COMPCODE

convert project files into shareable **After Effects scripts**

manual for v1.1

compCode is an After Effects tool that corverts 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 value: token in property expression to prompt user for custom value upon script run, 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.

In compCode v1.1 you have an option to have compPack toolkit to have thumbnails as well - save your images as PNG files with the same name as the script it is referring to and you done - compPack will pick it up and display images instead of boring text.

Table of contents

Installation

Follow these two simple steps to install script:

1. Unpack the archive you have downloaded and copy/paste files (both "compCode.jsx" and "compCode Help.pdf") to "ScriptUI Panels" folder:

    - Mac OS: Applications/Adobe After Effects <version>/Scripts
    - Windows: Program Files\Adobe\Adobe After Effects <version>\- Support Files\Scripts

If folder ScriptUI Panels does not exist, create a folder and name it "ScriptUI Panels". Then paste the copied files into it.

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 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.

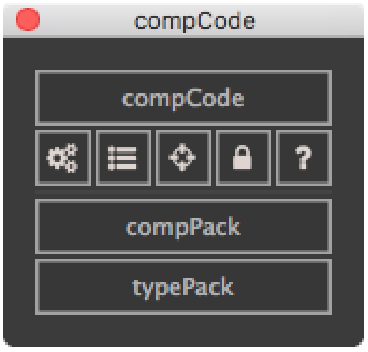
Theres no way to ruin your original design, as it is all built into the code. User clicks a button to build your template. Boom, it's there. In terms of failure, he's able to create another template from scratch hassle free.

compCode comes packed with tokens that lets you, as a designer, prompt user for property value, 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 animated shape layers, or even a set of solid layers - your entire design will be created in current composition.

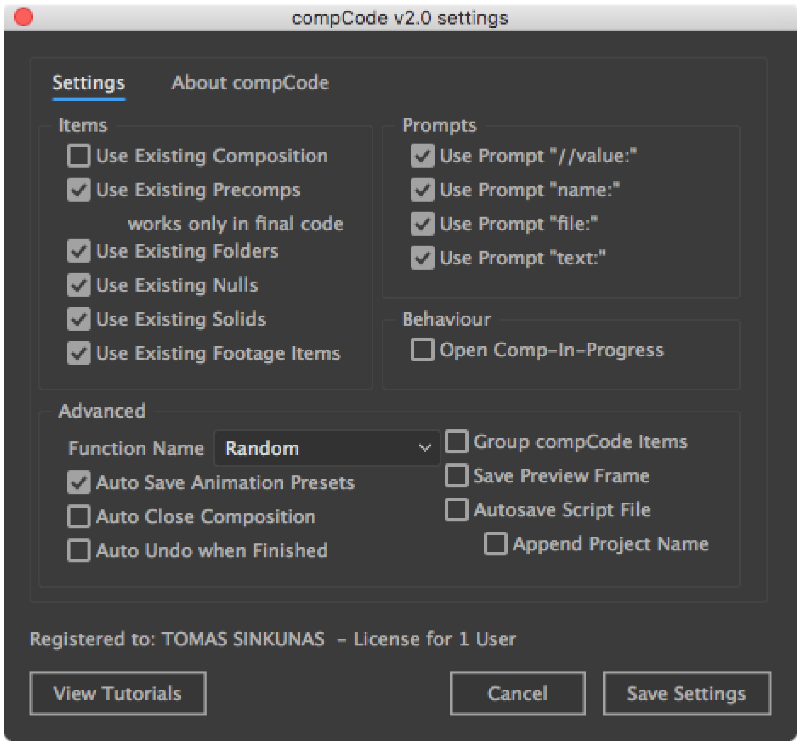
Another great compCode feature is that you don't need to worry about version compatibility - the code compCode produces is compatible with CS6. Design your animation in latest AE version and rest assured that CS6 users will be able to reuse it (as long as you don't use any new features, that aren't included in CS6 version).

User Interface



compCode starts the process of converting design to ExtendScript code.
Settings opens settings window.
CheckList provides few options to check AE project before exporting JSX.
Open Script File executes user selected script file.
JSX Compiler compiles multiple JSX to JSXBIN files.
Help opens this document.
compPack opens additional window with options for packaging external JSX files to a stand-alone After Effects toolkit-script. NOTE: this option works only with JSX files generated with compCode.
typePack provides option to package animated typefaces to JSX files for use with Font Manager http://aescripts.com/font-manager/

compCode settings



Items
- [ ] Use Existing Composition
- [✓] Use Existing Precomps
  works only in final code
- [✓] Use Existing Folders
- [✓] Use Existing Nulls
- [✓] Use Existing Solids
- [✓] Use Existing Footage Items

Prompts
- [✓] Use Prompt "//value:"
- [✓] Use Prompt "name:"
- [✓] Use Prompt "file:"
- [✓] Use Prompt "text:"

Behaviour
- [ ] Open Comp-In-Progress

Advanced
Function Name: Random
- [✓] Auto Save Animation Presets
- [ ] Auto Close Composition
- [ ] Auto Undo when Finished
- [ ] Group compCode Items
- [ ] Save Preview Frame
- [ ] Autosave Script File
  - [ ] Append Project Name

Registered to: TOMAS SINKUNAS – License for 1 User

Before diging 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.

Tokens

compCode provides 4 sets of tokens: value, name, file and text.

To prompt user for a property value (position, scale, color etc), simply add "value:" token into expression. However, there's a trick you should follow:

If current property does not have expression applied, then use toke value //value: expression. This looks rather silly, but AE will not evaluate it without first value, and for compCode to work it needs to have //value: token. So to keep it simple - if no expression exists on property - use value //value:

If current property already has an expression, then type in //value: token somewhere in the expression.

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

Function Name defines the function name in final code:

- Random will generate a unique function name each time you run compCode (compCode_-DATE_TIME()).
- Composition Name will use composition name. While this option is good for readability, make sure to have unique composition names, so no code collision appears once you use compPack.
- Custom will prompt to enter function name.

Auto Save Animation Presets will ask you to save Animation Presets when compCode encounters Pseudo Effects or properties with property value of type CUSTOM_VALUE or NO_VALUE (Levels, Curves, Hue & Saturation, Gradient Stroke, Gradient Fill etc). However, keep in mind that:

- If you have multiple identical instance of same effect (you duplicated Levels effect or Curves) compCode will save Animation Preset for each of these effects. This is because there's no way to compare the values of such effects. Therefore new Animation Preset must be saved once CUS-TOM_VALUE or NO_VALUE properties are encountered.
- Animation Presets are not backwards compatible. If you export these on CC2014, users will not be able to use final code on CC2013. To avoid this, you either use compCode in lowest AE version you want to support, or export Animation Presets manually from the lowest AE version you want to support.

Auto Close Composition automatically closes composition after compCode finishes working on it.

Auto Undo when Finished - since compCode creates new instances of compositions/lay-ers/items/other elements when compCoding, this option lets you automatically undo this step after process is finished.

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 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.

Append Project Name will append the name of After Effects project file to saved JSX file (aeProjectFile-Name_compositionName.jsx).

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.

Choose to use thumbnails instead of tree view list. Simply have thumbnails reside next to script files and have same name as script file (myScriptFile.png/myScriptFile.jsx).

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 then 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.

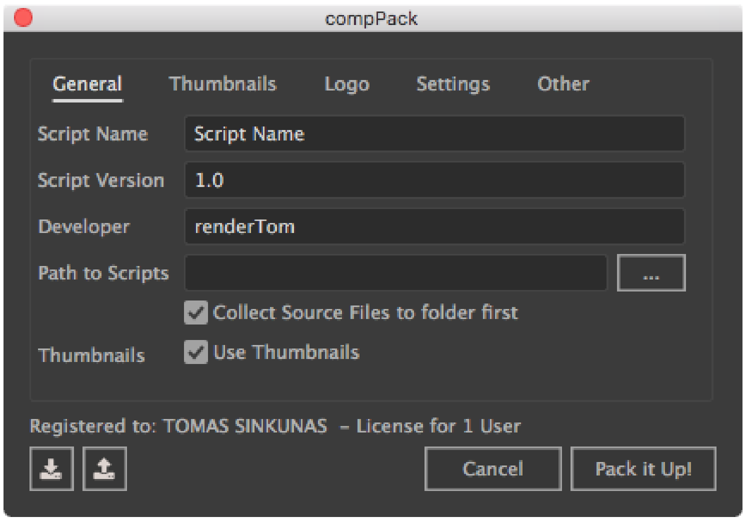
In case you decide to use compPack with your own scripts, make sure to include commented out function name (// functionName();)) in your JSX script at first line. This way compPack will know what function should be executed to fire-up your action.

There is just one additional step you should do before deploying toolkit to the masses - compile it to JSXBIN file. To do so, execute "JSX Compiler" from compCode's main UI window - this way all external scripts, that compPack is referring to, will be collected to a single file. NOTE for this to work you need to have ExtendScript, that ships together with After Effects, installed on your system.

That's it. Boom. Ship it out!

User Interface

General tab



Script Name: Script Name
Script Version: 1.0
Developer: renderTom
Path to Scripts:
✓ Collect Source Files to folder first
Thumbnails: ✓ Use Thumbnails

Registered to: TOMAS SINKUNAS – License for 1 User

Script Name - The name of your script.
Script Version - The version of your script. 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 Scripts - select a folder hierarchy with JSX files you want to include in your toolkit.
Collect Source Files to folder first - will collect all used (JSX, PNG, FFX) files to one folder. Very similar to After Effects "Collect Files..." feature. Recommended to have this option enabled, as compPack modifies JSX files it reads.
Use Thumbnails - in order to have thumbnails instead of text list, have thumbnails in same folder as JSX files with same name (myScriptFile.png/myScriptFile.jsx).
Load/Save compPack Metadata - saves and loads interfaces options (script name, path to script, checkbox values etc).

Logo tab

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 mouseClick 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 / Settings tab

Depending on Use Thumbnails is enabled or not, this tab will change it's name from Info to Settings.
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.

Thumbnails tab

Checkbox text - the text that is displayed for end user to show/hide thumbnails.
Enable Thumbnails by Default - choose if thumbnails should be displayed on first script launch.
Thumbnail Size - size of PNG file (width, height) in pixels. Use size 85/85 to have two columns that fit nicely to Characters tabs, or 55/55 to have three columns.
Spacing - spacing between thumbnails in pixels.
Make sure to compress image files with services such as www.tinypng.com to save script file size.

Other tab

Include other files that are not accessible within compPack UI to convert them to binary strings and be available in the scope of final code. PLEASE DO NOT include any unnecessary files, as the file size of script will grow.

TYPEPACK

A companion tool for Font Manager

typePack is compCode's extension (similar to compPack), that wraps JSX typeface files into a shareable library for Font Manager to use.

Font Manager is an After Effects tool for managing and creating animated typefaces. It brings ability to share animated typefaces as an After Effects project files. You simply import AEP file into your project and start using it. Simple as that.

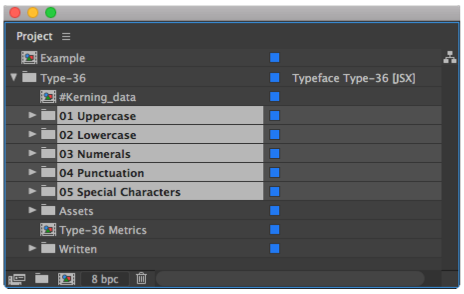
However, with this technique you have to import entire AEP typeface library, even though you might only need few characters from it. Your project gets messy and you end up with extra stuff you don't actually need in your project.

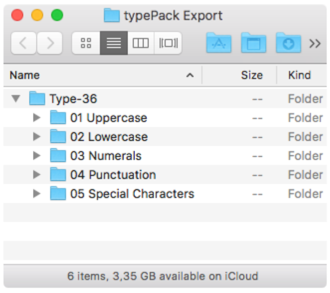
typePack to the rescue!

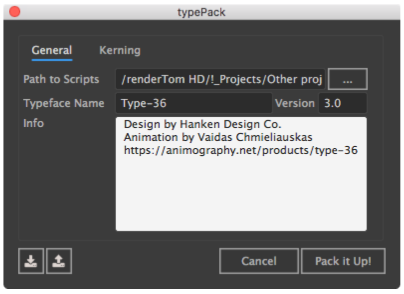
Use compCode to export animated typefaces to separate JSX files, fire-up typePack to package all JSX files into a shareable library and then feed the result to Font Manager. Font Manager will generate only those characters that you require to build a sentence. This means no duplicate compositions, no extra stuff, cleaner project, lower project file size. Everybody's happy!

Font Manager developed by Alex White http://aescripts.com/font-manager/

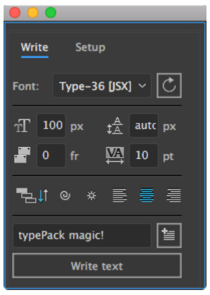
How to typePack







Path to Scripts: /renderTom HD/!_Projects/Other proj
Typeface Name: Type-36 Version: 3.0
Info: Design by Hanken Design Co.
Animation by Vaidas Chmieliauskas
https://animography.net/products/type-36



Font: Type-36 [JSX]
100 px auto px
0 fr 10 pt
typePack magic!

1. Create animated typeface with Font Manager as you would normally do.

2. Select multiple glyph folders and then use compCode to batch export them to JSX files.
NOTE - make sure you don't have duplicate composition names in same folder.

3. Launch typePack to package exported files into a library. Make sure to fill-in all necessary information. If your typeface has kerning data, go and select #Kerning_data composition in AE, then navigate to Kerning tab in typePack and click Read Kerning Data. Once finished click on Pack it Up!

4. Optional step. If you want to share your JSX typeface library, it's recommended you convert JSX to binary JSXBIN files. To do that, run JSX Compiler from compCode's UI.

5. Grab the resulting typeface folder and place it into Font Manager Library folder under ScriptsUI Panels folder. This is the place from where Font Manager reads JSX typefaces. For more information please refer to Font Managers documentation.

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.~~ Since compCode version 1.1 you can have these properties exported and applied as Animation Presets. Woo hoo!

- Master Properties are not supported
- 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;~~ Fixed in compCode version 1.1: Layer Styles now work in any AE interface.

COMP**CODE**

developed by TOMAS ŠINKŪNAS
www.rendertom.com