Excel, Word and PowerPoint

The same page, loaded the same way, written as an Office file instead of a PDF. Three classes stand beside HtmlToPdf, with the same `Render`, `RenderFile` and `RenderUrl` and their `Async` twins: HtmlToExcel, HtmlToWord and HtmlToPowerPoint. Nothing else to install: the files are written by Weblitz itself, with no Office on the machine.

Since WeblitzPdf 2.0.0-beta.7; a workbook's pictures and a deck's tables since 2.0.0-beta.8. Tell us what your pages need.

Which one

The three are not three copies of the page. Each keeps something else of it:

- **Excel keeps the data.** The page's tables become sheets whose numbers are numbers.
- **Word keeps the content.** The page becomes a flowing document that someone can go on writing in.
- **PowerPoint keeps the look.** Each printed page becomes a slide that shows it as it prints, with its text as text.

When the page must look exactly as it prints and nobody will change it, a PDF is still the answer.

Excel: the page's tables

```
var result = HtmlToExcel.RenderUrl(new Uri("https://example.com/prices"));
File.WriteAllBytes("prices.xlsx", result.Workbook!);
```

Each table that holds data is a sheet, named after its caption, its `aria-label` or the heading before it:

- cells at their rows and columns, with their row and column spans as merged cells;
- numbers, percentages and amounts of money as typed cells, read in the page's language (`1,250.50` in English, `1.250,50` in German) and shown with the page's decimals, thousands separators and currency; ISO dates and `<time datetime>` as dates;

- what only looks like a number stays text: a code with leading zeros, a telephone number, a number of more than 15 digits, a date written a local way (03/04/2026 is two different days);
- header rows frozen at the top; fonts, colours, alignments and column widths as the page shows them; a cell that is one link keeps it.

A cell's pictures are over the cell, where the page shows them (a country's flag beside its name), with their alternative text. A table that only lays the page out is not a sheet: the tables inside it are. A page without any table gives one sheet of its text, a block a row.

Word: a document to work on

```
var result = HtmlToWord.Render(html);
File.WriteAllBytes("page.docx", result.Document!);
```

The page's content in reading order, as Word's own blocks:

- headings in Word's heading styles, so the navigation pane and a table of contents find them;
- paragraphs whose runs keep their font, size, weight, slant, colour, underline and links;
- bulleted and numbered lists as Word's lists;
- tables with their spans, fills, borders and a header row that repeats on each page;
- pictures at the size the page shows them, with their alternative text, SVG, AVIF and WebP ones too; preformatted text line by line.

The paper and its margins are the page options' (Format, Landscape, Margin, or the page's own `@page`).

It is not a picture of the page: positions, columns and backgrounds are not kept; a menu the page lays out in a row is one line, its items apart by tabs. Hide leaves out what the document does not need.

PowerPoint: the printed pages as slides

```
var result = HtmlToPowerPoint.RenderFile("report.html");
File.WriteAllBytes("report.pptx", result.Presentation!);   // result.SlideCount slides
```

The page is laid out and broken into pages as for a PDF, on paper the size of a slide, and each page is a slide:

- its text as text boxes, a paragraph a box with the page's line breaks, at the place, size, font, weight, slant and colour the page prints it with: it can be selected, searched and changed;
- its tables as PowerPoint tables: columns and rows as the page lays them out, each cell's text where the page prints it, numbers flush right, over the page's own borders and backgrounds;
- a link's text as a box of its own, whose click follows the link in a slide show (in a table, the text's own click);
- everything else (pictures, backgrounds, borders, charts) as one picture under the text.

Slides are PowerPoint's own 16:9 size unless SlideWidth and SlideHeight say otherwise; a page's own `@page` size is the slide's, as it is the paper's in a PDF. A page break in the page (`break-before: page`) starts a slide.

```
var options = new HtmlPowerPointOptions { SlideWidth = "10in", SlideHeight = "7.5in" };
// 4:3
```

The text is in the page's fonts: a reader who does not have one of them sees that text in another font, a little wider or narrower. Text the page turns, clips or draws half transparent stays in the slide's picture, as it prints.

The same options as a PDF

Each class has its own options, HtmlExcelOptions, HtmlWordOptions and HtmlPowerPointOptions. Their `Page` is how the page is loaded and laid out: the PDF's own HtmlPdfOptions, so everything the other guides say about scripts, the network, fonts, consent boxes and advertisements holds here.

```
var page = new HtmlPdfOptions { Network = HtmlNetwork.Any, Ads = HtmlAds.Block, Consent
= HtmlConsent.Hide };
var result = HtmlToWord.RenderUrl(new Uri("https://example.com/news"), new
HtmlWordOptions { Page = page });
```

What only a PDF has is not read: PDF/A, passwords, compression, the title and author (the file's title is the page's). The results carry the same notes, `Completed`, `Elapsed` and `HttpStatus` as a PDF's.

Without a licence key the three files are written in full and carry no trial watermark; a PDF and a picture carry one.