Coding in Minecraft - Building Stuff

Today we're going to start coding in Minecraft.

We're going to create buildings with code written in Javascript.

If you finished the KhanAcademy Javascript course we started last month, then you'll have an idea how to code in Javascript.

We're going to run through seven short lessons. We'll start simple, and end up building a skyscraper. Awesomes.

Setup

You'll need three things:

- A computer
- A copy of Minecraft
- A text editor. Try these:
  - Mac? Use TextWrangler (http://www.barebones.com/products/textwrangler/download.html)
  - Windows? Use Notepad++ (http://notepad-plus-plus.org/download/v6.6.8.html)

Here's how things will work:

- You'll run Minecraft and connect to a Minecraft server we're running. It's special because it lets you create stuff using Javascript.
- You'll write small Javascript programs in a text-file called 'lessons.js'. This file will be saved to a special directory that lives on the Minecraft server too. This means the server can see your programs, and let you run them inside Minecraft.

Connecting to the Minecraft server

Follow these instructions to connect to the Minecraft server

- Run Minecraft on your computer
- Create a new profile called 'CoderDojo'. Make sure it uses version 1.6.4
- Select the 'CoderDojo' profile, and hit 'Play'
- Then select 'Multiplayer', then 'Direct Connect'
- Enter the server address specified by a CoderDojo mentor. It should be something like '192.168.0.18'

Check you can run Javascript inside Minecraft

- Once you've connected to the server, hit the / key
- And type..

```
js 1 + 1
```

- then hit return
- if the answer is 2.0, then you're running Javascript :-)

Now let's learn how to code in Minecraft..

Coding in Minecraft - lessons.js

Follow these instructions to start coding Javascript programs in Minecraft.

You'll need to connect to the special directory where your Minecraft programs will be stored (in a text-file called 'lessons.js')

If your computer is a Mac:

- open the Finder, open the 'Go' menu and select 'Connect to Server'
- then enter the directory address specified by a CoderDojo mentor. It should be something like 'smb://192.168.0.18/players'
- if asked for a username, select 'guest' (should be no password)

If your computer is running Windows:

- open Windows explorer, and type into the address bar the directory address specified by a CoderDojo mentor. It should be something like '\192.168.0.18\players'
- if asked for a username, select 'guest' (should be no password)

Now look for a sub-directory with your Minecraft username.

- Click on it. This is the directory where your 'lessons.js' text-file will go.

Let's set up lessons.js..

- Open your text-editor (TextWrangler on Mac, or Notepad++ on Windows)
- Start a new document.
- Type in the following Javascript program (exactly as it looks)

```
exports.lesson1 = function() {
    echo('This is my first Minecraft program :-)');
};
```

- Now save the document as 'lessons.js' somewhere you can find it (like the desktop)
- Close the text-editor.
- Now drag the 'lessons.js' file to the special directory we opened earlier: the one with your Minecraft username.
- Now open this file with your text-editor (just double-click on it)

Now switch back to Minecraft.

- Open the prompt again by typing /
- Now type...

```
js your-username.lesson();
# BUT USE YOUR REAL USERNAME!
# I'm greghuc, so I'd type: js greghuc.lesson1();
```

- Hit return
- If Minecraft said 'This is my first Minecraft program :-)', then it's all good.

YOU'VE JUST RUN CODE IN MINECRAFT - ROCK ON!

Or if it's all gone horribly wrong, then ask a CoderDojo mentor for help :-)

Lessons

Right, let's get on with learning how to build cool stuff in Minecraft.

You already did Lesson 1 - running your first Javascript program inside Minecraft.

Lesson 2 - building abox

When you did the KhanAcademy Javascript course, you got to create 2d shapes like circles and squares. We're going to do something similar in Minecraft - except we'll be creating 3d shapes instead, like boxes and cylinders.

Before we start, you'll be using something called 'Drone'. A Drone is like a special person who knows how to create different 3d shapes. You create a new 'Drone', and then tell it what shapes you want to create.

Let's start by creating a small box. In your lessons.js file, add this Javascript code to the bottom (exactly as it looks):

```
//Lesson 2: create box
exports.lesson2 = function() {
    var drone = new Drone();
    drone.box( blocks.iron );
};
```

Now save lessons.js, and..

- Switch back to Minecraft
- POINT THE CROSSHAIRS AT THE HORIZON (NOT THE GROUND)
- Open the prompt by typing /
- Type..

```
js your-username.lesson2();
```

- Hit return

You should have created an iron box just in front of you. Cool!

Lesson 2 - challenge

You'll learn later about different shapes you can create. For now, let's learn about some new materials:

- blocks.snow
- blocks.glass
- blocks.lava

Your challenge is... Change the lesson2 function you just typed in to create a box out of lava. And run it!

Lesson 3 - setting sizes of a shape

The 'box' function you just used created an box of width 1, height 1, and depth 1. But you can specify different sizes for the box (and for other shapes).

Coding time again. Same as last time - add this code to the bottom of lessons.js, save it, then run it inside Minecraft with

```
js your-username.lesson3();
```

AND REMEMBER TO POINT THE CROSSHAIRS AT THE HORIZON (NOT THE GROUND).

```
//Lesson 3 - create box with dimensions
exports.lesson3 = function() {
    var drone = new Drone();
    drone.box( blocks.snow, 3, 2, 1 );
};
```

If this ran ok in Minecraft, you should be looking at box made of snow.

Take a look at the size of the box. It should be 3 across (width), 2 high (height), and 1 block deep (depth).

Now take a look at how this matches up with the drone.box function: we're telling the drone to create a box of a particular material (blocks.snow), and a particular width (3), height (2) and depth (1).

Lesson 3 - challenge

Your challenge is... Change the lesson3 function to create a blocks.tnt box with depth 2, width 4, and height 3. And run it!

Lesson 4 - using function arguments

The last lesson let you create a box with certain dimensions. But it's a hassle to change the

dimensions: you have to go back into lessons.js, change the width, height and depth sizes, save lessons.js, go back into Minecraft, and run your function from the prompt...

But there's a better way.

Add this to the bottom of lessons.js and save it. BUT DON'T RUN IT YET!

```
//Lesson 4: create cylinder with parameter dimensions
exports.lesson4 = function(radius, height) {
    var drone = new Drone();
    drone.cylinder0( blocks.stone, radius, height );
};
```

This function creates a hollow cylinder out of stone. And you specify its radius and height dimensions.

Now let's pay attention to how we call the lesson4 function:

Back in Minecraft..

- POINT THE CROSSHAIRS AT THE HORIZON (NOT THE GROUND).
- And run...

```
js your-username.lesson4(4,2);
```

- A hollow cylinder should appear in front of you with radius 4 and height 2.
- Now point the crosshairs elsewhere and run..

```
js your-username.lesson4(5, 10);
```

- A hollow cylinder should appear in front of you with radius 5 and height 10.

So from inside Minecraft, it's now much easier to create hollow cylinders of different sizes with the lesson4 function: we just call it with different 'radius' and 'height' numbers. The proper Javascript name for the values we give when calling a function is 'arguments'.

Function arguments are pretty nifty. It means that the same function can behave in different ways, depending on the arguments given. With our lesson4 function, we're creating cylinders of different sizes. And we didn't need to update the code in lessons.js!

Lesson 4 - challenge

Your challenge is... Change the lesson4 function to create a blocks.glass hollow cylinder. Then call the function with arguments: radius 4 and height 1.

Lesson 5 - moving the drone around

It's been a bit mysterious so far where exactly the drone starts creating a shape. It's somewhere near you, but not clear where.

We can improve on that. In this lesson, we'll see how:

- you can tell the drone where to start creating a shape
- you can move the drone around from its starting location.

To cut a long story short, if you point the Minecraft crosshairs at an occupied tile (like a ground tile), the drone starts out there, and will create the shape there.

Let's think about this for a minute. If you point at the ground, the drone starts out in the ground. So the shape it creates is ***in*** the ground, not above it!

Type this in:

```
//Lesson 5: specifying where to build something
exports.lesson5part1 = function() {
    var drone = new Drone();
    drone.box( blocks.iron );
}
```

Switch to Minecraft, and..

- POINT THE CROSSHAIRS AT A GROUND TILE.
- Run the function.
- The ground tile under your crosshairs should now be an iron box.

So now we can tell the drone where to build stuff. Except it's stuck in the ground.

NO IT'S NOT! You can move the drone around. So we can move the drone up before it creates a shape.

Type this in and run it (pointing crosshairs at a ground tile):

```
//Lesson 5: specifying where to build something
exports.lesson5part2 = function() {
    var drone = new Drone();
    drone.up(2);
    drone.box( blocks.iron );
}
```

OMG floating box!

The drone started in the ground tile pointed at by your crosshairs, but then we told it to move up 2 blocks. So it created the box 2 blocks above the ground tile :-)

You can tell the drone to move in different directions. Try the functions: up, down, fwd, back, left, right and turn. All these functions take one argument, that tells the drone how many times to move.

Lesson 5 - challenge

Your challenge is... Change the lesson5part2 function to create a box 4 blocks up, and 3 to the right.

Lesson 6 - repeating stuff

Creating a shape just once is boring. Imagine the cool stuff we could do if we created something lots of times.

Type this in and run it (pointing crosshairs at a ground tile):

```
//Lesson 6: create a shape multiple times
exports.lesson6part1 = function() {
    var drone = new Drone();
    drone.up(1);
    drone.fwd(1);
    drone.box( blocks.tnt, 3, 1, 2 );
    drone.up(1);
    drone.fwd(1);
    drone.box( blocks.tnt, 3, 1, 2 );
    drone.up(1);
    drone.fwd(1);
    drone.box( blocks.tnt, 3, 1, 2 );
}
```

Cool - a staircase of TNT! But imagine we wanted a staircase with 50 steps. That would takes ages to type. But there's a better way - using a loop.

Type this in, but don't run it yet.

```
//Lesson 6: create a shape multiple times, with a loop
exports.lesson6part2 = function(numberOfSteps) {
    var drone = new Drone();
    var counter = 1;
    while (counter <= numberOfSteps) {
        drone.up(1);
        drone.fwd(1);
        drone.box( blocks.tnt, 3, 1, 2 );

        counter = counter + 1;
    }
    echo('Drew staircase!');
}
```

Back in Minecraft, point at the ground, and run the function with:

```
js your-username.lesson6part2(50);
```

Wow - staircase with 50 steps!

Take a look at the function. We didn't need to type in all the commands to create 50 different steps. Instead, we typed in the commands for just one step, then repeated the commands 50 times using a 'while' loop.

Let's look at this more. The function takes a 'numberOfSteps' argument - we set it to 50 when

we ran the function. Inside the function, we create a new drone, and a 'counter' variable set to 1. Now for the 'while' command:

- The while command checks the 'condition' in brackets: (counter <= numberOfSteps). In English, this means: 'is the value of counter less than or equal to the value of numberOfSteps'? So is (1 <= 50)? True or false? The condition is true.
- If the condition is true, the while command runs all the commands inside the squiggly brackets (they look like '{' and '}'). In our case, the drone creates a single step in the staircase. We also set the value of counter to be 'counter plus 1'. So if counter is 1, it will become 2. The while command then 'loops' back to the start, and checks the condition again: is (counter <= numberOfSteps)? This time round, is (2 <= 50)? The condition is still true. So we draw another step. We keep drawing steps until....
- If the condition becomes false, the while command ***does not*** run the commands inside '{' and '}'. So we stop drawing steps, and move onto the command afterwards (the echo command). When is (counter <= numberOfSteps) false? When the value of counter is larger than numberOfSteps (50): when counter is 51. So calling your-username.lesson6part2(50) makes the drone draw 50 steps, and then stop.

Lesson 6 - challenge

Draw a staircase with 100 steps.

Lesson 7 - skyscrapers

Let's put together what we've learnt and draw a skyscraper.

Type this in, and run it. You know the drill.

```
//Lesson 7: skyscraper
exports.lesson7 = function(levels){
    var drone = new Drone();
    drone.up();

    var counter = 1;
    while (counter <= levels) {
        drone.box(blocks.iron, 20, 1, 20);
        drone.up();
        drone.box0(blocks.glass_pane,20,3,20)
        drone.up(3);

        counter = counter + 1;
    }
};
```

Tada - a skyscraper with 10 levels! The only new bits are the new shape of box0: it's a hollow box. And blocks.glass_pane is a new type of material.

Lesson 7 - challenge

Go build stuff!

There's still lots to learn about Javascript and Minecraft, but here's some shapes, materials and drone movements to start with.

```
# Create a drone
var drone = new Drone();
```

```
# Create shapes
drone.box(material, width, height, depth) => box
drone.box0(material, width, height, depth) => hollow box
drone.prism(material, width, length) => prism (like a roof)
drone.prism0(material, width, length) => hollow prism
drone.cylinder(material, radius, height) => cylinder
drone.cylinder0(material, radius, height) => hollow cylinder
```

```
# Materials (there's way more than this!)
blocks.stone
blocks.grass
blocks.wood
blocks.glass
blocks.gold
blocks.chest
blocks.ice
blocks.pumpkin
blocks.tnt
blocks.water
blocks.lava
```

```
# Move the drone
drone.fwd(steps)
drone.back(steps)
drone.up(steps)
drone.down(steps)
drone.left(steps)
drone.right(steps)
drone.turn(turns90Degrees)
```

```
# Building random stuff
drone.door() => single door
drone.door2() => double door
drone.oak() => oak tree
drone.spruce() => spruce tree
drone.birch() => birch tree
drone.jungle() => jungle tree
```