Custom Items

This is a startup_scripts/ event

```
// Listen to item registry event
onEvent('item.registry', event => {

  // The texture for this item has to be placed in kubejs/assets/kubejs/textures/item/test_item.png
  // If you want a custom item model, you can create one in Blockbench and put it in
kubejs/assets/kubejs/models/item/test_item.json
  event.create('test_item')

  // You can chain builder methods as much as you like
  event.create('test_item_2').maxStackSize(16).glow(true)

  // You can specify item type as 2nd argument in create(), some types have different available methods
  event.create('custom_sword', 'sword').tier('diamond').attackDamageBaseline(10.0)
})
```

Valid item types:

- "basic"
  - default
- "sword"
- "pickaxe"
- "axe"
- "shovel"
- "hoe"
- "helmet"
- "chestplate"
- "leggings"
- "boots"

Other methods item builder supports:

You can chain these methods after create()

> Anything with a ??? may not be completely accurate

Physical Properties

- maxStackSize(size)
- unstackable()
  - Identical to maxStackSize(1)
- maxDamage(damage)
  - ie max durability of the item
- burnTime(ticks)
  - In a furnace
- fireResistant(true/false)

Non-Model Visual Stuff

- rarity('rarity')
  - Options are:
    - "common"
    - "uncommon"
    - "rare"
    - "epic"
- glow(true/false)
- tooltip(text...)
  - The text under the item name to provide details about it
- color(index, colorHex)
  - If you do not have a custom model, index is 0
  - If you do have a custom model, then index is the layer hat you want to affect
  - there is an example below
- color((item, number) => {...})
  - any code you want
  - must return a color
  - there is an example below
  - ???
- displayName(name)
- name(item => {...})
  - you can put whatever code in there you want
  - must return a string
  - ???
- translationKey(key)
  - ???
  - You don't need this unless you know what you are doing

Model Editing

There is an example below

- textureJson(json)

    - for example {layer0:"minecraft:item/sand",layer1:"minecraft:item/paper"}
    - The contents of the texture part of item model
    - ???
- modelJson(json)
    - the entire json that you would put for a item model, you can just put in here
    - ???
- parentModel(modelName)
    - Set the "parent" property of this items model to modelName
- texture(customTexturePath)
    - for example "minecraft:item/feather"
- texture(key, customTexturePath)
    - if key is "layer0", then its the same as texture(customTexturePath)
    - ???

Bar

There an example farther below

- barColor((item) => {...})
    - must return a color
    - any code you want
    - ???
- barWidth(width)
    - ???

Custom Uses

The is a section below for an example

- useAnimation(animation)
    - Can be:
        - "spear"
            - trident
        - "crossbow"
        - "eat"
            - food
        - "spyglass"
        - "block"
        - "none"
        - "bow"
        - "drink"
    - ???
- useDuration(itemstack => {...})
    - any code you want
    - for example useDuration(itemstack => 60)

        - three seconds
    - must return a whole number
    - if you want something that does not end on its own, then use something like 72000 (an hour)
    - ???
- use((level, player, hand) => {...})
    - for example use(() => true)
    - any code you want
    - item is usable if it is true
    - must return a boolean
    - ???
- finishUsing((itemstack, level, entity) => {...})
    - any code you want
    - when the duration completes
    - ???
- releaseUsing((itemstack, level, entity, tick) => {...})
    - any code you want
    - when released before the duration completes
    - ???

Miscellaneous

- type(type)
    - for 1.16
- tag(resourceLocation)
    - ???
- tool(type, level)
    - for 1.16
- modifyAttribute(attribute, identifier, d, operation)
    - ???
- group(group_id)
    - Creative mode tab
    - Vanilla tabs are:
        - "search"
        - "buildingBlocks"
        - "decorations"
        - "redstone"
        - "transportation"
        - "misc"
        - "food"
        - "tools"
        - "combat"
        - "brewing"
- containerItem(id)
    - A item to reference properties of
    - ???

- subtypes(item => {...})
  - must return a itemstack collection
  - This is for making JEI or creative menu have the same item multiple times with different NBT
  - any code you want
  - ???
- food(foodBuilder => {...})
  - There is an example farther down

Tool

Methods available if you use 'sword', 'pickaxe', 'axe', 'shovel' or 'hoe' type:

- tier(toolTier)
  - Can be:
    - "wood"
    - "stone"
    - "iron"
    - "gold"
    - "diamond"
    - "netherite"
- modifyTier(tier => ...)
  - Same syntax as custom tool tier, see below
- attackDamageBaseline(damage)
  - You only want to modify this if you are creating a custom weapon such as Spear, Battleaxe, etc.
- attackDamageBonus(damage)
- speedBaseline(speed)
  - Same as attackDamageBaseline, only modify for custom weapon types
- speed(speed)

Armor

Methods available if you use 'helmet', 'chestplate', 'leggings' or 'boots' type:

- tier('armorTier')
  - Can be:
    - "leather"
    - "chainmail"
    - "iron"
    - "gold"
    - "diamond"
    - "turtle"
    - "netherite"
- modifyTier(tier => ...) // Same syntax as custom armor tier, see below

Creating custom tool and armor tiers

All values are optional and by default are based on iron tier

```
onEvent('item.registry.tool_tiers', event => {
  event.add('tier_id', tier => {
    tier.uses = 250
    tier.speed = 6.0
    tier.attackDamageBonus = 2.0
    tier.level = 2
    tier.enchantmentValue = 14
    tier.repairIngredient = '#forge:ingots/iron'
  })
})
```

```
onEvent('item.registry.armor_tiers', event => {
  // Slot indicies are [FEET, LEGS, BODY, HEAD]
  event.add('tier_id', tier => {
    tier.durabilityMultiplier = 15 // Each slot will be multiplied with [13, 15, 16, 11]
    tier.slotProtections = [2, 5, 6, 2]
    tier.enchantmentValue = 9
    tier.equipSound = 'minecraft:item.armor.equip_iron'
    tier.repairIngredient = '#forge:ingots/iron'
    tier.toughness = 0.0 // diamond has 2.0, netherite 3.0
    tier.knockbackResistance = 0.0
  })
})
```

Examples:

Custom Foods

These methods are each optional, and you may include as many or as few as you like.

```
onEvent('item.registry', event => {
event.create('magic_steak').food(food => {
food
  .hunger(6)
  .saturation(6)//This value does not directly translate to saturation points gained
    //The real value can be assumed to be:
```

```
    //min(hunger * saturation * 2 + saturation, foodAmountAfterEating)
    .effect('speed', 600, 0, 1)
    .removeEffect('poison')
    .alwaysEdible()//Like golden apples
    .fastToEat()//Like dried kelp
    .meat()//Dogs are willing to eat it
    .eaten(ctx => {//runs code upon consumption
      ctx.player.tell('Yummy Yummy!')
        //If you want to modify this code then you need to restart the game.
        //However, if you make this code call a global startup function
        //and place the function *outside* of an 'onEvent'
        //then you may use the command:
        //  /kubejs reload startup_scripts
        //to reload the function instantly.
      })
})
})
```

Custom Uses

```
onEvent("item.registry", event => {
event.create("nuke_soda", "basic")
        .tooltip("§5Taste of Explosion!")
        .tooltip("§c...Inappropriate intake may cause disastrous result.")
        /**
         * The use animation of the item, can be  "spear" (trident),
         * "crossbow", "eat" (food), "spyglass", "block", "none", "bow", "drink"
         * When using certain animations, corresponding sound will be played.
         */
        .useAnimation("drink")
        /**
         * The duration before the item finishs its using,
         * if you need something like hold-and-charge time (like bow),
         * consider set this to 72000 (1h) or more.
         * A returned value of 0 or lower will render the item not usable.
         */
        .useDuration((itemstack) => 64)
        /**
         * When item is about to be used.
         * If true, item will starts it use animation if duration > 0.
```

```
        */
        .use((level, player, hand) => true)
        /**
         * When the item use duration expires.
         */
        .finishUsing((itemstack, level, entity) => {
            let effects = entity.potionEffects;
            effects.add("haste", 120 * 20)
            itemstack.itemStack.shrink(1)
            if (entity.player) {
                entity.minecraftPlayer.addItem(Item.of("minecraft:glass_bottle").itemStack)
            }
            return itemstack;
        })
        /**
         * When the duration is not expired yet, but
         * players release their right button.
         * Tick is how many ticks remained for player to finish using the item.
         */
        .releaseUsing((itemstack, level, entity, tick) => {
            itemstack.itemStack.shrink(1)
            level.createExplosion(entity.x, entity.y, entity.z).explode()
        })
})
```

Bar

```
event.create("hammer")
        //Determine how long the bar is, should be an integer between 0 (empty) and 13 (full)
        //If the value is below 0, it will be treated as 0.
        //The value is capped at 13, any value over 13 will be considered "full", thus making it not shown
        .barWidth(i => i.nbt.contains("hit_count") ? i.nbt.getInt("hit_count") / 13.0 : 0)
        //Determine what color should the bar be.
        .barColor(i => Color.AQUA)
```

Dynamic Tinting and Model Stuff

```
onEvent("item.registry", (event) => {
/**
 * Old style with just setting color by index still works!
```

```
 */
event
  .create("old_color_by_index")
  .textureJson({
    layer0: "minecraft:item/paper",
    layer1: "minecraft:item/ghast_tear",
  })
  .color(0, "#70F00F")
  .color(1, "#00FFF0");

event
  .create("cooler_sword", "sword")
  .displayName("Test Cooler Sword")
  .texture("minecraft:item/iron_sword")
  .color((itemstack) => {
    /**
     * Example by storing the color in the nbt of the itemstack
     * You have to return -1 to apply no tint.
     *
     * U can test this through: /give @p kubejs:cooler_sword{color:"#ff0000"}
     */
    if (itemstack.nbt && itemstack.nbt.color) {
      return itemstack.nbt.color;
    }

    return -1;
  });

event
  .create("test_item")
  .displayName("Test Item")
  .textureJson({
    layer0: "minecraft:item/beef",
    layer1: "minecraft:item/ghast_tear",
  })
  .color((itemstack, tintIndex) => {
    /**
     * If you want to apply the color to a specific layer, you can use the tintIndex
     * tintIndex is the texture layer index from the model: layer0 -> 0, layer1 -> 1, etc.
     * U can use the `Color` wrapper for some default colors
```

```
    * 
    * This example will apply the color to the ghast_tear texture.
    */
    if (tintIndex == 1) {
        return Color.BLUE;
    }
    return -1;
});

/**
 * Set a texture for a specific layer
 */
event.create("test_sword", "sword").displayName("Test Sword").texture("layer0", "minecraft:item/bell");

/**
 * Directly set your custom model json
 */
event.create("test_something").displayName("Test something").modelJson({
    parent: "minecraft:block/anvil",
});
});
```

Revision #17
Created 28 June 2020 17:19:33 by Lat
Updated 27 January 2023 21:18:26 by Q6