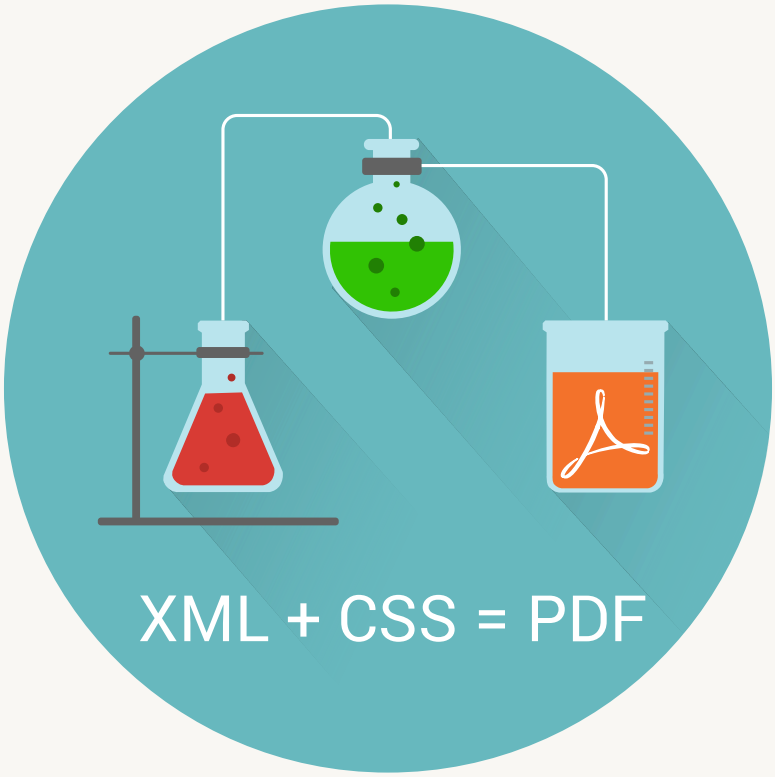
Oxygen PDF Chemistry 28.0
User Guide



Contents

Chapter 1. Getting Started **6**
Installing 7
Configuring Chemistry as an Oxygen External Tool 8
Command Line Interface 8
Generating PDF Output from DITA Content Using a Command Line 15
Environment Variables 15
Your First Document 16
Debugging the CSS 17
Logging 20
Security Policy Configuration 21
Chapter 2. Styling for Print **23**
Associating a CSS to a Document 23
Page Formatting 23
The @Page Rule 23
Columns 31
Controlling Page Breaks 32
Forcing Page Breaks 32
Avoiding Page Breaks 32
Page Breaks Between Named Pages 33
Page Breaks Between Lines: Widows and Orphans 34
Chapter Page Placement and Styling 34
Headers and Footers 36
Extracting Text from Document Using String Sets 36
Extracting Text from Document Using XPath 37
Multiple Lines in Headers and Footers 39
The Page Counter 39
How to Use Flexible Layout in Headers and Footers 40
How to Style a Part of the Text from the Header 40
How to Use Dynamic Images in Headers and Footers 41
How to Add a Link in Headers and Footers 42
Footnotes 43

How to Add a Separator Above the Footnotes....45
Cross References....45
Internal links....45
External links....46
Creating a Table of Contents (TOC)....47
Annotations....49
Comments and Tracked Changes - HTML Fragment....49
PDF Output....51
Bookmarks....51
Metadata....53
Named Destinations (Anchors)....56
Accessibility (508 Compliance)....56
Archiving....59
Signing....59
Hyphenation....60
Hyphenation Dictionaries....61
Installing New Hyphenation Dictionaries....62
How to Alter a Hyphenation Dictionary....62
How to Disable Hyphenation for a Word....63
How to Hide Hyphens....64
How to Force or Avoid Line Breaks at Hyphens....64
Using XPath in CSS....65
Using the :before(n) and :after(n) CSS Pseudo-Elements....66
Change Bars....67
Chapter 3. Layout....69
Tables....69
Lists....76
Aligning Blocks Horizontally....79
Rotating Blocks....80
Chapter 4. Graphics....82
Images....82
Supported Image Types....82
Setting Image Width and Height....83

Image Resolution....84
Background Images....85
Foreground Images....86
Supported Colors....87
SVG....87
SVG Templates....89
MathML....92
Image Maps (HTML)....93
Videos....94
Chapter 5. Fonts....97
Supported Fonts....97
Basic Fonts....97
Fallback Fonts....97
Using Web Fonts....98
Using Installed Fonts....98
Using Local Font Files....99
Font Embedding....100
Font Ligatures....104
Font Alternates....104
Using Simulated (Synthetic) Styles....105
Chapter 6. Localization....108
Chapter 7. Tutorials....111
How To Insert the Current Date in the Cover Page....111
How To Apply Chemistry on Multiple Files....111
Full Example: How to Format a Book....112
Chapter 8. Appendix....119
CSS Media Rules....119
CSS Selectors....119
CSS Supported Selectors....120
CSS Pseudo-Classes and Pseudo-Elements....120
CSS Combinators....122
CSS Functions....122
CSS Units....124

CSS Properties..... 125
Standard W3C CSS Properties..... 125
Extension CSS Properties..... 134
XML Support..... 145
Configuration File..... 145
Technical Issues..... 145
Index..... a
Copyright..... d

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.

Why Use CSS Instead of XSL-FO?

There is not one simple answer. It really depends on the needed output, but both methods will generate PDF documents without any problem. So why use CSS?

- CSS is a much easier language to learn and master than XSLT.
- More people know CSS rather than XSLT (and there are more CSS tutorials available than XSLT tutorials).
- The CSS debugging can be done directly from any browser or from **Author** mode/
- In most cases, the CSS customization will cover all of the user's needs.
- The CSS customization can be reused in any HTML output.
- The cascading priority scheme makes CSS easier to extend (no need to know previous rules to create new ones).
- PDF processors usually proposes both solutions (Oxygen PDF Chemistry, Antenna House, Prince XML).

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

Prerequisite:
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 21.
- **macOS** - Java JDK version 21.

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.
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
   ```

   macOS or Linux:

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

   Result: It should display the version.

Related Information:
How do I set system environment variables in Windows
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 open XML file, follow this procedure:

1. In *Oxygen XML Editor/Author*, select **Tools > External Tools > Configure**.
2. Click 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 7)*):
   - **Windows:** `cmd /C chemistry.bat -in "${cf}" -css "my_stylesheet.css" -out "${cfd}/${cfn}.pdf" -show-pdf`
   - **macOS or Linux:** `sh chemistry.sh -in "${cf}" -css "my_stylesheet.css" -out "${cfd}/${cfn}.pdf" -show-pdf`

   Note:
   If you have not 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`
   - **macOS 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 can 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 16)* 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).

-compress-pdf

Compresses the created PDF to reduce its size. This may increase processing time.

-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.

-drop-block-margins-at-page-boundary

Specifies that the top and bottom margins associated with a block element should be discarded when the block is at the top or bottom of the page. Allowed values:

- **yes** (default)
- **no**

-enable-latin-glyph-substitutions

When set to **yes** (default), glyph substitution is enabled (if the particular font supports it). This applies to Latin-based scripts only (the substitutions are always enabled in other types of scripts). If you encounter problems rendering or copying accented glyphs (i.e. *umlauts* or other *diacritics*), it might be helpful to set this parameter to **no** to disable the font glyph substitutions. Another example of a case when you might need to disable the substitutions is a situation where an accented character cannot be mapped to a compound glyph, resulting in the glyph not being rendered in the PDF output.

Warnings:

- Disabling substitutions also disables Latin ligatures.
- Disabling substitutions is not recommended unless absolutely necessary. It is better practice to use another font if you can find one that does not have the rendering issues.

-enable-latin-ligatures

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

> **Note:**
> This parameter is deprecated. Use the `font-variant-ligatures` CSS property instead.
> See Font Ligatures *(on page 104)* for more information.

-expand-xpath-in-svg-templates

When set to yes (default), each XPath expression that has a format of `${expression}` inside SVG templates files (with a `.template.svg` file extension) will be expanded.

-fonts-dir

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

-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
```

-hyph-dir

The directory that contains additional hyphenation dictionaries. The XML files should be placed directly in it (not a subdirectory) and they should be named using the language code (e.g. `en.xml`). For more information about adding or altering dictionaries, see: Hyphenation *(on page 60)*.

-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 84)*.

```
-image-resolution 72
```

-insert-after

A semicolon-separated list of PDF files, specified using URI or file syntax, to be inserted after the generated PDF.

-insert-before

A semicolon-separated list of PDF files, specified using URI or file syntax, to be inserted before the generated PDF.

-licensekey-dir

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

-log-file

Specifies the path to a custom `logback.xml` configuration file. See: Logging *(on page 20)*.

-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. This parameter is ignored when the `-security-policy` file is specified.

-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-a

Produces output that conforms to the PDF/A archiving standards. The conversion will fail if fonts are not embedded. See: Archiving *(on page 59)*.

-pdf-security-encrypt-metadata

Encrypts the metadata. By default, it is active when other security parameters are set.

-pdf-security-encrypt-structure-element

Encrypts the structure elements. By default, inactive when other security parameters are set.

-pdf-security-restrict-accesscontent

Restricts extracting text and graphics. Used for protecting the PDF Document. The restriction is off by default.

-pdf-security-restrict-annotations

Restricts filling in existing interactive forms. Used for protecting the PDF Document. The restriction is off by default.

-pdf-security-restrict-assembledoc

Restricts assembling document (e.g. adding pages). Used for protecting the PDF Document. The restriction is off by default.

-pdf-security-restrict-copy

Restricts copying content. Used for protecting the PDF Document. The restriction is off by default.

-pdf-security-restrict-edit

Restricts copying content. Used for protecting the PDF Document. The restriction is off by default.

-pdf-security-restrict-fillinforms

Restricts filling in existing interactive forms. Used for protecting the PDF Document. The restriction is off by default.

-pdf-security-restrict-print

Restricts printing. Used for protecting the PDF Document. The restriction is off by default.

-pdf-security-restrict-printhq

Restricts high quality printing. Used for protecting the PDF Document. The restriction is off by default.

-pdf-security-user-password

The user password. The document can be opened using either this password or the owner password. When the owner password is specified, the document can be opened by using the user password but with possible other restrictions.

-pdf-security-owner-password

The owner password (if this is provided when the document is opened in the reader application, with no restrictions).

-pdf-ua

Produces output that conforms to the PDF/UA-1 accessibility standards. The conversion will fail if fonts are not embedded. See: Accessibility (508 Compliance) *(on page 56)*.

-pdf-version

Use this parameter to specify the version of the produced PDF. It has no impact on the set of PDF features used by the engine, but may be used to signal a compatibility level to the PDF readers. The default is **1.5**.

-security-policy

Specifies the Java security policy file (in URL or file path syntax). This policy file can be used to restrict the processor access to certain resources (e.g. files, network). For more details, see: Security Policy Configuration *(on page 21)*.

-security-resources-dir1

-security-resources-dir2

-security-resources-dir3

-security-resources-dir4

-security-resources-dir5

Specifies additional directories (to the input folder) where resources (e.g. images, CSS stylesheets, etc.) are stored. These parameters are automatically expanded in the Java security policy file, with read access, when the `-security-policy` is specified. For more details, see: Security Policy Configuration *(on page 21)*.

-security-resources-host

Specifies the host where resources (e.g. images, CSS stylesheets, etc.) are stored. This parameter is automatically expanded in the Java security policy file, with connect access, when the `-security-policy` was specified. For more details, see: Security Policy Configuration *(on page 21)*.

-security-workspace

Specifies the directory where the temporary files and font cache is stored. This parameter is automatically expanded in the Java security policy file, with read and write access, when the `-security-policy` is specified. For more details, see: Security Policy Configuration *(on page 21)*.

-keystore-file

The keystore file used to sign the PDF. This parameter must be set if you want to sign your PDF. See: Signing *(on page 59)*.

-sign-name

The name of the person/company who certified the PDF.

-sign-location

The location of the person/company who certified the PDF.

-sign-reason

The reason for certifying the PDF.

-sign-password

The export password of the keystore used to sign the PDF.

-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-changed-text-in-pdf-sticky-notes-content

When set to **yes** (default), the inserted and deleted text is shown in the PDF annotations. When set to **no**, only the *inserted* and *deleted* labels are shown in the PDF annotations.

-show-image-map-area-numbers

When set to **yes**, a counter for each area from the image map will be displayed over the image, near the defined shape. The default is **no**.

-show-image-map-area-shapes

When set to **yes**, each of the image map area shapes will be displayed with a translucent fill over the image. You can use this to debug your image maps. The default is **no**. For customizing the aspect of the shapes, see Image Maps (HTML) *(on page 93)*.

-show-pdf

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

-use-css-for-embedded-svg

When set to **yes** (default), the CSS files specified by the `-css` parameter are also applied on embedded SVG elements. Allowed values are **yes** and **no**.

-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
```

Notes:

- The default is 512MB.
- 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.

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

Generating PDF Output from DITA Content Using a Command Line

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

1. Download and install the **Oxygen Publishing Engine**. This bundles a DITA Open Toolkit (http://www.dita-ot.org/download) with all the required plugins to generate PDF output, including the Oxygen PDF Chemistry processor.
2. Place the license key for the **Oxygen Publishing Engine** in a `licensekey.txt` file in the installation folder.
3. Run the publishing engine from a command line and make sure you specify the format to be **pdf-css-html5** and the path to your main DITA map. Specify additional parameters (such as `args.css`) if you want to customize the output. See the documentation for more information about customizing DITA output and accepted parameters.
   For example:
   ```
   oxygen-publishing-engine-dir/bin/dita --input=path/to/my.ditamap --format=pdf-css-html5 --output=path/to/output/folder -Dargs.css=/path/to/my.css
   ```

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 15)* 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 9)* command-line parameter together with this environment variable.

Your First Document

Prerequisite:
Make sure you followed the procedure in the Installing *(on page 7)* 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.

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.

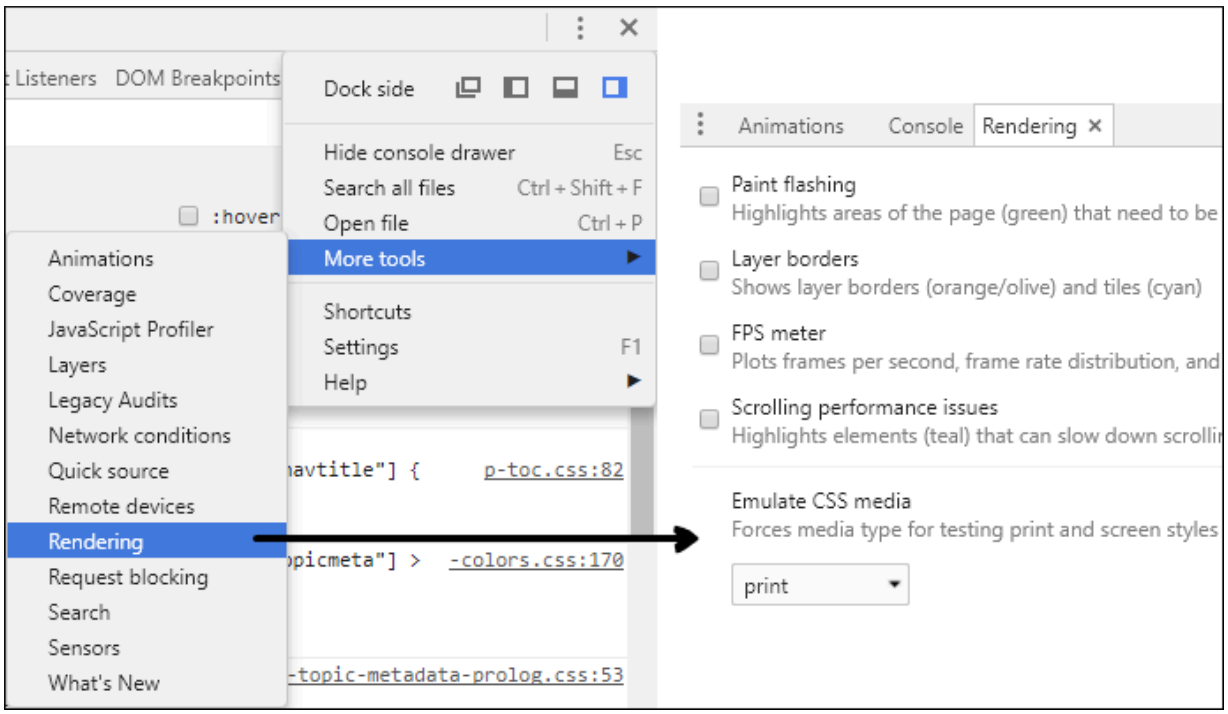
Comparing the CSS Against the Input File

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 a Browser

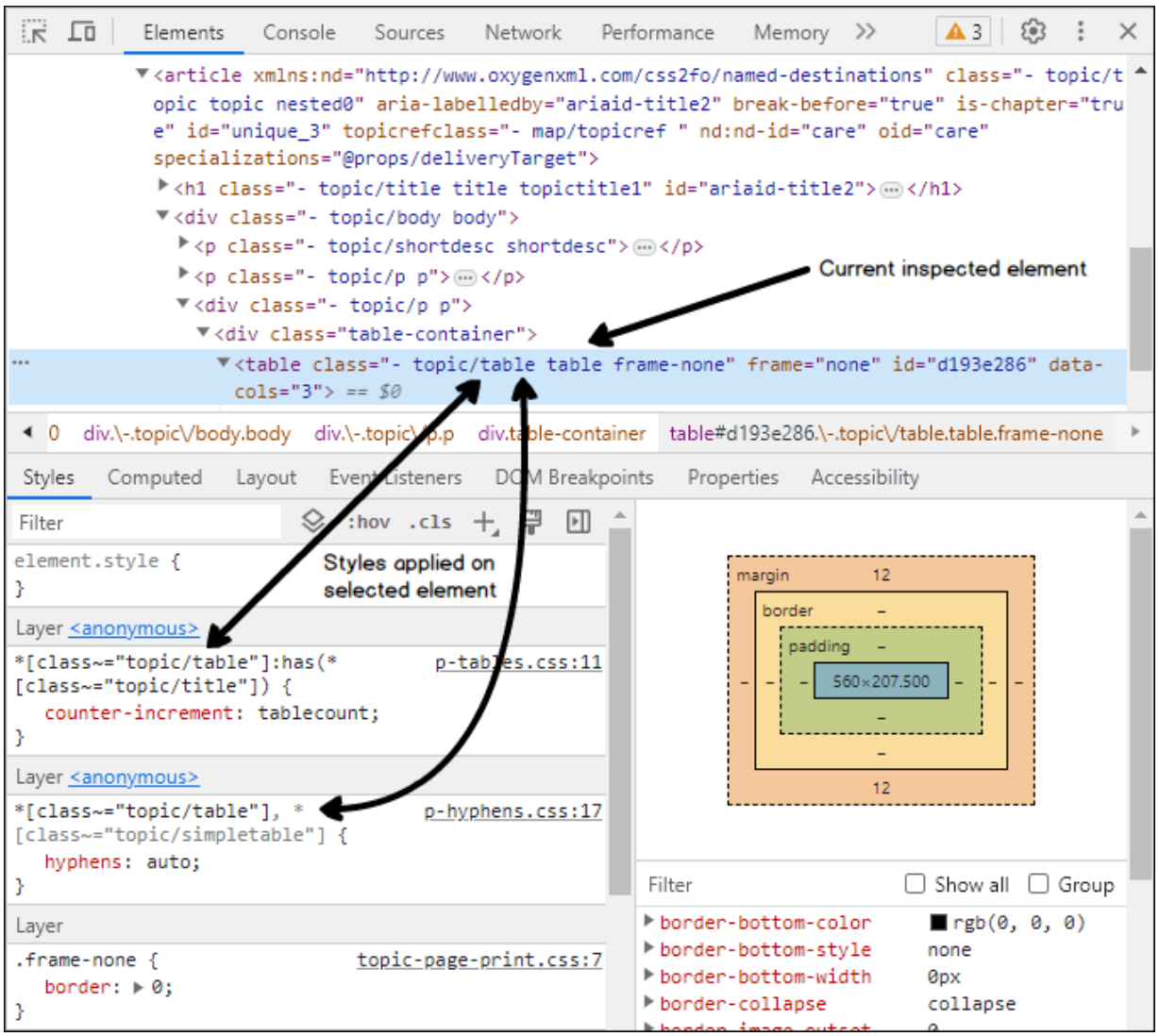
The following procedure explains how to inspect the applied CSS styles using Chrome, but any modern browser can be used and the procedure for each of them is similar:

1. Open the XML or HTML file.
2. Open the **Chrome Developer Tools** by using **> More Tools > Developer Tools** (or press **CTRL+SHIFT +I**).
3. Activate the **Rendering** pane by using **> More Tools > Rendering** then 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, the table of contents, and the index, but not the styles of the page margin boxes (headers, footers) or page breaks.

4. Right-click on the element you want to inspect and select the **Inspect** action. You will see the element (in the **Elements** pane) and the list of styles that are applied on it (in the **Styles** pane):



Tip:
Clicking any of the stylesheet links from the **Styles** pane opens the original CSS files in the **Sources** pane. Editing the rules in that pane results in a live preview of how the change will affect the output (these modifications will be lost on reload).

Inspecting the Applied Styles Using Oxygen XML Editor/Author

To inspect the applied CSS styles using **Oxygen**:

1. In **Oxygen XML Editor/Author**, open the XML or HTML file.
2. From the **Styles** toolbar, choose the **+ Print Ready** entry. This will activate certain CSS selectors enclosed in `@media print {..}`.

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

3. Right-click on the element you want to inspect and select the **Inspect Styles** action. The dedicated **CSS Inspector** view will be opened and it will show the applied CSS rules.

> **Tip:**
> With this file open 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.

Other Debugging Techniques

Here are some other debugging techniques you may find useful:

- Add background and border properties to the specific CSS rule. If they do not appear in the output, then there is a problem with the rule selector.
- Add the `!important` keyword to a property that is not applied, or make the selector more specific (by adding more parent selectors).
- Add the following fragment in your customization CSS to show how the elements are mapped to PDF:

```
* {
   border: 1pt solid blue !important;
}

*:before(1000) {
   content:  oxy_name() !important;
   color: orange;
}

*:before(900) {
   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 are hidden.

Logging

By default, when running the product, the default log root level in INFO so only INFO, WARN and ERROR message are displayed. You can change the logger configuration by following this procedure:

1. Create a custom `logback.xml` file in which, for example, you can filter the inner Apache FOP messages:

```
<configuration>

  <conversionRule conversionWord="chmsg"
    converterClass="com.oxygenxml.chemistry.logs.ChemistryMessageConverter"/>

  <appender name="CONSOLE" class="ch.qos.logback.core.ConsoleAppender">
    <layout class="ch.qos.logback.classic.PatternLayout">
      <pattern>%-5level %class{0} - %chmsg%n</pattern>
    </layout>
  </appender>

  <logger name="org.apache.fop" level="off" additivity="false">
    <appender-ref ref="CONSOLE"/>
  </logger>

  <root level="info">
    <appender-ref ref="CONSOLE"/>
  </root>

</configuration>
```

2. Set the configuration file from the command line using the `-log-file` parameter.

Security Policy Configuration

When running the product on a system where you do not control the input (XML files, CSS), you must take some steps to ensure that the transformation process does not access files outside the allowed locations, and does not connect to other hosts. Follow this procedure:

1. Create a Java policy file. A sample Java policy file can be found in `config/chemistry.policy`. You can use this as it is or as a starting point to grant or revoke permissions. Follow the instructions from this file.

2. Specify the Java policy file location (in URL or file path syntax) using the `-security-policy` command-line parameter:

```
chemistry.bat -security-policy file:/some/path/to/chemistry.policy
```

3. By default, the font cache file is stored in the home directory, while the temporary files are stored in the system temp folder. It is recommended to specify a workspace directory where these files are to be stored. The sample policy file automatically sets read and write permissions on this folder.

```
chemistry.bat \

  -security-policy file:/some/path/to/chemistry.policy \

  -security-workspace /path/to/dir
```

4. If your CSS files, images, fonts, or other resources are stored in a different folder than the one that contains the input file, you need to indicate those folders.

   ```
   chemistry.bat ... -security-resources-dir1 /path/to/resources \

                     -security-resources-dir2 /other/path/to/resources
   ```

5. If you access resources, from another server, you have to give access to connections to it (note that Google fonts servers are already added to the policy file).

   ```
   chemistry.bat ... -security-resources-host my.font.and.css.server:80
   ```

Related Information:

Java SE: Default Policy Implementation and Policy File Syntax

2.

Styling for Print

This section contains information about styling various components in the 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 8)*
Mozilla MDN Web Docs: HTML Element Reference

Page Formatting

This section contains information about how you can use the new concepts from the *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: 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 open 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 you may want 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

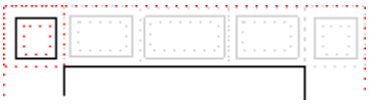
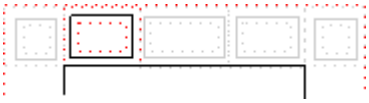
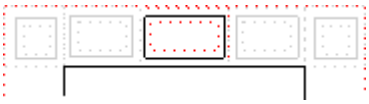
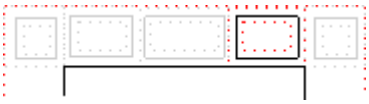
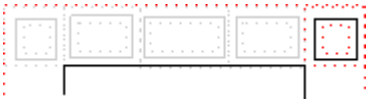
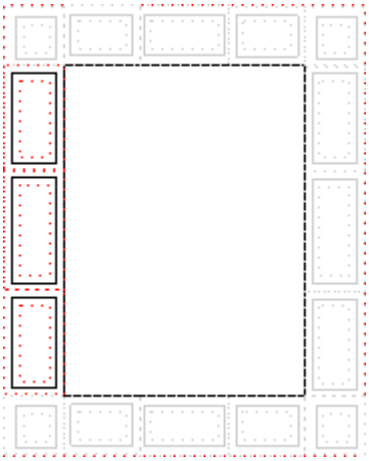
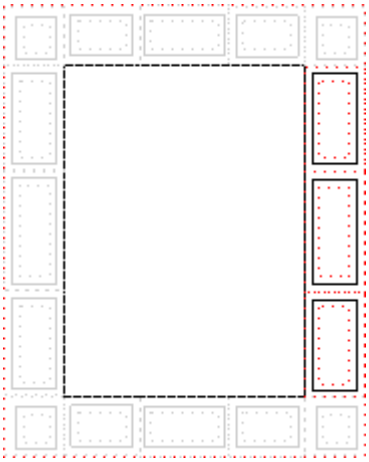
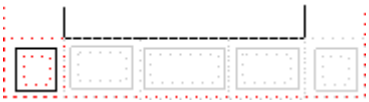
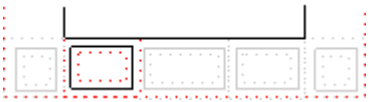
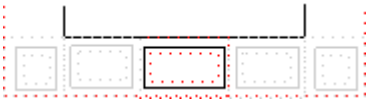
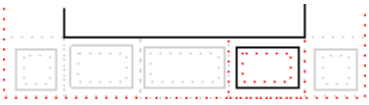
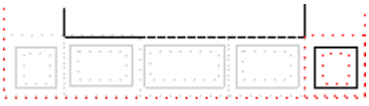
Box	Description	Placement
top-left-corner	A fixed-size box defined by the intersection of the top and left margins of the page box.	
top-left	A variable or fixed-width box that fills the top page margin between the top-left-corner and top-center page margin boxes.	
top-center	A variable or fixed-width box centered horizontally between the page's left and right border edges and fills the page top margin between the top-left and top-right page margin boxes.	
top-right	A variable or fixed-width box that fills 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.	

Table 1. Page-Margin Box Definitions (continued)

Box	Description	Placement
left-top	A variable-height box that fills 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 fills the left page margin between the left-top and left-bottom page margin boxes.	
left-bottom	A variable-height box that fills the left page margin between the left-middle and bottom-left-corner page margin boxes.	
right-top	A variable-height box that fills 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 fills the right page margin between the right-top and right-bottom page margin boxes.	
right-bottom	A variable-height box that fills 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 or fixed-width box that fills the bottom page margin between the bottom-left-corner and bottom-center page margin boxes.	
bottom-center	A variable or fixed-width box centered horizontally between the page's left and right border edges and fills the bottom page margin between the bottom-left and bottom-right page margin boxes.	
bottom-right	A variable or fixed-width box that fills 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.	

Notes:

- The rules for variable-width at-rules (top or bottom) are the following:
  - When **one** box is defined, it takes the **full** available width.
  - When **left** and **right** boxes are defined, they take **half** of the available width.
  - When **all 3** boxes are defined, each of them is **one-third** of the available width.
- The **width** property values must be set for all 3 at-rules when using fixed values (regardless if those values are relative or absolute).
- The **initial** and **unset** property values are not supported when applied to page margin boxes.

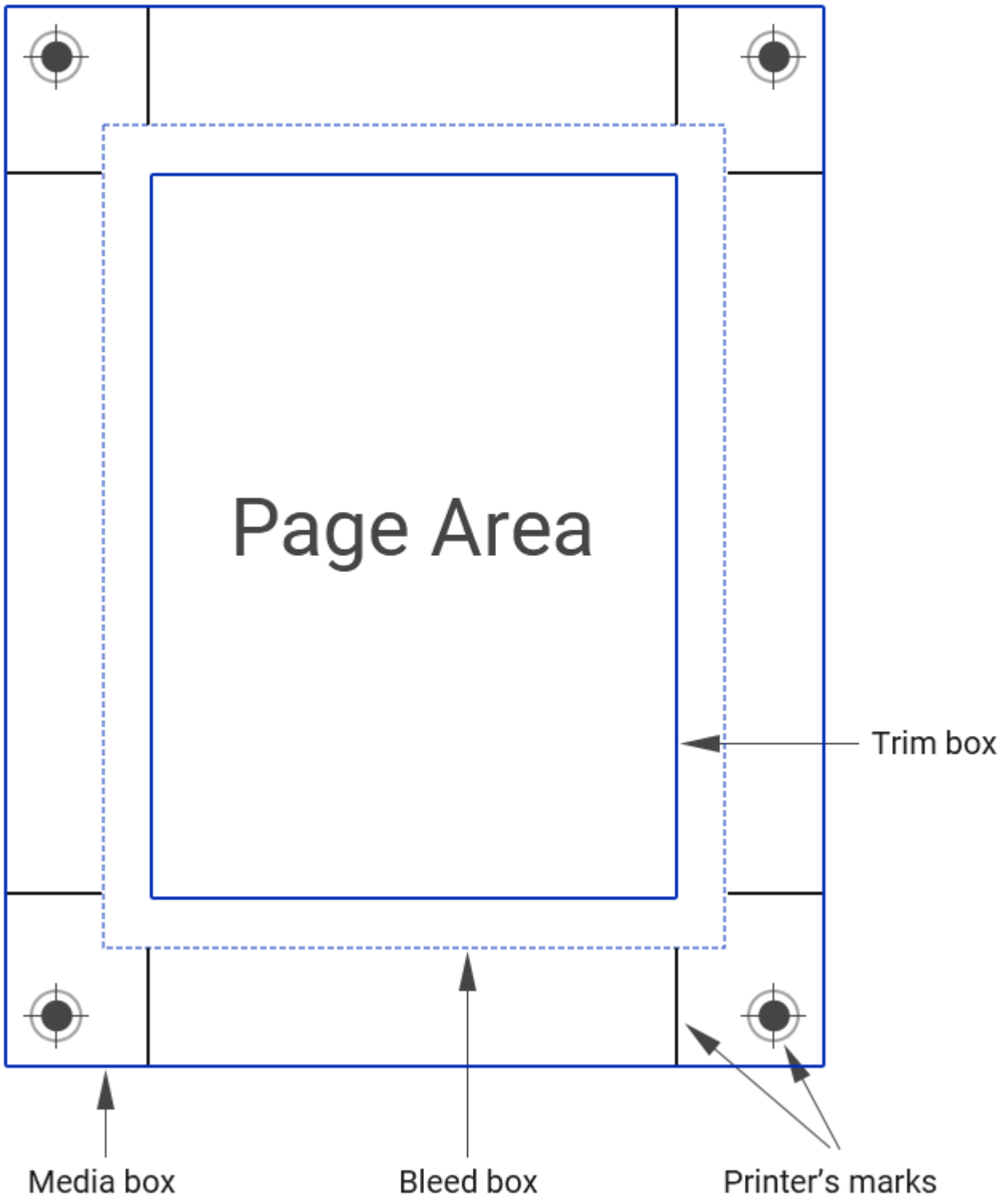
Page Boundaries

The PDF specification defines two printing medium: the finished medium (like a sheet of paper) and the intermediate medium used in prepress process (like a film or reproduction plate). The intermediate medium may contain additional content such as bleeds or printer marks that are presented outside the finished page boundaries. To handle these cases, a PDF page may define different boundaries to control each aspect:

- The *media box* defines the boundaries of the physical medium where the page is to be printed. It may include any extended area surrounding the finished page for bleeding, printing marks, etc.
- The *bleed box* defines the boundaries where the content of the page shall be cropped in a production environment. It may include any extra bleed area needed to accommodate the physical limitations of cutting, folding, and trimming equipment.
- The *trim box* defines the boundaries of the finished page after trimming.

The boundary can be set directly from the CSS by using the `-oxy-crop-box` *(on page 136)* property:

```
@page {
  -oxy-crop-box: trim-box;
}
```



Page Bleeding

Page bleeding can be controlled by using either one (or both) of the `bleed` and `-oxy-crop-offset` *(on page 136)* properties:

```
@page {
  -oxy-crop-offset: 1cm;
}
```

Note:
If only one of the properties is present, its value will be used. If both are set, then:

- If the crop box is set to media-box (default), the bigger value will be used.
- If the crop box is set to bleed-box, only the bleed value will be used.
- If the crop box is set to trim-box (finished page), no value will be used.

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 136)* `-oxy-column-break-before` *(on page 136)* `-oxy-column-break-after` *(on page 136)*.

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:

- **break-before**
- **break-after**
- **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 {
  break-before: always;
}
```

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

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

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

```
table, ol, ul {
  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

Page Breaks Between Lines: Widows and Orphans

There are cases where a page break is placed between paragraph lines. Typically, for aesthetic purposes, the first paragraph should have more than one line. Another constraint might be the number of lines that are moved to the next page and again, it should avoid leaving a single line. CSS defines two properties for this type of control:

- `orphans` - This property specifies the minimum number of line boxes that should be left in a paragraph before the page break occurs.
- `widows` - This 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;
}
```

Note:
As a difference from the W3C standard, the `widows` and `orphans` CSS properties are applied to lists as well (the default is 2). This means that a list that spans consecutive pages will have either zero or at least 2 lines on each of the pages.

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 138)*:

```
@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 136)* 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 example above, 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 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 41)*

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: "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 the 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);
...
```

How to Add a Link in Headers and Footers

Method 1: Using an SVG Link Attribute

It is possible to add a link inside the document header (or footer) by using the `<a>` element inside an SVG document. For example, suppose you have the following SVG document named *custom.svg*:

```
<svg width="180" height="20" viewBox="0 0 180 20" xmlns="http://www.w3.org/2000/svg"
  xmlns:xlink="http://www.w3.org/1999/xlink">
    <a xlink:href="https://www.oxygenxml.com/chemistry-html-to-pdf-converter.html">
        <rect x="0" y="0" width="180" height="20" opacity="0"/>
        <text x="5" y="15" fill="blue">Oxygen PDF Chemistry</text>
```

```
    </a>
</svg>
```

This creates an SVG link with **Oxygen PDF Chemistry** displayed as its text (the content of the `<text>` element).

Note:
If you just want to add a link without text, you can define a rectangle that contains the link instead of text.

To display the link, you just need to set your SVG file as the content of one of the page margin boxes:

```
@page {
  @top-left {
    content: url("custom.svg");
  }
}
```

Method 2: Using the CSS *-oxy-link* Property

It is also possible to add a link inside the document header (or footer) by using the `-oxy-link` property on the `@page` margin box declaration. The entire page margin box will behave as a link and will be clickable.

```
@page {
  @top-left {
    content: "Link";
    -oxy-link: "https://www.oxygenxml.com/";
    color:blue;
  }
}
```

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 a 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 47)*.

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 a *leader*.
- The page number of the target chapter/section.

It may look like this:

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

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 45)*, but 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 following example associates the defined page to the `<ul>` element that gives the structure:

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

To style the TOC entries, this next snippet removes the bullet decoration from the `<li>` elements, then marks the `<a>` element as being a link (the name for each TOC entry is defined inside an `<a>` element).

```
ul.toc li {
    list-style-type:none;
}
ul.toc a {
    display: block; /* Only necessary when using a leader */
    link: attr(href);
    text-decoration:none;
}
```

Note:
When using a *leader*, the alignment for each TOC entry is normally *justified*. The `display: block` property is used to treat the contents of the <a> element as a separate block with a different alignment (i.e. *Align Left*).

After the name for each TOC entry (the content defined inside the `<a>` elements), 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.

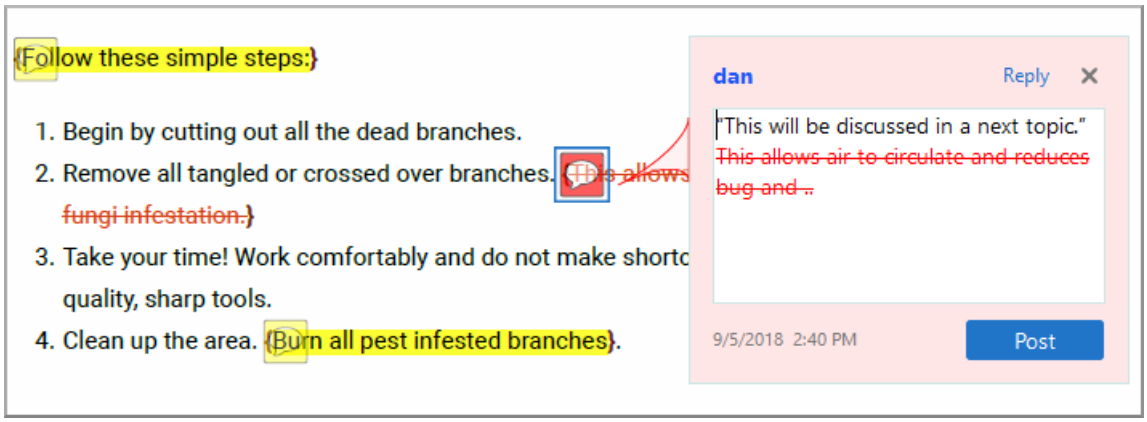
Also, it uses the page number of the target element (after the *leader*):

```
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 - 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-hl">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"><image 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:

```
body {
    counter-reset: chapter;
}
...
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 provides 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
- Copyright information

Oxygen PDF Chemistry automatically extracts this information 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'>
    <meta name='copyright' content='Copyright My Company 2021'>

...
```

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'`. Its value should be a list of tokens, separated by commas:

```
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);
}
```

-oxy-pdf-meta-copyright

-oxy-pdf-meta-copyrighted

-oxy-pdf-meta-copyright-url

These properties define the copyright metadata. Acrobat Reader Pro, for example, displays this in the **Details** tab of the **File/Document Properties** dialog box.

```
meta[name='copyright'] {
    -oxy-pdf-meta-copyright: attr(content);
    -oxy-pdf-meta-copyrighted: copyrighted;
    -oxy-pdf-meta-copyright-url: "https://my.company/copyright-notice.html";
}
```

The `-oxy-pdf-meta-copyright` property specifies the copyright text for its value, the `-oxy-pdf-meta-copyrighted` property specifies whether or not the publication is copyrighted (accepts only `copyrighted` or `public-domain` for the value), and the `-oxy-pdf-meta-copyright-url` property can be used to specify the location of an external copyright notice.

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 defines a list of pairs. Each pair contains the name and the value for the meta information field. The pairs must be separated by a comma: `name1 value1, name2 value2`

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

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

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

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

> In case of conflicts, when two or more elements trigger the 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.

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 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: Specify the language on the root of your document.

For XML documents, use `Use xml:lang on the root of your document. For HTML documents, use the @lang attribute.` on the root of your document. For HTML documents, use the `@lang` attribute.

Hint 3: Set alternate text on all 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 100)*.

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:

```
:root{
  font-family: Arial;
}

@page {
    @top-left {font-family: Arial }
    @top-right {font-family: Arial }
    @top-center {font-family: Arial }
    @top-left-corner {font-family: Arial }
    @top-right-corner {font-family: Arial }

    @bottom-left {font-family: Arial }
    @bottom-right {font-family: Arial }
    @bottom-center {font-family: Arial }
    @bottom-left-corner {font-family: Arial }
    @bottom-right-corner {font-family: Arial }
}
```

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 53)*

Archiving

PDF/A is the ISO standard for PDF specialized in the archiving and long-term preservation of electronic documents. To use this mode, you must:

1. Set a PDF/A mode from the command line using the `-pdf-a` parameter with one of the following values:
   - PDF/A-1a
   - PDF/A-1b
   - PDF/A-2a
   - PDF/A-2b
   - PDF/A-2u
   - PDF/A-3a
   - PDF/A-3b
   - PDF/A-3u
2. Embed all of the fonts. If you use 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 100)*.

Signing

The following Bouncy Castle libraries must be downloaded (in their last available version):

- bcpkix-jdk18on
- bcprov-jdk18on
- bcutil-jdk18on

Follow these steps to generate signed PDFs for XML or HTML content. For DITA content, check How to Sign a PDF.

1. Install an external **Oxygen PDF Chemistry** processor.

   Restriction:
   A separate installation is required to include the additional libraries.

2. Copy Bouncy Castle libraries into `[CHEMISTRY_INSTALL_DIR]/lib`.

3. Create a keystore file (if you do not already have one issued):

```
openssl req -new -newkey rsa:4096 -nodes -keyout my.key -out my.csr

openssl x509 -req -sha256 -days 365 -in my.csr -signkey my.key -out my.pem

cat my.key my.pem > my2.pem

openssl pkcs12 -export -in my2.pem -out mykeystore.pkcs12 -name myAlias
```

4. Set the keystore from the command line using the `-keystore-file` parameter. This file alone is enough to sign a PDF.

5. If you need to add more information (e.g. your company name, location, or reason) or to set a password, you can use the following parameters:
    - `-sign-name`
    - `-sign-location`
    - `-sign-reason`
    - `-sign-password`

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 `` in XML or `` in HTML.

auto

Words are hyphenated automatically according to an algorithm that is 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 12)*.

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 *(on page 60)*.

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

none

Currently not supported by Oxygen PDF Chemistry and it falls back to `manual` *(on page 60)*. 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 off the page:

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

Hyphenation Dictionaries

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

Code	Language
da	Danish
de	German
de_CH	German (Switzerland)
en	English
en-GB	English (Great Britain)
es	Spanish
fr	French
it	Italian

Code	Language
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

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

The `.xml` files allow you to access the licensing terms and you can use them as a starting point to create customized dictionaries (see How to Alter a Hyphenation Dictionary *(on page 62)*).

The `.hyp` files are the compiled dictionaries that the Oxygen PDF Chemistry actually uses.

The hyphenation dictionaries are located in: `[CHEMISTRY_INSTALL_DIR]/config/hyph`.

One simple way to add more dictionaries:

1. Download and extract the `offo-hyphenation-compiled.zip` file. This file is a bundle of many dictionary files.
2. Copy the `fop-hyph.jar` file to the `[CHEMISTRY_INSTALL_DIR]/lib` directory.
3. If you just need a single dictionary, place the `.hyp` or `.xml` file extracted from the above jar in the `[CHEMISTRY_INSTALL_DIR]/config/hyph` directory, or in another directory and use the `-hyph-dir` parameter.

How to Alter a Hyphenation Dictionary

The hyphenation dictionaries are stored as XML files in the `[CHEMISTRY_INSTALL_DIR]/config/hyph` directory.

You can copy the dictionaries you need to change in another directory, then use the `-hyph-dir` parameter to reference it inside your transformation.

Each file is named with the language code and has the following structure:

```
<hyphenation-info>


<hyphen-min before="2" after="3"/>


<exceptions>
o-mni-bus
...
</exceptions>


<patterns>
préémi3nent.
proémi3nent.
surémi3nent.
....
</patterns>


</hyphenation-info>
```

To change the behavior of the hyphenation, you can modify either the patterns or the exceptions sections:

exceptions

Contains the list of words that are not processed using the patterns, each on a single line. Each of the words should indicate the hyphenation points using the hyphen ("-") character. If a word does not contain this character, it will not be hyphenated.

For example, `o-mni-bus` will match the `omnibus` word and will indicate two possible hyphenation points.

Note:
Compound words (i.e. e-mail) cannot be controlled by exception words.

patterns

Contains the list of patterns, each on a single line. A pattern is a word fragment, not a word. The numbers from the patterns indicate how desirable a hyphen is at that position.

For example, `tran3s2act` indicates that the possible hyphenation points are "*tran-s-act*" and the preferable point is the first one, having the higher score of "3".

How to Disable Hyphenation for a Word

To disable hyphenation for a specific word, there are several possible approaches:

- If the word is a compound (like "T-shirt") and you want to keep it on the same line, you have two options:

  Manual Approach

  Wrap the word in an inline element with the `@outputclass` attribute set. In the CSS, change its style to `white-space:nowrap;`. For example:

  ```
  .. <ph outputclass="no-hyphenation">T-shirt</ph>...
  ```

  ```
  *[outputclass ~= "no-hyphenation"] {
   white-space: nowrap;
  }
  ```

  Automatic Approach

  A better alternative to this is to write an XSLT extension that matches the text nodes and performs automatic markup (to see an example, go to How to Wrap Words in Markup in the XSLT Extensions for PDF Transformations section). Then match the `compound-word` class the same as in the previous example:

  ```
  *[outputclass ~= "compound-word"] {
   white-space: nowrap;
  }
  ```

  Another Alternative

  In all the compound words from your documentation, replace the hyphen ("-") with a non-breaking hyphen character `U+2011` (or XML notation `ߛ`).

  Then change the *autocorrect* settings to automatically replace the compound word with its equivalent. For example: "T-shirt" would be replaced with "T[\u2011]shirt".

- If the word is not a compound, you have two options:
  - Use one of the approaches listed above.
  - Alter the hyphenation dictionaries as explained in: How to Alter a Hyphenation Dictionary *(on page 62)*.

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:" ";
}
```

How to Force or Avoid Line Breaks at Hyphens

It is possible to force or avoid line breaks inside words with hyphens (`U+2010`). This can be useful, for example, inside tables that have product references if you want the display to remain on a single line (or to split it on multiple lines). To achieve this, you can use the `-oxy-break-line-at-hyphens` property:

The accepted values are:

auto

Words are hyphenated automatically according to an algorithm that is driven by a hyphenation dictionary. This can lead to line breaks at hyphens.

avoid

Words are still hyphenated automatically except no line break will occur on hyphens.

always

Words are still hyphenated automatically except line breaks will be forced on hyphens.

Example:

Suppose you have a products table like this:

```
<table>
  <row>
    <cell>Product-1233-55-88</cell>
    <cell>120</cell>
  <row>
  <row>
    <cell>Product-1244-66-99</cell>
    <cell>112</cell>
  <row>
</table>
```

and the following rule in a CSS stylesheet:

```
table {
  -oxy-break-line-at-hyphens: avoid;
}
```

In the output, the list of product references will be displayed in a single line. On the contrary, setting the property value to `always`, will force a break after each hyphen.

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

Using the :before(n) and :after(n) CSS Pseudo-Elements

Although not standard, this extension may be useful if you want to style sections by adding multiple levels of 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 specific aspects of it (for example, just the chapter number with a larger font and with red). However, you can do it using multiple `before(n)` or `after(n)` 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;
}
```

Notes:

- The bigger the level, the more distant the pseudo-element is.
- Level 1 corresponds to normal `:before` or `:after` pseudo-elements.

Change Bars

Change bars are used to mark revised elements in the content. For example, they are useful for technical documentation to flag content that was added in a new version of the file.

Display Change Bars Using the `::changebar` Pseudo-Element

Suppose you have the following document:

```
<p>Once every 6000 kilometers or three months, you need to change the oil in your car.
It will extend your car lifetime.<span class="cb">This should be done by a specialist
in a controlled environment.</span></p>
```

To mark the `span` element as being a *change bar*, you can use the `::changebar` pseudo-element:

```
.cb::changebar {
  -oxy-changebar-offset: 1mm;
  -oxy-changebar-placement: start;
  -oxy-changebar-style: solid;
  -oxy-changebar-color: black;
  -oxy-changebar-width: 1pt;
}
```

Tip:

The *change bars* can be customized using the following properties:

- -oxy-changebar-offset *(on page 135)*
- -oxy-changebar-placement *(on page 135)*
- -oxy-changebar-style *(on page 135)*
- -oxy-changebar-color *(on page 135)*
- -oxy-changebar-width *(on page 135)*

Display Change Bars Using Start and End Markers

Suppose you have the following document:

```
<p>Once every 6000 kilometers or three months, you need to change the oil in your car.
It will extend your car lifetime.<change-bar-start color="blue">This should be done by
a specialist in a controlled environment.<change-bar-end></p>
```

To display the *change bar* inside the paragraph, you can use the `display: -oxy-changebar-start` and `display: -oxy-changebar-end` properties on the `<change-bar-start>` and `<change-bar-end>` custom elements:

```
change-bar-start {
  display: -oxy-changebar-start;
  -oxy-changebar-color: attr(color);
}
change-bar-end {
  display: -oxy-changebar-end;
}
```

Notes:

- All of the `-oxy-changebar` CSS properties support the `attr()`, `oxy_xpath()`, and `calc()` functions.
- If you do not want to create new elements to mark the starting and ending point of the *change bars*, you can use the `display` property on both the `::before` and `::after` pseudo-elements (you can also use `::changebar`).

3.

Layout

This section includes topics that describe how to style elements as tables or lists, how to rotate content, and how to use inline blocks.

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.

This is an example of a very simple table in an XML document:

```
<tbl>
    <caption>A table.</caption>
    <rw>
        <c rspan="2"> Spans vertically </c>
        <c> Other cell </c>
        <c> Other cell </c>
    </rw>
    <rw>
        <c cspan="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 span multiple rows or columns. For this, 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 define table captions 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

Any element marked with either the `table-header-group` or `table-footer-group` property is automatically repeated when a table is split over a sequence of pages. It is required that they contain only elements with the `display` property set to `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;
}
rw {
```

```
    display: table-row;
}
```

Repeating Captions

By default, the captions are not repeated on all pages a table spans. To enable this, you should mark the elements with a table `caption` display as with the `-oxy-caption-repeat-on-next-pages` property:

```
caption {
    -oxy-caption-repeat-on-next-pages: yes;
}
```

The allowed values are **yes** or **no** and it is inheritable.

If you want to indicate that the page is a continuation, you can associate a static text to the caption that appears only on subsequent pages. For example, you can use an `:after` element as in the following example:

```
caption:after(2) {
    display:inline;
    content: "(continued)";
    -oxy-show-only-when-caption-repeated-on-next-pages: yes;
}
```

This example uses an `:after` element with the value of **2** to avoid conflicts with normal `:after` static elements that you may already use in your CSS.

Column Width and Styles

Oxygen PDF Chemistry supports an automatic layout for tables. This means the allocation of column width 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;
}
```

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 uses an automatic layout, you should specify a width (such as 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 use the `table-column` value in a `display` property for the element that represents the column. Suppose you have 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 {
    disply:table;
}
colgr {
    display: table-column-group;
}
column {
    display: table-column;
    width: attr(wd);
}
column:first-of-type {
    background-color: yellow;
}
```

Proportional column widths (such as the ones used in the CALS tables from DITA or DocBook) are supported, but only when they are set in a `@width` attribute on the `<column>` element (the one with the `display` property set to `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 orientation is *portrait*, but if you have a wide table with lots of columns, it might bleed to the right of the page.

There are two ways of solving this:

- Associate a wider page (i.e. 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 also need to 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, there is static text in the top (header) for the normal pages 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;
  }
}
```

Rotating Cell Content

To rotate table cell content, you can use the `transform` CSS property. Remember that only the content of the cell is rotated, not the cell itself. The rotation angle is clockwise. In the following example, the cells marked with the class `rotate` on their side are rotated:

```
.rotate {
  transform:rotate(270deg);
  font-style: bold;
  text-align: left;
```

```
  vertical-align: bottom;


  white-space: pre;


  background-color: silver;
}
```

The rotated cells are in the header of the table.

```
<html xmlns="http://www.w3.org/1999/xhtml">
    <table>
        <tr>
          <th>
            Type
          </th>
          <th colspan="3">Features</th>
        </tr>
        <tr>
          <th></th>
          <th class="rotate">Sport</th>
          <th class="rotate">All Terrain</th>
          <th class="rotate">Family</th>
        </tr>
........
```

The result is:

Type	Features		
	Sport	All Terrain	Family
Dacia Duster		X	X
Nissan Leaf			X
Volvo XC90		X	X
Toyota Celica	X	X	

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.

The *list* element needs to have the `display` property set to `block`, and the children elements need to have the `display` property set to `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 that the bullet will have enough space to be painted inside the list item box.

List Marker Position

You can select whether 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, on 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**
- **<string>**

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. By default, it uses the following list of fallback fonts: `Times New Roman, Liberation Serif, DejaVu Serif, Symbol, Segoe UI Symbol, FreeSerif, ZapfDingbats`.

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-family: "Segoe UI";
  text-align:left;
  counter-increment: cnt;
  content:counter(cnt)" - \0430";
}
```

To change the marker symbol and its size:

```
li:marker {
  /* Club Symbol */
  content: "\2663";
  font-size: 0.8em;
}
```

To use an image instead of a number as a 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 unique or currently not supported (such as lists lots 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";
}
...
```

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 a percentage).

Align Center

This can be done by setting both the `margin-left` and `margin-right` properties 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` properties 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` properties 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); /*Clockwise*/
}
```

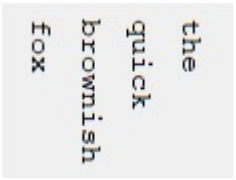
The block needs to have a specified `width`. If this is not specified, the built-in algorithm will try to compute a width and then rotate the block. For example, for rotating the text:

```
the quick brownish fox
```

The algorithm will try to break the line at each space (to wrap it as much as possible, so it gets a minimum width):

```
the
quick
brownish
fox
```

It will then use the maximum word size (the width of the `brownish` word). The result will be:



If you want the entire phrase to be rotated (without wrapping), then you should specify that all the whitespaces must be preserved:

```
code {
  transform: rotate(90deg); /*Clockwise*/
  white-space: pre;
}
```

The result will be:

```
the quick brownish fox
```

4.

Graphics

This section includes topics that describe how to style graphics in your PDF output.

Images

This topic contains information about how you can reference images 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: <img src="happy.png" />. </p>
...
```

For XML, you must add CSS rules that pick up the content of an attribute and use it as a source for the 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);
}
```

Note:
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.

Supported Image Types

Oxygen PDF Chemistry supports the following types of raster images:

- 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 types of vector images:

- SVG (Scalable Vector Graphics)
- PDF (PDF Documents)

PDF Images

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

```
<img src="my_doc.pdf"/>
```

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

```
<img src="my_doc.pdf#page=5"/>
```

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 84)* 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.

```
<img src="my_image.png" width="300" height="250" />
```

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

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 on 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 `-image-resolution` command-line parameter.
- 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 to 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;
}
```

Tip:

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 it 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 8)*

https://drafts.csswg.org/css-images-4/#the-image-resolution

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` property *(on page 84)*.

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 the same as for web pages:

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

Foreground Images

You can use foreground images to impose a texture above entire pages (in the foreground).

Supported property: **-oxy-foreground-image**.

Usually, the foreground images contain watermarks (for example, "Draft", "Copyright", or "Do Not Copy"). This is an example of how can you do it for all the pages:

```
@page {
    size: A4;
    -oxy-foreground-image: url("do-not-copy.svg");
}
```

Notes:

- Make sure you use an image type that supports transparency.
- It is better to use SVG for the page artwork because they do not suffer from *pixelation* and because foreground images are stretched to the full page size.

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)"`).
- **rbga** - 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.

SVG

Oxygen PDF Chemistry supports SVG images. The main advantage of using SVG is that the image looks good on paper no matter its size.

SVG Referenced or Embedded in the Document

These can either be referenced as external resources:

```
<p> This is an SVG showing a happy face: <img src="happy.svg"/></p>
```

or embedded in 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" />
   </svg>
</p>
```

The document styles are also applied to the SVG fragments. For instance, if the `<circle>` element has the `sun` class, you could change its appearance by using `.sun {fill="orange"}` in your main CSS. As a general rule of thumb, keep distinct names for the SVG fragment class attributes from the ones used for general content styling.

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

Using SVG for Styling

To use SVG to decorate an element:

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

To set an SVG image as the 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.

Linking from SVG to Parts of the Host Document

If you need to use the graphics as a type of table of contents, you can place links over parts of the image (used as callouts) that point to some descriptive sections in your document by using the `<a xlink:href="..">` markup. This is similar to what the `<imagemap>` HTML element does, but this is encoded directly in the graphics:

```
<p> This image has a link that points to a paragraph:
   <svg
         xmlns="http://www.w3.org/2000/svg"
```

```
         xmlns:xlink="http://www.w3.org/1999/xlink"
         viewBox="0 0 50 50"
         width="50"
         height="50">

      <a xlink:href="explain-circle">
        <circle cx="25" cy="25" r="10" />
      <a>

   </svg>
</p>
....
<p id="explain-circle">
  The circle is a round shape.
</p>
```

SVG Templates

Oxygen PDF Chemistry supports SVG template images (with a `.template.svg` file extension). These files can contain XPath expressions that will be expanded by the processor.

Using SVG Template as a Cover Page

A good use-case for SVG templates is when you want to create a custom cover page. For example:

1. Suppose the following HTML file should be printed with a cover page:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <title>How to Dynamically add Content to SVG</title>
        <link rel="stylesheet" href="styles.css"/>
        <meta name="author" content="John Doe"/>
        <meta name="version" content="1.1"/>
        <meta name="copyright" content="© Oxygen PDF Chemistry. For demonstration purpose."/>
    </head>
    <body>
        ...
    </body>
</html>
```

2. The cover image (for example, named `cover.template.svg`) should display `<meta>` nodes information and the main title. Both the *author* and *version* will be displayed in rectangular shapes

prefixed with the displayed information ("Author:" and "Version:"), while the title is positioned in the middle of the page, at some absolute coordinates.

```
<svg width="8.5in" height="11in" xmlns="http://www.w3.org/2000/svg">
  <g>
    <rect height="39" width="180" y="40" x="579" stroke="#000" fill="#fff"/>
    <rect height="39" width="180" y="79" x="579" stroke="#000" fill="#fff"/>

    <text font-size="8" stroke-width="0" y="52" x="582" stroke="#000" fill="#000000"
      >Author:</text>
    <text font-size="16" stroke-width="0" y="66" x="643" stroke="#000" fill="#000000"
      >${//meta[@name='author']/@content}</text>

    <text font-size="8" stroke-width="0" y="91" x="582" stroke="#000" fill="#000000"
      >Version:</text>
    <text font-size="16" stroke-width="0" y="104" x="659" stroke="#000" fill="#000000"
      >${//meta[@name='version']/@content}</text>

    <text font-size="30" stroke-width="0" y="252" x="123" stroke="#000" fill="#000000"
      >${//head/title}</text>
    <text font-size="12" stroke-width="0" y="993.40002" x="266" stroke="#000" fill="#000000"
      >${//meta[@name='copyright']/@content}</text>
  </g>
</svg>
```

Notes:

- XPath expressions are not expanded if the SVG template is open in **Author** mode.
- XPath expressions can be tested (without `${}`) using the XPath/XQuery Builder view.

Important:

- If you received the SVG image from someone else (e.g. a graphics designer), make sure that the text from the image was not converted to glyph shapes and that it is rendered using the `<text>` element.
- The SVG `<text>` element does not wrap the text if it overflows the image. If you have longer text that needs to be rendered, you might consider using multiple `<text>` elements and more evolved XPath expressions (for example, using the `substring()` function) to place the text on multiple lines.

Tip:

You can ask a designer to fill the image with some placeholders that you can later find and replace with your XPath expressions. In the above SVG, the designer could place the text `Here comes the version`, that you replace with `${//meta[@name='version']/@content}`:

```
<text font-size="8" stroke-width="0" y="91" x="582" stroke="#000" fill="#000000"
>Version:</text>
<text font-size="16" stroke-width="0" y="104" x="659" stroke="#000" fill="#000000"
>Here comes the version</text>
```

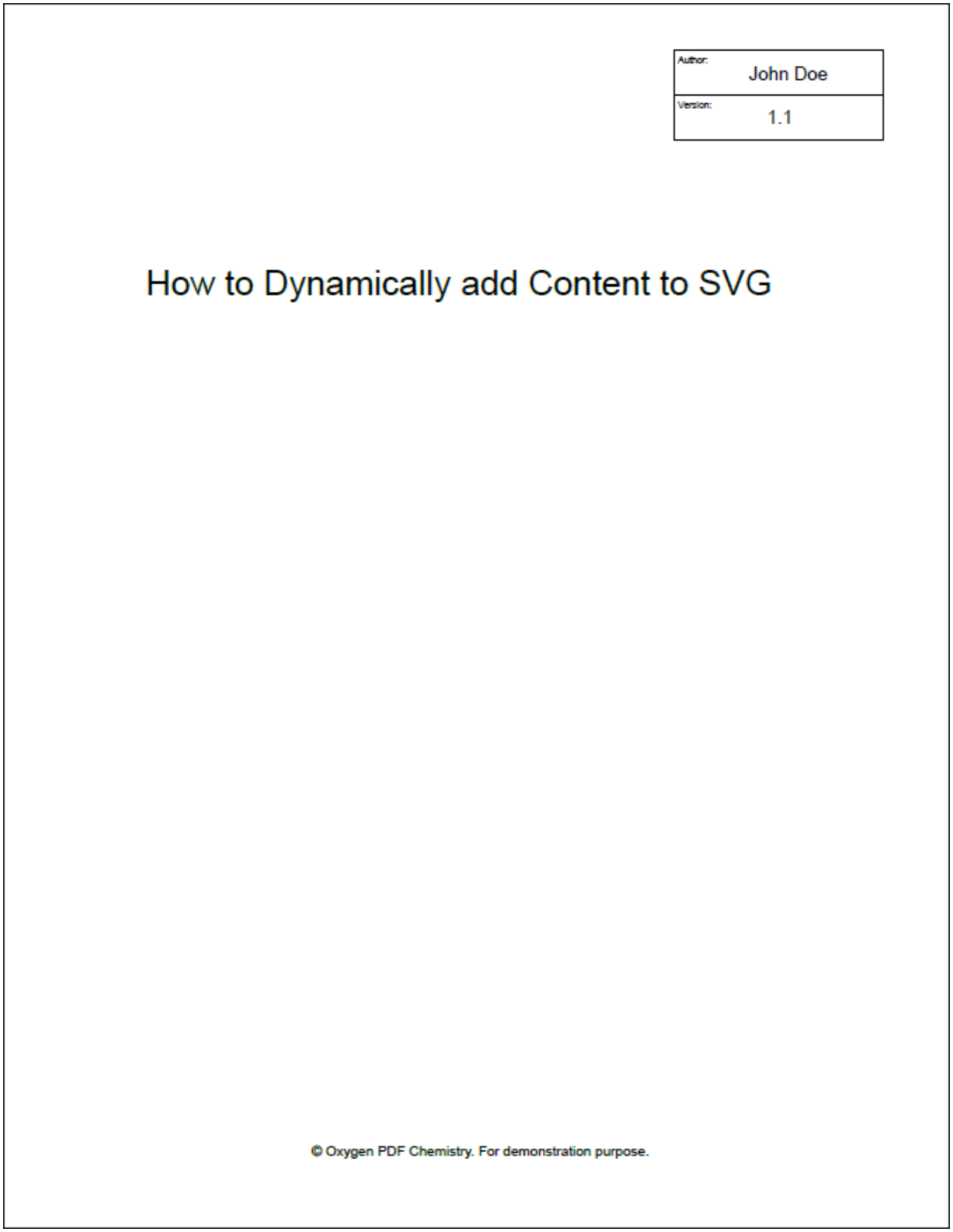
3. The CSS stylesheet should declare a synthetic page to be printed before the HTML document and display the SVG cover image on this new page:

```
@page cover-page {
  background-image: url('cover.template.svg');
  background-repeat: no-repeat;
}

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

You can even create a sequence of synthetic pages using `before(1)`, `before(2)`, …, `before(n)` pseudo-elements *(on page 66)* and multiple `@page` definitions for each of the `before` elements.

4. After the transformation, the final document cover will look like this:



MathML

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

```
<p> The quadratic formula: <img src="quadratic.mml"/></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>
    <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>
```

Notes:

- No styling is required to show the embedded MathML in the output. It works automatically for HTML or XML document types.
- For HTML5, the namespace declaration is not required.

Image Maps (HTML)

The HTML `<map>` element is supported by the processor. It allows you to define a set of shapes over your original image and each shape behaves like a link pointing to a part of your publication or to an external resource.

1. Start by specifying the width and height of your image using attributes. The size and coordinates are in pixels. The size you define here is very important when you specify the coordinates of the shapes. You can use any CSS unit, including percents. The percents are solved relative to the image size and represent a way of creating "responsive" image maps (reusing the map on the same image with different sizes depending on the position in the document). If you are using the same customization CSS for HTML web output as well, make sure you use only pixels as some of the browsers do not support other units.

```
<img src="engine-picture.png" width="400" height="300">
```

2. Create a map element and link the image to the map:

```
<img src="engine-picture.png" width="400" height="300" usemap="mapID">

<map name="mapID">...</map>
```

3. In the map element, add areas, each with a shape and a set of coordinates:

```
<area shape="rect" coords="100,70,120,90"

          href="#radiator" alt="Radiator" />

....
```

4. Verify how the shapes look in the output. You can make the shapes visible by using **one of the following methods**:
   - Using the `-show-image-map-area-shapes` and `-show-image-map-area-numbers` command-line arguments.
   - Adding a CSS snippet to your customization. The shapes have the `image-map-shape` class, the bullet around the image map number (`image-map-number`), and the text inside the bullet (`image-map-number-text`). To make them translucent yellow:

```
.image-map-shape{
 fill: yellow;
 fill-opacity: 0.5;
 stroke-opacity: 0.5;
}
.image-map-number-text {
 visibility: visible;
}
.image-map-number {
 fill: yellow;
 fill-opacity: 0.4;
 stroke-opacity: 0.7;
}
```

Videos

This topic contains information about how you can reference videos from your HTML or XML documents.

Warning:

- Only Acrobat Reader supports video embedding.
- Only the following extensions are supported: `.mov`, `.mp4` and `.m4v`.

Tip:
To process large video files, you need to increase the available memory (e.g. 3.5Gb of memory for a 2Gb video).

For HTML, the `<video>` tag can be used without any other styling in your CSS files:

```
...
<video data="path/to/video.mp4" format="mp4" width="480" height="270">
    <source src="path/to/video.mp4"/>
</video>
...
```

Note:
It is recommended to use absolute paths for `@data` and `@src` values like this: `src="file:/C:/absolute/path/to/video.mp4"`.

For XML, you must add CSS rules that pick up the content of an attribute and use it as a source for the video:

```
...
<para>Like shown in this video: <video src="files/earth.mp4"/>.</para>
...
```

For example, the CSS rules may look like this:

```
video {
  display: inline;
}
video[src] {
  content: attr(src, url);
}
```

Note:
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.

Change the Video Size

If your XML contains additional attributes such as `@width` or `@height`, you can complete the CSS:

```
video[width] {
  width: attr(width, length);
}
video[height] {
  height: attr(height, length);
}
```

Notes:

- It is also possible to set both `@width` and `@height` attributes directly on the video `<object>`.
- If no values are set for width/height (neither from the CSS nor attributes), the extracted video size will be used.

Customize Video Cover Placeholder

By default, a placeholder is displayed in place of the video. When clicked, this placeholder will launch the video.

It is possible to change this placeholder with a custom one by using the `-oxy-video-cover` *(on page 134)* property:

```
video {

  -oxy-video-cover: url("files/cover.png");

}
```

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 TrueType 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. Some features such as ligatures and style sets are supported.
- Using a Tamil font might result in content that is not searchable or copyable.

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*.

Fallback Fonts

Whenever a character is not covered by the font specified in the CSS, it can still be rendered by Oxygen PDF Chemistry *fallback* fonts. To use these fallback fonts, you need to add one of the three generic font families indicated below in your CSS.

Here is the list of fallback fonts that are automatically added for these generic font families:

- **Sans-Serif** - `Segoe UI, Symbol, Segoe UI Symbol, Microsoft YaHei, Yu Gothic, Malgun Gothic, Latha, Arial Unicode MS`
- **Serif** - `Times, Times New Roman, Symbol, Segoe UI Symbol, SimSun, MingLiU, MS Mincho, Batang, Vijaya, Arial Unicode MS`
- **Monospace** - `Segoe UI, Symbol, Segoe UI Symbol, Microsoft YaHei, Yu Gothic, Malgun Gothic, Latha, Arial Unicode MS`

If you are using a Linux machine, the list of fallback fonts is the following:

- **Sans-Serif** - `DejaVu Sans, FreeSans, Noto Sans CJK SC, Noto Sans CJK JP, Noto Sans CJK KR`
- **Serif** - `Times, DejaVu Serif, FreeSerif, Noto Serif CJK SC, Noto Serif CJK JP, Noto Serif CJK KR`
- **Monospace** - `DejaVu Sans Mono, FreeMono, Noto Sans CJK SC, Noto Sans CJK JP, Noto Sans CJK KR`

If you use the processor on multiple operating systems/machines, make sure you have a consistent list of fonts installed on each of them. For instance, if you are using both a Linux and a Windows machine to run transformations in parallel and the PDF produced on Linux has some missing characters (they are replaced with the # symbol), then you can fix this by inspecting the fonts listed in the **Document Properties** dialog box in the PDF reader and install them on the Linux machine (make sure you respect the licensing terms for the font).

Note:
It is not necessary to have all the fallback fonts installed on the system.

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;
}
```

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 8)*.

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;
}
```

Note:
When installing a font on Windows, make sure you select the **Install for all users** option.

Using Local Font Files

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;
}
```

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. Using this approach there is no need to define a `@font-face` structure in the CSS.

Tip:
Using `@font-face` declarations in your CSS and keeping the font files next to the CSS is the recommended way to customize your output. In this way you can create a self-contained customization, with no need to deploy fonts in other directories.

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`. Again, 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.

Note:
When using the **Oxygen Publishing Engine**, the Chemistry processor is installed in the `[OPE_INSTALL_DIR]/plugins/com.oxygenxml.pdf.css/lib/oxygen-pdf-chemistry` folder. You will need to add the fonts in the `config/fonts` sub-folder. However, the best approach is to use `@font-face` definitions in your CSS , avoid altering the Chemistry installation.

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

Related Information:
Command Line Interface *(on page 8)*

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");
}
```

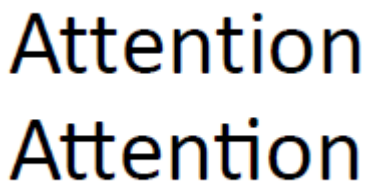
Font Ligatures

Ligatures for non-Latin scripts (such as Arabic) are enabled by default.

For Latin scripts, you have to use the `font-variant-ligatures` CSS property:

```
p {
  font-family: Calibri;
  font-variant-ligatures: common-ligatures;
}
```

This example results in the text being displayed with `font-variant-ligatures: none`:



Font Alternates

The `font-variant-alternates` CSS property controls the use of alternate glyphs, rather than the default ones.

> **Warning:**
> Currently, Oxygen PDF Chemistry supports only the `styleset()` function for `font-variant-alternates`.

Style Sets

The `styleset()` function defines a stylistic alternative for sets of characters. The parameter is a font-specific name mapped to a number. It corresponds to the OpenType value `ssXY` (e.g. `ss02`).

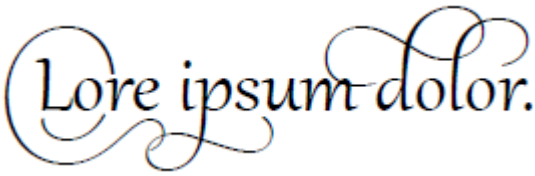
For example, if you have the following paragraph:

```
<p>Lore ipsum dolor.</p>
```

and use the following CSS customization:

```
p {
  font-family: Gabriola;
  font-variant-alternates: styleset(ss06);
}
```

it will result in the following:



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.

Important:
Before using simulated font-face variations, make sure you comply with the font licensing terms.

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

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

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

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

```
@font-face {
  font-family: "Syncro-Oxygen";
  font-style: normal;
  font-weight: 700; /* bold */
  -oxy-simulate-style: yes;
  src: url("fonts/raleway/Syncro-Oxygen-Regular.ttf");
}
@font-face {
  font-family: "Raleway";
  font-style: italic;
  font-weight: 400; /* normal */
  -oxy-simulate-style: yes;
  src: url("fonts/raleway/Syncro-Oxygen-Regular.ttf");
}
@font-face {
  font-family: "Raleway";
```

```
  font-style: italic;
  font-weight: 700; /*bold*/
  -oxy-simulate-style: yes;
  src: url("fonts/raleway/Syncro-Oxygen-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 fictitious `Syncro-Fusion` font (the `sf.ttc` file). Suppose this font provides 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: "Syncro-Fusion";
  font-style: normal;
  font-weight: 400;
  src: url("fonts/syncro/sf.ttc");
}
```

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

```
@font-face {
  font-family: "Syncro-Fusion";
  font-style: normal;
  font-weight: 700;
  -oxy-simulate-style: yes;
  src: url("fonts/syncro/sf.ttc");
}
@font-face {
  font-family: "Syncro-Fusion";
  font-style: italic;
  font-weight: 400;
  -oxy-simulate-style: yes;
  src: url("fonts/syncro/sf.ttc");
}
@font-face {
```

```
  font-family: "Syncro-Fusion";
  font-style: italic;
  font-weight: 700;
  -oxy-simulate-style: yes;
  src: url("fonts/syncro/sf.ttc");
}
@font-face {
  font-family: "Syncro-Fusion";
  font-style: normal;
  font-weight: 100;
  -oxy-simulate-style: yes;
  src: url("fonts/syncro/sf.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 97)*, makes the document accessible *(on page 56)*.

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, DejaVu Serif, Symbol, SimSun, MingLiU, MS Mincho, Batang, Vijaya, Noto Serif CJK SC, Noto Serif CJK JP, Noto Serif CJK KR, Arial Unicode MS`. This combination covers a wide character range, but the final result depends on the fonts from this list available on your OS. If a word contains a character that is not found in the current font, the fallback font list is iterated until one that supports all the word characters is found.

Support for Right-to-Left Languages

The Unicode BIDI algorithm is applied automatically. For the 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 12)* command-line parameter.

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 prefixes 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 to "*Chapitre*" for French.

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

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);
}
```

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 66)*). 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 ";
}
```

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

7.

Tutorials

This section 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.

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;
}
```

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 7)*. 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/macOS**
  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.

Full Example: How to Format a Book

This tutorial will show you how to format a book. For simplicity, the tutorial will use an HTML version of *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 "${cf}" -css "${cfd}/${cfn}.css" -out "${cfd}/${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:

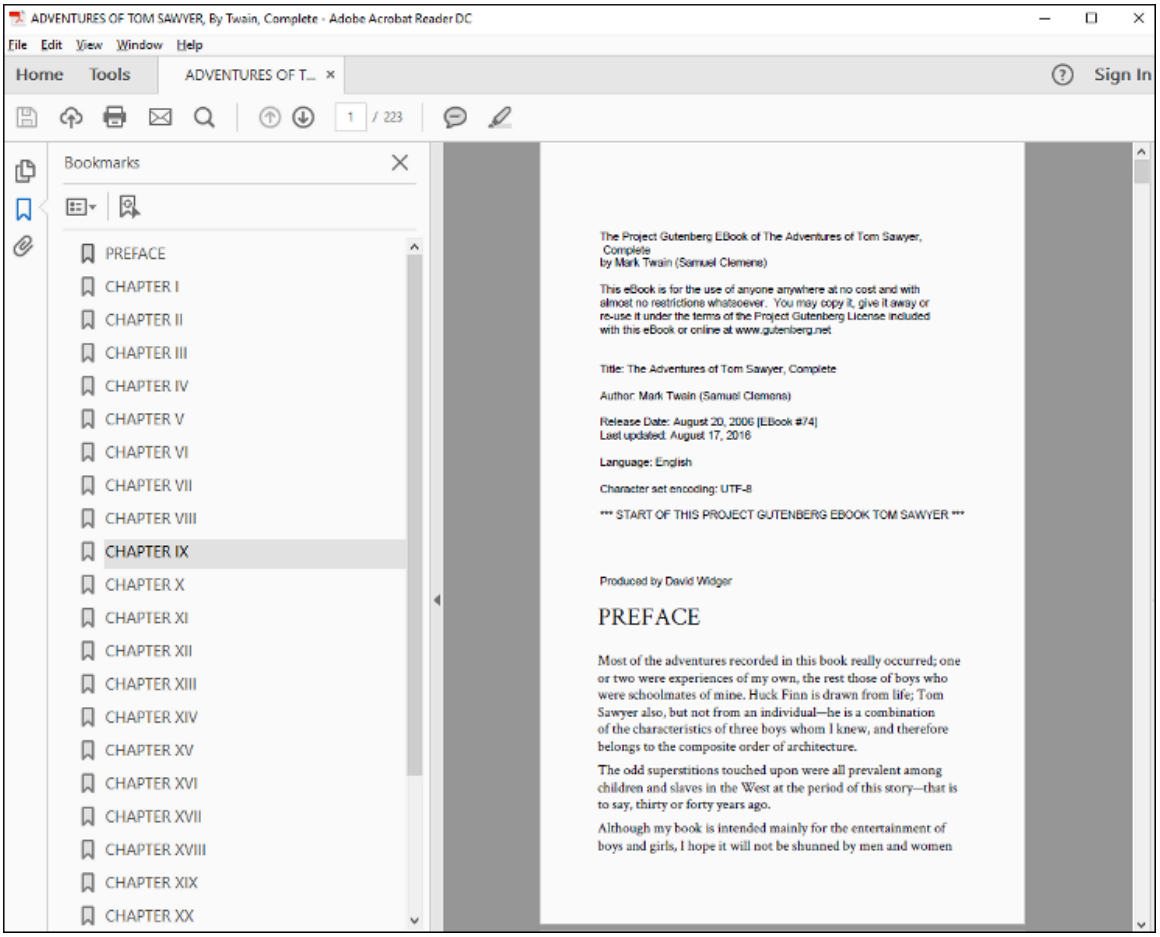
```
:root {
  font-family: 'Crimson Text', serif;
  font-size:11pt;
}
```

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 8)*. 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 {
  break-before:always;
}

body > pre {
  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;
}
```

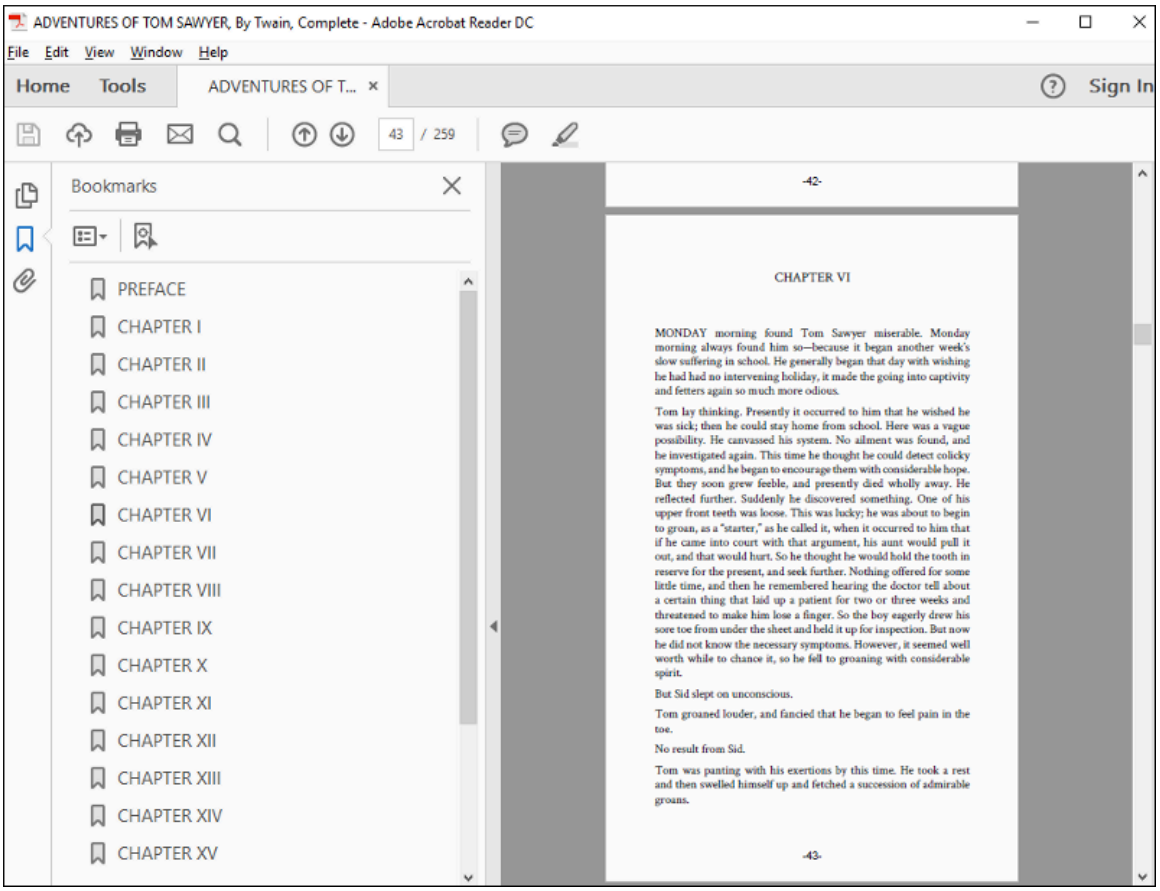
9. Add page numbers

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;
  }
}
```

Result: 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 as 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:

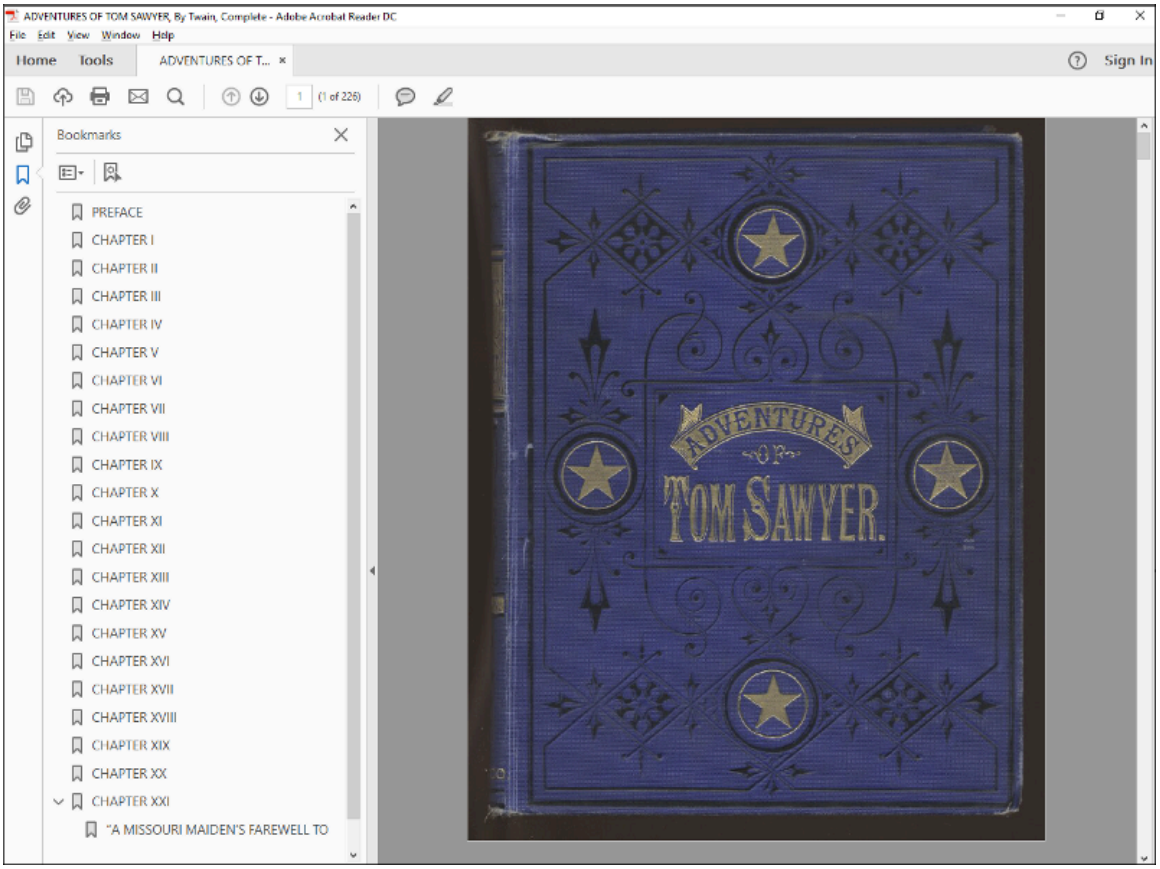
```
body > pre {
  counter-reset:page;
  page:copyright-license-page;
}
```

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:


ADVENTURES OF TOM SAWYER, By Twain, Complete - Adobe Acrobat Reader DC
File Edit View Window Help
Home Tools
ADVENTURES OF T...
Sign In
1 (1 of 226)
Bookmarks
PREFACE
CHAPTER I
CHAPTER II
CHAPTER III
CHAPTER IV
CHAPTER V
CHAPTER VI
CHAPTER VII
CHAPTER VIII
CHAPTER IX
CHAPTER X
CHAPTER XI
CHAPTER XII
CHAPTER XIII
CHAPTER XIV
CHAPTER XV
CHAPTER XVI
CHAPTER XVII
CHAPTER XVIII
CHAPTER XIX
CHAPTER XX
CHAPTER XXI
"A MISSOURI MAIDEN'S FAREWELL TO
ADVENTURES
OF
TOM SAWYER.

8.

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;
    }
}
```

Tip:
When debugging the results of the applied CSS styling, CSS selectors enclosed in `@media print {..}` are often not activated by default. There are several ways to activate them. For details, see Debugging the CSS *(on page 17)*.

CSS Selectors

CSS selectors are used to target elements in an XML document in order to style them. There are a wide variety of CSS selectors available, allowing for fine-grained precision when selecting elements to style.

CSS Supported Selectors

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

Pattern	Meaning	Described in section	CSS level
*	Any element.	Universal selector	2
E	An element of type E.	Type 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 "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 sub-string "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.myclass	An `<E>` element whose `@class` is equal to "myclass".	Class selectors	1
E#myid	An `<E>` element whose `@id` is equal to "myid".	ID selectors	1

CSS Pseudo-Classes and Pseudo-Elements

These keywords may be appended to the last sequence of simple selectors in a selector. It is possible to use one or more keywords in a single simple selector.

Pseudo-Classes

Each of these keywords specifies a special state of the selected element(s).

The following table summarizes the pseudo-classes supported by Oxygen PDF Chemistry:

Pattern	Meaning	Described in section	CSS level
E:root	An `<E>` element, root of the document.	Structural pseu-do-classes	3

Pattern	Meaning	Described in section	CSS level
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 `<E>` element in (human) language "c" (the document language specifies how the language is determined).	The :lang() pseudo-class	2
E:has(rs1)	An `<E>` element that has a relative element that matches the rs1 selector (as one of its children), when evaluated with E as the anchor element.	The :has() relational pseudo-class	4

Pseudo-Elements

Each of these keywords lets you style a specific part of the selected element(s).

The following table summarizes the pseudo-elements supported by Oxygen PDF Chemistry:

Pattern	Meaning	Described in section	CSS level
E::first-letter	The first formatted letter of an `<E>` element.	The ::first-letter pseudo-element	1
E::before	Generated content before an `<E>` element.	The ::before pseudo-element	2

Pattern	Meaning	Described in section	CSS level
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.	The ::after(n) pseudo-element	
E::after	Generated content after an `<E>` element.	The ::after pseudo-element	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.	The ::before(n) pseudo-element	
E::marker	Generated by list items to represent the item's marker (the bullet or number identifying each item). `<E>` should include `display: list-item`.	The ::marker pseudo-element	3

CSS Combinators

A selector is a chain of one or more sequences of simple selectors that can be separated by combinators. Combining simple selectors gives them a useful relationship to each other and the content location in the document.

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

Pattern	Meaning	Described in section	CSS level
E F	An `<F>` element descendant of an `<E>` element.	Descendant combinator	1
E > F	An `<F>` element child of an `<E>` element.	Child combinator	2
E + F	An `<F>` element immediately preceded by an `<E>` element.	Next-sibling combinator	2
E ~ F	An `<F>` element preceded by an `<E>` element.	Subsequent-sibling combinator	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. For more details, see MDN Web Docs: attr().

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 on 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 47)* for more examples.

string()

Retrieves the value of a string-set. The `string()` function is primarily intended to be used within Page-Margin Boxes *(on page 27)* (defined by the `@page` at-rule). See Headers and Footers *(on page 36)* for use-cases.

calc()

The `calc()` function allows mathematical expressions with addition (+), subtraction (-), multiplication (*), division (/) to be used as component values. Percentages are solved relative to the dimensions of the containing parent block. It can be used when length values are accepted:

```
elem {
  width: calc(100% - 1em);
}
```

For more information, see https://drafts.csswg.org/css-values-3/#calc-notation.

content()

Retrieves the text content from the current element. Used in a string-set definition. See Headers and Footers *(on page 36)* for use-cases.

counter()

Retrieves the value of the innermost counter of that name on the element. See 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. See 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.

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.

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. For the complete list, see: Custom CSS Functions.

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	**middle** | **-oxy-first-line**		You can use the `-oxy-first-line` value only for elements with a display value of `inline-block`. By default, the baseline of the inline block is taken from its last line (if content wraps). By using the `-oxy-first-line` value, the baseline of the inline block is taken from its first line.
background			
background-color			
background-image			
background-position			
background-repeat			
background-size	**<length>** | **<percentage>** | **initial** | **unset**		
bleed	**<length>**	**auto**	
bookmark-label			
bookmark-level			
bookmark-state			
border			
border-bottom			
border-bottom-color			

Table 2. The following standard properties are supported by Oxygen PDF Chemistry (continued)

Property	Supported Values	Unsupported Values	Notes
border-bottom-left-radius	**<length>** | **<percentage>** | **initial** | **unset**		
border-bottom-right-radius	**<length>** | **<percentage>** | **initial** | **unset**		
border-bottom-style			
border-bottom-width			
border-collapse	**separate** | **collapse** | **initial** | **unset**		
border-color			
border-left			
border-left-color			
border-left-style			
border-left-width			
border-right			
border-right-color			
border-radius	**<length>** | **<percentage>** | **initial** | **unset**		Cannot be used inside tables along with the **border-collapse** property set to `collapse`. Use **border-collapse: separate** to round the cell corners. It also has this effect on the background shape.
border-right-style			
border-right-width			

Table 2. The following standard properties are supported by Oxygen PDF Chemistry (continued)

Property	Supported Values	Unsupported Values	Notes
border-spacing	**<length>** | **initial** | **unset**		
border-style			
border-top			
border-top-color			
border-top-left-radius	**<length>** | **<percentage>** | **initial** | **unset**		
border-top-right-radius	**<length>** | **<percentage>** | **initial** | **unset**		
border-top-style			
border-top-width			
border-width			
bottom			
break-after	**auto** | **always** | **all** | **avoid** | **right** | **left** | **column** | **page** | **avoid-column** | **avoid-page** | **initial**	**recto** | **verso** | **region** | **avoid-region**	
break-before	**auto** | **always** | **all** | **avoid** | **right** | **left** | **column** | **page** | **avoid-column** | **avoid-page** | **initial**	**recto** | **verso** | **region** | **avoid-region**	
break-inside	**auto** | **avoid** | **initial**	**avoid-page** | **avoid-column** | **avoid-region**	
caption-side	**top** | **bottom** | **initial** | **unset**	**left** | **right**	

Table 2. The following standard properties are supported by Oxygen PDF Chemistry (continued)

Property	Supported Values	Unsupported Values	Notes
color			
column-span			
content			
counter-increment			• Counters do not add up: defining new counters on an element already defining a counter will reset the others. • Counters are "self-nesting": a counter defined in a parent can be reused in a child of this parent, read more at Nested Counters and Scope.
counter-reset			
direction			
display	**inline** | **block** | **inline-block** | **table** | **table-row** | **table-cell** | **table-caption** | **table-row-group** | **table-header-group** | **table-footer-group** | **table-column** | **-oxy-morph** | **initial** | **unset**		For `display:none`, the elements are not eliminated from the DOM. They are just collapsed. The text may remain in the document, but is not visible. This is needed for keeping the links working. For **-oxy-morph**, read more here.
empty-cells			
float			
font			
font-family			
font-size			
font-src			
font-style			

Table 2. The following standard properties are supported by Oxygen PDF Chemistry (continued)

Property	Supported Values	Unsupported Values	Notes
font-variant	**small-caps** | **normal** | **initial** | **unset**		Cannot be combined with the **:first-letter** pseudo-element or **overflow-wrap: break-word** CSS property.
font-variant-alternates	**styleset** | **initial** | **unset**	**normal** | **historical-forms** | **stylistic** | **character-variant** | **swash** | **ornaments** | **annotation** | **swash**	Can be used to map a different stylistic alternative for sets of characters. The parameter is a font-specific name mapped to a number. It corresponds to the OpenType value `ssXY` (e.g. `ss02`). Depends on the fonts capabilities, and applies to open type fonts.
font-variant-ligatures	**normal** | **none** | **common-ligatures** | **no-common-ligatures** | **discretionary-ligatures** | **no-discretionary-ligatures** | **historical-ligatures** | **no-historical-ligatures** | **contextual** | **no-contextual** | **initial** | **unset**		Controls which ligatures and contextual forms are used in styled text. Depends on the fonts capabilities and applies to open type fonts.
font-weight			
font-variant-numeric	**normal** | **none** | **lining-nums** | **oldstyle-nums** | **proportional-nums** | **tabular-nums** | **diagonal-fractions** | **stacked-fractions** | **ordinal** | **slashed-zero** | **initial** | **unset**		Controls the usage of alternate glyphs for numbers, fractions, and ordinal markers.
height			

Table 2. The following standard properties are supported by Oxygen PDF Chemistry (continued)

Property	Supported Values	Unsupported Values	Notes
hyphens	**auto** | **manual** | **initial** | **unset**		
image-resolution	**<dpi>**		
left			
letter-spacing			Discards the font ligatures.
line-height			
list-style			
list-style-image	**none** | **url()** | **initial** | **unset**		
list-style-position			
list-style-type	**<string>** | **box** | **check** | **circle** | **diamond** | **disc** | **hyphen** | **square** | **none** | **decimal** | **lower-roman** | **lower-latin** | **upper-roman** | **upper-latin** | **decimal-leading-zero**		
margin			
margin-bottom	**<length>** | **initial** | **unset**		
margin-left	**<length>** | **auto** | **initial** | **unset**		
margin-right	**<length>** | **auto** | **initial** | **unset**		
margin-top	**<length>** | **initial** | **unset**		
marks	**none** | **crop** | **cross**		

Table 2. The following standard properties are supported by Oxygen PDF Chemistry (continued)

Property	Supported Values	Unsupported Values	Notes
max-height	**<length>** | **initial** | **unset**		
max-width	**<length>** | **initial** | **unset**		
min-height	**<length>** | **initial** | **unset**		
min-width	**<length>** | **initial** | **unset**		Not supported for inline blocks and blocks.
orphans	**<number>** | **initial** | **unset**		As a difference from the CSS standard, the `orphans` property is also applied to lists, as well as paragraphs. This means that a list that spans consecutive pages will have either zero or at least two lines (not necessarily items) on the first page.
outline			
outline-color			
outline-style			
outline-width			
overflow-wrap	**break-word** | **initial** | **unset**		Limitations: Should be inserted on parent block elements (including tables) but will be applied on child inline elements. Not supported for justified or hyphenated text paragraphs.
padding			
padding-bottom			
padding-left			
padding-right			
padding-top			
page			
page-break-after	**auto** | **always** | **avoid** | **left** | **right** | **initial**		

Table 2. The following standard properties are supported by Oxygen PDF Chemistry (continued)

Property	Supported Values	Unsupported Values	Notes
page-break-before	**auto** | **always** | **avoid** | **left** | **right** | **initial**		
page-break-inside	**auto** | **avoid** | **initial**		
position	**fixed** | **relative** | **absolute** | **initial** | **unset**		The **absolute** and **fixed** values are not supported for inline elements.
right			
size			
string-set	**attr()** | **content()** | **counter()**		• String-sets do not add up: defining new string-sets on an element already containing string-sets definitions will reset the others. • Using `""` as string-set value will stop the propagation the past value to new elements.
table-column-span			
table-layout	**fixed** | **auto** | **initial** | **unset**		
table-row-span			
text-align	**start** | **end** | **left** | **right** | **center** | **justify** | **justify-all** | **initial** | **unset**		
text-decoration			
text-decoration-line			
text-decoration-style	**solid**	**double** | **dotted** | **dashed** | **wavy**	

Table 2. The following standard properties are supported by Oxygen PDF Chemistry (continued)

Property	Supported Values	Unsupported Values	Notes
text-indent			
text-transform			
top			
transform			
transform-rotation			
unicode-bidi			
vertical-align	**baseline** | **sub** | **super** | **text-top** | **text-bottom** | **middle** | **top** | **bottom** | **initial** | **unset**	**<length>** | **<percentage>**	
visibility			
white-space	**normal** | **nowrap** | **pre** | **pre-wrap** | **initial** | **unset**		
widows	**<number>** | **initial** | **unset**		As a difference from the CSS standard, the `widows` property is also applied to lists, as well as paragraphs. This means that a list that spans consecutive pages will have either zero or at least two lines (not necessarily items) on the next page.
width	**<length>** | **<percentage>** | **calc** | **fill** | **initial** | **unset**		The `fill` value can be used only when the matched element has the display value set to `table-cell` or `table-column` and the parent table has the `table-layout: auto` value. It will keep all columns to their minimums, distributing the rest of the space to the ones marked as fillable (or containing fillable cells).

Note:
The **initial** and **unset** values are not supported when applied to Page Margin Boxes *(on page 27)*.

Extension CSS Properties

Table 3. The following extension properties are supported by Oxygen PDF Chemistry

Property	Description	Supported Values	Default Value	Inheritable

-oxy-alt-text	Used to specify an alternative description for the element that is used by the PDF readers.
	`image {`
	`  -oxy-alt-text: "Image about: " attr(href);`
	`}`
		<string> | **initial** | **unset**		**no**

-oxy-avoid-breaking-line-at-hyphens	This can be used to avoid a line break inside a word that contains a hard hyphen. This property is deprecated and has been replaced by -oxy-break-line-at-hyphens *(on page 134)*.	**yes** | **no**	**no**	**yes**

-oxy-borders-conditionality	This can be used to configure whether or not bottom and top borders appear on a split element when a page break occurs. Supported values are:
	• **discard** - Makes the borders disappear.
	• **retain** - Forces the bottom and top border to be displayed.
	• **inherit** - Inherits the value specified in the parent element.
	`td {`
	`  -oxy-borders-conditionality: retain;`
	`}`
		discard | **retain** | **inherit** | **initial** | **unset**	**discard**	**no**

-oxy-break-line-at-hyphens	This can be used to set the line break inside a word that contains a hard hyphen. Supported values are: • **auto** - Let the default hyphenation algorithm process the words • **avoid** - Forces the algorithm not to break inside a word at hard hyphen • **always** - Forces the algorithm to break inside a word at hard hyphen	**auto** | **avoid** | **always**	**auto**	**yes**

Table 3. The following extension properties are supported by Oxygen PDF Chemistry (continued)

Property	Description	Supported Values	Default Value	Inheritable
	Note: Breaking the line at a hard hyphen can cause the PDF readers to join the lines but discard the hyphen character when copying and pasting content.			
-oxy-caption-repeat-on-next-pages	This can be used to enable table caption repetition on subsequent pages (when the table is long and it spans multiple pages). It only affects elements with the `table-caption` display. Also see the -oxy-show-only-when-caption-repeated-on-next-pages *(on page 144)* property that can be applied on optional caption static content.	**yes** | **no**	**no**	**yes**
-oxy-changebar-offset	Defines the distance between the text and the change bar. A positive distance is directed away from the column region and into the margin regardless of the **-oxy-change-bar-placement**. A negative distance is directed towards the content area.	**<length>** | **initial** | **unset**	**6pt**	**yes**
-oxy-changebar-placement	This property determines where, relative to the column areas, the change bars will occur.	**start** | **end** | **left** | **right** | **inside** | **outside** | **alternate** | **initial** | **unset**	**left**	**yes**
-oxy-changebar-color	Specifies the color of the change bar.	**<color>** | **initial** | **unset**	**black**	**yes**
-oxy-changebar-style	Specifies the style of the change bar.	**<border-style>** | **initial** | **unset**	**solid**	**yes**
-oxy-changebar-width	Specifies the thickness of the change bar.	**<border-width>** | **initial** | **unset**	**medium**	**yes**

Table 3. The following extension properties are supported by Oxygen PDF Chemistry (continued)

Property	Description	Supported Values	Default Value	Inheritable
-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: `always` - to force the column break, or `avoid` - to keep the end of the matching element together with the next sibling beginning in the same column.	**always** | **auto** | **avoid** | **initial** | **unset**	**auto**	**no**
-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: `always` - to force the column break, or `avoid` - to keep the end of the previous sibling together with the beginning of the matching element in the same column.	**always** | **auto** | **avoid** | **initial** | **unset**	**auto**	**no**
-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.	**always** | **auto** | **avoid** | **initial** | **unset**	**auto**	**no**
-oxy-crop-box	Specifies the region where the content of the page will be cropped when displayed or printed.	**trim-box** | **bleed-box** | **media-box**	**media-box**	**no**
-oxy-crop-offset	Specifies an extended area outside the original region where the content of the page will be cropped when displayed or printed. Up to 4 values can be set (`top`, `right`, `bottom`, `left`), separated by spaces.	**<length>**		**no**

-oxy-force-page-count	Used to impose a constraint on the number of pages in a page sequence. It may increase the total number of pages from the page sequence by adding blank pages.
	`@page chapter-page {`
	`-oxy-force-page-count: end-on-even;`
	`}`
	auto
	Force the last page in this page-sequence to be an odd-page if the initial-page-number of the next page-sequence is even. Force it to be an even-page if the initial-page-number of the next page-sequence is odd.
		auto | **even** | **odd** | **end-on-even** | **end-on-odd** | **no-force** | **inherit** | **initial** | **unset**	**auto**	**no**

Table 3. The following extension properties are supported by Oxygen PDF Chemistry (continued)

Property	Description	Supported Values	Default Value	Inheritable
	If there is no next page-sequence or if the value of its initial-page-number is "auto", do not force any page. **even** Force an even number of pages in this page-sequence. **odd** Force an odd number of pages in this page-sequence. **end-on-even** Force the last page in this page-sequence to be an even-page. **end-on-odd** Force the last page in this page-sequence to be an odd-page. **no-force** Do not force either an even or an odd number of pages in this page-sequence.			
-oxy-foreground-image	This can be used to define a foreground image that will be displayed above the content, like a watermark.	**url()** | **none** | **inherit** | **initial** | **unset**		**no**
-oxy-hyphenation-character	Used to specify the Unicode character to be presented when a hyphenation break occurs. To hide hyphens, use the space character. `pre { -oxy-hyphenation-character: " "; }`	**<character>** | **inherit** | **initial** | **unset**	**U+00AD**	**yes**
-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** | **initial** | **unset**	**2**	**yes**

Table 3. The following extension properties are supported by Oxygen PDF Chemistry (continued)

Property	Description	Supported Values	Default Value	Inheritable
	`p { -oxy-hyphenation-push-character-count: "2"; }`			
-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). `p { -oxy-hyphenation-remain-character-count: "2"; }`	**<number>** | **inherit** | **initial** | **unset**	**2**	**yes**
-oxy-initial-page-number	Used to set the initial page number to be used on this particular page sequence. `@page chapter-page { -oxy-initial-page-number: auto-odd; }`	**<number>** |**auto** | **auto-odd** | **auto-even** | **inherit** | **initial** | **unset**	**auto**	**no**
-oxy-link	Used to create links in the PDF output. `link { -oxy-link: attr(href); }`	**none** | **url()** | **initial** | **unset**		**no**
-oxy-page-group	Using the **start** value forces the creation of a new page group (sequence) even if the page before the element has the same name. The **auto** value uses the W3C algorithm normally, which would merge the current element with the open page sequence. `chapter { -oxy-page-group:start; page: chapter; }`	**start** | **auto** | **initial** | **unset**	**auto**	**no**
-oxy-pdf-extract-file	Used to generate individual fragments of a PDF as separate files, with the file name specified by the property value.			**no**

Table 3. The following extension properties are supported by Oxygen PDF Chemistry (continued)

Property	Description	Supported Values	Default Value	Inheritable

	``
	article[id] {
	  -oxy-pdf-extract-file: attr(id);
	}
	``			

-oxy-pdf-meta-author	Used to forward meta-information to the PDF. Represents the author of the publication.
	``
	meta[name='author'] {
	  -oxy-pdf-meta-author: attr(value);
	}
	``
		<string> | **initial** | **unset**		**no**

-oxy-pdf-meta-custom	Used to forward meta-information to the PDF. Represents a generic custom document property. Should have two strings, the name and the value.
	``
	metadata {
	  -oxy-pdf-meta-custom: attr(name) attr(value);
	}
	``
		<string> | **initial** | **unset**		**no**

-oxy-pdf-meta-copyright	Specifies the copyright text.
	``
	meta[name='copyright'] {
	    -oxy-pdf-meta-copyright: attr(value);
	}
	``
		<string> | **initial** | **unset**		**no**

-oxy-pdf-meta-copyrighted	Specifies whether or not the publication is copyrighted or from public domain.
	``
	:root {
	    -oxy-pdf-meta-copyright: copyrighted;
	}
	``
		copyrighted | **public-domain** | **initial** | **unset**		**no**

-oxy-pdf-meta-copyright-url	Specifies the location of the external copyright notice.
	``
	meta[name='copyright-url'] {
	    -oxy-pdf-meta-copyright-url: attr(value);
	}
	``
		<string> | **initial** | **unset**		**no**

-oxy-pdf-meta-description	Used to forward meta-information to the PDF. Represents the description of the publication.	**<string>** | **initial** | **unset**		**no**

Table 3. The following extension properties are supported by Oxygen PDF Chemistry (continued)

Property	Description	Supported Values	Default Value	Inheritable
	`meta[name='description'] { -oxy-pdf-meta-description: attr(value); }`			
-oxy-pdf-meta-keyword	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. `keyword { -oxy-pdf-meta-keyword: content(text); }`	**<string>** | **initial** | **unset**		**no**
-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. `meta[name='keywords'] { -oxy-pdf-meta-keywords: attr(value); }`	**<string>** | **initial** | **unset**		**no**
-oxy-pdf-meta-title	Used to forward meta-information to the PDF. Represents the title of the publication. `meta[name='title'] { -oxy-pdf-meta-title: attr(value); }`	**<string>** | **initial** | **unset**		**no**
-oxy-pdf-table-omit-footer-at-break	Flag that indicates if the table footer should be omitted after a page break.	**true** | **false** | **inherit** | **initial** | **unset**	**false**	**no**
-oxy-pdf-table-omit-header-at-break	Flag that indicates if the table header should be omitted after a page break.	**true** | **false** | **inherit** | **initial** | **unset**	**false**	**no**
-oxy-pdf-viewer-display-filename	Flag that indicates whether the PDF viewer should display the filename (**true**) or the document title (**false**) for the generated PDF.	**true** | **false** | **initial** | **unset**	**false**	**no**

Table 3. The following extension properties are supported by Oxygen PDF Chemistry (continued)

Property	Description	Supported Values	Default Value	Inheritable
	```:root { -oxy-pdf-viewer-display-filename: true; }```			
-oxy-pdf-viewer-fit-window	Flag that specifies whether or not to resize the document's window to fit it to the size of the first displayed page. ```:root { -oxy-pdf-viewer-fit-window: true; }```	**true** | **false** | **initial** | **unset**	**false**	**no**
-oxy-pdf-viewer-hide-menubar	Flag that indicates if the PDF viewer should hide its menu bar when opening the PDF document. ```:root { -oxy-pdf-viewer-hide-menubar: true; }```	**true** | **false** | **initial** | **unset**	**false**	**no**
-oxy-pdf-viewer-hide-toolbar	Flag that indicates whether or not the PDF viewer should hide its toolbar when opening the PDF document. ```:root { -oxy-pdf-viewer-hide-toolbar: true; }```	**true** | **false** | **initial** | **unset**	**false**	**no**
-oxy-pdf-viewer-page-layout	Specifies the page layout to be used when the document is opened. ```:root { -oxy-pdf-viewer-page-layout: one-column; }``` **single-page** Displays one page at a time. **one-column** Displays the pages in one column. **two-column-left** Displays the pages in two columns, with odd-numbered pages on the left. **two-column-right**	**single-page** | **one-column** | **two-column-left** | **two-column-right** | **two-page-left** | **two-page-right** | **initial** | **unset**		**no**

Table 3. The following extension properties are supported by Oxygen PDF Chemistry (continued)

Property	Description	Supported Values	Default Value	Inheritable
	Displays the pages in two columns, with odd-numbered pages on the right. **two-page-left** Displays the pages two at a time, with odd-numbered pages on the left. **two-page-right** Displays the pages two at a time, with odd-numbered pages on the right. **Note:** These values are similar to those from the PDF specification (the "Document Catalog/Page Layout" section).			
-oxy-pdf-viewer-page-mode	Specifies which left-side view should be visible when the PDF document is opened, or if the document should be full screen. The `use-outlines` value refers to the bookmark tree. `:root { -oxy-pdf-viewer-page-mode: use-outlines; }` **use-none** Neither document outlines nor thumbnail images are visible. **use-outlines** Document outlines (bookmarks) are visible. **use-thumbs** Thumbnail images are visible. **full-screen** Full-screen mode, with no menu bar, window controls, or any other window visible.	**use-none** | **use-outlines** | **use-thumbs** | **full-screen** | **initial** | **unset**		**no**

Table 3. The following extension properties are supported by Oxygen PDF Chemistry (continued)

Property	Description	Supported Values	Default Value	Inheritable
	Note: These values are similar to those from the PDF specification (the "Document Catalog/Page Mode" section).			

-oxy-pdf-viewer-zoom	A percentage or a fit value that indicates the zoom factor or the fitting factor of the PDF viewer.
	`:root { -oxy-pdf-viewer-zoom: 125%; }`
	`:root { -oxy-pdf-viewer-zoom: fit-height; }`
		<percentage> | fit-page | fit-width | fit-height | initial | unset		**no**

-oxy-pdf-tag-type	Maps an element to a PDF accessibility tag. `section { -oxy-pdf-tag-type: "Sect"; }`	**<string> | initial | unset**		**no**
-oxy-scale	Specifies a scale factor (on both the x and y axis) for the PDF output. If only one number is provided, it is used for both the x and y scales. A scale factor smaller than 1 shrinks the page. A scale factor greater than 1 enlarges the page.	**<number>**		**no**
-oxy-simulate-style	Set this flag to **yes** when the styles (bold, italic, or both) need to be simulated from a regular font. See: Using Simulated (Synthetic) Styles *(on page 105)*.	**yes | no**	**no**	
-oxy-space-after-conditionality	Specifies whether the bottom margin (`margin-bottom` property) of an element should be kept or discarded.	**discard | retain | inherit | initial | unset**	**discard**	**no**
-oxy-space-before-conditionality	Specifies whether the top margin (`margin-top` property) of an element should be kept or discarded.	**discard | retain | inherit | initial | unset**	**discard**	**no**

Table 3. The following extension properties are supported by Oxygen PDF Chemistry (continued)

Property	Description	Supported Values	Default Value	Inheritable

-oxy-style	An Oxygen extension used to define additional styles for an element (for example, collected from an attribute).
	`div { font-weight:bold; color:red; -oxy-style:attr(style); }`
	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 `color:green; border: 1pt solid red`, it will combine with the existing properties, resulting in:
	`font-weight:bold; color:green; border: 1pt solid red`
		CSS fragment		**no**

-oxy-show-only-when-caption-repeated-on-next-pages	This property only affects a `:before` or `:after` pseudo element that is associated with a table caption (an element with the table `caption` display marked with `-oxy-caption-repeat-on-next-pages:yes`). It signals that the static content is to be used only when the caption is displayed the second time (on the subsequent pages), when the table is long and it spans multiple pages. Also see: -oxy-caption-repeat-on-next-pages *(on page 135)*.	**yes** | **no**		**no**
-oxy-video-cover	This property can be used to set a cover image on videos. This image will appear as a placeholder in the PDF output. If it is not set, a default cover is used.	**none** | **url()** | **initial** | **unset**		**no**

Note:
The **initial** and **unset** values are not supported when applied to Page Margin Boxes *(on page 27)*.

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 in regards to 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 21.
- Use the **-Xmx<NNN>m** command-line parameter *(on page 15)* 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**).

Copying and Pasting Text from the PDF Reader does not Preserve Line Breaks

The generated PDF is accessible. This causes some of the readers (such as Acrobat Reader) to change the way the lines are saved in the clipboard. The effect is that codeblock lines may be joined when pasting them into a text editor.

One solution is to disable the accessibility feature. To do this, edit the `config/chemistry-fop.xconf` file and change the `<accessibility>` element to **false**:

```
<accessibility>false</accessibility>
```

Index

Special Characters

:after pseudo-element
66
:before pseudo-element
66

A

Accessibility (508 compliance
56
Aligning blocks
79
Anchors
56
Archiving
59
Associating a CSS to a document
23
Avoid line breaks at hyphens
64

B

Bookmarks
51

C

Change Bars
67
Command-line arguments
8
Configuration file
145
Configuring PDF Chemistry as an external tool
8
Cross references
45
CSS
combinators
122
pseudo classes
120
pseudo elements
120
CSS functions
122
CSS media rules
119
CSS properties
125
CSS selectors
119
CSS units
124

D

Debugging
17
DITA to PDF from a command line
15

E

Element layout
69
Embedding fonts
100
Ending chapters on odd/even page
34
Error messages
145

F

Font Variant
104
Fonts
97
Footer
Dynamic images
41
Footers
36
Footnotes
43
Force line breaks at hyphens
64

G

Getting started
6

Graphics
82
Background images
85
Foreground images
86
Image maps
93
Images
82
Image resolution
84
Image size
83
Supported image types
82
MathML
92
Supported colors
87
SVG
87
SVG Templates
89
Videos
94
H
Header
Dynamic images
41
Style text
40
Headers
36
Hyphenation
60, 64
I
Insert current date
111
Installing PDF Chemistry
7
Internationalization
108
Introduction
6
J
Java security policy
21
L
Ligatures
104
Lists
76
Localization
109
Logging
20
M
Metadata
53
Multiple lines in footer
39
Multiple lines in header
39
N
Named destinations
56
O
OutOfMemory error
145
P
Page breaks
32
Page formatting
23
Processing PDF Chemistry from command line
8
R
Right to left languages
108
Rotating blocks
80
S
Signing

59
Starting chapters on odd/even page
34
Styling for print
23
Styling the first page of a chapter
34

T

Table of contents creation
47
Tables
69
Technical issues
145
Transform multiple files
111
Tutorial
112

W

Web fonts
98
What is Chemistry
6

X

XML support
145
XPath in CSS
65

Copyright

Oxygen PDF Chemistry User Manual

Syncro Soft SRL.

Copyright © 2002-2023 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.