## NAME

IO::HTML − Open an HTML file with automatic charset detection

## VERSION

This document describes version 1.001 of IO::HTML, released June 28, 2014.

## SYNOPSIS

```
use IO::HTML;                 # exports html_file by default
use HTML::TreeBuilder;

my $tree = HTML::TreeBuilder->new_from_file(
             html_file('foo.html')
           );

# Alternative interface:
open(my $in, '<:raw', 'bar.html');
my $encoding = IO::HTML::sniff_encoding($in, 'bar.html');
```

## DESCRIPTION

IO::HTML provides an easy way to open a file containing HTML while automatically determining its encoding. It uses the HTML5 encoding sniffing algorithm specified in section 8.2.2.2 of the draft standard.

The algorithm as implemented here is:

1. If the file begins with a byte order mark indicating UTF−16LE, UTF−16BE, or UTF−8, then that is the encoding.
2. If the first 1024 bytes of the file contain a `<meta>` tag that indicates the charset, and Encode recognizes the specified charset name, then that is the encoding. (This portion of the algorithm is implemented by `find_charset_in`.)

   The `<meta>` tag can be in one of two formats:

   ```
   <meta charset="...">
   <meta http-equiv="Content-Type" content="...charset=...">
   ```

   The search is case-insensitive, and the order of attributes within the tag is irrelevant. Any additional attributes of the tag are ignored. The first matching tag with a recognized encoding ends the search.
3. If the first 1024 bytes of the file are valid UTF−8 (with at least 1 non-ASCII character), then the encoding is UTF−8.
4. If all else fails, use the default character encoding. The HTML5 standard suggests the default encoding should be locale dependent, but currently it is always `cp1252` unless you set `$IO::HTML::default_encoding` to a different value. Note: `sniff_encoding` does not apply this step; only `html_file` does that.

## SUBROUTINES

### html_file

```
$filehandle = html_file($filename, \%options);
```

This function (exported by default) is the primary entry point. It opens the file specified by `$filename` for reading, uses `sniff_encoding` to find a suitable encoding layer, and applies it. It also applies the `:crlf` layer. If the file begins with a BOM, the filehandle is positioned just after the BOM.

The optional second argument is a hashref containing options. The possible keys are described under `find_charset_in`.

If `sniff_encoding` is unable to determine the encoding, it defaults to `$IO::HTML::default_encoding`, which is set to `cp1252` (a.k.a. Windows−1252) by default. According to the standard, the default should be locale dependent, but that is not currently implemented.

It dies if the file cannot be opened.

**html_file_and_encoding**

```
  ($filehandle, $encoding, $bom)
    = html_file_and_encoding($filename, \%options);
```

This function (exported only by request) is just like `html_file`, but returns more information. In addition to the filehandle, it returns the name of the encoding used, and a flag indicating whether a byte order mark was found (if `$bom` is true, the file began with a BOM). This may be useful if you want to write the file out again (especially in conjunction with the `html_outfile` function).

The optional second argument is a hashref containing options. The possible keys are described under `find_charset_in`.

It dies if the file cannot be opened. The result of calling it in scalar context is undefined.

**html_outfile**

```
  $filehandle = html_outfile($filename, $encoding, $bom);
```

This function (exported only by request) opens `$filename` for output using `$encoding`, and writes a BOM to it if `$bom` is true. If `$encoding` is `undef`, it defaults to `$IO::HTML::default_encoding`. `$encoding` may be either an encoding name or an Encode::Encoding object.

It dies if the file cannot be opened.

**sniff_encoding**

```
  ($encoding, $bom) = sniff_encoding($filehandle, $filename, \%options);
```

This function (exported only by request) runs the HTML5 encoding sniffing algorithm on `$filehandle` (which must be seekable, and should have been opened in `:raw` mode). `$filename` is used only for error messages (if there's a problem using the filehandle), and defaults to "file" if omitted. The optional third argument is a hashref containing options. The possible keys are described under `find_charset_in`.

It returns Perl's canonical name for the encoding, which is not necessarily the same as the MIME or IANA charset name. It returns `undef` if the encoding cannot be determined. `$bom` is true if the file began with a byte order mark. In scalar context, it returns only `$encoding`.

The filehandle's position is restored to its original position (normally the beginning of the file) unless `$bom` is true. In that case, the position is immediately after the BOM.

Tip: If you want to run `sniff_encoding` on a file you've already loaded into a string, open an in-memory file on the string, and pass that handle:

```
  ($encoding, $bom) = do {
    open(my $fh, '<', \$string);  sniff_encoding($fh)
  };
```

(This only makes sense if `$string` contains bytes, not characters.)

**find_charset_in**

```
  $encoding = find_charset_in($string_containing_HTML, \%options);
```

This function (exported only by request) looks for charset information in a `<meta>` tag in a possibly incomplete HTML document using the "two step" algorithm specified by HTML5. It does not look for a BOM. Only the first 1024 bytes of the string are checked.

It returns Perl's canonical name for the encoding, which is not necessarily the same as the MIME or IANA charset name. It returns `undef` if no charset is specified or if the specified charset is not recognized by the Encode module.

The optional second argument is a hashref containing options. The following keys are recognized:

`encoding`

If true, return the Encode::Encoding object instead of its name. Defaults to false.

`need_pragma`

If true (the default), follow the HTML5 spec and examine the `content` attribute only of `<meta http-equiv="Content-Type"`. If set to 0, relax the HTML5 spec, and look for "charset=" in the `content` attribute of *every* meta tag.

## EXPORTS

By default, only `html_file` is exported. Other functions may be exported on request.

For people who prefer not to export functions, all functions beginning with `html_` have an alias without that prefix (e.g. you can call `IO::HTML::file(...)` instead of `IO::HTML::html_file(...)`. These aliases are not exportable.

The following export tags are available:

`:all`

All exportable functions.

`:rw`

`html_file`, `html_file_and_encoding`, `html_outfile`.

## SEE ALSO

The HTML5 specification, section 8.2.2.2 Determining the character encoding: <http://www.w3.org/TR/html5/syntax.html#determining–the–character–encoding>

## DIAGNOSTICS

`Could not read %s: %s`

The specified file could not be read from for the reason specified by `$!`.

`Could not seek %s: %s`

The specified file could not be rewound for the reason specified by `$!`.

`Failed to open %s: %s`

The specified file could not be opened for reading for the reason specified by `$!`.

`No default encoding specified`

The `sniff_encoding` algorithm didn't find an encoding to use, and you set `$IO::HTML::default_encoding` to `undef`.

## CONFIGURATION AND ENVIRONMENT

IO::HTML requires no configuration files or environment variables.

## DEPENDENCIES

IO::HTML has no non-core dependencies for Perl 5.8.7+. With earlier versions of Perl 5.8, you need to upgrade Encode to at least version 2.10, and you may need to upgrade Exporter to at least version 5.57.

## INCOMPATIBILITIES

None reported.

## BUGS AND LIMITATIONS

No bugs have been reported.

## AUTHOR

Christopher J. Madsen `<perl AT cjmweb.net>`

Please report any bugs or feature requests to `<bug-IO-HTML AT rt.cpan.org>` or through the web interface at <http://rt.cpan.org/Public/Bug/Report.html?Queue=IO–HTML>.

You can follow or contribute to IO-HTML's development at <https://github.com/madsen/io–html>.

## COPYRIGHT AND LICENSE

## DISCLAIMER OF WARRANTY

BECAUSE THIS SOFTWARE IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE SOFTWARE, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE SOFTWARE "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE SOFTWARE IS WITH YOU. SHOULD THE SOFTWARE PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR, OR CORRECTION.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE SOFTWARE AS PERMITTED BY THE ABOVE LICENSE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE SOFTWARE (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE SOFTWARE TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.