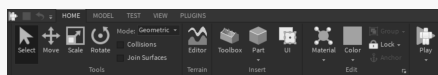


Home Tab



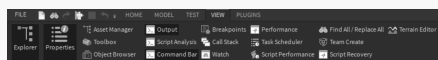
The Home tab is where we make and edit Parts.

Expanding the **Part** button shows other Part types.

To edit the Part, use these buttons: **Select**, **Move**, **Scale** and **Rotate**.

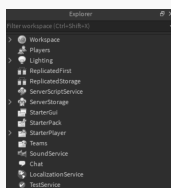
The **Play** button can be used to run the game.

View Tab



The View tab contains all the windows in Roblox Studio. We will mostly be using **Explorer**, **Properties**, **Output** and **Toolbox**.

Explorer Window



The Explorer Window shows the location of your objects in the game. Each Service has its own use. For example, objects located in **Workspace** will be rendered in the 3D space and **Players** contains all the player clients that join the game.

Explorer

game Parent of everything in Explorer

Workspace Used to hold objects that will be rendered in the 3D space

Players List of all Player Clients that join the game

Explorer (cont)

Replicate-dFirst Replicates all objects under this tab to all the Clients (and not the server)

Replicate-dStorage Storage available to Clients and Server

ServerScriptService Storage for ModuleScripts and ServerScripts

ServerStorage Storage available only to the Server

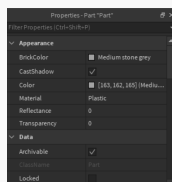
StarterGUI Used to hold GUI objects that will be copied to all clients

StarterPack Used to hold items that are then copied into the Player's backpack

StarterPlayerScripts Used to store LocalScripts for the Player

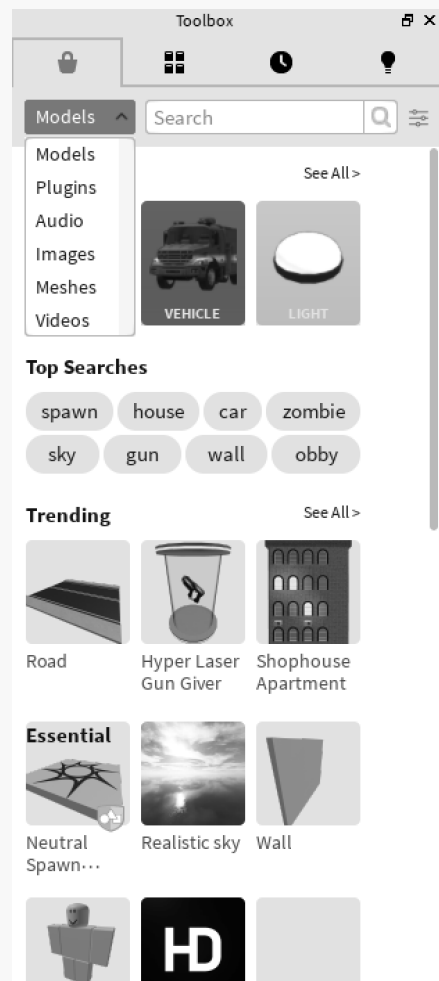
StarterCharacterScripts Used to store LocalScripts for the Player's character

Properties Window



The Properties Window shows the properties/attributes available to the object. Select an object to show its properties.

Toolbox



There are a lot of resources available in the Toolbox like Models, Images, Audio, etc.

Objects

Part A physical brick in the world

Model A container for Parts

Script A container for *Lua* source code that is run on the Server

LocalScript A container for *Lua* source code that is run on a Client



By immortaltfmous

Not published yet.

Last updated 30th June, 2022.

Page 1 of 3.

Sponsored by [ApolloPad.com](https://apollopod.com)

Everyone has a novel in them. Finish Yours!

<https://apollopod.com>

Variables

```
myNumber = 17
myName = " Cat hy"
print( "My name is ", myName,
      "and I'm ", myNumber)
```

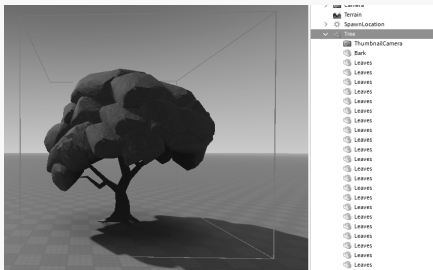
Variables can be used to store anything from numbers, strings and other objects. You do not need to specify the type of variable (i.e. int, String, bool).

Referencing Objects

```
-- create a variable for a Part
located in the Workspace
part = game.Workspace.Part
*--[
create a variable for a Script
in StarterPlayerScripts
      (Which is
located in StarterPlayer)
]*
script = game.StarterPlayer.StarterPlayerScripts
```

To reference an object, use a period to go through the hierarchy (Parent to Child). Referencing is similar to finding a pathway to the object.

Models



Models are Parts that are grouped together. It makes it easier to move objects that consists of a lot of objects like a Tree (which can have Trunk Parts, Leaf Parts, etc).

Part Properties

```
-- declare a variable for the
Part
part = game.Workspace.Part
-- changes the Part's name (Name
shown in Workspace)
part.Name = "New Part Name"
-- changes the Part's BrickColor
to Colour Name
part.BrickColor = BrickColor.new(
"Colour Name")
-- change the position using the
x, y, and z coordinates
part.Position = Vector3.new(x,
y, z)
-- change the size using x, y,
and z coordinates
part.Size = Vector3.new(x, y,
z)
-- anchors the Part so it cannot
be moved around
part.Anchored = true
```

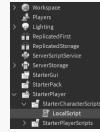
Here are some common properties used for Parts

Creating an Object

```
newObject = Instance.new("Part")
--create a new object, Part,
called newObject
newObject.Name = " myObject"
--assigns a name to the new
object
newObject.Parent = game.Workspace
--assigns a Parent to the object
secondObject = newObject:Clone()
--clones the original object
newObject:Destroy()
--destroys the object
```

Here's some general code on how to create a new object.

Parent vs Child



A child is an object that is under a Parent (another object). In this image, the LocalScript is a Child of StarterCharacterScripts. Easiest way to tell if something has a child is to see if you can expand it (arrow to the left side). Anything that comes up after you expand something is the child of that object.

Operators

| | |
|----|--------------------------|
| == | Equals to |
| ~= | Not Equal |
| > | Greater than |
| < | Less than |
| >= | Greater than or equal to |
| <= | Less than or equal to |

| | |
|---|----------------|
| + | Add |
| - | Subtract |
| * | Multiplication |
| / | Division |
| ^ | Exponentiation |
| % | Modulus |

Function

| | |
|-------------------------------------|---|
| <code>wait(10)</code> | Waits for 10 seconds |
| <code>print(" Hello World!")</code> | Prints the message in the Output window |



By immortaltfmous

cheatography.com/immortaltfmous/

Not published yet.

Last updated 30th June, 2022.

Page 2 of 3.

Sponsored by [ApolloPad.com](https://apollopod.com)

Everyone has a novel in them. Finish

Yours!

<https://apollopod.com>

Custom Functions

```
-- this function adds 2 numbers
function sum(num1, num2)
    print(num1 + num2)
end
sum(1, 2)
-- assigns a variable to the
result of the function
function calculateSquare(n)
    return n * n
end
result = calculateSquare(3)
```

Conditional Statements

```
if
workspace:FindFirstChild("Tree")
then
    print("There is a
tree here.")
end
if coins < 5 then
    print("You need more
money.")
else
    print("You have
enough money!")
end
if player.Name == "Jake" then
    print("You are an
awesome guy, Jake")
elseif player.Name == "Sally"
then
    print("You are a
sweetheart, Sally")
else
    print("You are a
pretty cool person")
end
```

If statements will run their code if the value between **if/then** is true (or not nil). They can be one **else** block, or any number of **elseif** blocks.

Loops

```
i = 0
while i < 10 do
    i += 1
end
--while loop, adds 1 to i until
i is greater than 10
while true do
    print("while loop")
    wait(1)
end
--while loop, infinite loop
since it is always true
for i = 1, 10 do
    print(i)
end
--for loop, prints i until i
reaches 10
for i = 0, 10, 2 do
    print(i)
end
--for loop, prints i until i
reaches 10 (i adds 2 each time)
```

Player vs Character vs Humanoid

Player The Player's Client, stores information relating to the player's account (UserID, SpawnLocation, etc)

Character The Player's physical character in the 3D world. It is a model and contains all the Player's body parts (HumanoidRootPart, Head, etc)

Player vs Character vs Humanoid (cont)

Humanoid A child of the Player's character. Includes properties such as Health, JumpHeight, WalkSpeed, etc.



By immortalfmous

cheatography.com/immortalfmous/

Not published yet.

Last updated 30th June, 2022.

Page 3 of 3.

Sponsored by [ApolloPad.com](https://apollopad.com)

Everyone has a novel in them. Finish Yours!

<https://apollopad.com>