

DEPRECATED FEATURES

View online: <https://www.construct.net/en/make-games/manuals/construct-3/tips-and-guides/deprecated-features>

Construct has been in development for many years. Much like with [superseded features](#), over time new features are occasionally introduced that replace older features. Usually the older features are supported as long as possible to improve backwards compatibility. However sometimes the maintenance burden and additional complications caused by having to continue to have the older features means that they must ultimately be retired.

When you open a project, Construct will warn you of any deprecated or retired features the project uses. The *Deprecated features* dialog may appear when opening a project to list all such features. It is strongly recommended to act as soon as possible to resolve all the items listed, thereby removing the use of all the deprecated features from your project. This will ensure your project continues to be supported and work correctly in future. If you don't take action, you may find your project stops working, or possibly cannot be opened, in a future release of Construct.

You can view the deprecated feature list for a project at any time, including if you previously checked *Don't show again for this project* in the dialog, by right-clicking the project name in the Project Bar and selecting Tools▶View deprecated features. If the option does not appear, it means the project does not use any deprecated features.

This guide covers every type of deprecation or retirement warning and describes steps to take to stop using the affected feature.

Legacy SDK v1 plugins/behaviors

The industry-standard approach for addon systems is to use [encapsulation](#) to limit what addons can do to known safe features only, which ensures they continue to work in the long-term. Historically, partly due to past limitations of the JavaScript programming language, Construct did not use encapsulation in its addon system. This meant addons could use unsafe features that they were not meant to. The proliferation of addons using unsafe features ended up breaking a lot of user's projects, and continued support for those addons meant facing the prospect of continuing to break user's projects on a regular basis. This old system is referred to as SDK (Software Development Kit) v1.

To fix this Construct moved its addon system to an industry-standard approach with encapsulation. The new system is referred to as SDK v2. This should guarantee that addons are permanently supported and safe to use in the long-term with virtually no risk of breaking your project. However plugins and behaviors must be updated by the addon developer to support the newer SDK v2.

As of Construct r450+, released in 2025, support for SDK v1 addons was removed. Construct will notify you that you cannot open a project if it uses SDK v1 addons. If you see such a message,

consider the following options to resolve it:

- Addons installed from the official [Addons website](#) should auto-update to the latest version. If you see a prompt about addon updates being available on startup, click the notification and update addons with the [Addon Manager](#). If the addon developer has published an update using SDK v2, this will then update the addon to that version.
- If the addon was not installed from the official addons website, check with the addon developer to see if an update is available, and manually download and install the updated addon.
- If there is no update for the addon using SDK v2, contact the addon developer and ask them to provide an update.
- Consider replacing the addons with other features. For example if your project uses a third-party addon for tweening, it may be possible to use the built-in Tween behavior instead. Alternatively a different third-party addon may be available that uses SDK v2. It may even be the case that you can just remove the addon if it turns out it is not really necessary.
- Otherwise SDK v1 addons are still supported in the r449.x [LTS releases](#), which will be supported up until the end of 2026.

The SDK v2 was first made available to addon developers in May 2024. LTS support up to the end of 2026 should provide plenty of time for addon developers to update their addons, or for existing projects to either be updated or completed.

Export file structure 'Flat' mode

Historically Construct exported projects with all project files in the root folder and with lowercased filenames, referred to as *flat* mode. In 2022 Construct was updated to preserve the folder structure and filename case of project files, referred to as *folders* mode. This is a much better mode as it preserves your folder organization and is important for features like JavaScript Modules. However changing the mode could break some projects, as it changes where some project files are found, such as when requesting project files by URL.

For example consider a project using a file named "Hello.txt" in a subfolder named "myfolder". The file path is in fact "myfolder/Hello.txt". However in *flat* mode, where project files are referred to by a string, this file could be loaded from the path "hello.txt", as it exports all files in the root folder with lowercased names. After changing to *folders* mode, the file must be referred to with the string "myfolder/Hello.txt", reflecting its true path; the old string will no longer refer to the file and may return an error such as 404 Not Found. This also applies to audio files referred to by a string, although the file extension is omitted for those.

In r450+, released in 2025, the legacy *flat* mode was removed, and all projects automatically use *folders* mode. Older projects may show a warning that the project used to use *flat* mode, which may mean you need to update any changed file references as described above.

Cordova iOS/Android scheme

In the past mobile exports internally ran on the *file:* scheme, much like opening a local HTML file on your computer in your browser. This mode is inefficient, has limited capabilities, and is extremely difficult to support due to severe technical restrictions on the limited features available with the *file:* scheme.

In 2020 Construct added support for the *app:* scheme for iOS, and in 2021 the *https:* scheme for Android. These modes work almost identically to the way a real HTTP server does, and are much more efficient, more capable, and easier to support. In r450+, released in 2025, support for the legacy *file:* scheme was removed, and all projects automatically use the *app:* or *https:* schemes. Older projects may show a warning that the project used to use the legacy *file:* scheme.

Warning: publishing a mobile app with a changed scheme will have the effect of clearing storage, as it changes the URL used internally to load the project, and storage is remembered based on the URL. Therefore you should not publish an update to an existing app that changes the scheme if that app uses storage. If you still need to maintain an already-published mobile project using the legacy *file:* scheme, you should continue using the r449.x *LTS releases* to update your app, which are the last releases that still support the *file:* scheme.

NW.js exporter

For many years Construct used a framework called NW.js for its desktop exports, along with an NW.js plugin for further integration. Modern versions of Construct switched over to separate Windows, macOS and Linux exporters using different technologies, and as of r450+ released in 2025, support for NW.js has been retired. The [Browser plugin](#) provides most of the windowing features of the old NW.js plugin, and the [File System plugin](#) provides access to local files and folders. The legacy Greenworks plugin for Steam integration with NW.js has also been replaced by the [Steamworks plugin](#) which works with the modern export options.

Deprecated plugins/behaviors

Construct will list any plugins or behaviors used in your project that are marked deprecated. This means the entire plugin or behavior is no longer supported and should not be used any more. Usually this is because it has been replaced by a newer addon or feature, or the addon was for a third-party service that was shut down. If your project uses deprecated addons, you should delete them from your project, and where applicable replace them with a newer addon or different feature.

To remove a deprecated plugin, all object types based on the plugin must be deleted from the Project Bar. To remove a deprecated behavior, all objects using the behavior must have the behavior removed.

To help you identify where a deprecated addon is used, follow these steps.

- 1 Right-click the name of the project in the Project Bar.

- 2 Choose Tools▶View used addons.
- 3 Find the deprecated addon in the list.
- 4 In the *References* column, the first few object names using the plugin or behavior are listed. To see a comprehensive list, right-click on the addon and choose Find all references....

Deprecated plugins and behaviors are hidden from the list when adding a new object or behavior. Therefore any plugins or behaviors you can still add from Construct are not deprecated. Any of these can be used as replacements for deprecated addons.

Functions

The legacy Function plugin will also be listed as a deprecated plugin. In 2019, Construct 3 introduced [a new built-in functions feature](#) which replaces the old Function plugin. Projects using legacy Functions should switch to using the new built-in system.

To help with converting to built-in functions, you can right-click an *On function* condition in the old Function plugin, and select *Replace with built-in function*. Note due to differences between the features, it may not always be possible to automatically replace the function, and you will need to do it manually instead.

For more information on built-in functions, see the manual section on [Functions](#).

Classic scripts

Construct's JavaScript coding feature relies on JavaScript Modules, using `import` and `export` statements. However it originally used "classic" mode scripts, which don't support `import` or `export` statements. In 2020 Construct added support for modules, and in 2021 it removed support for the legacy "classic" mode scripts. To help you identify compatibility problems with old projects, Construct continues to warn you if you open a project that was still using "classic" mode scripts.

To update the project to use modules, refer to the tutorial [Upgrading projects from classic scripts to modules](#). The next time you save the project the deprecation warning will no longer appear.

Other deprecated features

Sometimes individual conditions, actions or expressions are marked deprecated, or effects are deprecated. Construct does not currently notify you about these, because they are generally minor and easy to continue to support in the long term. However you may notice these as some conditions, actions, expressions or effects that you can access from the user interface differ from the ones referenced in an old project. If this is annoying for any reason, you can delete the old usage and add it back with the new usage to update it, but usually this is not necessary to keep the project working.