NAME

Pod::Simple::HTML - convert Pod to HTML

SYNOPSIS

```
perl -MPod::Simple::HTML -e Pod::Simple::HTML::go thingy.pod
```

DESCRIPTION

This class is for making an HTML rendering of a Pod document.

This is a subclass of Pod::Simple::PullParser and inherits all its methods (and options).

Note that if you want to do a batch conversion of a lot of Pod documents to HTML, you should see the module Pod::Simple::HTMLBatch.

CALLING FROM THE COMMAND LINE

TODO

```
perl -MPod::Simple::HTML -e Pod::Simple::HTML::go Thing.pod Thing.html
```

CALLING FROM PERL

Minimal code

```
use Pod::Simple::HTML;
my $p = Pod::Simple::HTML->new;
$p->output_string(\my $html);
$p->parse_file('path/to/Module/Name.pm');
open my $out, '>', 'out.html' or die "Cannot open 'out.html': $!\n";
print $out $html;
```

More detailed example

```
use Pod::Simple::HTML;
```

Set the content type:

```
$Pod::Simple::HTML::Content_decl = q{<meta http-equiv="Content-Type" content="te

my $p = Pod::Simple::HTML->new;
```

Include a single javascript source:

```
 $p->html_javascript('http://abc.com/a.js');
```

Or insert multiple javascript source in the header (or for that matter include anything, thought this is not recommended)

```
$p->html_javascript('
     <script type="text/javascript" ></script>" -P src=" -- http://abc.com/b.js
     <script type="text/javascript" ></script>');" -P src=" -- http://abc.com/c.
```

Include a single css source in the header:

```
$p->html_css('/style.css');
```

or insert multiple css sources:

```
$p->html_css('
     <link rel="stylesheet" type="text/css" title="pod_stylesheet" >" -P href="
<link rel="stylesheet" type="text/css" title="pod_stylesheet" href="/style.css">
```

Tell the parser where should the output go. In this case it will be placed in the `$html` variable:

```
my $html;
$p->output_string(\$html);
```

Parse and process a file with pod in it:

```
$p->parse_file('path/to/Module/Name.pm');
```

METHODS

TODO all (most?) accessorized methods

The following variables need to be set **before** the call to the ->new constructor.

Set the string that is included before the opening <html> tag:

```
$Pod::Simple::HTML::Doctype_decl = qq{<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.
        >\n};" -P " -- http://www.w3.org/TR/html4/loose.dtd
```

Set the content-type in the HTML head: (defaults to ISO-8859-1)

```
$Pod::Simple::HTML::Content_decl = q{<meta http-equiv="Content-Type" content="te
```

Set the value that will be embedded in the opening tags of F, C tags and verbatim text. F maps to , C maps to <code>, Verbatim text maps to <pre> (Computerese defaults to "")

```
$Pod::Simple::HTML::Computerese = ' class="some_class_name';
```

html_css

html_javascript

title_prefix

title_postfix

html_header_before_title

This includes everything before the <title> opening tag including the Document type and including the opening <title> tag. The following call will set it to be a simple HTML file:

```
$p->html_header_before_title('<html><head><title>');
```

top_anchor

By default Pod::Simple::HTML adds a dummy anchor at the top of the HTML. You can change it by calling

```
$p->top_anchor('<a name="zz" >');
```

html_h_level

Normally =head1 will become <h1>, =head2 will become <h2> etc. Using the html_h_level method will change these levels setting the h level of =head1 tags:

```
$p->html_h_level(3);
```

Will make sure that =head1 will become <h3> and =head2 will become <h4> etc...

index

Set it to some true value if you want to have an index (in reality a table of contents) to be added at the top of the generated HTML.

```
$p->index(1);
```

html_header_after_title

Includes the closing tag of </title> and through the rest of the head till the opening of the body

```
$p->html_header_after_title('</title>...</head><body id="my_id">');
```

html_footer

The very end of the document:

```
$p->html_footer( qq[\n<!-- end doc -->\n\n</body></html>\n] );
```

SUBCLASSING

Can use any of the methods described above but for further customization one needs to override some of the methods:

```
package My::Pod;
use strict;
use warnings;

use base 'Pod::Simple::HTML';
```

```
# needs to return a URL string such
 # http://some.other.com/page.html
# #anchor_in_the_same_file
# /internal/ref.html
sub do_pod_link {
# My::Pod object and Pod::Simple::PullParserStartToken object
my ($self, $link) = @_;

say $link->tagname; # will be L for links
say $link->attr('to'); #
say $link->attr('type'); # will be 'pod' always
say $link->attr('section');

# Links local to our web site
if ($link->tagname eq 'L' and $link->attr('type') eq 'pod') {
my $to = $link->attr('to');
if ($to =~ /^Padre::/) {
$to =~ s{::}{/}g;
return "/docs/Padre/$to.html";
}
}

# all other links are generated by the parent class
my $ret = $self->SUPER::do_pod_link($link);
return $ret;
}

1;
```

Meanwhile in script.pl:

```
use My::Pod;

my $p = My::Pod->new;

my $html;
$p->output_string(\$html);
$p->parse_file('path/to/Module/Name.pm');
open my $out, '>', 'out.html' or die;
print $out $html;
```

TODO

maybe override do_beginning do_end

SEE ALSO

Pod::Simple, Pod::Simple::HTMLBatch

TODO: a corpus of sample Pod input and HTML output? Or common idioms?

SUPPORT

Questions or discussion about POD and Pod::Simple should be sent to the pod-people@perl.org mail list. Send an empty email to pod-people-subscribe@perl.org to subscribe.

This module is managed in an open GitHub repository, <https://github.com/perl-pod/pod-simple/>. Feel free to fork and contribute, or to clone <git://github.com/perl-pod/pod-simple.git> and send patches!

Patches against Pod::Simple are welcome. Please send bug reports to <bug-pod-simple@rt.cpan.org>.

COPYRIGHT AND DISCLAIMERS

ACKNOWLEDGEMENTS

Thanks to Hurricane Electric <http://he.net/> for permission to use its Linux man pages online <http://man.he.net/> site for man page links.

Thanks to search.cpan.org <http://search.cpan.org/> for permission to use the site for Perl module links.

AUTHOR

Pod::Simple was created by Sean M. Burke <sburke@cpan.org>. But don't bother him, he's retired.

Pod::Simple is maintained by:

- Allison Randal `allison@perl.org`
- Hans Dieter Pearcey `hdp@cpan.org`
- David E. Wheeler `dwheeler@cpan.org`