Custom Blocks

> This is a startup script, meaning that you will need to *restart your game* each time you want to make changes to it.

You can register many types of custom blocks in KubeJS. Here's the simplest way:

```
StartupEvents.registry("block", (event) => {
  event.create("example_block") // Create a new block with ID "kubejs:example_block"
})
```

That's it! Launch the game, and assuming you've left KubeJS's auto-generated resources alone, there should be a fully-textured block in the Creative menu under KubeJS (purple dye). KubeJS will also generate the name "Example Block" for you.

To make modifications to this block, we use the **block builder** returned by the `event.create()` call. The block builder allows us to chain together multiple modifications. Let's try making some of the more common modifications:

```
StartupEvents.registry("block", (event) => {
    event.create("example_block") // Create a new block
    .displayName("My Custom Block") // Set a custom name
    .material("wood") // Set a material (affects the sounds and some properties)
    .hardness(1.0) // Set hardness (affects mining time)
    .resistance(1.0) // Set resistance (to explosions, etc)
    .tagBlock("my_custom_tag") // Tag the block with `#minecraft:my_custom_tag` (can have multiple tags)
    .requiresTool(true) // Requires a tool or it won't drop (see tags below)
    .tagBlock("my_namespace:my_other_tag") // Tag the block with `#my_namespace:my_other_tag`
    .tagBlock("mineable/axe") //can be mined faster with an axe
    .tagBlock("mineable/pickaxe") // or a pickaxe
    .tagBlock('minecraft:needs_iron_tool') // the tool tier must be at least iron
})
```

All Block Builder Methods

In case it wasn't covered above, here's list of each method you can use when building a block.

- `displayName('name')`
  - Sets the item's display name.
- `material('material')` (No longer supported in 1.20+, see `mapColor` and `soundType` below!)
  - Set the item's material to an available material from the Materials List:

Materials List

air
amethyst
bamboo
bamboo_sapling
barrier
bubble_column
buildable_glass
cactus
cake
clay
cloth_decoration
decoration
dirt
egg
explosive
fire
froglight
frogspawn
glass
grass
heavy_metal
ice
ice_solid
lava
leaves
metal
moss
nether_wood
piston
plant
portal
powder_snow
replaceable_fireproof_plant
replaceable_plant
replaceable_water_plant
sand
sculk

shulker_shell
snow
sponge
stone
structural_air
top_snow
vegetable
water
water_plant
web
wood
wool

- `mapColor(MapColor)` (1.20.1+ only)
  - Set block map color, you can find the entire list here, use ID in lowercase, e.g. `'color_light_green'`.
- `soundType(SoundType)` (1.20.1+ only)
  - Set block sound type:

SoundType List

Instead of using `soundType(SoundType)` you can also use one of these shortcut methods:

- `noSoundType()`
- `woodSoundType()`
- `stoneSoundType()`
- `gravelSoundType()`
- `grassSoundType()`
- `sandSoundType()`
- `cropSoundType()`
- `glassSoundType()`

wood
gravel
grass
lily_pad
stone
metal
glass
wool
sand
snow
powder_snow

ladder
anvil
slime_block
honey_block
wet_grass
coral_block
bamboo
bamboo_sapling
scaffolding
sweet_berry_bush
crop
hard_crop
vine
nether_wart
lantern
stem
nylium
fungus
roots
shroomlight
weeping_vines
twisting_vines
soul_sand
soul_soil
basalt
wart_block
netherrack
nether_bricks
nether_sprouts
nether_ore
bone_block
netherite_block
ancient_debris
lodestone
chain
nether_gold_ore
gilded_blackstone
candle
amethyst
amethyst_cluster
small_amethyst_bud
medium_amethyst_bud
large_amethyst_bud
tuff

calcite
dripstone_block
pointed_dripstone
copper
cave_vines
spore_blossom
azalea
flowering_azalea
moss_carpet
pink_petals
moss
big_dripleaf
small_dripleaf
rooted_dirt
hanging_roots
azalea_leaves
sculk_sensor
sculk_catalyst
sculk
sculk_vein
sculk_shrieker
glow_lichen
deepslate
deepslate_bricks
deepslate_tiles
polished_deepslate
froglight
frogspawn
mangrove_roots
muddy_mangrove_roots
mud
mud_bricks
packed_mud
hanging_sign
nether_wood_hanging_sign
bamboo_wood_hanging_sign
bamboo_wood
nether_wood
cherry_wood
cherry_sapling
cherry_leaves
cherry_wood_hanging_sign
chiseled_bookshelf
suspicious_sand

suspicious_gravel
decorated_pot
decorated_pot_cracked

You can construct your own sound type with `new SoundType(volume, pitch, breakSound, stepSound, placeSound, hitSound, fallSound)` where volume and pitch are floats 0.0 - 1.0 (usually leave it as 1.0) and all sounds are SoundEvents.

- `property(BlockProperty)`
  - Adds more blockstates to the block, like being waterlogged or facing a certain direction. A full list of properties is available in the Properties List:

Properties List

Usage: `.property(BlockProperties.PICKLES)`

Boolean Properties (true/false):

attached,
berries,
bloom,
bottom,
can_summon,
conditional,
disarmed,
down,
drag,
east,
enabled,
extended,
eye,
falling,
hanging,
has_book,
has_bottle_0,
has_bottle_1,
has_bottle_2,
has_record,
inverted,
in_wall,
lit,
locked,
north,

occupied,
open,
persistent,
powered,
short,
shrieking,
signal_fire,
snowy,
south,
triggered,
unstable,
up,
vine_end,
waterlogged,
west

Integer properties:

age_1,
age_2,
age_3,
age_4,
age_5,
age_7,
age_15,
age_25,
bites,
candles,
delay,
distance,
eggs,
hatch,
layers,
level,
level_cauldron,
level_composter,
level_flowing,
level_honey,
moisture,
note,
pickles,
power,
respawn_anchor_charges,
rotation_16,

stability_distance,
stage

Directional Properties:

facing,
facing_hopper,
horizontal_facing,
vertical_direction

Other (`enum`) Properties:

attach_face,
axis,
bamboo_leaves,
bed_part,
bell_attachment,
chest_type,
door_hinge,
double_block_half,
dripstone_thickness,
east_redstone,
east_wall,
half,
horizontal_axis,
mode_comparator,
north_redstone,
north_wall,
noteblock_instrument,
orientation,
piston_type,
rail_shape,
rail_shape_straight,
sculk_sensor_phase,
slab_type,
south_redstone,
south_wall,
stairs_shape,
structureblock_mode,
tilt,
west_redstone,
west_wall

- `tagBlock('namespace:tag_name')`
  - adds a tag to the block

- `tagItem('namespace:tag_name')`
  - adds a tag to the block's item, if it has one
- `tagBoth('namespace:tag_name')`
  - adds both block and item tag if possible
- `hardness(float)`
  - Sets the block's Hardness value. Used for calculating the time it takes for the block to be destroyed.
- `resistance(float)`
  - Set's the block's resistance to things like explosions
- `unbreakable()`
  - Shortcut to set the resistance to MAX_VALUE and hardness to -1 (like bedrock)
- `lightLevel(number)`
  - Sets the block's light level.
  - Passing an integer (0-15) will set the block's light level to that value.
  - Passing a float (0.0-1.0) will multiply that number by 15, then set the block's light level to the nearest integer
- `opaque(boolean)`
  - Sets whether the block is opaque. Full, opaque blocks will not let light through.
- `fullBlock(boolean)`
  - Sets whether the block renders as a full block. Full blocks have certain optimizations applied to them, such as not rendering terrain behind them. If you're using `.box()` to make a custom hitbox, please set this to `false`.
- `requiresTool(boolean)`
  - If `true`, the block will use certain block tags to determine whether it should drop an item when mined. For example, a block tagged with `#minecraft:mineable/axe`, `#minecraft:mineable/pickaxe`, and `#minecraft:needs_iron_tool` would drop nothing unless it was mined with an axe or pickaxe that was at least iron level.
- `renderType('solid'|'cutout'|'translucent')`
  - Sets the render type.
    - `cutout` is required for blocks with texture like glass, where pixels are either transparent or not
    - `translucent` is required for blocks like stained glass, where pixels can be semitransparent
    - otherwise, use `solid` if all pixels in your block are opaque.
- `color(tintindex, color)`
  - Recolors a block to a certain color
- `textureAll('texturepath')`
  - Textures all 6 sides of the block to the same texture.
  - The path must look like `kubejs:block/texture_name` (which would be included under `kubejs/assets/kubejs/textures/block/texture_name.png`).
  - Defaults to `kubejs:block/<block_name>`
- `texture('side', 'texturepath')`
  - Texture one side by itself. Valid sides are `up`, `down`, `north`, `south`, `east`, and `west`.
- `model('modelpath')`
  - Specify a custom model.
  - The path must look like `kubejs:block/texture_name` (which would be included under `kubejs/assets/kubejs/models/block/texture_name.png`).

    - Defaults to `kubejs:block/<block_name>`.
- `noItem()`
    - Removes the associated item. Minecraft does this by default for a few blocks, like nether portal blocks. Use this if the player should never be able to hold or place the block.
- `box(x0, y0, z0, x1, y1, z1, boolean)`
- `box(x0, y0, z0, x1, y1, z1)` // defaults to true
    - Sets a custom hitbox for the block, affecting collision. You can use this multiple times to define a complex shape composed of multiple boxes.
    - Each box is a rectangular prism with corners at (x0,y0,z0) and (x1,y1,z1)
    - You will probably want to set up a custom block model that matches the shape you define here.
    - The final boolean determines the coordinate scale of the box. Passing in `true` will use the numbers 0-16, while passing in `false` will use coordinates ranging from 0.0 to 1.0
- `noCollision()`
    - Removes the default full-block hitbox, allowing you to fall through the block.
- `notSolid()`
    - Tells the renderer that the block isn't solid.
- `waterlogged()`
    - Allows the block to be waterloggable.
- `noDrops()`
    - The block will not drop itself, even if mined with silk touch.
- `slipperiness(float)`
    - Sets the slipperiness of the block. Affects how much entities slide while moving on it. Almost every block in Vanilla has a slipperiness value of 0.6, except slime (0.8) and ice (0.98).
- `speedFactor(float)`
    - A modifier affecting how quickly players walk on the block.
- `jumpFactor(float)`
    - A modifier affecting how high players can jump off the block.
- `randomTick(consumer<randomTickEvent>)`
    - A function to run when the block recieves a random tick.
- `item(consumer<itemBuilder>)`
    - Modify certain properties of the block's item (see link)
- `setLootTableJson(json)`
    - Pass in a custom loot table JSON directly
- `setBlockstateJson(json)`
    - Pass in a custom blockstate JSON directly
- `setModelJson(json)`
    - Pass in a custom model JSON directly
- `noValidSpawns(boolean)`
    - If `true`, the block is not counted as a valid spawnpoint for entities
- `suffocating(boolean)`
    - Whether the block will suffocate entities that have their head inside it
- `viewBlocking(boolean)`

    - Whether the block counts as blocking a player's view.
- `redstoneConductor(boolean)`
    - Sets whether the block will conduct redstone. True by default.
- `transparent(boolean)`
    - Sets whether the block is transparent or not
- `defaultCutout()`
    - batches a bunch of methods to make blocks such as glass
- `defaultTranslucent()`
    - similar to defaultCutout() but using translucent layer instead

---

Revision #11
Created 31 August 2023 23:08:41 by Nat
Updated 11 September 2023 10:12:25 by Lat