Tidy

Introduction

Tidy is a binding for the Tidy HTML clean and repair utility which allows you to not only clean and otherwise manipulate HTML documents, but also traverse the document tree.

Installing/Configuring

Requirements

To use Tidy, you will need libtidy installed, available on the tidy homepage » http://tidy.sourceforge.net/.

Installation

Tidy is currently available for PHP 4.3.x and PHP 5 as a PECL extension from » http://pecl.php.net/package/tidy.

Note
Tidy 1.0 is just for PHP 4.3.x, while Tidy 2.0 is just for PHP 5.

If » PEAR is available on your *nix-like system you can use the pear installer to install the tidy extension, by the following command: *pecl install tidy*.

You can always download the tar.gz package and install tidy by hand:

Example #1 - tidy install by hand in PHP 4.3.x

```
gunzip tidy-xxx.tgz
tar -xvf tidy-xxx.tar
cd tidy-xxx
phpize
./configure && make && make install
```

Windows users can download the extension dll from » http://pecl4win.php.net/ext.php/php_tidy.dll.

In PHP 5 you need only to compile using the *--with-tidy* option.

Runtime Configuration

The behaviour of these functions is affected by settings in *php.ini*.

Tidy Configuration Options

Name	Default	Changeable	Changelog

tidy.default_config	""	PHP_INI_SYSTEM	Available since PHP 5.0.0.
tidy.clean_output	"0"	PHP_INI_USER	PHP_INI_PERDIR in PHP 5. Available since PHP 5.0.0.

For further details and definitions of the PHP_INI_* constants, see the php.ini directives.

Here's a short explanation of the configuration directives.

`tidy.default_config` string
Default path for tidy config file.

`tidy.clean_output` boolean
Turns on/off the output repairing by Tidy.

Warning
Do not turn on *tidy.clean_output* if you are generating non-html content such as dynamic images.

Resource Types

This extension has no resource types defined.

Predefined Constants

The constants below are defined by this extension, and will only be available when the extension has either been compiled into PHP or dynamically loaded at runtime.

Each *TIDY_TAG_XXX* represents a HTML tag. For example, **TIDY_TAG_A** represents a link tag. Each *TIDY_ATTR_XXX* represents a HTML atribute. For example **TIDY_ATTR_HREF** would represent the href atribute in the previous example.

The following constants are defined:

tidy tag constants

constant
TIDY_TAG_UNKNOWN
TIDY_TAG_A
TIDY_TAG_ABBR
TIDY_TAG_ACRONYM
TIDY_TAG_ALIGN
TIDY_TAG_APPLET
TIDY_TAG_AREA
TIDY_TAG_B
TIDY_TAG_BASE
TIDY_TAG_BASEFONT
TIDY_TAG_BDO
TIDY_TAG_BGSOUND
TIDY_TAG_BIG
TIDY_TAG_BLINK
TIDY_TAG_BLOCKQUOTE
TIDY_TAG_BODY
TIDY_TAG_BR
TIDY_TAG_BUTTON

TIDY_TAG_CAPTION
TIDY_TAG_CENTER
TIDY_TAG_CITE
TIDY_TAG_CODE
TIDY_TAG_COL
TIDY_TAG_COLGROUP
TIDY_TAG_COMMENT
TIDY_TAG_DD
TIDY_TAG_DEL
TIDY_TAG_DFN
TIDY_TAG_DIR
TIDY_TAG_DIV
TIDY_TAG_DL
TIDY_TAG_DT
TIDY_TAG_EM
TIDY_TAG_EMBED
TIDY_TAG_FIELDSET
TIDY_TAG_FONT
TIDY_TAG_FORM
TIDY_TAG_FRAME
TIDY_TAG_FRAMESET
TIDY_TAG_H1
TIDY_TAG_H2
TIDY_TAG_H3
TIDY_TAG_H4
TIDY_TAG_H5

TIDY_TAG_H6
TIDY_TAG_HEAD
TIDY_TAG_HR
TIDY_TAG_HTML
TIDY_TAG_I
TIDY_TAG_IFRAME
TIDY_TAG_ILAYER
TIDY_TAG_IMG
TIDY_TAG_INPUT
TIDY_TAG_INS
TIDY_TAG_ISINDEX
TIDY_TAG_KBD
TIDY_TAG_KEYGEN
TIDY_TAG_LABEL
TIDY_TAG_LAYER
TIDY_TAG_LEGEND
TIDY_TAG_LI
TIDY_TAG_LINK
TIDY_TAG_LISTING
TIDY_TAG_MAP
TIDY_TAG_MARQUEE
TIDY_TAG_MENU
TIDY_TAG_META
TIDY_TAG_MULTICOL
TIDY_TAG_NOBR
TIDY_TAG_NOEMBED

TIDY_TAG_NOFRAMES
TIDY_TAG_NOLAYER
TIDY_TAG_NOSAVE
TIDY_TAG_NOSCRIPT
TIDY_TAG_OBJECT
TIDY_TAG_OL
TIDY_TAG_OPTGROUP
TIDY_TAG_OPTION
TIDY_TAG_P
TIDY_TAG_PARAM
TIDY_TAG_PLAINTEXT
TIDY_TAG_PRE
TIDY_TAG_Q
TIDY_TAG_RP
TIDY_TAG_RT
TIDY_TAG_RTC
TIDY_TAG_RUBY
TIDY_TAG_S
TIDY_TAG_SAMP
TIDY_TAG_SCRIPT
TIDY_TAG_SELECT
TIDY_TAG_SERVER
TIDY_TAG_SERVLET
TIDY_TAG_SMALL
TIDY_TAG_SPACER
TIDY_TAG_SPAN

TIDY_TAG_STRIKE
TIDY_TAG_STRONG
TIDY_TAG_STYLE
TIDY_TAG_SUB
TIDY_TAG_TABLE
TIDY_TAG_TBODY
TIDY_TAG_TD
TIDY_TAG_TEXTAREA
TIDY_TAG_TFOOT
TIDY_TAG_TH
TIDY_TAG_THEAD
TIDY_TAG_TITLE
TIDY_TAG_TR
TIDY_TAG_TR
TIDY_TAG_TT
TIDY_TAG_U
TIDY_TAG_UL
TIDY_TAG_VAR
TIDY_TAG_WBR
TIDY_TAG_XMP

tidy attribute constants

constant
TIDY_ATTR_UNKNOWN
TIDY_ATTR_ABBR

TIDY_ATTR_ACCEPT
TIDY_ATTR_ACCEPT_CHARSET
TIDY_ATTR_ACCESSKEY
TIDY_ATTR_ACTION
TIDY_ATTR_ADD_DATE
TIDY_ATTR_ALIGN
TIDY_ATTR_ALINK
TIDY_ATTR_ALT
TIDY_ATTR_ARCHIVE
TIDY_ATTR_AXIS
TIDY_ATTR_BACKGROUND
TIDY_ATTR_BGCOLOR
TIDY_ATTR_BGPROPERTIES
TIDY_ATTR_BORDER
TIDY_ATTR_BORDERCOLOR
TIDY_ATTR_BOTTOMMARGIN
TIDY_ATTR_CELLPADDING
TIDY_ATTR_CELLSPACING
TIDY_ATTR_CHAR
TIDY_ATTR_CHAROFF
TIDY_ATTR_CHARSET
TIDY_ATTR_CHECKED
TIDY_ATTR_CITE
TIDY_ATTR_CLASS
TIDY_ATTR_CLASSID
TIDY_ATTR_CLEAR

TIDY_ATTR_CODE
TIDY_ATTR_CODEBASE
TIDY_ATTR_CODETYPE
TIDY_ATTR_COLOR
TIDY_ATTR_COLS
TIDY_ATTR_COLSPAN
TIDY_ATTR_COMPACT
TIDY_ATTR_CONTENT
TIDY_ATTR_COORDS
TIDY_ATTR_DATA
TIDY_ATTR_DATAFLD
TIDY_ATTR_DATAPAGESIZE
TIDY_ATTR_DATASRC
TIDY_ATTR_DATETIME
TIDY_ATTR_DECLARE
TIDY_ATTR_DEFER
TIDY_ATTR_DIR
TIDY_ATTR_DISABLED
TIDY_ATTR_ENCODING
TIDY_ATTR_ENCTYPE
TIDY_ATTR_FACE
TIDY_ATTR_FOR
TIDY_ATTR_FRAME
TIDY_ATTR_FRAMEBORDER
TIDY_ATTR_FRAMESPACING
TIDY_ATTR_GRIDX

TIDY_ATTR_GRIDY
TIDY_ATTR_HEADERS
TIDY_ATTR_HEIGHT
TIDY_ATTR_HREF
TIDY_ATTR_HREFLANG
TIDY_ATTR_HSPACE
TIDY_ATTR_HTTP_EQUIV
TIDY_ATTR_ID
TIDY_ATTR_ISMAP
TIDY_ATTR_LABEL
TIDY_ATTR_LANG
TIDY_ATTR_LANGUAGE
TIDY_ATTR_LAST_MODIFIED
TIDY_ATTR_LAST_VISIT
TIDY_ATTR_LEFTMARGIN
TIDY_ATTR_LINK
TIDY_ATTR_LONGDESC
TIDY_ATTR_LOWSRC
TIDY_ATTR_MARGINHEIGHT
TIDY_ATTR_MARGINWIDTH
TIDY_ATTR_MAXLENGTH
TIDY_ATTR_MEDIA
TIDY_ATTR_METHOD
TIDY_ATTR_MULTIPLE
TIDY_ATTR_NAME
TIDY_ATTR_NOHREF

TIDY_ATTR_NORESIZE
TIDY_ATTR_NOSHADE
TIDY_ATTR_NOWRAP
TIDY_ATTR_OBJECT
TIDY_ATTR_OnAFTERUPDATE
TIDY_ATTR_OnBEFOREUNLOAD
TIDY_ATTR_OnBEFOREUPDATE
TIDY_ATTR_OnBLUR
TIDY_ATTR_OnCHANGE
TIDY_ATTR_OnCLICK
TIDY_ATTR_OnDATAAVAILABLE
TIDY_ATTR_OnDATASETCHANGED
TIDY_ATTR_OnDATASETCOMPLETE
TIDY_ATTR_OnDBLCLICK
TIDY_ATTR_OnERRORUPDATE
TIDY_ATTR_OnFOCUS
TIDY_ATTR_OnKEYDOWN
TIDY_ATTR_OnKEYPRESS
TIDY_ATTR_OnKEYUP
TIDY_ATTR_OnLOAD
TIDY_ATTR_OnMOUSEDOWN
TIDY_ATTR_OnMOUSEMOVE
TIDY_ATTR_OnMOUSEOUT
TIDY_ATTR_OnMOUSEOVER
TIDY_ATTR_OnMOUSEUP
TIDY_ATTR_OnRESET

TIDY_ATTR_OnROWENTER
TIDY_ATTR_OnROWEXIT
TIDY_ATTR_OnSELECT
TIDY_ATTR_OnSUBMIT
TIDY_ATTR_OnUNLOAD
TIDY_ATTR_PROFILE
TIDY_ATTR_PROMPT
TIDY_ATTR_RBSPAN
TIDY_ATTR_READONLY
TIDY_ATTR_REL
TIDY_ATTR_REV
TIDY_ATTR_RIGHTMARGIN
TIDY_ATTR_ROWS
TIDY_ATTR_ROWSPAN
TIDY_ATTR_RULES
TIDY_ATTR_SCHEME
TIDY_ATTR_SCOPE
TIDY_ATTR_SCROLLING
TIDY_ATTR_SELECTED
TIDY_ATTR_SHAPE
TIDY_ATTR_SHOWGRID
TIDY_ATTR_SHOWGRIDX
TIDY_ATTR_SHOWGRIDY
TIDY_ATTR_SIZE
TIDY_ATTR_SPAN
TIDY_ATTR_SRC

TIDY_ATTR_STANDBY
TIDY_ATTR_START
TIDY_ATTR_STYLE
TIDY_ATTR_SUMMARY
TIDY_ATTR_TABINDEX
TIDY_ATTR_TARGET
TIDY_ATTR_TEXT
TIDY_ATTR_TITLE
TIDY_ATTR_TOPMARGIN
TIDY_ATTR_TYPE
TIDY_ATTR_USEMAP
TIDY_ATTR_VALIGN
TIDY_ATTR_VALUE
TIDY_ATTR_VALUETYPE
TIDY_ATTR_VERSION
TIDY_ATTR_VLINK
TIDY_ATTR_VSPACE
TIDY_ATTR_WIDTH
TIDY_ATTR_WRAP
TIDY_ATTR_XML_LANG
TIDY_ATTR_XML_SPACE
TIDY_ATTR_XMLNS

tidy nodetype constants

constant	**description**

TIDY_NODETYPE_ROOT	root node
TIDY_NODETYPE_DOCTYPE	doctype
TIDY_NODETYPE_COMMENT	HTML comment
TIDY_NODETYPE_PROCINS	Processing Instruction
TIDY_NODETYPE_TEXT	Text
TIDY_NODETYPE_START	start tag
TIDY_NODETYPE_END	end tag
TIDY_NODETYPE_STARTEND	empty tag
TIDY_NODETYPE_CDATA	CDATA
TIDY_NODETYPE_SECTION	XML section
TIDY_NODETYPE_ASP	ASP code
TIDY_NODETYPE_JSTE	JSTE code
TIDY_NODETYPE_PHP	PHP code
TIDY_NODETYPE_XMLDECL	XML declaration

Examples

Examples

This simple example shows basic Tidy usage.

Example #2 - Basic Tidy usage

```
<?php
ob_start();
?>
<html>a html document</html>
<?php
$html = ob_get_clean();

// Specify configuration
$config = array(
          'indent'         => true,
          'output-xhtml'   => true,
          'wrap'           => 200);

// Tidy
$tidy = new tidy;
$tidy->parseString($html, $config, 'utf8');
$tidy->cleanRepair();

// Output
echo $tidy;
?>
```

Tidy Functions

Predefined Classes

tidyNode

Methods

- tidyNode::getParent - Returns the parent of the current node
- tidyNode->hasChildren - Returns **TRUE** if the current node has children
- tidyNode->hasSiblings - Returns **TRUE** if the current node has siblings
- tidyNode->isAsp - Returns **TRUE** if the current node is ASP code
- tidyNode->isComment - Returns **TRUE** if the current node is a comment
- tidyNode->isHtml - Returns **TRUE** if the current node is HTML
- tidyNode->isJste - Returns **TRUE** if the current node is JSTE
- tidyNode->isPhp - Returns **TRUE** if the current node is PHP
- tidyNode->isText - Returns **TRUE** if the current node is Text (no markup)

Properties

- value - the value of the node (e.g. the html text)
- name - the name of the tag (e.g. html, a, etc..)
- type - the type of the node (one of the constants above, e.g. **TIDY_NODETYPE_PHP**)
- line* - the line where the node starts
- column* - the column where the node starts
- proprietary* - **TRUE** if the node refers to a proprietary tag
- id - the ID of the tag (one of the constants above, e.g. **TIDY_TAG_FRAME**)
- attribute - an array with the attributes of the current node, or **NULL** if there aren't any
- child - an array with the child tidyNode s, or **NULL** if there aren't any

Note
The properties marked with * are just available since PHP 5.1.0.

ob_tidyhandler

ob_tidyhandler -- ob_start callback function to repair the buffer

Description

string **ob_tidyhandler** (string `$input` [, int `$mode`])

ob_tidyhandler() is intended to be used as a callback function for ob_start() to repair the buffer.

Parameters

input
The buffer.

mode
The buffer mode.

Return Values

Returns the modified buffer.

Examples

Example #3 - ob_tidyhandler() example

```
<?php
ob_start('ob_tidyhandler');

echo '<p>test</i>';
?>
```

The above example will output:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 3.2//EN">
<html>
<head>
<title></title>
</head>
<body>
<p>test</p>
</body>
</html>
```

See Also

- ob_start()

tidy_access_count

tidy_access_count -- Returns the Number of Tidy accessibility warnings encountered for specified document

Description

int **tidy_access_count** (tidy `$object`)

tidy_access_count() returns the number of accessibility warnings found for the specified document.

Parameters

`object`
The Tidy object.

Return Values

Returns the number of warnings.

Examples

Example #4 - tidy_access_count() example

```
<?php

$html ='<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 3.2//EN">
<html><head><title>Title</title></head>
<body>

<p><img src="img.png"></p>

</body></html>';


// select the accessibility check level: 1, 2 or 3
$config = array('accessibility-check' => 3);

$tidy = new tidy();
$tidy->parseString($html, $config);
$tidy->CleanRepair();

/* Never forget to call this! */
$tidy->diagnose();

echo tidy_access_count($tidy); //5
```

```
?>
```

Notes

Note
Due to the design of the TidyLib, you must call tidy_diagnose() before tidy_access_count() or it will return always *0*. You must also need to enable the *accessibility-check* option.

See Also

- tidy_error_count()
- tidy_warning_count()

tidy_clean_repair

tidy_clean_repair -- Execute configured cleanup and repair operations on parsed markup

Description

Procedural style:

bool **tidy_clean_repair** (tidy `$object`)

Object oriented style:

bool **tidy->cleanRepair** (void)

This function cleans and repairs the given tidy *`object`*.

Example #5 - tidy_clean_repair() **example**

```
<?php
$html = '<p>test</I>';

$tidy = tidy_parse_string($html);
tidy_clean_repair($tidy);

echo $tidy;
?>
```

The above example will output:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 3.2//EN">
<html>
<head>
<title></title>
</head>
<body>
<p>test</p>
</body>
</html>
```

See also tidy_repair_file() and tidy_repair_string().

tidy_config_count

tidy_config_count -- Returns the Number of Tidy configuration errors encountered for specified document

Description

int **tidy_config_count** (tidy `$object`)

tidy_config_count() returns the number of errors encountered in the configuration of the specified tidy *`object`*.

Example #6 - tidy_config_count() **example**

```
<?php
$html = '<p>test</I>';

$config = array('doctype' => 'bogus');

$tidy = tidy_parse_string($html, $config);

/* This outputs 1, because 'bogus' isn't a valid doctype */
echo tidy_config_count($tidy);
?>
```

tidy::__construct

tidy::__construct -- Constructs a new tidy object

Description

tidy **tidy::__construct** ([string `$filename` [, mixed `$config` [, string `$encoding` [, bool `$use_include_path`]]]])

tidy::__construct() constructs a new tidy object.

If the *`filename`* parameter is given, this function will also read that file and initialize the object with the file, acting like tidy_parse_file().

The *`config`* parameter can be passed either as an array or as a string. If a string is passed, it is interpreted as the name of the configuration file, otherwise, it is interpreted as the options themselves. Check » http://tidy.sourceforge.net/docs/quickref.html for an explanation about each option.

The *`encoding`* parameter sets the encoding for input/output documents. The possible values for *`encoding`* are: *ascii*, *latin0*, *latin1*, *raw*, *utf8*, *iso2022*, *mac*, *win1252*, *ibm858*, *utf16*, *utf16le*, *utf16be*, *big5* and *shiftjis*.

Example #7 - tidy::__construct() example

```
<?php

$html = <<< HTML

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head><title>title</title></head>
<body>
<p>paragraph <bt />
text</p>
</body></html>

HTML;

$tidy = new tidy;
$tidy->parseString($html);

$tidy->CleanRepair();

if ($tidy->errorBuffer) {
   echo "The following errors were detected:\n";
   echo $tidy->errorBuffer;
}
```

```
?>
```

The above example will output:

```
The following errors were detected:
line 8 column 14 - Error: <bt> is not recognized!
line 8 column 14 - Warning: discarding unexpected <bt>
```

See also tidy_parse_file() and tidy_parse_string().

tidy_diagnose

tidy_diagnose -- Run configured diagnostics on parsed and repaired markup

Description

Procedural style:

bool **tidy_diagnose** (tidy `$object`)

Object oriented style:

bool **tidy->diagnose** (void)

tidy_diagnose() runs diagnostic tests on the given tidy *`object`*, adding some more information about the document in the error buffer.

Returns **TRUE** on success or **FALSE** on failure.

Example #8 - tidy_diagnose() **example**

```
<?php

$html = <<< HTML
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<p>paragraph</p>
HTML;

$tidy = tidy_parse_string($html);
$tidy->CleanRepair();

// note the difference between the two outputs
echo tidy_get_error_buffer($tidy) . "\n";

$tidy->diagnose();
echo tidy_get_error_buffer($tidy);

?>
```

The above example will output:

```
line 5 column 1 - Warning: <p> isn't allowed in <head> elements
line 5 column 1 - Warning: inserting missing 'title' element

line 5 column 1 - Warning: <p> isn't allowed in <head> elements
line 5 column 1 - Warning: inserting missing 'title' element
Info: Doctype given is "-//W3C//DTD XHTML 1.0 Strict//EN"
Info: Document content looks like XHTML 1.0 Strict
2 warnings, 0 errors were found!
```

See also tidy_get_error_buffer().

tidy_error_count

tidy_error_count -- Returns the Number of Tidy errors encountered for specified document

Description

int **tidy_error_count** (tidy `$object`)

tidy_error_count() returns the number of Tidy errors encountered for the specified document.

Example #9 - tidy_error_count() **example**

```
<?php
$html = '<p>test</i>
<bogustag>bogus</bogustag>';

$tidy = tidy_parse_string($html);

echo tidy_error_count($tidy) . "\n"; //1

echo $tidy->errorBuffer;
?>
```

The above example will output:

```
1
line 1 column 1 - Warning: missing <!DOCTYPE> declaration
line 1 column 8 - Warning: discarding unexpected </i>
line 2 column 1 - Error: <bogustag> is not recognized!
line 2 column 1 - Warning: discarding unexpected <bogustag>
line 2 column 16 - Warning: discarding unexpected </bogustag>
line 1 column 1 - Warning: inserting missing 'title' element
```

See also tidy_access_count() and tidy_warning_count().

tidy_get_body

tidy_get_body -- Returns a tidyNode Object starting from the <body> tag of the tidy parse tree

Description

Procedural style:

tidyNode **tidy_get_body** (tidy `$object`)

Object oriented style:

tidyNode **tidy->body** (void)

This function returns a tidyNode object starting from the <body> tag of the tidy parse tree.

Example #10 - tidy_get_body() example

```
<?php
$html = '
<html>
 <head>
   <title>test</title>
 </head>
 <body>
   <p>paragraph</p>
 </body>
</html>';

$tidy = tidy_parse_string($html);

$body = tidy_get_body($tidy);
echo $body->value;
?>
```

The above example will output:

```
<body>
<p>paragraph</p>
</body>
```

Note

This function is only available with Zend Engine 2 (PHP >= 5.0.0).

See also tidy_get_head() and tidy_get_html().

tidy_get_config

tidy_get_config -- Get current Tidy configuration

Description

Procedural style:

array **tidy_get_config** (tidy `$object`)

Object oriented style:

array **tidy->getConfig** (void)

tidy_get_config() returns an array with the configuration options in use by the given tidy *`object`*.

For an explanation about each option, visit » http://tidy.sourceforge.net/docs/quickref.html.

Example #11 - tidy_get_config() example

```
<?php
$html = '<p>test</p>';
$config = array('indent' => TRUE,
                'output-xhtml' => TRUE,
                'wrap' => 200);

$tidy = tidy_parse_string($html, $config);

print_r(tidy_get_config($tidy));
?>
```

The above example will output:

```
Array
(
   [indent-spaces] => 2
   [wrap] => 200
   [tab-size] => 8
   [char-encoding] => 1
   [input-encoding] => 3
   [output-encoding] => 1
   [newline] => 1
   [doctype-mode] => 1
   [doctype] =>
   [repeated-attributes] => 1
   [alt-text] =>
   [slide-style] =>
   [error-file] =>
   [output-file] =>
   [write-back] =>
   [markup] => 1
```

```
    [show-warnings] => 1
    [quiet] =>
    [indent] => 1
    [hide-endtags] =>
    [input-xml] =>
    [output-xml] => 1
    [output-xhtml] => 1
    [output-html] =>
    [add-xml-decl] =>
    [uppercase-tags] =>
    [uppercase-attributes] =>
    [bare] =>
    [clean] =>
    [logical-emphasis] =>
    [drop-proprietary-attributes] =>
    [drop-font-tags] =>
    [drop-empty-paras] => 1
    [fix-bad-comments] => 1
    [break-before-br] =>
    [split] =>
    [numeric-entities] =>
    [quote-marks] =>
    [quote-nbsp] => 1
    [quote-ampersand] => 1
    [wrap-attributes] =>
    [wrap-script-literals] =>
    [wrap-sections] => 1
    [wrap-asp] => 1
    [wrap-jste] => 1
    [wrap-php] => 1
    [fix-backslash] => 1
    [indent-attributes] =>
    [assume-xml-procins] =>
    [add-xml-space] =>
    [enclose-text] =>
    [enclose-block-text] =>
    [keep-time] =>
    [word-2000] =>
    [tidy-mark] =>
    [gnu-emacs] =>
    [gnu-emacs-file] =>
    [literal-attributes] =>
    [show-body-only] =>
    [fix-uri] => 1
    [lower-literals] => 1
    [hide-comments] =>
    [indent-cdata] =>
    [force-output] => 1
    [show-errors] => 6
    [ascii-chars] => 1
    [join-classes] =>
    [join-styles] => 1
    [escape-cdata] =>
    [language] =>
    [ncr] => 1
    [output-bom] => 2
    [replace-color] =>
    [css-prefix] =>
    [new-inline-tags] =>
    [new-blocklevel-tags] =>
```

```
    [new-empty-tags] =>
    [new-pre-tags] =>
    [accessibility-check] => 0
    [vertical-space] =>
    [punctuation-wrap] =>
    [merge-divs] => 1
)
```

See also tidy_reset_config() and tidy_save_config().

tidy_get_error_buffer

tidy_get_error_buffer -- Return warnings and errors which occurred parsing the specified document

Description

Procedural style:

string **tidy_get_error_buffer** (tidy `$object`)

Object oriented style (property):

tidy

string *errorBuffer*;

tidy_get_error_buffer() returns warnings and errors which occurred parsing the specified document.

Example #12 - tidy_get_error_buffer() **example**

```
<?php
$html = '<p>paragraph</p>';

$tidy = tidy_parse_string($html);

echo tidy_get_error_buffer($tidy);
/* or in OO: */
echo $tidy->errorBuffer;
?>
```

The above example will output:

```
line 1 column 1 - Warning: missing <!DOCTYPE> declaration
line 1 column 1 - Warning: inserting missing 'title' element
```

See also tidy_access_count(), tidy_error_count() and tidy_warning_count().

tidy_get_head

tidy_get_head -- Returns a tidyNode Object starting from the <head> tag of the tidy parse tree

Description

Procedural style:

tidyNode **tidy_get_head** (tidy `$object`)

Object oriented style:

tidyNode **tidy->head** (void)

This function returns a tidyNode object starting from the <head> tag of the tidy parse tree.

Example #13 - tidy_get_head() **example**

```
<?php
$html = '
<html>
 <head>
   <title>test</title>
 </head>
 <body>
   <p>paragraph</p>
 </body>
</html>';

$tidy = tidy_parse_string($html);

$head = tidy_get_head($tidy);
echo $head->value;
?>
```

The above example will output:

```
<head>
<title>test</title>
</head>
```

Note

This function is only available with Zend Engine 2 (PHP >= 5.0.0).

See also tidy_get_body() and tidy_get_html().

tidy_get_html_ver

tidy_get_html_ver -- Get the Detected HTML version for the specified document

Description

Procedural style:

int **tidy_get_html_ver** (tidy `$object`)

Object oriented style:

int **tidy->getHtmlVer** (void)

tidy_get_html_ver() returns the detected HTML version for the specified tidy *`object`*.

Warning
This function is not yet implemented in the Tidylib itself, so it always return *0*.

tidy_get_html

tidy_get_html -- Returns a tidyNode Object starting from the <html> tag of the tidy parse tree

Description

Procedural style:

tidyNode **tidy_get_html** (tidy `$object`)

Object oriented style:

tidyNode **tidy->html** (void)

This function returns a tidyNode object starting from the <html> tag of the tidy parse tree.

Example #14 - tidy_get_html() **example**

```
<?php
$html = '
<html>
 <head>
   <title>test</title>
 </head>
 <body>
   <p>paragraph</p>
 </body>
</html>';

$tidy = tidy_parse_string($html);

$html = tidy_get_html($tidy);
echo $html->value;
?>
```

The above example will output:

```
<html>
<head>
<title>test</title>
</head>
<body>
<p>paragraph</p>
</body>
</html>
```

Note
This function is only available with Zend Engine 2 (PHP >= 5.0.0).

See also tidy_get_body() and tidy_get_head().

tidy_get_opt_doc

tidy_get_opt_doc -- Returns the documentation for the given option name

Description

Procedural style:

string **tidy_get_opt_doc** (tidy `$object`, string `$optname`)

Object oriented style:

string **tidy->getOptDoc** (string `$optname`)

tidy_get_opt_doc() returns the documentation for the given option name.

Note
You need at least libtidy from 25 April, 2005 for this function be available.

Parameters

`object`
A tidy object

`optname`
The option name

Return Values

Returns a string if the option exists and has documentation available, or **FALSE** otherwise.

Examples

Example #15 - Print all options along with their documentation and default value

```
<?php

$tidy = new tidy;
$config = $tidy->getConfig();

ksort($config);

foreach ($config as $opt => $val) {
```

```
    if (!$doc = $tidy->getOptDoc($opt))
        $doc = 'no documentation available!';

    $val = ($tidy->getOpt($opt) === true)  ? 'true'  : $val;
    $val = ($tidy->getOpt($opt) === false) ? 'false' : $val;

    echo "<p><b>$opt</b> (default: '$val')<br />".
         "$doc</p><hr />\n";
}

?>
```

See Also

- tidy_get_config()
- tidy_getopt()

tidy_get_output

tidy_get_output -- Return a string representing the parsed tidy markup

Description

string **tidy_get_output** (tidy `$object`)

Gets a string with the repaired html.

Parameters

`object`
The tidy object.

Return Values

Returns the parsed tidy markup.

Examples

Example #16 - tidy_get_output() example

```
<?php

$html = '<p>paragraph</i>';
$tidy = tidy_parse_string($html);

$tidy->CleanRepair();

echo tidy_get_output($tidy);
?>
```

The above example will output:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 3.2//EN">
<html>
<head>
<title></title>
</head>
<body>
<p>paragraph</p>
</body>
</html>
```

tidy_get_release

tidy_get_release -- Get release date (version) for Tidy library

Description

Procedural style:

string **tidy_get_release** (void)

Object oriented style:

string **tidy->getRelease** (void)

Gets the release date of the Tidy library.

Return Values

Returns a string with the release date of the Tidy library.

tidy_get_root

tidy_get_root -- Returns a tidyNode object representing the root of the tidy parse tree

Description

Procedural style:

tidyNode **tidy_get_root** (tidy `$object`)

Object oriented style:

tidyNode **tidy->root** (void)

Returns a tidyNode object representing the root of the tidy parse tree.

Example #17 - dump nodes

```
<?php

$html = <<< HTML
<html><body>

<p>paragraph</p>
<br/>

</body></html>
HTML;

$tidy = tidy_parse_string($html);
dump_nodes($tidy->root(), 1);


function dump_nodes($node, $indent) {

   if($node->hasChildren()) {
       foreach($node->child as $child) {
           echo str_repeat('.', $indent*2) . ($child->name ? $child->name :
'"'.$child->value.'"'). "\n";

           dump_nodes($child, $indent+1);
       }
   }
}
?>
```

The above example will output:

```
..html
....head
......title
....body
```

```
......p
........"paragraph"
......br
```

Note

This function is only available with Zend Engine 2 (PHP >= 5.0.0).

tidy_get_status

tidy_get_status -- Get status of specified document

Description

Procedural style:

int **tidy_get_status** (tidy `$object`)

Object oriented style:

int **tidy->getStatus** (void)

tidy_get_status() returns the status for the specified tidy *`object`*. It returns 0 if no error/warning was raised, 1 for warnings or accessibility errors, or 2 for errors.

Example #18 - tidy_get_status() example

```
<?php
$html = '<p>paragraph</i>';
$tidy = tidy_parse_string($html);

$html2 = '<bogus>test</bogus>';
$tidy2 = tidy_parse_string($html2);

echo tidy_get_status($tidy); //1

echo tidy_get_status($tidy2); //2
?>
```

tidy_getopt

tidy_getopt -- Returns the value of the specified configuration option for the tidy document

Description

Procedural style:

mixed **tidy_getopt** (tidy `$object`, string `$option`)

Object oriented style:

mixed **tidy->getOpt** (string `$option`)

tidy_getopt() returns the value of the specified *`option`* for the specified tidy *`object`*. The return type depends on the type of the specified *`option`*. You will find a list with each configuration option and their types at: » http://tidy.sourceforge.net/docs/quickref.html.

Example #19 - tidy_getopt() example

```
<?php

$html ='<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 3.2//EN">
<html><head><title>Title</title></head>
<body>

<p><img src="img.png"></p>

</body></html>';

$config = array('accessibility-check' => 3,
                'alt-text' => 'some text');

$tidy = new tidy();
$tidy->parseString($html, $config);


var_dump($tidy->getOpt('accessibility-check')); //integer
var_dump($tidy->getOpt('lower-literals')); //boolean
var_dump($tidy->getOpt('alt-text')); //string

?>
```

The above example will output:

```
int(3)
bool(true)
string(9) "some text"
```

tidy_is_xhtml

tidy_is_xhtml -- Indicates if the document is a XHTML document

Description

Procedural style:

bool **tidy_is_xhtml** (tidy `$object`)

Object oriented style:

bool **tidy->isXhtml** (void)

This function returns **TRUE** if the specified tidy *`object`* is a XHTML document, or **FALSE** otherwise.

Warning
This function is not yet implemented in the Tidylib itself, so it always return **FALSE**.

tidy_is_xml

tidy_is_xml -- Indicates if the document is a generic (non HTML/XHTML) XML document

Description

Procedural style:

bool **tidy_is_xml** (tidy `$object`)

Object oriented style:

bool **tidy->isXml** (void)

This function returns **TRUE** if the specified tidy *`object`* is a generic XML document (non HTML/XHTML), or **FALSE** otherwise.

Warning
This function is not yet implemented in the Tidylib itself, so it always return **FALSE**.

tidy_load_config

tidy_load_config -- Load an ASCII Tidy configuration file with the specified encoding

Description

void **tidy_load_config** (string `$filename`, string `$encoding`)

This function loads a Tidy configuration file, with the specified *`encoding`*.

Parameters

`filename`

`encoding`

Return Values

No value is returned.

Notes

Note
This function is only available in Tidy 1.0. It became obsolete in Tidy 2.0, and thus has been removed.

tidy_node->get_attr

tidy_node->get_attr -- Return the attribute with the provided attribute id

Description

tidy_attr **tidy_node->get_attr** (int `$attrib_id`)

Warning
This function is currently not documented; only its argument list is available.

tidy_node->get_nodes

tidy_node->get_nodes -- Return an array of nodes under this node with the specified id

Description

array **tidy_node->get_nodes** (int `$node_id`)

Warning
This function is currently not documented; only its argument list is available.

tidy_node->next

tidy_node->next -- Returns the next sibling to this node

Description

tidy_node **tidy_node->next** (void)

Warning
This function is currently not documented; only its argument list is available.

tidy_node->prev

tidy_node->prev -- Returns the previous sibling to this node

Description

tidy_node **tidy_node->prev** (void)

Warning
This function is currently not documented; only its argument list is available.

tidy_parse_file

tidy_parse_file -- Parse markup in file or URI

Description

Procedural style:

tidy **tidy_parse_file** (string `$filename` [, mixed `$config` [, string `$encoding` [, bool `$use_include_path`]]])

Object oriented style:

bool **tidy->parseFile** (string `$filename` [, mixed `$config` [, string `$encoding` [, bool `$use_include_path`]]])

This function parses the given file.

The *`config`* parameter can be passed either as an array or as a string. If a string is passed, it is interpreted as the name of the configuration file, otherwise, it is interpreted as the options themselves. Check » http://tidy.sourceforge.net/docs/quickref.html for an explanation about each option.

The *`encoding`* parameter sets the encoding for input/output documents. The possible values for *`encoding`* are: *ascii*, *latin0*, *latin1*, *raw*, *utf8*, *iso2022*, *mac*, *win1252*, *ibm858*, *utf16*, *utf16le*, *utf16be*, *big5* and *shiftjis*.

Example #20 - tidy_parse_file() example

```
<?php
$tidy = tidy_parse_file('file.html');

$tidy->cleanRepair();

if(!empty($tidy->errorBuffer)) {
   echo "The following errors or warnings occured:\n";
   echo $tidy->errorBuffer;
}
?>
```

Note

The optional parameters *`config`* and *`encoding`* were added in Tidy 2.0.

See also tidy_parse_string(), tidy_repair_file() and tidy_repair_string().

tidy_parse_string

tidy_parse_string -- Parse a document stored in a string

Description

Procedural style:

tidy **tidy_parse_string** (string `$input` [, mixed `$config` [, string `$encoding`]])

Object oriented style:

bool **tidy->parseString** (string `$input` [, mixed `$config` [, string `$encoding`]])

tidy_parse_string() parses a document stored in a string.

The *`config`* parameter can be passed either as an array or as a string. If a string is passed, it is interpreted as the name of the configuration file, otherwise, it is interpreted as the options themselves. Check » http://tidy.sourceforge.net/docs/quickref.html for an explanation about each option.

The *`encoding`* parameter sets the encoding for input/output documents. The possible values for *`encoding`* are: *ascii*, *latin0*, *latin1*, *raw*, *utf8*, *iso2022*, *mac*, *win1252*, *ibm858*, *utf16*, *utf16le*, *utf16be*, *big5* and *shiftjis*.

Example #21 - tidy_parse_string() example

```
<?php
ob_start();
?>

<html>
 <head>
  <title>test</title>
 </head>
 <body>
  <p>error<br>another line</i>
 </body>
</html>

<?php

$buffer = ob_get_clean();
$config = array('indent' => TRUE,
                'output-xhtml' => TRUE,
                'wrap' => 200);

$tidy = tidy_parse_string($buffer, $config, 'UTF8');

$tidy->cleanRepair();
echo $tidy;
```

```
?>
```

The above example will output:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
   <title>
     test
   </title>
 </head>
 <body>
   <p>
     error<br />
     another line
   </p>
 </body>
</html>
```

Note

The optional parameters *`config`* and *`encoding`* were added in Tidy 2.0.

See also tidy_parse_file(), tidy_repair_file() and tidy_repair_string().

tidy_repair_file

tidy_repair_file -- Repair a file and return it as a string

Description

string **tidy_repair_file** (string `$filename` [, mixed `$config` [, string `$encoding` [, bool `$use_include_path`]]])

This function repairs the given file and returns it as a string.

The *`config`* parameter can be passed either as an array or as a string. If a string is passed, it is interpreted as the name of the configuration file, otherwise, it is interpreted as the options themselves. Check » http://tidy.sourceforge.net/docs/quickref.html for an explanation about each option.

The *`encoding`* parameter sets the encoding for input/output documents. The possible values for *`encoding`* are: *ascii*, *latin0*, *latin1*, *raw*, *utf8*, *iso2022*, *mac*, *win1252*, *ibm858*, *utf16*, *utf16le*, *utf16be*, *big5* and *shiftjis*.

Example #22 - tidy_repair_file() example

```
<?php
$file = 'file.html';

$repaired = tidy_repair_file($file);
rename($file, $file . '.bak');

file_put_contents($file, $repaired);
?>
```

Note

The optional parameters *`config`* and *`encoding`* were added in Tidy 2.0.

See also tidy_parse_file(), tidy_parse_string() and tidy_repair_string().

tidy_repair_string

tidy_repair_string -- Repair a string using an optionally provided configuration file

Description

string **tidy_repair_string** (string `$data` [, mixed `$config` [, string `$encoding`]])

This function repairs the given string.

The *`config`* parameter can be passed either as an array or as a string. If a string is passed, it is interpreted as the name of the configuration file, otherwise, it is interpreted as the options themselves. Check » http://tidy.sourceforge.net/docs/quickref.html for an explanation about each option.

The *`encoding`* parameter sets the encoding for input/output documents. The possible values for *`encoding`* are: *ascii*, *latin0*, *latin1*, *raw*, *utf8*, *iso2022*, *mac*, *win1252*, *ibm858*, *utf16*, *utf16le*, *utf16be*, *big5* and *shiftjis*.

Example #23 - tidy_repair_string() **example**

```
<?php
ob_start();
?>

<html>
 <head>
   <title>test</title>
 </head>
 <body>
   <p>error</i>
 </body>
</html>

<?php

$buffer = ob_get_clean();
$tidy = tidy_repair_string($buffer);

echo $tidy;
?>
```

The above example will output:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 3.2//EN">
<html>
<head>
<title>test</title>
</head>
<body>
<p>error</p>
```

```
</body>
</html>
```

Note
The optional parameters *`config`* and *`encoding`* were added in Tidy 2.0.

See also tidy_parse_file(), tidy_parse_string() and tidy_repair_file().

tidy_reset_config

tidy_reset_config -- Restore Tidy configuration to default values

Description

bool **tidy_reset_config** (void)

This function restores the Tidy configuration to the default values.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Notes

Note
This function is only available in Tidy 1.0. It became obsolete in Tidy 2.0, and thus has been removed.

tidy_save_config

tidy_save_config -- Save current settings to named file

Description

bool **tidy_save_config** (string `$filename`)

Saves current settings to the specified file. Only non-default values are written.

Parameters

`filename`
: Path to the config file.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Notes

Note
This function is only available in Tidy 1.0. It became obsolete in Tidy 2.0, and thus has been removed.

See Also

- tidy_get_config()
- tidy_getopt()
- tidy_reset_config()
- tidy_setopt()

tidy_set_encoding

tidy_set_encoding -- Set the input/output character encoding for parsing markup

Description

bool **tidy_set_encoding** (string `$encoding`)

Sets the encoding for input/output documents.

Parameters

`encoding`
: Possible values for *`encoding`* are ascii, latin0, latin1, raw, utf8, iso2022, mac, win1252, ibm858, utf16, utf16le, utf16be, big5 and shiftjis.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Notes

Note
This function is only available in Tidy 1.0. It became obsolete in Tidy 2.0, and thus has been removed.

tidy_setopt

tidy_setopt -- Updates the configuration settings for the specified tidy document

Description

bool **tidy_setopt** (string `$option`, mixed `$value`)

tidy_setopt() updates the specified *`option`* with a new *`value`*. You will find a list with each configuration option at: » http://tidy.sourceforge.net/docs/quickref.html.

Example #24 - tidy_setopt() example

```
<?php
$html = '<p>test</i>';

$tidy = tidy_parse_string($html);

tidy_setopt('indent', FALSE);
?>
```

See also tidy_getopt(), tidy_get_config(), tidy_reset_config() and tidy_save_config().

Note

This function is only available in Tidy 1.0. It became obsolete in Tidy 2.0, and thus has been removed.

tidy_warning_count

tidy_warning_count -- Returns the Number of Tidy warnings encountered for specified document

Description

int **tidy_warning_count** (tidy `$object`)

tidy_warning_count() returns the number of Tidy warnings encountered for the specified document.

Parameters

`object`
The Tidy object.

Return Values

Returns the number of warnings.

Examples

Example #25 - tidy_warning_count() example

```
<?php
$html = '<p>test</i>
<bogustag>bogus</bogustag>';

$tidy = tidy_parse_string($html);

echo tidy_error_count($tidy) . "\n"; //1
echo tidy_warning_count($tidy) . "\n"; //5
?>
```

See Also

- tidy_error_count()
- tidy_access_count()

tidyNode->hasChildren

tidyNode->hasChildren -- Returns true if this node has children

Description

bool **tidyNode->hasChildren** (void)

Warning
This function is currently not documented; only its argument list is available.

Note
This function was named **tidy_node->has_children()** in PHP 4/Tidy 1.

tidyNode->hasSiblings

tidyNode->hasSiblings -- Returns true if this node has siblings

Description

bool **tidyNode->hasSiblings** (void)

Warning
This function is currently not documented; only its argument list is available.

Note
This function was named **tidy_node->has_siblings()** in PHP 4/Tidy 1.

tidyNode->isAsp

tidyNode->isAsp -- Returns true if this node is ASP

Description

bool **tidyNode->isAsp** (void)

This functions returns **TRUE** if the current node is ASP, or **FALSE** otherwise.

Note
This function was named **tidy_node->is_asp()** in PHP 4/Tidy 1.

tidyNode->isComment

tidyNode->isComment -- Returns true if this node represents a comment

Description

bool **tidyNode->isComment** (void)

Warning
This function is currently not documented; only its argument list is available.

Note
This function was named **tidy_node->is_comment()** in PHP 4/Tidy 1.

tidyNode->isHtml

tidyNode->isHtml -- Returns true if this node is part of a HTML document

Description

bool **tidyNode->isHtml** (void)

Warning
This function is currently not documented; only its argument list is available.

Note
This function was named **tidy_node->is_html()** in PHP 4/Tidy 1.

tidyNode->isJste

tidyNode->isJste -- Returns true if this node is JSTE

Description

bool **tidyNode->isJste** (void)

Warning
This function is currently not documented; only its argument list is available.

Note
This function was named **tidy_node->is_jste()** in PHP 4/Tidy 1.

tidyNode->isPhp

tidyNode->isPhp -- Returns true if this node is PHP

Description

bool **tidyNode->isPhp** (void)

Returns **TRUE** if the current node is PHP code, **FALSE** otherwise.

Example #26 - get the PHP code from a mixed HTML/PHP document

```
<?php

$html = <<< HTML
<html><head>
<?php echo '<title>title</title>'; ?>
</head>
<body>

<?php
echo 'hello world!';
?>

</body></html>
HTML;


$tidy = tidy_parse_string($html);
$num = 0;

get_php($tidy->html());


function get_php($node) {

   // check if the current node is PHP code
   if($node->isPhp()) {
       echo "\n\n# PHP node #" . ++$GLOBALS['num'] . "\n";
       echo $node->value;
   }

   // check if the current node has childrens
   if($node->hasChildren()) {
       foreach($node->child as $child) {
           get_php($child);
       }
   }
}

?>
```

The above example will output:

```
# PHP node #1
<?php echo '<title>title</title>'; ?>

# PHP node #2
<?php
echo 'hello world!';
?>
```

Note

This function was named **tidy_node->is_php()** in PHP 4/Tidy 1.

tidyNode->isText

tidyNode->isText -- Returns true if this node represents text (no markup)

Description

bool **tidyNode->isText** (void)

Warning
This function is currently not documented; only its argument list is available.

Note
This function was named **tidy_node->is_text()** in PHP 4/Tidy 1.

tidyNode::getParent

tidyNode::getParent -- returns the parent node of the current node

Description

tidyNode **tidyNode::getParent** (void)

Returns the parent node of the current node.

Return Values

Returns a tidyNode if the node has a parent, or **NULL** otherwise.