Tutorial 1 - Halloween Treasure Hunt

Table Of Contents

This tutorial will treat the following concepts:

- Denizen objects and general syntax.
- Browsing through the documentation.
- Reloading scripts and debug modes.
- World script containers, events, switches and context tags.
- Common script commands and arguments.
- Constructing tags with simple bases and modifiers.
- First contact with foreach loops.

Introduction

Welcome to the **Denizen2Sponge Scripting Engine**. This is our first tutorial, so we'll start with the basics. After downloading the plugin and running the server once to create the needed files, the first thing we need to do is locate the Denizen **scripts folder** inside *config/denizen/*. This is where we'll create our script files, which have the *.yml* extension. Now we have to familiarize a bit with Denizen syntax, and there's no better place for that than the documentation. We'll focus on some of the most common ingredients we'll be using in our scripts:

- **Script Containers:** As their name suggests, they're meant to store scripts. There are different types, depending on the nature of the script we want to make. They consist of a **name** key, a *type:* subkey and value and some other subkeys that belong to each type. You can also specify the level of debugging of a script container through the *debug:* subkey.

- **Commands:** These are placed after a `-` in executable parts of our scripts and are in charge of making something happen in the Minecraft server. From giving items to players to creating artificial explosions, commands will be a key component of any script.

- **Tags:** These are text placeholders that Denizen replaces with objects at runtime. They are always wrapped in `< >` and consits of a tag base, which retrieves or

creates an object; and modifiers, which transform the original base into other types of objects. As many modifiers as you want can be concatenated inside a tag, as long as they are always separated by a `.` and their input object type matches the previous output type.

- **Events:** These are placed inside world script containers and their job is to listen to actions happening within our Minecraft server. When triggered, they'll execute their script section, filled with commands and tags. Events have special tags that are stored within the **context** base. These unique tags offer information directly attached to the action that took place, such as the location when an explosion occurs.

First Steps

Now let's jump (or slowly crawl) into our first script. In our case, we want to make a **Halloween treasure hunt event** in our Hub world. We plan on manually placing some hidden pumpkins ourselves, and reward players for finding (and left clicking) them. Note: this tutorial assumes our Hub world is in fact called *Hub*.

First of all, we'll need to create a new script file (for example *New_Script.yml*) and build a world script container, which listens to events happening within our server. We just have to give it a name, like *Halloween_Treasure_Hunt*, and set the *type:* key as *world*. In addition to this, we'll also set the *debug:* key as *full*. This will make our script print in the console everything it does and help us solve errors. It is not needed, so we can just remove it later on. We'll now place an events: subkey, which will hold the executable code we're going to write.

It's worth noting that Denizen scripts follow **YAML**'s indentation rules, with either **2** or **4-space** tabs. We'll be using **D2IDE** ourselves, so this tutorial's code will be formatting according to this text editor. You can also just use **Notepad++** until the specialized IDE is officially released, but you'll have to make sure the tabs are replaced with actual spaces. There's a settings option in that automatically does that for us, so no problem.

Now we need to find an event that fits our case. Looking through the event documentation list, *on player left clicks block* looks like our best bet. We'll just test it for now, so we add it under the events key and end the line with a `:`. Inside this event we can place our first command. A good option for testing purposes is just to show a word in console. The command for this is *echo*. As the documentation explains, its syntax is very simple. We feel the hype, so we'll go with an *- echo "yay"*.

Our script should be the following at this point:

```
Halloween_Treasure_Hunt:
  type: world
  debug: full
  events:
    on player left clicks block:
    - echo "yay"
```

It's time to **save the script** file, **reload scripts** ingame with */ex reload* and trigger the event by left clicking a block. We should now be able to see a cute little *yay* (along with some debug information) in the console, just as we expected. That's great, but the event is being triggered no matter what block type we click on, and that's not ideal.

Core Functionality

Our next step should be to limit this so it works with pumpkins only. The best way to achieve that is with event **switches**. Switches are internal checks that decide whether or not an event should fire. Each event has different available switches, so make sure to read the documentation. We can see there's a *type* switch that checks the broken block type, so we'll use that. Adding *type:pumpkin* right after the original event (and before the ending colon) should work for us.

We should not forget that we don't want players to get rewards from pumpkins in other worlds, so why not use another switch? According to the same documentation, our event has a *world* switch, so it's as easy as adding *world:Hub* to the event line as well.

The updated event is now:

```
    on player left clicks block type:pumpkin world:Hub:
```

We should save, reload and test again now. After a bit of testing, we've noticed clicking any block in any world no longer echoes *yay* to the console, except for **pumpkin** ones in the **Hub** world. Perfect!

We're ready to move further ahead and actually give a reward to the player clicking the block. Since we're nice server owners, the prize will be a free **diamond**. This is when the *give* command comes in handy. Its syntax requires two **arguments**: a **player** and an **item**.

When reading command documentation, It's important to keep in mind that arguments inside `< >` are **not literal** and need to be replaced, while *`[ ]`* means an argument is **optional**.

As we explained earlier, events have specific information that can be accessed through the **context** tag. For example, player events are always linked to the **player** that triggered them. This player object can be easily accessed through the *<context.player>* tag, although this is a special case and we can just use *<player>* as a shortcut. This player object will be the target of the **give** command. For the second argument, just specifying *diamond* will be enough.

The full command line will then be:

```
    - give <player> diamond
```

Now it's time to make sure it works. After **saving** and **reloading scripts** again, it should be giving us a **diamond** everytime we click the **pumpkin**. While players will totally love this, we should probably avoid giving out unlimited diamonds.

That's easy to fix though, we just have to remove the pumpkin once it's clicked. If we do it before even giving out the reward, we make sure it won't be clicked twice. We'll use the *setblock* command (syntax here), which requires a **location** and a **block type**. Yet again, we'll use a **context** tag to retrieve the location, in this case *<context.location>*. The block type, on the other hand, will be just *air* as we want to remove the original pumpkin.

Our script with these new commands should look like this:

```
Halloween_Treasure_Hunt:
  type: world
  debug: full
  events:
    on player left clicks block type:pumpkin world:Hub:
    - echo "yay"
    - setblock <context.location> air
    - give <player> diamond
```

Rinse and repeat: save, reload scripts and do a quick test. Amazing! This deserves a "yay". Speaking of yays… we don't need to echo *yay* anymore, so we better remove that.

Topping It Off

Let's make it even more fun. What if **jack o' lanterns** gave a diamond to **every online player**? Yeah, we can make that happen too! Let's start by making a copy of the event we already have and its contents. We now want to change the *type:* switch from *pumpkin* to *lit_pumpkin*.

Inside the event, we need to wrap the give command with a *foreach* loop block (with syntax). This loop takes a list when it **starts** and executes some commands **for every object on the list**. In our case, the list of online players can be accessed through *<server.online_players>*.

Then inside the foreach, we can retrieve the currently looped object with *<def[foreach_value]>*. Replace *<player>* in the give command with this new object tag and we're ready to go.

Here's the complete second event:

```
    on player left clicks block type:lit_pumpkin world:Hub:
    - setblock <context.location> air
    - foreach start <server.online_players>:
      - give <def[foreach_value]> diamond
```

We'll also let the players know they have received a reward. For the first event we can use the *tell* command. Its syntax is familiar by now, so we'll go ahead and write:

```
    - tell <player> "You've found a pumpkin! Here's your reward!"
```

But we can make it even fancier. In the second event we'll be using *announce* instead so everybody knows who their new hero is. According to its syntax, this command only needs a message argument, but we'd like to know the **name** of the player who found the hidden block. Well, that's not a problem. We already know the player who clicked the block can be retrieved with *<player>*, so why not use a **tag modifier** to know this information? Our command would be as easy as this:

```
    - announce "<player.name> has found a jack o' lantern. Everybody
gets a reward!"
```

We just have to make sure it works as intended, and finally set the *debug:* key to *minimal* so only error messages are shown. No more console spam!

Finally, this is the script we've created:

```
Halloween_Treasure_Hunt:
  type: world
  debug: minimal
  events:
    on player left clicks block type:pumpkin world:Hub:
    - setblock <context.location> air
    - give <player> diamond
    - tell <player> "You've found a pumpkin! Here's your reward!"

    on player left clicks block type:lit_pumpkin world:Hub:
    - setblock <context.location> air
    - foreach start <server.online_players>:
      - give <def[foreach_value]> diamond
    - announce "<player.name> has found a jack o' lantern. Everybody
gets a reward!"
```

This should be it for now. Enjoy your brand new Halloween treasure hunt event and **happy scripting**!