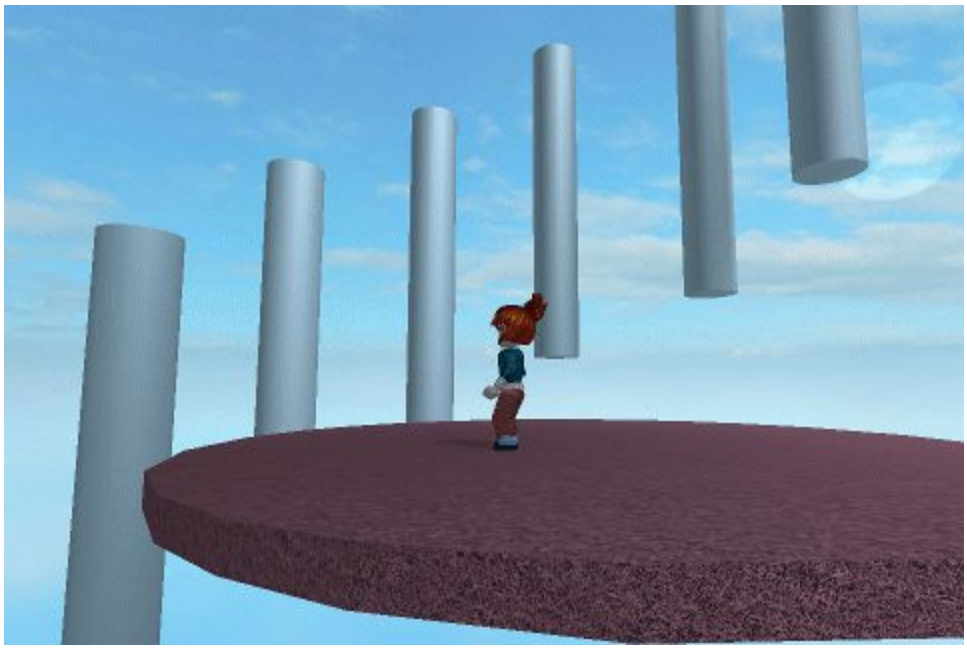
Coding for Multiple Parts - script.Parent

Required Previous Lessons/Knowledge: *Strings, Variables, Commenting, Creating Scripts, wait(), while true do*
Lesson Running Time: *10 -15 minutes. Can run longer if giving time for independent practice*
Optional Handouts: *Intro to Lua*
Learning Objectives and Outcomes: *Add a script directly to a part rather than ServerScriptService to create a part that can be duplicated multiple times within a game.*

The code you've used so far won't run if there are multiple parts of the same name.

What if you want more than one copy of your color changing part in your game and don't want to have to write a new script every time?

You can use the same code over and over with just a few tweaks and place copies of your color changing part all over your game. To do that, we'll add a script to the part instead of into ServerScriptService.



Adding a Script to a Part

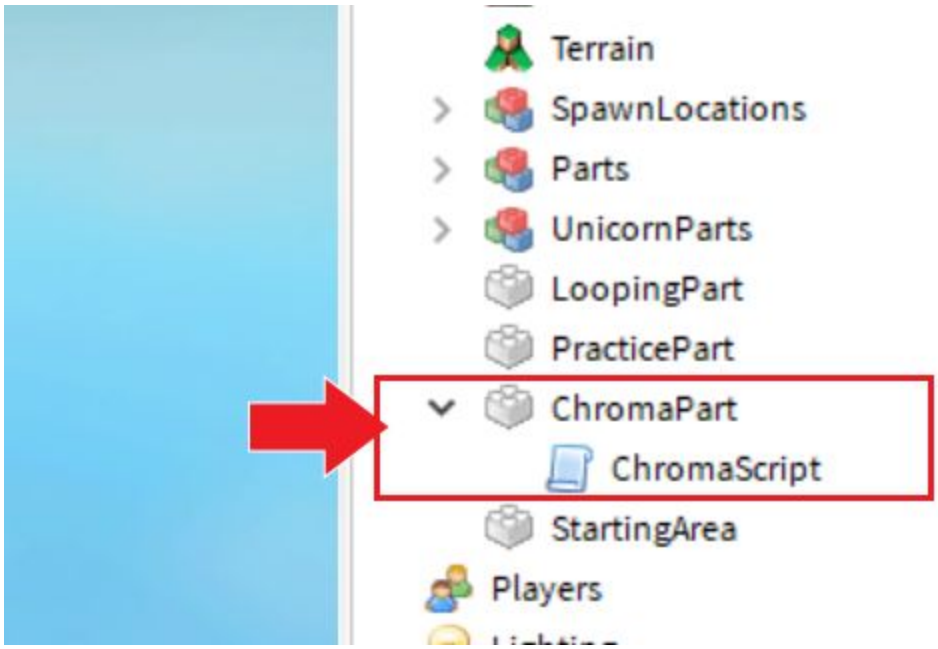
Instead of creating the script underneath ServiceScriptService, add a script straight to a part.

1. Create a new part and rename it. The example code will use *ChromaPart* and *ChromaScript*.
2. Right-click the **part**, *not ServerScriptService*, and select **Insert Object** > **New Script**.
3. Rename the script and add a comment.
4. Delete Hello world.

Meaning of Chroma
Chroma is latin for color, so it's a cool sounding name for a part that changes colors all the time.

Parents and Children

A **parent** is any part that has objects such as scripts or other parts grouped below it. The objects beneath the parent part are **children**. With the script and part you just created, the part is parent, and the script is the child. This is important to know for the next bit of code.



Use script.Parent

Once again, you are going to have to give the script directions on how to get to the part. This time, the part is the script's parent. How often do children get to boss around their parents?

1. Create a new local variable.
    - Example: `local chromaPart`
2. Tell the variable to store the script's parent part.
    - On the same line type: `= script.Parent`

Script is not a variable
Cool fact, script is not a variable. `script.Parent` will always look for the script's parent. No matter what the script is named.

Code Example

```
-- Changes the color of the script's parent every few seconds

-- Variables
local chromaPart= script.Parent
```