Other

- Default Options
- Changing Window Title and Icon
- Loading Assets and Data
- Changing Mod Display Names
- KubeJS 6.1 Update

Default Options

You can ship default options from options.txt with KubeJS. This includes keybindings, video settings, enabled resource packs, controls like autojump and toggle sprint and wierd things like advanced tooltips.

Why use this instead of just shipping options.txt? If you ship options.txt then the users options will get overridden every time they update your modpack, where-as KubeJS only sets the options once, on the first time the modpack boots.

To use it simply make a file called `defaultoptions.txt` in the `kubejs/config` folder. Then copy any lines you want to set by default over from the normal options.txt file. You can also just copy the entire file if you want to include everything.

A full list of what options the options.txt file can contain is available on the Minecraft Wiki:

https://minecraft.fandom.com/wiki/Options.txt

Changing Window Title and Icon

Yes, you can do that with KubeJS too.

> To change title, all you have to do is change `title` in `kubejs/config/client.properties`.

> To change icon, you create a `kubejs/config/packicon.png` image in standard Minecraft texture size preferably (64x64, 128x128, 256x256, that kind of size).

> The image has to be saved as 32-bit PNG, not Auto-detect/24-bit, otherwise you will get a JVM crash!

Here's how to do that in PaintNET:

2021-01-14_15.34.54.png

Example result:

2021-01-14_15.37.48.png

2021-01-14_15.37.30.png

> Currently incompatible with Fancy Menu!

Loading Assets and Data

You can also use KubeJS to load assets from resource packs and data from datapacks! While this isn't the only method, its one of the easiest. Other options are <TODO: make and link server datapack load page and client generate assets event page>

The data folder is loaded identically to the data folder in a datapack. If you already have a datapack just copy the folder(s) from inside the datapacks data folder to KubeJS' data folder.

The assets folder is loaded identically to the assets folder in a resourcepack. If you already have a resourcepack just copy the folder(s) from inside the resourcepacks assets folder to KubeJS' assets folder.

Changing Mod Display Names

Yes, it's cursed, but possible!

In a startup script, add this line:

```
Platform.mods.kubejs.name = 'My Modpack Name'
```

This is useful when you add a bunch of items with KubeJS but want them to show your modpack name instead of "KubeJS"

And yes, you can change name of other mods as well:

```
Platform.mods.botania.name = 'Plant Tech Mod'
```

KubeJS 6.1 Update

For script and pack developers

- **Scheduled events now take in durations** (especially strings such as `200 t` for tick durations as well) for their delays!
- **`NetworkEvents.fromServer` and `NetworkEvents.fromClient` have been merged into `NetworkEvents.dataReceived`**, which will handle incoming data from the corresponding side based on the script type.
- **Registry: `event.custom(T)` is now `event.createCustom(() => T)`**, which takes in a *supplier* rather than an object directly in order to avoid possible early loading of other registry elements it might depend on. Note that custom still exists, but is HEAVILY discouraged for this very reason!
- **Event `.cancel()` now exits event block** - This may be a small change but it may affect some scripts. Previously it would only mark event as cancelled and didn't do anything, but now it will act as return; call as well.
- **Event results have been added!** You now have more granular control over how events work, closer to how they are handled on the Architectury / Minecraft side as well! For example:

```
 ItemEvents.rightClicked('minecraft:stick', event => {
    // (note that only one of these will work at a time since they all immediately
return!)
    event.cancel() // cancels the event and prevents the click from going through
    event.success() // cancels the event and forces the click to go through
    event.exit() // cancels the event without setting a result
    // in events that support custom results like item stacks, you can also do the
following:
    event.success('minecraft:apple') // success + the result is an apple □□
 })
```

  Right now, this new system is only actively used for item right click events, but will be expanded to more events as time goes on (obviously without breaking scripts, and just using `event.cancel()` will still work just fine)!
- **Massive backend rewrites, improved performance a lot** - Lat did another pass over the recipe event and has reworked the way recipes are parsed, as well as fixed async recipe operations, so you should generally notice a decrease in reload times if all works as intended! In some cases, recipes should now load even faster with KJS than they do with just vanilla!

- **No more tag workarounds!** (hopefully) - We have fixed resolving tag ingredients during the recipe event on first world load and generally improved the way recipe filters work, so you shouldn't have to use hacky double-reload workarounds anymore (please just... stop using them already :ioa:)
- **Registries have been fixed on both Forge and Fabric** - We have ironed out some issues with the registry events, so you should now again be able to properly register Fluids, modded registries, etc.
- **Renamed kubejs/logs files from .txt to .log** - So you can now have formatting in VSCode, etc.
- **Fixed resource and data pack order** - User added resource packs and datapacks will now be above KJS generated packs, so you should be able to change textures and other things with them.
- **Added .zip loading from kubjes/data and kubejs/assets** - You simply drop a .zip file in that folder and it will force-load it (above KJS, under user packs)
- **Moved `debugInfo` config** from `kubejs/config/common.properties` to `local/kubejsdev.properties`. No idea why it was in common properties in first place, its a *debug* config for devs.
- **Improved `Platform.mods.modid.name = 'Custom Name'`** It should work with custom mod IDs on REI and ModNameTooltip now. You should use `Platform.getInfo('custom_mod_id').name = 'Custom Name'` for non-existent mods.
- **Better recipe integration with ProbeJS** - Because of new schema system in KJS, probe is able to much better display what ingredients go where, with less hacks!
- **.stage(string)** recipe function no longer requires Recipe Stages to work.
- **Fixed flowing fluid textures** on Fabric and other fluid related issues.
- **Fixed errors being way too long in logs** - Believe or not, KJS was not supposed to spit out 150 lines of errors for each recipe.
- **Added a new wrapper `FluidAmounts` for... fluid amounts!** For those of you who can't remember just how many blocks, ingots and nuggets are needed to make a bucket, or who just want to have cross-platform script compatibility with their scripts (since Fabric uses "81000 droplets" rather than "1000 mB" for more precise fluid measurements)
- **Added custom toast notifications** - You can use `player.notify(title)`, `(title, subtitle)` or `(Notification.make(...))`.
- **Added /kubejs reload config command** - No longer you have to restart the game to update configs!
- **Added /kubejs packmode [mode] command** - Same as above, but you don't have to mess with files at all.
- **Added /kubejs help command** - Useful links directly in-game.
- **Removed /kjs_hand command** - Instead added /kjs hand (with space) redirect. Some might hate this change, but _ is much harder to reach than space, and I'm sure you'll get used to it quickly and like it better.
- **Fluid registry .tag() fixed** - Now tags flowing fluids too.
- **You can now replace and match fluids** - You must use `Fluid.of('minecraft:water')` instead of plain string, but you can use it in both `{input: Fluid.of('minecraft:water')}` recipe filter and `event.replaceInput('*', Fluid.of('minecraft:water'), Fluid.of('minecraft:lava'))` replace functions for supported recipe types.

For addon mod developers

- **Complete rewrite of recipe system** - Recipes now use recipe schemas, a new system that (almost) fully replaces the old RecipeJS objects. More on that in the Discord announcement
- **Events now have results** for more precise control over return values and we've added a `hasListeners()` check for performance reasons. The most noticeable change for you is going to be that your own events will need to return a `EventResult`, as well.
- **Fixed datagen issue** - KJS should no longer keep datagens from closing game forever in dev environment. We truly live in an age of wonders!

Update Primer (sorted by topics, still incomplete):

Recipe Schemas

From the announcement:

> This is the big one. Recipe schemas *completely* change the way custom recipe handlers are registered in KubeJS, and should hopefully also mean a lot less boilerplate code down the line for you. Each recipe is now defined by a *schema* containing a set of *recipe components*, with those components acting as "codecs" for the underlying values. For you, this means the following:
> - Instead of primarily using RecipeJS subclasses, you will now have to define a **RecipeSchema**
> - Each schema uses a set of **RecipeKeys**, which are named **RecipeComponents** with some additional properties like optional default values and settings for automatic constructor and method generation
> - A **RecipeComponent** is a reusable definition of a recipe element (such as an in/output item, a fluid, or even just a number value) that has a role (input, output, other), a description (for use in addon mods like ProbeJS) and contains logic for (de)serialisation and bulk recipe operations (i.e. recipe filtering and replacement). There are lots of standard components provided in the `dev.latvian.mods.kubejs.recipe.component` package, including blocks, items and fluids, generic group and logic components (array, map, and, or), and all kinds of primitives (including specialised ones such as number ranges and characters)
> - While the recipe schema *will* generate constructors by default, you can override this behaviour by defining one yourself using `constructor(factory, keys)`. Note that this will stop the default constructor from being generated, so if you want to keep that, you will have to define it yourself again.
> (A good example of complex custom recipe constructors is `ShapedRecipeSchema`)

> - While schemas replace **RecipeJS** on the java side, on the JS side, the user is still passed a **RecipeJS** object after creation, with additional autogenerated "builder" methods for each component to allow for the user to set e.g. optional values after recipe creation (e.g. `event.smelting(...).xp(20).cookingTime(100)`). and you can add even more properties or do additional after-load validation by overriding the recipe factory entirely!

Download

You can download KubeJS 6.1 at https://kubejs.com/downloads!