Mod Configs Implementation Guide

By Snek

This doc serves to show how to use Mod Configs in your mod.

Adding a Config:

Adding a config is the first thing you'll want to do.
You'll need a JavaScript file - either an existing one in your mod or a new one just for this.

If you don't already have one, you'll need an "immediately executing anonymous function", which just means it runs as soon as it's made.

You_TestConfig.js

```
(function () {

})();
```

Now that you have that, you need somewhere in that function to add the config itself. Unfortunately, it doesn't seem to work if you put it straight into the function, so you'll have to hijack/"hook" another function to add the config.
In my testing, the best* function to use for this is **`Scene_OmoriTitleScreen.prototype.create`**, so go ahead and hook into that.
**aka the only one I tested*

You_TestConfig.js

```
(function () {
    oldCreate = Scene_OmoriTitleScreen.prototype.create;
    Scene_OmoriTitleScreen.prototype.create = function () {
        oldCreate.call(this);
    };
})();
```

In short, the added code tacks all the code of this **`create`** function onto the end of a new one, which means it'll run when the title screen is created.

Now that all that's done, you can actually write the code that matters / isn't just setup.

First, you need to check if Mod Configs is even active. Fortunately, GOMORI can provide that easily:

You_TestConfig.js

```
(function () {
    oldCreate = Scene_OmoriTitleScreen.prototype.create;
    Scene_OmoriTitleScreen.prototype.create = function () {

        if ($modLoader.mods.get("modconfigs")) {

        }

        oldCreate.call(this);
    };
})();
```

You can do a few things if this check fails depending on if your mod genuinely needs configs or not. For now, though, just don't put an **`else`** clause.

All configs are created using the following function: **`Snek.ModConfigs.addConfig(header, options, helpText)`**. The check for Mod Configs being active ensures that this method exists.
It has three arguments:

- header: The name of this config, which is used to reference it later. Must be a string.
- options: An array of the possible options. Must be an array, and must have at least one element. It *can* have as many elements as you want, but be aware that adding too many will cause options to go off-screen.
- helpText: The text to display at the bottom of the Configs menu when this config is selected. Must be a string.

The arguments are given in that order, so adding a config would go like this:

You_TestConfig.js

```
(function () {
    oldCreate = Scene_OmoriTitleScreen.prototype.create;
    Scene_OmoriTitleScreen.prototype.create = function () {

        if ($modLoader.mods.get("modconfigs")) {
            Snek.ModConfigs.addConfig({
                header: 'Test Config - Remove Jumpscare',
                options: ['NO', 'YES'],
                helpText: 'Should the jumpscare added by this mod be
removed?'
            });
        }

        oldCreate.call(this);
    };
})();
```

And there you have it: You've added a config! Granted, it doesn't *do* anything yet, but it's something!

Checking a Config:

Checking a config is comparatively easier / requires less setup. You'll still need access to JavaScript, though.

Configs are checked with the following method: **`Snek.ModConfigs.checkConfig(header)`**. It takes in the header defined earlier, and returns the config if it exists. Make sure the header is exact: Capitalization matters.

The returned config has two bits of data that are of use:

- config.index, which is the zero-based index of the selected option (first option is 0, second is 1, etc.)
- config.options, which is just the list of options you gave the config. Not always useful, but options that need the value selected outright can use it.

So let's say you use the config for removing a jumpscare. Your map event would look something like this…

```
◆Comment : Options: ['NO', 'YES'], so 0 means the jumpscare won't be
:        : removed.
◆If : Script : Snek.ModConfigs.checkConfig('Test Config - Remove
Jumpscare').index === 0
 ◆Comment : Do jumpscare
: End
◆
```

And that's it! You can now make and use configs all you want.