
xlsx2html Documentation

Release 0.4.0

Apkawa

Jan 13, 2022

CONTENTS

1	Install	3
2	Usage	5
3	Limitations	9
4	Indices and tables	21
	Python Module Index	23
	Index	25

A simple export from xlsx format to html tables with keep cell formatting

INSTALL

- From pip

```
pip install xlsx2html
```

- From github dev version

```
pip install -e "git+https://github.com/Apkawa/xlsx2html.git#egg=xlsx2html"
```

1.1 Compatibly

Python	xlsx2html
2.7	0.1.10
3.5	0.2.1
>=3.6	latest

2.1 Simple usage

```
from xlsx2html import xlsx2html

out_stream = xlsx2html('path/to/example.xlsx')
out_stream.seek(0)
print(out_stream.read())
```

pass output file

```
from xlsx2html import xlsx2html

xlsx2html('path/to/example.xlsx', 'path/to/output.html')
```

use file like objects

```
import io
from xlsx2html import xlsx2html

# must be binary mode
xlsx_file = open('path/to/example.xlsx', 'rb')
out_file = io.StringIO()
xlsx2html(xlsx_file, out_file, locale='en')
out_file.seek(0)
result_html = out_file.read()
```

from shell

```
python -m xlsx2html path/to/example.xlsx path/to/output.html
```

2.2 Advanced usage

2.2.1 Use converter class

```
from xlsx2html import XLSX2HTMLConverter
converter = XLSX2HTMLConverter(
    filepath='path/to/example.xlsx',
    locale='de_DE',
    parse_formula=True,
    inline_styles=False
)
html = converter.get_html(sheet="sheet name")
```

2.2.2 Export sheet to only table

```
from xlsx2html import XLSX2HTMLConverter
converter = XLSX2HTMLConverter(
    filepath='path/to/example.xlsx',
    locale='de_DE',
    parse_formula=True,
    inline_styles=False
)
result = converter.get_table(sheet="sheet name", extra_attrs={'id': 'table_id'})

print(f"""
<html>
  <head>
    <style type="text/css">
      {result.css}
    </style>
  </head>
  <body>
    {result.html}
  </body>
</html>""")
```

2.2.3 Export all sheets

```
from xlsx2html import XLSX2HTMLConverter
converter = XLSX2HTMLConverter(
    filepath='path/to/example.xlsx',
    locale='de_DE',
    parse_formula=True,
    inline_styles=False
)
results = converter.get_tables(extra_attrs={'class': 'xlsx_sheet'})

css_str = '\n'.join([r.css for r in results])
tables_str = '\n'.join([r.html for r in results])
```

(continues on next page)

(continued from previous page)

```
print(f"""
<html>
  <head>
    <style type="text/css">
      {css_str}
    </style>
  </head>
  <body>
    {tables_str}
  </body>
</html>""")
```

2.2.4 use openpyxl.Workbook instance

```
import openpyxl
from xlsx2html import xlsx2html

XLSX_FILE = 'path/to/example.xlsx'

# Simple, but no work with parse_formula=True
out_file = xlsx2html(load_workbook(XLSX_FILE, data_only=True))

# Use converter

converter = XLSX2HTMLConverter(
    filepath=load_workbook(XLSX_FILE, data_only=True),
    parse_formula=True,
    formula_wb=load_workbook(XLSX_FILE, data_only=False),
)
out_file = converter.get_html_stream()
```


LIMITATIONS

- No support:
 - [] overline (no working with openpyxl and LibreOffice with xlsx)
 - [] conditional styles
 - [] charts
 - [] freezing panes
 - [] diagonal borders
 - [] pattern cell fill
- [] Have issue with border-collapse: collapse and merged cells

3.1 API Reference

3.1.1 Core

`xlsx2html.core.xlsx2html(filepath, output=None, locale='en', sheet=None, parse_formula=False, default_cell_border=None, inline_styles=False)`

Parameters

- **filepath** (Union[BinaryIO, str, Workbook]) – xlsx file
- **output** (Union[TextIO, str, None]) – to path or file like, defaults to *None*
- **locale** (str) – en or zh_TW. defaults to en
- **sheet** (Union[int, str, None]) – sheet name or idx, defaults to *None* what means get active sheet
- **parse_formula** (bool) – If *True* - enable parse formulas. defaults to *False*
- **default_cell_border** (Union[str, Dict[str, Optional[str]], None]) – default border style. Can use short str like 1px solid black or dict like {'width': '1px', 'style': 'solid', 'color': 'black'}
- **inline_styles** (bool) – store styles inline

Return type TextIO

Returns File like object

```
class xlsx2html.core.XLSX2HTMLConverter(filepath, locale='en', parse_formula=False, inline_styles=False,
                                         display_grid=False, default_border_style=None,
                                         formula_wb=None, parser=None, renderer=None)
```

Bases: object

Parameters

- **filepath** (*str* | *BinaryIO* | *openpyxl.Workbook*) – xlsx file
- **locale** (*str*) – en or zh_TW. defaults to en
- **parse_formula** (bool) – If *True* - enable parse formulas. defaults to *False*
- **formula_fb** – If *parse_formula* set to *True* and type *filepath* as *openpyxl.Workbook* then pass *formula_wb=openpyxl.load_workbook(filepath, data_only=False)*
- **default_border_style** (Union[*str*, Dict[*str*, Optional[*str*]], None]) – default border style. Can use short str like 1px solid black or dict like {'width': '1px', 'style': 'solid', 'color': 'black'}
- **inline_styles** (bool) – store styles inline
- **display_grid** (bool) – Show column letters and row numbers. If *XLSX2HTMLConverter.default_border_style* is none - do enabled gray grid

filepath: Union[BinaryIO, str, openpyxl.workbook.workbook.Workbook]

locale: str = 'en'

parse_formula: bool = False

inline_styles: bool = False

display_grid: bool = False

default_border_style: Optional[Union[str, Dict[str, Optional[str]]]] = None

wb: openpyxl.workbook.workbook.Workbook

formula_wb: Optional[openpyxl.workbook.workbook.Workbook] = None

parser: dataclasses.InitVar = None

renderer: dataclasses.InitVar = None

get_table(*sheet=None, extra_attrs=None*)

Parameters

- **sheet** (Union[int, str, None]) – sheet name or idx, defaults to *None* what means get active sheet
- **extra_attrs** (Optional[Dict[str, Optional[str]]]) – additional attributes for *<table>* like class or id

Return type *ConverterTableResult*

Returns

get_tables(*sheets=None, extra_attrs=None*)

Parameters

- **sheets** (Optional[Iterable[Union[int, str, None]]]) – list of sheet name or idx. By defaults get all sheets

- **extra_attrs** (Optional[Dict[str, Optional[str]]]) – additional attributes to `<table ...>` like class or id

Return type List[[ConverterTableResult](#)]

Returns

get_html(*sheet=None*)

Get full html with table

Parameters **sheet** (Union[int, str, None]) – sheet name or idx, defaults to *None* what means get active sheet

Return type str

Returns full html as string

get_html_stream(*output=None, sheet=None*)

Parameters

- **output** (Union[TextIO, str, None]) – to path or file like, defaults to *None*
- **sheet** (Union[int, str, None]) – sheet name or idx, defaults to *None* what means get active sheet

Return type TextIO

Returns File like object

class xlsx2html.core.**ConverterTableResult**(*html, css=""*)

Bases: object

Parameters

- **html** (str) – html of table
- **css** (str) – css contents If *XLSX2HTMLConverter.optimize_styles* set to *False* then css is empty

html: str

css: str = ''

3.1.2 Parser

class xlsx2html.parser.parser.**Column**(*index, letter, width, hidden=False*)

Bases: object

index: int

letter: str

width: float

hidden: bool = False

class xlsx2html.parser.parser.**MergedCellInfo**(*colspan=None, rowspan=None, cells=<factory>*)

Bases: object

colspan: Optional[int] = None

rowspan: Optional[int] = None

cells: List[openpyxl.cell.cell.Cell]

```
class xlsx2html.parser.parser.ParserResult(cols, rows, images)
    Bases: object
    cols: List[xlsx2html.parser.parser.Column]
    rows: List[List[xlsx2html.parser.cell.CellInfo]]
    images: Dict[Tuple[int, int], List[xlsx2html.parser.image.ImageInfo]]

class xlsx2html.parser.parser.XLSXParser(wb, locale='en', parse_formula=False, fb=None)
    Bases: object
    get_sheet_names()

        Return type List[str]
    get_sheet(sheet=None)

        Return type ParserResult
    static get_columns(ws, max_col)

        Return type List[Column]
    static get_images(ws)

        Return type Dict[Tuple[int, int], List[ImageInfo]]
    get_cell_data(cell, f_cell=None)

        Return type CellInfo
    static merge_borders(cells)

        Return type Optional[Borders]

class xlsx2html.parser.cell.Border(style, color=None)
    Bases: object
    style: str
    color: Optional[str] = None

class xlsx2html.parser.cell.Borders(top=None, left=None, right=None, bottom=None,
                                     diagonal_up=None, diagonal_down=None)
    Bases: object
    top: Optional[xlsx2html.parser.cell.Border] = None
    left: Optional[xlsx2html.parser.cell.Border] = None
    right: Optional[xlsx2html.parser.cell.Border] = None
    bottom: Optional[xlsx2html.parser.cell.Border] = None
    diagonal_up: Optional[xlsx2html.parser.cell.Border] = None
    diagonal_down: Optional[xlsx2html.parser.cell.Border] = None

class xlsx2html.parser.cell.Fill(pattern, color)
    Bases: object
```



```

    pattern: str
    color: Optional[str]
class xlsx2html.parser.cell.Font(size, color=None, italic=False, underline=False, bold=False,
                                strike=False, overline=False, outline=False, shadow=False)
    Bases: object
    size: int
    color: Optional[str] = None
    italic: bool = False
    underline: bool = False
    bold: bool = False
    strike: bool = False
    overline: bool = False
    outline: bool = False
    shadow: bool = False
class xlsx2html.parser.cell.Alignment(horizontal=None, vertical=None, indent=None,
                                     text_rotation=None)
    Bases: object
    horizontal: Optional[str] = None
    vertical: Optional[str] = None
    indent: Optional[float] = None
    text_rotation: Optional[int] = None
class xlsx2html.parser.cell.CellInfo(id, column, column_letter, row, coordinate, value, formatted_value,
                                     alignment, colspan=None, rowspan=None, height=19,
                                     border=None, fill=None, font=None)
    Bases: object
    id: str
    column: int
    column_letter: str
    row: int
    coordinate: str
    value: Any
    formatted_value: str
    alignment: xlsx2html.parser.cell.Alignment
    colspan: Optional[int] = None
    rowspan: Optional[int] = None
    height: int = 19
    border: Optional[xlsx2html.parser.cell.Borders] = None
    fill: Optional[xlsx2html.parser.cell.Fill] = None

```

font: Optional[[xlsx2html.parser.cell.Font](#)] = None

classmethod from_cell(*cell*, *f_cell*, *_locale=None*)

Return type [CellInfo](#)

static get_border(*cell*)

Return type Optional[[Borders](#)]

class xlsx2html.parser.image.Point(*x*, *y*)

Bases: object

x: int

y: int

class xlsx2html.parser.image.ImageInfo(*col*, *row*, *offset*, *width*, *height*, *src*)

Bases: object

col: int

row: int

offset: [xlsx2html.parser.image.Point](#)

width: int

height: int

src: str

classmethod from_ws_image(*ws_image*)

Return type [ImageInfo](#)

3.1.3 Renderer

class xlsx2html.render.html.HtmlRenderer(*display_grid=False*, *default_border_style=None*,
table_attrs=None, *inline_styles=False*)

Bases: object

render(*result*)

Return type str

render_table(*result*, *attrs=None*)

Return type str

render_header(*cols*)

Return type str

render_lineno(*lineno*)

Return type str

render_columns(*cols*)

Return type `str`

render_column(*col*)

Return type `str`

render_cell(*cell*, *images*, *attrs=None*)

Return type `str`

get_border_style_from_cell(*cell*)

Return type `Dict[str, Optional[str]]`

get_diagonal_border_style(*border*)

Return type `Dict`

get_styles_from_cell(*cell*, *extra_style=None*)

Return type `Dict[str, Optional[str]]`

render_image(*image*)

Return type `str`

build_style_cache(*rows*)

Return type `None`

render_css()

Return type `str`

3.1.4 Formatters

class `xlsx2html.format.number.ColorNumberPattern(*args, **kwargs)`

Bases: `babel.numbers.NumberPattern`

apply(*value*, *locale*, ***kwargs*)

Renders into a string a number following the defined pattern.

Forced decimal quantization is active by default so we'll produce a number string that is strictly following CLDR pattern definitions.

Parameters

- **value** (*decimal.Decimal* / *float* / *int*) – The value to format. If this is not a Decimal object, it will be cast to one.
- **locale** (*str* / *babel.core.Locale*) – The locale to use for formatting.
- **currency** (*str* / *None*) – Which currency, if any, to format as.

- **currency_digits** (*bool*) – Whether or not to use the currency’s precision. If false, the pattern’s precision is used.
- **decimal_quantization** (*bool*) – Whether decimal numbers should be forcibly quantized to produce a formatted output strictly matching the CLDR definition for the locale.
- **force_frac** – DEPRECATED - a forced override for *self.frac_prec* for a single formatting invocation.

Returns Formatted decimal string.

Return type str

apply_color(*value, formatted*)

Return type str

class xlsx2html.format.number.**PatternParser**(*pattern*)

Bases: object

apply(*number, locale*)

Return type str

xlsx2html.format.number.format_decimal(*number, format=None, locale='en_US_POSIX'*)

Return the given decimal number formatted for a specific locale.

```
>>> format_decimal(1.2345, locale='en_US')
'1.234'
>>> format_decimal(1.2346, locale='en_US')
'1.235'
>>> format_decimal(-1.2346, locale='en_US')
'-1.235'
>>> format_decimal(1.2345, locale='sv_SE')
'1,234'
>>> format_decimal(1.2345, locale='de')
'1,234'
```

The appropriate thousands grouping and the decimal separator are used for each locale:

```
>>> format_decimal(12345.5, locale='en_US')
'12,345.5'
```

Parameters

- **number** (Any) – the number to format
- **format** (Optional[str]) –
- **locale** (str) – the *Locale* object or locale identifier

Return type str

xlsx2html.format.dt.normalize_datetime_format(*fmt, fixed_for_time=False*)

Return type str

xlsx2html.format.dt.format_date(*date, fmt, locale='en_US_POSIX'*)

Return type str

`xlsx2html.format.dt.format_datetime(datetime, fmt, locale='en_US_POSIX', tzinfo=None)`

Return type str

`xlsx2html.format.dt.format_time(time, fmt, locale='en_US_POSIX', tzinfo=None, date=None)`

Return type str

`xlsx2html.format.dt.format_timedelta(timedelta, fmt)`

Return type str

`xlsx2html.format.locale.parse_locale_code(code)`

```
>>> parse_locale_code('-404')
'zh_Hant_TW'
>>> parse_locale_code('404')
'zh_Hant_TW'
>>> parse_locale_code('0404')
'zh_Hant_TW'
>>> parse_locale_code('58050')
```

Return type Optional[str]

`xlsx2html.format.locale.extract_locale_from_format(fmt)`

```
>>> extract_locale_from_format('[$-404]e/m/d')
('zh_Hant_TW', 'e/m/d')
>>> extract_locale_from_format('[$USD-404]e/m/d')
('zh_Hant_TW', 'e/m/d')
>>> extract_locale_from_format('[$$-404]#.00')
('zh_Hant_TW', '#.00')
>>> extract_locale_from_format('[RED]e/m/d')
(None, '[RED]e/m/d')
```

Return type Tuple[Optional[str], str]

class `xlsx2html.format.hyperlink.Hyperlink`(title, location=None, target=None, href=None)

Bases: object

title: str

location: Optional[str] = None

target: Optional[str] = None

href: Optional[str] = None

`xlsx2html.format.hyperlink.resolve_cell(worksheet, coord)`

Return type Cell

`xlsx2html.format.hyperlink.resolve_hyperlink_formula(cell, f_cell=None)`

Return type Optional[[Hyperlink](#)]

`xlsx2html.format.hyperlink.get_hyperlink(value, cell, f_cell=None)`

Return type Optional[[Hyperlink](#)]

`xlsx2html.format.hyperlink.format_hyperlink(value, cell, f_cell=None)`

Return type str

3.2 Contributors

3.2.1 Run tests

```
pip install -r requirements.txt
pytest # run tests
tox # run test matrix
```

3.2.2 Run tests with pyenv with specific python and pypy

```
pyenv install 3.10-dev pypy3.7-7.3.5
pyenv local 3.10-dev pypy3.7-7.3.5
pip install -r requirements.txt
tox -e py310,pypy3
```

3.2.3 Type checks

```
tox -e type
```

3.2.4 Lint code

```
tox -e qa
```

3.2.5 Before commit

Install git hook

```
pip install -r requirements.txt
pre-commit install
```

For pycharm needs install tox to global

3.2.6 Docs

```
pip install -r requirements.txt
cd docs
make html
```


INDICES AND TABLES

- `genindex`
- `modindex`
- `search`

PYTHON MODULE INDEX

X

- `xlsx2html.format.dt`, 16
- `xlsx2html.format.hyperlink`, 17
- `xlsx2html.format.locale`, 17
- `xlsx2html.format.number`, 15
- `xlsx2html.parser.cell`, 12
- `xlsx2html.parser.image`, 14
- `xlsx2html.parser.parser`, 11
- `xlsx2html.render.html`, 14

A

Alignment (class in *xlsx2html.parser.cell*), 13
alignment (*xlsx2html.parser.cell.CellInfo* attribute), 13
apply() (*xlsx2html.format.number.ColorNumberPattern* method), 15
apply() (*xlsx2html.format.number.PatternParser* method), 16
apply_color() (*xlsx2html.format.number.ColorNumberPattern* method), 16

B

bold (*xlsx2html.parser.cell.Font* attribute), 13
Border (class in *xlsx2html.parser.cell*), 12
border (*xlsx2html.parser.cell.CellInfo* attribute), 13
Borders (class in *xlsx2html.parser.cell*), 12
bottom (*xlsx2html.parser.cell.Borders* attribute), 12
build_style_cache() (*xlsx2html.render.html.HtmlRenderer* method), 15

C

CellInfo (class in *xlsx2html.parser.cell*), 13
cells (*xlsx2html.parser.parser.MergedCellInfo* attribute), 11
col (*xlsx2html.parser.image.ImageInfo* attribute), 14
color (*xlsx2html.parser.cell.Border* attribute), 12
color (*xlsx2html.parser.cell.Fill* attribute), 13
color (*xlsx2html.parser.cell.Font* attribute), 13
ColorNumberPattern (class in *xlsx2html.format.number*), 15
cols (*xlsx2html.parser.parser.ParserResult* attribute), 12
colspan (*xlsx2html.parser.cell.CellInfo* attribute), 13
colspan (*xlsx2html.parser.parser.MergedCellInfo* attribute), 11
Column (class in *xlsx2html.parser.parser*), 11
column (*xlsx2html.parser.cell.CellInfo* attribute), 13
column_letter (*xlsx2html.parser.cell.CellInfo* attribute), 13
ConverterTableResult (class in *xlsx2html.core*), 11
coordinate (*xlsx2html.parser.cell.CellInfo* attribute), 13
css (*xlsx2html.core.ConverterTableResult* attribute), 11

D

default_border_style (*xlsx2html.core.XLSX2HTMLConverter* attribute), 10
diagonal_down (*xlsx2html.parser.cell.Borders* attribute), 12
diagonal_up (*xlsx2html.parser.cell.Borders* attribute), 12
display_grid (*xlsx2html.core.XLSX2HTMLConverter* attribute), 10

E

extract_locale_from_format() (in module *xlsx2html.format.locale*), 17

F

filepath (*xlsx2html.core.XLSX2HTMLConverter* attribute), 10
Fill (class in *xlsx2html.parser.cell*), 12
fill (*xlsx2html.parser.cell.CellInfo* attribute), 13
Font (class in *xlsx2html.parser.cell*), 13
font (*xlsx2html.parser.cell.CellInfo* attribute), 13
format_date() (in module *xlsx2html.format.dt*), 16
format_datetime() (in module *xlsx2html.format.dt*), 17
format_decimal() (in module *xlsx2html.format.number*), 16
format_hyperlink() (in module *xlsx2html.format.hyperlink*), 18
format_time() (in module *xlsx2html.format.dt*), 17
format_timedelta() (in module *xlsx2html.format.dt*), 17
formatted_value (*xlsx2html.parser.cell.CellInfo* attribute), 13
formula_wb (*xlsx2html.core.XLSX2HTMLConverter* attribute), 10
from_cell() (*xlsx2html.parser.cell.CellInfo* class method), 14
from_ws_image() (*xlsx2html.parser.image.ImageInfo* class method), 14

G

`get_border()` (*xlsx2html.parser.cell.CellInfo* static method), 14

`get_border_style_from_cell()` (*xlsx2html.render.html.HtmlRenderer* method), 15

`get_cell_data()` (*xlsx2html.parser.parser.XLSXParser* method), 12

`get_columns()` (*xlsx2html.parser.parser.XLSXParser* static method), 12

`get_diagonal_border_style()` (*xlsx2html.render.html.HtmlRenderer* method), 15

`get_html()` (*xlsx2html.core.XLSX2HTMLConverter* method), 11

`get_html_stream()` (*xlsx2html.core.XLSX2HTMLConverter* method), 11

`get_hyperlink()` (in module *xlsx2html.format.hyperlink*), 18

`get_images()` (*xlsx2html.parser.parser.XLSXParser* static method), 12

`get_sheet()` (*xlsx2html.parser.parser.XLSXParser* method), 12

`get_sheet_names()` (*xlsx2html.parser.parser.XLSXParser* method), 12

`get_styles_from_cell()` (*xlsx2html.render.html.HtmlRenderer* method), 15

`get_table()` (*xlsx2html.core.XLSX2HTMLConverter* method), 10

`get_tables()` (*xlsx2html.core.XLSX2HTMLConverter* method), 10

H

`height` (*xlsx2html.parser.cell.CellInfo* attribute), 13

`height` (*xlsx2html.parser.image.ImageInfo* attribute), 14

`hidden` (*xlsx2html.parser.parser.Column* attribute), 11

`horizontal` (*xlsx2html.parser.cell.Alignment* attribute), 13

`href` (*xlsx2html.format.hyperlink.Hyperlink* attribute), 17

`html` (*xlsx2html.core.ConverterTableResult* attribute), 11

`HtmlRenderer` (class in *xlsx2html.render.html*), 14

`Hyperlink` (class in *xlsx2html.format.hyperlink*), 17

I

`id` (*xlsx2html.parser.cell.CellInfo* attribute), 13

`ImageInfo` (class in *xlsx2html.parser.image*), 14

`images` (*xlsx2html.parser.parser.ParserResult* attribute), 12

`indent` (*xlsx2html.parser.cell.Alignment* attribute), 13

`index` (*xlsx2html.parser.parser.Column* attribute), 11

`inline_styles` (*xlsx2html.core.XLSX2HTMLConverter* attribute), 10

`italic` (*xlsx2html.parser.cell.Font* attribute), 13

L

`left` (*xlsx2html.parser.cell.Borders* attribute), 12

`letter` (*xlsx2html.parser.parser.Column* attribute), 11

`locale` (*xlsx2html.core.XLSX2HTMLConverter* attribute), 10

`location` (*xlsx2html.format.hyperlink.Hyperlink* attribute), 17

M

`merge_borders()` (*xlsx2html.parser.parser.XLSXParser* static method), 12

`MergedCellInfo` (class in *xlsx2html.parser.parser*), 11

module

- xlsx2html.format.dt*, 16
- xlsx2html.format.hyperlink*, 17
- xlsx2html.format.locale*, 17
- xlsx2html.format.number*, 15
- xlsx2html.parser.cell*, 12
- xlsx2html.parser.image*, 14
- xlsx2html.parser.parser*, 11
- xlsx2html.render.html*, 14

N

`normalize_datetime_format()` (in module *xlsx2html.format.dt*), 16

O

`offset` (*xlsx2html.parser.image.ImageInfo* attribute), 14

`outline` (*xlsx2html.parser.cell.Font* attribute), 13

`overline` (*xlsx2html.parser.cell.Font* attribute), 13

P

`parse_formula` (*xlsx2html.core.XLSX2HTMLConverter* attribute), 10

`parse_locale_code()` (in module *xlsx2html.format.locale*), 17

`parser` (*xlsx2html.core.XLSX2HTMLConverter* attribute), 10

`ParserResult` (class in *xlsx2html.parser.parser*), 11

`pattern` (*xlsx2html.parser.cell.Fill* attribute), 12

`PatternParser` (class in *xlsx2html.format.number*), 16

`Point` (class in *xlsx2html.parser.image*), 14

R

`render()` (*xlsx2html.render.html.HtmlRenderer* method), 14

`render_cell()` (*xlsx2html.render.html.HtmlRenderer* method), 15

`render_column()` (*xlsx2html.render.html.HtmlRenderer* method), 15

[render_columns\(\)](#) (*xlsx2html.render.html.HtmlRenderer method*), 14
[render_css\(\)](#) (*xlsx2html.render.html.HtmlRenderer method*), 15
[render_header\(\)](#) (*xlsx2html.render.html.HtmlRenderer method*), 14
[render_image\(\)](#) (*xlsx2html.render.html.HtmlRenderer method*), 15
[render_lineno\(\)](#) (*xlsx2html.render.html.HtmlRenderer method*), 14
[render_table\(\)](#) (*xlsx2html.render.html.HtmlRenderer method*), 14
[renderer](#) (*xlsx2html.core.XLSX2HTMLConverter attribute*), 10
[resolve_cell\(\)](#) (*in module xlsx2html.format.hyperlink*), 17
[resolve_hyperlink_formula\(\)](#) (*in module xlsx2html.format.hyperlink*), 17
[right](#) (*xlsx2html.parser.cell.Borders attribute*), 12
[row](#) (*xlsx2html.parser.cell.CellInfo attribute*), 13
[row](#) (*xlsx2html.parser.image.ImageInfo attribute*), 14
[rows](#) (*xlsx2html.parser.parser.ParserResult attribute*), 12
[rowspan](#) (*xlsx2html.parser.cell.CellInfo attribute*), 13
[rowspan](#) (*xlsx2html.parser.parser.MergedCellInfo attribute*), 11

S

[shadow](#) (*xlsx2html.parser.cell.Font attribute*), 13
[size](#) (*xlsx2html.parser.cell.Font attribute*), 13
[src](#) (*xlsx2html.parser.image.ImageInfo attribute*), 14
[strike](#) (*xlsx2html.parser.cell.Font attribute*), 13
[style](#) (*xlsx2html.parser.cell.Border attribute*), 12

T

[target](#) (*xlsx2html.format.hyperlink.Hyperlink attribute*), 17
[text_rotation](#) (*xlsx2html.parser.cell.Alignment attribute*), 13
[title](#) (*xlsx2html.format.hyperlink.Hyperlink attribute*), 17
[top](#) (*xlsx2html.parser.cell.Borders attribute*), 12

U

[underline](#) (*xlsx2html.parser.cell.Font attribute*), 13

V

[value](#) (*xlsx2html.parser.cell.CellInfo attribute*), 13
[vertical](#) (*xlsx2html.parser.cell.Alignment attribute*), 13

W

[wb](#) (*xlsx2html.core.XLSX2HTMLConverter attribute*), 10
[width](#) (*xlsx2html.parser.image.ImageInfo attribute*), 14
[width](#) (*xlsx2html.parser.parser.Column attribute*), 11

X

[x](#) (*xlsx2html.parser.image.Point attribute*), 14
[xlsx2html\(\)](#) (*in module xlsx2html.core*), 9
[xlsx2html.format.dt](#) *module*, 16
[xlsx2html.format.hyperlink](#) *module*, 17
[xlsx2html.format.locale](#) *module*, 17
[xlsx2html.format.number](#) *module*, 15
[xlsx2html.parser.cell](#) *module*, 12
[xlsx2html.parser.image](#) *module*, 14
[xlsx2html.parser.parser](#) *module*, 11
[xlsx2html.render.html](#) *module*, 14
[XLSX2HTMLConverter](#) (*class in xlsx2html.core*), 9
[XLSXParser](#) (*class in xlsx2html.parser.parser*), 12

Y

[y](#) (*xlsx2html.parser.image.Point attribute*), 14