a large heading "Blender SVN", a paragraph describing it as a Blender add-on for Subversion, and a link to a "Blender-SVN Demo Video". Below this is a section titled "Installation" with a three-step list: 1. Download [latest release](#), 2. Launch Blender, navigate to `Edit > Preferences`, select `Addons` and then `Install`, 3. Navigate to the downloaded add-on and select `Install Add-on`.

Blender Studio

Search Ctrl K

Films ↗ Training ↗ Blog ↗ Pipeline Characters ↗ ...

Pipeline Overview

Introduction

Design Principles

Organization ▾

- Infrastructure
- Task Review

Pre-Production ▾

- Storyboard
- Editorial
- Previz

Research and Development

Concept and Design

Asset Creation ▾

- Modeling and Sculpting
- Shading

On this page >

Blender SVN


blender-svn is a Blender add-on to interact with the Subversion version control system from within Blender.


[Blender-SVN Demo Video](#)

Installation

1. Download [latest release](#)
2. Launch Blender, navigate to `Edit > Preferences` select `Addons` and then `Install`,
3. Navigate to the downloaded add-on and select `Install Add-on`

Sounds Familiar?

 **Flamenco**

[About](#) [Download](#) [Documentation](#) [FAQ](#) [Get Involved](#) 

[Edit page](#)

Navigation

Usage

- Quickstart
- Shared Storage
 - Shaman Storage System
- Worker Actions
- Variables
 - Blender
 - FFmpeg
 - Two-way Variables for Multi-Platform Support
- Manager Configuration
- Worker Configuration
- Jobs, Tasks, and Commands
 - List of Commands

Shared Storage

Flamenco needs some form of *shared storage*: a place for files to be stored that can be accessed by all the computers in the farm.

Basically there are three approaches to this:

Approach	Simple	Efficient	Render jobs are isolated
Work directly on the shared storage	✓	✓	✗
Create a copy for each render job	✓	✗	✓
Shaman Storage System	✗	✓	✓



Proposal

- Material for MkDocs*
- Continuous delivery via buildbot
- Edit with preview in Gitea
- Git (LFS?) repository
- Pulled with `make update?`
- developer.blender.org/docs

* Alternatives: Sphinx, VitePress, Hugo



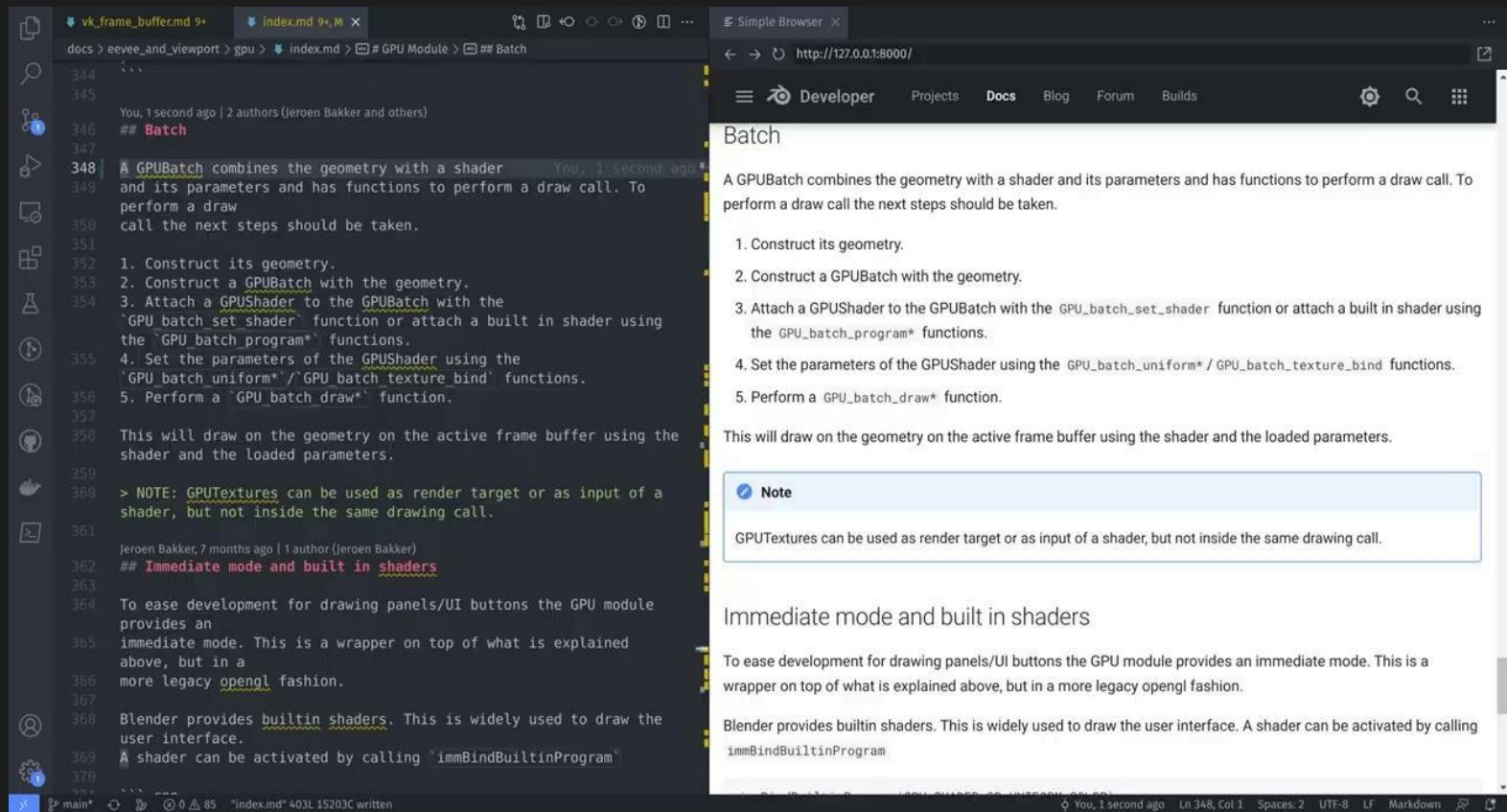
Goodbye Wiki?

- **What moves to the new platform?**
Most pages, including release notes
- **Personal pages & weekly reports:**
Personal repository on Gitea
- **Transition requires manual work**
Basic Wiki to Markdown converters available (<https://projects.blender.org/brecht/wiki-to-markdown>).
People volunteered to help.

Demo Time

developer.blender.org/docs

Demo: Offline Editing



The image shows a side-by-side comparison of a code editor and a browser rendering the same content. The code editor on the left shows the source code for a document, including a header, a main section titled "Batch", and a section titled "Immediate mode and built in shaders". The browser on the right shows the rendered output of this code, including a title "Batch", a paragraph of text, a numbered list of five steps, a note box, and another section titled "Immediate mode and built in shaders".

```
docs > eevee_and_viewport > gpu > index.md > ## GPU Module > ## Batch
344 ...
345
346 You, 1 second ago | 2 authors (Jeroen Bakker and others)
347 ## Batch
348 A GPUBatch combines the geometry with a shader
349 and its parameters and has functions to perform a draw call. To
350 perform a draw
351 call the next steps should be taken.
352
353 1. Construct its geometry.
354 2. Construct a GPUBatch with the geometry.
355 3. Attach a GPUShader to the GPUBatch with the
356 GPU_batch_set_shader function or attach a built in shader using
357 the GPU_batch_program* functions.
358 4. Set the parameters of the GPUShader using the
359 GPU_batch_uniform*/GPU_batch_texture_bind functions.
360 5. Perform a GPU_batch_draw* function.
361
362 This will draw on the geometry on the active frame buffer using the
363 shader and the loaded parameters.
364
365 > NOTE: GPUTextures can be used as render target or as input of a
366 shader, but not inside the same drawing call.
367
368 Jeroen Bakker, 7 months ago | 1 author (Jeroen Bakker)
369 ## Immediate mode and built in shaders
370
371 To ease development for drawing panels/UI buttons the GPU module
372 provides an
373 immediate mode. This is a wrapper on top of what is explained
374 above, but in a
375 more legacy opengl fashion.
376
377 Blender provides builtin shaders. This is widely used to draw the
378 user interface.
379
380 A shader can be activated by calling immBindBuiltinProgram
```

Batch

A GPUBatch combines the geometry with a shader and its parameters and has functions to perform a draw call. To perform a draw call the next steps should be taken.

1. Construct its geometry.
2. Construct a GPUBatch with the geometry.
3. Attach a GPUShader to the GPUBatch with the GPU_batch_set_shader function or attach a built in shader using the GPU_batch_program* functions.
4. Set the parameters of the GPUShader using the GPU_batch_uniform*/GPU_batch_texture_bind functions.
5. Perform a GPU_batch_draw* function.

This will draw on the geometry on the active frame buffer using the shader and the loaded parameters.

Note

GPUTextures can be used as render target or as input of a shader, but not inside the same drawing call.

Immediate mode and built in shaders

To ease development for drawing panels/UI buttons the GPU module provides an immediate mode. This is a wrapper on top of what is explained above, but in a more legacy opengl fashion.

Blender provides builtin shaders. This is widely used to draw the user interface. A shader can be activated by calling immBindBuiltinProgram

Demo: Offline Editing

The image shows a side-by-side comparison of a code editor and a browser. The code editor on the left displays a markdown file named `index.md` with the following content:

```
296 (attributes, uniforms, uniform buffers, textures and shader storage
297 buffer objects).
298
299 Jeroen Bakker, 7 months ago | 1 author (Jeroen Bakker)
300 ## Geometry
301
302 Geometry is defined by a GPUPrimType, one index buffer (IBO) and
303 one or more vertex
304 buffers (VBOs). The GPUPrimType defines how the index buffer should
305 be interpreted.
306
307 Indices inside the index buffer define the order how to read
308 elements from the vertex
309 buffer(s). Vertex buffers are a table where each row contains the
310 data of an element.
311 When multiple vertex buffers are used they are considered to be
312 different columns of
313 the same table. This matches how GL backends organize geometry on
314 GPUs.
315
316 Index buffers can be created by using a GPUIndexBufferBuilder
317
318 ``` cpp title="Create Index Buffer"
319 GPUIndexBufBuilder ibuf
320 /* Construct a builder to create an index buffer that has 6 indexes.
321  * And the number of elements in the vertex buffer is 12. */
322 GPU_indexbuf_init(&ibuf GPU_PRIM_TRIS, 6, 12);
323
324 GPU_indexbuf_add_tri_verts(&ibuf, 0, 1, 2);
325 GPU_indexbuf_add_tri_verts(&ibuf, 2, 1, 3);
326 GPU_indexbuf_add_tri_verts(&ibuf, 4, 5, 6);
327 GPU_indexbuf_add_tri_verts(&ibuf, 6, 5, 7);
328 GPU_indexbuf_add_tri_verts(&ibuf, 8, 9, 10);
329 GPU_indexbuf_add_tri_verts(&ibuf, 10, 9, 11);
330
331 GPUIndexBuf *ibo = GPU_indexbuf_build(&builder)
```

The browser on the right shows the rendered HTML version of the same content. The title is "Geometry". The text is rendered with appropriate HTML tags, including code blocks for the C++ code. The browser's address bar shows `http://127.0.0.1:8000/` and the page title is "Developer".



And Beyond...

- We need to **lead by example**
- **Technical documentation days**
Last Friday of every month.
- **Education**
Google's technical writing courses:
<https://developers.google.com/tech-writing>
- **Examples & templates**



And Beyond...

Documentation Structure



Building Blender

Windows
macOS
Linux

Develop

Getting Started
Code Documentation
Style Guide
Release Notes

Process

Communication
Modules
Bug Tracker
Code Review

More

Google Summer of Code
Python
FAQ

New Developer Introduction

Welcome! Advice on how to get started.

Communication

The most important thing.

Code & Design Documentation

Technical documentation about the code and big picture design.

Building Blender

Instructions for compiling Blender locally.

Modules

Blender components and their owners.

Style Guide

Coding guidelines and committer etiquette.

Tools

Setup your development environment.

Process

Release cycle, BugTracker, Code Reviews and Testing.

Release Notes

What changed in each Blender version.

Google Summer of Code

A program that introduces students to open source software development.

Python

Learn about scripting and add-ons.

Translation

Blender UI internationalization

Infrastructure

Details about the online ecosystem that supports Blender development.

FAQ

Common questions about the development process.

Building Blender

Windows
macOS
Linux

Develop

Getting Started
Code Documentation
Style Guide
Release Notes

Process

Communication
Modules
Bug Tracker
Code Review

More

Google Summer of Code
Python
FAQ

New Developer Introduction

Welcome! Advice on how to get started.

Communication

The most important thing.

Code & Design Documentation

Technical documentation about the code and big picture design.

Building Blender

Instructions for compiling Blender locally.

Modules

Blender components and their owners.

Style Guide

Coding guidelines and committer etiquette.

Tools

Setup your development environment.

Process

Release cycle, BugTracker, Code Reviews and Testing.

Release Notes

What changed in each Blender version.

Google Summer of Code

A program that introduces students to open source software development.

Python

Learn about scripting and add-ons.

Translation

Blender UI internationalization

Infrastructure

Details about the online ecosystem that supports Blender development.

FAQ

Common questions about the development process.

Blender Developer **Handbook**



Developer Handbook

New Developer Introduction

[Advice](#)

[Choosing a First Task](#)

Building Blender

[Linux](#)

[Windows](#)

[macOS](#)

Process

[Release Cycle](#)

[Contributing \(Review & Commit\)](#)

Modules

Guidelines

[C/C++ Code Style](#)

[Python Code Style](#)

[Commit Messages](#)

[Release Notes](#)

Testing

[C/C++](#)

[Python](#)

Tools

[Address Sanitizer](#)

Blender Developer Handbook



This is the Blender Developer Handbook. It tries to provide all the necessary general information needed to develop Blender. As such it is not only aimed at beginner developers, experienced developers use this handbook too.

Proof of Concept

The entire structure of the handbook is just another proof of concept. Pages are empty or don't even exist. Expect a bunch of 404 links.

Last update: 5 minutes ago

Created: 5 minutes ago



Developer Handbook

New Developer Introduction

Advice

Choosing a First Task

Building Blender

Linux

Windows

macOS

Process

Release Cycle

Contributing (Review & Commit)

Modules

Guidelines

C/C++ Code Style

Python Code Style

Commit Messages

Release Notes

Testing

C/C++

Python

Tools

Address Sanitizer

Blender Developer Handbook



This is the Blender Developer Handbook. It tries to provide all the necessary general information needed to develop Blender. As such it is not only aimed at beginner developers, experienced developers use this handbook too.

Proof of Concept

The entire structure of the handbook is just another proof of concept. Pages are empty or don't even exist. Expect a bunch of 404 links.

Last update: 5 minutes ago

Created: 5 minutes ago



Learning **Flow**

Asset System

Introduction

Fundamentals

Important Concepts

Architectural Overview

From File Browser to Asset System

Backend

Asset Catalogs

Asset List API

Asset Indexing

User Interface

Asset Browser

Asset Shelf

FAQ

Asset System



The asset system brings a native understanding of assets (entities packaged for sharing/reuse) to Blender's core design, and enriches it with a number of features for great asset based workflows.

For example it includes: The asset browser, asset shelves, asset libraries, asset library loading, asset catalogs, asset metadata, etc.

The [backend](#) implements all the core types and functionality, which various parts Blender can access. The design is user experience driven, and as such, the backend very much serves the [user interface](#). They work in close collaboration to provide an experience that makes assets feel like first-class citizens in Blender.

Note

The asset system is still in early development so expect this documentation to receive regular updates. In various places, it will refer to designs that are not there yet (at least not in the `master` branch), partially there or just temporary. This will be clearly indicated.

Last update: 18 minutes ago

Created: 18 minutes ago

Asset System

Introduction

Fundamentals

Important Concepts

Architectural Overview

From File Browser to Asset System

Backend

Asset Catalogs

Asset List API

Asset Indexing

User Interface

Asset Browser

Asset Shelf

FAQ

Important Concepts



Table of contents

What is an Asset anyway?

Asset Representation

A New Core (non-Main) Database

What is an Asset anyway?

The [Blender reference manual](#) gives a [user level answer](#) to this question. As far as the asset system design goes, **assets are arbitrary entities that are packaged for organized sharing/reuse.**

The term *entity* is used here, because the design is meant to support assets that are not [Blender data-blocks](#) (also called [IDs](#) or [ID Datablocks on a technical level](#)), even if the current implementation is limited to that. In future it should be possible to let the asset system deal with any data as assets, like files on disk, USD prims, SQL data-base entities, data fetched from the web, etc. This only works because the asset system doesn't deal with the actual underlying entity itself (the object, the material, the brush, etc.). It only deals with the package of the entity.

Asset Representation

If assets are packaged entities, what does the package look like? **An asset representation is the package for an asset, which enables the asset system to work with it.** When loading an asset library, an asset representation is created for each detected asset and put into the asset library storage.

Note that an asset representation is just the package itself, and usually doesn't contain the actual entity. It contains information on how/where to find the entity, so that the asset can be loaded when the user asks for it.



Asset System

Introduction

Fundamentals

Important Concepts

Architectural Overview

From File Browser to Asset System

Backend

Asset Catalogs

Asset List API

[Asset Indexing](#)**User Interface**

Asset Browser

Asset Shelf

FAQ

`asset-library-hash`[↑ Back to top](#)

Hash of the absolute file path of the asset library.

`asset-index-hash`

Hash of the absolute file path of the asset file.

`asset_file`

Filename of the asset file. Not used by Blender, but is added for discoverability convenience.

Content

```
{
  "version": <file version number>,
  "entries": [{
    "name": "<asset name>",
    "catalog_id": "<catalog_id>",
    "catalog_name": "<catalog_name>",
    "description": "<description>",
    "author": "<author>",
    "tags": ["<tag>"],
    "properties": [...]
  }]
}
```

`version`

the version of the asset index file. It is used to identify the structure of the content. Asset indexes with a different version than used by Blender would be regenerated. Blender 3.1-3.4 expect version attribute to be `1`. Later versions might require to change it.

Table of contents[Glossary](#)[Logical Overview](#)[Problem statement](#)[Solution](#)[Requirements](#)[Asset library loading process](#)[Asset Index](#)[Storage](#)[Content](#)[Performance](#)[Future expected changes](#)[References](#)

Asset System

Introduction

Fundamentals

Important Concepts

Architectural Overview

From File Browser to Asset System

Backend

Asset Catalogs

Asset List API

Asset Indexing

User Interface

Asset Browser

Asset Shelf

FAQ



is a proper asset system API for add-ons to use.

↑ Back to top

Python API

At this point, there is no proper Python API for the asset system yet, it is still in the planning. There are still ways to achieve commonly requested functionality through other parts of the Python API however.

How do I load custom previews? ¶

There is no simple `asset.load_custom_preview(filepath)` method or similar available. But there are still two ways to do the job:

- `bpy.ops.ed.lib_id_load_custom_preview()`: Attempts to load an image file from a `filepath` property (opens a File Browser if not set) to the active ID set in context.

```
override = context.copy()
# Set context "id" member to some ID, e.g. a material.
override["id"] = my_material
with context.temp_override(**override):
    bpy.ops.ed.lib_id_load_custom_preview(filepath="path/to/image.png")
```

Or to call the operator from a button:

```
# Set layout context "id" member to some ID, e.g. a material.
layout.context_pointer_set("id", my_material)
props = layout.operator("ed.lib_id_load_custom_preview")
props.filepath = "path/to/image.png"
```

- `bpy.types.ID.preview`: ID assets (and currently all assets have to be IDs) share the ID's

Table of contents

[How to port my asset add-on to the Asset Browser/System?](#)

Python API

[How do I load custom previews?](#)



Recap

- **New developer documentation culture?**
- Others did it, let's learn from them
- **Docs as code** based developer documentation, replacing Wiki
- Simple workflows, **optimized for engineers, open to the community**
- Restructured: Blender Developer handbook & learning flows



Status

Phase 0:

- ✓ Research, experiments & testing setup
- ✓ Get buy-in

Phase 1 - Setup:

- ✓ Buildbot CD setup
- ✓ Hosting on developer.blender.org/docs
 - Custom theme, navigation & landing page
 - Setup repository (Git LFS setup?)



Status

Phase 2 - Transition:

- Move relevant pages from Wiki
- Setup developer handbook
- Archive Wiki

Phase 3 - Source code integration:

- Checkout docs with source code?