Guide contents

Choose your path

Java Edition: choose a loader

Bedrock Edition: Add-Ons

Keeping both editions on one page

First complete projects for each edition

Status and original sources

Sources / References

Documentation: In progress **Quick start: In progress**

Offline copy from ModDocs. Source links remain clickable.

Minecraft modding

Minecraft Java Edition and Bedrock Edition share a name, but they do not share a mod project format. Choose the edition first. On Java, choose a loader and a target Minecraft version. On Bedrock, choose an Add-On pack and the creator APIs available for your target version.

> **Quick start:** for Java, pick one loader and follow its current first-mod tutorial for your Minecraft version. For Bedrock, start with one behavior or resource pack. Test a single visible change before adding more content.

Choose your path

You want to...	Start here
Add Java items, blocks, entities, or gameplay code	Fabric, Forge, NeoForge, or Quilt
Change Java textures or models without new code	A resource pack may be enough

You want to...	Start here
Add Bedrock entities, blocks, items, or behavior	Resource and behavior packs
Add Bedrock programmable behavior	Bedrock Script APIs, where supported

These paths are alternatives, not interchangeable layers. A Forge mod cannot be loaded as a Fabric mod merely by renaming its file. A Java mod `.jar` is not a Bedrock Add-On. Pick your audience and version before writing code.

Java Edition: choose a loader

Fabric, Forge, NeoForge, and Quilt each provide a project setup and their own API documentation. There is no single universal project template that can be followed without choosing one. Check which loader supports the Minecraft version you want and which APIs your mod needs. Then use that loader's **matching documentation version**, project generator or development kit, and Gradle wrapper.

- Fabric developer guides cover project setup and common game features. Creating a project is the first stop.
- Forge getting started explains its MDK and development environment.
- NeoForge getting started explains its project setup; use the site's version selector for the target game.
- Quilt developer wiki has an introductory path, but its documentation is still developing.

A small first Java mod

1. Write down the Minecraft version and one loader. Install the Java development kit required by that loader's current guide.
2. Create a project from that loader's official template. Open it in an IDE and run its development client before changing code. This proves the starting setup works.
3. Add one small feature using the same loader's guide, such as an item or block. Give it a unique namespace and name; do not reuse a vanilla identifier.

4. Test the feature in the development client. Check the log if it is missing or the client fails during startup.
5. Build with the project's Gradle wrapper. Test the built mod in a separate installation with the same Minecraft version, loader, and required dependencies.

Keep source code and assets in version control. A build success does not prove the finished mod loads or its textures and translations appear. Test the installed artifact, not just the editor run.

Common Java mistakes

- Following code from a different Minecraft or loader version. Registry and event APIs change.
- Mixing examples from Fabric, Forge, NeoForge, and Quilt in one project without an explicit cross-loader design.
- Testing only the development client, then shipping a build missing a dependency or resource.
- Using identifiers owned by the base game or another mod.

Bedrock Edition: Add-Ons

Microsoft's Bedrock Creator Documentation organizes Add-Ons around resource packs, behavior packs, JSON references, and scripting. A resource pack changes presentation; a behavior pack describes supported gameplay behavior. Script APIs provide programmatic features that the stable API supports. The right combination depends on what your content needs.

The getting-started guide explains the Add-On folder structure and development pack folders. Work in the development folders for quick testing, keep your own project copy, and activate the pack in a test world. Add a single visible change first. If it does not appear, check that the pack is enabled, the JSON is valid, identifiers match, and the world is using the intended pack version.

A small first Bedrock Add-On

1. Follow Microsoft's current resource-pack or behavior-pack tutorial, depending on whether your first change is visual or behavioral.

2. Give the pack its own manifest and identifiers according to the current schema. Avoid copying identifiers from a different project unchanged.
3. Put the files in the appropriate development pack folder and activate the pack in a disposable world.
4. Test one change, then add other resources or a script only if needed. Validate every JSON file against the versioned reference.
5. Package and test the final Add-On as a fresh install rather than assuming the development folder behaves exactly like distribution.

The Creator Documentation includes JSON references for blocks, entities, items, recipes, and more, plus the Script API reference. Check whether a particular script API is stable or experimental for the version you target.

Keeping both editions on one page

This page groups Minecraft editions for discovery, but the instructions remain separate. If you make a feature for both, treat it as two projects with a shared design brief. Test each edition independently and publish the supported game versions clearly.

First complete projects for each edition

For **Java**, make one new item in **one** loader project. Fabric's first-item tutorial shows the whole chain: registration, a name translation, texture, model, and client item file for its selected game version. Follow its matching project setup, run the development client, give yourself the item, and check its name and appearance. Build the JAR with the project's wrapper, put that JAR in a clean matching Fabric installation, and repeat the test. For Forge, NeoForge, or Quilt, use the same *goal* but their own Forge registry, NeoForge setup, or Quilt guide; do not paste Fabric registration code into those projects.

For **Bedrock**, the custom grass-block resource-pack tutorial is a complete first visual pack. Create its manifest with **new UUIDs**, add the texture at the documented path, enable it in a test world, and compare the target block before and after. Export the pack and import it on a clean profile to test delivery. If you want gameplay changes,

continue with Microsoft's behavior-pack introduction and Add-On development workflow; a resource pack alone cannot supply arbitrary behavior.

For both editions, write the exact game, loader or pack API, and dependency versions in the release notes. The most common false success is a project that works in a development environment only because source assets or other installed mods are present. Test the distributable file after temporarily removing the working project from the game's active mod/pack path.

Status and original sources

Unfinished: this page gives a practical route and troubleshooting checks, but a complete mod for each Java loader and each Bedrock pack type has not been built and verified here. Use the official Fabric, Forge, NeoForge, Quilt, and Bedrock Creator sites for current APIs and exact setup commands.

Sources / References

External project and documentation links cited in this guide. Check the upstream release and version before following version-sensitive steps.

- Fabric developer guides
- Creating a project
- Forge getting started
- NeoForge getting started
- Quilt developer wiki
- Bedrock Creator Documentation
- getting-started guide
- Script API reference
- first-item tutorial
- Forge registry
- custom grass-block resource-pack tutorial
- Add-On development workflow

- Forge
- NeoForge
- Quilt