```
The leading hashref, if given, will be used as extra arguments to the block
object's constructor.

=cut

sub quote (@elements) {
  my $arg = _HASH0($elements[0]) ? (shift @elements) : {};

  Slack::BlockKit::Block::RichText::Quote->new({
    %$arg,
    elements => _rtextify(@elements),
  });
}

=func channel

  my $rich_text_channel = channel($channel_id);
  # or
  my $rich_text_channel = channel(\%arg, $channel_id);

This function returns a L<channel mention
object|Slack::BlockKit::Block::RichText::Channel>, which can be used among
other rich text elements to "mention" a channel.  The C<$channel_id> should be
the alphanumeric Slack channel id, not a channel name.

If given, the C<%arg> hash is extra parameters to pass to the Channel
constructor.

=cut

sub channel {
  my ($arg, $id)
    = @_ == 2 ? @_
    : @_ == 1 ? ({}, $_[0])
    : Carp::croak("BlockKit channel sugar called with wrong number of arguments");

  Slack::BlockKit::Block::RichText::Channel->new({
    %$arg,
    channel_id => $id,
  });
}

=func date

  my $rich_text_date = date($timestamp, \%arg);

This returns a L<rich text date object|Slack::BlockKit::Block::RichText::Date>
for the given time (a unix timestamp).  If given, the referenced C<%arg> can
contain additional arguments to the Date constructor.

Date formatting objects have a mandatory C<format> property.  If none is given
in C<%arg>, the default is:

  \x{200b}{date_short_pretty} at {time}

Why that weird first character?  It's a zero-width space, and suppresses the
capitalization of "yesterday" (or other words) at the start.  This
capitalization seems like a bug (or bad design) in Slack.

=cut

sub date ($timestamp, $arg=undef) {
  $arg //= {};

  Slack::BlockKit::Block::RichText::Date->new({
    format => "\x{200b}{date_short_pretty} at {time}",
    %$arg,
    timestamp => $timestamp,
  });
}

=func emoji
```



```
  my $rich_text_emoji = emoji($emoji_name);

This function returns an L<emoji
object|Slack::BlockKit::Block::RichText::Emoji> for the named emoji.

=cut

sub emoji ($name) {
  Slack::BlockKit::Block::RichText::Emoji->new({
    name => $name,
  });
}

=func link

  my $rich_text_link = link($url);
  # or
  my $rich_text_link = link($url, \%arg);
  # or
  my $rich_text_link = link($url, $text_string);
  # or
  my $rich_text_link = link($url, $text_string, \%arg);

This function returns a rich text L<link
object|Slack::BlockKit::Block::RichText::Link> for the given URL.

If given, the C<$text_string> string will be used as the display text for the
link.  If not, URL itself will be displayed.

The optional C<\%arg> parameter contains additional attributes to be passed to
the Link object constructor.

=cut

sub link {
  Carp::croak("BlockKit link sugar called with wrong number of arguments")
    if @_ > 3 || @_ < 1;

  my $url  = $_[0];
  my $text = ref $_[1] ? undef : $_[1];
  my $arg  = ref $_[1] ? $_[1] : ($_[2] // {});

  Slack::BlockKit::Block::RichText::Link->new({
    %$arg,
    (defined $text ? (text => $text) : ()),
    url => $url,
  });
}

=func richtext

  my $rich_text = richtext($text_string);
  # or
  my $rich_text = richtext(\@styles, $text_string);

This function returns a new L<rich text text
object|Slack::BlockKit::Block::RichText::Text> for the given text string.  If a
an arrayref is passed as the first argument, it is taken as a list of styles to
apply to the string.  So,

  richtext(['bold','italic'], "Hi");

...will produce an object that has this structure form:

  {
    type   => 'text',
    styles => { bold => true(), italic => true() },
    text   => "Hi",
  }

=cut

sub richtext {
```