

# COMPCODE

convert project files into shareable *After Effects* scripts

manual for v1.0

`compCode` is an [After Effects](#) tool that converts your design to a dynamic and shareable After Effects script. If you are familiar with [Photoshop](#) actions, and have ever wished for a way to do such things in After Effects, `compCode` is the solution. Its simple, just create a composition, even include pre-comps, multiple layers, effects, expressions, shape layers, you name it. Then hit the `compCode` button, It will magically recreate your entire composition with javascript code, saving hours upon hours of work. To make things more [smart](#) special tokens are coded into the script Use the `text:` token to prompt user to input text when they run the script, `file:` to import footage, or `name:` to prompt user for item/layer/property name.

This tool is perfect for templates, creating tools to sell, or even corporate identity packages, in an easy to deploy JSX file.

Included is the `compPack` feature, quickly build script UI and package together multiple scripts into one. Have folders of scripts inside of other folders, `compPack` is smart enough to rebuild the UI to match your existing folder structure. In addition add artwork and information buttons to polish your script, and get it ready to share with others.

## Table of contents

|                          |    |
|--------------------------|----|
| 1. Installation          | 3  |
| 2. <code>compCode</code> | 4  |
| 2.1 User Interface       | 5  |
| 2.2 Settings             | 5  |
| 3. <code>compPack</code> | 9  |
| 3.1 User Interface       | 10 |
| 4 Limitations            | 11 |

# Installation

Follow these two simple steps to install script:

1. Unpack the archive you have downloaded and choose folder based on your operating system:

- Mac

copy [compCode.aex](#) to plugins folder:

Applications/Adobe After Effects <version>/Plugins

copy [compCode.jsx](#) to Scripts UI folder:

Applications/Adobe After Effects <version>/Scripts/ScriptUI Panels

- Win.

copy [compCode.aex](#) to plugins folder:

Program Files\Adobe\Adobe After Effects <version>\- Support Files\Plugins

copy [compCode.jsx](#) to Scripts UI folder:

Program Files\Adobe\Adobe After Effects <version>\- Support  
Files\Scripts\ScriptUI Panels

Install [compCode Demo Toolkit.jsx](#) to Scripts UI folder to follow up with tutorial or to see the flexibility and power compCode can bring to your users,

2. [Allow script to access network](#) to avoid unnecessary problems while loading GUI. This option is under General tab of After Effects Preference pane:

- Mac: After Effects > Preferences > General

- Windows: Edit > Preferences > General

Once installation is complete run the script in After Effects by clicking [Window > compCode](#).

# COMPCODE

Deliver your project files as After Effects scripts

compCode is dedicated to AE template builders that ship their products as a single AEP file. While this workflow is a standard procedure, it's not really comfortable for end user, as he is forced to import entire template just to use a few element of it.

compCode eliminates this in a heart beat.

First of, user is able to choose WHAT he needs from your template - a single button click rebuilds your design from code. Simple as that. No more duplicate items once user chooses to build two instances of same design.

Theres no way to ruin your original design, as it is all built into the code. User clicks a button to build your template. Boom, it's there. In terms of failure, he's able to create another template from scratch hassle free.

compCode comes packed with tokens that lets you, as a designer, prompt user for item/layer name, external footage, such as video, image or image sequence or sound. Even more, user can be prompted to enter custom text for text player in order to enter custom web address, info string or something similar. And all that is build into code. Upon user runs your script, he is automatically prompted to enter such information - no more pain digging into project to find that layer, that needs to be replaced, or a text layer, where you enter your company name. It's all automatic and takes no effort from user side.

You have an option build your design in users current composition, without creating a new one and later copy/pasting your design. This is useful if you need to deploy text animations or extravaganza animated shape layers, or even a set of solid layers - your entire design will be created in current composition.

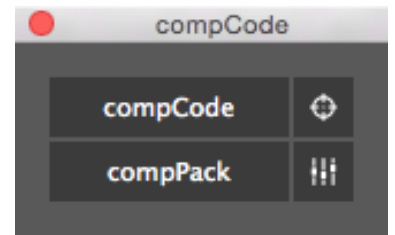
## User Interface

[compCode](#) starts the process of converting your design to javascript code.

[compPack](#) opens additional window with options for packing external JSX files to toolkit.

[Open Script File](#) executes user selected script file.

[Settings](#) opens settings window.



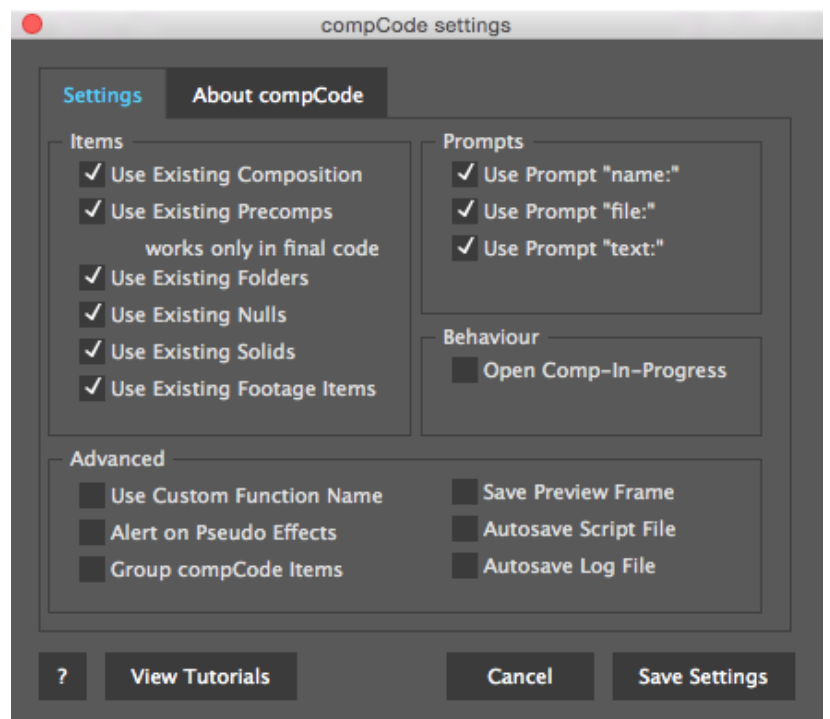
## compCode settings

Before digging into details, lets define some keywords:

[Final script](#) - the javascript (JSX) file that compCode produces.

[End user](#) - is the person who runs Final Script produced by CompCode.

[Main composition](#) - currently opened composition.



Settings window is divided into 4 parts: [Items](#), [Prompts](#), [Behaviour](#) and [Advanced](#).

Options that are set in [Items](#)/[Prompts](#)/[Behaviour](#) will have a direct influence for the end user. For instance, if you disable [Use Existing Folders](#), then new folder hierarchy will be created once end user runs final script. If you enable [Open Comp-In-Progress](#) then pre-comps, that script is currently working on, will also be opened on user end.

[Advanced](#) options does not have any impact on final script behaviour.

## Use Existing

Enable [Use Existing Composition](#) to target currently active composition - your design will be created in composition that is currently opened in the viewport. If you leave this option off, then each time user runs the script, the new composition will be created. For the most part this checkbox should be disabled.

However, [Use Existing Precomps](#) will only reuse pre-comps, not the main composition. If main composition contains pre-comps, then those pre-comps will be reused (if they exist in project panel) each time user runs the script.

*/NOTE this option works only in final code.*

Disable the rest of [use existing XXX](#) checkboxes in case your design requires new footage items or solid/nulls/pre-comps recreated each time user builds your design. For the most part those should be enabled.

If [Use Existing Folder](#) option is enabled, script will look for folder hierarchy in end users project panel and reuse it if it's found. If not - then folder hierarchy will be recreated from scratch. So if end user accidentally deletes entire folder hierarchy from AE project, rest assured that it will be recreated again once user runs the script. If you leave this option off, then entire folder structure will be created each time user runs your script.

Same principle applies to rest of items.

Having all these options ON will save disk space on users project file, as all items will be reused and not duplicated.

## Prompts

`compCode` provides 3 sets of tokens: `name`, `file` and `text`.

Use `name`: token to prompt user for layer, property or item name. This feature is useful if you deploy templates equipped with expressions that rely on unique composition or layer names.

`File`: token provides option to select custom footage items, such as image, image sequence, video or audio file. Powerful when deploying templates that use footage items in compositions. User is prompted to select custom footage items upon script launch, so all your placeholder items gets automatically replaced with user items. Apply `file`: token to item name.

`Text`: token is used on text layers. Once script encounters this token, user gets automatically prompted to enter custom text. To evoke this token, apply `text`: string to text layers name.

User prompts will be ignored if `use prompt` checkboxes are turned off. That is, end user will not get prompted for input when code encounters any of the tokens.

Disable these option for project testing purposes only and enable them once you build for final code.

## Behaviour

`Open Comp-In-Progress` - sometimes it's nice to see what's happening when code is being executed. If you are using lots of pre-comps, then the script will automatically open a pre-comp that it is currently working on and end user will be able to see what layers are being added to that composition, what properties are applied to layers and so on. Doesn't really have much of a use except of visual feedback on what's going on.

## Advanced

`Use Custom Function Name` - by default `compCode` will generate a unique function name each time you run it (`compCode_DATE_TIME()`). Feel free to enable it - it might be useful for better code reading experience. Doesn't have any effect on code performance.

**Alert on Pseudo Effects** - `compCode` will alert you in case you use Pseudo Effects in your project. It's very likely that end user will not have Pseudo Effect installed on his system, and you have completely forgotten about using one, so it's better to be safe than sorry. Enable this to be alerted on such cases.

**Group `compCode` items** - depending on **Use existing XXX** options, `compCode` will generate new items in the project panel. Enabling this option will put all duplicates to a dedicated folder `_compCode items`.

**Save Preview Frame** will automatically save a screenshot of main composition once you save final code.

**Autosave Script File** - final code will be auto-saved once progress window is closed to the location of your current After Effects project file and will have a name of current main composition. Useful when working on a large project.

**Autosave Log File** - if `compCode` encounters any problems that it cannot auto fix (such as encountering Pseudo Effects, Level, Curve or other Custom Value properties), it will write them to log file. Enabling this option will auto-save log file if it contains any logs when you save your final code.



# COMPPACK

Pack them up into a personalized toolkit

[compPack](#) generates a standalone toolkit that encapsulates the collection of JSX files. Each script will be available for launch in the tree view, that also has a search option.

Originally built to work hand in hand with [compCode](#), it can be easily used separately with your own JSX file collection.

Select a folder containing JSX file and [compPack](#) will pack them up. Feel free to use multilevel folder hierarchy - code will pick it up and will generate a multi-level tree selection window.

More than that, you are free to personalise your toolkit with your branding or a logo, and have it linked to your website upon mouse click. Add additional information about your tool with yet another option to have an info icon in the interface.

In case you decide to use [compPack](#) with your own scripts, make sure to include commented out function name (`// functionName();`) in your JSX script at first line. This way [compPack](#) will know what function should be executed to fire-up your action.

There is just one additional step you should do before deploying toolkit to the masses - open it up with ESTK, that ships together with After Effects, and export your script as Binary file. This way all external scripts, that [compPack](#) is referring to, will be collected to a single file.

That's it. Boom. Ship it out!

# User Interface

## General

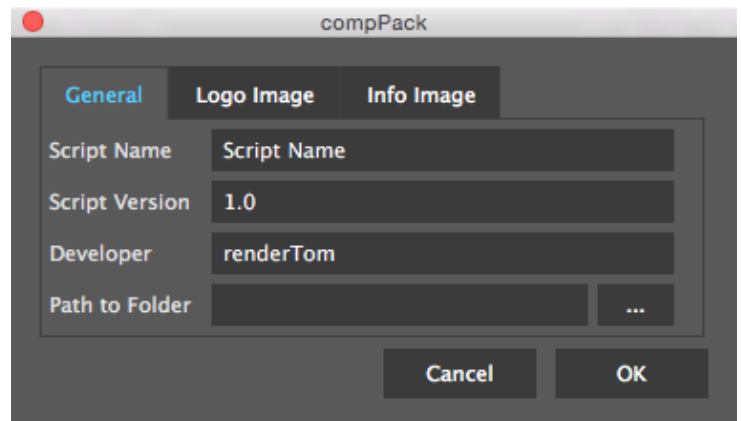
**Script Name** - The name of your script.

**Script Version** - The version of your script.

Please don't add "v" in front of version number as it's being added automatically.

**Developer** - your name or branding. This option is required when using logo image.

**Path to Folder** - select a folder hierarchy with JSX files you want to include in your toolkit.



## Logo Image

**Regular** - regular state of image (png file only);

**Over** - mouse over state of image (png file only) - must be same dimensions as regular image. If no file is selected, then mouseOver event will be ignored

**URL Link** - link to webpage that will be opened once user clicks on your logo. If you leave it blank, then mouseClicked event will have no effect.

**Orientation H** - horizontal alignment of you image. Choose either Left, Centre or Right;

**Help Tip** - short information displayed once mouse pointer is over image;

## Info Image

**Regular** - regular state of info image icon (png file only). Recommended dimensions 12x12. Image shouldn't be bigger then logo image. This image is positioned on the right side of logo image.

**Over** - Mouse over state of info icon (png file only) - must be same dimensions as regular icon. If no file is selected, then mouseOver event will be ignored

**Orientation H** - horizontal alignment of icon image. Choose either Left, Centre or Right;

**Orientation V** - vertical alignment of you icon image. Choose either Top, Centre or Bottom;

**Help Tip** - short information displayed once mouse pointer is over icon;

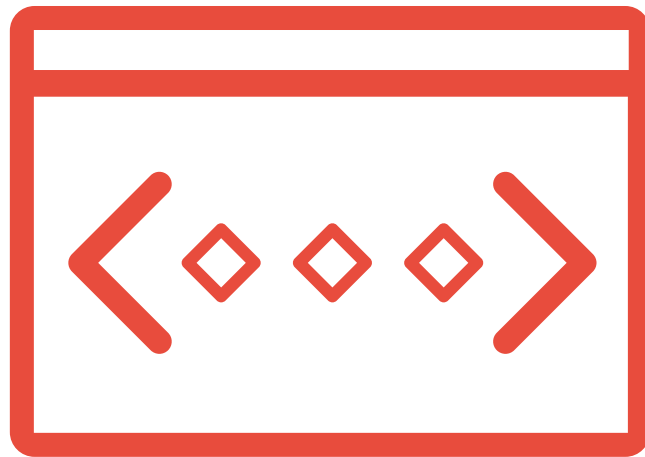
**Info Message** - enter text information that should be displayed in Alert window once user clicks on icon image.

## Limitations

Not all properties are available via scripting in After Effects. Here's a small list where [compCode](#) will struggle converting compositions to javascript file:

- [Custom data values](#). Effect properties with custom data values can't be used, such as the Levels Histogram and the Hue/Saturation effect. In these cases you could use the Levels (Individual Controls) and Color Balance (HLS) effects instead.
- Paint, Rotobrush and Puppet Pin effect values are not supported
- Not all text properties are supported:
  - Faux Bold, Faux Italic, All Caps, Small Caps, Superscript, Subscript properties are read only, so [compCode](#) will not be able to set them.
  - Vertical Scale, Horizontal Scale, Baseline Shift, Tsume properties supported from 2014.2 and newer.
  - Multiple text styles on one layer are not supported
- Non-english AE interface does not support [Layer Styles](#);

# COMP**CODE**



developed by **TOMAS ŠINKŪNAS**  
[www.rendertom.com](http://www.rendertom.com)