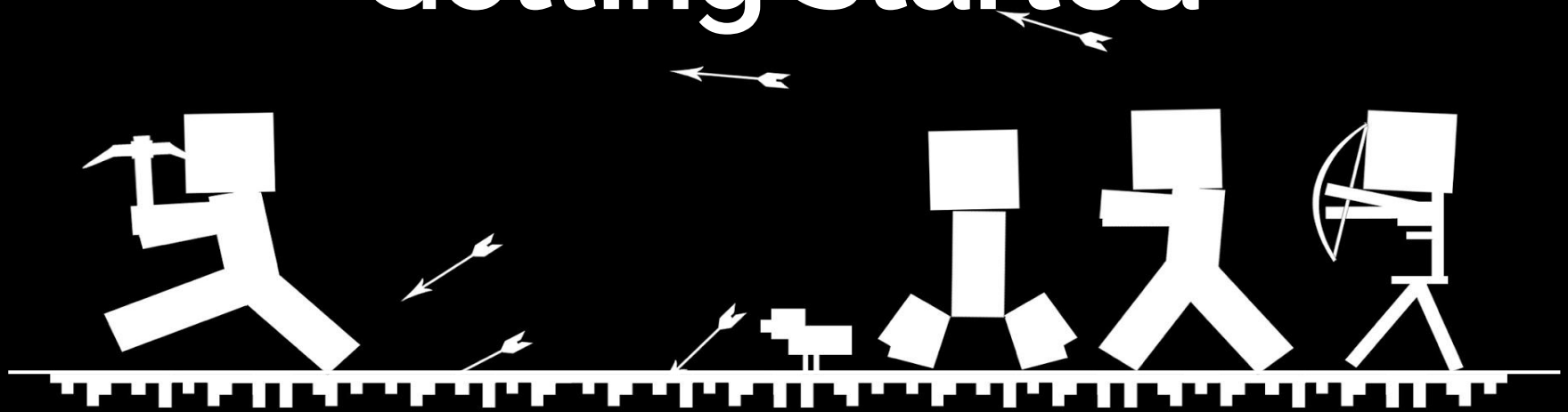
Writing Plugins in Minecraft with JavaScript

Getting Started

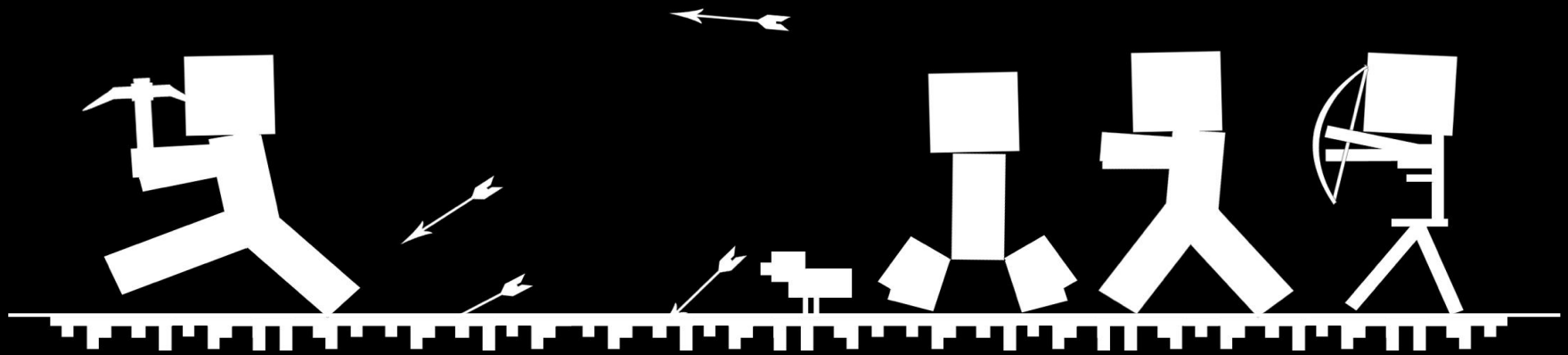
Getting Started



What You'll Need

1. Minecraft *(installed and running)*
2. Java *(if you have Minecraft running, then java is already installed)*
3. Class source code: https://github.com/LetsCodeBlacksburg/ScriptCraft *(download this to your desktop and unzip it)*
4. Text editor *(Sublime Text is recommended)*

Installing CanaryMod



Find your OS's launcher script

In `ScriptCraft-master/`:

- Windows: `Windows/run.bat`
- Mac: `Mac/start_server.command`
- Linux: `Linux/canarymod.sh`

Copy this file into your `server/` directory

Making the script executable (Mac/Linux only)

Open the terminal and type the following:

```
cd ~/Desktop/ScriptCraft-master/server
```

Mac: `chmod a+x ./start_server.command`

Linux: `chmod a+x ./canarymod.sh`

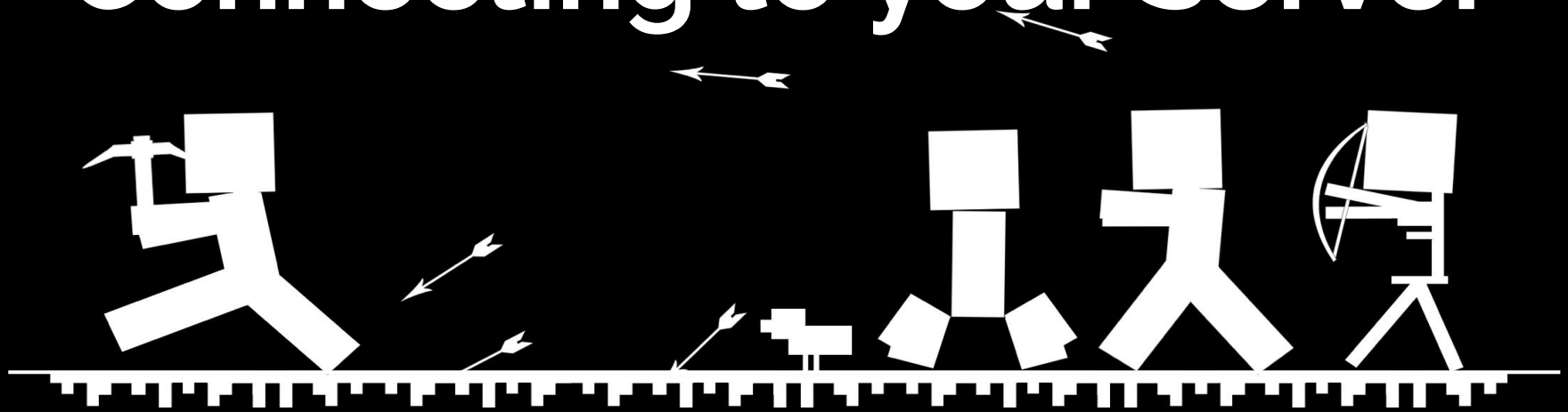
Accepting the EULA

Your server won't start until you accept the terms of use.

In your text editor:

- open `ScriptCraft-master/server/eula.txt`
- change `eula=false` to `eula=true`

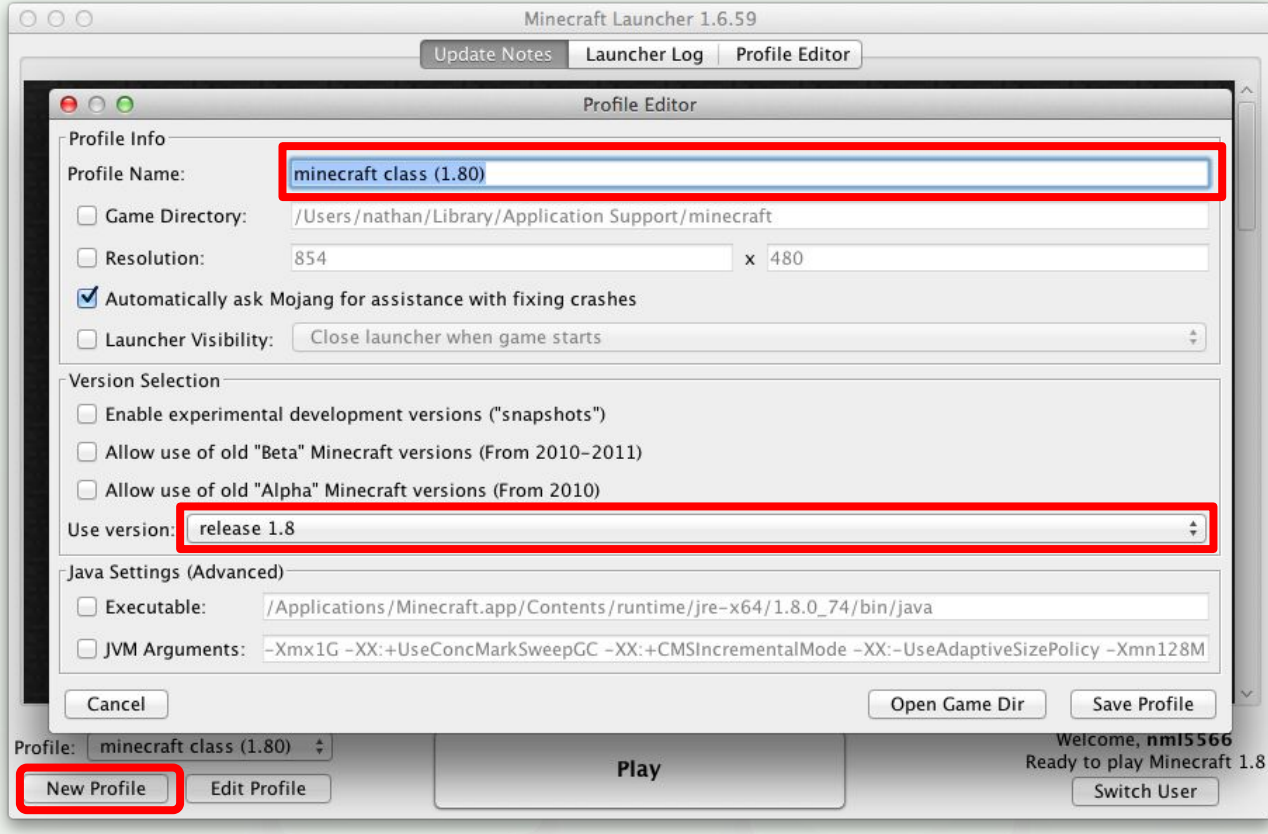
Connecting to your Server



Minecraft Profile Editor

Click on the *New Profile* button

Create a profile that uses release version 1.8.0



Connecting Client to Server

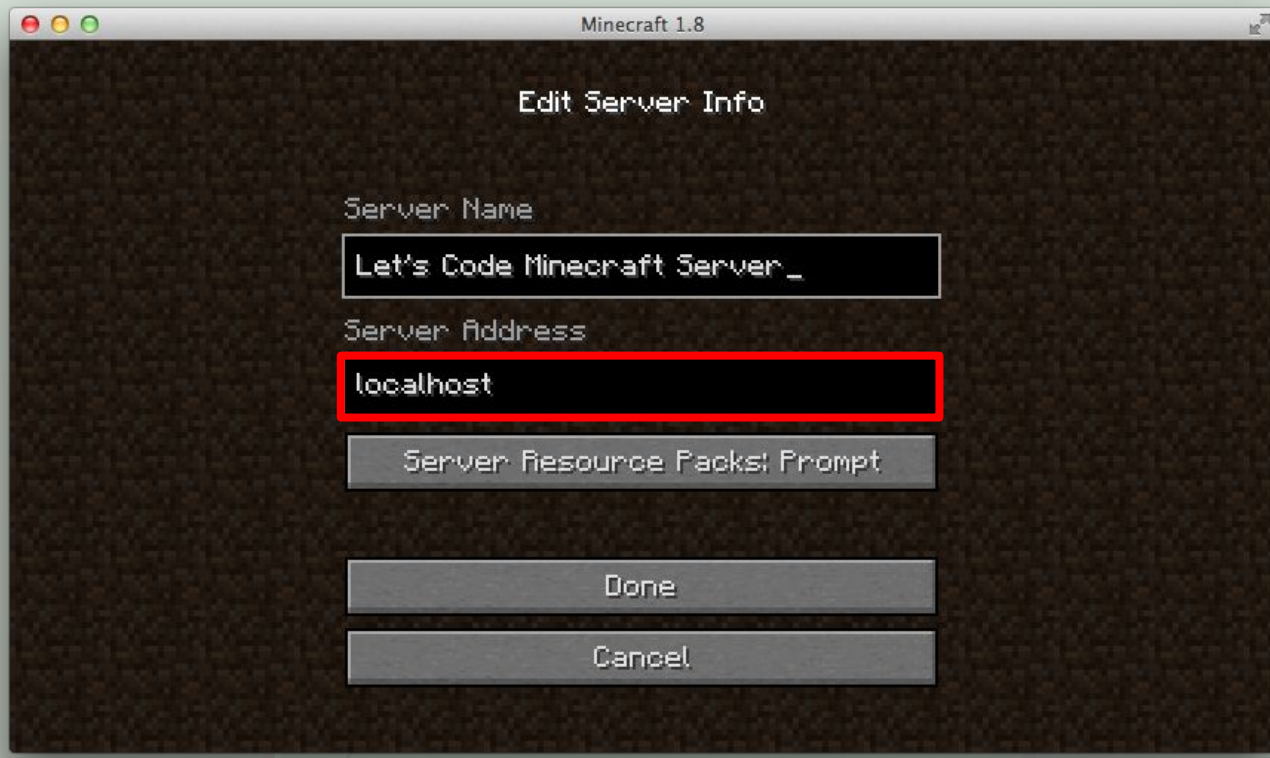
Launch the game and click on *Multiplayer*.

Next, click *Add Server* and type your server's name

Adding Your Server

Give your server a distinct name.

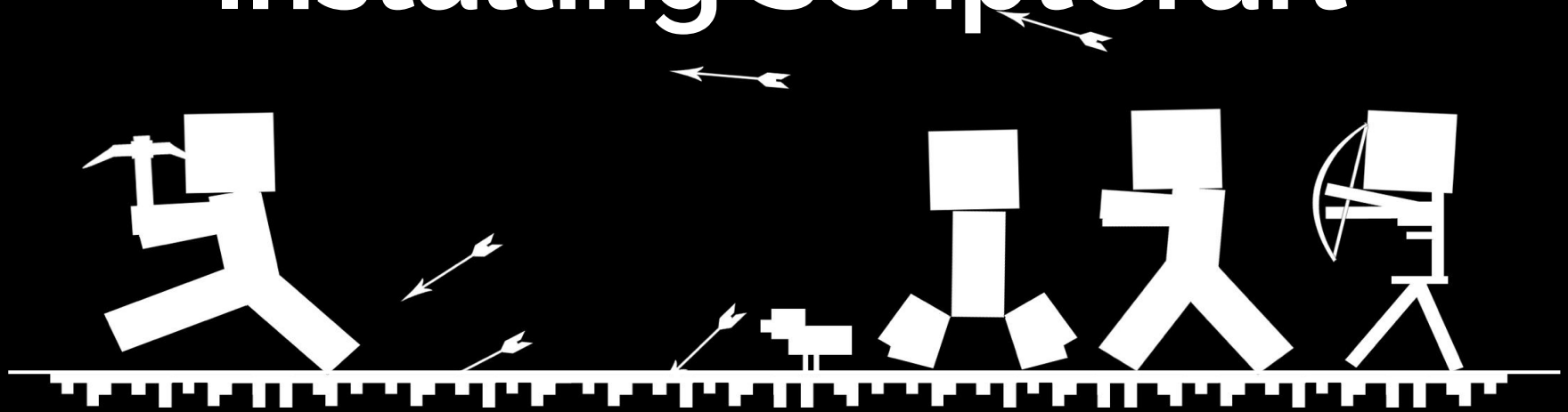
Type `localhost` in the *Server Address* field.



Joining Your Server

Try clicking *Refresh* if nothing shows up.

Installing ScriptCraft



Adding ScriptCraft to Plugins

In `ScriptCraft-master/plugins/`:

1. Find `scriptcraft.jar`
2. Copy this file into your `server/plugins/` directory
3. Restart the server
   (type the `stop` command into the server console, then relaunch it by double-clicking on your startup script)

```
>stop
[10:50:10] [CanaryMod] [INFO] [NOTICE]: Console issued a manual shutdown
[10:50:10] [net.minecraft.server.MinecraftServer] [INFO]: Stopping server
...
[10:50:10] [CanaryMod] [INFO]: Disabling Plugins ...
$
```

Verifying ScriptCraft is Installed

Type the following command exactly into the server console:

```
js "Hello world"
```

The server console will also print the following:

```
[19:22:21] [CanaryMod] [INFO]: Enabling plugin ScriptCraft
```

Giving yourself OP

This is necessary to run JavaScript commands in-game and break blocks. You can only do this *after* you've logged into your server.

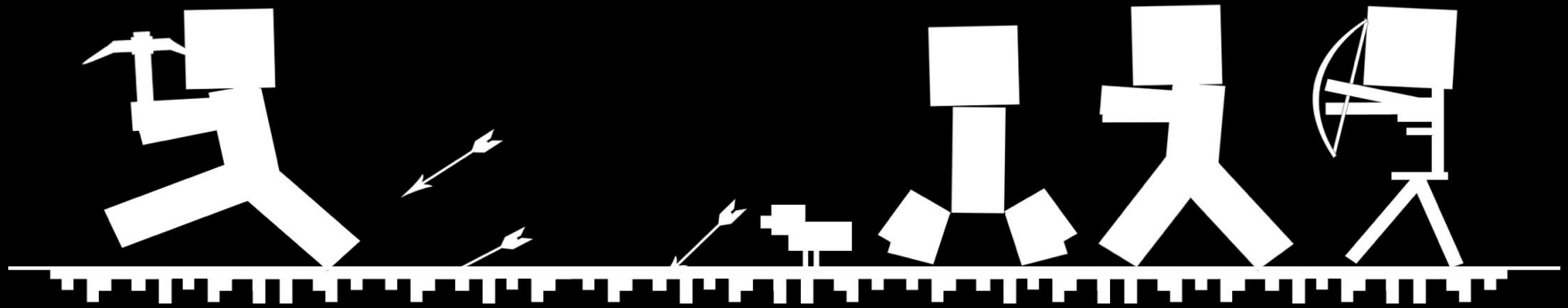
Type the following command exactly into the server console:

```
op <username>
```

The server console will print the following:

```
[11:31:09] [CanaryMod] [INFO]: [SERVER] Opped <username>
```

Exploring JavaScript in Minecraft



Running Commands

Console VS Client

- Javascript commands run on in-game (on the client) must always start with a / (to open the chat window)
- Console commands don't need the /
- Not every command works in both

Basic Math

Javascript can act as a calculator:

```
js 2 + 3
```

```
js 2 * 3
```

```
js 2 - 3
```

It can also compare numbers:

```
js 3 > 5
```

```
js 3 < 5
```

```
js 3 == 5
```

Storing Data in Variables

Start with a variable:	`js var hearts`
Set it to a value:	`js hearts = 8`
Check the current value:	`js hearts`
Change the value:	`js hearts = 9`
Do math with it:	`js hearts + 5`
	`js hearts - 2`
	`js hearts * 1`
	`js hearts / 3`

NOTE: variables can't begin with numbers

Strings

```
js "double string"

js 'single string'

js 'I\'m an escaped string'

js "Here’s a \"double-quote\" escaped string"

js "I'm un-escaped"

js var healthMessage = 'You have ' + hearts + " health remaining"
```

The null Keyword

```js
var hearts = null
```

null means “no value”. It’s useful for marking that a variable is empty.

This is different from *undefined*, which is the default initial setting for any declared variable.

Adding and Subtracting

```
js hungerBar = hungerBar + 1

js hungerBar += 1

js hungerBar = hungerBar + 1

js ++hungerBar

js hungerBar--
```

Data Types

- Number
- String
- Boolean
- Object
- Undefined
- Function

- `js typeof false`
- `js typeof true`
- `js typeof 5`
- `js typeof 9.99`
- `js typeof 'Hello'`
- `js typeof "5"`
- `js typeof console`
- `js typeof Herobrine`
- `js typeof parseInt`

Functions

Collections of code that can be easily called and reused.

Values passed in between the **(** and **)** called *parameters*.

```
js parseInt('4 hours until sunset')

js parseInt('This is not a number')

js parseInt('3 blind mice')
```

Writing Your Own Functions

Type the following on one line *:

```
js function add(first, second) { return first + second; }
```

* NOTE: If you get the error below, just ignore it

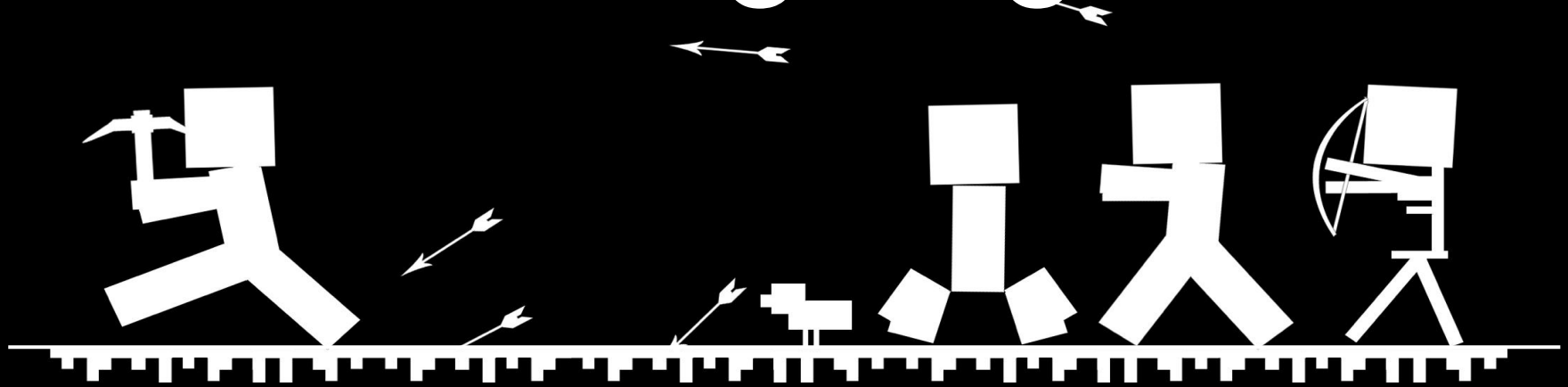
Error: InternalError: Cannot convert NaN to java.util.Iterator (<Unknown source>#415)

Call your new function:

```
js add(5, 6)

js add(9, 1)
```

Creating Plugins



Your First Minecraft Plugin

In `ScriptCraft-master/server/scriptcraft/plugins/`:

- create a new folder called `learning/`
- use your text editor to create a file inside `learning/` called `helloWorld.js`

Add the following inside your file:

```
console.log('Hello World');
```

Save your file, then type the following in the server console:

```
js refresh()
```

Making Your Code Reusable

Let's put our helloWorld.js code into a function:

```
function helloWorld() {
    console.log('Hello World');
}
```

And refresh our server:

```
js refresh()
```

What Happened to Our Message?

Add the new code and refresh:

```
function helloWorld() {
    console.log('Hello World');
}
helloWorld();
```

Making helloWorld() public

To call functions directly, we must first export them:

```
function helloWorld() {
    console.log('Hello World');
}
helloWorld();
exports.helloWorld = helloWorld();
```

Objects

Can hold other variables and functions (called *properties*) accessible via dot notation. `exports` is an example of this. `self` is another example that refers to you, the player.

Try this in-game (note the slash in front of the command):

```
/js self.health = 10
```

```
/js self.invisible = true
```

```
/js self.hunger = 10
```