USING AN EXTERNAL EDITOR

View online: https://www.construct.net/en/make-games/manuals/construct-3/scripting/guides/using-external-editor

Construct provides its own built-in code editor based on Monaco to help you conveniently write code in your project. However you may wish to use an external code editor, such as an industry-standard tool like Visual Studio Code (aka VS Code). Construct has special features to help you use external editors, and this guide describes how they work.

Using TypeScript requires some extra steps which are detailed below. However the basic setup is explained first, which works well for JavaScript projects.

Step 1: save a folder project

First of all, make sure you save your project to a folder. This means all your project files, including your script files, are saved as individual files within the project folder, rather than all contained within a .c3p file. For more information see the section on project folders in Saving projects.

Step 2: enable auto-reload

Once saved to a folder, right-click the main *Scripts* folder in the Project Bar, and select the *Auto reload all on preview* menu option. After doing that the menu option will appear with a tick to indicate the auto-reload mode is enabled.

Enabling this setting means that every time you preview in Construct, it will re-load all your script files from the project folder, ensuring any changes made by an external editor are loaded.

Step 3: open external editor

Now you can open your script files in your favorite external code editing tool like VS Code. If the tool has an option to open a folder, you probably want to open the *scripts* subfolder in your project folder, as that is the folder that contains all your JavaScript code. Then you should be able to view and edit your project's script files.

Step 4: make changes

Now try making a change in the external editor, save the change, and go to Construct and preview the project. The changes made in the external editor should be immediately reflected in preview.

When auto-reload mode is enabled, Construct also shows notifications when you preview indicating how many script files were changed. If you also have a script file open in Construct when you preview, Construct will also reload the file from the project folder to show its latest contents.

Be warned that in auto-reload mode, if you make a change to a script file in Construct, don't save the changes, and then preview the project, your changes will be lost as they will be replaced with the contents of the file in your project file. It's best to consistently edit your scripts from the same editor to avoid this.

Using an external editor with TypeScript

The previous steps are the basic setup for being able to edit code with an external editor, which works well with JavaScript. For TypeScript there are some extra steps involved.

To help you get started, this section describes the necessary steps to set up TypeScript with VS Code, but note the steps may be different for other code editors - refer to their documentation for specific instructions.

Installation

With some external editors like VS Code, you may need to install TypeScript separately. You will only need to do this once to set up your code editor.

1. First install VS Code if you haven't installed it already
2. Next, install Node.js if you haven't installed it already
3. Then in a terminal run run the command `npm install -g typescript` to install 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 refer to TypeScript in Visual Studio Code in VS Code's official documentation.

Setting up the Construct project

Assuming you've followed the steps above, you've already got a project saved to a folder. There are some extra steps you need to do to make sure you can use TypeScript with an external editor though. This includes exporting the TypeScript definitions that Construct uses. To do this, right-click on the *Scripts* folder in the Project Bar, and select TypeScript ▶ Set up TypeScript for external editor. This only needs to be done once per project and performs the initial setup for using TypeScript in your project. It does three things:

1. It creates a TypeScript (.ts) file copy of every JavaScript (.js) file in your project. The TypeScript files aren't shown in the Project Bar, but they'll appear in your project folder. If the project doesn't have any JavaScript files, Construct creates the initial two default script files *main.js* and *importsForEvents.js*, and then creates TypeScript copies of those.
2. A file named *tsconfig.json* is created in the project folder. This is a configuration file for TypeScript. This also does not appear in the Project Bar, as it's just for the external code editor.
3. A subfolder named *ts-defs* is created with lots of *.d.ts* files. These are TypeScript definition files, which tells TypeScript about all of Construct's built-in APIs, as well as some types that are specific to your project. This also doesn't appear in the Project Bar.

If you run *Set up TypeScript* again, it won't overwrite any existing tsconfig.json or .ts files. This means it's safe to run again later on, for example if you add more .js files and want a quick way to make .ts copies of them.

TypeScript workflow

Once you are up and running, you will likely want to make repeated changes to your TypeScript code, and easily be able to preview the result in Construct. To make this process work smoothly, 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.

Then, assuming that you have enabled *Auto reload all on preview* as described above, the workflow goes 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. Then preview the project in Construct, at which point the .js file is re-loaded from the project folder

Sometimes you may make changes to the project that affect the TypeScript definition files Construct generated for you. Alternatively when updating to new versions of Construct, the TypeScript definition files could change too. To make sure the TypeScript definition files are up-to-date, right-click on the *Scripts* folder in the Project Bar, and select TypeScript ▶ Update TypeScript definitions. This essentially does only step 3 from the *Set up TypeScript* steps.

Don't add .ts files in Construct

Note that we recommend using the workflow described here, where you only have the .js files in your Construct project, and the .ts files are only used by the external editor. This basically out-sources compilation of TypeScript to JavaScript to your external editor, with Construct only using the compiled JavaScript output.

If your Construct project has both a .ts and .js version of the same file, Construct gives precedence to the .js file. This should mean your external editor workflow keeps working as expected, but there's no reason to keep the .ts file in the project - Construct isn't actually using it at all, and so to avoid confusion, we would recommend not having it in your project. In particular this could be confusing if you try to edit the .ts file from within Construct: it would not change the .js file and so not affect the way your project works, and auto-reload mode would overwrite the file on the next preview, so your changes would be lost anyway.

For more advice about TypeScript workflows, see TypeScript in Construct.