

```

249:     elements => _rsectionize(@sections),
250:   });
251: }
252:
253: =func preformatted
254:
255:   my $rich_text_pre = preformatted(@elements);
256:   # or
257:   my $rich_text_pre = preformatted(\%arg, @elements);
258:
259: This returns a new L<preformatted rich
260: text|Slack::BlockKit::Block::RichTextPreformatted> block object, with the given
261: elements as its C<elements>. Any non-references in the list will be turned
262: into text objects.
263:
264: The leading hashref, if given, will be used as extra arguments to the block
265: object's constructor.
266:
267: B<Beware>: The Slack documentation suggests that C<emoji> objects can be
268: present in the list of elements, but in practice, this always seems to get
269: rejected by Slack.
270:
271: =cut
272:
273: sub preformatted (@elements) {
274:   my $arg = _HASH0($elements[0]) ? (shift @elements) : {};
275:
276:   Slack::BlockKit::Block::RichText::Preformatted->new({
277:     %$arg,
278:     elements => _rtextify(@elements),
279:   });
280: }
281:
282: =func quote
283:
284:   my $rich_text_quote = quote(@elements);
285:   # or
286:   my $rich_text_quote = quote(\%arg, @elements);
287:
288: This returns a new L<quoted rich
289: text|Slack::BlockKit::Block::Quote> block object, with the given elements as
290: its C<elements>. Any non-references in the list will be turned into text
291: objects.
292:
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
365:
366:   my $rich_text_emoji = emoji($emoji_name);
367:
368: This function returns an L<emoji
369: object|Slack::BlockKit::Block::RichText::Emoji> for the named emoji.
370:
371: =cut
372:

```