

Jul 05, 2024 14:12

Sugar.pm

Page 5/10

```

293 The leading hashref, if given, will be used as extra arguments to the block
294 object's constructor.
295
296 =cut
297
298 sub quote (@elements) {
299     my $arg = _HASH0($elements[0]) ? (shift @elements) : {};
300
301     Slack::BlockKit::Block::RichText::Quote->new({
302         %$arg,
303         elements => _rtextify(@elements),
304     });
305 }
306
307 =func channel
308
309 my $rich_text_channel = channel($channel_id);
310 # or
311 my $rich_text_channel = channel(\%arg, $channel_id);
312
313 This function returns a L<channel mention
314 object|Slack::BlockKit::Block::RichText::Channel>, which can be used among
315 other rich text elements to "mention" a channel. The C<$channel_id> should be
316 the alphanumeric Slack channel id, not a channel name.
317
318 If given, the C<%arg> hash is extra parameters to pass to the Channel
319 constructor.
320
321 =cut
322
323 sub channel {
324     my ($arg, $id)
325         = @_ == 2 ? @_
326         : @_ == 1 ? ({}, $_[0])
327         : Carp::croak("BlockKit channel sugar called with wrong number of arguments");
328
329     Slack::BlockKit::Block::RichText::Channel->new({
330         %$arg,
331         channel_id => $id,
332     });
333 }
334
335 =func date
336
337 my $rich_text_date = date($timestamp, \%arg);
338
339 This returns a L<rich text date object|Slack::BlockKit::Block::RichText::Date>
340 for the given time (a unix timestamp). If given, the referenced C<%arg> can
341 contain additional arguments to the Date constructor.
342
343 Date formatting objects have a mandatory C<format> property. If none is given
344 in C<%arg>, the default is:
345
346     \x{200b}{date_short_pretty} at {time}
347
348 Why that weird first character? It's a zero-width space, and suppresses the
349 capitalization of "yesterday" (or other words) at the start. This
350 capitalization seems like a bug (or bad design) in Slack.
351
352 =cut
353
354 sub date ($timestamp, $arg=undef) {
355     $arg //= {};
356
357     Slack::BlockKit::Block::RichText::Date->new({
358         format => "\x{200b}{date_short_pretty} at {time}",
359         %$arg,
360         timestamp => $timestamp,
361     });
362 }
363
364 =func emoji

```

Jul 05, 2024 14:12

Sugar.pm

Page 6/10

```

365     my $rich_text_emoji = emoji($emoji_name);
366
367
368 This function returns an L<emoji
369 object|Slack::BlockKit::Block::RichText::Emoji> for the named emoji.
370
371 =cut
372
373 sub emoji ($name) {
374     Slack::BlockKit::Block::RichText::Emoji->new({
375         name => $name,
376     });
377 }
378
379 =func link
380
381 my $rich_text_link = link($url);
382 # or
383 my $rich_text_link = link($url, \%arg);
384 # or
385 my $rich_text_link = link($url, $text_string);
386 # or
387 my $rich_text_link = link($url, $text_string, \%arg);
388
389 This function returns a rich text L<link
390 object|Slack::BlockKit::Block::RichText::Link> for the given URL.
391
392 If given, the C<$text_string> string will be used as the display text for the
393 link. If not, URL itself will be displayed.
394
395 The optional C<%arg> parameter contains additional attributes to be passed to
396 the Link object constructor.
397
398 =cut
399
400 sub link {
401     Carp::croak("BlockKit link sugar called with wrong number of arguments")
402         if @_ > 3 || @_ < 1;
403
404     my $url = $_[0];
405     my $text = ref $_[1] ? undef : $_[1];
406     my $arg = ref $_[1] ? $_[1] : ($_[2] // {});
407
408     Slack::BlockKit::Block::RichText::Link->new({
409         %$arg,
410         (defined $text ? (text => $text) : ()),
411         url => $url,
412     });
413 }
414
415 =func richtext
416
417 my $rich_text = richtext($text_string);
418 # or
419 my $rich_text = richtext(\@styles, $text_string);
420
421 This function returns a new L<rich text text
422 object|Slack::BlockKit::Block::RichText::Text> for the given text string. If a
423 an arrayref is passed as the first argument, it is taken as a list of styles to
424 apply to the string. So,
425
426     richtext(['bold', 'italic'], "Hi");
427
428 ...will produce an object that has this structure form:
429
430     {
431         type => 'text',
432         styles => { bold => true(), italic => true() },
433         text => "Hi",
434     }
435
436 =cut
437

```