

NAME

HTML::PullParser – Alternative HTML::Parser interface

SYNOPSIS

```
use HTML::PullParser;

$p = HTML::PullParser->new(file => "index.html",
                           start => 'event, tagname, @attr',
                           end   => 'event, tagname',
                           ignore_elements => [qw(script style)],
                           ) || die "Can't open: $!";

while (my $token = $p->get_token) {
    #...do something with $token
}
```

DESCRIPTION

The HTML::PullParser is an alternative interface to the HTML::Parser class. It basically turns the HTML::Parser inside out. You associate a file (or any IO::Handle object or string) with the parser at construction time and then repeatedly call `$parser->get_token` to obtain the tags and text found in the parsed document.

The following methods are provided:

```
$p = HTML::PullParser->new( file => $file, %options )
$p = HTML::PullParser->new( doc => \$doc, %options )
```

A HTML::PullParser can be made to parse from either a file or a literal document based on whether the `file` or `doc` option is passed to the parser's constructor.

The `file` passed in can either be a file name or a file handle object. If a file name is passed, and it can't be opened for reading, then the constructor will return an undefined value and `$!` will tell you why it failed. Otherwise the argument is taken to be some object that the HTML::PullParser can **read()** from when it needs more data. The stream will be **read()** until EOF, but not closed.

A `doc` can be passed plain or as a reference to a scalar. If a reference is passed then the value of this scalar should not be changed before all tokens have been extracted.

Next the information to be returned for the different token types must be set up. This is done by simply associating an `argspec` (as defined in HTML::Parser) with the events you have an interest in. For instance, if you want `start` tokens to be reported as the string `'S'` followed by the tagname and the attributes you might pass an `start-option` like this:

```
$p = HTML::PullParser->new(
    doc   => $document_to_parse,
    start => 'S', tagname, @attr',
    end   => 'E', tagname',
);
```

At last other HTML::Parser options, like `ignore_tags`, and `unbroken_text`, can be passed in. Note that you should not use the `event_h` options to set up parser handlers. That would confuse the inner logic of HTML::PullParser.

```
$token = $p->get_token
```

This method will return the next *token* found in the HTML document, or `undef` at the end of the document. The token is returned as an array reference. The content of this array match the `argspec` set up during HTML::PullParser construction.

```
$p->unget_token( @tokens )
```

If you find out you have read too many tokens you can push them back, so that they are returned again the next time `$p->get_token` is called.

EXAMPLES

The 'eg/hform' script shows how we might parse the form section of HTML::Documents using HTML::PullParser.

SEE ALSO

HTML::Parser, HTML::Tokenizer

COPYRIGHT

Copyright 1998–2001 Gisle Aas.

This library is free software; you can redistribute it and/or modify it under the same terms as Perl itself.