TYPESCRIPT SUPPORT

View online: https://www.construct.net/en/make-games/manuals/addon-sdk/guide/typescript-support

The Addon SDK supports using TypeScript to develop plugins and behaviors. All the addon SDK samples have TypeScript support, with a .ts equivalent of every .js file, to help make it easy to get going with TypeScript. However using TypeScript is optional - if you prefer to stick to just JavaScript, then ignore or delete any .ts files in the addon SDK samples and just keep working with .js files.

Note this process is similar to using TypeScript for Construct projects, but with some altered steps due to the fact that addons are not associated with a project.

Installation

To use TypeScript with the Addon SDK, you will need a TypeScript-compatible code editor. This guide uses Visual Studio Code, or VS Code for short.

Install VS Code using the link above if you don't already have it. You'll also need to install TypeScript support, which you can do by following these steps:

1. Install Node.js if you don't already have it
2. In a terminal, run the command `npm install -g typescript`

You can check the TypeScript compiler, or `tsc` for short, is installed by running `tsc --version` in the terminal. It should print the version installed.

> *Modern versions of Windows use PowerShell for the terminal, and running some of the above commands could return an error like tsc.ps1 cannot be loaded because running scripts is disabled on this system due to the security restrictions set by default. To fix this, run the command* `Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser` *to allow permission to run the command, and then try the original command again.*

For more details see TypeScript in Visual Studio Code.

Setting up the addon files

The SDK sample addons come with TypeScript support, having a .ts file for every .js file, so you don't need to do anything in this step if you're starting with one of those samples.

If you want to add TypeScript support to an existing JavaScript-only addon, then create a copy of every .js file with the file extension renamed to .ts. You'll then need to go through every .ts file

and add type annotations to it. The SDK samples should provide a guide of what kinds of type annotations and adjustments to make. Note that in some cases you may need to import and export types between files to ensure all types are checked correctly.

Set up TypeScript

For proper type checking, you'll need to export Construct's type definitions to your addon folder. To do that, follow these steps.

1. In Construct, enable developer mode and then reload Construct.
2. In the main menu, a new *Developer mode* submenu should have appeared. In that submenu, select *Set up TypeScript for addon*. Note this requires a browser that supports the File System Access API - try using Chrome or Edge if the option does not appear.
3. A folder picker appears. Select the folder that contains your addon (where addon.json is located).

This will make two changes to your addon folder:

1. A subfolder *ts-defs* is created, with a range of TypeScript definition files for Construct's APIs.
2. The file tsconfig.json (TypeScript configuration file) is created in the addon folder with default TypeScript settings, but only if the file does not exist. If tsconfig.json already exists then it does not alter it.

Note that Construct's TypeScript definition files will be added to and possibly revised over time. To update the TypeScript definition files to the latest version, choose *Set up TypeScript for addon* again in a future release of Construct, and it will update all the TypeScript definition files to the latest versions. Remember that if tsconfig.json already exists then this does not change it, so any alterations you've made to the TypeScript configuration will be persisted.

Workflow

Once you are up and running, you will likely want to make repeated changes to your TypeScript code, and easily be able to test your addon in Construct.

In VS Code, press `Ctrl` + `Shift` + `B` and then select *tsc: watch*. This enables a mode where VS Code will automatically compile your .ts files to .js whenever you save the file. Note this must be done once per session.

Assuming you are using a developer mode addon, then your workflow can go like this:

1. You make a change to a TypeScript file and save the change
2. TypeScript then automatically compiles the .ts file to .js (or reports errors if you made a mistake)
3. When you next preview a project in Construct, it will reload the latest .js files from your local web server.
4. Then the preview starts up running your latest code changes.

Summary

TypeScript support takes a couple of steps to set up - in particular using the *Set up TypeScript for addon* option - but brings many benefits to addon development, such as type checking and better auto-complete in compatible editors like VS Code. Once set up the workflow is largely automatic and you can go from changing a .ts file to seeing it in Construct on your next preview. Construct's type definitions will change over time, but you can just repeat the *Set up TypeScript for addon* step to update the type definitions to the latest version.