Components, KubeJS and you!

In 1.18.2 and beyond KubeJS uses Components in a lot of places. It returns them for entity names, item names and accepts them for everything from tooltips to sending messages to players.

> All examples use `event.player.tell` from the `player.chat` event to output their example, but they will with anywhere that accepts a Component!

Making your own Components starts from the ComponentWrapper class, invokable with just `Component` or `Text` from anywhere. The examples all use `Component` but `Text` works just the same.

ComponentWrapper methods:

Name	**Return Type**	**Info**
of(Object o)	MutableComponent	Returns a component based on what was input. Accepts strings, primitives like numbers, snbt/nbt format of Components and a couple others.
clickEventOf(Object o)	ClickEvent	Returns a ClickEvent based on what was input. See examples below
prettyPrintNbt(Tag tag)	Component	Returns a component with a prettified version of the input NBT.
join(MutableComponent seperator, Iterable<? extends Component> texts)	MutableComponent	Returns the result of looping through `texts` and joining them, separating each one with `seperator`.
string(String text)	MutableComponent	Returns a basic unformatted TextComponent with just the input text
translate(String key)	MutableComponent	Returns a basic unformatted TranslatableComponent with the input key.
translate(String key, Object... objects)	MutableComponent	Returns an unformatted TranslatableComponent with `objects` as the replacements for %s in the translation output.

Name	Return Type	Info
keybind(String keybind)	MutableComponent	Returns a basic unformatted KeybindComponent with the specified keybind.
<color>(Object text)	MutableComponent	Returns a basic Component with the specified color for text coloring. Valid colors are in the list below. Do not include the <> brackets.

A list of colors accepted in various places:

- black
- darkBlue
- darkGreen
- darkAqua
- darkRed
- darkPurple
- gold
- gray
- darkGray
- blue
- green
- aqua
- red
- lightPurple
- yellow
- white

Basic examples:

```
onEvent('player.chat', event => {
  // Tell the player a normal message
  event.player.tell(Component.string('Hello world'))
  // Now in black
  event.player.tell(Component.black('Welcome to the dark side, we have cookies!'))
  // Tell them the diamond item, in whatever language they have set
  event.player.tell(Component.translate('item.minecraft.diamond'))
  // Now tell them whatever key they have crouching set to
  event.player.tell(Component.keybind('key.sneak'))
  // And finally show them the nbt data of the item they are holding
  event.player.tell(Component.prettyPrintNbt(event.player.mainHandItem.nbt))
})
```

MutableComponent

These are methods you can call on any MutableComponent. This includes ComponentKJS, which is a KubeJS extension for vanilla's components and is injected into vanillas code on runtime. All methods from ComponentKJS are included, but only relevant ones from vanilla are included.

Name	Return Type	Info
iterator()	Iterator<Component>	Returns an Iterator for the components contained in this component, useful for when multiple have been joined or appended. From ComponentKJS.
self()	MutableComponent	Returns the component you ran it on. From ComponentKJS.
toJson()	JsonElement	Returns the Json representation of this Component. From ComponentKJS.
<color>()	MutableComponent	Modifies the Component with the specified color applied as formatting, and returns itself. Do not include the <> brackets. From ComponentKJS.
color(Color c)	MutableComponent	Modifies the Component to have the input Color, and returns itself. (Color is a Rhino color). From ComponentKJS.
noColor()	MutableComponent	Modifies the Component to have no color, and returns itself. From ComponentKJS.

bold()
italic()
underlined()
strikethrough()
obfuscated()
	MutableComponent	Modifies the Component to have said formatting and returns itself. From ComponentKJS.

bold(@Nullable Boolean value)
italic(@Nullable Boolean value)
underlined(@Nullable Boolean value)
strikethrough(@Nullable Boolean value)
obfuscated(@Nullable Boolean value)
	MutableComponent	Modifies the Component to have said formatting and returns itself. From ComponentKJS.

Name	Return Type	Info
insertion(@Nullable String s)	MutableComponent	Makes the Component insert the specified string into the players chat box when shift clicked (does not send it) and returns itself. From ComponentKJS.
font(@Nullable ResourceLocation s)	MutableComponent	Changes the Components font to the specified font and returns itself. For more information on adding fonts see the Minecraft Wiki's Resource packs page. From ComponentKJS.
click(@Nullable ClickEvent s)	MutableComponent	Sets this components ClickEvent to the specified ClickEvent. From ComponentKJS.
hover(@Nullable Component s)	MutableComponent	Sets the hover tooltip for this Component to the input Component. From ComponentKJS.
setStyle(Style style)	MutableComponent	Sets the style to the input Style (net.minecraft.network.chat.Style) and returns itself. Not recommended for use, use the specific methods added by CompontentKJS instead.
append(String string)	MutableComponent	Appends the input string as a basic TextComponent to this Component then returns itself.
append(Component component)	MutableComponent	Appends the input Component to this Component then returns itself.
withStyle(Style style)	MutableComponent	Merges the input style with the current style, preffering properties from the new style if a conflict exists.
getStyle()	Style	Returns this Components current Style.
getContents()	MutableComponent	Returns this Components contents. Will return the text for TextComponents, the pattern for SelectorComponents and an empty string for all other Components.
getSiblings()	List<Component>	Returns a list of all Components which have been append()ed to this Component
plainCopy()	BaseComponent	Returns a basic copy of this, preserving only the contents and not the style or siblings.
copy()	MutableComponent	Returns a full copy of this Component, preserving style and siblings

Name	Return Type	Info
getString()	String	Returns this components text as a String. Will return a blank string for any non-text component

More complex examples:

```
// First a prefix, like a rank. This won't be changing so we can just declare it up here.
const prefix = Component.darkRed('[Admin]').underlined()

onEvent('player.chat', event => {

  // First cancel the event because we are going to be sending the message ourselves
  event.cancel()

  // The main Component we will be adding stuff to. It is just a copy of the prefix component for now
  let component = prefix.copy() // If we didn't copy it all the modifications we made to it would be applied to the
original as well!

  // Make a component of the players name and then surround with < > and make it white again. Then append it
our main copmponent.
  // A component will inherit any styiling it doesnt have from whatever it has been .append()ed to, so you need to
apply formatting rather liberally some times!
  let playerName = Component.string(event.getUsername())
  // Doing it this way means we only have to apply the white formatting and no underline once to the name
  let nameComponent = Component.white(' <').underlined(false).append(playerName).append('> ')
  component.append(nameComponent)

  // Finnally add the message (obfuscated, of course) and send it!
  // We make sure to set its color and underline though, otherwise it will end up inheriting the red and underline
from the prefix!
  component.append(Component.string(event.message).obfuscated().white().underlined(false))
  event.server.tell(component)

})
```

Revision #3
Created 23 June 2022 06:20:24 by ChiefArug

Updated 15 November 2022 17:22:14 by Lexxie