

The logo features the text 'RDC' in a large, bold, white sans-serif font. To the upper right of the 'C' is a smaller '18' in a bright blue color. The entire logo is centered within a thick, bright blue square border that is slightly rotated clockwise. The background is a dark grey-blue with a pattern of faint, light grey square outlines of varying sizes and orientations.

RDC¹⁸

ROBLOX DEVELOPER CONFERENCE



Studio Tips & Tricks

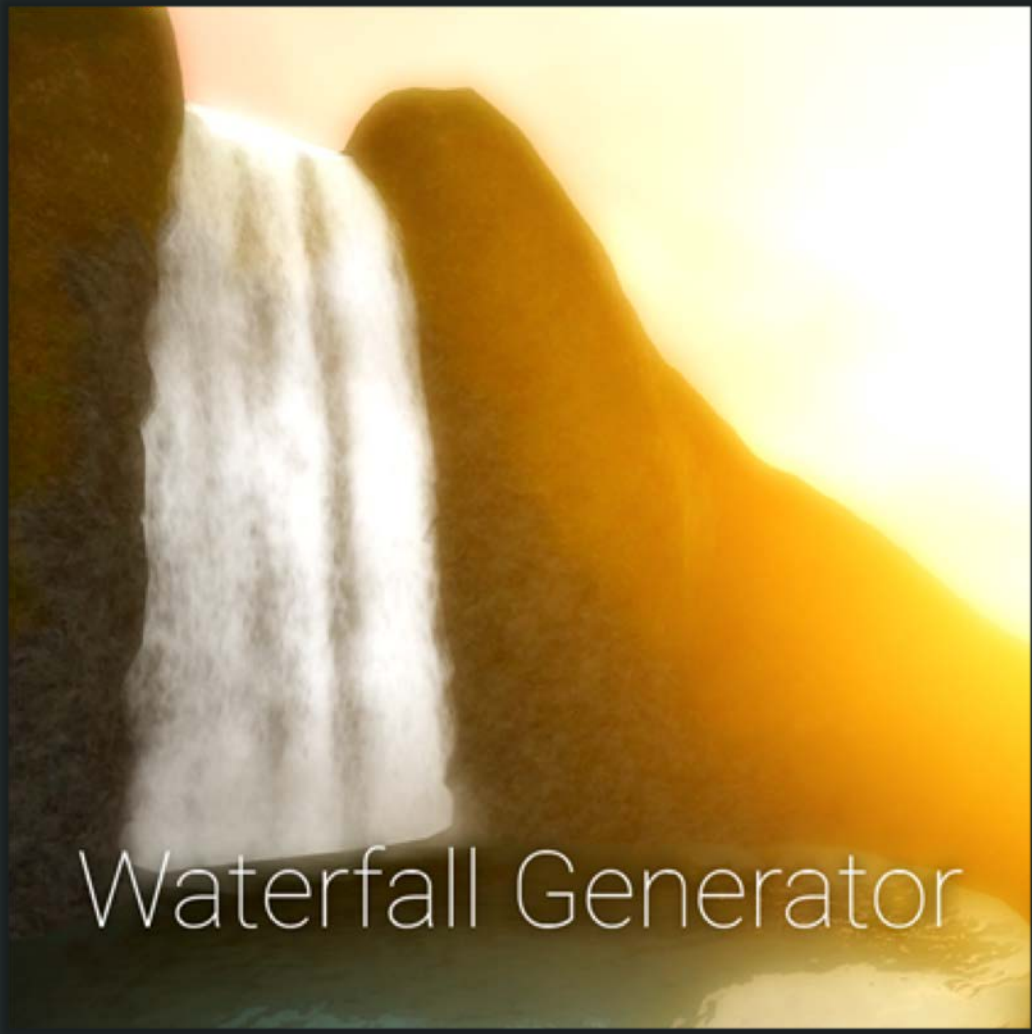
Brad Sharp
@bradsharppp

Who I Am

Brad Sharp (woot3)

- Roblox Engineering Intern
- Developer since 2008, worked on Egg Hunt 2018
- **Plugin Developer**





Waterfall Generator



Light Editor

Overview

- Settings
- Shortcuts
- Command Bar
- Code Management
- Plugins
- Q&A



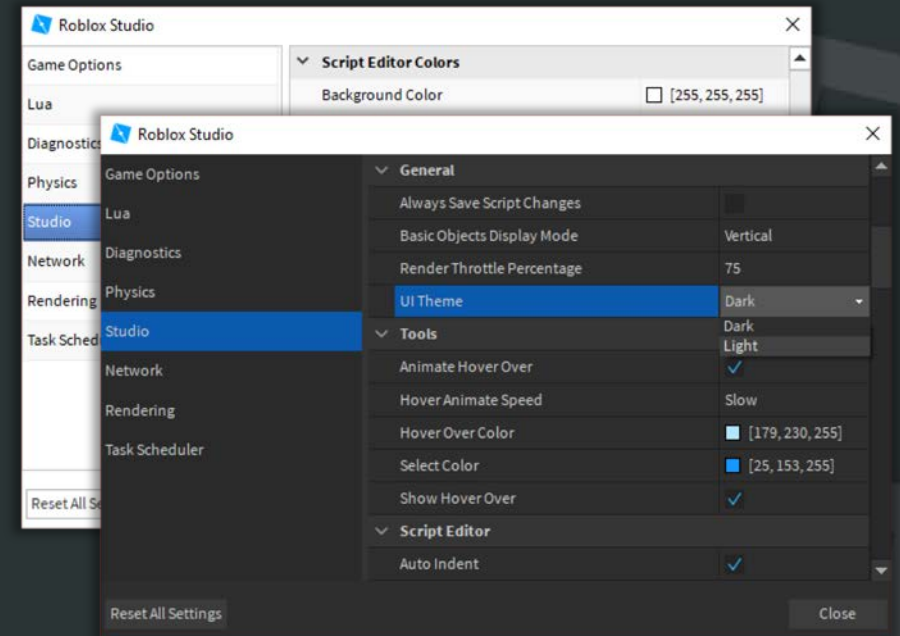


Settings



Settings

- Customize studio to suit you
 - Adjust camera configuration
 - Change script editor color scheme
 - Display physics debugger
 - Enable **Dark Theme**



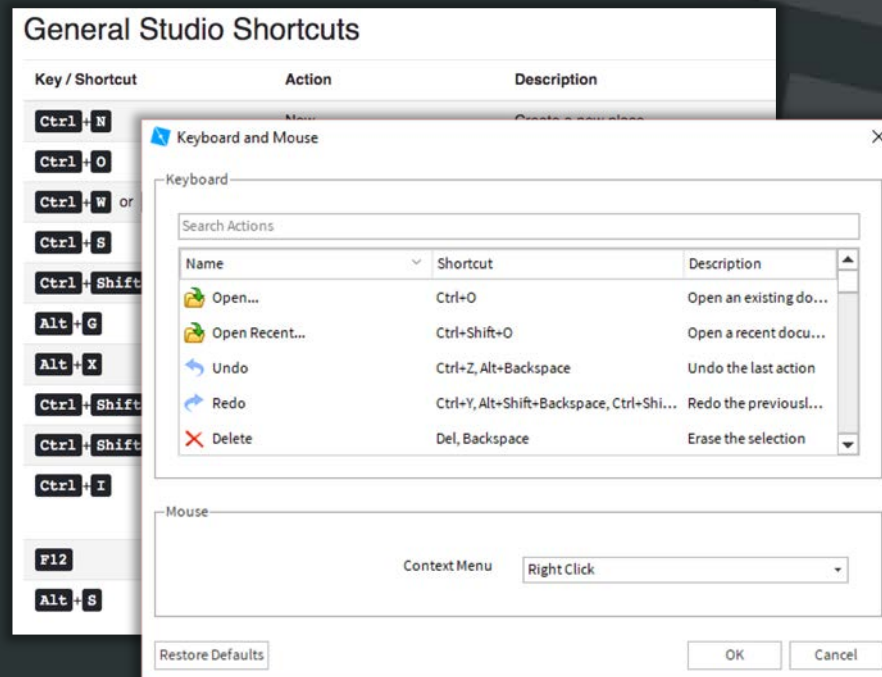


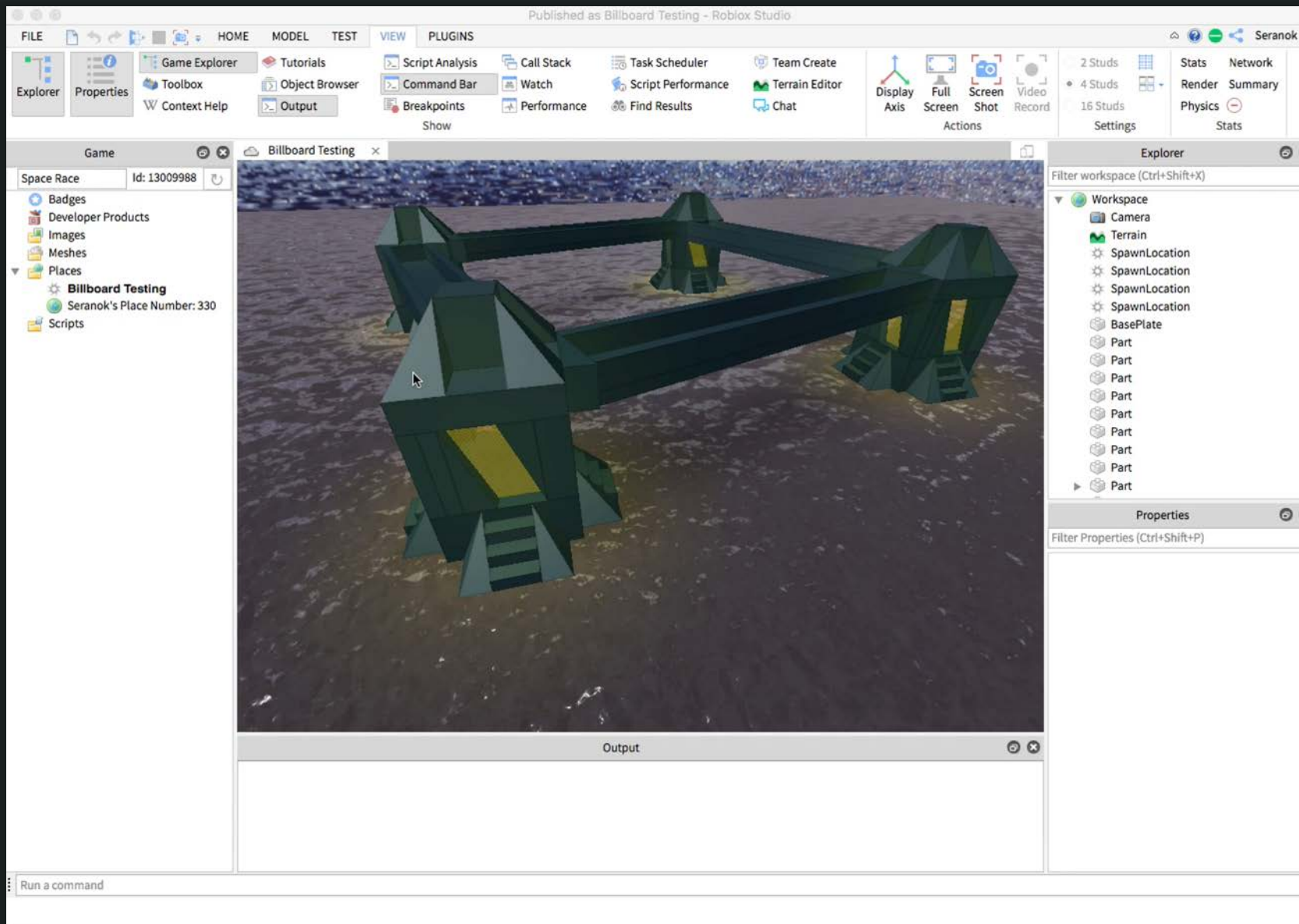
Shortcuts



Keyboard Shortcuts

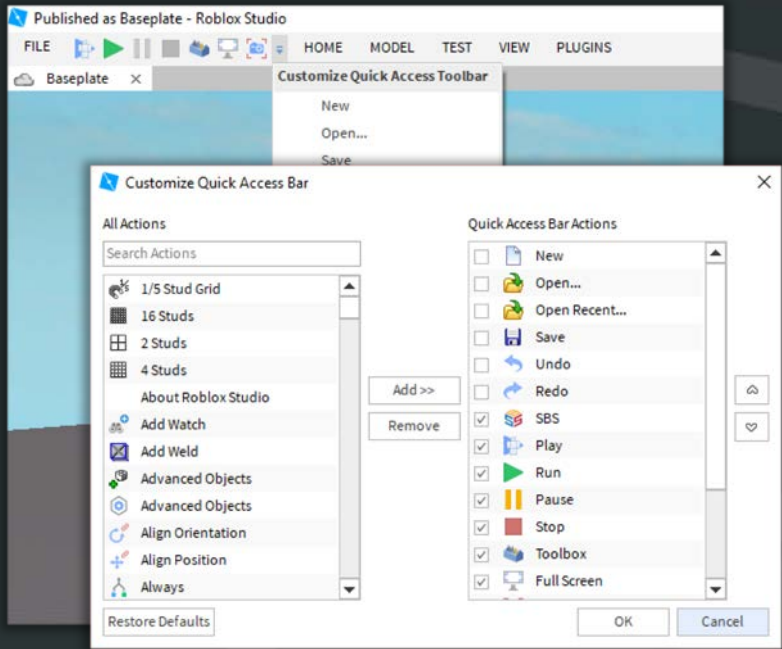
- Speed up development
- Customizable
- Plugins can create their own
- Full list is available online
 - <http://robloxdev.com/articles/Roblox-Studio-Shortcuts>

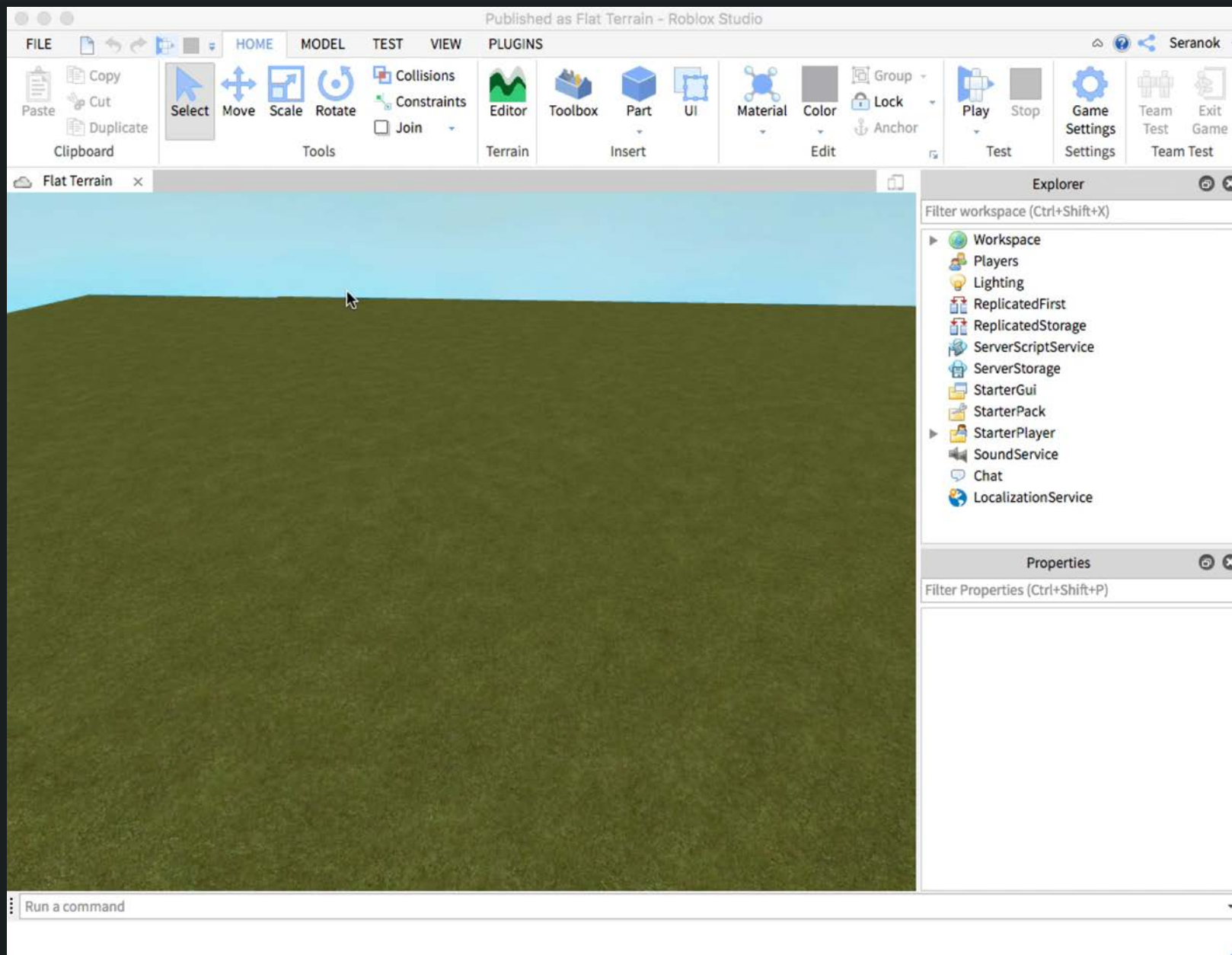




Quick Access Toolbar

- Toolbar for accessing common functions quickly
- Customizable
- Plugins can be added to it





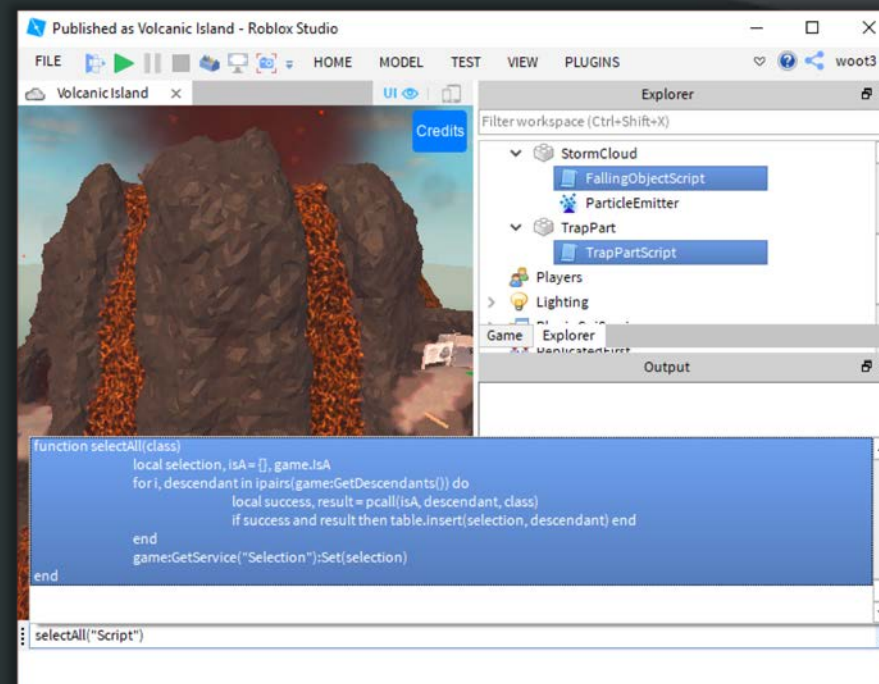


Command Bar



Command Bar

- Run code snippets
- Register functions for later use
 - Write a function
 - Run it in command bar
 - Call the function again later





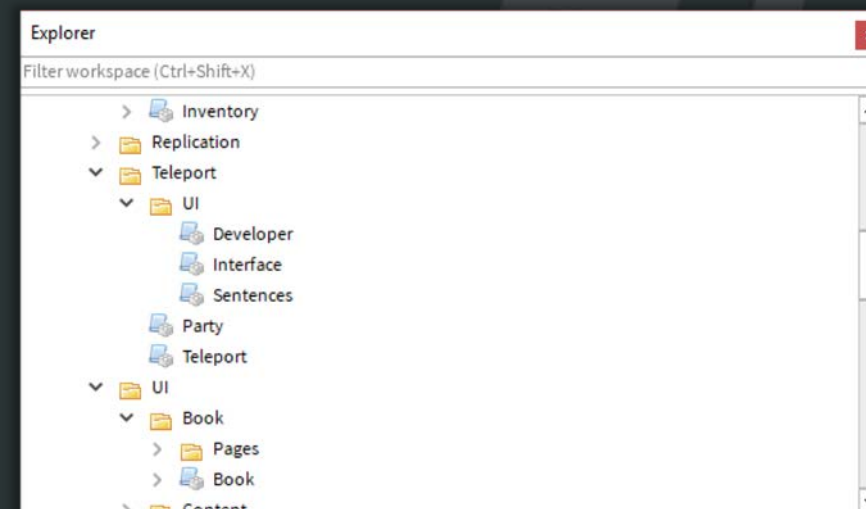
Code Management & Practices



Code Management

Use Module scripts to separate code

- Makes it easy to find specific code
- Especially useful when collaborating
 - This year's Egg Hunt had over 40,000 lines of code managed by 7 programmers
- Common industry practice



Good Coding Practices

- Break down large chunks of code into smaller more modular blocks
- Use folders to group together related modules
- Name your scripts descriptively

```
_init
2
3
4 local replicatedStorage = game:GetService("ReplicatedStorage")
5 local base, permitted = {}, {}
6
7 local function getObjectForInstance(instance)
8     if not instance then return end
9     if instance:IsA("ModuleScript") then
10         permitted[instance] = true
11         return require(instance)
12     end
13     return instance
14 end
15
16 local function getNextScriptInStack()
17     local depth, source = 0
18     repeat
19         depth = depth + 1
20         source = getfenv(depth).script
21     until source == script
22     return source
23 end
24
25 local function fp(path, relative)
26     if not relative then relative = game end
27     if path:sub(1, 1) == "/" then
28         local service = path:match("[^/]+")
29         relative = game:GetService(service)
30         path = path:sub(service:len() + 3)
31     elseif path:sub(1, 2) == "./" then
32         local source = getNextScriptInStack()
```

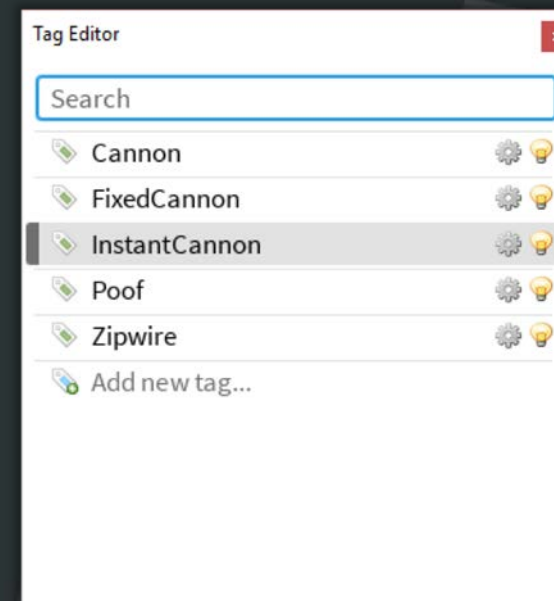


Plugins



Plugins


- Allow you to add extra functionality to Studio
- Many exist that will make specific tasks quicker and easier to do
- You can always create your own, with Lua, to suit your needs



Plugin Actions

- Actions can be used with shortcuts
 - Keyboard
 - Quick-access
- Developers no longer need to rely on `UserInputService` for hotkeys

API: Class/Plugin/CreatePluginAction

Function of  Plugin

```
PluginAction CreatePluginAction (  
    string actionId,  
    string text,  
    string statusTip  
)
```

Parameters:

actionId

Type: `string`

Required

text

Type: `string`

Required

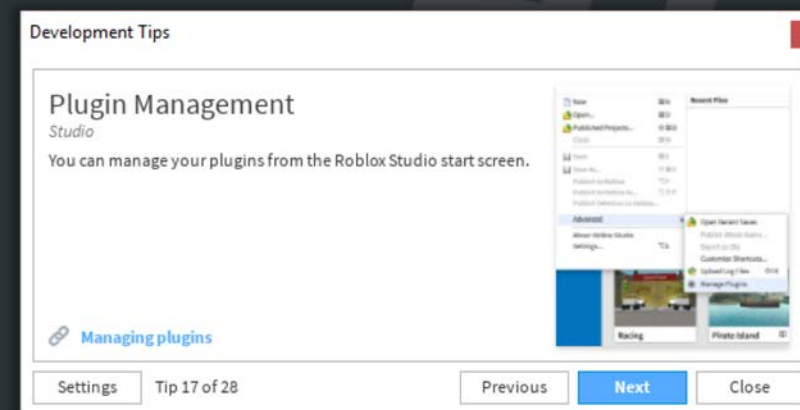
statusTip

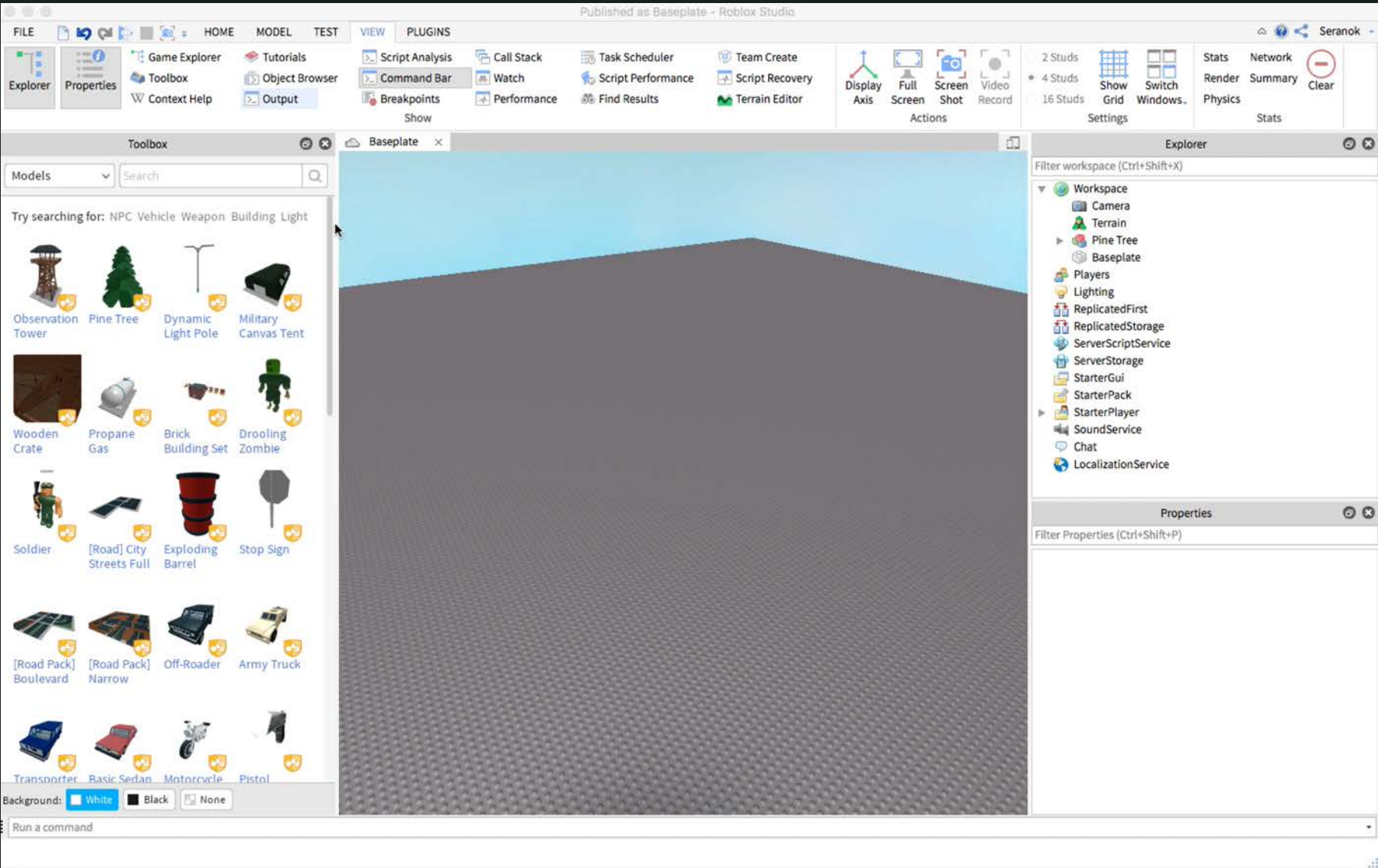
Type: `string`

Required

Plugin Widgets

- 2D dockable widgets for plugins
- Allow you to simulate existing Studio tools
- Our existing plugins use them
 - Terrain Editor





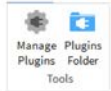
Plugin Creations

- If it doesn't exist already, you can create your own
- <http://robloxdev.com/articles/Intro-to-Plugins>

LEARN ROBLOX > CODING AND SCRIPTS

Intro to Plugins

10 min



A **plugin** is a custom add-on to Studio which adds new behavior and features that are not normally included. Both the Animation Editor and Terrain Tools were originally developed as plugins. There are also many plugins made by the Roblox community that you can use to help make games and experiences. Plugins don't have to be complicated; they can be simple tools that make your development life easier.

In this guide, we'll be creating a custom plugin that will let us insert new scripts into our game without the default "Hello world" print function.

Creating a New Plugin

While plugins can have many objects in them, they all start off as a **Script**. We can create this plugin script in Studio.

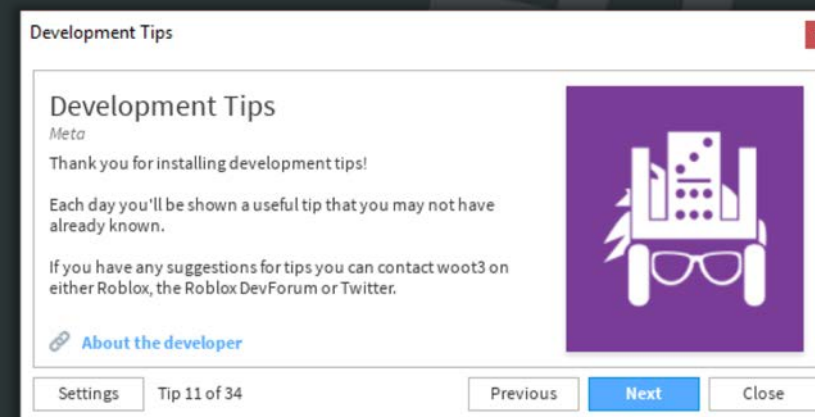
1. Create a new **Script** in **ServerStorage**.
2. Name the script *EmptyScriptAdder*.

CONTENT

- Creating a New Plugin
- Creating a Toolbar Button
- Supporting Undo/Redo
- Sharing Your Plugin
 - Share Plugin Locally
 - Publish to Roblox
- Managing Installed Plugins
- Other Plugin Examples
 - Experimental Mode Checker
 - Insert Empty Folder

Further Tips

You can get more Studio tips by installing the 'Development Tips' plugin; you can find this by searching for 'Tips' in the plugin library.





Reflection





Q&A

Roblox Username: woot3

Twitter: @bradsharppp