2D Navigation Multi-User Template

AKA *LikeLike Template User Guide*, *v1*
by Hand Eye Society (Brendan Lehman and Len Predko)

Table of Contents

Acknowledgements

Hand Eye Society
Brendan Lehman – Guide, Template, Super FESTival
Len Predko – Template, Super FESTival
Jae Stuart – Super FESTival
Jordan Sparks – Super FESTival

LIKELIKE
Paolo Pedercini – LIKELIKE Online

Biome Collective
Niall Moody – Biome Gallery
James Morwood – Biome Gallery
Matthew Cormack – Biome Gallery

MacKenzie Art Gallery
Cat Bluemke – Digital Exhibitions Consultant
Jonathan Carroll – Digital Exhibitions Consultant

Introduction

Hello! Welcome to the LikeLike Template User Guide! This document will help you through some of the many ways to use the LikeLike platform to host live events and shows online. It is written with three paths of increasing difficulty, depending on how much you want to customize the space and interactivity.

The easy path involves copying the template and updating the links, keeping the default room layout and art. No serious experience is required.

The medium path adds customization and decoration of the rooms. Some knowledge of image editing software and JSON is helpful here.

The hard path goes over customizing and creating game mechanics to really make the space yours. Knowledge of Node.js, socket.io, and coding in Javascript is recommended.

The platform relies on a website called Glitch.

The guide references the Glitch project `likelike-template`.

I Setup

1 Overview

In this section we'll go over getting set up to work on the project.

For the easy path, you can skip part 3 Local Development as you'll be able to do everything on Glitch.

For the medium path, doing part 3 is optional. If you know your way around, go for it but you can also do everything on Glitch as well.

For the hard path, setting up a local development environment is highly recommended. It'll save a lot of time and energy versus trying to code in the browser.

2 Glitch

Head over to Glitch and make an account using your sign-on method of choice.

Once you're logged in, go to the template project.

Glitch's term for duplicating or cloning a project is 'Remix'. Click the Remix button on the template page to make your own to work with.

Once loaded, the first thing to do is to change the project name. Go to Settings > Edit project details and enter a new name under PROJECT NAME. Click Save.

The Glitch edit interface allows everything you need to do to work on the project, including a code editor, terminal, version control, and asset handling. As you might expect with a web interface, it has limitations. The next step outlines how to set up a local development environment to work on your project out of the browser if you are planning on making bigger changes.

Now you're ready to customize! Check out Section II to learn about what all the files you just copied are, Section III to learn about what the files do, or skip straight to Section IV to start working.

3 Local Development

Working locally is great for many reasons, especially for speeding up your workflow if you're planning on making big changes or customizations.

In this guide we'll use VS Code. It's a great lightweight IDE with a lot of extensions. Give it a download for your platform of choice.

Node.js

The event platform application uses a runtime called Node.js. It creates a space on a webpage for more advanced code to run in a browser.

Download the node.js installer and follow the instructions. This will install node.js on your machine as well as npm (Node Package Manager) which will help install the Glitch project to work on.

Glitch Project

The template project is automatically set up as a Git repository and includes a library called express which allows it to be installed with its dependencies (other packages of code it needs to work). ref

First we'll get the code from Glitch. Navigate to your new project's edit page. On the bottom toolbar, click through *Tools* > *Import/Export*. Press *Copy* beside the project's Git URL. This will copy the address of the git repository that Glitch uses to store your project.

Create a folder for your projects to live on your computer. Launch VS Code and open the folder you just created. Open a Terminal through *View* > *Terminal* or press *Ctrl*+`.

Verify you are in your new project folder. Type `git clone` and paste the git URL you copied from the Glitch project. Press Enter and wait for the project to clone.

Still in the Terminal, change directory into the newly created project folder with cd `myProjectFolder`.

Next, run `npm install`. Once complete, run `npm start`. Some text should appear in the Terminal.

To test if it worked, open a browser and go to `https://127.0.0.1:3000` . If it did, you'll see the template project load!

Further instructions for setting up a parallel Git repository independent of Glitch can be found here.

Workflow

From here you are free to make changes to the files. When you save files the nodemon package will automatically restart your node project to load your changes.

Each commit you make to Git as you work will synchronize the project with Glitch. It's good to switch over to Glitch to test your work on the web version as different bugs than the local version can sometimes arise.

Some quirks exist when working with multiple people on the project and some are working locally and some with the Glitch workflow. If someone is working in the Glitch editor, or leaves the glitch editor open in a browser tab, local code cannot be pushed as the Glitch editor leaves changes unstaged until a particular session is complete. If this happens, you'll have to send a merge commit as well after eventually pulling the changes you were waiting for. Stash, pull, and pop often!

More specific details on the workflow can be found in Section IV.

II File Structure

1 Overview

The folders and files of the project are set up in the standard client-server method used for Node.js projects on Glitch. Server files are on the project root and client files are in a `/public` folder, with our project's media assets (images, audio, etc) and other files for configuration.
To illustrate how the project is set up we'll use the metaphor of an event.

2 Server Files

Server files contain the code that run on Glitch and are shared among each user of the app. Using our event metaphor, you can think of the server code as the venue our event is in.

server.js

This file is the core of the project. It determines the behaviour of the space and the shared features of the platform. It loads all the code packages the project uses, in particular the socket.io package that handles communication with clients/users. It also loads the game state, room, follower, and

NPC data, as well as chat filters and admin functions. Big changes to this file are generally not needed as it works with the JSON files that contain these customizable data.

serverMod.js

This file extends server.js with specific interactivity functions that allow custom games and player interactions with rooms and objects. Functions may include checking certain conditions (player inventory, progress flags) and changing dialogue or making other objects appear based on these conditions. This file is critical for adding quests and pickup items. We'll cover its use more thoroughly in Section IV: Build Flow.

3 Client Files

Client files contain the code that runs in the browser of each player, communicates with the server, and determines what appears on-screen. Sticking to our metaphor, these files are, of course, our event attendee.

client.js

The `client.js` file communicates with the server to determine what to draw to the screen, including the rooms (backgrounds and objects, other players, any text appearing (from the chat or environment), and the audio playing. It includes the classic "game loop" as well as objects for players and text bubbles, and many functions to affect change on-screen. In particular, the `newGame()` function sets up what the client will do upon receiving specific messages from the server through *socket.io*. These rules include critical functions of the software including updating the position of the player and other players in the room, and displaying chat text. *socket.io* allows this communication of events to occur in (effectively) real-time. We'll play with this file plenty in Section IV: Build Flow.

clientMod.js

Similar to `serverMod.js`, this file extends `client.js` beyond the base functions to script behaviours for specific custom conditions. Data is often passed back and forth to the `serverMod.js` code. For example, entering a room will trigger `serverMod.js` to update a progress flag, which `clientMod.js` will react to by changing the spoken text of NPCs. Working with this file is covered in Section IV: Build Flow as well.

4 Media Assets

Working with media assets is critical if you are taking the medium or hard paths and want to customize the platform. Media assets are available to clients and are loaded in-browser to display.

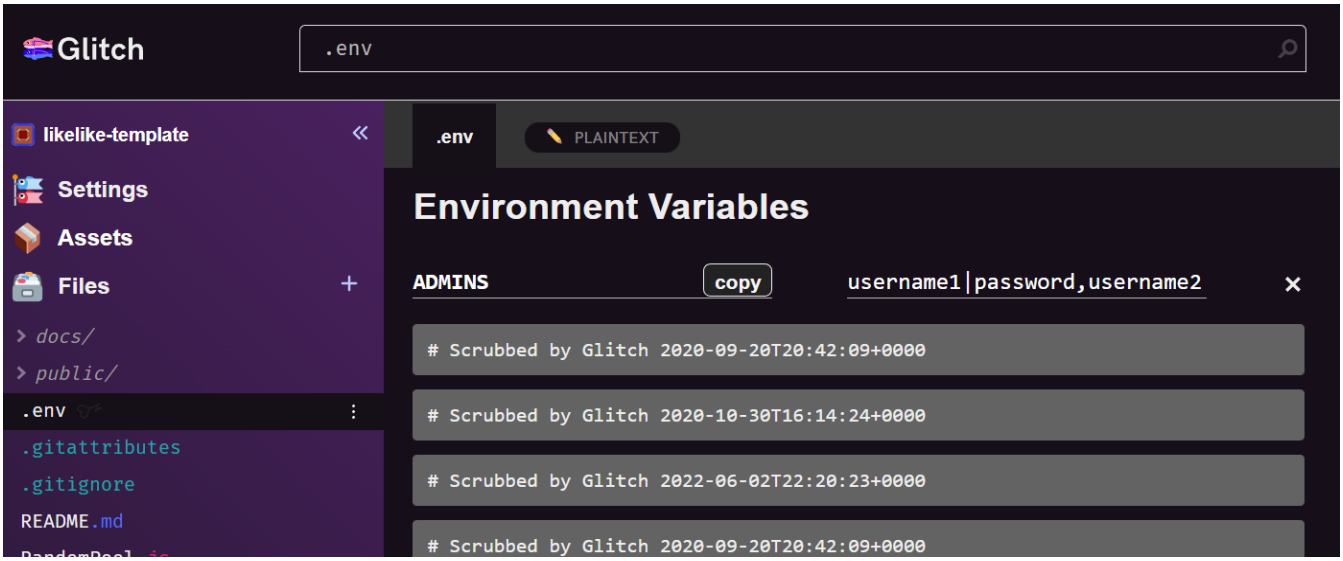
Asset Handling

There are a few considerations when working with assets in Glitch. Primarily, all assets must be uploaded through the browser interface into the Assets section. We recommend this even if you're working locally, so you don't have to update their reference URLs. You can drag-and-drop assets into the Asset section from your computer or use the UPLOAD AN ASSET button.
Once uploaded, clicking on the asset will bring up its URL. Copy this URL to use when customizing your space and referencing these assets in JSON files, etc; either in the browser code editor or locally. Locations where you'll use these URLs will come up in Section IV: Build Flow.

5 .env File

The .env file is a special file specific to each place you're working on the project. It keeps track of secure information. Specifically it is the place where you set the information about the admin users and their passwords.
In Glitch, one is created for you. In it, you can set the admins and their passwords.
In a local project, you need to manually create one. In the project's root folder you can create a new file in VS Code and name it .env.

You can set your admins with the format in the following image. To set multiple admins, repeat the format and separate using a comma with no spaces.



More information about admin functions can be found in Section V, Part 3: Mod Tools.

6 Legacy Code

This section is legacy code from previous LikeLike projects in /public. By the time of publication, we aim to have made this section redundant.

III Current Features

1 Overview

This section will outline the current features of the platform, and give some examples of how they are or could be used. Since we talked about the back-end in Section II, this section is more about the front-end and the user facing features. The platform experience is centred around rooms as a unit of space, each containing server-side rules for what is drawn to the screen, what is available to do for each player, and what NPCs are hanging out. On the client-side, the platform supports live chat and chat commands for emoting, customizing, and interacting with the rooms, as well as time-based events, pickup items, and a player inventory.

2 Core Features

Rooms & Objects

Rooms are the core unit of space in this platform. They set up where the player can go and what they can do, so they are essential to understand for working with this platform. A room is represented by a `.json` file in the `/public/rooms` folder, and requires a background image, an areas image, and can include any number of "things" to decorate the room and create some basic interactivity. The details of setup are down in Section IV: Build Flow § 2 Basic Build.

The background image is the main graphic for the room and is commonly drawn to replicate the walls of a room with some perspective, however this is not mandatory and can be used to create many different spatial possibilities (i.e. zoomed out for city streets, or a solar system).

The areas image is a colour-coded map used by the platform to determine the areas of the screen that can be collided with or clicked on. Paired with the rules in the room's JSON file, this is the core of the platform's navigation and a big part of the interactivity (along with the chat commands).

"Things" are objects that can be placed in the room to decorate it but also display text and/or external links (i.e. an arcade cabinet that links to games on showcase). Background images and things can be animated, and is covered in Section IV: Build Flow § 3 Aesthetic Customization.

The room areas and the Things in the room are the elements that get the point-and-click mechanic to function. Mouse interactions with these elements, including mouse over and click, can show text, open links, or run more code. The features of this additional code is explained in 3 Game Mechanics.

Chat and Commands

The platform features a live chat where text input by the player is "spoken" in a speech bubble and appears to all others in the room.
The chat also supports certain commands preceded by a backslash. For example, a player can enter /dance into their chat box and their avatar will emote on a loop until another action is taken. The following is a list of the basic commands

Basic Command List

`/help`	Resets the player's position in the room
`/size`	Changes the player avatar size
`/dance`	Emote on a loop
`/reset`	Return the player avatar to normal
`/decoration`	Turn the player into a Thing
`/woo`	Make a woo! sound
`/clap`	Make a clap sound
`/vibe`	Vibrate
`/say`	Uses a text-to-speech engine to dictate the text entered

The full list can be found in the project folder at `/docs/commands.md`.

3 Game Mechanics

Interactivity Overview

Various interactivity mechanics are supported to extend the basic chat command and point-and-click inputs.

The back-end of calling custom functions, and how to get started making your own, is covered in Section IV: 4 Interactivity Customization.

NPCs

The platform supports non-player characters (NPCs) in a couple of ways. Primarily, they can be set up to appear as a player would with a chosen avatar. Their behaviour can be customized including what they say, and how they move around the room. The other way to set up an NPC is to draw them into the background images and add interactions through the room JSON and areas image. In both cases their speech can be customized, and their behaviour can be extended to unlocking things in the room such as Pickup Things or hidden doors.

Image caption: A regular NPC in the lobby.

Image caption: Hand Eye Society mascot Bob is drawn into the background and interacted with as an `areaColor`.

Command Triggers

Custom functions can be run with chat commands that interact with the world. For example, the `/laugh` command in Super FESTival's comedy club reveals a gem.

Image caption: Having a laugh in the Super FESTival Comedy Club

Timers

Player actions can trigger timers that do something after a certain amount of time. For example, a player has to `/dance` for 5 seconds to get the gem.

Image caption: Feel the rhythm and get a gem at the Super FESTival Rave

Pickups & Inventory

The pickup mechanic adds the checking for and tracking of certain Things in the world. The gems are an example of this mechanic. They can appear based on certain triggers, and are stored client-side in an inventory that can be checked with /inventory.

Image caption, left to right: Finding and picking up the Moonlight Catcher. Can you find the Gallery Attendant's keys in the template?

Inventory

The inventory is an array of objects unique to each player (client-side) that keeps track of the items they have picked up while playing. The list can be checked when an action is attempted to determine if the action can be completed and can be modified if any item is removed. The player can check their inventory with the chat command /inventory. The gems are the best example of this – collecting them in the world and placing them when interacting with their pedestals.

Image caption: Checking inventory with `/inventory` in Super FESTival.

Room Scope

Rooms can be designed with elements/events/actions/mechanics that are visible only to the player (client-side) or shared between players (server-side). Understanding these scopes can be used in creative ways to design more engaging and interactive rooms. For example, there is a room with a sleeping monster that occasionally wakes up. In this room, the monster wakes up at the same time for everyone in that room, but checking what to do if a player moves while the monster is awake happens for each player individually. Another example is the room where the gems are placed on their pedestals. Anything that happens in this room is only visible to the player taking the actions.

Image caption: Monster asleep, monster awake, monster angry. The monster's state is updated on the server so it's shared with everyone simultaneously and everyone who moves while the monster is awake gets teleported to another room.

Progress Flags and Questing

These flags keep track of a player's progression through pre-defined sequences (i.e. a questline). They are booleans that are set upon certain actions and can be checked to determine the state of the world for the player. These states may include Thing and/or room area visibility, and NPC dialogue.

IV Build Flow

1 Overview

This section will guide you through the actual steps to work with and customize the platform for your show.

2 Basic Build will cover the simplest way to modify the template for your own show, with the least amount of experience required.

3 Aesthetic Customization will build on this by covering how to create and work with art assets and how to use them to make the showcase more visually unique.

4 Mechanic Customization will go even deeper and explain how to use the existing mechanics and how to create your own for turning the venue into a game itself.

Our starting point requires completing Section I: 2 Glitch to have a project ready to work with.

2 Basic Build

This basic build flow will use the super cute template gallery we created and use it visually as-is, updating the arcade cabinet info and links to the games you want to showcase. The first floor of the template project is the standard gallery. There's an entrance, a lobby, and two rooms on either side. Each room has a game cabinet and a gallery attendant, except the entrance which doesn't have a cabinet. The lobby also has a didactic on the wall. We can change the games that the cabinets link to, what the attendants say, and the didactic. Once you've done these, jump up to Section V: Going Live !

Room Data

Let's look at the lobby first. We can customize both the didactic and the cabinets at the same time by changing the *room data*. You can read more about this in Section II: 2 Core Features § Rooms & Objects. Open up the file `/public/rooms/lobby.json`.

For the didactic, we can look under the `areaColors` section for the block with the `didactic` comment. Here we can update the text. If you have a lot to say, you can break it up with line breaks (`\n`) and specify how many lines of black background you need by updating the value for `lines`. Save your changes and reload the project to try it out. Play with it a bit until it feels right.

For the lobby cabinet, we can scroll down the `lobby.json` file a bit until we get to the things section and the `cabinet` block. Here we can update the info, in particular:
`label` is the text that will display when you hover your mouse over the cabinet.
`txt` is more descriptive text that pops up when you click on the cabinet. Like the didactic you can edit how many lines your text will be with line breaks and changing the value of `lines`. Game name, developer name, a blurb, or anything else you want to write can go here.
`url` is of course the link to the game. This can be any URL but if you have itch.io or Steam URLs this is where they go!

NPC Data

You can also quickly change what the NPC gallery attendants say. Open up the `serverMod.js` file in the root folder. The first function in the file, called `module.exports.initMod` contains what we want. Starting on line 26, we can see which NPCs are saying what. Edit here at your leisure. Try to keep the lines short so they fit on the screen; you can't set the number of lines here like with the didactic or the cabinets.

3 Aesthetic Customization

This chapter covers some deeper customization, notably how to add your own art and music assets and do more with the rooms and app audio.

As Google Docs isn't our favourite for sharing code, and for convenience while working in the project environment, this chapter can be found as a markdown file in the Glitch template project at `/docs/how-to-add-a-room.md`.

Avatars

Additionally, if you're curious about the avatars and possibly modifying them or adding new ones, you can read up about it at the `/docs/avatar-guide.md` markdown doc.

Splash Image

You can create your own splash image. Use our `splash.png` or `Logo_blink.png` as an example. It can be animated if you want. Put your new logo in the Glitch project assets.
Some things in `client.js` must be updated if you want to change your logo. The URL must be updated at `client.js:93` . If your new logo is animated, you can change the dimensions and frame count at `client.js:408` and the frame delay at `client.js:410` .

4 Mechanic Customization

Likewise to the previous chapter, this one can also be found as markdown in the project at `/docs/how-this-works.md`.

Commands

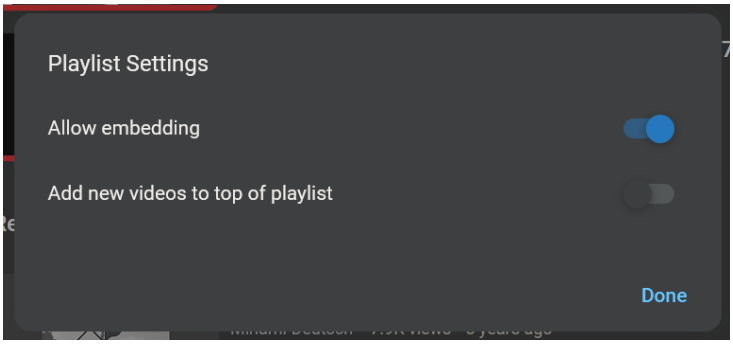
The current list of chat commands and information on how to modify and add to them is within the `/docs/commands.md markdown doc.`

5 YouTube and Twitch

On the second floor are example rooms for embedding external video content in a kind of theatre/cinema. We've done YouTube and Twitch. Players on the page are created and destroyed with room event functions in `clientMod.js`.

YouTube

To get the YouTube player linked up with your content, first organize your videos on YT into a playlist and set to at least *Unlisted* permission.
On the page for that playlist, navigate through the context menu to the 'Playlist Settings' and verify that the 'Allow Embedding' switch is ON.



Still on the playlist page, copy the code at the end of the URL from your browser's location bar.

Put this code into the `id:` field at `data.js:16`. That's it! Your playlist will load up when a player enters the room with the player.

Twitch

There are a couple quick things you need to do to get Twitch to embed inside a room.

At `data.js:11`, update the sites in the twitchParent setting with any external websites you might embed your LikeLike in. This could be your own website or another for the event.

Next, update `clientMod.js:80` with the channel name you want to embed. This is probably the channel you're streaming on. That's it! Whatever is scheduled for that channel will play when a player enters the room.

Additionally, the chat of the channel specified will appear in the page border around the LikeLike frame (replacing the command list when in the Twitch room).

Our Twitch embed code is current with this Twitch documentation in May 2023. They update the embed API as security practice changes so this may break in a few years.

V Going Live

1 Overview

Congratulations on getting the content complete stage! There are still a couple things we should go over before showtime. We need to update our webpage headers so everything shows up nice in search engines and social media previews, set up Glitch for the extra public traffic, and learn about the admin and moderation tools built into the template.

2 Header Info and Metadata

Title and Favicon

The first and easiest thing to do is update the page's title at `index.html:5`. This will update the text in the browser tab and on the taskbar/dock.
To update the icon that appears on the tab, you can make your own. It's a 16x16 pixel PNG you can create pixel art-style, or make a slightly bigger image and crunch it down with a good algorithm (Photoshop, GIMP, etc). Put the icon in the Glitch project assets, get the URL, and paste it over the URL at `index.html:6`.

```
5     <title>LikeLike Template</title>
6     <link rel="icon" type="image/x-icon" href="https://cdn.glitch.global/028048c3-4b00-495b-8199-069e60bba66c/Favicon.png?v=1665632817584">
```

SEO and Social Media

To get your project properly indexed by search engines and display the right thing in social media previews we need to set some `<meta>` tags. Update `index.html:10-27` to your preference.

Other social media sites not shown here may have their own formats. Quickly Googling for your social media's preview meta tags will find what you need.

```
<!-- SEO / social media-->
<meta name="title" content="LikeLike Template" />
<meta name="description" content="A tiny MMO exhibition template" />
<meta name="author" content="Hand Eye Society, Biome Collective, Molleindustria, MacKenzie Art Gallery" />
<meta name="copyright" content="LGPLv2.1" />
<meta name="robots" content="index, follow" />
<meta name="description" content="A tiny MMO exhibition template" />
<meta name="keywords" content="Game,Art,gallery,videogame,digtal,exhibition,hand eye" />

<meta property="og:image" content="https://cdn.glitch.global/028048c3-4b00-495b-8199-069e60bba66c/preview_image.png?v=1683819990210" />
<meta property="og:title" content="LikeLike Template" />
<meta property="og:description" content="A tiny MMO exhibition template" />

<meta name="twitter:card" content="summary" />
<meta name="twitter:site" content="@handeyesociety" />
<meta name="twitter:title" content="LikeLike Template" />
<meta name="twitter:description" content="A tiny MMO exhibition template" />
<meta name="twitter:image" content="https://cdn.glitch.global/028048c3-4b00-495b-8199-069e60bba66c/preview_image.png?v=1683819990210" />
```

3 Glitch

Depending on how many people you're expecting to attend your event, we recommend making some changes to the project on Glitch to make it a nicer time.

If you're expecting more than 20 concurrent active users, we suggest paying for Glitch Pro for the months your show is up or the months you have events focusing on it to boost your server CPU and memory capacity. Server lag can build up quickly on the free plan, especially if you've done some customizing and have added a lot of rooms and assets and a bunch of people are trying to load it at once. It's $8 USD per month.

Glitch usually supports being pointed at by custom domains, but currently has a backend issue preventing this. Keep an eye on this linked page if you find yourself here in the future.

4 Mod Tools

The LikeLike platform has a few admin and mod tools built-in that you can use.

First, you can set users to be admins with the .env file. Remember we chatted about it back in Section II: 5 .env File.

To log in as an admin, enter the username|password combo into the name prompt as-is (including the bar and exposed password).

Being an admin lets you do extra things within the game space. On the UI you can click to see an analytics page with a simple visitors per day chart. You can also see a player list that you can refresh.

There are special chat commands available to admins for moderating players among other things. There is a full list available in the commands markdown doc in the Glitch project at `/docs/commands.md`. Some of the important ones are listed here.

Basic Admin Commands List

`/stats`	Display (to the admin who called it) some stats about the number of players on the server, and where they are
`/list-players <room>`	Display (to the admin who called it) a list of players in the named room
`/god <message>`	Send a message to every player
`/kick <player-name>`	Kick the named player
`/mute <player-name>`	Mute the named player
`/unmute <player-name>`	Unmute the named player
`/popup <player_name> <message>`	Trigger a popup message to the named player
`/ban <player-name>`	Ban the IP of the named player
`/unban`	Unbans *all* banned players (some tech debt here: need to modify this to un-ban specific players)
`/nuke`	Disconnect all players

Info on how to add new admin commands can be found in the `/docs/commands.md` file in the project.

Outro

You made it! You're now more than ready to put up a little thing and have a show. Good luck with your event! We'd love to hear from you should you have any questions, comments, shows to promote, or nerdy stuff to link. Get at us!

See you out there :D

HES

Editor note: this document can be viewed in its original format at the link available here.