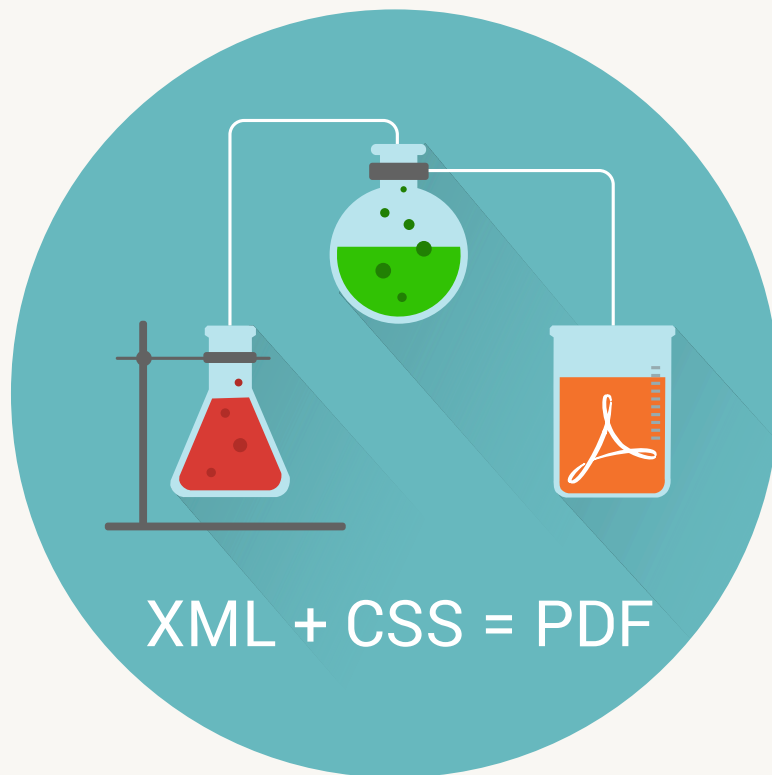


# Oxygen PDF Chemistry

## 21.1

### User Guide



# Contents

<b>Chapter 1. Getting Started.....</b>	<b>1</b>
Installing.....	2
Configuring Chemistry as an Oxygen External Tool.....	3
Command Line Interface.....	3
How to Obtain PDF Output from DITA Content using a Command Line.....	7
Environment Variables.....	8
Your First Document.....	9
Debugging the CSS.....	9
<b>Chapter 2. Styling for Print.....</b>	<b>13</b>
Associating a CSS to a Document.....	13
Page Formatting.....	13
The @Page Rule.....	14
Columns.....	19
Controlling Page Breaks.....	20
Forcing Page Breaks.....	21
Avoiding Page Breaks.....	21
Page Breaks Between Named Pages.....	21
Breaks Between Lines: Widows and Orphans.....	22
Chapter Page Placement and Styling.....	23
Headers and Footers.....	25
Extracting Text from Document Using String Sets.....	25
Extracting Text from Document Using XPath.....	27
Multiple Lines in Headers and Footers.....	28
The Page Counter.....	28
How to Use Flexible Layout in Headers and Footers.....	29
How to Style a Part of the Text from the Header.....	29
How to Use Dynamic Images in Headers and Footers.....	31
Footnotes.....	32
How to Add a Separator Above the Footnotes.....	33

Cross References.....	34
Internal links.....	34
External links.....	35
Creating a Table of Contents (TOC).....	35
Annotations.....	37
Comments and Tracked Changes - XML Fragment.....	38
Comments and Tracked Changes - HTML Fragment.....	41
PDF Output.....	43
Bookmarks.....	43
Metadata.....	45
Named Destinations (Anchors).....	47
Accessibility (508 Compliance).....	48
<b>Chapter 3. Element Layout.....</b>	<b>51</b>
<b>Chapter 4. Graphics.....</b>	<b>62</b>
<b>Chapter 5. Fonts.....</b>	<b>70</b>
Supported Fonts.....	70
Basic Fonts.....	70
Using Web Fonts.....	70
Using Installed Fonts.....	71
Using Local Font Files.....	71
Font Embedding.....	72
Using Simulated (Synthetic) Styles.....	76
<b>Chapter 6. Localization.....</b>	<b>79</b>
<b>Chapter 7. Advanced Styling.....</b>	<b>83</b>
<b>Chapter 8. Tutorials and How-To.....</b>	<b>88</b>
Example Tutorial.....	88
How To Apply Chemistry on Multiple Files.....	94
How To Insert the Current Date in the Cover Page.....	95
<b>Chapter 9. Appendix.....</b>	<b>97</b>
CSS Media Rules.....	97
CSS Selectors.....	97

CSS Functions.....	99
CSS Units.....	101
CSS Properties.....	101
Standard W3C CSS Properties.....	102
Extension CSS Properties.....	106
XML Support.....	110
Configuration File.....	111
Technical Issues.....	111
Index.....	a
Copyright.....	3

# 1.

## Getting Started

### What is Oxygen PDF Chemistry?

Oxygen PDF Chemistry allows you to obtain PDF output from HTML or XML documents simply by styling them with CSS. It is a *CSS Paged Media* processor based on the open-source Apache FOP XSL-FO engine. Its main purpose is to provide you with a simple tool that allows you to leverage your CSS knowledge to create printable deliverables. While the support for fancy formatting is limited, it is a great processor for generating technical documentation.

### Using CSS for Print


While CSS was designed as a language to style web pages, it turned out to be a good candidate to also format printed materials. The so called *CSS paged media module* was thus added to the existing standard by W3C.

New concepts added by this module include:

- Page size
- Page breaks
- Headers and footers
- Footnotes
- Links containing the target page number
- Leaders that are used to fill spaces

For more information, see the following paged media specifications:

- [CSS Paged Media Module](#)
- [CSS Generated Content for Paged Media Module](#)

 **Note:** There are many other processors that implement paged media standard: Prince XML, Antenna House, PDF Reactor.

Related Information:

- [Apache FOP Project](#)

- [Smashing Magazine: Designing For Print With CSS](#)

## Installing

Oxygen PDF Chemistry is a Java application and requires the following Oracle Java version, depending on your operating system:

- **Windows** or **Linux** - Java JRE version 1.8.
- **Mac OS X** - Java JDK version 1.8.

To install Oxygen PDF Chemistry on your platform, follow this procedure:

1. Download the distribution for your particular platform from the *Oxygen* website: [https://oxygenxml.com/pdf\\_chemistry/download.html](https://oxygenxml.com/pdf_chemistry/download.html).
2. Follow the instructions provided on that web page.
3. Add the full path for that folder to your PATH environment variable (for information about how to do this, see the related links below, depending on your operating system).
4. If you purchased the product, place the license key in the installation directory. The name of this file should be `licensekey.txt`. Note that unlicensed Oxygen PDF Chemistry produces watermarked output.
5. Test the installation by typing the following command in a console:

### Windows:

```
chemistry -v
```

### Mac OS X or Linux:

```
sh chemistry.sh -v
```

**Result:** It should display the version.

### Related Information:

- [How do I set system environment variables in Windows 10](#)

- [Setting PATH environmental variables in OSX permanently](#)
- [How to permanently set \\$PATH on Linux](#)

## Configuring Chemistry as an Oxygen External Tool


Oxygen PDF Chemistry can be run as an external tool, directly from the *Oxygen XML Editor/Author* environment.

To configure *Oxygen XML Editor/Author* to run Oxygen PDF Chemistry over the currently opened XML file, follow this procedure:

1. In *Oxygen XML Editor/Author*, select **Tools > External Tools > Configure**.
2. Press the **New** button to add a new external tool.
3. In the **Name** field, enter "chemistry".
4. In the **Command Line** field, enter the following command, depending on your operating system (make sure you replace `my_stylesheet.css` with the path to your CSS):

If you are using Oxygen PDF Chemistry distributed as a standalone product ([installed outside Oxygen \(on page 2\)](#)):

- **Windows:** `cmd /C chemistry.bat -in "${cf}" -css "my_stylesheet.css" -out "${cfd}/${cfn}.pdf" -show-pdf`
- **Mac OS X or Linux:** `sh chemistry.sh -in "${cf}" -css "my_stylesheet.css" -out "${cfd}/${cfn}.pdf" -show-pdf`

 **Note:** If you haven't configured the PATH environment variable to point to the installation directory, use the full path to the `chemistry` executable script. If the installation directory contains spaces, use quotes around the full path of the script.

If you are using Oxygen PDF Chemistry distributed within *Oxygen*:

- **Windows:** `cmd /C "${oxygenInstallDir}/oxygenChemistry.bat" -in "${cf}" -css "my_stylesheet.css" -out "${cfd}/${cfn}.pdf" -show-pdf`
- **Mac OS X or Linux:** `sh "${oxygenInstallDir}/oxygenChemistry.sh" -in "${cf}" -css "my_stylesheet.css" -out "${cfd}/${cfn}.pdf" -show-pdf`

## Command Line Interface

You can process an XML or HTML document from a command-line interface like this:

```
chemistry -in my_file.xml -out my_file.pdf
```

```
chemistry -in my_file.html -out my_file.pdf
```

You can specify one or more CSS files to be used. If the document is an HTML document, it may include references to other stylesheets or styles could be embedded using the `style` element. The command-line CSS files take precedence over files referenced from the document:

```
chemistry -in my_file.html -out my_file.pdf -css style1.css style2.css
```

### Required Parameters

#### **-in**

The input XML or HTML file in URI or File syntax.

```
-in http://my.example.com/my_file.html
```

```
-in C:\my.folder\my_file.xml
```

#### **-out**

The output PDF file in File syntax.

```
-out C:\publishing\my_file.pdf
```

### Optional Parameters

#### **-catalog-prefer**

Catalog preference mode. Possible values are: 'system' or 'public'. Default is 'public'.

#### **-catalogs**

The path to one or more XML catalogs, in URI or File syntax. You can separate the paths by using the ";" (semi-colon) character. You can use a catalog to solve DTD or Schema references to local files. If the list of catalog files is big, you may run into command-line size limitations. In this case, consider passing it as the `XML_CATALOG_FILES` ([on page 8](#)) environment variable.

#### **-css**

A list of CSS files, in URI or File syntax, separated by spaces.

```
-css http://my.example.com/css/pages.css http://my.example.com/css/fonts-
and-colors.css
```

If you want to process an HTML document, this parameter is optional. If you specify a CSS, it is considered to be more important than the ones referenced from the document (for example, using the `link` element or the `xml-stylesheet` processing instruction).

#### **-disable-xinclude**

A security setting that disables XML inclusions (XInclude). This is recommended when using Chemistry in a Web context. In addition, you should use a Java Security Manager to control the exact permissions granted to the processor.

#### **-disable-xxe**

A security setting that disables XML external entities. This is recommended when using Chemistry in a Web context. In addition, you should use a Java Security Manager to control the exact permissions granted to the processor.

#### **-dump-fo**

Dumps the FO file (before converting it to PDF) in the same location as the output file. This can be used for debugging purposes.

#### **-dump-styled-content**

Dumps the intermediate, annotated XML file in the same location as the output file. This can be used for debugging purposes.

#### **-enable-latin-ligatures**

Used to enable ligatures between some of the characters from the Latin glyph range. The character sequences that might be combined is highly dependent on the font, but usually they are: "fi", "fl", "tt".

**!** **Warning:** The combined characters might not be searchable in PDF readers (for instance, the sequence "fi" from the word "file" can be replaced by a single character and the word will not be found when searching). Use this only when you intend to print on physical media, (such as paper). Ligatures for non-Latin scripts, (such as Arabic) are enabled by default, and do not cause search problems.

#### **-fonts-dir**

The directory with additional fonts. The TTF files should be placed directly in it (no subdirectory).

#### **-generate-named-destinations**

Controls whether or not the `id` attributes from the input document are used to generate PDF named destinations (anchors). Accepted values are `yes` (default) or `no`. For more informations, see [Named Destinations \(Anchors\) \(on page 47\)](#).

**-help**

Prints the usage information.

**-http-proxy-host**

The HTTP proxy host.

**-http-proxy-port**

The HTTP proxy port.

```
-http-proxy-host my.proxy.server -http-proxy-port 3128
```

**-image-resolution**

The resolution (in DPI) of the raster images (an integer), for images that do not provide this meta information. For changing the resolution using CSS, see: [Image Resolution \(on page 65\)](#).

```
-image-resolution 72
```

**-licensekey-dir**

Specifies the directory where the license key file is located. The license key file should have the name `licensekey.txt`.

**-log4j-file**

Full path to a [log4j configuration file](#). Example of configuration file:

```
log4j.rootCategory=info, R

log4j.category.com.oxygenxml.chemistry.stages=info
log4j.category.org.apache.fop.fonts.autodetect.FontFileFinder=info
log4j.category.org.apache.fop.apps.FopConfParser=error

# R is the standard output
log4j.appender.R=org.apache.log4j.ConsoleAppender
log4j.appender.R.layout=org.apache.log4j.PatternLayout
log4j.appender.R.layout.ConversionPattern=%r %p [ %t ] %c - %m%n
```

**-no-aggressive-hyphenation**

Oxygen PDF Chemistry normally uses an aggressive technique to create hyphenation points at underscores, dots, and case changes. To disable this functionality, use this `-no-aggressive-hyphenation` parameter.

**-no-network**

Denies access to all your network connections. If your XML document or CSS files reference remote resources, the transformation will fail.

#### **-no-rtl-mirroring**

Disables switching of left and right margins, padding and borders for right-to-left content. To make stylesheet development easier, the left margin automatically becomes the right margin when the paragraph has an RTL orientation.

#### **-pdf-ua**

Produces output that conforms to PDF/UA-1 accessibility standards. Conversion will break if fonts are not embedded. See: [Accessibility \(508 Compliance\) \(on page 48\)](#).

#### **-show.changes.and.comments.as.pdf.sticky.notes**

When set to **yes** (default), the review elements are shown as PDF annotations. When set to **no**, the review elements are left in the document object model and can be styled using CSS rules.

#### **-show-pdf**

Opens the created PDF file in the default application (Windows only).


#### **-v**


Shows the version of the processor.

#### **-Xmx<NNN>m**

Used to specify the maximum amount of memory that is available to the Oxygen PDF Chemistry process. For example, to allow the process to use 1GB of memory:

```
-Xmx1024m
```

 **Note:** The default is 512MB.

 **Note:** Larger memory settings (beyond 1GB) are permitted only when the Java Virtual Machine that runs Oxygen PDF Chemistry is 64 bits and there is enough physical memory accessible to the operating system.

 **Note:** If the `JAVA_ARG_LINE` environment variable is set, this parameter is ignored.

## How to Obtain PDF Output from DITA Content using a Command Line

If you are using Oxygen PDF Chemistry distributed as a standalone product ([installed outside Oxygen \(on page 2\)](#)) and you want to obtain PDF output from DITA content, you need to install an

additional plugin (**dita-ot-css-pdf**) so that all the references are resolved and expanded properly. This plugin makes it possible to use the Oxygen PDF Chemistry processor to obtain DITA to PDF output by running the DITA OT from a command line interface.

To process and publish DITA to PDF using CSS and Oxygen PDF Chemistry, follow this procedure:

1. Install the Chemistry processor (*on page 2*).
2. Add a reference to Chemistry's parent folder in your system's PATH environment variable.
3. Download and install the DITA Open Toolkit (<http://www.dita-ot.org/download>).
4. Download the **dita-ot-css-pdf** plugin that is available on GitHub (<https://github.com/oxygenxml/dita-ot-css-pdf>).
5. Install the plugin in the DITA Open Toolkit (<http://www.dita-ot.org/dev/topics/plugins-installing.html>).
6. Run the DITA Open Toolkit from a command line and make sure you specify the format to be **pdf-css**, the processor type to be **chemistry**, and the path to your main DITA map.

For example:

```
dita-ot-dir/bin/dita --input=path/to/my.ditamap --format=pdf-css -
Dcss.processor.type=chemistry --output=path/to/output/folder
```

## Environment Variables


### JAVA\_HOME

Normally it is used by the `java` executable that is found in the `PATH` environment variable. By using this variable, you can specify a different Java Virtual Machine installation directory.

### JAVA\_ARG\_LINE

You can specify a set of arguments for the Java Virtual Machine. The following example specifies the maximum and initial memory setting to be 300MB:

```
SET JAVA_ARG_LINE=-Xmx300m -Xms300m
```

 **Note:** Setting this environment variable disables the usage of the `-Xmx` (*on page 7*) parameter.

### XML\_CATALOG\_FILES

The path to one or more XML catalogs, in URI or File syntax. You can separate the catalogs by using the ":" (colon) or ";" (semi-colon) symbols.

```
SET XML_CATALOG_FILES="file:/D:/catalogs/docbook.xml;file:/D:/catalogs/  
other.xml;"
```

 **Note:** You can also use the `-catalog-prefer` (on page 4) command-line parameter together with this environment variable.

## Your First Document

Make sure you followed the procedure in the [Installing](#) (on page 2) section.

The following procedure provides a simple example for using Oxygen PDF Chemistry to style an XML document:

1. Start with a simple XML document to test the installation. Save it as `my_doc.xml`.

```
<article>  
  <title>Hello World!</title>  
  <p>My first document.</p>  
</article>
```

2. Create a CSS stylesheet in the same folder. Save it as `my_style.css`.

```
article, title, p {  
  display: block  
}  
  
title {  
  font-size: larger;  
  border-bottom: 1pt solid blue;  
}
```

3. Open a command-line console, change directory to the folder that contains your samples, and invoke:

```
chemistry -in my_doc.xml -css my_style.css -out my_doc.pdf
```

4. Open the result in your PDF reader.

## Debugging the CSS



If you notice that some of the CSS properties were not applied as expected, some of the tips offered in this topic might help you with the debugging process.

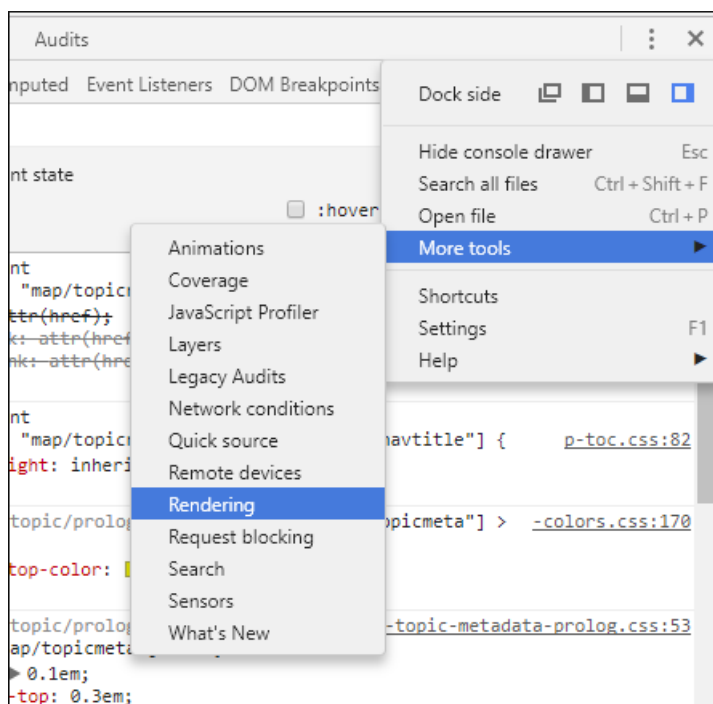
### Comparing the CSS Against the Input File

The first thing that you should try is to check the file structure of the input XML or HTML file and compare it to the CSS selectors to make sure they are written correctly against the document structure. If you still cannot identify the problem, then inspect how the styles are applied (you can try any of the methods listed below).

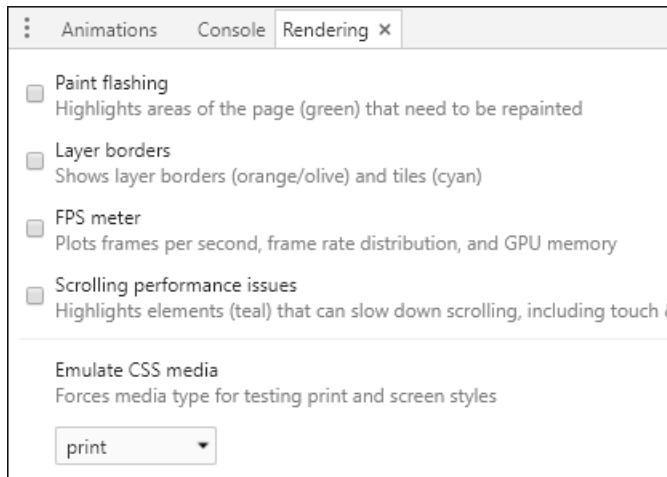
### Inspecting the Applied Styles Using the Chrome Browser

To inspect the applied CSS styles using Chrome:

1. In Chrome, open the input XML or HTML file.
2. Click on the element you want to inspect.
3. Activate the **Chrome Developer Tools** by using  > **More Tools > Developer Tools**, or press **CTRL +SHIFT+I**.
4. Activate the **Rendering** pane by using  > **More Tools > Rendering**:



5. In the **Rendering** pane, select **print** from the **Emulate CSS media** section. This will activate the CSS selectors enclosed in `@media print {..}`.



**Note:** This allows you to debug the styling of elements, table of contents, and index, but not the styles of the page margin boxes (headers, footers) or page breaks.

**Tip:** In the left pane of the **Developer Tools** interface, you can inspect elements and their styles in the **Elements** tab. You can click on any of the links to display the applied CSS files in the **Styles** tab in the right pane. Editing the styles in that pane results in a live preview of how the change will affect the output.

## Inspecting the Applied Styles Using Oxygen XML Editor/Author

To inspect styles:

1. In **Oxygen XML Editor/Author**, open the input XML or HTML file.
2. **[Optional]** From the **Styles** toolbar, choose the **+ Print Ready** entry. This will activate certain CSS selectors enclosed in `@media print {..}`.
3. Click on the element you want to style. Use the **Inspect Styles** action from the Contextual Menu. A specialized **CSS Inspector** view will show the built-in CSS rules.

**Tip:** With this file opened in **Author** mode, it might be helpful to switch the **Tags Display Mode** to **Full Tags with Attributes**. You might be able to identify the selector you need to style without using the **CSS Inspector** view.

**Note:** This allows you to debug styling of elements, but not of the page margin boxes (headers, footers) or page breaks.

**CAUTION:** Do not modify the built-in rules directly in the CSS files from the **Oxygen XML Editor/Author** installation. Instead, copy the rules to your own customization CSS.

## Other techniques

These are some other techniques you may find useful:

- Add background and borders properties to the specific CSS rule. If they do not appear in the output then there is a problem with the rule selector.
- Try to use the `!important` notation to the property that is not applied, or make the selector more specific (you can add more parent selectors).
- To figure out how the elements are mapped into PDF, you can use this fragment in the customization CSS:

```
* {  
  border: 1pt solid blue !important;  
}  
  
*:before(1000) {  
  content: oxy_name() !important;  
  color: orange;  
}  
  
*:before(999) {  
  content: "[ class= " attr(class) "]" !important;  
  color: orange;  
}
```

This will show the element name, its class attribute, and will paint a blue border around each of the elements in the output. It will not show the page margin boxes or some content elements that were hidden.

# 2.

## Styling for Print

This section contains information about styling various components in printed output.

### Associating a CSS to a Document

There are several ways to associate the CSS to your document:

- Specify it as a PI at the beginning of the XML document:


```
<?xml-stylesheet type="text/css" href="my-style.css" ?>
```

- Specify it in the meta section of the HTML document:

```
<link rel="stylesheet" type="text/css" href="my-style.css" />
```

- Specify it from the command line using the `-css` argument:

```
chemistry -in my-doc.xml -css my-style.css -out my-doc.pdf
```

 **Note:** The rules from the CSS file indicated using the `-css` command line argument take precedence over the ones from CSS files referred from the input document.

Related Information:

- [Command Line Interface \(on page 3\)](#)
- [Mozilla MDN Web Docs: HTML Element Reference](#)

### Page Formatting

This section contains information about how you can use the new concepts from *CSS media paged module*, such as `@page` rules, page margin boxes, and more.

## The @Page Rule

### Setting the Page Size

If you do not set the size, Oxygen PDF Chemistry uses portrait **US-LETTER** with margins of **1in**.

This is a basic choice you will have to make for your output. You can use the `@page` rule like this:

```
@page {  
  size: us-letter;  
}
```

The page sizes can be selected by name, for example, **A4**, **A3**, **US-LEGAL** (these are case insensitive), or if you are planning to print on a non-standard sheet, you can specify the width and height:

```
@page {  
  size: 5in 7in;  
}
```

You can also specify the page orientation using the `portrait` or `landscape` keywords.

```
@page {  
  size: A3 landscape;  
}
```

You can use the `@page` rule multiple times and the properties will merge as in ordinary CSS element styling rules.

### Using Page Selectors to Style the Blank, Left, Right, First, and Last Pages

There are cases where you need to have different page settings depending on the position of the page in the printed material.

```
@page :left {  
  border-right: 0.5in solid yellow;  
}  
  
@page :right {  
  border-left: 0.5in solid yellow;  
}
```

The `:first` and `:last` selectors are used to style the first and last page from a sequence of pages.

```
@page :first,:last {
  border-top: 5pt solid yellow;
}
```

The `:blank` selector is used to style the blank pages that appear as a result of [forced page breaks](#).

## Using Named Pages

In the examples above, the default page settings were changed.

Suppose that you need to put a particular `table` element on a landscape page. The example below defines a named page `table-landscape-page` that switches the orientation to landscape. Then, to maximize the space available to the table, the margin is reduced for this page. Also, the A4 size defined by the default `@page` rule is inherited by the named page.

```
@page {
  size: A4;
  margin: 1in;
}

@page table-landscape-page {
  size: landscape;
  margin: 0.5in;
}

table {
  page: table-landscape-page;
}
```

Also, Oxygen PDF Chemistry allows you to break a sequence of elements that have the same page in different page groups. A simple example is a set of chapter elements where you need to apply specific formatting for the first page from each chapter.

```
<chapter/>...</chapter><chapter/>...</chapter><chapter/>...</chapter>
```

The CSS would be:

```
@chapter {
  ...
}

@chapter:first {
```

```

    background-color:yellow;
}

chapter {
    page: chapter;
}

```

But this is not enough. According to the W3C specification, all chapters that have the same page will be merged in a single page group (sequence), and the first selector will apply only on the first page from the first chapter. Therefore, you need to use the `-oxy-page-group` property on each of the chapter elements.

```

@chapter {
    ...
}

@chapter:first {
    background-color:yellow;
}

chapter {
    -oxy-page-group:start;
    page: chapter;
}

```

The accepted values for the `-oxy-page-group` property are:

- **start** - Forces the creation of a new page group (sequence) even if the page before the element has the same name. The W3C specification algorithm (<https://www.w3.org/TR/css3-page/#using-named-pages>) would merge the current element with the opened page sequence.
- **auto** - Uses the W3C algorithm normally.

## Setting Page Margins

The same margin model applies to the page as for any element. The content is surrounded by padding, then a border, then margins. The page margins are between the physical end of the page and the text (if no border or padding is defined).

For example, to create a default page with margins of two inches:

```

@page {
    margin: 2in;
}

```

```
}

```

Or you can specify them for all the sides:

```
@page {
  margin-left: 2in;
  margin-right: 1in;
  margin-top: 0.5in;
}
```

Or, as a common case, you may want that the inner margins of the pages of a book to be larger:

```
@page {
  margin: 0.5in;
}

@page :left {
  margin-right: 1in;
}

@page :right {
  margin-left: 1in;
}
```

If no margins are specified in the CSS, the page margins are set at **1in**.


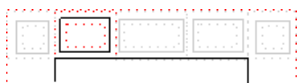

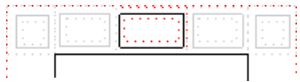
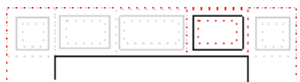
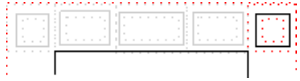
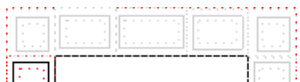
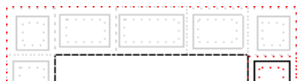
## Page Margin Boxes

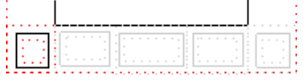
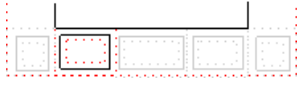

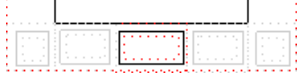
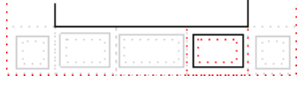
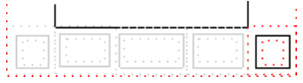
The CSS specification defines numerous rectangular areas placed in the margins that surround the content of the page. These are called *margin boxes* and may be used to display static CSS generated content such as page numbers, publication titles, or other artwork.

```
@top-left {
  content: url('company-logo.png');
  background-color:red;
}
```

The following table shows each of the 16 margin boxes (taken from the CSS specification: <https://www.w3.org/TR/css3-page/#margin-boxes>):

**Table 1. Page-Margin Box Definitions**

Box	Description	Placement
top-left-corner	a fixed-size box defined by the intersection of the top and left margins of the page box	
top-left	a variable-width box filling the top page margin between the top-left-corner and top-center page-margin boxes	
top-center	a variable-width box centered horizontally between the page's left and right border edges and filling the page top margin between the top-left and top-right page-margin boxes   <b>Note:</b> Using a <code>top-center</code> box together with a <code>top-left</code> or <code>top-right</code> will create a sequence of three areas of identical width, with the <code>top-center</code> box occupying the center.	
top-right	a variable-width box filling the top page margin between the top-center and top-right-corner page-margin boxes	
top-right-corner	a fixed-size box defined by the intersection of the top and right margins of the page box	
left-top	a variable-height box filling the left page margin between the top-left-corner and left-middle page-margin boxes	
left-middle	a variable-height box centered vertically between the page's top and bottom border edges and filling the left page margin between the left-top and left-bottom page-margin boxes	
left-bottom	a variable-height box filling the left page margin between the left-middle and bottom-left-corner page-margin boxes	
right-top	a variable-height box filling the right page margin between the top-right-corner and right-middle page-margin boxes	
right-middle	a variable-height box centered vertically between the page's top and bottom border edges and filling the right page margin between the right-top and right-bottom page-margin boxes	

Box	Description	Placement
right-bottom	a variable-height box filling the right page margin between the right-middle and bottom-right-corner page-margin boxes	
bottom-left-corner	a fixed-size box defined by the intersection of the bottom and left margins of the page box	
bottom-left	a variable-width box filling the bottom page margin between the bottom-left-corner and bottom-center page-margin boxes	
bottom-center	a variable-width box centered horizontally between the page's left and right border edges and filling the bottom page margin between the bottom-left and bottom-right page-margin boxes   <b>Note:</b> Using a <code>bottom-center</code> box together with a <code>bottom-left</code> or <code>bottom-right</code> will create a sequence of three areas of identical width, with the <code>bottom-center</code> box occupying the center.	
bottom-right	a variable-width box filling the bottom page margin between the bottom-center and bottom-right-corner page-margin boxes	
bottom-right-corner	a fixed-size box defined by the intersection of the bottom and right margins of the page box	

## Columns

If you need to spread content in multiple columns, use the two CSS properties: `column-count` and `column-gap`.

Suppose that you have an HTML section that will be shown in a two-column layout, and as a special constraint, you want the title to span on both columns.

```
<section class='two-columns'>
  <h2>The section on two columns</h2>
  <p> The section content... </p>
</section>
```

You can define a page with two columns:

```
@page two-columns-page {
  column-count: 2;
  column-gap: 1in;
```


```
}
```

Then associate the section element with this page:

```
section.two-columns {
  page: two-columns-page;
}
```

To make the title span the entire page width, use the `column-span` property:

```
section.two-columns h2 {
  column-span: all;
}
```

 **Limitation:** You cannot use multiple column configurations on the same page. Oxygen PDF Chemistry only takes the `column-count` and `column-gap` properties into account if they are set on `@page` rules, not on elements from the content.

To control exactly the column breaking algorithm you can use the following extension properties:

`-oxy-column-break-inside` ([on page 110](#)) `-oxy-column-break-before` ([on page 110](#)) `-oxy-column-break-after` ([on page 110](#)).

For example, to eliminate the possibility where a heading `<h3>` element remains at the end of a column and the text that follows it moves to the next (a column break just after the heading) you can use:

```
section.two-columns h3 {
  -oxy-column-break-after: avoid;
}
```

## Controlling Page Breaks

If you want to start a new page with each chapter title, or if you want to avoid breaking a table onto two pages or avoid a break after a section title, you can use CSS properties to control the breaks.

CSS offers the following properties to control the page breaking process:

- `page-break-before`
- `page-break-after`
- `page-break-inside`

## Forcing Page Breaks

To always move the content to the beginning of the page (for example, before the title of a section), you can use:

```
section h1 {
  page-break-before: always;
}
```

Or if you only want a specific element to appear on a single page:

```
div.notice {
  page-break-before: always;
  page-break-after: always;
}
```

## Avoiding Page Breaks

Sometimes page breaks should be avoided. For instance, to avoid breaking between a title and the subsequent content:

```
h1, h2, h3, h4, h5, h6 {
  page-break-after: avoid;
}
```

Or if you want to avoid breaks inside tables and lists:

```
table, ol, ul {
  page-break-inside: avoid;
}
```

## Page Breaks Between Named Pages

A page break will be created each time there is a change in the `page` property associated to the elements in a sequence. Suppose that you have a sequence of `div` elements, one of them associated with a "cover" page, and others with a "chapter" page:

```
div.cover {
  page: cover;
}

div.chapter {
```

```

    page: chapter;
}

```

The XML document:

```

...
<div class="cover"> Welcome to the User Guide... </div>
<div class="chapter"> Here are the details... </div>
<div class="chapter"> Here are some more details... </div>

```

In this example, there will be a forced page break between the first `div` (associated to the "cover" page) and the second because of the `page` change.

The next two `div` elements are not separated by page breaks because they have the same `page` name ("chapter") and they are grouped in the same page sequence. If you want to style the first page from that sequence in a different way, the selector:

```

@page content:first{
    background-color: yellow;
}

```

will apply to the first page from the first "chapter" `div`.

Suppose you want each of the chapters to start a new page sequence, with the first page colored in yellow. To do so, you must declare a new sequence start on the `div` element. This can be done using the extension property `-oxy-page-group`:

```

div.chapter {
    page: chapter;
    -oxy-page-group:start;
}

```

Related Information:

- <https://www.w3.org/TR/css3-page/#using-named-pages>

## Breaks Between Lines: Widows and Orphans

There are cases where the page break is placed between paragraph lines. For aesthetic purposes, it is required that at least several lines remain in the first (for example, the paragraph is not left with just a single line). Another constraint might be the number of lines that are moved to the next page. Again, it should avoid leaving just a single line. [CSS defines two properties](#) for this type of control:

1. The `orphans` property specifies the minimum number of line boxes that should be left in a paragraph before the page break occurs.
2. The `widows` property specifies the minimum number of line boxes of a block container that must be left in a paragraph created on the next page after a break.

The following example shows how to keep the paragraphs at least four lines on the page before the break, and two lines on the page following the break:

```
p {
  orphans: 4;
  widows: 2;
}
```

## Chapter Page Placement and Styling

Oxygen PDF Chemistry provides specialized support to help you customize how chapters are handled in the output.

### How to Start Chapters on an Odd or Even Page

A common use case is to arrange the chapters of the publication to start on an odd page number so that in the printed output, the chapter starts on the right side of the book.

By default, pages that have the same name are merged together in a single page sequence. If you want each of the chapters to have its own sequence, you can use the `-oxy-page-group:start` CSS property to create separate page sequences. This allows you to control the start page of each of the chapters.

```
.chapter {
  page: chapter-page;
  -oxy-page-group: start;
}
```

To specify which page the chapter should start on, you can use another CSS property, the `-oxy-initial-page-number` [property \(on page 107\)](#):

```
@page chapter-page {
  -oxy-initial-page-number: auto-odd;
}
```

Supported values for `-oxy-initial-page-number` are: **auto**, **auto-even**, **auto-odd**, **<number>**.

Moving the chapter on a specific page number may create blank padding pages at the end of the previous page sequence. If you want to style those blank pages, use the `:blank` page selector:

```
@page :blank{
  @top-center{content:"Intentionally left blank"};
}
```

## How to End Chapters on an Odd or Even Page

Another use case is to specify how a page sequence should end. If the sequence should have a total number of odd or even pages (or if it should end on an even or odd page).

Suppose you have a table of contents that follows the cover page and you need to want to end up with an even number of pages.

```
<div class='toc'>Table of Contents..
```

```
.toc {
  page: toc-page;
}
```

Now you can use the `-oxy-force-page-count` property (*on page 107*) with an **even** value:

```
@page toc-page {
  -oxy-force-page-count: even;
}
```

Supported values for `-oxy-force-page-count` are: **even, odd, end-on-even, end-on-odd, auto, no-force**.

## How to Style the First Page of a Chapter

You can use the `:first` page rule selector to control how the first page of a chapter will look. For example, suppose you do not want the header to appear on the chapter first page. Your CSS might look like this:

```
@page {
  @top-right-corner{content: counter(page);}
  @top-left{content: 'My Publication';}
}

.chapter {
  page: chapter-page;
```

```

-oxy-page-group: start;
}

@page chapter-page:first {
  @top-right-corner{content: none;}
  @top-left{content: none;}
}

```

## Headers and Footers

Most printed publications use page headers and footers for almost all of their pages. To define them, you will use the page margin boxes.

```

@page :first {
  @top-center {
    content: "How to Grow Flowers";
    font-size: smaller;
    color: gray;
  }
}

```

You can set content and style to multiple page margin boxes. You should place them in the same parent `@page` rule. In the above example, the text is displayed only in the `top-center` of the first page, due to the `:first` page selector.

## Extracting Text from Document Using String Sets

To display the title of the publication or the title of the current chapter, you will need to extract some content from the document and use it in one or more page margin boxes. This is possible by using a `string-set` property. It is similar to a variable that is initialized to a content each time a specific element is matched.

In the following example, the text content of the `H1` element is extracted and used as a publication title and the `H2` element defines the chapter title:

```

h1 {
  string-set: publication_title content();
}

h2 {
  string-set: chapter_title content();
}

```

}

**⚠ Important:** To define multiple string sets for an element, use a **single** `string-set` **property** with a list of comma-separated definitions:

```
h1 {
  string-set: publication_title content(), publication_author attr(data-author);
}
```

The following example uses the collected strings in the top margins of the pages. It joins the publication title and the chapter title by a "/" character, then places them in the outer side of the pages (to the left for the left-side pages, to the right for the right-side pages).

```
@page :left {
  @top-left {
    content: string(publication_title) " / " string(chapter_title);
  }
}

@page :right {
  @top-right {
    content: string(publication_title) " / " string(chapter_title);
  }
}
```

A string set may contain static text, content from the document, attributes from the element, or counters. This is a more complex example, where a chapter number is added to the `chapter_title` string set:

```
h1 {
  counter-reset: chapter;
}

h2 {
  string-set: chapter_title "Chapter (" counter(chapter) ")" content();
  counter-increment: chapter;
}
```

#### Related Information:


- [How to Use Dynamic Images in Headers and Footers \(on page 31\)](#)


## Extracting Text from Document Using XPath

You can use the `oxy_xpath` CSS extension function to execute an XPath expression over the input document. The main advantage is that you can execute any XPath processing, including:

- Document data retrieval
- Mathematical calculations
- If/then/else conditions

You can use `oxy_xpath` in values of all properties defined in a page rule.

 **Important:** This technique is not standard and is guaranteed to work only with this processor.

 **Note:** The XPath expression from the page rules is evaluated in the context of the document root element, so you will need to use absolute expressions starting with `/` or `//`. This is different from the case when the `oxy_xpath` is used in CSS rules that match an element. In this case, the XPath expressions are evaluated in the context of the matched element and you can use relative paths.

 **Tip:** XPath 2.0 is supported (not schema aware).

Suppose your document defines a creation date in a metadata section. This section may be anywhere in your document. To place the creation date in the center of the publication header:

```
@page {
  @top-center {
    content: "Created: " oxy_xpath("//div[@class='created']/text()");
  }
}
```

Another example is to use an image from the document in the publication header:

```
@page {
  @top-center {
    content: url(oxy_xpath('//img[@class='product-img']/@href'));
  }
}
```

If the URL returned by `oxy_xpath` is not absolute, it is considered to be relative to the CSS file. To obtain a URL relative to the XML document, you can use in the XPath expression functions, such as `resolve-uri` and `document-uri`:

```
@page {
  @top-center {
```

```

        content: url(oxy_xpath(resolve-uri(//img[@class='product-img']/@href, document-
uri())));
    }
}

```

## Multiple Lines in Headers and Footers

Sometimes you need to format the text from the header (or any page margin box) on two or more lines.

For example, suppose you want to have the following notice in the footer:

```

Confidential Document.
Do not distribute it without written consent!

```

The solution is to use `\a` in the static content from your CSS. This is an escape representing the line feed character in *ISO-10646 (U+000A)*. This character represents the generic notion of "newline" in CSS.

```

@page {
    @bottom-center {
        content: "Confidential Document. \a Do not distribute it without written consent!";
    }
}

```

## The Page Counter

Besides the CSS counters that can be set on elements (for numbering sections, lists, tables, etc.), the CSS *paged media module* defines two more counters:

### **page**

This counter returns the number of the current page.

### **pages**

The number of total pages from the publication.

These counters are automatically updated by the publishing processor and can be used from the page margin boxes.

```

@page {
    @bottom-center {
        content: "Page: " counter(page);
    }
}

```


```
}
```

Or if you need to obtain "Page 4 of 100", you simply use:

```
content: "Page: " counter(page) " of " counter(pages);
```

You can format the page counter with styles such as `decimal`, `roman`, `lower-roman`:

```
@page table-of-contents {
  @top-right {
    content: "Contents | " counter(page, lower-roman);
  }
}
```

 **Note:** Using different counter styles under the same page name (for instance, using `lower-roman` for the left page and `decimal` on the right page) is not supported.

## How to Use Flexible Layout in Headers and Footers

In typical usage, the sub-regions of the header (the page margin boxes `@top-left`, `@top-center`, `@top-right`) and footer (`@bottom-left`, `@bottom-center`, `@bottom-right`) are distinct regions. If you are specifying content for all of them, the content set in one does not use space from the others. Instead, it wraps lines in its own region.

```
@page {
  @top-left { content: "A very long publication title..."}
  @top-center { content: "The long organization name..."}
  @top-right { content: counter(page)}
}
```

This creates a sort of table with fixed equal-sized columns, with the text wrapping inside of them.

You may need to 'merge' the center and right sub-regions for the header so that the layout engine has more room for topic titles before it wraps the title to a new line.

The solution is to eliminate the center part from the header and footer (`@top-center` and `@bottom-center`), and move the content to one of the sides:

```
@page {
  @top-left { content: content: "A very long publication title..."}
  @top-right { content: "The long organization name..." " " counter(page)}
}
```

## How to Style a Part of the Text from the Header


If you need to style a fragment of text (for example, a company slogan) with certain colors or font styles, you have several options:


- Use an SVG image as background for a page margin box or for the entire page.
- Use the `oxy_label` constructor. This is a function that creates a text label with a set of styles.

```
@page {
  @top-right {
    content: oxy_label(text, "My Company", styles, "color:red; font-size:
larger;")
    ' '
    oxy_label(text, "Product", styles, "color:blue; text-
decoration:underline;"));
  }
}
```

You can combine the `oxy_label` with `oxy_xpath`, to extract and style a piece of text from the document:

```
content: oxy_label(text, oxy_xpath("/some/xpath"), styles, "color:blue; ");
```

 **Note:** These functions work only with the Chemistry CSS processor.

 **Note:** You cannot use `string()` inside an `oxy_label()`. As a workaround, to apply styling on the dynamic text retrieved by a `string()` function you can define some overall styles for the entire page margin box and then use the `oxy_label` to style differently the static text.

```
@page {
  @top-right {
    color: red;
    content: oxy_label(text, "My Company", styles, "color:black")
    ' '
    string(chaptertitle); /* This inherits the styling from @top-
right*/
  }
}
```

- Use two adjacent page margin boxes, and style them differently:

```
@page {
  @top-center {
    content: "First part";
    color: red;
```

```

    text-align:right;
}
@top-left {
    content: "- Second part";
    color: blue;
    text-align:left;
}
}

```

## How to Use Dynamic Images in Headers and Footers

It is possible to dynamically change the images from the publication header or footer depending on the section.

For example, you want to place an image that describes the current section. Assuming that the titles of the sections define some metadata attribute pointing to some image file and the image references should be absolute URIs or relative to the input document.

```

<h2 -data-header-image="installation.png">
...
<h2 -data-header-image="configuring.png">
...

```

The CSS should define a `string-set` that extracts the attribute value and builds an image for it using the `attr(.., url)` or `url` function.

**⚠ Important:** The `attr(name, url)` function resolves the reference relative to the input XML document URI. The `url(attr(name))` resolves the reference relative to the CSS file URI.

```

@page {
    margin: 1in;
    @top-right-corner {
        content: string(str);
        font-size:0; /* Get rid of ascent and descent to avoid spaces around the image */
    }
}

h2 {
    string-set: str attr(-data-header-image, url); /* references relative to the input
    document */
}

```

Besides the `attr` function, you can add text:

```
...
    string-set: str "Section... " attr(-data-header-image, url);
...
```

## Footnotes

Footnotes are especially important in books with a lot of references and annotations. To mark an element as being a footnote, you should use the `float: footnote` property.

Suppose that you have the following document fragment:

```
<p>Changing the oil <span class='fnote'>This should be done
by a specialist in a controlled environment</span> in your car.</p>
```

To mark the `span` with the class `footnote` as an actual one:

```
span.fnote {
    float: footnote;
}
```


There is a counter named `footnote` that is incremented automatically by the formatting processor each time it encounters an element with `float: footnote` set on it. Sometimes it makes sense to reset this counter at each of the chapters or sections of the document.

```
section {
    counter-reset: footnote 1;
}
```

### Styling the Footnote Call

The number that remains in the content is called footnote call. To style it, use the `footnote-call` pseudo element:

```
span.fnote:footnote-call{
    color: red;
}
```


 **Note:** By default, Oxygen PDF Chemistry considers all the `:footnote-call` pseudo elements to have the content to be the value of the footnote counter, a smaller font size, and to be aligned at top of the line. You can change these properties if you need something different:

```
:footnote-call {
  content: counter(footnote);
  vertical-align: super;
  font-size: 0.8em;
}
```

## Styling the Footnote Marker

The footnote marker is the number that is shown at the left side of the footnote text, in the lower part of the page. It has the same text as the footnote call. To style it, you can use the `footnote-marker` pseudo element:

```
span.fnote:footnote-marker {
  font-weight: bold;
}
```

 **Note:** Oxygen PDF Chemistry considers all the `:footnote-marker` pseudo elements to have the content to be the value of the footnote counter, followed by a dot, and to be aligned to the right, towards the footnote text. You can change these properties if you need something different:

```
:footnote-marker {
  content: counter(footnote) '.';
  list-style-position: outside;
  text-align: right;
}
```

## How to Add a Separator Above the Footnotes

The `@footnote` part of a `@page` declaration controls the style of the separator between the page content and the footnotes. For the content, you should set a `leader`. The leader uses a letter or a line style to fill the entire width of the page.

```
@page {
  margin: 0.5in;
  ....
  @footnote {
    content: leader(solid);
    color: silver;
  }
}
```

To create a dotted line, you can use the dot character: `leader(' . ')`. Other commonly used characters are: `"-"` (dash) and `"_"` (underscore).

## Cross References

Technical documentation excels in cross references between sections and links to external resources. The end user must be able to follow these links both in printed form, and in on-screen PDF rendering software.

### Internal links

For printed material, a cross reference cannot just be a simple link (although PDF renderers support them for on-screen display). It should also display the page number of the target. In CSS you can do this by using the `target-counter` function.

For example, to get:

```
For details see [Installation on page 34].
```

from:

```
<p> For details see <a href="#installation">Installation</a>.</p>
```

you can use a static content that is shown after the text from the link, consisting of a fixed string " on page " and the number of the page of the element referred by the `href` attribute:

```
a:after {
  content: " on page " target-counter(attr(href), page);
}
```

The `target-counter` function may be used together with the `leader` function to create table of contents. See: [Creating a Table of Contents \(TOC\) \(on page 35\)](#).

The processor supports both `target-counter` and `target-counters` functions, on page or other counters associated to your document elements. For example, you can use the `target-counter` to fetch the number of the chapter that contains the target:

```
<div class="chapter" id="intro">
... For details see the chapter: <a href="#install" class="number"/>.
</div>
<div class="chapter" id="install">
...
</div>
```

The text should render like:

```
For details see the chapter 2.
```

you can use the CSS:

```
:root {
  counter-reset: chapter;
}
div.chapter{
  counter-increment: chapter;
}
a.number {
  content: target-counter(attr(href), chapter, decimal);
  oxy-link: attr(href);
}
```

Related Information:

- [W3 CSS Generated Content for Paged Media Module: The 'target-counter' function](#)

## External links

Usually when linking to resources outside the documentation, normal web links are used.

There are two aspects to take in consideration when styling them:

- When printed on paper, show the entire URL so that the user can see it and type it in a browser.
- When displayed in a PDF reader, mark it as a link so that the user can click on it.

For example:

```
<p>This is a link to the <a href='http://www.w3.org/'>W3C</a> website. </p>
```

To fulfill both conditions, you can add text after the "W3C" text, with the entire value of the `href` attribute, and use the `link` or `-oxy-link` property to mark the generated content as being a link:

```
a:after {
  content: "(" attr(href) ";";
  link: attr(href);
}
```

## Creating a Table of Contents (TOC)

A TOC (table of contents) is a special page that contains links to the chapters and sections of your book. Each line contains:

- The title of the chapter or section.
- A line of dots or other decoration, called leader.
- The page number of the target chapter/section.

It may look like this:

```
Installing the Software .....10
  On Windows .....12
  On Mac.....17
```

Obviously there should be some structure in your document that reflects the tree of the TOC, with ID links.

```
<ul class="toc">
  <li><a href="#introduction">Introduction</a></li>
  <li><a href="#installing">Installing the Software</a></li>
  <li><ul>
    <li><a href="#installing_win">On Windows</a></li>
  </ul></li>
  <li><ul>
    <li><a href="#installing_mac">On Mac</a></li>
  </ul></li>
</ul>
```

You can use the same `target-counter` function as for [Cross References \(on page 34\)](#), but first suppose that you want to create a special page for the TOC:

```
@page toc {
  @top-center {
    content: "Table of Contents";
  }
  @bottom-center {
    content: counter(page, lower-roman);
  }
}
```

This page places the "Table of Contents" text in the header of the page hosting the TOC and puts the number of the TOC page in the footer, with lower roman digits.

The next example associates the defined page to the `ul` element that gives the structure:

```
ul.toc {
  page: toc;
}
```

To create the TOC entries, the next example removes the bullet decoration from the `li` elements, then marks the `a` element as being a link.

```
ul.toc li {
  list-style-type:none;
}
ul.toc a {
  link: attr(href);
  text-decoration:none;
}
```

In the next example, after the `a` element, a `leader` is used to expand to the available width. You can specify a character pattern for the leader:

- **dotted** - Creates an area filled with dots.
- **solid** - Creates an area filled with a dash.
- **space** - Creates an area filled with spaces.

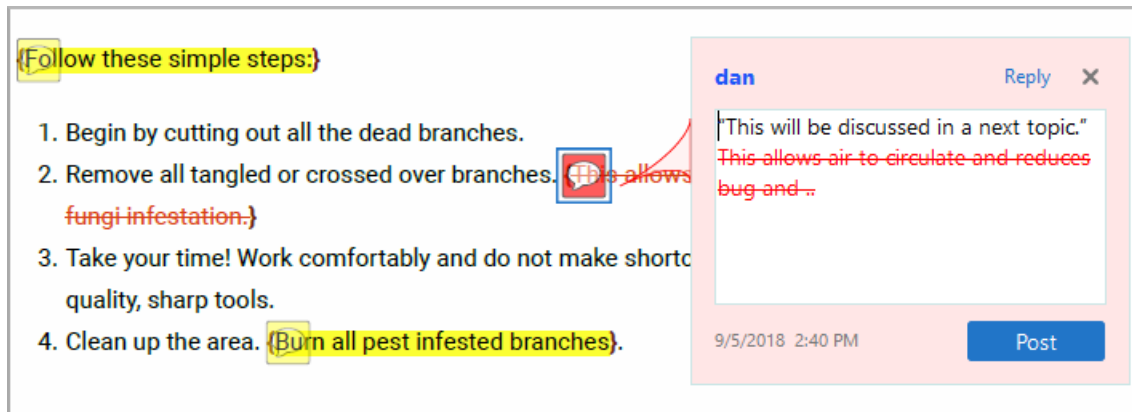
Also, after the leader, it uses the page number of the target element:

```
ul.toc a:after{
  color:blue;
  content: leader(".") target-counter(attr(href), page);
  link: attr(href);
}
```

## Annotations

You may create generic comments or mark a particular document fragment as being inserted, deleted, or highlighted. The rendering in each of these cases will be different.

Figure 1. Chemistry Annotations in Acrobat Reader



In the above image, commented sections are shown in yellow and deleted content is in red.

To create annotations in your PDF output, create a specific structure in your document. The child topics in this section contain information about how this structure could look like for XML documents and HTML documents.

If you want to use CSS rules to style the change elements as plain elements in the content, you can disable the annotation processing using the command-line parameter: `-annotations-for-change-tracking-and-comments`.

## Comments and Tracked Changes - XML Fragment

This section contains information about how each type of tracked change is structured in an XML file.

### Insertions

For an insertion type of tracked change, the structure that defines the insertion details is inside the `range` ( `<oxy-range-start>` to `<oxy-range-end>` ), the inserted text is highlighted by an `<oxy-insert-hl>` element, and the details are stored in the `<oxy-insert>` element.

```
<oxy:oxy-range-start id="sc_1" hr_id="1"/>

<oxy:oxy-insert href="#sc_1" hr_id="1">
  <oxy:oxy-author>dan</oxy:oxy-author>
  <oxy:oxy-content>insert</oxy:oxy-content>
  <oxy:oxy-date>2018/03/15</oxy:oxy-date>
  <oxy:oxy-hour>09:38:29</oxy:oxy-hour>
  <oxy:oxy-tz>+02:00</oxy:oxy-tz>
</oxy:oxy-insert>

<oxy:oxy-insert-hl>This is an insert!!</oxy:oxy-insert-hl>
```

```
<oxy:oxy-range-end hr_id="1" />
```

## Comments


Similar to insertions, comments are defined in a *range* ( `<oxy-range-start>` to `<oxy-range-end>` ), the comment details is in the `<oxy-comment>` element, and the highlighted content is wrapped in the `<oxy-comment-hl>` element.

```
<oxy:oxy-range-start id="sc_1" hr_id="1" />

<oxy:oxy-comment href="#sc_1" hr_id="1">
  <oxy:oxy-author>dan</oxy:oxy-author>
  <oxy:oxy-comment-text>This is a comment.</oxy:oxy-comment-text>
  <oxy:oxy-date>2018/03/15</oxy:oxy-date>
  <oxy:oxy-hour>09:56:59</oxy:oxy-hour>
  <oxy:oxy-tz>+02:00</oxy:oxy-tz>
</oxy:oxy-comment>

<oxy:oxy-comment-hl>Context</oxy:oxy-comment-hl>

<oxy:oxy-range-end hr_id="1" />
```

 **Note:** Comments that are marked as done have a `flag="done"` attribute:

```
<oxy:oxy-comment href="#sc_6" hr_id="6" flag="done">
```

## Attribute changes

The attribute changes are more complex. The *range* is empty, and is directly above the affected element (the one that has modified attributes). The `<oxy-attributes>` element contains details about multiple attribute changes, each stored in the `<oxy-attributed-change>` element.

```
<element>

<oxy:oxy-range-start id="sc_3" hr_id="3" />
<oxy:oxy-range-end hr_id="3" />

<oxy:oxy-attributes href="#sc_3" hr_id="3">

  <oxy:oxy-attributed-change type="inserted" name="platform">
    <oxy:oxy-author>dan</oxy:oxy-author>
```

```

<oxy:oxy-current-value>windows</oxy:oxy-current-value>

<oxy:oxy-date>2018/03/15</oxy:oxy-date>

<oxy:oxy-hour>10:05:04</oxy:oxy-hour>

<oxy:oxy-tz>+02:00</oxy:oxy-tz>

</oxy:oxy-attribute-change>

....

<oxy:oxy-attribute-change type="removed" name="audience">

....

</oxy:oxy-attribute-change>

</oxy:oxy-attributes>

...

</element>

```

## Deletions

For a deletion, there are some elements that define the start and end of the deletion, and the highlighted text is wrapped in the `<oxy-delete-hl>` element.

```

<oxy:oxy-range-start id="sc_2" hr_id="2" />
<oxy:oxy-delete-hl> This is a deleted text. </oxy:oxy-delete-hl>
<oxy:oxy-range-end hr_id="2" />

```

There is a structure that offers details about the deletion change, using the `<oxy-delete>` element. This is linked to the above deletion range by the same ID value:

```

<oxy:oxy-delete href="#sc_2" hr_id="2">
  <oxy:oxy-author>dan</oxy:oxy-author>
  <oxy:oxy-content><img href="../img/ex.gif"></oxy:oxy-content>
  <oxy:oxy-date>2018/03/14</oxy:oxy-date>
  <oxy:oxy-hour>11:38:06</oxy:oxy-hour>
  <oxy:oxy-tz>+02:00</oxy:oxy-tz>
</oxy:oxy-delete>

```

## Colored Highlights

To show some text as highlighted with a background color:

```

<oxy:oxy-color-hl color="rgba(140,255,140,50)">Some colored text.</oxy:oxy-color-hl>

```

## Comments and Tracked Changes - HTML Fragment

This section contains information about how each type of tracked change is structured in an HTML file.

### Insertions

For an insertion type of tracked change, the structure that defines the insertion details is inside a *range* ( `oxy-range-start` to `oxy-range-end` ), the inserted text is highlighted by a `<span>` element with the class `oxy-insert-hl` , and the details are stored in a `<span>` element with the `oxy-insert` class.

```
<span class="oxy-range-start" id="sc_1" hr_id="1"/>

  <span class="oxy-insert" href="#sc_1" hr_id="1">
    <span class="oxy-author">dan</span>
    <span class="oxy-content">insert</span>
    <span class="oxy-date">2018/03/15</span>
    <span class="oxy-hour">09:38:29</span>
    <span class="oxy-tz">+02:00</span>
  </span>
  <span class="oxy-insert-hl">This is an insert!!</span>

<span class="oxy-range-end" hr_id="1"/>
```

### Comments


Similar to insertions, comments are defined in a *range* ( `oxy-range-start` to `oxy-range-end` ), the comment details in an element with the class `oxy-comment` , and the highlighted content is wrapped in the `oxy-comment-hl` element.

```
<span class="oxy-range-start" id="sc_1" hr_id="1"/>

  <span class="oxy-comment" href="#sc_1" hr_id="1">
    <span class="oxy-author">dan</span>
    <span class="oxy-comment-text">This is a comment.</span>
    <span class="oxy-date">2018/03/15</span>
    <span class="oxy-hour">09:56:59</span>
    <span class="oxy-tz">+02:00</span>
  </span>
```

```
<span class="oxy-comment-h1">The commented text.</span>

<span class="oxy-range-end" hr_id="1"/>
```

 **Note:** Comments that are marked as done have a `flag="done"` attribute:

```
<span class="oxy-comment" href="#sc_6" hr_id="6" flag="done">
```

## Attribute changes

The attribute changes are more complex. The range is empty, and is directly above the affected element (the one that has modified attributes). The element with the class `oxy-attributes` contains details about multiple attribute changes, each stored in an element with the class `oxy-attribute-change`.

```
<element>

<span class="oxy-range-start" id="sc_3" hr_id="3"/>
<span class="oxy-range-end" hr_id="3"/>

<span class="oxy-attributes" href="#sc_3" hr_id="3">

  <span class="oxy-attribute-change" type="inserted" name="platform">
    <span class="oxy-author">dan</span>
    <span class="oxy-current-value">windows</span>
    <span class="oxy-date">2018/03/15</span>
    <span class="oxy-hour">10:05:04</span>
    <span class="oxy-tz">+02:00</span>
  </span>
  ....
  <span class="oxy-attribute-change" type="removed" name="audience">
    ....
  </span>
</span>
...
</element>
```

## Deletions

For a deletion, there are some elements that define the start and end of the deletion, and the highlighted text is wrapped in an element with the class `oxy-delete-hl`.

```
<span class="oxy-range-start" id="sc_2" hr_id="2"/>
<span class="oxy-delete-hl"> This is a deleted text. </span>
<span class="oxy-range-end" hr_id="2"/>
```

There is a structure that offers details about the deletion change, using the element with the class `oxy-delete`. This is linked to the above deletion range by the same ID value:

```
<span class="oxy-delete" href="#sc_2" hr_id="2">
  <span class="oxy-author">dan</span>
  <span class="oxy-content"><img href="../img/ex.gif"></span>
  <span class="oxy-date">2018/03/14</span>
  <span class="oxy-hour">11:38:06</span>
  <span class="oxy-tz">+02:00</span>
</span>
```

## Colored Highlights

To show some text as highlighted with a background color:

```
<span class="oxy-color-hl" color="rgba(140,255,140,50)">Some colored text.</span>
```

## PDF Output

You may have specific requirements for the PDF files you need to produce (such as the set of metadata, bookmarks, the level of accessibility, or the PDF format).

## Bookmarks

PDF bookmarks provide an additional way of navigating, similar to a table of contents. The tree bookmark structure is intended to be used by the PDF readers, usually displayed in a side view. More often, the bookmarks show the logical hierarchy of the book, with pointers to the chapters and section, similar to a TOC. Creating bookmarks has no effect on the printed material.

Oxygen PDF Chemistry can create PDF bookmarks by using the standard CSS properties: `bookmark-level`, `bookmark-label`, and `bookmark-state`.

For an HTML document, you can collect the titles from the heading elements text.

```
h1, h2, h3, h4, h5, h6 {
    bookmark-label: content(text);
}
```

In the following example, the `:before` pseudo element is concatenated. That prefixes each of the **h1** with the value of the chapter number, with the text from the element:

```
h1 {
    bookmark-label: content(before) " / " content(text);
}


h1:before{
    content: counter(chapter);
    counter-increment: chapter;
}
```

You can define the level (depth in the hierarchy) of the bookmarks. The deeper the section, the higher the level:

```
h1 { bookmark-level: 1; }
h2 { bookmark-level: 2; }
h3 { bookmark-level: 3; }
h4 { bookmark-level: 4; }
h5 { bookmark-level: 5; }
h6 { bookmark-level: 6; }
```

Also, you can control if the bookmarks are shown expanded or collapsed in the bookmark view. By default, all bookmarks are open. To close all the nodes from the level 2, you can use:

```
h2 {
    bookmark-state: closed;
}
```

 **Note:** In the built-in CSS that Oxygen PDF Chemistry uses for processing HTML, the bookmarks are already configured using the `bookmark-level` and `bookmark-label` properties. If you need to set the closed/open state, you should use the `bookmark-state` property in your custom CSS file.

Related Information:

- [W3C Working Draft: Generated Content for Paged Media Module: Bookmarks](#)

## Metadata

PDF files may contain metadata. Metadata provide additional information about a certain document, such as its title, author, organization, creation date, format, or copyright.

HTML defines the `meta` element for keeping track of information that describes your content. Most of this information should migrate to the PDF document properties. The property values may be either *static* (specified directly from the CSS) or *dynamic* (collected from the document) using the following functions:

- `content(text)`
- `attr()`
- `oxy_xpath()`

## Predefined Meta Fields

Examples of common metadata:

- Publication title
- Author
- Keywords
- Short description

Oxygen PDF Chemistry automatically extracts this informations from HTML documents.

Suppose that you have the following arbitrary XML document:

```
<doc>
  <title>Publication title</title>
  <meta name='keywords' content='software, network'>
  <meta name='description' content='This is a publication about software
products...'>
  <meta name='author' content='John, jo@mysite.example.com'>
  ...
```

You could use any of the following CSS selectors to extract the metadata:

### **-oxy-pdf-meta-title**

It is used to extract the publication title. You can use it by matching the `title` element:

```
title {
  -oxy-pdf-meta-title: content(text);
}
```

If this CSS selector matches multiple elements, only the first element in the document order will be used to extract the title.

### **-oxy-pdf-meta-author**

It is used to extract the publication author. You can use it by matching the `meta` element with the attribute `name='author'` :

```
meta[name='author'] {
  -oxy-pdf-meta-author: attr(content);
}
```

If this CSS selector matches multiple elements, only the first element in the document order will be used to extract the author.

### **-oxy-pdf-meta-description**

It is used to extract the publication description. You can use it by matching the `meta` element with the attribute `name='description'` or `name='description'` :

```
meta[name='description'],
meta[name='subject'] {
  -oxy-pdf-meta-description: attr(content);
}
```

If this CSS selector matches multiple elements, only the first element in the document order will be used to extract the description.

### **-oxy-pdf-meta-keywords**

It is used to extract the publication keywords. For example you can use it by matching the `meta` element with the attribute `name='keywords'` :

```
meta[name='keywords'] {
  -oxy-pdf-meta-keywords: attr(content);
}
```

If this CSS selector matches multiple elements, only the first element in the document order will be used to extract the keywords.

### **-oxy-pdf-meta-keyword**

It is used to extract a single keyword. Individual keywords are accumulated from elements that match the CSS rule that uses this property and then concatenated into a single string. This single string is then set in the PDF 'keywords' section. For example, if you mark keywords in your HTML document with a span with a "kw" class, you can collect them all by using:

```
span.kw {
  -oxy-pdf-meta-keyword: content(text);
}
```

## Custom Meta Fields

Metadata is not restricted to the above cases. You may have custom metadata fields. It is usually displayed in a tabular format (for example, in Acrobat Reader™ it is in the **Custom** tab in the **Properties** dialog box).

### **-oxy-pdf-meta-custom**

This property has two parts, one defining the name, and one the value for the field.

In the following example, all the meta tags are dumped as custom meta fields in PDF:

```
meta {
  -oxy-pdf-meta-custom: attr(name) attr(content);
}
```

Or if somewhere in the document content, you have a span that defines the document creation date, you can use:

```
span.created {
  -oxy-pdf-meta-custom: "CreationDate" content(text);
}
```

In case of conflicts, when two or more elements trigger setting of a meta field with the same name, only the first definition of a meta field will be used in the PDF output.

## Named Destinations (Anchors)


The *named destinations* FO extension provides a way to link to a particular anchor within a PDF document.

Suppose your PDF output is published on a website and accessible at the URL `http://my_site.com/files/my_document.pdf`, and the original XML document has a `section` element with an `id` attribute set to `installation`.

```
...
<section id="installation"
...
</section>
...
```

To open it in the PDF reader exactly at that particular section (with the `id` value of `installation`), you can use the `#installation` anchor in the URL: `http://my_site.com/files/my_document.pdf#installation`.

Oxygen PDF Chemistry declares named destinations for any `id` or `xml:id` attributes from your input XML document. As an alternative, if you do not want to alter the ids in the document, the `nd:nd-id` attribute can be used. In this case, make sure the `nd` prefix is bound to the `xmlns:nd="http://www.oxygenxml.com/css2fo/named-destinations"` namespace.

 **Note:** The *named destinations* functionality can be disabled by using the `-generate-named-destination` command-line parameter (on page 5).

## Accessibility (508 Compliance)

It is recommended that you make your PDF output accessible for people who are blind or visually impaired. Many government organizations require the documents to be accessible.

### PDF Accessibility Tagging

By default, Oxygen PDF Chemistry partially creates accessible PDF documents in the sense that most of the paragraphs, tables, lists, headers, and footers are tagged automatically for any XML vocabulary, and PDF readers use this information to present the content.

In addition, the default CSS files used by Oxygen PDF Chemistry to generate PDF based on HTML defines accessibility tags for headings (`H1..H6`), quotations (`Q`), sections (`SECTION`), and pre-formatted text (`PRE`).

However, this tagging just takes the element name into account. If your element has a different semantic, you can impose a different PDF accessibility tag by using the `-oxy-pdf-tag-type` extension. In the following example, a paragraph with the `note` class will be marked:

```
p.note {
  -oxy-pdf-tag-type: "Note";
```

```
}
```

**Note:** The headers and footers (or other text placed in the page margins) are automatically marked as artifacts, so they are ignored by the screen readers.

## Hints for Making Documents More Accessible

**Hint 1:** The title of the document must be marked using the metadata.

This is important for accessibility since it will allow the screen reader to identify the publication title.

This is an example using the `-oxy-pdf-meta-title` extension:

```
title {
  -oxy-pdf-meta-title: content(text);
}
```

**Note:** The default CSS files for generating PDF based on HTML already contains this rule.

**Hint 2:** Use `xml:lang` on the root of your document. For HTML documents, use the `lang` attribute.

**Hint 3:** Put an alternate text on all the images.

Oxygen PDF Chemistry supports the `-oxy-alt-text` extension that can be used to associate the alternate text.

The following is an example from the Oxygen PDF Chemistry default CSS for generating PDF based on HTML, where it maps the property to the value of the `alt` attribute of the `img` tag:

```
img {
  -oxy-alt-text: attr(alt);
}
```

For embedded SVG, Oxygen PDF Chemistry automatically uses the `title` element as the alternate text of the image.


For embedded MathML, Oxygen PDF Chemistry automatically uses the `alttext` attribute as the alternate text of the equation.

## Fully Accessible PDF (PDF/UA1)

To make the PDF fully accessible, you have to activate the PDF/UA-1 mode. PDF/UA documents meet the regulations set in Section 508. This mode has special requirements:

1. Activate the PDF UA-1 mode from the command line using the `-pdf-ua` parameter.


2. All the fonts must be embedded. If you are using one of the basic fonts (such as "Times", "Helvetica", etc.), make sure you explicitly define CSS font faces for them. For details, see: [Font Embedding \(on page 72\)](#).

 **Troubleshooting:** If you are using fonts other than the basic ones and still have problems embedding the basic default fonts, make sure all elements are styled using one of your fonts of choice. A catch all CSS rule might be helpful:

```
* {  
  font-family: lato;  
}
```

3. The title of the document must be marked using the metadata. This is important for accessibility since it will allow the screen reader to identify the publication title. This is important for accessibility since it will allow the screen reader to identify the publication title. This is an example using the `-oxy-pdf-meta-title` extension:

```
title {  
  -oxy-pdf-meta-title: content(text);  
}
```

 **Note:** The default CSS files for generating PDF based on HTML already contains this rule.

## Tools for Checking the Document Accessibility

- For smaller documents, this site might be helpful: <http://www.access-for-all.ch/ch/pdf-werkstatt/pdf-accessibility-checker-pac.html>
- From Adobe: <https://helpx.adobe.com/acrobat/using/create-verify-pdf-accessibility.html>

### Related Information:

- [Metadata \(on page 45\)](#)

# 3.

## Element Layout

### Tables

This section is of special interest if you are creating a CSS stylesheet for a custom XML document. For HTML, Oxygen PDF Chemistry already defines the needed selectors for the `table` element.

Suppose you have a very simple table in an XML document:

```
<tbl>
  <caption>A table.</caption>
  <rw>
    <c rowspan="2"> Spans vertically </c>
    <c> Other cell </c>
    <c> Other cell </c>
  </rw>
  <rw>
    <c colspan="2"> Other cell, spanning to the right </c>
  </rw>
</tbl>
```

### Defining Rows and Cells

First, mark the `tbl` element as being a table:

```
tbl {
  display: table;
}
```

Next, the rows and cells:

```
rw {
  display: table-row;
}
c {
  display: table-cell;
}
```

## Defining the Column and Row Span

The processor needs to know how the cells are spanning multiple rows or columns. For these, there are two properties available: `table-column-span` and `table-row-span`.

```
c[cspan] {
    table-column-span: attr(cspan, integer);
}
c[rspan] {
    table-row-span: attr(rspan, integer);
}
```

## Defining the Table Caption

You can do this by using the `display:table-caption`. To change the position of the caption relative to the table grid, you need to use the `caption-side` property:

```
caption {
    display:table-caption;
    caption-side:bottom;
}
```

## Repeating Headers and Footers

The elements marked with one of the properties `table-header-group` or `table-footer-group` are automatically repeated when a table is split over a sequence of pages. It is required that they contain only elements with the display `table-row`.

```
<tbl>
  <head>
    <rw>
      <c> Name</c>
      <c> Value </c>
    </rw>
  </head>
  <body>
    <rw> ... </rw>
    <rw> ... </rw>
  </body>
</tbl>
```

The CSS:

```
head {
  display: table-header-group;
}
```

## Column Widths and Styles

Oxygen PDF Chemistry supports automatic layout of the tables. This means allocation of column widths is done automatically based on the content size. In the following example, the HTML table has an automatic layout (this is also the default):

```
table {
  table-layout: auto;
}
```

**i Tip:** For large tables with long words that bleed out of the page, you can choose to hyphenate the words from the cells. If the table is using an automatic layout, you should specify a width (like 100%), otherwise the table columns will just be wide enough to accommodate the smaller hyphenated words:

```
table {
  table-layout: auto;
  width: 100%;
  hyphens: auto;
}
```

To switch to a fixed layout:

```
table {
  table-layout: fixed;
}
```

If you use the fixed layout and you are not satisfied with equal column widths, for HTML you should use the `style` attribute on the `col` element:

```
<table style="width:2in;">
  <colgroup>
    <col style="width: 40%; background-color:cyan"/>
    <col style="width: 60%"/>
  </colgroup>
  ...
```

For arbitrary XML, you should set a `display` `table-column` to the element representing the column. Consider the following XML:

```
<tbl>
  <colgr>
    <column wd="30%" />
    <column wd="70%" />
  </colgr>
...
```

The following CSS links the `wd` attribute to the width property and defines a different background for the first column.

```
tbl {
  display: table;
}
colgr {
  display: table-column-group;
}
column {
  display: table-column;
  width: attr(wd);
}
column:first-of-type {
  background-color: yellow;
}
```

Proportional column widths like the ones used in the CALS tables from DITA or DocBook are supported, but only when set in a `width` attribute on the column element (the one with `display` `table-column`):

```
...
<column width="3*" />
<column width="7*" />
...
```

## Rotating Tables

There are cases where you have large tables and you need to rotate them to make them fit on your page. For instance, the default page is portrait, but because you have a table with many columns (and thus very wide), it would bleed to the right of the page.

There are two ways of doing this:


- Associate a wider page (e.g. landscape) to the table that needs more space. The disadvantage is that the table will force a page break before and after it.

```
@page landscape-page-for-large-tables {
    size: A4 landscape;
}

tbl {
    display: table;
    page: landscape-page-for-large-tables;
}
```

- Rotate the table using the `transform` CSS property. The table will not create page breaks, but is susceptible to bleeding if its height exceeds the page width.

```
tbl {
    display: table;
    width: 200pt;
    transform: rotate(90deg);
}
```

 **Note:** The table needs to have `table-layout: fixed`, and a `width`.

The page is now landscape and you probably need to also change the headers and footers to match this new orientation. One way of doing this is to move the header content from the `@top-left`, `@top-center`, and `@top-right` rules into page margin boxes from the right, and apply a `transform` property on them.

In the following example, for the normal pages there is static text in the top (header) that is either placed in the left or right side of the page, depending on the page position:

```
@page :left {
    @top-left {
        content: string(maptitle) string(parttitle) string(chaptertitle) " | "
        counter(page);
        font-size: 8pt;
    }
    @bottom-left {
        /**/
    }
    @bottom-right {
        /**/
    }
}
```

```

}
}

@page :right{
  @top-right {
    content: string(maptitle) string(parttitle) string(chaptertitle) " | "
counter(page);
    font-size:8pt;
  }
  @bottom-right {
    /**/
  }
  @bottom-left {
    /**/
  }
}
}

```

For the landscape page, you can move content to either the right-bottom or right-top, depending on the page position:

```

/*
 * Table orientation.
 */
@page table-landscape:right {
size:landscape;

@top-left{
  content:none;
}
@top-center{
  content:none;
}
@top-right{
  content:none;
}

  @right-bottom{
    content: string(chaptertitle) " | " counter(page);
    font-size:8pt;
transform:rotate(90);
vertical-align: middle;

```

```

text-align: right;
}
}
@page table-landscape:left {
size:landscape;

@top-left{
content:none;
}
@top-center{
content:none;
}
@top-right{
content:none;
}

@right-top{
content: string(maptitle) string(parttitle) string(chaptertitle) " | "
counter(page);
font-size:8pt;
transform:rotate(90);
vertical-align: middle;
text-align: left;
}
}

```

## Lists

For the HTML lists ( `ol` , `ul` ), Oxygen PDF Chemistry already defines the needed selectors, but sometimes you need greater control over the spacing or style of the marker.

You need the list element to have the `display: block` , and the children, the `display: list-item` .

```

ul {
display:block;
}

li {
display:list-item;
list-style-type: disc;
}

```

```
margin-left: 0.5in;  
}
```

Make sure you have a margin left so the bullet will have enough space to be painted inside the list item box.

## List Marker Position

You can select that the marker should be considered a decorator outside the box of the list item element (this is the default), or if it should be inline and enter the first line of the content of the element.

```
li.inside {  
    list-style-position: inside;  
}  
li.outside {  
    list-style-position: outside;  
}
```

## The *list-style-type* and *list-style-image* CSS Properties

Oxygen PDF Chemistry supports the following values for the `list-style-type` property:

- **box**
- **check**
- **circle**
- **diamond**
- **disc**
- **hyphen**
- **square**
- **none**
- **decimal**
- **lower-roman/lower-latin**
- **upper-roman/upper-latin**
- **decimal-leading-zero**

To use an image as a marker instead of a standard bullet or number, you can use the `list-style-image` property. You have to use the `url` function to point to an image resource:

```
li {  
    list-style-image: url("images/my_list_bullet.svg");  
}
```

## Using a `:marker` Pseudo Selector

There is a CSS pseudo element that allows you to associate styles with the list marker. The following example changes the background color, font, width, and even the content of a list marker:

```
ol {
  ...
  counter-reset: cnt;
}

li:marker {
  width: 3em;
  background-color: silver;
  color: red;
  font-weight: bold;
  text-align: left;
  counter-increment: cnt;
  content: counter(cnt) " - \0430";
}
```

To use an image instead of a number as marker:

```
li:marker{
  content: url("images/my_list_bullet.svg");
}
```

You can even implement a custom list numbering using this selector. Such a technique may be useful for other list numbering schemes that are currently not supported, or are more exotic, for lists with up to tens of elements. You can use the `nth-of-type()` selector to choose the labels of each item, individually:

```
li:nth-of-type(1):marker{
  content: "alpha";
}
li:nth-of-type(2):marker{
  content: "beta";
}
li:nth-of-type(3):marker{
  content: "gamma";
}
...
```

## Floats

Floats are not supported by Oxygen PDF Chemistry.

## Aligning Blocks Horizontally

If you want to align just the text paragraphs from the block, you might find the `text-align` property useful.

To align a block, it must have a specified `width` (absolute or as percentage).

### Align Center

This can be done by setting both the `margin-left` and `margin-right` to `auto`:

```
div {  
  width: 200pt;  
  margin-left: auto;  
  margin-right: auto;  
}
```

### Align Left

This can be done by setting the `margin-left` to `0` and `margin-right` to `auto`:

```
div {  
  width: 200pt;  
  margin-left: 0;  
  margin-right: auto;  
}
```

### Align Right


This can be done by setting the `margin-left` to `auto` and the `margin-right` to `0`:

```
div {  
  width: 200pt;  
  margin-left: auto;  
  margin-right: 0;  
}
```

## Rotating Blocks

There are cases where you need to rotate some block elements for visual effects. For this, you should use the `transform` CSS property.

```
code {  
  transform: rotate(90deg);  
}
```

 **Note:** The block needs to have a specified `width`.

# 4.

## Graphics

### Images

Images are very important in any printed material. This section contains information about how you can reference them from your HTML or XML documents.

For HTML, the `img` tag is recognized as an image without any other styling in your CSS files:

```
...  
<p> And this is the picture of a happy face: . </p>  
...
```

For XML, you must add CSS rules that pick up the content of an attribute and uses it as a source for an image.

```
...  
<para> And this is the picture of a happy face: <imagedata fileref="happy.png" />.</  
para>  
...
```

The following example uses static content on the `imagedata` before pseudo element:

```
imagedata[fileref]:before {  
    content: attr(fileref, url);  
}
```

It is important to use the `url` keyword when retrieving the attribute value. It signals that the value is a pointer to an external image.

### Setting Image Width and Height

The image size can be determined from the number of pixels of the image, taking the [image resolution \(on page 65\)](#) into account (if available). There are cases where this computed size is not what you need, and you want to specify the size explicitly.

For HTML, it is enough to use the image attributes directly in your document.

```


```

For an arbitrary XML, you can indicate the image width and height through a rule that matches the element (or its pseudo elements `:before` or `:after`) and sets the `width` and `height` CSS properties.

```
imagedata {
  display: inline;
  content: attr(src, url);
  width: attr(width, length);
  height: attr(height, length);
}
```

Or, if you use an image as a decorator, you can specify fixed dimensions in the CSS:

```
chapter title:before {
  content: url("my_artwork.png");
  width: 300px;
}
```

 **Note:** For static content as in the example above, Oxygen PDF Chemistry tries to use the width and height set on the pseudo element, then the ones that are set on the parent element, but only if the static content is composed of a single image. Mixing text and images in the `content` property disables the width and height specification.

If you want to limit the width of the images to a maximum size, you can use the `max-width` property. The image will be scaled down to fit the maximum size, if it is larger.

```
imagedata {
  ...
  content: attr(src, url);
  max-width: attr(width, length);
  ...
}
```

## Background Images

You can use background images to impose a texture. You can use them to decorate an entire page, or a specific element from your document.

Supported properties:


- **background-image**
- **background-repeat**
- **background-position**


## Page Background Images

You can set a background for a page. Usually you do this for the cover page to impose a full page artwork, or to add graphics to the header and footer of the page. Here is an example of how can you do it for the page:

```
@page cover {
  size:A4;
  margin:1in;
  background-image:url("images/my_book_cover_artwork.png");
  background-repeat:no-repeat;
}

div.cover {
  page: cover;
}
```

 **Note:** It is better to use SVG for the page artworks. It does not suffer from pixelation. If you are limited to using raster images, you can fine-tune their resolution by using the [image-resolution](#) (*on page 65*) property.

 **Note:** To separate the header and footer from the main content using visual effects (lines, shadows, etc.), you can use a full page size artwork in SVG and set it to the default page:

```
@page {
  background-image:url("images/my_header_footer_artwork.svg");
  background-repeat:no-repeat;
}
```

If your artwork is smaller, consider a "DRAFT" watermark (for instance). You should use the `background-position` to place it where you need:

```
@page {
  ...
  background-image:url("images/draft.svg");
  background-position: bottom center;
  ...
}
```

```
}
```

## Element Background Images

You can style the background of your elements as you do for web pages:

```
section {
  background-image: url("my_repeating_pattern.svg");
  background-repeat: repeat-y;
}
```

## Image Resolution


Some raster images (pixels, not vector) may have a default resolution, set by the designer, using an image-editing software. Usually the image size and resolution are set to look best on the screen. The advantage of a resolution set in the image itself is that it will have the same effective size on the screen and on paper. For example, if the image has 144 dots in width, and an embedded resolution of 72dpi, it will be two inches on screen and on paper.

The problems start to arise when the resolution is not set in the image, and the PDF processor has to decide what resolution to use to determine the size of the graphic. To solve this, the processor extracts the DPI from:

- The `image-resolution` CSS property associated to the element that contains the image.
- The command-line parameter `-image-resolution`.
- The built-in fallback resolution of 96 DPI.

The recommended way to change this is by using the CSS Level 4 `image-resolution` property:

```
img {
  image-resolution: 300dpi;
}
```

 **Note:** The `image-resolution` is inheritable, so you can associate it on the root element. It does not apply on the page generated content (margin boxes).


```
:root {
  image-resolution: 300dpi;
}
```

To reset the image resolution to the one set in the image itself, you can use the constant `from-image` instead of a DPI value:

```

title:before {
  content: url("chapter-decorator.png");
  image-resolution: from-image;
}

```

 **Note:** To change the resolution for images that appear in a page margin box, set this property on that margin box, or directly on the @page rule, to apply to all page margin boxes:

```

@page front-page {
  image-resolution: 600dpi;
}

@page {
  @top-center {
    image-resolution: 600dpi;
    content: url("company-logo.png");
  }
}

```

#### Related Information:

- [Command Line Interface \(on page 3\)](#)
- <https://drafts.csswg.org/css-images-4/#the-image-resolution>

## Supported Image Types

Oxygen PDF Chemistry supports the following raster image types:

- BMP (Microsoft Windows Bitmap)
- GIF (Graphics Interchange Format)
- JPEG (Joint Photographic Experts Group)
- PNG (Portable Network Graphic)
- TIFF (Tag Image Format File)

And the following vector images:

- SVG (Scalable Vector Graphics)
- WMF (Windows Metafile)

- PDF (PDF Documents)

## PDF Images

You can reference PDF images the same way other image files are referenced:

```

```

To point to a single page from your PDF document, use the following syntax (this example points to page 5):

```

```

## SVG

Oxygen PDF Chemistry supports SVG images. The advantage of using SVG is that the image looks good on paper.


### SVG Referenced or Embedded into the Document

These can be either referenced as external resources:

```
<p> This is an SVG showing a happy face: </p>
```

Or embedded into the document, as SVG fragments:

```
<p> This is a red circle:
  <svg xmlns='http://www.w3.org/2000/svg' viewBox="0 0 50 50" width="50" height="50">
    <circle cx="25" cy="25" r="10" fill="#F00" />
  </svg>
</p>
```

 **Note:** No styling is required to show the embedded SVG in the output. It works automatically for HTML or other XML.

 **Note:** For HTML5, the namespace declaration is not required.

### Using SVG for Styling

To use it to decorate an element:


```
div.note:before {
  content:url("images/note.svg");
```

```
}
```

To set an SVG image as background of a page, or a page margin box:

```
@page coverpage{
  background-image: url("images/clipart.svg");
  background-repeat:no-repeat;
  background-position:center center;

  @top-left {
    background-image: url("images/company.svg");
    background-repeat:no-repeat;
  }
}
```

 **Note:** The `image-resolution` CSS property does not apply for SVG vectors.

## MathML

MathML equations can be either referenced from the document as external resources (the file should be ending with the "mml" extension):

```
<p> The quadratic formula: </p>
```


Or embedded directly in the document:

```
<p> The quadratic formula:
<math>
  <mi>x</mi>
  <mo>=</mo>
  <mfrac>
    <mrow>
      <mo form="prefix">#</mo> <mi>b</mi>
      <mo>±</mo>
    </mrow>
    <msqrt>
      <msup> <mi>b</mi> <mn>2</mn> </msup>
      <mo>#</mo>
      <mn>4</mn> <mo>#</mo> <mi>a</mi> <mo>#</mo> <mi>c</mi>
    </msqrt>
  </mrow>
  <mrow>
    <mn>2</mn> <mo>#</mo> <mi>a</mi>
```

```

</mrow>
</mfrac>
</math>
</p>

```

 **Note:** No styling is required to show the embedded MathML in the output. It works automatically for HTML or other XML.

 **Note:** For HTML5 the namespace declaration is not required.

## Supported Colors

Along with the usual color values, Oxygen PDF Chemistry supports the following special values for all the color properties:

- **rgb** - Used to specify red, green, and blue components (for example, `color="rgb(255 , 0 , 153) "`).
- **rgba** - Used for transparent colors by specifying each color channel and the transparency. For example, the following would result in the background color being magenta since the red color from the parent `div` element will be visible through the blue color of the `p` element:

```

div{
  background-color: rgba(255,0,0,0.3)
}
p{
  background-color: rgba(0,0,255,0.3)
}

```

For more information about color properties, see [MDN Web Docs: Color](#).

# 5.

## Fonts

This section contains information about using fonts to style the output, including the supported fonts, how to use Web fonts and other special fonts, and font embedding.

### Supported Fonts

Oxygen PDF Chemistry supports TTF fonts, either as local files or as web fonts, with the following limitations:

- Some fonts may use features that are unsupported, such as the `MarkGlyphSet` in the GDEF table.
- The advanced features of OTF are not supported, but the Compact Font Format (CFF) extracted from the OTF is embedded as a Type1C font.

### Basic Fonts

By default, all fonts are automatically embedded in PDF documents by Oxygen PDF Chemistry. The exception to this is some fonts that are considered to be available in all PDF rendering applications. These are called basic fonts and include:

- Times (v3) (in regular, italic, bold, and bold italic)
- Courier (in regular, oblique, bold and bold oblique)
- Helvetica (v3) (in regular, oblique, bold and bold oblique)
- Symbol.
- Zapf Dingbats.

### Using Web Fonts

If the font is available from a website (such as [Google Fonts](#)), simply select the font, along with its font weights and insert the generated `@import` declaration in your CSS:

```
@import url('https://fonts.googleapis.com/css?
family=Montserrat:300,300i,400,500,700i');
```

```
section h2 {
```

```
font-family: Montserrat, Serif;
font-weight: 500;
}
```

**i Tip:** It is possible to add a fallback to the remote font ( `Serif` in the above example).

**⚠ Important:** Make sure the website hosting the font is accessible. If you get errors in the console (for example, `Unknown host` or `I/O Exception`) regarding one of the font resources, check your networking proxy settings or your firewall settings. For the parameters that control the HTTP proxy, see [Command Line Interface \(on page 3\)](#).

## Using Installed Fonts

Suppose you want to style certain elements using a font that is available on your system. On Windows, it means it was installed in the `Windows/Fonts` directory. In this case, you can reference it directly like this:

```
section h2 {
    font-family: Calibri;
}
```

## Using Local Font Files

### Font Files from a Directory

If you have your font files located in a particular directory, you can instruct Oxygen PDF Chemistry to load them. To do this, use the `-fonts-dir` command-line argument and just specify the name of the font directly in the `font-family` property, as you would for the built-in fonts. There is no need to define a `@font-face` structure in the CSS.

### Font Files from the Oxygen PDF Chemistry Installation

Another way is to copy your font files in the following subfolder: `[CHEMISTRY_INSTALL_DIR]/config/fonts`. In this case, there is no need to define a `@font-face` structure in the CSS, just specify the name of the font directly in the `font-family` property.

### Font Files Next to the CSS

If the font file is not installed in the system, you can place it next to your custom CSS file. You will have to declare one or more `@font-face` structures, with the same `font-family`, but with

possible different `font-weight` and `font-style` properties (as in the example below where the `TitilliumWeb` font is a bundle of multiple TTF files, each one for a specific font-weight and style). The TTF files were placed in a folder next to the CSS named `fonts/titillium`.

```
@font-face {
  font-family: titillium;
  font-style: normal;
  font-weight: 400;
  src: url(fonts/titillium/TitilliumWeb-Regular.ttf) ;
}
@font-face {
  font-family: titillium;
  font-style: normal;
  font-weight: 300;
  src: url(fonts/titillium/TitilliumWeb-Light.ttf) ;
}
@font-face {
  font-family: titillium;
  font-style: normal;
  font-weight: 200;
  src: url(fonts/titillium/TitilliumWeb-ExtraLight.ttf) ;
}
@font-face {
  font-family: titillium;
  font-style: normal;
  font-weight: 600;
  src: url(fonts/titillium/TitilliumWeb-SemiBold.ttf) ;
}

section h2 {
  font-family: titillium, Serif;
}
```

In more simple cases, you might create a single `@font-face` structure.

#### Related Information:

- [Command Line Interface \(on page 3\)](#)


# Font Embedding

## CSS Font Embedding

All the font families that are referenced from the CSS **are embedded automatically** in the PDF by Oxygen PDF Chemistry.

## Basic Fonts Embedding

Although the basic fonts are guaranteed to be available in all PDF readers, there are some situations where you will have to embed them explicitly when using PDF/Universal Accessibility or Archiving. Because there are some copyright restrictions on these fonts, Oxygen PDF Chemistry cannot redistribute them. You will have to locate the files on your system and declare a set of font faces with the same name as the default ones.

 **Note:** The following example assumes the fonts are placed in a `fonts` directory next to the stylesheet. The names of the fonts may differ on your platform.

```
@font-face{
  font-family:Times;
  font-style: normal;
  font-weight:400;
  src: url("fonts/TIMES.TTF");
}
@font-face{
  font-family:Times;
  font-style: italic;
  font-weight:400;
  src: url("fonts/TIMESI.TTF");
}
@font-face{
  font-family:Times;
  font-style: italic;
  font-weight:700;
  src: url("fonts/TIMESBI.TTF");
}
@font-face{
  font-family:Times;
  font-style: normal;
  font-weight:700;
```

```
src: url("fonts/TIMESBD.TTF");
}

/* ----- */

@font-face{
  font-family:Helvetica;
  font-style: normal;
  font-weight:400;
  src: url("fonts/ARIAL.TTF");
}
@font-face{
  font-family:Helvetica;
  font-style: italic;
  font-weight:400;
  src: url("fonts/ARIALI.TTF");
}
@font-face{
  font-family:Helvetica;
  font-style: italic;
  font-weight:700;
  src: url("fonts/ARIALBI.TTF");
}
@font-face{
  font-family:Helvetica;
  font-style: normal;
  font-weight:700;
  src: url("fonts/ARIALBD.TTF");
}

/* ----- */

@font-face{
  font-family:Courier;
  font-style: normal;
  font-weight:400;
  src: url("fonts/COUR.TTF");
}
@font-face{
  font-family:Courier;
```

```

font-style: italic;
font-weight:400;
src: url("fonts/COURI.TTF");
}
@font-face{
font-family:Courier;
font-style: italic;
font-weight:700;
src: url("fonts/COURBI.TTF");
}
@font-face{
font-family:Courier;
font-style: normal;
font-weight:700;
src: url("fonts/COURBD.TTF");
}

/* ----- */

@font-face{
font-family:Symbol;
font-style: normal;
font-weight:400;
src: url("fonts/SYMBOL.TTF");
}

/* ----- */

@font-face{
font-family:"Zapf Dingbats";
font-style: normal;
font-weight:400;
src: url("fonts/WINGDING.TTF");
}

/* ----- */

@font-face{
font-family:Any;
font-style: normal;
font-weight:400;

```

```

src: url("fonts/TIMES.TTF");
}
@font-face{
  font-family:Any;
  font-style: italic;
  font-weight:400;
  src: url("fonts/TIMESI.TTF");
}
@font-face{
  font-family:Any;
  font-style: italic;
  font-weight:700;
  src: url("fonts/TIMESBI.TTF");
}
@font-face{
  font-family:Any;
  font-style: normal;
  font-weight:700;
  src: url("fonts/TIMESBD.TTF");
}

```

## Using Simulated (Synthetic) Styles

Some fonts do not have a bold or italic variant, and Oxygen PDF Chemistry automatically falls back to the regular font. This topic demonstrates how to instruct Oxygen PDF Chemistry to synthesize font variants. Do not use this technique if you have all the font components.

### Using a Simulated Styles Font from a True Type File (ttf)

Suppose you plan to style the output with the `Raleway` font and you just have the `Raleway-Regular.ttf` font available. The regular font face definition will be:

```

@font-face {
  font-family: "Raleway";
  font-style: normal;
  font-weight: 400; /* normal */
  src: url("fonts/raleway/Raleway-Regular.ttf");
}

```

For the missing style combinations, this example uses the `-oxy-simulate-style: yes` CSS extension property.

```
@font-face {
  font-family: "Raleway";
  font-style: normal;
  font-weight: 700; /* bold */
  -oxy-simulate-style: yes;
  src: url("fonts/raleway/Raleway-Regular.ttf");
}

@font-face {
  font-family: "Raleway";
  font-style: italic;
  font-weight: 400; /* normal */
  -oxy-simulate-style: yes;
  src: url("fonts/raleway/Raleway-Regular.ttf");
}

@font-face {
  font-family: "Raleway";
  font-style: italic;
  font-weight: 700; /*bold*/
  -oxy-simulate-style: yes;
  src: url("fonts/raleway/Raleway-Regular.ttf");
}
```

## Using a Simulated Font Style from a True Type Font Collection (ttc)

### Chemistry Limitation:

The `@font-face` rules in your CSS work as expected when they are pointing to a `.ttf` file. If you intend to use a `.ttc` collection, the value of the `font-family` property should match exactly one of the font names from the collection (a collection contains multiple fonts).

The following example uses the Microsoft JhengHei font (the `msjh.ttc`). Many of the Chinese fonts provide regular, light, bold, but no italic or bold-italic variants. In this case, you may want Oxygen PDF Chemistry to generate the missing variants for you.

To use the regular variant, use a simple `@font-face` definition, making sure the font family is found in the collection:

```
@font-face {
  font-family: "Microsoft JhengHei";
```

```

font-style: normal;
font-weight: 400;
src: url("fonts/jenghei/msjh.ttc");
}

```

For the missing style combinations, the example uses the `-oxy-simulate-style: yes` CSS extension property.

```

@font-face {
  font-family: "Microsoft JhengHei";
  font-style: normal;
  font-weight: 700;
  -oxy-simulate-style: yes;
  src: url("fonts/jenghei/msjh.ttc");
}
@font-face {
  font-family: "Microsoft JhengHei";
  font-style: italic;
  font-weight: 400;
  -oxy-simulate-style: yes;
  src: url("fonts/jenghei/msjh.ttc");
}
@font-face {
  font-family: "Microsoft JhengHei";
  font-style: italic;
  font-weight: 700;
  -oxy-simulate-style: yes;
  src: url("fonts/jenghei/msjh.ttc");
}
@font-face {
  font-family: "Microsoft JhengHei";
  font-style: normal;
  font-weight: 100;
  -oxy-simulate-style: yes;
  src: url("fonts/jenghei/msjh.ttc");
}

```

# 6.

## Localization


### Choosing the Fonts

An important step in making sure your document is published properly in multiple scripts (languages) is choosing a font that covers the entire set of characters from that script (language).

However, you can specify a series of font families (using the CSS `font-family` property) that are used as fall-backs. If a word cannot be rendered using the first specified font, the processor tries the next font family, and so on. The following example uses some common fonts available in Windows to make a CSS stylesheet capable of properly displaying a large number of languages, from all European to Asian languages:

```
* {  
  font-family: Calibri, SimSun, "Malgun Gothic";  
}
```

Assigning a font for all the elements, and not relying on [basic fonts \(on page 70\)](#), makes the document [accessible \(on page 48\)](#).

 **Note:** If your CSS does not assign a font to a part of your document, then the following sequence of fonts is used: Serif, Times, Times New Roman, Symbol, MingLiU, MS Mincho, Arial Unicode MS. This combination covers a wide character range, but might be different from what you need.

### Support for Right to Left Languages

The Unicode BIDI algorithm is applied automatically. For best results in HTML, make sure you mark the right-to-left content, or the left-to-right content embedded in right-to-left, with proper direction attributes:


```
<p dir='rtl'>SOME ARABIC TEXT <p dir='ltr'>Some latin words.</p>.</p>
```

For arbitrary XML, use the `unicode-bidi` and `direction` CSS properties:

```
<para dir='right-to-left'>SOME ARABIC TEXT.</p>
```

CSS:

```
para[dir='right-to-left'] {
  direction: rtl;
  unicode-bidi: embed;
}
```


 **Note:** There are built-in rules in Oxygen PDF Chemistry that automatically match the `dir` attribute in any XML vocabulary with the same semantic as for HTML. The accepted values are: **ltr**, **rtl**, **auto**. If you use other attribute names or other values for this attribute, you should add CSS rules similar to the one above.

For elements in a right to left context, Oxygen PDF Chemistry automatically switches the left borders, paddings, and margins with the ones from the right. This keeps your CSS as simple as possible.

In the following example, the `p` element has a left border.

```
p {
  border-left: 1pt solid orange;
}
```

Suppose it is placed in a `div` with the default `ltr` direction. The orange line is painted in its left border. But if it is placed in a `div` with the `rtl` direction, the orange line will be painted in its right border because that is where the text begins.

 **Note:** To disable this behavior, you may use the `-no-rtl-mirroring` (on page 7) command-line parameter.


## Ligatures

Ligatures for the Latin scripts are disabled by default. To enable them, see the `-enable-latin-ligatures` (on page 5) command-line parameter.

Other non-Latin scripts have the ligatures enabled.

## Changing Labels Depending on Language

When developing a CSS that will apply to output localized for multiple languages, you should consider changing the static text (CSS generated) depending on the language.

 **Note:** Your document must use the `xml:lang` or `lang` attributes to specify the content language, ideally on the root element. The value must be specified using a language identifier (such as "en", "en-US", "en-CA", "fr", "fr-CA").

Consider a case where all the chapter titles are prefixed with the word "Chapter", followed by the figure counter. Depending on the language of the XML/HTML document, you need this word to change to: "Kapitel" for German, or "Chapitre" for French.

```
<div class='chp'>
  <h2>Introduction</h2>
  ...
</div>
```

The CSS may be written starting with a default rule that will be used when the content has languages other than the ones that are expected (for example, in English):

```
div.chp > h2:before{
  content: "Chapter " counter(chp);
}
```

Next, write rules for each of the specific languages:

```
div.chp > h2:lang(de):before{
  content: "Kapitel " counter(chp);
}
div.chp > h2:lang(fr):before{
  content: "Chapitre " counter(chp);
}
```

**i Tip:** A good practice is to keep all the static text for a specific language in a separate CSS.


To make the maintenance easier, you can separate the strings from the counter value by using one of the advanced features of Oxygen PDF Chemistry (the `:before` and `:after` pseudo elements) with **multiple levels** (*on page 86*). So you could write the default rule as:

```
div.chp > h2:before(2){
  content: "Chapter ";
}
div.chp > h2:before(1){
  content: counter(chp);
}
```

Now, the more specific rules are more simple:

```
div.chp > h2:lang(de):before(2){
  content: "Kapitel ";
}
div.chp > h2:lang(fr):before(2){
```

```
content: "Chapitre ";  
}
```

 **Note:** The multiple-level pseudo elements are non-standard features, and might not work if you switch to another processor.

# 7.

## Advanced Styling

### Hyphenation

The CSS `hyphens` property specifies how the words should be hyphenated when the paragraph text wraps on multiple lines.

The accepted values are:

#### **manual**

Words are only hyphenated when there are characters inside the word that explicitly suggest hyphenation opportunities. Those characters are:

#### **U+2010 (HYPHEN) -**


The "hard" hyphen character indicates a visible line break opportunity. The hyphen is always shown in the output.

#### **U+00AD (SHY)**

An invisible, "soft" hyphen. This character is not rendered visibly. It marks a breaking place for the word if hyphenation is necessary. You can use `&#xAD;` in XML or `&shy;` in HTML.

#### **auto**

Words are hyphenated automatically according to an algorithm driven by a hyphenation dictionary. Also, Oxygen PDF Chemistry uses an aggressive technique to create hyphenation points at underscores, dots, and case changes. This is useful when your publication contains snippets of code (Java, JS). To disable this functionality, you can use the `-no-aggressive-hyphenation` parameter ([on page 6](#)).

 **Note:** The element, or one of its parents, must have a `lang` or `xml:lang` attribute present for the processor to identify the hyphenation dictionary. If this is missing, the `manual` hyphenation is performed.

```
<table lang="en">
...
</table>
```

#### **none**

Currently not supported by Oxygen PDF Chemistry and it falls back to `manual`. If your document does not use the `HYPHEN` or `SHY` characters, no hyphenation is done.

### **Example: Hyphenation**

To perform hyphenation on all paragraphs from an HTML document, you can use:

```
p {
  hyphens: auto;
}
```

Usually, it is best to activate hyphenation for elements that are known to have a limited width (for example, on tables) where long words could bleed out of the bounds:

```
table {
  hyphens: auto;
}
```

### **Hyphenation Dictionaries**

Oxygen PDF Chemistry provides built-in hyphenation patterns for the following languages:

<b>Code</b>	<b>Language</b>
da	Danish
de	German
de_CH	German (Switzerland)
en	English
en-GB	English (Great Britain)
es	Spanish
fr	French
it	Italian
nb	Norwegian Bokmål
nl	Dutch
ro	Romanian
ru	Russian
sv	Swedish
th	Thai
pt	Portuguese
da	Danish

The built-in hyphenation pattern license terms are listed in the XML files in the `[CHEMISTRY_INSTALL_DIR]/config/hyph` folder. Most of them comply with the LaTeX distribution policy.

## Installing New Hyphenation Dictionaries

The hyphenation dictionaries are located in: `[CHEMISTRY_INSTALL_DIR]/config/hyph`. The `.xml` files allow you to read the licensing terms. The `.hyp` files are the compiled dictionaries that Oxygen PDF Chemistry actually uses.

Oxygen PDF Chemistry uses the TeX hyphenation dictionaries converted to XML by the `OFFO` project: <https://sourceforge.net/projects/offo/>.

One simple way to add more dictionaries:

1. Download and extract the `offo-hyphenation-compiled.zip` file.
2. Copy the `fop-hyph.jar` file to the `[CHEMISTRY_INSTALL_DIR]/lib` directory.
3. If you just need a single dictionary, extract the corresponding `.hyp` file from that jar file and place it in the `[CHEMISTRY_INSTALL_DIR]config/hyph` directory.

## How to Hide Hyphens

It is possible to hide hyphens for cases where they are not needed (for example, when hyphenation occurs in a section of code). To hide the hyphens, use the space character in the `-oxy-hyphenation-character` property:

```
pre {
  -oxy-hyphenation-character: " ";
}
```

## Using XPath in CSS

You can collect and process the document contents using XPath, directly from CSS. The following example counts the words from a section and shows it in a static text after the section:

```
section:after {
  content: "Number of words: "
    oxy_xpath("count(tokenize(normalize-space(string-join(text(), ' '), '
  ')))");
}
```

The following is an example of using the `oxy_xpath` function in a page rule property:

```
@page {
  @top-center{
    content: oxy_xpath('/book/title');
  }
}
```

All of the standard XPath 2.0 functions are supported.

#### Related Information:

- [Oxygen XML Editor User Guide: oxy\\_xpath\(\) Function](#)

## Multiple :before and :after Pseudo Elements

Although not standard, this extension may come in very handy if you want to style sections from static content. To add static content to an element, you would normally use a `:before` or `:after` pseudo element.


This example adds static text before the title ("Chapter 1", "Chapter 2", etc.):


```
h1:before {
  content: "Chapter " counter(chapter) ".";
  color: blue;
}
```

All of this is styled with the same color, blue in this example. Using standard CSS, it is impossible to style just the chapter number with a larger font and with red, for example. However, you can do it using multiple `before` pseudo elements:

```
h1:before(3) {
  content: "Chapter ";
  color: blue;
}
h1:before(2) {
  content: counter(chapter);
  color: red;
  font-size: large;
}
h1:before(1) {
  content: ".";
}
```

```
color: blue;  
}
```

 **Note:** The bigger the level, the more distant the pseudo element is.

 **Note:** Level 1 corresponds to normal `:before` or `:after`.

# 8.

## Tutorials and How-To

This sections contains a simple tutorial that shows you how to format various aspects of an HTML version of a book to publish it in PDF output. This is also where you can find some *how to* topics that offer guidance for achieving certain results.

### Example Tutorial

This tutorial will show you how to format a book. For simplicity, the tutorial will use an HTML version of the *The Adventures of Tom Sawyer* by *Mark Twain*, without many structures. It mainly consists of titles, paragraphs, and some meta-information. You can find the free ebook that is used for this tutorial at: <https://www.gutenberg.org> (search for **The Adventures of Tom Sawyer**).

Before getting started, save this file with the name "book.html" then create a CSS file named "book.css" in the same directory. Note that for the purposes of this tutorial, it is assumed that you are using *Oxygen XML Editor/Author* for your XML IDE.

#### 1. Clean up the existing styles

To make things easier, remove the `<style>` element from the header of the book. This will prevent mixing CSS rules coming from your CSS with the ones that were created for the browser display.

Also, remove the `style="width:100%;"` attribute from all the images in the document.

#### 2. Set up Chemistry in Oxygen

To transform this book to PDF, configure Oxygen PDF Chemistry as an external tool in *Oxygen XML Editor/Author*. Go to **Tools > External Tools > Configure** menu, click the **New** button, and configure it as a new external tool. Set the **Command line** to:

```
cmd /c "${oxygenInstallDir}\oxygenChemistry.bat" -catalogs ${xmlCatalogFilesList}
-in "${cf}" -css "${cfid}/${cfn}.css" -out "${cfid}/${cfn}.pdf" -show-pdf
```

**Result:** Every time you select the HTML book file in *Oxygen XML Editor/Author*, you can use this external tool from the toolbar.

### 3. Define the page size

To accommodate printing this book in a format similar to the original edition of the book, add the following to your CSS:

```
@page {
  size: 6in 7.5in;
  margin: 0.5in;
}
```

### 4. Select fonts

Good novel books usually have clean serif fonts. You can choose one from Google Fonts by adding the following import at the beginning of the CSS file:

```
@import url('https://fonts.googleapis.com/css?family=Crimson+Text');
```

Then, set it on the root of the document:

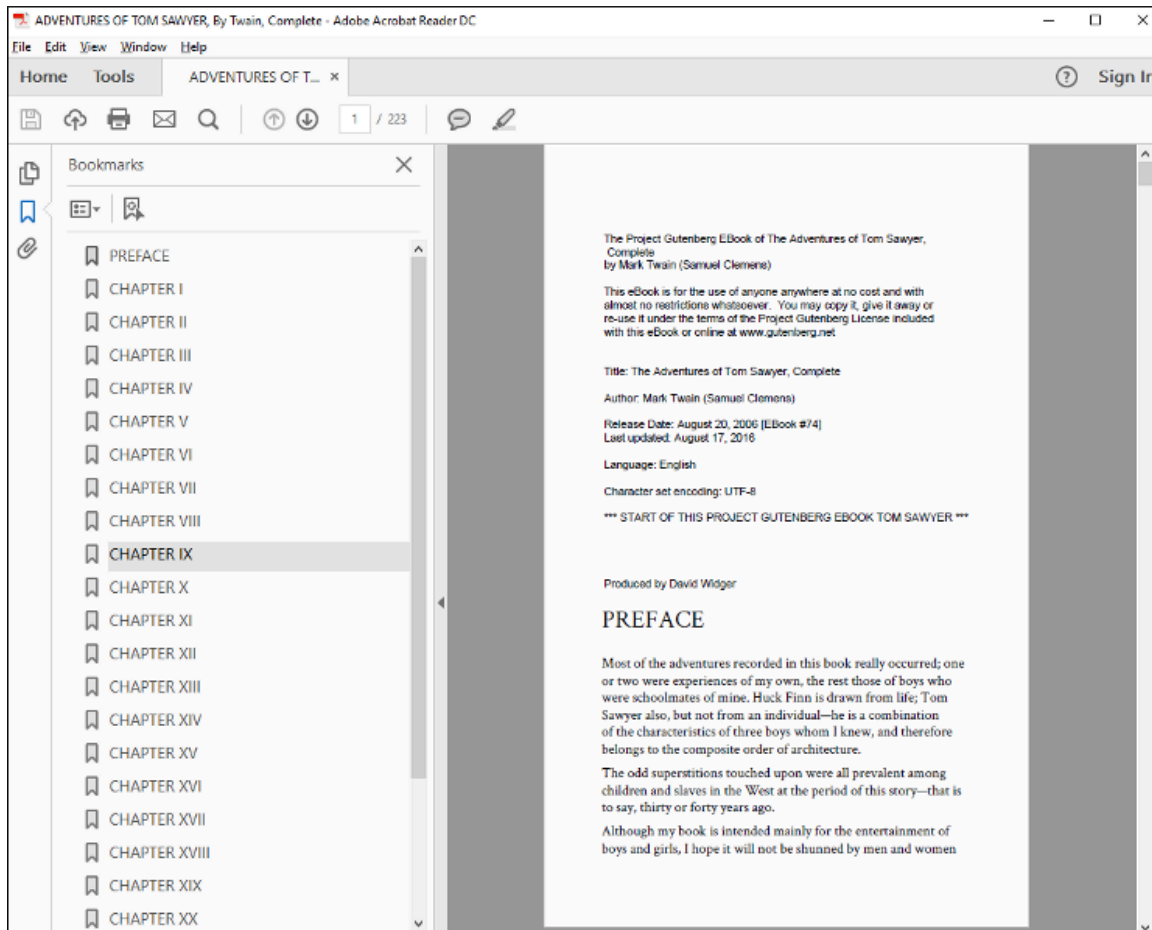
```
:root {
  font-family: 'Crimson Text', serif;
  font-size: 11pt;
}
```

Besides the main content, the HTML book contains some descriptions and licensing terms, located in `<pre>` elements, directly under the `<body>` element. For these, to wrap the content (the default is not to wrap), use a smaller font:

```
body > pre {
  font-family: sans-serif;
  font-size: 7pt;
  white-space: pre-wrap;
}
```

### 5. Transform the book with Chemistry

Try to [transform the HTML book with Chemistry \(on page 3\)](#). You will get something similar to this:



Notice that besides the formatting, Chemistry already helped with the following:

- Detected the publication title and set it as PDF. See the window title-bar.
- Created a tree of bookmarks by taking the `<H2>` elements into account.

For the purposes of this tutorial, the following still needs to be addressed:

- The preface, and each of the chapters should start on a new page.
- The chapter titles need to be formatted.
- The publication needs page numbers, page headers, and other styling.

## 6. Justify text

To improve the alignment of the right side of the book, justify the text by adding the following in the CSS:

```
p {
  text-align: justify;
}
```

## 7. Make chapters start on a new page

Currently, the document is very flat and the chapters are just marked by the `<H2>` title elements between `<p>` elements. There are also `<pre>` elements used for the copyright and licensing:

```
<h2>
    CHAPTER III
</h2>
...
<p>
    TOM presented himself before Aunt Polly, who was sitting by an open window
    in a pleasant rearward apartment, which was bedroom, breakfast-room,
```

To make each of the chapters start on a new page, the CSS paged media module defines a way to forcibly break the page before an element:

```
h2 {
    page-break-before:always;
}

body > pre {
    page-break-before:always;
}
```

**Result:** Now all the chapters start at the beginning of a new page, and the book is starting to look like a real publication. If you want the chapters to always start on a page from the right side, use the `right` value for the property.

## 8. Format the chapter titles

Currently, the titles look rather dull, aligned to the left. Center them and give them styling:

```
h2 {
    text-align: center;
    font-size: larger;
}
```

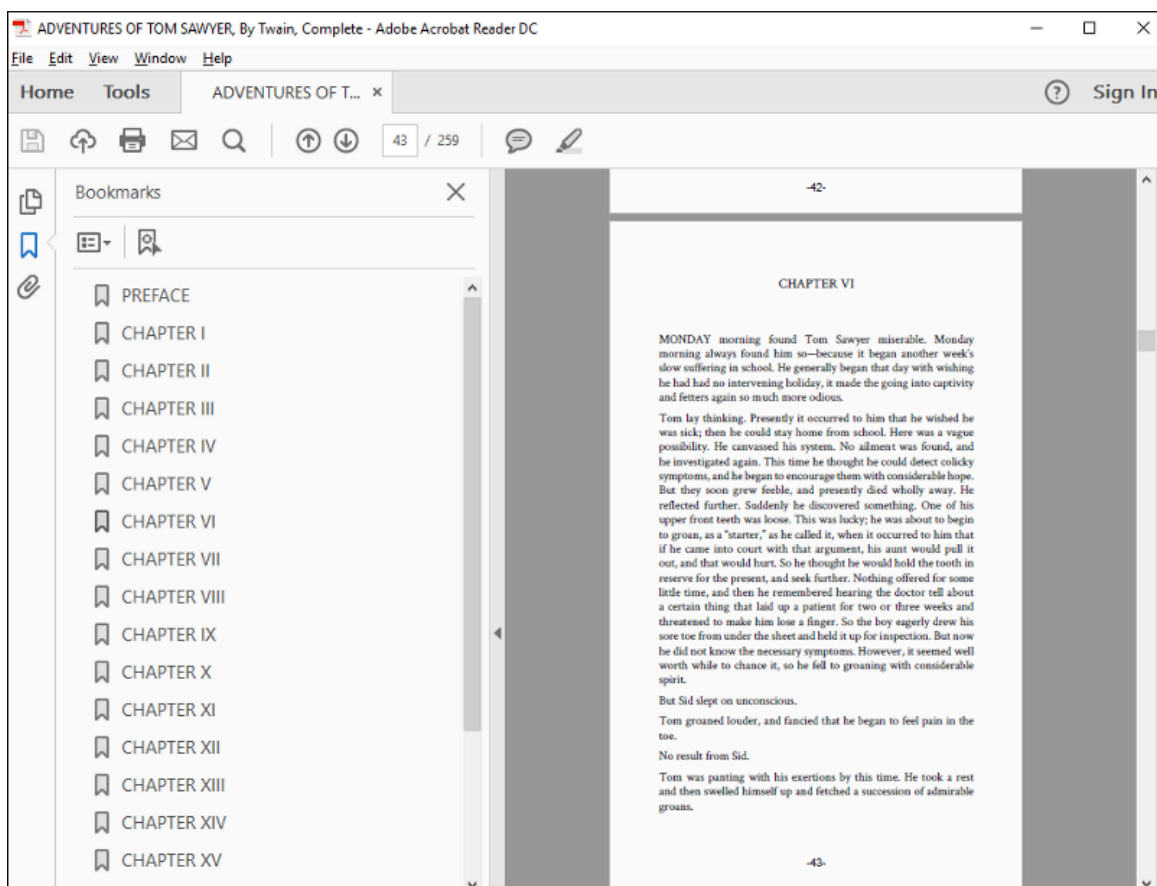
## 9. Add page numbers

Most novels have the page numbers shown in the bottom center of the page. To achieve this, you can use the `page` CSS counter set in a page margin box:

```
@page {
  @bottom-center {
    content: "-" counter(page) "-";
    font-size: 8pt;
  }
}
```

## What the document looks like so far

You have solved the text justification, page breaks, and page numbers. This is what the document looks like now:



Remaining things to be addressed:

- It needs a cover page.
- The page numbering should restart on the first chapter, after the preface, and should end before the licensing terms.

## 10. Add a cover page

You can find the original cover of the book on the same website of the Gutenberg project. For this tutorial, use this as artwork for the first page.

Start by defining a named page in your CSS file, with no page counter in the `bottom-center` region:

```
@page cover-page{
  background-image:url('https://www.gutenberg.org/files/74/74-h/images/
bookcover.jpg');
  background-size: 6in 7.5in;
  background-repeat:no-repeat;

  @bottom-center {
    content:none;
  }
}
```

When using images for your cover pages, make sure they respect the same aspect ratio as your page (width/height ratio), then use the `background-size` property to stretch it exactly to the page size.

Next, link this page to a synthetic element placed before the root. You can use a `:before` pseudo element in the `<html>` root element:

```
html:before{
  content:" ";
  page:cover-page;
}
```

You could place text over the cover image, but for the moment, just leave the content with blank text (a whitespace). It is necessary to have a `content` property that is not empty because Oxygen PDF Chemistry discards all the pseudo elements without one.

## 11. Reset and style page numbers

To make the page numbers be restarted at the beginning of the first chapter, you can use the first title that follows the metadata at the beginning of the document:

```
pre + h2{
  counter-reset: page 1;
}
```

The licensing terms at the end of the book can be numbered independently, and styled differently:

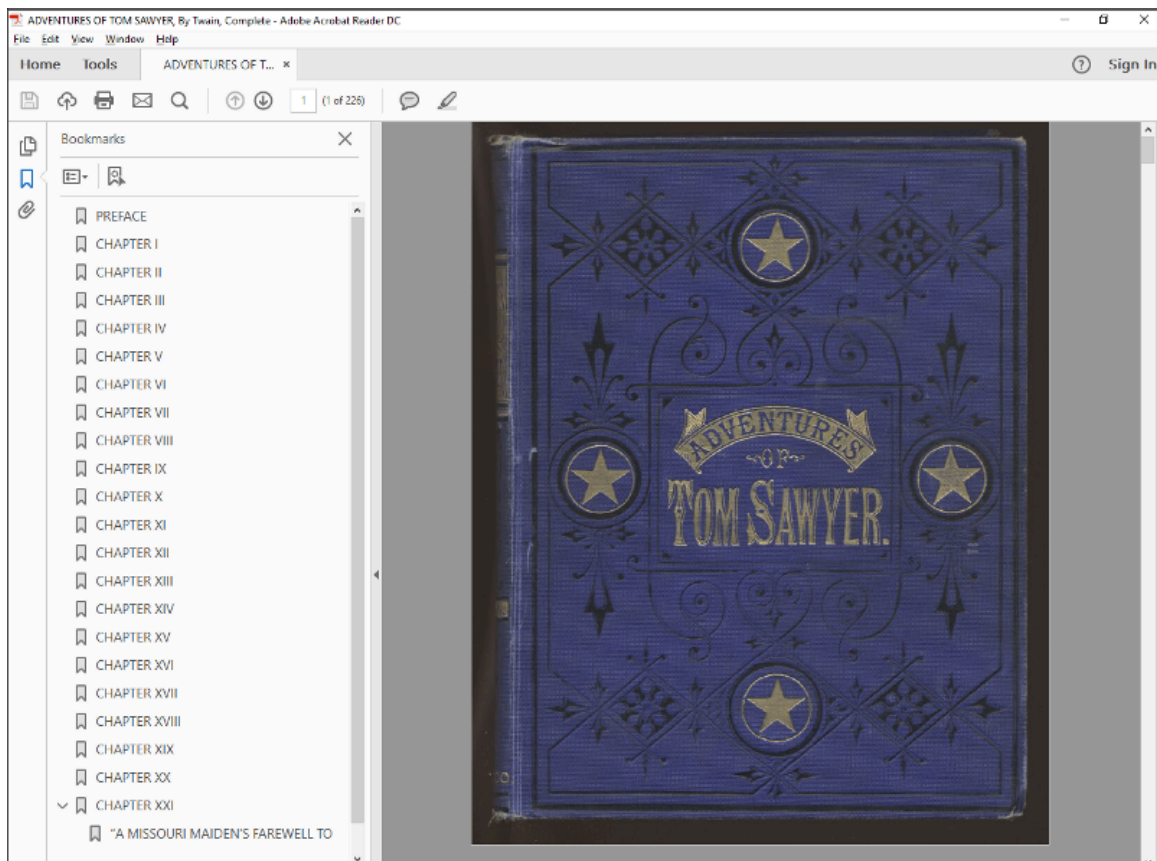
```
body > pre {  
  counter-reset: page;  
  page: copyright-license-page;  
}
```

For the copyright page, use a lower-roman numbering style:

```
@page copyright-license-page {  
  background-color: yellow;  
  @bottom-center {  
    content: counter(page, lower-roman);  
  }  
}
```

## Final Result

Now you have a nice looking book that can be distributed electronically, or printed:



## How To Apply Chemistry on Multiple Files

It is possible to use Oxygen PDF Chemistry to transform multiple files in a directory at once. This can be done by using a particular script, depending on your operating system.

The following examples assume that you installed Oxygen PDF Chemistry and modified the `PATH` environment variable to include the installation directory (*on page 2*). Also, you can change the `INPUT_DIR`, `CSS`, and `*xml` file pattern according to your needs.

### • Windows

1. Create a `.bat` file with the following content:

```
@echo off

rem This is the input directory
set "INPUT_DIR=C:\some\input\directory"
rem This is the CSS used to style the xml documents.
set "CSS=C:\some\css\file"

rem Iterate all the files (%f) that match the "*.xml" pattern in the input
directory.
for /f "usebackq delims=|" %%f in (`dir /b "%INPUT_DIR%\*.xml"`)
do chemistry.bat -in %INPUT_DIR%%f -css %CSS% -out %INPUT_DIR%%f.pdf
```

2. Run the `.bat` file from a command prompt.

### • Linux/Mac OS X

1. Create a script with the following content:

```
#!/bin/sh

# This is the input directory
INPUT_DIR=/some/input/directory"
# This is the CSS used to style the xml documents.
CSS=/some/css/file"

# Iterate all the files ($f) that match the "*.xml" pattern in the input
directory.
for f in $INPUT_DIR/*.xml; do chemistry.sh -in $f -out $f.pdf -css $CSS ; done
```

2. Run the script from the Terminal application.

## How To Insert the Current Date in the Cover Page

Oxygen PDF Chemistry provides a CSS extension, `oxy_xpath`, that can be used to execute XPath functions, including the `fn:current-date` function that simply returns the current date.

For example, if the text of the title is set in a `booktitle` element, you can use a synthetic block with the date content like this:

```
booktitle:after {  
  content:oxy_xpath("current-date()");  
  color:gray;  
  display:block;  
}
```

# 9.

## Appendix

### CSS Media Rules

Oxygen PDF Chemistry applies all the element CSS selectors that are:

- not enclosed in a `@media` rule.
- enclosed in a `@media` rule with type `print` or `oxygen-chemistry`.

It is recommended to enclose the rules that use *Oxygen* extensions in `@media oxygen-chemistry`, and the ones that apply only to print into a `@media print`.

```
chapter {  
    margin-left: 2em;  
}  
  
@media oxygen-chemistry {  
    chapter:before(2) {  
        content: counter(chapter);  
    }  
}  
  
@media print {  
    * {  
        color:black;  
    }  
}
```

### CSS Selectors

The following table summarizes the selectors supported by Oxygen PDF Chemistry:

Pattern	Meaning	Described in section	CSS level
*	Any element	Universal selector	2

Pattern	Meaning	Described in section	CSS level
E	An element of type E	Type element selector	1
E[foo]	An E element with the "foo" attribute set	Attribute selectors	2
E[foo="bar"]	An E element whose "foo" attribute value is exactly equal to "bar"	Attribute selectors	2
E[foo~="bar"]	An E element whose "foo" attribute value is a list of space-separated values, one of which is exactly equal to "bar"	Attribute selectors	2
E[foo^="bar"]	An E element whose "foo" attribute value begins exactly with the string "bar"	Attribute selectors	3
E[foo\$="bar"]	An E element whose "foo" attribute value ends exactly with the string "bar"	Attribute selectors	3
E[foo*="bar"]	An E element whose "foo" attribute value contains the substring "bar"	Attribute selectors	3
E[lang = "en"]	An E element whose "lang" attribute has a hyphen-separated list of values beginning (from the left) with "en"	Attribute selectors	2
E:root	An E element, root of the document	Structural pseudo-classes	3
E:first-child	An E element, first child of its parent	Structural pseudo-classes	2
E:last-child	An E element, last child of its parent	Structural pseudo-classes	3
E:first-of-type	An E element, first sibling of its type	Structural pseudo-classes	3
E:last-of-type	An E element, last sibling of its type	Structural pseudo-classes	3
E:only-of-type	An E element, only sibling of its type	Structural pseudo-classes	3
E:only-child	An E element, only child of its parent	Structural pseudo-classes	3
E:nth-child(n)	An E element, the n-th child of its parent	Structural pseudo-classes	3
E:nth-of-type(n)	An E element, the n-th sibling of its type	Structural pseudo-classes	3
E:empty	An E element that has no children (including text nodes)	Structural pseudo-classes	3
E:lang(c)	An element of type E in (human) language c (the document language specifies how language is determined)	The :lang() pseudo-class	2

Pattern	Meaning	Described in section	CSS level
E:has(F)	An E element with the condition F applying to one of its children. Similar to the subject selector described in: <a href="#">The subject element pseudo-class</a> .	Extension.	3
E:first-letter	The first formatted letter of an E element	<a href="#">The :first-letter pseudo element</a>	1
E:before	Generated content before an E element	<a href="#">The :before pseudo element</a>	2
E:after(n)	Generated content after an E element. Setting a value of 1 creates a normal :before. A higher value means the generated content is further from the element.	Extension	
E:after	Generated content after an E element	<a href="#">The :after pseudo element</a>	2
E:before(n)	Generated content before an E element. Setting a value of 1 creates a normal :before. A higher value means the generated content is further from the element.	Extension	
E:marker	Generated by list items to represent the item's marker (the bullet or number identifying each item). E is supposed to have <code>display list-item</code>	<a href="#">Marker pseudo element</a>	3
E#myid	An E element, its ID being equal to "myid"	<a href="#">ID selectors</a>	1
E F	An F element descendant of an E element	<a href="#">Descendant combinator</a>	1
E > F	An F element child of an element E	<a href="#">Child combinator</a>	2
E + F	An F element immediately preceded by an element E	<a href="#">Direct adjacent combinator</a>	2
E ~ F	An F element preceded by an element E	<a href="#">Indirect adjacent combinator</a>	3

## CSS Functions

Supported CSS functions include:

### **attr()**

It is used to retrieve the value of an attribute of the selected element and use it in the style sheet. The `attr()` function can be used with any CSS property. See more details: [here](#).

### **url()**

The `<url>` CSS data type denotes a pointer to a resource, such as an image or a font. It has no proper syntax and can only be expressed through the `url()` functional notation. URLs are used in numerous CSS properties, such as `background-image`, `cursor`, `@font-face`, and `list-style-image`.

### leader()

Used to fill a space with a pattern.

```
elem:before {
  content: "A" leader(".") "B";
  display: block;
}
```

The text "A" and "B" should be at the left and right sides of the page, with a line full of dots between them. See [Creating a Table of Contents \(TOC\) \(on page 35\)](#) for more examples.

### string()

Retrieves the value of a string-set. See [Headers and Footers \(on page 25\)](#) for use-cases.

### content()

Retrieves the text content from the current element. Used in a string-set definition. See [Headers and Footers \(on page 25\)](#) for use-cases.

### counter()

Retrieves the value of the innermost counter of that name on the element. <https://www.w3.org/TR/css-lists-3/#funcdef-counter>

### counters()

Retrieves all the values of a counter of that name on the element, separated by a delimiter. <https://www.w3.org/TR/css-lists-3/#funcdef-counters>

### target-counter()

The `target-counter()` function retrieves the value of the innermost counter with a given name. The required arguments are the url of the target and the name of the counter. An optional `counter-style` argument can be used to format the result. <https://www.w3.org/TR/css-gcpm-3/#target-counter>

### target-counters()

The `target-counters()` function fetches the value of all counters of a given name from the end of a link, and formats them by inserting a given string between the value of each nested counter. <https://www.w3.org/TR/css-gcpm-3/#funcdef-target-counters>

### oxy\_ ... ()

This is a collection of extension functions that are only recognized by *Oxygen* products. They can be used to process strings, do mathematical calculations, execute XPath queries over the document, and retrieve text from the document. See the full list: [here](#).

The following example counts the number of words from a paragraph (including [tracked changes](#)) and displays the result in front of it:

```
p:before{
  content:
    concat(" |Number of words:",
      oxy_xpath(
        "count(tokenize(normalize-space(string-join(text(), '')), ' '))",
        processChangeMarkers,
        true),
      " | ");
}
```

## CSS Units

Oxygen PDF Chemistry supports the following absolute CSS Units:

- **in** (inches) - 1in is equal to 2.54cm.
- **cm** (centimeters)
- **mm** (millimeters)
- **pt** (points) - One CSS point is equal to 1/72 of 1in.
- **pc** (picas) - 1pc is equal to 12pt.
- **px** (pixel units) - 1px is equal to 0.75pt.

The supported relative units are:

- **em** - Equal to the computed font-size property of the element on which it is used.
- **rem** - Equal to the computed font-size property of the document root element.

## CSS Properties

### Standard W3C CSS Properties

**Table 2. The following standard properties are supported by Oxygen PDF Chemistry**

Property	Supported Values	Unsupported Values	Notes
alignment-baseline	<b>middle</b>		
background			
background-color			
background-image			
background-position			
background-repeat			
background-size	<b>length   percent</b>		
bookmark-label			
bookmark-level			
bookmark-state			
border			
border-bottom			
border-bottom-color			
border-bottom-style			
border-bottom-width			
border-collapse	<b>separate   collapse</b>		
border-color			
border-left			
border-left-color			

Property	Supported Values	Unsupported Values	Notes
border-left-style			
border-left-width			
border-right			
border-right-color			
border-right-style			
border-right-width			
border-spacing	<b>&lt;length&gt;</b>		
border-style			
border-top			
border-top-color			
border-top-style			
border-top-width			
border-width			
bottom-property			
caption-side	<b>top bottom</b>	<b>left right</b>	
color			
column-span			
content			
counter-increment			
counter-reset			
direction			
display	<b>inline block inline-block table table-row table-cell table-caption table-row-group table-</b>		For the display:none, the elements are not eliminated from the DOM, are just collapsed. The text may remain in the document, but is not visible. This is needed for keeping the links working.

Property	Supported Values	Unsupported Values	Notes
	<b>header-group table-footer-group table-column  -oxy-morph</b>		For <b>-oxy-morph</b> , read more <a href="#">here</a> .
empty-cells			
float			
font			
font-family			
font-size			
font-src			
font-style			
font-variant	<b>small-caps normal</b>		Cannot be combined with the <b>:first-letter</b> pseudo element or <b>overflow-wrap: break-word</b> CSS property.
font-weight			
height			
image-resolution	<b>&lt;dpi&gt;</b>		
left-property			
letter-spacing			
line-height			
list-style			
list-style-image	<b>none url()</b>		
list-style-position			
list-style-type			
margin			
margin-bottom	<b>&lt;length&gt;</b>		
margin-left	<b>auto &lt;length&gt;</b>		
margin-right	<b>auto &lt;length&gt;</b>		
margin-top	<b>&lt;length&gt;</b>		
max-height	<b>&lt;length&gt;</b>		
max-width	<b>&lt;length&gt;</b>		
min-height	<b>&lt;length&gt;</b>		
min-width	<b>&lt;length&gt;</b>		Not supported for inline blocks and blocks.

Property	Supported Values	Unsupported Values	Notes
orphans	<b>&lt;number&gt;</b>		
outline			
outline-color			
outline-style			
outline-width			
overflow-wrap	<b>break-word</b>		It works by adding soft hyphens (invisible hyphens) in words with more than 3 characters, so it might behave slightly different from the specification. The soft hyphen specifies a place in the text where a hyphenated break is allowed. It becomes visible only after word wrapping at the end of a line.
padding			
padding-bottom			
padding-left			
padding-right			
padding-top			
page			
page-break-after			
page-break-before			
page-break-inside			
position	<b>fixed absolute</b>	<b>relative</b>	The absolute positioning is done relative to the direct parent. This is a limitation - CSS specifies that you can mark a parent block with position:relative, and absolute children will be placed relative to this.
right-property			
size			
string-set	<b>content() counter()</b>		
table-column-span			
table-layout	<b>fixed auto</b>		

Property	Supported Values	Unsupported Values	Notes
table-row-span			
text-align	<b>start end left right center justify justify-all</b>		
text-decoration			
text-decoration-color			
text-decoration-line			
text-decoration-style			
text-indent			
text-transform			
top-property			
transform			
transform-rotation			
unicode-bidi			
vertical-align	<b>baseline   sub   super   text-top   text-bottom   middle   top   bottom</b>	<b>&lt;length&gt; &lt;percentage&gt;</b>	
visibility			
white-space	<b>normal   nowrap   pre   pre-wrap</b>		
widows	<b>&lt;number&gt;</b>		
width			

## Extension CSS Properties

**Table 3. The following extension properties are supported by Oxygen PDF Chemistry**

Property	Description	Supported Values
-oxy-alt-text	Used to specify an alternative description for the element that is used by the PDF readers.	<b>string</b>
	<pre>image {</pre>	

Property	Description	Supported Values
	<pre>-oxy-alt-text: "Image about: " attr(href); }</pre>	
-oxy-hyphenation-character	Used to specify the Unicode character to be presented when a hyphenation break occurs. To hide hyphens, use the space character.	<character> inherit
	<pre>pre {   -oxy-hyphenation-character: " "; }</pre>	
-oxy-hyphenation-push-character-count	Used to specify the minimum number of characters in a hyphenated word after the hyphenation character (the minimum number of characters in the word pushed to the next line).	<number> inherit
	<pre>p {   -oxy-hyphenation-push-character-count: "2"; }</pre>	
-oxy-hyphenation-remain-character-count	Used to specify the minimum number of characters in a hyphenated word before the hyphenation character (the minimum number of characters in the word left on the line ending with the hyphenation character).	<number> inherit
	<pre>p {   -oxy-hyphenation-remain-character-count: "2"; }</pre>	
-oxy-initial-page-number	Used to set the initial page number to be used on this particular page sequence.	auto auto-odd auto-even <number> inherit
	<pre>@page chapter-page {   -oxy-initial-page-number: auto-odd; }</pre>	
-oxy-force-page-count	Used to impose a constraint on the number of pages in a page sequence. If this constraint is not satisfied, an additional page will be added to the end of the sequence. This page becomes the "last" page of that sequence.	auto evenodd end-on-even end-on-odd no-force inherit
	<pre>@page chapter-page {   -oxy-initial-page-number: auto-odd; }</pre>	
-oxy-link	Used to create links in the PDF output.	none url()

Property	Description	Supported Values
	<pre>link {   -oxy-link: attr(href); }</pre>	
-oxy-page-group	<p>Using the <b>start</b> value forces the creation of a new page group (sequence) even if the page before the element has the same name. The <b>auto</b> value uses the W3C algorithm normally, which would merge the current element with the opened page sequence.</p> <pre>chapter {   -oxy-page-group: start;   page: chapter; }</pre>	<b>start auto</b>
-oxy-pdf-meta-author	<p>Used to forward meta-information to the PDF. Represents the author of the publication.</p> <pre>meta[name='author'] {   -oxy-pdf-meta-author: attr(value); }</pre>	<b>string</b>
-oxy-pdf-meta-custom	<p>Used to forward meta-information to the PDF. Represents a generic custom document property. Should have two strings, the name and the value.</p> <pre>metadata {   -oxy-pdf-meta-custom: attr(name) attr(value); }</pre>	<b>string</b>
-oxy-pdf-meta-description	<p>Used to forward meta-information to the PDF. Represents the description of the publication.</p> <pre>meta[name='description'] {   -oxy-pdf-meta-description: attr(value); }</pre>	<b>string</b>
-oxy-pdf-meta-keyword	<p>Used to forward meta-information to the PDF. Represents a single keyword from the publication. The processor should aggregate all keyword definitions and separate them by comma.</p> <pre>keyword {   -oxy-pdf-meta-keyword: content(text); }</pre>	<b>string</b>

Property	Description	Supported Values
-oxy-pdf-meta-keywords	Used to forward meta-information to the PDF. Represents the keywords of the publication. The value should contain the keywords separated by commas. <pre>meta[name='keywords'] {   -oxy-pdf-meta-keywords: attr(value); }</pre>	<b>string</b>
-oxy-pdf-meta-title	Used to forward meta-information to the PDF. Represents the title of the publication. <pre>meta[name='title'] {   -oxy-pdf-meta-title: attr(value); }</pre>	<b>string</b>
-oxy-pdf-tag-type	Maps an element to a PDF accessibility tag. <pre>section {   -oxy-pdf-tag-type: "Sect"; }</pre>	<b>string</b>
-oxy-style	An Oxygen extension used to define additional styles for an element (for example, collected from an attribute). <pre>div {   font-weight:bold;   color:red;   -oxy-style:attr(style); }</pre> <p>The value of the style attribute from the div element is parsed as a collection of CSS properties and applied over the current element styles. For instance, if the style attribute has the value <code>color:green; border: 1pt solid red</code>, it will combine with the existing properties, resulting in:</p> <pre>font-weight:bold; color:green; border: 1pt solid red</pre>	<b>CSS fragment</b>
-oxy-simulate-style	Set this flag to <b>yes</b> when the styles (bold, italic, or both) need to be simulated from a regular font. See: <a href="#">Using Simulated (Synthetic) Styles (on page 76)</a> .	<b>yes no</b>

Property	Description	Supported Values
-oxy-column-break-inside	This can be used to avoid column breaks inside the matching element. Use with care, if the element has large content, it may bleed out of the page.	<b>always auto</b>
-oxy-column-break-before	This can be used to either force or eliminate a column break between the matching element and the previous sibling. Supported values are: <code>always</code> - to force the column break, or <code>avoid</code> - to keep the end of the previous sibling together with the beginning of the matching element in the same column.	<b>always auto avoid</b>
-oxy-column-break-after	This can be used to either force or eliminate a column break between the matching element and the next sibling. Supported values are: <code>always</code> - to force the column break, or <code>avoid</code> - to keep the end of the matching element together with the next sibling beginning in the same column.	<b>always auto avoid</b>

## XML Support

### Supported Schema Languages

Although it does not validate the input, Oxygen PDF Chemistry uses information from the schema associated with the document. The supported schema languages are:

- XML Schema
- DTD
- Relax NG

### Entities

All XML entities are expanded prior to processing.

### xi:include

All the *xi:included* fragments are expanded prior to processing.


### Default Attributes

The default and fixed attributes defined in the associated schema are taken into account when the CSS rules are applied.

For instance, the DITA `class` attributes are defined in the DTD rather than the XML instance (the CSS rules usually match this attribute, not the element names).

## Configuration File

Oxygen PDF Chemistry uses the XSL-FO FOP processor as a last processing stage. You can control FOP by changing the `config/chemistry-fop.xconf` configuration file.

 **Note:** This file is a template that Oxygen PDF Chemistry uses to fill it with information about fonts. If you change it, make sure you preserve the font-related variables that are automatically expanded, or at least make sure you define the set of fonts that are used from the CSS.

Related Information:

- [Apache FOP: Configuration](#)

## Technical Issues

Information about error messages and known issues for Chemistry.

This topic contains information about error messages, possible known issues, and other things that you might want to be aware of regarding Oxygen PDF Chemistry.

### Out of Memory Error

If you receive an error message indicating an *Out of Memory Error*, you can try the following:

- Make sure you are using Java 1.8.
- Use the `-Xmx<NNN>m` command-line parameter (*on page 7*) to specify a bigger value for the maximum allocated memory. If you are running Java on 64-bit, you can specify a larger memory size (for example, `-Xmx2048m`).

# Index

## A

Accessibility (508 compliance)

48

Advanced styling

83

after pseudo element

86

Aligning blocks

60

Anchors

47

Associating a CSS to a document

13

## B

before pseudo element

86

Bookmarks

43

## C

Command line arguments

3

Configuration file

111

Configuring PDF Chemistry as an external tool

3

Cross references

34

CSS functions

99

CSS media rules

97

CSS properties

101

CSS selectors

97

CSS units

101

## D

Debugging

9

DITA to PDF from a command line

7

## E

Element layout

51

Embedding fonts

72

Ending chapters on odd/even page

23

Error messages

111

## F

Floats

60

Fonts

70

Footer

Dynamic images

31

Footers

25

Footnotes

32

## G

Getting started

1

Graphics

62

## H

Header

Dynamic images

31

Style text

29

Headers

25

Hyphenation

83

**I**

Images  
62  
Insert current date  
95  
Installing PDF Chemistry

2

Internationalization

79

Introduction

1

**L**

Ligatures

80

Lists

57

Localization

80

**M**

MathML

68

Metadata

45

Multiple lines in footer

28

Multiple lines in header

28

**N**

Named destinations

47

**O**

OutOfMemory error

111

**P**

Page breaks

20

Page formatting

13

Processing PDF Chemistry from command  
line

3

**R**

Right to left languages

79

Rotating blocks

60

**S**

Starting chapters on odd/even page

23

Styling for print

13

Styling the first page of a chapter

23

SVG

67

**T**

Table of contents creation

35

Tables

51

Technical issues

111

Transform multiple files

94

Tutorial

88

**W**

Web fonts

70

What is Chemistry

1

**X**

XML support

110

XPath in CSS

85

# Copyright

Oxygen PDF Chemistry User Manual

Syncro Soft SRL.

Copyright © 2002-2019 Syncro Soft SRL. All Rights Reserved.

**All rights reserved.** No parts of this work may be reproduced in any form or by any means - graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems - without the written permission of the publisher. Products that are referred to in this document may be either trademarks and/or registered trademarks of the respective owners. The publisher and the author make no claim to these trademarks.

**Trademarks.** Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this document, and Syncro Soft SRL was aware of a trademark claim, the designations have been rendered in caps or initial caps.

**Notice.** While every precaution has been taken in the preparation of this document, the publisher and the author assume no responsibility for errors or omissions, or for damages resulting from the use of information contained in this document or from the use of programs and source code that may accompany it. In no event shall the publisher and the author be liable for any loss of profit or any other commercial damage caused or alleged to have been caused directly or indirectly by this document.

**Link disclaimer.** Syncro Soft SRL is not responsible for the contents or reliability of any linked Websites referenced elsewhere within this documentation, and Syncro Soft SRL does not necessarily endorse the products, services, or information described or offered within them. Syncro Soft cannot guarantee that these links will work all the time and has no control over the availability of the linked pages.

**Warranty.** Syncro Soft SRL provides a limited warranty on this product. Refer to your sales agreement to establish the terms of the limited warranty. In addition, Oxygen PDF Chemistry End-User License Agreement, as well as information regarding support for this product, while under warranty, is available through the [Oxygen PDF Chemistry End-User License Agreement](#).

**Terms and conditions.** For the terms and conditions for using Oxygen PDF Chemistry, see [Oxygen PDF Chemistry End-User License Agreement](#).

**Documentation.** For the most current versions of documentation, see the [Oxygen PDF Chemistry User Manual](#).

**Contact Syncro Soft SRL.** Syncro Soft SRL provides telephone numbers and e-mail addresses for you to report problems or to ask questions about your product, see the [Oxygen support webpage](#).