PHPWord Documentation

Release 0.18.2

The PHPWord Team

May 30, 2023

Contents

PHPWord is a library written in pure PHP that provides a set of classes to write to and read from different document file formats. The current version of PHPWord supports Microsoft Office Open XML (OOXML or OpenXML), OASIS Open Document Format for Office Applications (OpenDocument or ODF), and Rich Text Format (RTF).

CHAPTER 1

Introduction

PHPWord is a library written in pure PHP that provides a set of classes to write to and read from different document file formats. The current version of PHPWord supports Microsoft Office Open XML (OOXML or OpenXML), OASIS Open Document Format for Office Applications (OpenDocument or ODF), and Rich Text Format (RTF).

PHPWord is an open source project licensed under the terms of LGPL version 3. PHPWord is aimed to be a high quality software product. You can learn more about PHPWord by reading this Developers' Documentation.

1.1 Features

- Set document properties, e.g. title, subject, and creator.
- Create document sections with different settings, e.g. portrait/landscape, page size, and page numbering
- Create header and footer for each sections
- Set default font type, font size, and paragraph style
- Use UTF-8 and East Asia fonts/characters
- Define custom font styles (e.g. bold, italic, color) and paragraph styles (e.g. centered, multicolumns, spacing) either as named style or inline in text
- Insert paragraphs, either as a simple text or complex one (a text run) that contains other elements
- Insert titles (headers) and table of contents
- Insert text breaks and page breaks
- Insert right-to-left text
- Insert and format images, either local, remote, or as page watermarks
- Insert binary OLE Objects such as Excel or Visio
- Insert and format table with customized properties for each rows (e.g. repeat as header row) and cells (e.g. background color, rowspan, colspan)

- Insert list items as bulleted, numbered, or multilevel
- Insert hyperlinks
- Insert footnotes and endnotes
- Insert drawing shapes (arc, curve, line, polyline, rect, oval)
- Insert charts (pie, doughnut, bar, line, area, scatter, radar)
- Insert form fields (textinput, checkbox, and dropdown)
- Insert comments
- Create document from templates
- Use XSL 1.0 style sheets to transform headers, main document part, and footers of an OOXML template
- ... and many more features on progress

1.2 File formats

Below are the supported features for each file formats.

1.2.1 Writers

Features		OOXML	ODF	RTF	HTML	PDF
Document Properties	Standard	✓	✓	✓	✓	✓
	Custom	✓	✓			
Element Type	Text	✓	✓	✓	✓	✓
	Text Run	✓	✓	✓	✓	✓
	Title	✓	✓		✓	✓
	Link	✓	✓	✓	✓	✓
	Preserve Text	✓				
	Text Break	✓	✓	✓	✓	✓
	Page Break	✓		✓		
	List	✓				
	Table	✓	✓	✓	✓	✓
	Image	✓	✓	✓	✓	
	Object	✓				
	Watermark	✓				
	Table of Contents	✓				
	Header	✓				
	Footer	✓				
	Footnote	✓			✓	
	Endnote	✓			✓	
	Comments	✓				
Graphs	2D basic graphs	✓				
	2D advanced graphs					
	3D graphs	✓				
Math	OMML support					
	MathML support					
Bonus	Encryption					
	Protection					

1.2.2 Readers

Features		OOXML	DOC	ODF	RTF	HTML
Document Properties	Standard	✓				
	Custom	✓				
Element Type	Text	✓	✓	✓	✓	✓
	Text Run	✓				
	Title	✓		✓		
	Link	✓	✓			
	Preserve Text	✓				
	Text Break	✓	✓			
	Page Break	✓				
	List	✓		✓		✓
	Table	✓				✓
	Image	✓	✓			
	Object					
	Watermark					
	Table of Contents					
	Header	✓				
	Footer	✓				
	Footnote	✓				
	Endnote	✓				
	Comments					
Graphs	2D basic graphs					
	2D advanced graphs					
	3D graphs					
Math	OMML support					
	MathML support					
Bonus	Encryption					
	Protection					

1.3 Contributing

We welcome everyone to contribute to PHPWord. Below are some of the things that you can do to contribute.

- Read our contributing guide.
- Fork us and request a pull to the master branch.
- Submit bug reports or feature requests to GitHub.
- Follow @PHPWord and @PHPOffice on Twitter.

CHAPTER 2

Installing/configuring

2.1 Requirements

Mandatory:

- composer
- PHP 7.1+
- XML Parser extension

Optional:

- Zip extension
- GD extension
- XMLWriter extension
- XSL extension
- dompdf library

2.2 Installation

PHPWord is installed via Composer. You just need to add dependency on PHPWord into your package.

Example:

```
composer require phpoffice/phpword
```

If you are a developer or if you want to help us with testing then fetch the latest branch for developers. Notice: all contributions must be done against the developer branch.

Example:

```
composer require phpoffice/phpword:dev-master
```

2.3 Using samples

More examples are provided in the `samples` directory. For an easy access to those samples launch `php -S localhost:8000` in the samples directory then browse to http://localhost:8000 to view the samples.

CHAPTER 3

General usage

3.1 Basic example

The following is a basic example of the PHPWord library. More examples are provided in the samples folder.

```
<?php
require_once 'bootstrap.php';

// Creating the new document...
$phpWord = new \PhpOffice\PhpWord\PhpWord();

/* Note: any element you append to a document must reside inside of a Section. */

// Adding an empty Section to the document...
$section = $phpWord->addSection();
// Adding Text element to the Section having font styled by default...
$section->addText(
    '"Learn from yesterday, live for today, hope for tomorrow. '
        . 'The important thing is not to stop questioning." '
        . '(Albert Einstein)'
);

/*
 * Note: it's possible to customize font style of the Text element you add in three␣
↪ways:
 * - inline;
 * - using named font style (new font style object will be implicitly created);
 * - using explicitly created font style object.
 */

// Adding Text element with font customized inline...
$section->addText(
    '"Great achievement is usually born of great sacrifice, '
        . 'and is never the result of selfishness." '
```

(continues on next page)

(continued from previous page)

```
        . '(Napoleon Hill)',
    array('name' => 'Tahoma', 'size' => 10)
);

// Adding Text element with font customized using named font style...
$fontStyleName = 'oneUserDefinedStyle';
$phpWord->addFontStyle(
    $fontStyleName,
    array('name' => 'Tahoma', 'size' => 10, 'color' => '1B2232', 'bold' => true)
);
$section->addText(
    '"The greatest accomplishment is not in never falling, '
        . 'but in rising again after you fall." '
        . '(Vince Lombardi)',
    $fontStyleName
);

// Adding Text element with font customized using explicitly created font style␣
↪object...
$fontStyle = new \PhpOffice\PhpWord\Style\Font();
$fontStyle->setBold(true);
$fontStyle->setName('Tahoma');
$fontStyle->setSize(13);
$myTextElement = $section->addText('"Believe you can and you\'re halfway there."␣
↪(Theodor Roosevelt)');
$myTextElement->setFontStyle($fontStyle);

// Saving the document as OOXML file...
$objWriter = \PhpOffice\PhpWord\IOFactory::createWriter($phpWord, 'Word2007');
$objWriter->save('helloWorld.docx');

// Saving the document as ODF file...
$objWriter = \PhpOffice\PhpWord\IOFactory::createWriter($phpWord, 'ODText');
$objWriter->save('helloWorld.odt');

// Saving the document as HTML file...
$objWriter = \PhpOffice\PhpWord\IOFactory::createWriter($phpWord, 'HTML');
$objWriter->save('helloWorld.html');

/* Note: we skip RTF, because it's not XML-based and requires a different example. */
/* Note: we skip PDF, because "HTML-to-PDF" approach is used to create PDF documents.␣
↪*/
```

3.2 PHPWord Settings

The `PhpOffice\PhpWord\Settings` class provides some options that will affect the behavior of PHPWord. Below are the options.

3.2.1 XML Writer compatibility

This option sets XMLWriter::setIndent and XMLWriter::setIndentString. The default value of this option is `true` (compatible), which is required for OpenOffice to render OOXML document correctly. You can set this option to `false` during development to make the resulting XML file easier to read.

```
\PhpOffice\PhpWord\Settings::setCompatibility(false);
```

3.2.2 Zip class

By default, PHPWord uses Zip extension to deal with ZIP compressed archives and files inside them. If you can't have Zip extension installed on your server, you can use pure PHP library alternative, PclZip, which is included in PHPWord.

```
\PhpOffice\PhpWord\Settings::setZipClass(\PhpOffice\PhpWord\Settings::PCLZIP);
```

3.2.3 Output escaping

Writing documents of some formats, especially XML-based, requires correct output escaping. Without it your document may become broken when you put special characters like ampersand, quotes, and others in it.

Escaping can be performed in two ways: outside of the library by a software developer and inside of the library by built-in mechanism. By default, the built-in mechanism is disabled for backward compatibility with versions prior to v0.13.0. To turn it on set `outputEscapingEnabled` option to `true` in your PHPWord configuration file or use the following instruction at runtime:

```
\PhpOffice\PhpWord\Settings::setOutputEscapingEnabled(true);
```

3.2.4 Default Paper

By default, all sections of the document will print on A4 paper. You can alter the default paper by using the following function:

```
\PhpOffice\PhpWord\Settings::setDefaultPaper('Letter');
```

3.2.5 Default font

By default, every text appears in Arial 10 point. You can alter the default font by using the following two functions:

```
$phpWord->setDefaultFontName('Times New Roman');
$phpWord->setDefaultFontSize(12);
```

3.3 Document settings

Settings for the generated document can be set using `$phpWord->getSettings()`

3.3.1 Magnification Setting

The default zoom value is 100 percent. This can be changed either to another percentage

```
$phpWord->getSettings()->setZoom(75);
```

Or to predefined values `fullPage`, `bestFit`, `textFit`

```
$phpWord->getSettings()->setZoom(Zoom::BEST_FIT);
```

3.3.2 Mirroring the Page Margins

Use mirror margins to set up facing pages for double-sided documents, such as books or magazines.

```
$phpWord->getSettings()->setMirrorMargins(true);
```

3.3.3 Spelling and grammatical checks

By default spelling and grammatical errors are shown as soon as you open a word document. For big documents this can slow down the opening of the document. You can hide the spelling and/or grammatical errors with:

```
$phpWord->getSettings()->setHideGrammaticalErrors(true);
$phpWord->getSettings()->setHideSpellingErrors(true);
```

You can also specify the status of the spell and grammar checks, marking spelling or grammar as dirty will force a re-check when opening the document.

```
$proofState = new \PhpOffice\PhpWord\ComplexType\ProofState();
$proofState->setGrammar(\PhpOffice\PhpWord\ComplexType\ProofState::CLEAN);
$proofState->setSpelling(\PhpOffice\PhpWord\ComplexType\ProofState::DIRTY);

$phpWord->getSettings()->setProofState($proofState);
```

3.3.4 Track Revisions

Track changes can be activated using `setTrackRevisions`, you can furture specify

- Not to use move syntax, instead moved items will be seen as deleted in one place and added in another
- Not track formatting revisions

```
$phpWord->getSettings()->setTrackRevisions(true);
$phpWord->getSettings()->setDoNotTrackMoves(true);
$phpWord->getSettings()->setDoNotTrackFormatting(true);
```

3.3.5 Decimal Symbol

The default symbol to represent a decimal figure is the `.` in english. In french you might want to change it to `,` for instance.

```
$phpWord->getSettings()->setDecimalSymbol(',');
```

3.3.6 Document Language

The default language of the document can be change with the following.

```
$phpWord->getSettings()->setThemeFontLang(new Language(Language::FR_BE));
```

`Language` has 3 parameters, one for Latin languages, one for East Asian languages and one for Complex (Bi-Directional) languages. A couple of language codes are provided in the `PhpOffice\PhpWord\Style\Language` class but any valid code/ID can be used.

In case you are generating an RTF document the language need to be set differently.

```
$lang = new Language();
$lang->setLangId(Language::EN_GB_ID);
$phpWord->getSettings()->setThemeFontLang($lang);
```

3.4 Document information

You can set the document information such as title, creator, and company name. Use the following functions:

```
$properties = $phpWord->getDocInfo();
$properties->setCreator('My name');
$properties->setCompany('My factory');
$properties->setTitle('My title');
$properties->setDescription('My description');
$properties->setCategory('My category');
$properties->setLastModifiedBy('My name');
$properties->setCreated(mktime(0, 0, 0, 3, 12, 2014));
$properties->setModified(mktime(0, 0, 0, 3, 14, 2014));
$properties->setSubject('My subject');
$properties->setKeywords('my, key, word');
```

3.5 Measurement units

The base length unit in Open Office XML is twip. Twip means "TWentieth of an Inch Point", i.e. 1 twip = 1/1440 inch.

You can use PHPWord helper functions to convert inches, centimeters, or points to twip.

```
// Paragraph with 6 points space after
$phpWord->addParagraphStyle('My Style', array(
    'spaceAfter' => \PhpOffice\PhpWord\Shared\Converter::pointToTwip(6))
);

$section = $phpWord->addSection();
$sectionStyle = $section->getStyle();
// half inch left margin
$sectionStyle->setMarginLeft(\PhpOffice\PhpWord\Shared\Converter::inchToTwip(.5));
// 2 cm right margin
$sectionStyle->setMarginRight(\PhpOffice\PhpWord\Shared\Converter::cmToTwip(2));
```

3.6 Document protection

The document (or parts of it) can be password protected.

```
$documentProtection = $phpWord->getSettings()->getDocumentProtection();
$documentProtection->setEditing(DocProtect::READ_ONLY);
$documentProtection->setPassword('myPassword');
```

3.7 Automatically Recalculate Fields on Open

To force an update of the fields present in the document, set updateFields to true

```
$phpWord->getSettings()->setUpdateFields(true);
```

3.8 Hyphenation

Hyphenation describes the process of breaking words with hyphens. There are several options to control hyphenation.

3.8.1 Auto hyphenation

To automatically hyphenate text set `autoHyphenation` to `true`.

```
$phpWord->getSettings()->setAutoHyphenation(true);
```

3.8.2 Consecutive Hyphen Limit

The maximum number of consecutive lines of text ending with a hyphen can be controlled by the `consecutiveHyphenLimit` option. There is no limit if the option is not set or the provided value is `0`.

```
$phpWord->getSettings()->setConsecutiveHyphenLimit(2);
```

3.8.3 Hyphenation Zone

The hyphenation zone (in *twip*) is the allowed amount of whitespace before hyphenation is applied. The smaller the hyphenation zone the more words are hyphenated. Or in other words, the wider the hyphenation zone the less words are hyphenated.

```
$phpWord->getSettings()->
↪setHyphenationZone(\PhpOffice\PhpWord\Shared\Converter::cmToTwip(1));
```

3.8.4 Hyphenate Caps

To control whether or not words in all capital letters shall be hyphenated use the *doNotHyphenateCaps* option.

```
$phpWord->getSettings()->setDoNotHyphenateCaps(true);
```

CHAPTER 4

Containers

Containers are objects where you can put elements (texts, lists, tables, etc). There are 3 main containers, i.e. sections, headers, and footers. There are 3 elements that can also act as containers, i.e. textruns, table cells, and footnotes.

4.1 Sections

Every visible element in word is placed inside of a section. To create a section, use the following code:

```
$section = $phpWord->addSection($sectionStyle);
```

The `$sectionStyle` is an optional associative array that sets the section. Example:

```
$sectionStyle = array(
    'orientation' => 'landscape',
    'marginTop' => 600,
    'colsNum' => 2,
);
```

4.1.1 Page number

You can change a section page number by using the `pageNumberingStart` style of the section.

```
// Method 1
$section = $phpWord->addSection(array('pageNumberingStart' => 1));

// Method 2
$section = $phpWord->addSection();
$section->getStyle()->setPageNumberingStart(1);
```

4.1.2 Multicolumn

You can change a section layout to multicolumn (like in a newspaper) by using the `breakType` and `colsNum` style of the section.

```
// Method 1
$section = $phpWord->addSection(array('breakType' => 'continuous', 'colsNum' => 2));

// Method 2
$section = $phpWord->addSection();
$section->getStyle()->setBreakType('continuous');
$section->getStyle()->setColsNum(2);
```

4.1.3 Line numbering

You can apply line numbering to a section by using the `lineNumbering` style of the section.

```
// Method 1
$section = $phpWord->addSection(array('lineNumbering' => array()));

// Method 2
$section = $phpWord->addSection();
$section->getStyle()->setLineNumbering(array());
```

Below are the properties of the line numbering style.

- `start` Line numbering starting value
- `increment` Line number increments
- `distance` Distance between text and line numbering in *twip*
- `restart` Line numbering restart setting continuous|newPage|newSection

4.2 Headers

Each section can have its own header reference. To create a header use the `addHeader` method:

```
$header = $section->addHeader();
```

Be sure to save the result in a local object. You can use all elements that are available for the footer. See "Footer" section for detail. Additionally, only inside of the header reference you can add watermarks or background pictures. See "Watermarks" section.

You can pass an optional parameter to specify where the header/footer should be applied, it can be

- `Footer::AUTO` default, all pages except if overridden by first or even
- `Footer::FIRST` each first page of the section
- `Footer::EVEN` each even page of the section. Will only be applied if the evenAndOddHeaders is set to true in phpWord->settings

To change the evenAndOddHeaders use the `getSettings` method to return the Settings object, and then call the `setEvenAndOddHeaders` method:

```
$phpWord->getSettings()->setEvenAndOddHeaders(true);
```

4.3 Footers

Each section can have its own footer reference. To create a footer, use the `addFooter` method:

```
$footer = $section->addFooter();
```

Be sure to save the result in a local object to add elements to a footer. You can add the following elements to footers:

- Texts `addText` and `createTextrun`
- Text breaks
- Images
- Tables
- Preserve text

See the "Elements" section for the detail of each elements.

4.4 Other containers

Textruns, table cells, and footnotes are elements that can also act as containers. See the corresponding "Elements" section for the detail of each elements.

CHAPTER 5

Elements

Below are the matrix of element availability in each container. The column shows the containers while the rows lists the elements.

Num	Element	Section	Header	Footer	Cell	Text Run	Footnote
1	Text	v	v	v	v	v	v
2	Text Run	v	v	v	v	•	•
3	Link	v	v	v	v	v	v
4	Title	v	?	?	?	?	?
5	Preserve Text	?	v	v	v*	•	•
6	Text Break	v	v	v	v	v	v
7	Page Break	v	•	•	•	•	•
8	List	v	v	v	v	•	•
9	Table	v	v	v	v	•	•
10	Image	v	v	v	v	v	v
11	Watermark	•	v	•	•	•	•
12	OLEObject	v	v	v	v	v	v
13	TOC	v	•	•	•	•	•
14	Footnote	v	•	•	v**	v**	•
15	Endnote	v	•	•	v**	v**	•
16	CheckBox	v	v	v	v	v	•
17	TextBox	v	v	v	v	•	•
18	Field	v	v	v	v	v	v
19	Line	v	v	v	v	v	v
20	Chart	v			v		

Legend:

- `v`. Available.
- `v*`. Available only when inside header/footer.
- `v**`. Available only when inside section.
- `-`. Not available.
- `?`. Should be available.

5.1 Texts

Text can be added by using `addText` and `addTextRun` methods. `addText` is used for creating simple paragraphs that only contain texts with the same style. `addTextRun` is used for creating complex paragraphs that contain text

with different style (some bold, other italics, etc) or other elements, e.g. images or links. The syntaxes are as follow:

```
$section->addText($text, [$fontStyle], [$paragraphStyle]);
$textrun = $section->addTextRun([$paragraphStyle]);
```

- `$text`. Text to be displayed in the document.
- `$fontStyle`. See *Font*.
- `$paragraphStyle`. See *Paragraph*.

For available styling options see *Font* and *Paragraph*.

If you want to enable track changes on added text you can mark it as INSERTED or DELETED by a specific user at a given time:

```
$text = $section->addText('Hello World!');
$text->setChanged(\PhpOffice\PhpWord\Element\ChangedElement::TYPE_INSERTED, 'Fred',␣
↪(new \DateTime()));
```

5.1.1 Titles

If you want to structure your document or build table of contents, you need titles or headings. To add a title to the document, use the `addTitleStyle` and `addTitle` method. If *depth* is 0, a Title will be inserted, otherwise a Heading1, Heading2, ...

```
$phpWord->addTitleStyle($depth, [$fontStyle], [$paragraphStyle]);
$section->addTitle($text, [$depth]);
```

- `depth`.
- `$fontStyle`. See *Font*.
- `$paragraphStyle`. See *Paragraph*.
- `$text`. Text to be displayed in the document. This can be *string* or a *PhpOfficePhpWordElementTextRun*

It's necessary to add a title style to your document because otherwise the title won't be detected as a real title.

5.1.2 Links

You can add Hyperlinks to the document by using the function addLink:

```
$section->addLink($linkSrc, [$linkName], [$fontStyle], [$paragraphStyle]);
```

- `$linkSrc`. The URL of the link.
- `$linkName`. Placeholder of the URL that appears in the document.
- `$fontStyle`. See *Font*.
- `$paragraphStyle`. See *Paragraph*.

5.1.3 Preserve texts

The `addPreserveText` method is used to add a page number or page count to headers or footers.

```
$footer->addPreserveText('Page {PAGE} of {NUMPAGES}.');
```

5.2 Breaks

5.2.1 Text breaks

Text breaks are empty new lines. To add text breaks, use the following syntax. All parameters are optional.

```
$section->addTextBreak([$breakCount], [$fontStyle], [$paragraphStyle]);
```

- `$breakCount`. How many lines.
- `$fontStyle`. See *Font*.
- `$paragraphStyle`. See *Paragraph*.

5.2.2 Page breaks

There are two ways to insert a page break, using the `addPageBreak` method or using the `pageBreakBefore` style of paragraph.

```
$section->addPageBreak();
```

5.3 Lists

Lists can be added by using `addListItem` and `addListItemRun` methods. `addListItem` is used for creating lists that only contain plain text. `addListItemRun` is used for creating complex list items that contains texts with different style (some bold, other italics, etc) or other elements, e.g. images or links. The syntaxes are as follow:

Basic usage:

```
$section->addListItem($text, [$depth], [$fontStyle], [$listStyle], [$paragraphStyle]);
$listItemRun = $section->addListItemRun([$depth], [$listStyle], [$paragraphStyle])
```

Parameters:

- `$text`. Text that appears in the document.
- `$depth`. Depth of list item.
- `$fontStyle`. See *Font*.
- `$listStyle`. List style of the current element TYPE_NUMBER, TYPE_ALPHANUM, TYPE_BULLET_FILLED, etc. See list of constants in PHPWord\Style\ListItem.
- `$paragraphStyle`. See *Paragraph*.

See `Sample_14_ListItem.php` for more code sample.

Advanced usage:

You can also create your own numbering style by changing the `$listStyle` parameter with the name of your numbering style.

```
$phpWord->addNumberingStyle(
    'multilevel',
    array(
        'type' => 'multilevel',
        'levels' => array(
            array('format' => 'decimal', 'text' => '%1.', 'left' => 360, 'hanging' =>␣
↪360, 'tabPos' => 360),
            array('format' => 'upperLetter', 'text' => '%2.', 'left' => 720, 'hanging
↪' => 360, 'tabPos' => 720),
        )
    )
);
$section->addListItem('List Item I', 0, null, 'multilevel');
$section->addListItem('List Item I.a', 1, null, 'multilevel');
$section->addListItem('List Item I.b', 1, null, 'multilevel');
$section->addListItem('List Item II', 0, null, 'multilevel');
```

For available styling options see *Numbering level*.

5.4 Tables

To add tables, rows, and cells, use the `addTable`, `addRow`, and `addCell` methods:

```
$table = $section->addTable([$tableStyle]);
$table->addRow([$height], [$rowStyle]);
$cell = $table->addCell($width, [$cellStyle]);
```

Table style can be defined with `addTableStyle`:

```
$tableStyle = array(
    'borderColor' => '006699',
    'borderSize'  => 6,
    'cellMargin'  => 50
);
$firstRowStyle = array('bgColor' => '66BBFF');
$phpWord->addTableStyle('myTable', $tableStyle, $firstRowStyle);
$table = $section->addTable('myTable');
```

For available styling options see *Table*.

5.4.1 Cell span

You can span a cell on multiple columns by using `gridSpan` or multiple rows by using `vMerge`.

```
$cell = $table->addCell(200);
$cell->getStyle()->setGridSpan(5);
```

See `Sample_09_Tables.php` for more code sample.

5.5 Images

To add an image, use the `addImage` method to sections, headers, footers, textruns, or table cells.

```
$section->addImage($src, [$style]);
```

- `$src`. String path to a local image, URL of a remote image or the image data, as a string. Warning: Do not pass user-generated strings here, as that would allow an attacker to read arbitrary files or perform server-side request forgery by passing file paths or URLs instead of image data.
- `$style`. See *Image*.

Examples:

```
$section = $phpWord->addSection();
$section->addImage(
    'mars.jpg',
    array(
        'width'         => 100,
        'height'        => 100,
        'marginTop'     => -1,
        'marginLeft'    => -1,
        'wrappingStyle' => 'behind'
    )
);
$footer = $section->addFooter();
$footer->addImage('http://example.com/image.php');
$textrun = $section->addTextRun();
$textrun->addImage('http://php.net/logo.jpg');
$source = file_get_contents('/path/to/my/images/earth.jpg');
$textrun->addImage($source);
```

5.5.1 Watermarks

To add a watermark (or page background image), your section needs a header reference. After creating a header, you can use the `addWatermark` method to add a watermark.

```
$section = $phpWord->addSection();
$header = $section->addHeader();
$header->addWatermark('resources/_earth.jpg', array('marginTop' => 200, 'marginLeft'␣
↪=> 55));
```

5.6 Objects

You can add OLE embeddings, such as Excel spreadsheets or PowerPoint presentations to the document by using `addOLEObject` method.

```
$section->addOLEObject($src, [$style]);
```

5.7 Table of contents

To add a table of contents (TOC), you can use the `addTOC` method. Your TOC can only be generated if you have add at least one title (See “Titles”).

```
$section->addTOC([$fontStyle], [$tocStyle], [$minDepth], [$maxDepth]);
```

- `$fontStyle`. See font style section.
- `$tocStyle`. See available options below.
- `$minDepth`. Minimum depth of header to be shown. Default 1.
- `$maxDepth`. Maximum depth of header to be shown. Default 9.

Options for `$tocStyle`:

- `tabLeader`. Fill type between the title text and the page number. Use the defined constants in `\PhpOffice\PhpWord\Style\TOC`.
- `tabPos`. The position of the tab where the page number appears in *twip*.
- `indent`. The indent factor of the titles in *twip*.

5.8 Footnotes & endnotes

You can create footnotes with `addFootnote` and endnotes with `addEndnote` in texts or textruns, but it's recommended to use textrun to have better layout. You can use `addText`, `addLink`, `addTextBreak`, `addImage`, `addOLEObject` on footnotes and endnotes.

On textrun:

```
$textrun = $section->addTextRun();
$textrun->addText('Lead text.');
$footnote = $textrun->addFootnote();
$footnote->addText('Footnote text can have ');
$footnote->addLink('http://test.com', 'links');
$footnote->addText('.');
$footnote->addTextBreak();
$footnote->addText('And text break.');
$textrun->addText('Trailing text.');
$endnote = $textrun->addEndnote();
$endnote->addText('Endnote put at the end');
```

On text:

```
$section->addText('Lead text.');
$footnote = $section->addFootnote();
$footnote->addText('Footnote text.');
```

By default the footnote reference number will be displayed with decimal number starting from 1. This number uses the `FooterReference` style which you can redefine with the `addFontStyle` method. Default value for this style is `array('superScript' => true);`

The footnote numbering can be controlled by setting the FootnoteProperties on the Section.

```
$fp = new \PhpOffice\PhpWord\ComplexType\FootnoteProperties();
//sets the position of the footnote (pageBottom (default), beneathText, sectEnd,␣
↪docEnd)
$fp->setPos(\PhpOffice\PhpWord\ComplexType\FootnoteProperties::POSITION_BENEATH_TEXT);
//set the number format to use (decimal (default), upperRoman, upperLetter, ...)
$fp->setNumFmt(\PhpOffice\PhpWord\SimpleType\NumberFormat::LOWER_ROMAN);
//force starting at other than 1
```

(continues on next page)

(continued from previous page)

```
$fp->setNumStart(2);
//when to restart counting (continuous (default), eachSect, eachPage)
$fp->setNumRestart(\PhpOffice\PhpWord\ComplexType\FootnoteProperties::RESTART_NUMBER_
↪EACH_PAGE);
//And finaly, set it on the Section
$section->setFootnoteProperties($fp);
```

5.9 Checkboxes

Checkbox elements can be added to sections or table cells by using `addCheckBox`.

```
$section->addCheckBox($name, $text, [$fontStyle], [$paragraphStyle])
```

- `$name`. Name of the check box.
- `$text`. Text to be displayed in the document.
- `$fontStyle`. See *Font*.
- `$paragraphStyle`. See *Paragraph*.

5.10 Textboxes

To be completed

5.11 Fields

Currently the following fields are supported:

- PAGE
- NUMPAGES
- DATE
- XE
- INDEX

```
$section->addField($fieldType, [$properties], [$options], [$fieldText], [$fontStyle])
```

- `$fontStyle`. See *Font*.

See `\PhpOffice\PhpWord\Element\Field` for list of properties and options available for each field type. Options which are not specifically defined can be added. Those must start with a `\`.

For instance for the INDEX field, you can do the following (See Index Field for list of available options):

```
//the $fieldText can be either a simple string
$fieldText = 'The index value';

//or a 'TextRun', to be able to format the text you want in the index
$fieldText = new TextRun();
```

(continues on next page)

(continued from previous page)

```
$fieldText->addText('My ');
$fieldText->addText('bold index', ['bold' => true]);
$fieldText->addText(' entry');
$section->addField('XE', array(), array(), $fieldText);

//this actually adds the index
$section->addField('INDEX', array(), array('\\e "   " \\h "A" \\c "3"'), 'right click␣
↪to update index');
```

5.12 Line

Line elements can be added to sections by using `addLine`.

```
$lineStyle = array('weight' => 1, 'width' => 100, 'height' => 0, 'color' => 635552);
$section->addLine($lineStyle);
```

Available line style attributes:

- `weight`. Line width in *twip*.
- `color`. Defines the color of stroke.
- `dash`. Line types: dash, rounddot, squaredot, dashdot, longdash, longdashdot, longdashdotdot.
- `beginArrow`. Start type of arrow: block, open, classic, diamond, oval.
- `endArrow`. End type of arrow: block, open, classic, diamond, oval.
- `width`. Line-object width in *pt*.
- `height`. Line-object height in *pt*.
- `flip`. Flip the line element: true, false.

5.13 Chart

Charts can be added using

```
$categories = array('A', 'B', 'C', 'D', 'E');
$series = array(1, 3, 2, 5, 4);
$chart = $section->addChart('line', $categories, $series, $style);
```

For available styling options see *Chart*.

check out the Sample_32_Chart.php for more options and styling.

5.14 Comments

Comments can be added to a document by using `addComment`. The comment can contain formatted text. Once the comment has been added, it can be linked to any element with `setCommentStart`.

```
// first create a comment
$comment= new \PhpOffice\PhpWord\Element\Comment('Authors name', new \DateTime(), 'my_
↪initials');
$comment->addText('Test', array('bold' => true));

// add it to the document
$phpWord->addComment($comment);

$textrun = $section->addTextRun();
$textrun->addText('This ');
$text = $textrun->addText('is');
// link the comment to the text you just created
$text->setCommentStart($comment);
```

If no end is set for a comment using the `setCommentEnd`, the comment will be ended automatically at the end of the element it is started on.

5.15 Track Changes

Track changes can be set on text elements. There are 2 ways to set the change information on an element. Either by calling the *setChangeInfo()*, or by setting the *TrackChange* instance on the element with *setTrackChange()*.

```
$phpWord = new \PhpOffice\PhpWord\PhpWord();

// New portrait section
$section = $phpWord->addSection();
$textRun = $section->addTextRun();

$text = $textRun->addText('Hello World! Time to ');

$text = $textRun->addText('wake ', array('bold' => true));
$text->setChangeInfo(TrackChange::INSERTED, 'Fred', time() - 1800);

$text = $textRun->addText('up');
$text->setTrackChange(new TrackChange(TrackChange::INSERTED, 'Fred'));

$text = $textRun->addText('go to sleep');
$text->setChangeInfo(TrackChange::DELETED, 'Barney', new \DateTime('@' . (time() -␣
↪3600)));
```

CHAPTER 6

Styles

6.1 Section

Available Section style options:

- `borderBottomColor`. Border bottom color.
- `borderBottomSize`. Border bottom size in *twip*.
- `borderLeftColor`. Border left color.
- `borderLeftSize`. Border left size in *twip*.
- `borderRightColor`. Border right color.
- `borderRightSize`. Border right size in *twip*.
- `borderTopColor`. Border top color.
- `borderTopSize`. Border top size in *twip*.
- `breakType`. Section break type (nextPage, nextColumn, continuous, evenPage, oddPage).
- `colsNum`. Number of columns.
- `colsSpace`. Spacing between columns.
- `footerHeight`. Spacing to bottom of footer.
- `gutter`. Page gutter spacing.
- `headerHeight`. Spacing to top of header.
- `marginTop`. Page margin top in *twip*.
- `marginLeft`. Page margin left in *twip*.
- `marginRight`. Page margin right in *twip*.
- `marginBottom`. Page margin bottom in *twip*.

- **`orientation`. Page orientation (`portrait`, which is default, or `landscape`).** See `\PhpOffice\PhpWord\Style\Section::ORIENTATION_...` class constants for possible values
- `pageSizeH`. Page height in *twip*. Implicitly defined by `orientation` option. Any changes are discouraged.
- `pageSizeW`. Page width in *twip*. Implicitly defined by `orientation` option. Any changes are discouraged.
- **`vAlign`. Vertical Page Alignment** See `\PhpOffice\PhpWord\SimpleType\VerticalJc` for possible values

6.2 Font

Available Font style options:

- `allCaps`. All caps, *true* or *false*.
- `bgColor`. Font background color, e.g. *FF0000*.
- `bold`. Bold, *true* or *false*.
- `color`. Font color, e.g. *FF0000*.
- `doubleStrikethrough`. Double strikethrough, *true* or *false*.
- **`fgColor`. Font highlight color, e.g. *yellow*, *green*, *blue*.** See `\PhpOffice\PhpWord\Style\Font::FGCOLOR_...` class constants for possible values
- `hint`. Font content type, *default*, *eastAsia*, or *cs*.
- `italic`. Italic, *true* or *false*.
- `name`. Font name, e.g. *Arial*.
- `rtl`. Right to Left language, *true* or *false*.
- `size`. Font size, e.g. *20*, *22*.
- `smallCaps`. Small caps, *true* or *false*.
- `strikethrough`. Strikethrough, *true* or *false*.
- `subScript`. Subscript, *true* or *false*.
- `superScript`. Superscript, *true* or *false*.
- **`underline`. Underline, *single*, *dash*, *dotted*, etc.** See `\PhpOffice\PhpWord\Style\Font::UNDERLINE_...` class constants for possible values
- **`lang`. Language, either a language code like *en-US*, *fr-BE*, etc. or an object (or as an array) if you need to set eastAsian o**
  See `\PhpOffice\PhpWord\Style\Language` class for some language codes.
- `position`. The text position, raised or lowered, in half points
- `hidden`. Hidden text, *true* or *false*.

6.3 Paragraph

Available Paragraph style options:

- **`alignment`. Supports all alignment modes since 1st Edition of ECMA-376 standard up till ISO/IEC 29500:2012.** See `\PhpOffice\PhpWord\SimpleType\Jc` class constants for possible values.

- `basedOn`. Parent style.
- `hanging`. Hanging indentation in *half inches*.
- `indent`. Indent (left indentation) in *half inches*.
- **`indentation`. An array of indentation key => value pairs in *twip*. Supports *left*, *right*, *firstLine* and *hanging* indentation** See `\PhpOffice\PhpWord\Style\Indentation` for possible identation types.
- `keepLines`. Keep all lines on one page, *true* or *false*.
- `keepNext`. Keep paragraph with next paragraph, *true* or *false*.
- `lineHeight`. Text line height, e.g. *1.0*, *1.5*, etc.
- `next`. Style for next paragraph.
- `pageBreakBefore`. Start paragraph on next page, *true* or *false*.
- `spaceBefore`. Space before paragraph in *twip*.
- `spaceAfter`. Space after paragraph in *twip*.
- `spacing`. Space between lines in *twip*. If spacingLineRule is auto, 240 (height of 1 line) will be added, so if you want a double line height, set this to 240.
- **`spacingLineRule`. Line Spacing Rule. *auto*, *exact*, *atLeast*** See `\PhpOffice\PhpWord\SimpleType\LineSpacin` class constants for possible values.
- `suppressAutoHyphens`. Hyphenation for paragraph, *true* or *false*.
- `tabs`. Set of custom tab stops.
- `widowControl`. Allow first/last line to display on a separate page, *true* or *false*.
- `contextualSpacing`. Ignore Spacing Above and Below When Using Identical Styles, *true* or *false*.
- `bidi`. Right to Left Paragraph Layout, *true* or *false*.
- `shading`. Paragraph Shading.
- **`textAlignment`. Vertical Character Alignment on Line.** See `\PhpOffice\PhpWord\SimpleType\TextAlignmen` class constants for possible values.

6.4 Table

Available Table style options:

- **`alignment`. Supports all alignment modes since 1st Edition of ECMA-376 standard up till ISO/IEC 29500:2012.** See `\PhpOffice\PhpWord\SimpleType\JcTable` and `\PhpOffice\PhpWord\SimpleType\Jc` class constants for possible values.
- `bgColor`. Background color, e.g. '9966CC'.
- `border(Top|Right|Bottom|Left)Color`. Border color, e.g. '9966CC'.
- `border(Top|Right|Bottom|Left)Size`. Border size in *twip*.
- `cellMargin(Top|Right|Bottom|Left)`. Cell margin in *twip*.
- `indent`. Table indent from leading margin. Must be an instance of `\PhpOffice\PhpWord\ComplexType\TblWidth`.
- `width`. Table width in Fiftieths of a Percent or Twentieths of a Point.

- `unit`. The unit to use for the width. One of `\PhpOffice\PhpWord\SimpleType\TblWidth`. Defaults to *auto*.
- `layout`. Table layout, either *fixed* or *autofit* See `\PhpOffice\PhpWord\Style\Table` for constants.
- `cellSpacing` Cell spacing in *twip*
- `position` Floating Table Positioning, see below for options
- `bidiVisual` Present table as Right-To-Left

Floating Table Positioning options:

- `leftFromText` Distance From Left of Table to Text in *twip*
- `rightFromText` Distance From Right of Table to Text in *twip*
- `topFromText` Distance From Top of Table to Text in *twip*
- `bottomFromText` Distance From Top of Table to Text in *twip*
- `vertAnchor` Table Vertical Anchor, one of `\PhpOffice\PhpWord\Style\TablePosition::VANCHOR_*`
- `horzAnchor` Table Horizontal Anchor, one of `\PhpOffice\PhpWord\Style\TablePosition::HANCHOR_*`
- `tblpXSpec` Relative Horizontal Alignment From Anchor, one of `\PhpOffice\PhpWord\Style\TablePosition::XA`
- `tblpX` Absolute Horizontal Distance From Anchorin *twip*
- `tblpYSpec` Relative Vertical Alignment From Anchor, one of `\PhpOffice\PhpWord\Style\TablePosition::YALI`
- `tblpY` Absolute Vertical Distance From Anchorin *twip*

Available Row style options:

- `cantSplit`. Table row cannot break across pages, *true* or *false*.
- `exactHeight`. Row height is exact or at least.
- `tblHeader`. Repeat table row on every new page, *true* or *false*.

Available Cell style options:

- `bgColor`. Background color, e.g. '9966CC'.
- `border(Top|Right|Bottom|Left)Color`. Border color, e.g. '9966CC'.
- `border(Top|Right|Bottom|Left)Size`. Border size in *twip*.
- `border(Top|Right|Bottom|Left)Style`. Border style. You can use constants from `\PhpOffice\PhpWord\SimpleType\Border`
- `gridSpan`. Number of columns spanned.
- **`textDirection(btLr|tbRl)`. Direction of text.** You can use constants `\PhpOffice\PhpWord\Style\Cell::TEXT_DIR_BTLR` and `\PhpOffice\PhpWord\Style\Cell::TEXT_DIR_TBRL`
- `valign`. Vertical alignment, *top*, *center*, *both*, *bottom*.
- `vMerge`. *restart* or *continue*.
- `width`. Cell width in *twip*.

6.5 Image

Available Image style options:

- `alignment`. See `\PhpOffice\PhpWord\SimpleType\Jc` class for the details.
- `height`. Height in *pt*.
- `marginLeft`. Left margin in inches, can be negative.
- `marginTop`. Top margin in inches, can be negative.
- `width`. Width in *pt*.
- `wrappingStyle`. Wrapping style, *inline*, *square*, *tight*, *behind*, or *infront*.
- `wrapDistanceTop`. Top text wrapping in pixels.
- `wrapDistanceBottom`. Bottom text wrapping in pixels.
- `wrapDistanceLeft`. Left text wrapping in pixels.
- `wrapDistanceRight`. Right text wrapping in pixels.

6.6 Numbering level

Available NumberingLevel style options:

- **`alignment`. Supports all alignment modes since 1st Edition of ECMA-376 standard up till ISO/IEC 29500:2012.** See `\PhpOffice\PhpWord\SimpleType\Jc` class constants for possible values.
- `font`. Font name.
- `format`. Numbering format bullet|decimal|upperRoman|lowerRoman|upperLetter|lowerLetter.
- `hanging`. See paragraph style.
- `hint`. See font style.
- `left`. See paragraph style.
- `restart`. Restart numbering level symbol.
- `start`. Starting value.
- `suffix`. Content between numbering symbol and paragraph text tab|space|nothing.
- `tabPos`. See paragraph style.
- `text`. Numbering level text e.g. %1 for nonbullet or bullet character.

6.7 Chart

Available Chart style options:

- `width`. Width (in EMU).
- `height`. Height (in EMU).
- `3d`. Is 3D; applies to pie, bar, line, area, *true* or *false*.
- `colors`. A list of colors to use in the chart.

- `title`. The title for the chart.
- `showLegend`. Show legend, *true* or *false*.
- `LegendPosition`. Legend position, *r* (default), *b*, *t*, *l* or *tr*.
- `categoryLabelPosition`. Label position for categories, *nextTo* (default), *low* or *high*.
- `valueLabelPosition`. Label position for values, *nextTo* (default), *low* or *high*.
- `categoryAxisTitle`. The title for the category axis.
- `valueAxisTitle`. The title for the values axis.
- `majorTickMarkPos`. The position for major tick marks, *in*, *out*, *cross*, *none* (default).
- `showAxisLabels`. Show labels for axis, *true* or *false*.
- `gridX`. Show Gridlines for X-Axis, *true* or *false*.
- `gridY`. Show Gridlines for Y-Axis, *true* or *false*.

CHAPTER 7

Templates processing

You can create an OOXML document template with included search-patterns (macros) which can be replaced by any value you wish. Only single-line values can be replaced. By default Macros are defined like this: `${search-pattern}` but you can define custom macros. To load a template file, create a new instance of the TemplateProcessor.

```
$templateProcessor = new TemplateProcessor('Template.docx');
```

7.1 setValue

Given a template containing

```
Hello ${firstname} ${lastname}!
```

The following will replace `${firstname}` with `John`, and `${lastname}` with `Doe` . The resulting document will now contain `Hello John Doe!`

```
$templateProcessor->setValue('firstname', 'John');
$templateProcessor->setValue('lastname', 'Doe');
```

7.2 setValues

You can also set multiple values by passing all of them in an array.

```
$templateProcessor->setValues(array('firstname' => 'John', 'lastname' => 'Doe'));
```

7.3 setMacroOpeningChars

You can define a custom opening macro. The following will set `{#` as the opening search pattern.

```
$templateProcessor->setMacroOpeningChars('{#');
```

7.4 setMacroClosingChars

You can define a custom closing macro. The following will set `#}` as the closing search pattern.

```
$templateProcessor->setMacroClosingChars('#}');
```

7.5 setMacroChars

You can define a custom opening and closing macro at the same time . The following will set the search-pattern like this: `{#search-pattern#}` .

```
$templateProcessor->setMacroChars('{#', '#}');
```

7.6 setImageValue

The search-pattern model for images can be like:

- `${search-image-pattern}`
- `${search-image-pattern:[width]:[height]:[ratio]}`
- `${search-image-pattern:[width]x[height]}`
- `${search-image-pattern:size=[width]x[height]}`
- `${search-image-pattern:width=[width]:height=[height]:ratio=false}`

Where:

- [width] and [height] can be just numbers or numbers with measure, which supported by Word (cm, mm, in, pt, pc, px, %, em, ex)
- [ratio] uses only for `false`, `-` or `f` to turn off respect aspect ration of image. By default template image size uses as 'container' size.

Example:

```
${CompanyLogo}
${UserLogo:50:50} ${Name} - ${City} - ${Street}
```

```
$templateProcessor = new TemplateProcessor('Template.docx');
$templateProcessor->setValue('Name', 'John Doe');
$templateProcessor->setValue(array('City', 'Street'), array('Detroit', '12th Street
↪'));

$templateProcessor->setImageValue('CompanyLogo', 'path/to/company/logo.png');
```

(continues on next page)

(continued from previous page)

```
$templateProcessor->setImageValue('UserLogo', array('path' => 'path/to/logo.png',
↪'width' => 100, 'height' => 100, 'ratio' => false));
$templateProcessor->setImageValue('FeatureImage', function () {
    // Closure will only be executed if the replacement tag is found in the template

    return array('path' => SlowFeatureImageGenerator::make(), 'width' => 100, 'height
↪' => 100, 'ratio' => false);
});
```

7.7 cloneBlock

Given a template containing See `Sample_23_TemplateBlock.php` for an example.

```
${block_name}
Customer: ${customer_name}
Address: ${customer_address}
${/block_name}
```

The following will duplicate everything between `${block_name}` and `${/block_name}` 3 times.

```
$templateProcessor->cloneBlock('block_name', 3, true, true);
```

The last parameter will rename any macro defined inside the block and add #1, #2, #3 ... to the macro name. The result will be

```
Customer: ${customer_name#1}
Address: ${customer_address#1}

Customer: ${customer_name#2}
Address: ${customer_address#2}

Customer: ${customer_name#3}
Address: ${customer_address#3}
```

It is also possible to pass an array with the values to replace the marcros with. If an array with replacements is passed, the `count` argument is ignored, it is the size of the array that counts.

```
$replacements = array(
    array('customer_name' => 'Batman', 'customer_address' => 'Gotham City'),
    array('customer_name' => 'Superman', 'customer_address' => 'Metropolis'),
);
$templateProcessor->cloneBlock('block_name', 0, true, false, $replacements);
```

The result will then be

```
Customer: Batman
Address: Gotham City

Customer: Superman
Address: Metropolis
```

7.8 replaceBlock

Given a template containing

```
${block_name}
This block content will be replaced
${/block_name}
```

The following will replace everything between `${block_name}` and `${/block_name}` with the value passed.

```
$templateProcessor->replaceBlock('block_name', 'This is the replacement text.');
```

7.9 deleteBlock

Same as previous, but it deletes the block

```
$templateProcessor->deleteBlock('block_name');
```

7.10 cloneRow

Clones a table row in a template document. See `Sample_07_TemplateCloneRow.php` for an example.

```
+-----------+----------------+
| ${userId} | ${userName}    |
|           |----------------+
|           | ${userAddress} |
+-----------+----------------+
```

```
$templateProcessor->cloneRow('userId', 2);
```

Will result in

```
+-------------+------------------+
| ${userId#1} | ${userName#1}    |
|             |------------------+
|             | ${userAddress#1} |
+-------------+------------------+
| ${userId#2} | ${userName#2}    |
|             |------------------+
|             | ${userAddress#2} |
+-------------+------------------+
```

7.11 cloneRowAndSetValues

Finds a row in a table row identified by *$search* param and clones it as many times as there are entries in *$values*.

```
+-----------+----------------+
| ${userId} | ${userName}    |
|           |----------------+
```

(continues on next page)

(continued from previous page)

```
|           | ${userAddress} |
+-----------+----------------+
```

```
$values = [
    ['userId' => 1, 'userName' => 'Batman', 'userAddress' => 'Gotham City'],
    ['userId' => 2, 'userName' => 'Superman', 'userAddress' => 'Metropolis'],
];
$templateProcessor->cloneRowAndSetValues('userId', $values);
```

Will result in

```
+---+-------------+
| 1 | Batman      |
|   |-------------+
|   | Gotham City |
+---+-------------+
| 2 | Superman    |
|   |-------------+
|   | Metropolis  |
+---+-------------+
```

7.12 applyXslStyleSheet

Applies the XSL stylesheet passed to header part, footer part and main part

```
$xslDomDocument = new \DOMDocument();
$xslDomDocument->load('/path/to/my/stylesheet.xsl');
$templateProcessor->applyXslStyleSheet($xslDomDocument);
```

7.13 setComplexValue

Replaces a ${macro} with the ComplexType passed. See `Sample_40_TemplateSetComplexValue.php` for examples.

```
$inline = new TextRun();
$inline->addText('by a red italic text', array('italic' => true, 'color' => 'red'));
$templateProcessor->setComplexValue('inline', $inline);
```

7.14 setComplexBlock

Replaces a ${macro} with the ComplexType passed. See `Sample_40_TemplateSetComplexValue.php` for examples.

```
$table = new Table(array('borderSize' => 12, 'borderColor' => 'green', 'width' =>␣
↪6000, 'unit' => TblWidth::TWIP));
$table->addRow();
$table->addCell(150)->addText('Cell A1');
$table->addCell(150)->addText('Cell A2');
```

(continues on next page)

(continued from previous page)

```
$table->addCell(150)->addText('Cell A3');
$table->addRow();
$table->addCell(150)->addText('Cell B1');
$table->addCell(150)->addText('Cell B2');
$table->addCell(150)->addText('Cell B3');
$templateProcessor->setComplexBlock('table', $table);
```

7.15 setChartValue

Replace a variable by a chart.

```
$categories = array('A', 'B', 'C', 'D', 'E');
$series1 = array(1, 3, 2, 5, 4);
$chart = new Chart('doughnut', $categories, $series1);
$templateProcessor->setChartValue('myChart', $chart);
```

7.16 save

Saves the loaded template within the current directory. Returns the file path.

```
$filepath = $templateProcessor->save();
```

7.17 saveAs

Saves a copy of the loaded template in the indicated path.

```
$pathToSave = 'path/to/save/file.ext';
$templateProcessor->saveAs($pathToSave);
```

CHAPTER 8

Writers & readers

8.1 OOXML

The package of OOXML document consists of the following files.

- _rels/
  - .rels
- docProps/
  - app.xml
  - core.xml
  - custom.xml
- word/
  - rels/
    - document.rels.xml
  - media/
  - theme/
    - theme1.xml
  - document.xml
  - fontTable.xml
  - numbering.xml
  - settings.xml
  - styles.xml
  - webSettings.xml
- [Content_Types].xml

8.2 OpenDocument

8.2.1 Package

The package of OpenDocument document consists of the following files.

- META-INF/
  - manifest.xml
- Pictures/
- content.xml
- meta.xml
- styles.xml

8.2.2 content.xml

The structure of `content.xml` is described below.

- office:document-content
  - office:font-facedecls
  - office:automatic-styles
  - office:body
    - office:text
      - draw:*
      - office:forms
      - table:table
      - text:list
      - text:numbered-paragraph
      - text:p
      - text:table-of-contents
      - text:section
    - office:chart
    - office:image
    - office:drawing

8.2.3 styles.xml

The structure of `styles.xml` is described below.

- office:document-styles
  - office:styles
  - office:automatic-styles

- office:master-styles
  - office:master-page

8.3 RTF

To be completed.

8.4 HTML

To be completed.

8.5 PDF

To be completed.

CHAPTER 9

Recipes

9.1 Create float left image

Use absolute positioning relative to margin horizontally and to line vertically.

```
$imageStyle = array(
    'width' => 40,
    'height' => 40,
    'wrappingStyle' => 'square',
    'positioning' => 'absolute',
    'posHorizontalRel' => 'margin',
    'posVerticalRel' => 'line',
);
$textrun->addImage('resources/_earth.jpg', $imageStyle);
$textrun->addText($lipsumText);
```

9.2 Download the produced file automatically

Use `php://output` as the filename.

```
$phpWord = new \PhpOffice\PhpWord\PhpWord();
$section = $phpWord->addSection();
$section->addText('Hello World!');
$file = 'HelloWorld.docx';
header("Content-Description: File Transfer");
header('Content-Disposition: attachment; filename="' . $file . '"');
header('Content-Type: application/vnd.openxmlformats-officedocument.wordprocessingml.
↪document');
header('Content-Transfer-Encoding: binary');
header('Cache-Control: must-revalidate, post-check=0, pre-check=0');
header('Expires: 0');
```

(continues on next page)

(continued from previous page)

```
$xmlWriter = \PhpOffice\PhpWord\IOFactory::createWriter($phpWord, 'Word2007');
$xmlWriter->save("php://output");
```

9.3 Create numbered headings

Define a numbering style and title styles, and match the two styles (with `pStyle` and `numStyle`) like below.

```
$phpWord->addNumberingStyle(
    'hNum',
    array('type' => 'multilevel', 'levels' => array(
        array('pStyle' => 'Heading1', 'format' => 'decimal', 'text' => '%1'),
        array('pStyle' => 'Heading2', 'format' => 'decimal', 'text' => '%1.%2'),
        array('pStyle' => 'Heading3', 'format' => 'decimal', 'text' => '%1.%2.%3'),
        )
    )
);
$phpWord->addTitleStyle(1, array('size' => 16), array('numStyle' => 'hNum', 'numLevel
↪' => 0));
$phpWord->addTitleStyle(2, array('size' => 14), array('numStyle' => 'hNum', 'numLevel
↪' => 1));
$phpWord->addTitleStyle(3, array('size' => 12), array('numStyle' => 'hNum', 'numLevel
↪' => 2));

$section->addTitle('Heading 1', 1);
$section->addTitle('Heading 2', 2);
$section->addTitle('Heading 3', 3);
```

9.4 Add a link within a title

Apply 'HeadingN' paragraph style to TextRun or Link. Sample code:

```
$phpWord = new \PhpOffice\PhpWord\PhpWord();
$phpWord->addTitleStyle(1, array('size' => 16, 'bold' => true));
$phpWord->addTitleStyle(2, array('size' => 14, 'bold' => true));
$phpWord->addFontStyle('Link', array('color' => '0000FF', 'underline' => 'single'));

$section = $phpWord->addSection();

// Textrun
$textrun = $section->addTextRun('Heading1');
$textrun->addText('The ');
$textrun->addLink('https://github.com/PHPOffice/PHPWord', 'PHPWord', 'Link');

// Link
$section->addLink('https://github.com/', 'GitHub', 'Link', 'Heading2');
```

9.5 Remove [Compatibility Mode] text in the MS Word title bar

Use the `Metadata\Compatibility\setOoxmlVersion(n)` method with `n` is the version of Office (14 = Office 2010, 15 = Office 2013).

```
$phpWord->getCompatibility()->setOoxmlVersion(15);
```

CHAPTER 10

Frequently asked questions

10.1 How contribute to PHPWord?

- Improve the documentation (Sphinx Format)

CHAPTER 11

Credits

CHAPTER 12

References

12.1 ISO/IEC 29500, Third edition, 2012-09-01

- Part 1: Fundamentals and Markup Language Reference
- Part 2: Open Packaging Conventions
- Part 3: Markup Compatibility and Extensibility
- Part 4: Transitional Migration Features

12.2 Formal specifications

- Oasis OpenDocument Standard Version 1.2
- Rich Text Format (RTF) Specification, version 1.9.1

12.3 Other resources

- DocumentFormat.OpenXml.Wordprocessing Namespace on MSDN

CHAPTER 13

Indices and tables

- genindex
- modindex
- search