Addons

Scripts using various KubeJS addons for recipes.

- KubeJS UI
- KubeJS Thermal
- KubeJS Create
- 3rd Party addons
- KJSPKG

KubeJS UI

You can also always look at existing modpack using KubeJS UI to see how they do it

```
onEvent('ui.main_menu', event => {
  event.replace(ui => {
    //ui.background('kubejsui:textures/example_background.png')
    ui.tilingBackground('kubejsui:textures/example_background.png', 256)
    ui.minecraftLogo(30)

    ui.button(b => {
      b.name = 'Test'
      b.x = 10
      b.y = 10
      b.action = 'minecraft:singleplayer'
    })

    ui.button(b => {
      b.name = 'Test but in bottom right corner'
      b.x = ui.width - b.width - 10
      b.y = ui.height - b.height - 10
      b.action = 'https://feed-the-beast.com/'
    })

    ui.label(l => {
      l.name = Text.yellow('FTB Stranded')
      l.x = 2
      l.y = ui.height - 12
      l.action = 'https://feed-the-beast.com/'
    })

    ui.image(i => {
      i.x = (ui.width - 40) / 2
      i.y = (ui.height - 30) / 2
      i.width = 40
      i.height = 30
      i.action = 'https://feed-the-beast.com/'
```

```
    })

    ui.label(l => {
      l.name = Text.aqua('Large label')
      l.x = 100
      l.y = ui.height - 20
      l.height = 15
      l.shadow = true
    })
  })
})
```

KubeJS Thermal

You can use KubeJS Thermal to add recipes to a lot of the machines from the Thermal Series.

Tip: you can use Ctrl/Cmd + F to search this page for the machine you are looking for.

```
onEvent('recipes', event => {
    // Redstone Furnace
    // Turn four coal into one diamond
    event.recipes.thermal.furnace('minecraft:diamond', '4x minecraft:coal')
    // Dried kelp to leather, with a high energy cost
    event.recipes.thermal.furnace('minecraft:leather', 'minecraft:dried_kelp').energy(20000)

    // Sawmill
    // Input one oak leaf and have a 5% chance of an apple, and 10% of a sapling
    event.recipes.thermal.sawmill([Item.of('minecraft:apple').withChance(0.05),
Item.of('minecraft:oak_sapling').withChance(0.1)], 'minecraft:oak_leaves')
    // Turn an acacia slab into 4 buttons
    event.recipes.thermal.sawmill('4x minecraft:acacia_button', 'minecraft:acacia_slab')

    // Pulverizer
    // Turn any leaf block into 4 sticks with a 50% chance of a fifth. Has a low energy cost.
    event.recipes.thermal.pulverizer(Item.of('minecraft:stick').withChance(4.5),
'#minecraft:leaves').energy(100)
    // Pulverise a flint into an iron nugget with a 10% chance of a second
    event.recipes.thermal.pulverizer(Item.of('minecraft:iron_nugget').withChance(1.1),
'minecraft:flint')

    // Induction Smelter
    // Turn one coal block into 4 diamonds with a 50% chance of a fifth
    event.recipes.thermal.smelter(['4x minecraft:diamond',
Item.of('minecraft:diamond').withChance(0.5)], 'minecraft:coal_block')
    // Turn an iron ingot and a copper ingot into a gold ingot and require 10,000 FE
    event.recipes.thermal.smelter('minecraft:gold_ingot', ['minecraft:iron_ingot',
'minecraft:copper_ingot']).energy(10000)
```

```
    // Centrifugal Separator
    // Centrifuge one sapling into 50% chance of a stick and 300mb of water
    event.recipes.thermal.centrifuge([Item.of('minecraft:stick').withChance(0.5),
Fluid.of('minecraft:water', 300)], '#minecraft:saplings')
    // Turn 2 sweet berries into red dye
    event.recipes.thermal.centrifuge('minecraft:red_dye', '2x minecraft:sweet_berries')

    // Multiservo Press
    // Press seven bonemeal into a bone.
    event.recipes.thermal.press('minecraft:bone', '7x minecraft:bone_meal')
    // Press an iron dust into an iron nugget using the coin die. To use an item as a die they
must have the thermal:crafting/dies tag!
    event.recipes.thermal.press('minecraft:iron_nugget', ['#forge:dusts/iron',
'thermal:press_coin_die'])

    // Magma Crucible
    // Turn a sapling into 400mb of water
    event.recipes.thermal.crucible(Fluid.of('minecraft:water', 400),
'#minecraft:saplings').energy(100)
    // Melt ores into lava
    event.recipes.thermal.crucible(Fluid.of('minecraft:lava', 500), '#forge:ores')

    // Blast Chiller
    // Chill an arrow into an arrow of slowness
    event.recipes.thermal.chiller(Item.of('minecraft:tipped_arrow',
'{Potion:"minecraft:slowness"}'), [Fluid.of('minecraft:water', 100), 'minecraft:arrow'])
    // Chill lava into raw iron using the ball cast. For an item to count as a cast it needs
to have the thermal:crafting/casts tag!
    event.recipes.thermal.chiller('minecraft:raw_iron', [Fluid.of('minecraft:lava', 1000),
'thermal:chiller_ball_cast'])

    // Fractionating Still
    // Refine Creosete oil into Tree oil and latex, with a chance of producing rubber
    event.recipes.thermal.refinery([Item.of('thermal:rubber').withChance(0.8),
Fluid.of('thermal:tree_oil', 100), Fluid.of('thermal:latex', 50)],
Fluid.of('thermal:creosote', 200))
    // Refine tree oil into a small amount of refined fuel with a high energy cost
    event.recipes.thermal.refinery(Fluid.of('thermal:refined_fuel', 50),
Fluid.of('thermal:tree_oil', 100)).energy(20000)
```

```
    // Unbrew an awkward potion. This uses the cofh core potion fluid with some nbt.
    event.recipes.thermal.refinery([Fluid.of('minecraft:water', 1000),
'minecraft:nether_wart'], Fluid.of('cofh_core:potion', 1000, '{Potion:"minecraft:awkward"}'))

    // Alchemical Imbuer
    // Combine a redstone dust and 200mb of lava to make 200mb of destabilized redstone
    event.recipes.thermal.brewer(Fluid.of('thermal:redstone', 200),
[Fluid.of('minecraft:lava', 200), 'minecraft:redstone'])
    // Brew an uncraftable potion (potion with no nbt) with 64 bedrock and an awkward potion.
Oh, and an insane energy cost
    event.recipes.thermal.brewer(Fluid.of('cofh_core:potion', 1000),
[Fluid.of('cofh_core:potion', 1000, '{Potion:"minecraft:awkward"}'), '64x minecraft:bedrock'])

    // Fluid Encapsulator
    // Fill a sponge with water. Why? Well why not?
    event.recipes.thermal.bottler('minecraft:wet_sponge', [Fluid.of('minecraft:water', 10000),
'minecraft:sponge'])
    // Turn any gear into a machine frame by filling it with destabilized redstone. Nice and
low energy cost too
    event.recipes.thermal.bottler('thermal:machine_frame', ['#forge:gears',
Fluid.of('thermal:redstone', 500)]).energy(500)
})
```

KubeJS Create

Create integration for KubeJS. This mod allows you to add and properly edit recipes of Create mod in KubeJS scripts. All supported recipe types and examples are below. See Recipes page for more info.

Simple Recipe Types

- createCrushing
- createCutting
- createMilling
- createBasin
- createMixing
  - supports *.heated()* and *.superheated()*
- createCompacting
  - supports *.heated()* and *.superheated()*
  - Can have any number of inputs
  - Used basin
- createPressing
  - Only has one item input
  - Used on any surface
- createSandpaperPolishing
- createSplashing
  - AKA Bulk Washing
- createDeploying
- createFilling
- createEmptying
- createHaunting

> Bulk Smoking and Bulk Blasting recipes are auto generated from vanilla smelting, smoking, and blasting recipes.

- Bulk Smoking is vanilla smoking.
- Bulk Blasting is vanilla smelting (as long as there is not a smoking recipe) or vanilla blasting.

Syntax

event.recipes.create.mixing(output[], input[])
or
event.recipes.createMixing(output[], input[])

Output can be an item, fluid, or an array of multiple.

Input can be an ingredient, fluid, or an array of multiple.

Examples

```
onEvent('recipes', event => {
	event.recipes.createCrushing([
		'2x bone_meal',
		Item.of('5x bone_meal').withChance(0.5)
	], 'bone_block')

	event.recipes.create.mixing(Fluid.of('create:builders_tea',500),[
		Fluid.of('milk',250),
		 	Fluid.of('water',250),
		'#leaves'
	]).heated()

	event.recipes.createFilling('create:blaze_cake', [
		'create:blaze_cake_base',
		Fluid.of('minecraft:lava', 250)
	])

	event.recipes.createEmptying([
		'minecraft:glass_bottle',
		Fluid.of('create:honey', 250)
	], 'minecraft:honey_bottle')
})
```

Mechanical Crafter

Syntax

event.recipes.create.mechanicalCrafting(output, pattern[], {patternKey: input})
or
event.recipes.createMechanicalCrafting(output, pattern[], {patternKey: input})

This recipe type is the same as regular crafting table shaped recipe, however the pattern can be up to 9x9, instead of 3x3.

Examples

```
onEvent('recipes', event => {
  event.recipes.createMechanicalCrafting('minecraft:piston', [
    'CCCCC',
    'CPIPC',
    'CPRPC'
  ], {
    C: '#forge:cobblestone',
    P: '#minecraft:planks',
    R: '#forge:dusts/redstone',
    I: '#forge:ingots/iron'
  })
})
```

Sequenced Assembly

Syntax

event.recipes.create.sequencedAssembly(output[], input, sequence[]).transitionalItem(transitionalItem).loops(loops)
or
event.recipes.createSequencedAssembly(output[], input, sequence[]).transitionalItem(transitionalItem).loops(loops)

Output is an item or an array of items.

If it is an array:

- The first item is the real output, the remainder are scrap.
- Only one item is chosen, with equal chance of each.
- You can use `Item.of('create:shaft').withChance(2)` to double the chance of that specific item to being chosen.

Input is an ingredient.

Transitional Item is any item* and is used during the intermediate stages of the assembly.

Sequence is an array of recipes.

- The only legal recipes are:
  - createCutting
  - createPressing
  - createDeploying
  - createFilling
- The transitional item needs to be the output of each of these recipes.
- The transitional item needs to be the an input of each of these recipes.

Loops is the number of time that the recipes repeats. Calling .loops() is optional, and defaults to 4.

Examples

```
onEvent('recipes', event => {
□event.recipes.createSequencedAssembly([ // start the recipe
□□Item.of('create:precision_mechanism').withChance(130.0), // this is the item that will appear
in JEI as the result
□□Item.of('create:golden_sheet').withChance(8.0), // the rest of these items will part of the
scrap
□□Item.of('create:andesite_alloy').withChance(8.0),
□□Item.of('create:cogwheel').withChance(5.0),
□□Item.of('create:shaft').withChance(2.0),
□□Item.of('create:crushed_gold_ore').withChance(2.0),
□□Item.of('2x gold_nugget').withChance(2.0),
□□'iron_ingot',
□□'clock'
□],'create:golden_sheet',[ // 'create:golden_sheet' is the input
□□// the transitional item set by "transitionalItem('create:incomplete_large_cogwheel')" is the
item used during the intermediate stages of the assembly
□□event.recipes.createDeploying('create:incomplete_precision_mechanism',['create:incomplete_pr
ecision_mechanism','create:cogwheel']),
 □ □// like a normal recipe function, is used as a sequence step in this array. Input and outpu
have the transitional item
□□event.recipes.createDeploying('create:incomplete_precision_mechanism',['create:incomplete_pr
ecision_mechanism','create:large_cogwheel']),
□□event.recipes.createDeploying('create:incomplete_precision_mechanism',['create:incomplete_pr
ecision_mechanism','create:iron_nugget'])
□]).transitionalItem('create:incomplete_precision_mechanism').loops(5) // set the transitional
item and the loops (amount of repetitions)



□// for this code to work, kubejs:incomplete_spore_blossom need to be added to the game
```

```
let inter = 'kubejs:incomplete_spore_blossom' // making a varrible to store the transition
item makes the code more readable
event.recipes.createSequencedAssembly([
    Item.of('spore_blossom').withChance(16.0), // this is the item that will appear in JEI as th
result
    Item.of('flowering_azalea_leaves').withChance(16.0), // the rest of these items will part of
the scrap
    Item.of('azalea_leaves').withChance(2.0),
    'oak_leaves',
    'spruce_leaves',
    'birch_leaves',
    'jungle_leaves',
    'acacia_leaves',
    'dark_oak_leaves'
  ],'flowering_azalea_leaves', [ // 'flowering_azalea_leaves' is the input
    // the transitional item is a varrible, that is "kubejs:incomplete_spore_blossom", and is us
during the intermediate stages of the assembly
    event.recipes.createPressing(inter, inter),
    // like a normal recipe function, is used as a sequence step in this array. Input and output
have the transitional item
    event.recipes.createDeploying(inter, [inter, 'minecraft:hanging_roots']),
    event.recipes.createFilling(inter, [inter, Fluid.of('minecraft:water',420)]),
    event.recipes.createDeploying(inter, [inter, 'minecraft:moss_carpet']),
      event.recipes.createCutting(inter, inter)
  ]).transitionalItem(inter).loops(2) // set the transitional item and the loops (amount of
repetitions)
})
```

Transitional Items

As mentioned earlier, any item can be a transition item. However, this is not completely recommended.

If you wish to make your own transitional item, its best if you make the type `create:sequenced_assembly`.

1.16 syntax

```
onEvent('item.registry', event => {
event.create('incomplete_spore_blossom').displayName('Incomplete Spore
Blossom').type('create:sequenced_assembly')
```

```
})
```

1.18 syntax

```
onEvent('item.registry', event => {
event.create('incomplete_spore_blossom','create:sequenced_assembly')
})
```

Mysterious Conversion

> Mysterious Conversion recipes are client side only, so the only way to add them currently is using reflection.

Example

Goes inside of **client scripts** and **not in an event**.

```
//makes the varribles used
let MysteriousItemConversionCategory =
java('com.simibubi.create.compat.jei.category.MysteriousItemConversionCategory')
let ConversionRecipe = java('com.simibubi.create.compat.jei.ConversionRecipe')

//adds in the recipes
MysteriousItemConversionCategory.RECIPES.add(ConversionRecipe.create('minecraft:apple',
'minecraft:carrot'))

MysteriousItemConversionCategory.RECIPES.add(ConversionRecipe.create('minecraft:golden_apple',
'minecraft:golden_carrot'))
```

Preventing Recipe Auto-Generation

If you don't want a smelting, blasting, smoking, crafting, or stone-cutting to get an auto-generated counter part, then include `manual_only` at the end of the recipe id.

Example

```
onEvent('recipes', event => {
event.shapeless('wet_sponge',['water_bucket','sponge']).id('kubejs:moisting_the_sponge_manual
_only')
```

```
})
```

Other types of prevention, can be done in the create config (the goggles button leads you there).

If it is not in the config, then you can not change it.

3rd Party addons

3rd party add-ons: (Not including mods with optional dependencies of KubeJS)

Name:	Description	Links	Loader	Versions
Ponder for KubeJS	Make custom Create Ponder scenes with KubeJS.	Wiki CurseForge Discord Github	Forge	1.16.5 1.18.2
LootJS	A mod for **packdevs** to easily modify the loot system with KubeJS.	Wiki CurseForge Modrinth Discord Github	Forge & Fabric	1.18.2
MoreJS	A mod for **packdevs** to extend KubeJS with more events and utilities.	Wiki CurseForge Modrinth Discord Github	Forge & Fabric	1.18.2
ProbeJS	A typing generator mod to generate KubeJS typings. Enabling Intellisense for your KubeJS environments!	Wiki CurseForge Github	Forge & Fabric	1.18.2
KubeJS ComputerCraft	Adds support for KubeJS to add ComputerCraft peripherals to any block.	CurseForge Github	Forge & Fabric	1.18.2
KubeJS Borealis	Adds a form of "documentation" to the mod KubeJS using the mod Borealis	Example CurseForge Github	Forge	1.16.5 1.18.2
KubeJS TwitchIntegration	Cool twitch integration	Events Examples CurseForge Github	Forge	1.16.5
KubeJS: RTJC	A proof of concept add-on that allows you to compile and run Java code at runtime.	Description CurseForge Github	Forge	1.16.5
Kubejs Debug Adapter	A Debug Adapter Protocol implementation for KubeJS scripts.	Modrinth Github	Forge	1.18.2

KJSPKG

KJSPKG is a package manager for KubeJS that can allow you to download different example scripts and libraries into your instance. It works with legacy versions, as well as KubeJS 6. **More info on the new wiki**.