



REALOBJECTS

PDFreactor[®]

MANUAL

Version 7.0.7447.1

Copyright (c) 2006-2015 RealObjects

PDFreactor is a registered trademark of RealObjects GmbH.

TABLE OF CONTENTS

Introduction.....	7
1. Installation.....	8
1.1. The PDFreactor Library	8
1.2. The PDFreactor Service.....	8
1.3. Jetty	9
1.4. PHP Requirements	9
1.5. .NET Requirements.....	9
1.6. Perl/Python/Ruby Requirements	9
2. Integration.....	10
2.1. Memory	10
2.2. Using the Java library	10
2.2.1. Using PDFreactor in a Servlet.....	11
2.2.2. Logging	11
2.3. Running PDFreactor Without Graphics Environment	12
2.4. Using a Wrapper.....	12
2.4.1. Using PHP	13
2.4.2. Using .NET.....	13
2.4.3. Using Python	14
2.4.4. Using Perl.....	15
2.4.5. Using Ruby	15
2.4.6. Jetty Configuration	16
2.4.7. Logging.....	18
2.5. Using the Command Line Interface	19
2.5.1. Quick Introduction to the Command Line Interface.....	19
2.5.2. Advanced Command Line Options.....	20
2.5.3. Command Line Configuration.....	20
2.6. License Key	21
2.6.1. Evaluation Mode	21
2.6.2. Receiving a License Key	21
2.6.3. Setting the License Key.....	21
3. Input Formats.....	22
3.1. HTML + CSS	22
3.2. XML + CSS.....	23
3.3. Compound Formats.....	23
3.3.1. Images	23
3.3.2. SVG	24
3.3.3. Barcode.....	25
3.3.4. MathML.....	25
3.3.5. QR Code	25
3.3.6. Object and Embed	26
3.3.7. iframes.....	26
3.3.8. Canvas Element	28
3.4. JavaScript.....	29
3.4.1. Proprietary Functions and Members	30
3.4.2. awesomizr.js	32
4. Output Formats.....	33
4.1. PDF.....	33
4.1.1. Bookmarks	33
4.1.2. Links	34

4.1.3. Comments.....	34
4.1.4. Metadata.....	37
4.1.5. Interactive PDF Forms.....	38
4.1.6. Tagged PDF.....	38
4.1.7. PDF/A Conformance.....	39
4.1.8. Print Dialog Prompt.....	41
4.1.9. PDF Compression.....	41
4.1.10. Encryption and Restrictions.....	41
4.1.11. Page Preview Images.....	42
4.1.12. Viewer Preferences.....	42
4.1.13. Merging PDFs.....	44
4.1.14. Digital Signing.....	44
4.1.15. Font Embedding.....	45
4.1.16. Overprinting.....	45
4.1.17. Attachments.....	46
4.2. Images.....	47
4.2.1. Selecting a page.....	47
4.2.2. Converting a Document Into Multiple Images.....	47
4.2.3. Continuous Output.....	47
5. Layout Documents.....	49
5.1. Basic Page Layout.....	49
5.1.1. Page Selectors.....	49
5.1.2. Page Size & Orientation.....	50
5.1.3. Named Pages.....	50
5.2. Breaking Text.....	50
5.2.1. Automatic Hyphenation.....	51
5.2.2. Widows & Orphans.....	51
5.3. Generated Content.....	52
5.3.1. Generated Text.....	52
5.3.2. Generated Images.....	53
5.3.3. Counters.....	53
5.4. Page Header & Footer.....	53
5.4.1. Header, Footer & Page Side Boxes.....	53
5.4.2. Running Elements.....	54
5.4.3. Running Documents.....	57
5.5. Generated Content for Pages.....	57
5.5.1. Page Counters.....	57
5.5.2. Named Strings.....	58
5.5.3. Cross-references.....	59
5.5.4. Footnotes.....	61
5.6. Transforms.....	62
5.6.1. 2D Transforms.....	62
5.7. Multi-column Layout.....	63
5.8. Region Layout.....	65
5.8.1. Adding Regions to Region Chains.....	65
5.8.2. Adding Content to a Named Flow.....	66
5.8.3. Region Generated Content.....	67
5.9. Breaking Boxes.....	67
5.9.1. Breaking Around Boxes.....	67
5.9.2. Avoid Breaking Inside Boxes.....	68
5.9.3. Adaptive Page Breaks.....	68
5.10. Advanced Page Layout.....	68
5.10.1. PDF Page Boxes.....	69

5.10.2. Printer Marks.....	70
5.11. Leaders	71
5.12. Table of Contents	72
5.13. Shrink-to-Fit.....	74
5.13.1. The method <code>setPixelsPerInchShrinkToFit</code>	74
5.13.2. The property <code>-ro-scale-content</code>	74
5.14. Page Order	74
5.15. Pages Per Sheet.....	75
5.16. Booklet.....	76
5.17. Pixels per Inch.....	76
6. PDFreactor Cookbook	77
6.1. How Do I Create Running Table Headers?	77
6.2. How Do I Set CSS & XSLT Stylesheets?.....	77
6.3. How Do I Handle Relative References?.....	78
6.4. How Do I Set Styles for Print or Screen Only?	79
6.5. Automatic Resizing of Form Controls	79
6.6. How Do I Set Colors in CSS?	79
6.7. How Do I Resize Background Images?	80
6.8. How Do I Create Rounded Corners?	81
6.9. How Do I Place an Image in the Header?	81
6.10. How Do I Use HTML in Headers and Footers?.....	81
6.11. How Do I Create a Document With a Text Direction of Right-to-Left?.....	81
6.12. How Do I Save Memory if a Document Refers to Many or Very Large Image Files?	82
6.13. How can I determine programatically that content fits into its pages.....	82
6.14. How Can I Retrieve the Number of Pages of a Converted Document Programatically?.....	83
6.15. How Do I Access Resources That Are Secured Via Basic or Digest Authentication?.....	83
6.16. How Can I Set Request Headers And Cookies For The Outgoing Connections of PDFreactor?	83
6.17. How Can I Add a Smooth Color Transition to the Background of an Element?	83
6.18. How Can I Rotate Text by 90 Degrees?	84
A. Fonts	85
A.1. Font Sources.....	85
A.1.1. Core Fonts Pack	85
A.1.2. System and JVM Font Directories.....	85
A.1.3. Additional Fonts & Font Directories	86
A.1.4. CSS Defined Fonts	86
A.2. Font Installation.....	86
A.2.1. Recommended Fonts	87
A.2.2. Font Installation on Windows.....	87
A.2.3. Font Installation on Linux.....	87
A.2.4. Font Installation on Mac OS X.....	87
A.3. The Font Cache Mechanism	87
A.3.1. Controlling the Font Registration and Caching Mechanism	88
A.4. Font Matching	88
A.4.1. Matching CSS Font Families	88
A.4.2. Font Alias Names	88
A.4.3. Automatic Font Fallback	88
B. CSS Support	89
B.1. CSS Pseudo-elements and -classes	89
B.2. CSS Attribute Selector	89
B.3. CSS At-Rules	90
B.4. Supported Page Size Formats.....	90
B.5. Hyphenation Dictionaries	91
B.6. Supported length units.....	92

B.7. CSS Color Keywords 92

B.8. Counter and Ordered List Style Types 97

B.9. CSS Property Documentation 101

INTRODUCTION

RealObjects PDFReactor is a powerful formatting processor that enables server-side PDF creation from HTML¹ and XML² documents using CSS³ to define page layout and styles. You can dynamically generate PDF⁴ documents such as reports, invoices, statements and others on-the-fly. Since PDFReactor runs on your server, the end-user does not need any software other than a Web browser with a free PDF viewer.

By implementing the W3C⁵ specification for Paged Media and print, PDFReactor provides great control over the paged output via CSS. It is very easy to set page sizes, margins and pagination for specific elements, page headers and footers. Using proprietary configuration and CSS properties, it is possible to set PDF specific information, to generate outlines and links for your documents and even to add meta information about the author or keywords.

PDFReactor is provided as a Java library. The PDFReactor Java API⁶ is perfectly complemented by .NET, PHP, Perl, Python and Ruby APIs which enable a smooth cooperation of PDFReactor and your web applications. Being an integrator or developer with focus on Java, Servlet, EJB, CGI, .NET, PHP, Perl, Python or Ruby programming, you can easily integrate PDFReactor into your servlet, EJB⁷ or Web Service by using the comprehensive APIs at your disposal.

In the simplest case you only have to specify an input and output file to generate a PDF document. It is also possible to use the operating system's standard input (stdin) and output (stdout) streams to read and write content.

The typical integration scenario for PDFReactor is Server-side PDF creation as part of large host applications (e.g. ERP⁸, database, e-commerce) on a web application server in order to print or stream PDF documents - generated from HTML or XML sources - over the Web.

¹ Hypertext Markup Language (<http://www.w3.org/html/>)

² Extensible Markup Language (<http://www.w3.org/XML/>)

³ Cascading Style Sheets (<http://www.w3.org/Style/CSS/>)

⁴ Portable Document Format (http://www.adobe.com/devnet/pdf/pdf_reference.html)

⁵ The World Wide Web Consortium (<http://www.w3.org/>)

⁶ Application Programming Interface

⁷ Enterprise JavaBeans (<http://www.oracle.com/technetwork/java/javasee/ejb/>)

⁸ Enterprise Resource Planning, a system that is used to manage and coordinate all the resources of a business

1. INSTALLATION

PDFReactor can be deployed as a Java library, command line tool, or can be used with a .NET, PHP, Perl, Python or Ruby API. If it is used as a Java library or command line, no further installation is required.

Note:

For details about system requirements and information about the latest changes, please see the `readme.txt` contained within the PDFReactor installation package.

However, if the .NET⁹, PHP¹⁰, Perl, Python or Ruby APIs are used, an application server running PDFReactor as a service is required. PDFReactor is delivered with a pre-configured version of Jetty, which enables you to quickly deploy PDFReactor as a service on your server.

This chapter details the system requirements that must be met in order to run PDFReactor as a service.

1.1 The PDFReactor Library

The PDFReactor package comes with two PDFReactor libraries located in the "PDFReactor/libs" directory:

- `pdfreactor.jar`
- `pdfreactorcore.jar`

It is generally recommended to use the "pdfreactor.jar", since it not only contains PDFReactor itself but also all 3rd party libraries required by PDFReactor. This JAR¹¹ file is a stand-alone PDFReactor library. No other libraries are required.

If some of the 3rd party libraries are already installed on the server or if certain functionality is not required, the "pdfreactorcore.jar" can be used. It only contains PDFReactor, optional 3rd party libraries contained in the "3rdparty" directory should be added to the PDFReactor class path manually depending on whether or not they are already installed on the server or their functionality is desired.

Note:

Please refer to the "libs.txt" in the "PDFReactor/libs/3rdparty" directory for more information about the 3rd party libraries.

1.2 The PDFReactor Service

If PDFReactor is deployed using the PDFReactor installer, the installation provides an option to automatically install the PDFReactor service with PDFReactor. No further configuration is required in this case.

On Unix and Linux platforms, no installer is available. Therefore, the PDFReactor service must be started manually on these systems. To do so, after unzipping the PDFReactor installation archive go to the "path-to-PDFReactor/bin" directory and use this command to start the service:

```
./pdfreactorwebservice start
```

To stop the service, use:

```
./pdfreactorwebservice stop
```

⁹ pronounced "dot net", a software framework by Microsoft (<http://msdn.microsoft.com/en-us/netframework/default.aspx>)

¹⁰ PHP: Hypertext Preprocessor, an open-source server-side scripting language (<http://www.php.net/>)

¹¹ Java **AR**chive, a file container used for Java classes.

To display whether the service is already running, use:

```
./pdfreactorwebservice status
```

1.3 Jetty

The PDFReactor service is run on the application server Jetty. It is a requirement for the .NET, PHP, Perl, Python and Ruby wrappers.

Jetty is configured for a development environment, and will only listen at localhost at the port used by the wrappers.

By default, Jetty will listen at localhost:9423.

See:

<http://www.eclipse.org/jetty/> for details about Jetty and ways to configure it.

1.4 PHP Requirements

To use PDFReactor with the PHP API a webserver (e.g. Apache) with a PHP-installation (Version >4.3 or >5.0) is required.

The PDFReactor service must be running within Jetty on the same machine.

1.5 .NET Requirements

The PDFReactor .NET API requires the Microsoft .NET framework 2.0 including the latest patches.

The PDFReactor service must be running within Jetty on the same machine.

Additional Requirements for ASP.NET

The .NET framework 2.0 must be registered at your IIS¹²-server.

See:

[http://msdn2.microsoft.com/en-us/library/k6h9cz8h\(VS.80\).aspx](http://msdn2.microsoft.com/en-us/library/k6h9cz8h(VS.80).aspx)

1.6 Perl/Python/Ruby Requirements

The Perl/Python/Ruby API can be used via CGI¹³ on your webserver, or by the corresponding modules for the Apache webserver (mod-python, mod-perl, mod-ruby).

The PDFReactor service must be running within Jetty on the same machine.

For specific installation requirements please have a look at the install.txt of the related wrapper.

¹² Internet Information Services (<http://www.iis.net/>)

¹³ Common Gateway Interface, a protocol for calling external software via web server (<http://www.w3.org/CGI/>)

2. INTEGRATION

You can integrate PDFReactor by directly using it as a Java library, by using its .NET, PHP, Perl, Python or Ruby API, or by running it as a command line application. Running PDFReactor as a command line application is not recommended for interactive use, as there is a significant time overhead caused by the startup time of the JVM¹⁴ and the initial load caused by the JIT¹⁵ compiler.

2.1 Memory

Depending on the input documents, PDFReactor may require additional memory. Large and especially complex documents, e.g. documents containing several hundred pages or documents using a complex nested HTML structure, may require larger amounts of memory.

The exact amount of memory required depends nearly entirely on the input document. Should you run into any issues converting a document, we recommend increasing the memory to e.g. 2GB or higher before attempting another conversion.

Please see the individual chapters [Jetty Configuration \(p. 16\)](#) and [Command Line Configuration \(p. 20\)](#) for how to increase the memory available to PDFReactor.

Parallel Conversions

When doing multiple parallel PDF conversions, it is important to adapt the available memory to the number of parallel conversions.

Generally, a common document requires no more than 64MB of memory. To safely convert up to 16 of these documents in parallel, PDFReactor requires at least 1GB of memory (16 * 64MB). Keep in mind that this is merely a rule of thumb and that the amount of required memory may vary depending on the documents and integration environments.

2.2 Using the Java library

With just a few lines you can create PDFs inside your applications and servlet.

The following sample program converts <http://www.realobjects.com.com/> to PDF and saves it as output.pdf.

¹⁴ *Java Virtual Machine*

¹⁵ *Just In Time compilation, a technique for improving the performance of virtual machines.*

```

import java.io.FileOutputStream;
import java.io.OutputStream;

import com.realobjects.pdfreactor.PDFReactor;

public class FirstStepsWithPDFReactor {
    public static void main(String[] args) {
        try {
            PDFReactor pdfReactor = new PDFReactor();

            // configuration settings
            pdfReactor.setAddLinks(true);
            pdfReactor.setAddBookmarks(true);

            // render the PDF document
            byte[] pdf = pr.renderDocumentFromURL("http://www.realobjects.com");

            OutputStream outputStream = new FileOutputStream("output.pdf");
            outputStream.write(pdf);
            outputStream.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

See:

the API documentation for details.

2.2.1 Using PDFReactor in a Servlet

When used in a Servlet to generate a PDF that is returned to the client (e.g. a browser) PDFReactor can write directly to the `ServletOutputStream`:

```

ServletOutputStream out = response.getOutputStream();
response.setContentType("application/pdf");
pdfReactor.renderDocument(inputSource, out);
out.close();

```

2.2.2 Logging

PDFReactor uses the Java Logging API to output information about it's progress. A simple console logger can be created like this:

```

pdfReactorLogger = Logger.getAnonymousLogger();
pdfReactorLogger.setLevel(Level.INFO);
pdfReactorLogger.addHandler(new DefaultHandler());
pdfReactor.setLogger(pdfReactorLogger);

```

See:

<http://docs.oracle.com/javase/6/docs/technotes/guides/logging/>

Additionally, you can append the log to the generated PDF by using the method `setAppendLog` like this:

```
pdfReactor.setAppendLog(true);
```

Note:

To enable logging you have to set an appropriate log level first using the method `setLogLevel`, e.g. like this:

```
pdfReactor.setLogLevel(PDFReactor.LOG_LEVEL_WARN);
```

2.3 Running PDFReactor Without Graphics Environment

If you are using PDFReactor on a system without a graphics environment like X11, you need to enable the headless mode of Java. This can be done by setting the appropriate Java system property. You can either set the property as a Java VM argument or you can set it inside your Java code. It is recommended to set it as early as possible, as changing it affects the entire Java VM instance. In any case it is important to set the property before PDFReactor is instantiated.

As a Java VM Argument

```
java -Djava.awt.headless=true
```

In Java Code

```
public class MyPDFReactorIntegration {
    // set the headless system property
    static {
        System.setProperty("java.awt.headless", "true");
    }

    public void createPDF() {
        PDFReactor pdfReactor = new PDFReactor()
        // ...
    }
}
```

Note:

Enabling the headless mode is not necessary when using the PDFReactor Command Line Interface or the PDFReactor Web Service.

Important:

If the headless mode is not enabled on a system without a graphics environment, you might experience an error similar to this:

```
java.lang.InternalError: Can't connect to X11 window server using '' as the value of the DISPLAY variable
```

2.4 Using a Wrapper

PDFReactor can also be easily integrated in your web apps using one of the wrappers APIs, i.e. PHP, .NET, Python, Perl or Ruby. This has to be used in conjunction with the PDFReactor Web Service which is run by a Jetty web application server (see chapter [Jetty \(p. 9\)](#)).

See also [The PDFReactor Service \(p. 8\)](#) for information on how to start the service.

2.4.1 Using PHP

To use the PDFReactor PHP API simply copy the "PDFReactor.class.php" to a directory of your webserver where PHP is enabled.

Then include the "PDFReactor.class.php" with:

```
include("/path/to/PDFReactor.class.php");
```

With just a few lines you can create and directly show PDFs inside your PHP webapplication:

```
<?php
include("../PDFReactor.class.php");
$pdfReactor = new PDFReactor();
$result = $pdfReactor->renderDocumentFromURL("http://www.pdfreactor.com");

// Check if successful
if (is_null($result)) {
    // Not successful, print error and log
    echo "<h1>Error During Rendering</h1>";
    echo "<h2>".$pdfReactor->getError()."</h2>";
    echo "<pre>".$pdfReactor->getLog()."</pre>";
} else {
    // Set the correct header for PDF output and echo PDF content
    header("Content-Type: application/pdf");
    echo $result;
}
?>
```

See:

PDFReactor methods in the PHP API docs for all available options.

PHP API specific issues

PHP Script timeout: Generally the timeout of PHP scripts is set to 30s within the "php.ini". When rendering large documents this limit may be exceeded.

2.4.2 Using .NET

You can easily access the PDFReactor service from any .NET language. The library assembly "PDFReactor.dll" offers you a large subset of the Java-API and takes care of all communication with the service.

A simple usage in C# would be:

```
PDFReactor pdfReactor = new PDFReactor();
byte[] result = pdfReactor.RenderDocumentFromURL("http://www.pdfreactor.com/");
```

A class reference and an XML documentation file is included.

Using ASP.NET

To use the .NET API from ASP.NET¹⁶ copy "PDFReactor.dll" from "wrappers\dotnet\bin" in your PDFReactor installation directory to "bin" in the root of your IIS-Application or to the global assembly cache.

An ASP.NET example would be:

¹⁶ Active Server Pages .NET, a framework by Microsoft to build dynamic web sites and web applications

```
<%@ Page Language="C#" Debug="false" %>
<%
PDFReactor pdfReactor = new PDFReactor();
byte[] result = pdfReactor.RenderDocumentFromURL("http://www.pdfreactor.com/");

// Check if successful
if (result == null)
{
    // Not successful, print error and log
    Response.Write("<h1>Error During Rendering</h1>>");
    Response.Write("<h2>" + pdfReactor.GetError() + "</h2>");
    Response.Write("<pre>" + pdfReactor.GetLog() + "</pre>");
}
else
{
    // Set the correct header for PDF output and echo PDF content
    Response.ContentType = "application/pdf";
    Response.BinaryWrite(result);
}
%>
```

2.4.3 Using Python

To use the PDFReactor Python API simply copy the "PDFReactor.py" to a directory of your webserver where Python is enabled (by e.g. CGI or mod-python).

Then include the "PDFReactor.py" with:

```
import sys
sys.path.append("path/to/PDFReactor.py/")
from PDFReactor import *
```

With just a few lines you can create and directly show PDFs inside your Python webapplication:

```
pdfReactor = PDFReactor()
result = pdfReactor.renderDocumentFromURL("http://www.pdfreactor.com")

# Check if successful
if result == None:
    # Not successful, print error and log
    print "Content-Type: text/html\n"
    print "<h1>Error During Rendering</h1>"
    print "<h2>" + pdfReactor.getError() + "</h2>"
    print "<pre>" + pdfReactor.getLog() + "</pre>"
else:
    # Used to prevent newlines are converted to Windows newlines (\n --> \r\n)
    # when using Python on Windows systems
    if sys.platform == "win32":
        import os, msvcrt
        msvcrt.setmode(sys.stdout.fileno(), os.O_BINARY)
    # Set the correct header for PDF output and echo PDF content
    print "Content-Type: application/pdf\n"
    sys.stdout.write(result)
```

Windows specific issues:

To directly output the PDF to the browser please use the following code:

```
if sys.platform == "win32":
    import os, msvcrt
    msvcrt.setmode(sys.stdout.fileno(), os.O_BINARY)
    print "Content-Type: application/pdf\n"
    sys.stdout.write(result)
```

See:

PDFreactor methods in the Python API docs for all available options.

2.4.4 Using Perl

To use the PDFreactor Perl API simply copy the "PDFreactor.pm" to a directory of your webserver where Perl is enabled (by e.g. CGI or mod-perl).

Then include the "PDFreactor.pm" with:

```
require "PDFreactor.pm";
```

With just a few lines you can create and directly show PDFs inside your Perl webapplication:

```
my $pdfReactor = PDFreactor -> new();
$result = $pdfReactor -> renderDocumentFromURL("http://www.pdfreactor.com");

# Check if successful
if (!defined $result) {
    # Not successful, print error and log
    print "Content-type: text/html\n\n";
    print "<h1>Error During Rendering</h1>";
    print "<h2>".$pdfReactor -> getError() . "</h2>";
    print "<pre>".$pdfReactor -> getLog() . "</pre>";
} else {
    # Set the correct header for PDF output and echo PDF content
    print "Content-type: application/pdf\n\n";
    binmode(STDOUT);
    print $result;
}
```

Windows specific issues:

To directly output the PDF to the browser please use the following code before printing the result:

```
binmode(STDOUT);
```

See:

PDFreactor methods in the Perl API docs for all available options.

2.4.5 Using Ruby

To use the PDFreactor Ruby API simply copy the "PDFreactor.rb" to a directory of your webserver where Ruby is enabled (by e.g. CGI or mod-ruby).

Then include the "PDFreactor.rb" with:

```
require 'PDFreactor.rb'
```

With just a few lines you can create and directly show PDFs inside your Ruby web application:

```
pdfReactor = PDFReactor.new()
result = pdfReactor.renderDocumentFromURL("http://www.pdfreactor.com")

# Check if successful
if result == nil
  # Not successful, print error and log
  print "Content-type: text/html\n\n"
  puts "<h1>Error During Rendering</h1>"
  puts "<h2>"+pdfReactor.getError()+"</h2>"
  puts "<pre>"+pdfReactor.getLog()+"</pre>"
else
  # Set the correct header for PDF output and echo PDF content
  print "Content-type: application/pdf\n\n"
  $stdout.binmode
  print result
end
```

Windows specific issues:

To directly output the PDF to the browser please use the following code before printing the result:

```
$stdout.binmode
```

See:

PDFReactor methods in the Ruby API docs for all available options.

2.4.6 Jetty Configuration

The Jetty application server can be configured in several ways. Most commonly, as described in the chapter [Memory \(p. 10\)](#), you may want to increase the amount of memory available.

Increasing Memory

To increase the amount of memory available to Jetty and thus to PDFReactor, you need to adapt the `-Xmx1024m` parameter in the file `"PDFReactor/bin/pdfreactorweb.service.vmoptions"`.

To increase the memory to e.g. 2GB, change the parameter to `-Xmx2048m` and restart the web service.

Increasing Maximum Threads

The number of maximum threads limits the number of parallel conversions. For machines with multiple CPU cores, this value can be increased to allow more parallel conversions. Keep in mind that more parallel conversions will result in increased memory usage. Please also see the chapter [Memory \(p. 10\)](#) for more information.

The number of maximum threads can be increased in the following line of the `"jetty.xml"`:

```
...
<Set name="maxThreads">16</Set>
...
```

Changing the Port

Sometimes it may be necessary to change the port of the PDFReactor web service. The port can be changed in the `"jetty.xml"` file located in the `"PDFReactor/jetty/etc"` directory.

You can change the port in this line of the "jetty.xml":

```
...
<Set name="port">9423</Set>
...
```

When the port of the PDFReactor web service was changed, it is also necessary to specify the new port in the constructor of the PDFReactor class. The following examples show how to use the constructor of the various APIs if the port was changed to 9424:

Example: Using PHP

```
$pdfReactor = new PDFReactor("localhost", 9424);
```

Example: Using .NET

```
PDFReactor pdfReactor = new PDFReactor("localhost", 9424);
```

Example: Using Python

```
pdfReactor = PDFReactor("localhost", 9424);
```

Example: Using Perl

```
my $pdfReactor = PDFReactor -> new("localhost", 9424);
```

Example: Using Ruby

```
pdfReactor = PDFReactor.new("localhost", 9424);
```

Changing the Host

Usually it is recommended to run the PDFReactor web service on the same machine as the PDFReactor integration. This is not strictly necessary and the host for the service can be changed.

You have to remove the following line from the "jetty.xml":

```
...
<Set name="host">localhost</Set>
...
```

This will enable the PDFReactor Web Service to be accessible from other machines.

Note:

Removing the host from the Jetty configuration file causes it to be accessible from *all* other machines, so this is only recommended for internal or closed environments.

The new host for the PDFReactor Web Service then needs to be specified in your integration code. The following example shows how to use the constructor of the various APIs if the host was changed to "my-server.com":

Example: Using PHP

```
$pdfReactor = new PDFReactor("my-server.com");
```

Example: Using .NET

```
PDFReactor pdfReactor = new PDFReactor("my-server.com");
```

Example: Using Python

```
pdfReactor = PDFReactor("my-server.com");
```

Example: Using Perl

```
my $pdfReactor = PDFReactor -> new("my-server.com");
```

Example: Using Ruby

```
pdfReactor = PDFReactor.new("my-server.com");
```

2.4.7 Logging

The logging mechanism for the APIs that use the web service is different from the logging mechanism of the Java API. Here, the PDFReactor instance has two additional methods `getLog` and `getError` which can be called after the conversion process to retrieve the log or any errors which may have occurred during the conversion, respectively.

Another way of retrieving the log is using the method `setAppendLog`. This will append the log to the generated PDF you have both the generated PDF and the log in one document.

Additionally, the entire log output of the Jetty application server is written into the "output.log" file located in the "PDFReactor/bin" directory.

Note:

To enable logging you have to set an appropriate log level first using the method `setLogLevel`.

Examples

The following examples show how to enable logging by setting an appropriate log level and then appending the log to the generated PDF.

Example: Using PHP

```
$pdfReactor->setLogLevel(LOG_LEVEL_DEBUG);  
$pdfReactor->setAppendLog(true);
```

Example: Using .NET

```
pdfReactor.SetLogLevel(PDFReactor.LOG_LEVEL_DEBUG);  
pdfReactor.SetAppendLog(true);
```

Example: Using Python

```
pdfReactor.setLogLevel(PDFReactor.LOG_LEVEL_DEBUG)  
pdfReactor.setAppendLog(True)
```

Example: Using Perl

```
$pdfReactor -> setLogLevel($pdfReactor -> LOG_LEVEL_DEBUG);  
$pdfReactor -> setAppendLog('true');
```

Example: Using Ruby

```
pdfReactor.setLogLevel(PDFReactor::LOG_LEVEL_DEBUG)
pdfReactor.setAppendLog(true)
```

2.5 Using the Command Line Interface

2.5.1 Quick Introduction to the Command Line Interface

In order to run PDFReactor from the command line, please make sure that a JRE¹⁷ is installed (you can find the minimum requirements in the readme file). To verify that the JRE is installed properly and see the current version just type the following command in your shell:

```
java -version
```

This should result in an output similar to the following:

```
C:\>java -version
java version "1.6.0_17"
Java(TM) SE Runtime Environment, Standard Edition (build 1.6.0_17-b04)
Java HotSpot(TM) Client VM (build 14.3-b01, mixed mode, sharing)
```

In order to run PDFReactor, you must use java with the jar option:

```
java -jar pdfreactor.jar
```

The `-h` option of PDFReactor will display an overview of all available command line options:

```
java -jar pdfreactor.jar -h
```

All of these sample instructions assume that you are currently in the same directory as pdfreactor.jar file.

Generating PDF documents

In the simplest case, PDFReactor only needs two parameters to generate a PDF file: the input and the output file:

```
java -jar pdfreactor.jar <input file> <output file>
```

For example:

```
java -jar pdfreactor.jar c:\test\test.html c:\test\test.pdf
```

Instead of an input file you can also specify a URL:

```
java -jar pdfreactor.jar http://www.myserver.com/mytestfile.xml myfile.pdf
```

¹⁷ Java Runtime Environment (<http://www.oracle.com/technetwork/java/javase/downloads/>)

Note:

The directory separator is different for each platform. On Windows it is "\" whereas under Linux and Mac OS it is "/".

```
java -jar pdfreactor.jar /user/home/testuser/test.xml test.pdf
```

Note:

Paths to files which contain whitespaces must be surrounded with quotation marks:

```
java -jar pdfreactor.jar "c:\documents and settings\user\test.html" test.pdf
```

2.5.2 Advanced Command Line Options

This section covers the most important command line calls.

Displaying Information about the Rendering Process

Using `-v` or `--verbose` you can specify several levels of output: they reach from `debug` for all possible information available to `fatal` which only displays fatal errors. The following example displays all available debug information:

```
java -jar pdfreactor.jar -v debug c:\test\test.html test.pdf
```

This example only displays fatal errors:

```
java -jar pdfreactor.jar -v fatal c:\test\test.html test.pdf
```

When specifying `-v` without indicating a level, it is set to `info` by default.

Note:

By default, all information is written to the standard output. When using `stdout` as output for the PDF document, all logging information is automatically written to `stderr`.

Applying External Stylesheets

Styles can be applied in various ways. Usually they're linked with the rendered document using specific elements such as `<link>` in HTML documents. You can also specify styles via command line parameter directly as string or by indicating a CSS as URL:

```
java -jar pdfreactor.jar  
  --style "*" {font-weight:bold}" c:\test\test.html test.pdf
```

```
java -jar pdfreactor.jar  
  --styleurl http://myserver/style.css c:\test\test.html test.pdf
```

2.5.3 Command Line Configuration

To increase the amount of memory available to PDFReactor, you need start the Java VM with the `-Xmx` parameter which specifies the amount of memory.

To increase the memory to e.g. 2GB, start the VM with the parameter `-Xmx2048m` like this:

```
java -Xmx2048m -jar pdfreactor.jar
```

2.6 License Key

2.6.1 Evaluation Mode

Without a license key PDFReactor runs in evaluation mode. In evaluation mode it is possible to integrate and test PDFReactor just like the full version but the resulting PDF document will include watermarks and additional evaluation pages.

2.6.2 Receiving a License Key

To obtain a license key, please visit the [PDFReactor website \(http://www.pdfreactor.com\)](http://www.pdfreactor.com). It provides information about all available licenses and how to receive license keys.

2.6.3 Setting the License Key

RealObjects provides you a license key file in XML format.

The license key can be set as a string using the `setLicenseKey` method of the PDFReactor class.

Example:

```
String licensekey = "<license>... your license ...</license>";  
pdfReactor.setLicenseKey(licensekey);
```

Note:

You can ensure that no eval or license notices are added to PDF documents using the following method:

```
pdfReactor.setThrowLicenseExceptions(true)
```

This forces PDFReactor to throw an exception instead of adding notices to PDF documents.

3. INPUT FORMATS

PDFreactor can process the following input formats. By default, it automatically tries to identify the right format.

3.1 HTML + CSS

HTML is directly rendered by PDFreactor using a default CSS style sheet for HTML in addition to the document's style.

PDFreactor automatically renders HTML form controls such as buttons, input fields and text areas. PDFreactor can even be used to automatically generate interactive PDF forms (sometimes referred to as AcroForms) from HTML forms. For details please see the chapter [Interactive PDF Forms. \(p. 38\)](#)

HTML is parsed by the built-in HTML5 parser which parses the document according to HTML5 rules. This means that elements missing closing tags (such as `<p>` without `</p>`) are handled as demanded by the HTML5 specifications, and SVG and MathML can be used without having to specify their respective namespace.

When using XHTML¹⁸, the code is also automatically cleaned, when a parse error occurs, e.g. if the document is not well-formed.

The following Cleanup processes are available and only used for XHTML content:

CyberNeko

CyberNeko is the default cleanup used by PDFreactor. This XHTML parser fixes the following XHTML incompatibilities:

- adds missing parent elements
- automatically closes elements
- handles mismatched end tags

It is generally recommended to use the CyberNeko clean-up process.

jTidy

If the cleanup performed by CyberNeko is not sufficient, use jTidy. This cleanup handles content a bit more aggressively than CyberNeko, and may drop elements if it can not clean them. JTidy is a Java port of HTML Tidy, a HTML syntax checker and pretty printer. jTidy provides the following features (among others):

- Missing or mismatched end tags are detected and corrected
- End tags in the wrong order are corrected
- Recovers from mixed up tags
- Adding the missing `"/` in end tags for anchors
- Correcting lists by putting in tags missed out
- Missing quotes around attribute values are added

¹⁸ Extensible HyperText Markup Language. (<http://www.w3.org/Markup/>)

- Unknown/Proprietary attributes are reported
- Proprietary elements are recognized and reported as such
- Tags lacking a terminating '>' are spotted

TagSoup

The TagSoup cleanup is able to fix most namespace issues that may occur when importing content from non-standard sources such as Office applications. It has the following cleanup features:

- It always returns a cleaned document, i.e. it does not throw an exception
- Unbound namespace prefixes are fixed.

This clean-up process is recommended for documents exported from Office applications.

Note:

To use a cleanup on a document fragment (or any other document with no "<html>" root element) you must manually set the doctype to XHTML.

3.2 XML + CSS

XML documents can be styled with CSS and be processed through the built-in XSLT¹⁹ processor for more sophisticated tasks, e.g. to calculate the total from values of a XML-based invoice. Because XML does not have a default CSS style sheet, you usually will have to write your own one for your specific XML language.

3.3 Compound Formats

In addition to rendering HTML and XML styled with CSS, PDFreactor is also able to render documents with compound formats such as images, SVGs or barcodes, so-called replaced elements.

The replaced elements can be mapped to arbitrary elements using styles.

You can use namespaces to include other document formats to integrate XML elements from a different namespace directly within your document.

3.3.1 Images

PDFreactor supports the `img` element per default in HTML. For other XML languages, you can use proprietary CSS extensions to define an image element. For example, in an XML vocabulary where an image element is `<image source='test.jpg'>`, the corresponding CSS definition would be:

```
image {
  -ro-replacdelement: image;
  -ro-source: ro-attr(source);
}
```

¹⁹ Extensible Stylesheet Language Transformations (<http://www.w3.org/TR/xslt>)

To define an element as image element, you must specify the replaced element formatter for images for this element, as displayed in the example above. Using the `-ro-source` (p. 156) property, you can select an attribute of this element. The value of this attribute must always be of the type URI²⁰ and is used to load the image.

Note:

PDFReactor supports the image formats PNG, JPG, GIF and TIFF.

3.3.2 SVG

PDFReactor automatically embeds and renders SVG²¹ documents referenced via the `img` element. Example:

```

```

Alternatively, you can embed SVG directly into your documents:

```
a circle:<br/>
<svg width="100" height="100">
  <circle cx="50" cy="50" r="45" fill="yellow" stroke="black" />
</svg>
<br/>sometext.....
```

Note:

When using non-HTML5 documents, an SVG namespace has to be added and used:

```
<svg:svg xmlns:svg="http://www.w3.org/2000/svg" width="100" height="100">
  <svg:circle cx="50" cy="50" r="45" fill="yellow" stroke="black" />
</svg:svg>
```

Rasterization

SVGs are embedded into the PDF as vector graphics, keeping them resolution independent. However SVGs containing masks, filters or non-default composites have to be rasterized²². This behavior can be configured using CSS:

The style `-ro-rasterization: avoid` (p. 153) disables the aforementioned SVG features to avoid having to rasterize the image.

The property `-ro-rasterization-supersampling` (p. 153) configures the resolution of the rasterization. The default value is 2, meaning twice the default CSS resolution of 96dpi. Accepted values are all positive integers. Higher resolution factors increase the quality of the image, but also increase the conversion time and the size of the output documents.

CMYK Colors in SVG

PDFReactor supports CMYK colors in SVGs. Those are passed to the PDF as-is, as long as the SVG is not rasterized.

Example: Setting the stroke color to black

```
stroke="cmyk(0.0, 0.0, 0.0, 1.0)"
```

²⁰ Uniform Resource Identifier (<http://www.w3.org/Addressing/>)

²¹ Scalable Vector Graphics (<http://www.w3.org/Graphics/SVG/>)

²² Rasterization is the task of taking an image described in a vector graphics format and converting it into a raster (pixel) image.

3.3.3 Barcode

PDFReactor supports displaying barcodes in documents using the Barcode XML format from Barcode4J:

```
<p><b>EAN-13:</b></p>
<barcode:barcode xmlns:barcode="http://barcode4j.krysalis.org/ns"
  message="123456789012">
  <barcode:ean-13/>
</barcode:barcode>
<br>sometext.....
```

For details about Barcode XML see:

<http://barcode4j.sourceforge.net>

3.3.4 MathML

PDFReactor supports displaying MathML²³ in documents using the MathML XML format.

```
<math:math mode="display" xmlns:math="http://www.w3.org/1998/Math/MathML">
  <math:mrow>
    <math:munderover>
      <math:mo>&#x222B;</math:mo>
      <math:mn>1</math:mn>
      <math:mi>x</math:mi>
    </math:munderover>
    <math:mfrac>
      <math:mi>dt</math:mi>
      <math:mi>t</math:mi>
    </math:mfrac>
  </math:mrow>
</math:math>
```

For details about the available MathML elements see:

<http://jeuclid.sourceforge.net/testsuite/index.html>

3.3.5 QR Code

PDFReactor supports displaying QR codes²⁴ in documents using the following style:

```
.qrcode {
-ro-replacdelement: qrcode;
}
```

If the replaced element is applied to an HTML link, the reference URL (resolved against the base URI) is used as the content of the QR code, e.g.:

```
<a href="http://www.pdfreactor.com" class="qrcode"></a>
```

²³ **Mathematical Markup Language** (<http://www.w3.org/Math/>)

²⁴ **Quick Response Code** (<http://www.denso-wave.com/qrcode/index-e.html>)

In any other case the text content of the element is used, e.g.:

```
<span class="qrcode">
BEGIN:VCARD
VERSION:2.1
N:Doe
FN:John
TEL:+1-555-123-456
TEL;FAX:+1-555-123-457
EMAIL:johndoe@johndoe.com
URL:http://www.johndoe.com
END:VCARD
</span>
```

QR Codes can be tweaked using the following CSS properties:

- `-ro-qrcode-errorcorrectionlevel` (p. 151) – Sets the error correction level of the QR code. Possible values are L (default), M, Q and H.
- `-ro-qrcode-quietzonesize` (p. 152) – Sets the size of the quiet (empty) zone around the QR code in modules (QR code "square" widths). The default value is 1. Possible values are 0 (no quiet zone) and positive integers.
- `-ro-qrcode-forcedcolors` (p. 151) – By default, QR codes are black on white. When setting this property to none, the CSS properties `color` (p. 115) and `background-color` (p. 104) are used instead.
- `-ro-qrcode-quality` (p. 152) – By default, The QR code is built from multiple squares. This method is fast and looks correct in print. However, in PDF viewers on screen the edges of neighboring squares may be visible. When setting this property to high the squares are combined into one object, ensuring a seamless look, at the cost of performance.

3.3.6 Object and Embed

PDFReactor supports the `object` and `embed` elements of HTML. You can use either element or a combination of both to embed any type of data such as for example a flash animation. The most simple code to do so is:

```
<embed src="myflash.swf" width="256" height="256"
type="application/x-shockwave-flash"/>
```

Note:

Besides flash you can also embed various other formats, e.g. videos. The data is automatically embedded in the PDF, but whether or not it is displayed depends on the formats supported by your PDF viewer.

3.3.7 iframes

An `iframe` allows another document, for example content from other pages, to be embedded inside an existing one.

The source document

There are two ways to define the inner document of an `iframe`. The first option is to use the `src` attribute and specifying the URL from which the document should be loaded. The URL might be absolute or relative and should refer to an HTML document.

The second option is useful if the inner document is very short and simple. When using the `srcdoc` attribute, its value is set to be the inner document's source code.

```
<iframe src="http://www.pdfreactor.com" width="600" height="400">
</iframe>

<iframe srcdoc="<p>Hello World</p>">
  <b>This is fallback text in case the user-agent does not support
    iframes.</b>
</iframe>
```

Note:

If both attributes have been set, `srcdoc` has priority over `src`.

Seamless

If the `seamless` attribute has been set, the `iframe`'s document behaves as it would be in the document that contains the `iframe`. That means that the width and height of the `iframe` are ignored and the inner document is shown completely if possible.

Furthermore, the borders of the `iframe` are removed and most importantly all styles from the outer document are inherited by the inner document.

When generating the PDF, the headings and other bookmark styles inside the `iframe` are passed through, so they can be found in the bookmark list.

The `seamless` attribute is a boolean attribute, which means that if it exists it is set to true and if it does not exist, it is set to false. The only valid values of `seamless` are an empty string "" or "seamless". The attribute can also be used without any value:

```
<iframe src="http://www.pdfreactor.com" width="600" height="400"
  seamless>
</iframe>
```

Note:

Generally, `true` and `false` are INVALID values for boolean attributes.

Customization

Using CSS styles, it is possible to customize the look and functionality of iframes.

The border, padding and margin can be set or removed with the appropriate styles.

```
iframe {
  border: none;
  padding: 0px;
  margin: 0px;
}
```

By default, if `seamless` is false neither style sheets nor inline styles are passed down to the `iframe`'s document. However, by using the property `-ro-passdown-styles` (p. 148), this behavior can be customized.

When generating a PDF with the bookmarks feature enabled, the headings in the document are added as bookmarks to quickly navigate the document.

Using the property `-ro-bookmarks-enabled` (p. 107) it is possible to enable or disable this feature for iframes, thus allowing the headings of the inner document to be added to the bookmarks list or not. The property can be either set to `true` or `false`. If the iframe is seamless, it is set to `true` by default.

```
<iframe src="http://www.pdfreactor.com" width="600" height="400"
  seamless="seamless" style="-ro-passdown-styles:stylesheets-only;
  -ro-bookmarks-enabled:false;">
</iframe>
```

3.3.8 Canvas Element

PDFReactor has built-in support for the `canvas` element of HTML5. The canvas element is a dynamic image for rendering graphics primitives on the fly. In contrast to other replaced elements the content of the canvas element must be generated dynamically via [JavaScript \(p. 29\)](#), instead of referencing an external resource that contains the content to be displayed (as is the case for example for images).

Below is a simple code fragment which renders shadowed text into a `canvas` element:

```
<head>
  <script type="text/javascript">
function draw() {
  var ctx = document.getElementById("canvas").getContext('2d');
  ctx.font = "50px 'sans-serif'";
  ctx.shadowBlur = 5;
  ctx.shadowColor = "#aaa";
  ctx.shadowOffsetX = 2;
  ctx.shadowOffsetY = 2;
  ctx.fillStyle = "black";
  ctx.fillText("PDFReactor",0,50);
}
  </script>
</head>
...
<body onload="draw();">
  <canvas id="canvas" width="400" height="300">
    Canvas element is not supported.
  </canvas>
</body>
```

Note:

The creation of shadows is a time-consuming task and can, depending on the content to generate, considerably increase the creation time of the PDF. Thus shadows should be used with caution if the creation time of the PDF is important.

Resolution Independence

PDFReactor by default does not use a resolution-dependent bitmap as the core of the canvas. Instead it converts the graphics commands from JavaScript to resolution-independent PDF objects. This avoids resolution-related issues like blurriness or pixelation.

Shadows can not be convert to PDF objects. So those are added as images. This does not affect other objects in the canvas.

Accessing `ImageData` of a canvas or setting a non-default composite causes that canvas to be rasterized entirely.

This behavior can be configured using CSS:

The style `-ro-rasterization: avoid` (p. 153) disables functionality that causes the rasterization of the canvas.

The style `-ro-rasterization`: always forces the canvas to be rasterized in any case.

The property `-ro-rasterization-supersampling` (p. 153) configures the resolution at which the canvas or shadows are rasterized. The default value is 2, meaning twice the default CSS resolution of 96dpi. Accepted values are 1 to 4. Higher resolution factors increase the quality of the image, but also increase the conversion time and the size of the output documents. This does not affect canvas objects that are not rasterized.

3.4 JavaScript

PDFReactor can be configured to process JavaScript that is embedded into or linked from input HTML documents. This functionality can be enabled as follows:

```
pdfReactor.setJavaScriptMode(PDFReactor.JAVASCRIPT_MODE_ENABLED);
```

It is also possible to manually add scripts:

```
pdfReactor.addUserScript("console.log(\"test\");", null, false);
```

See:

The PDFReactor API documentation for details on these API methods.

JavaScript processing during PDF conversion works like it does in a browser, with some exceptions:

- Alerts and other dialogs are logged and do not stop script processing.
- The delay parameter of `setTimeout` is used only for scheduling, and does not cause actual delays. `setInterval` is treated like `setTimeout`.
- There are no security measures based on the origin of URLs ("cross-site scripting").

JavaScript processing is subject to a few other limitations that will be eliminated in future versions of PDFReactor:

- Reading from and manipulating form elements is not fully supported.
- Access to SVG specific members of DOM-elements is limited. Other compound formats, like MathML, are not specifically supported.
- Coordinates (e.g. retrieved via `getClientRects`) are relative to their pages, which might lead to unexpected results in some situations.
- Redirects (e.g. changing `window.location`) are not possible.
- After setting a CSS short-hand, the long hand values cannot be retrieved.

The following alternative JavaScript modes can be selected:

Mode	Description	Notes
no layout	There is no access to layout information and no re-layouts will be triggered.	Faster, but will not provide correct results for scripts that require coordinates or sizes from the layout.
time-lapse	Value of <code>Date.getTime()</code> increases faster.	jQuery animations finish immediately, e.g. for Highcharts
real-time	No converter-specific optimizations for <code>setInterval</code> and <code>setTimeout</code> .	This mode is slower than time-lapse. It should only be used when it is the only mode providing correct results, e.g. for amCharts

The following JavaScript libraries and frameworks were tested:

Library	Test result
jQuery	functional, extensively tested
MooTools	functional
Prototype	functional, except for event functionality
Modernizr	functional
Flotr2	functional
Highcharts	functional, requires <code>time-lapse</code> mode
amCharts	functional, requires <code>real-time</code> mode
Chart.js	functional

3.4.1 Proprietary Functions and Members

PDFReactor allows JavaScript access to some layout information while rendering a document. To get the information use the following methods and objects:

ro.layout.getPageDescription(index)

Returns a [PageDescription](#) (p. 31) for the page with the given *index*. The first page has the index 0.

The returned object is a snapshot of that particular moment. Changing the document after getting a description object has no effect on it.

ro.layout.getBoxDescriptions(element)

Returns an array of [BoxDescription](#) (p. 31) objects for the given *element*. Note that one element can have several boxes (e.g. a paragraph that does not fit on a single page).

The returned objects are snapshots of that particular moment. Changing the document after getting description objects has no effect on them.

ro.layout.numberOfPages

Returns the current total number of pages of the document.

Return Types

ClientRect

A ClientRect contains the position and dimensions of a rectangle:

ClientRect

Key	Description
top	The y-coordinate in px.

Key	Description
right	The x-coordinate plus the width in px.
bottom	The y-coordinate plus the height in px.
left	The x-coordinate in px.
width	The width in px.
height	The height in px.

PageDescription

Describes the dimensions of page and its rectangles. The Rectangles are described by using [ClientRect \(p. 30\)](#).

PageDescription

Key	Description
marginRect	Returns a <code>ClientRect</code> describing the margin rectangle. The position of this rectangle is always 0,0
borderRect	Returns a <code>ClientRect</code> describing the border rectangle.
paddingRect	Returns a <code>ClientRect</code> describing the padding rectangle.
contentRect	Returns a <code>ClientRect</code> describing the content rectangle. An element with absolute position and <code>top:0, left:0</code> would be in the upper left corner of this rectangle.
pageIndex	The index of this page. The first page has the index 0.

BoxDescription

Describes the position and dimensions of the rectangles of a box. The Rectangles are described by using [ClientRect \(p. 30\)](#).

BoxDescription

Key	Description
marginRect	Returns a <code>ClientRect</code> describing the margin rectangle. The point of origin is the upper left corner of the page content rectangle.
borderRect	Returns a <code>ClientRect</code> describing the border rectangle. This is the rectangle that is required in most cases. The point of origin is the upper left corner of the page content rectangle.
paddingRect	Returns a <code>ClientRect</code> describing the padding rectangle. The point of origin is the upper left corner of the page content rectangle.
contentRect	Returns a <code>ClientRect</code> describing the content rectangle. The point of origin is the upper left corner of the page content rectangle.
marginRectInPage	Returns a <code>ClientRect</code> describing the margin rectangle. The point of origin is the upper left corner of the page rectangle.
borderRectInPage	Returns a <code>ClientRect</code> describing the border rectangle. The point of origin is the upper left corner of the page rectangle.

Key	Description
paddingRectInPage	Returns a <code>ClientRect</code> describing the padding rectangle. The point of origin is the upper left corner of the page rectangle.
contentRectInPage	Returns a <code>ClientRect</code> describing the content rectangle. The point of origin is the upper left corner of the page rectangle.
pageIndex	The index of the page of this box. The first page has the index 0.
pageDescription	The PageDescription (p. 31) of the page of this box. It contains the data of the page from the moment when this <code>BoxDescription</code> was created.

3.4.2 awesomizr.js

The JavaScript library *awesomizr.js* is a collection of helpful functions for the use with PDFReactor. You have to import the JavaScript and in some cases the corresponding CSS.

You can add the library by using the PDFReactor API method `addUserScript()`. To add the respective CSS, use the method `addUserStyleSheet()`:

```
pdfReactor.addUserStyleSheet("", "", "", "awesomizr.css");
pdfReactor.addUserScript(null, "awesomizr.js", false);
pdfReactor.addUserScript("Awesomizr.createTableOfContents();", null, false);
```

Note:

Of course, the library and the stylesheet can alternatively be imported by the document itself. However, please note that some functions only work with PDFReactor.

The capabilities of *awesomizr.js* include:

- [Rotating table headers to reduce the table header width \(p. 62\)](#)
- [Adaptive Page Break Insertion \(p. 68\)](#)
- [Creating a Table of Contents \(p. 72\)](#)

4. OUTPUT FORMATS

PDFReactor supports multiple output formats, including PDF and various image formats:

4.1 PDF

PDF is the default output Format of PDFReactor. Some features of PDFReactor are specific to this output format:

4.1.1 Bookmarks

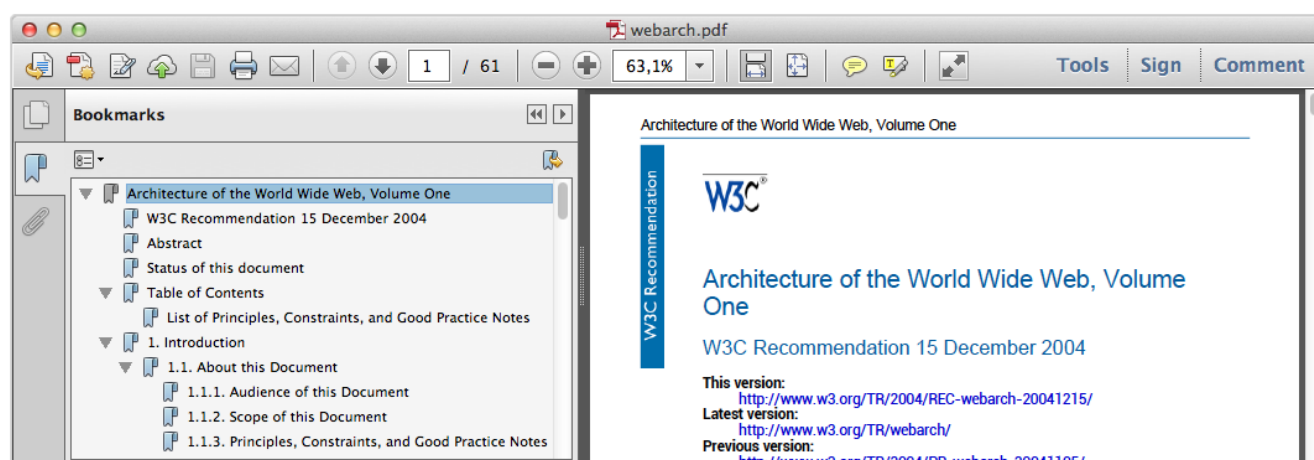


Fig. 1: Bookmarks in the Adobe Reader

PDFReactor adds bookmarks to your document by using the following API method:

```
pdfReactor.setAddBookmarks(true);
```

When the default HTML mode is enabled, the following bookmark levels are applied by default:

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

Using the `-ro-pdf-bookmark-level` (p. 149) style you can create bookmarks which link to arbitrary XML elements in your PDF files.

```
element { -ro-pdf-bookmark-level: 1 }
```

Using this property, one can structure the specified elements within the bookmark view of the PDF viewer. The elements are ordered in ascending order. The element with the lowest bookmark level is on top of the bookmark hierarchy (similar to HTML headlines). Several bookmark levels can be set using the `-ro-pdf-bookmark-level` style.

4.1.2 Links

PDFReactor can add links to your documents when using the following API method:

```
pdfReactor.setAddLinks(true);
```

When the default HTML mode is enabled, the following link styles are applied by default:

```
a[href] { -ro-link: ro-attr(href); }
a[name] { -ro-anchor: ro-attr(name ro-ident); }
```

Using the styles **-ro-link** (p. 137) and **-ro-anchor** (p. 102) arbitrary elements can be defined to be links or anchors.

```
linkElement[linkAttribute] { -ro-link: ro-attr(linkAttribute); }
anchorElement[anchorAttribute] { -ro-anchor: ro-attr(anchorAttribute ro-ident); }
```

Please also see **Attachments** (p. 46).

The clickable areas of links

The style **-ro-link-area** (p. 138) can be used to specify how the 'clickable' areas of a link are determined:

The effects of the values of -ro-link-area on block and split inline elements

Value	Clickable areas for block elements	Clickable areas for split inline elements ²⁵
content	one for each piece of content ²⁶	one for each part
block	one for the whole block	the bounding rectangle of all parts
content-block	the bounding rectangle of the content	the bounding rectangle of all parts

This style is not inherited. It has to be set on the same element as -ro-link.

4.1.3 Comments

It is possible to add PDF comments to the document using the following API method:

```
pdfReactor.setAddComments(true);
```

²⁵ split inline: inline element (e.g. span) spread over multiple lines and therefore split into at least two parts

²⁶ In this case a "piece of content" can be text, an image or an empty block

Depending on how the comment information is stored in your HTML source document, there are several style rules that can be applied. The most common use-cases are to either create a comment from an empty element (where any information is stored in its attributes) or to create a comment from a non-empty element (where the content is the text encompassed by the element):

Example: Creating a comment from an empty element

HTML

```
<span class="comment" text="My Comment."></span>
```

CSS

```
span.comment {
  -ro-comment-content: ro-attr(text);
}
```

Example: Creating a comment from a non-empty element

HTML

```
<span class="comment">This text is commented</span>
```

CSS

```
span.comment {
  -ro-comment-content: content();
}
```

There are different styles to visualize a comment in the PDF:

- **note** Creates a small icon. This is the default style for all comments.
- **invisible** Does not create any visual effects.
- **highlight** Highlights the background of a section of text.
- **underline** Underlines a section of text with a straight line.
- **strikeout** Strikes out a section of text.
- **squiggly** Underlines a section of text with a squiggly line.

The comment styles **highlight**, **underline**, **strikeout** and **squiggly** are only applicable to comments that encompass a section of text.

The following example demonstrates how to style a simple comment.

Example: Styling a comment

HTML

```
<span class="comment">This is a styled comment</span>
```

CSS

```
span.comment {
  -ro-comment-content: content();
  -ro-comment-style: underline;
}
```

Note

The visualization is ultimately dependant on the PDF viewer and may vary across viewers and/or platforms.

Comments can be customized further by using a variety of style rules. Besides content and style, you can also customize the following properties:

- **Title:** The title of the comment. In some cases, this is also used for the author. Use the CSS property `-ro-comment-title` (p. 125) to specify the title.
- **Color:** The color of the comment. The default value for the color depends on the comment style chosen. Use the CSS property `-ro-comment-color` (p. 122) to set a color.
- **Date:** The date of the comment. When no date is specified, the current date is used. Use the CSS property `-ro-comment-date` (p. 123) to set the date.
- **Date Format:** The format of the date you specified. The syntax is identical to Java's `SimpleDateFormat`²⁷. Use the CSS property `-ro-comment-dateformat` (p. 123) to specify a date format for the comment's date.
- **Position:** The position of the comment icon (only applicable for the comment style note). Use the CSS property `-ro-comment-position` (p. 124) to specify a position for the comment's note icon.
- **Initial state:** The initial state of the comment, i.e. whether the comment should be open or closed when the PDF is opened in a viewer. Use the CSS property `-ro-comment-state` (p. 124) to specify the initial state of the comment bubbles. The state can be either `open` or `closed` with the latter being the default value.

The following sample shows how to customize all of the aforementioned properties.

Example: Creating a customized comment

```
.comment {
  /* Content: get the content of the comment from the text content of the element */
  -ro-comment-content: content();
  /* Title: get the title from the "author" attribute of the element */
  -ro-comment-title: ro-attr(author);
  /* Style: set the comment style to "note" */
  -ro-comment-style: note;
  /* Color: specify a color for the comment */
  -ro-comment-color: steelblue;
  /* Date: get the date from the "date" attribute of the element */
  -ro-comment-date: ro-attr(date);
  /* Date Format: specify a custom date format */
  -ro-comment-dateformat: "yyyy/dd/MM HH:mm:ss";
  /* Position: shift the comment icon to the right side of the page */
  -ro-comment-position: page-right;
  /* Initial state: open comment bubbles when the PDF is opened */
  -ro-comment-state: open;
  /* additional styles */
}
```

Please see the documentation of the individual CSS properties for more information.

²⁷ `SimpleDateFormat` API documentation: <http://docs.oracle.com/javase/7/docs/api/java/text/SimpleDateFormat.html>

Advanced Comments

In some cases, comments have a separate start and end tag. In this case the additional style rules `-ro-comment-start` (p. 124) or `-ro-comment-end` (p. 124) have to be set to match the comment's start and end elements.

Example: A comment with different start and end tags

```
commentstart {
  /* some customizations */
  -ro-comment-content: ro-attr(text);
  -ro-comment-title: ro-attr(author);
  -ro-comment-style: highlight;

  /* define the comment start element */
  -ro-comment-start: ro-attr(uid)
}

commentend {
  /* define the comment end element */
  -ro-comment-end: ro-attr(uid);
}
```

4.1.4 Metadata

The title of a generated PDF document, as well as the additional metadata author, subject and keywords, can be specified in multiple ways:

By default the `<title>` tag as well as various `<meta>` tags are read.

The metadata can also be read from other elements using the properties `-ro-title` (p. 159), `-ro-author` (p. 102), `-ro-subject` (p. 157) and `-ro-keywords` (p. 136).

Note:

When a metadata property applies to multiple elements the values are concatenated. Therefore it is recommended to disable the default set elements when specifying other ones:

Example: Set title from first heading

```
/* Disable setting title from title or meta tags */
head * {
  -ro-title: none;
}

/* Set title from first heading */
body > h1:first-of-type {
  -ro-title: content();
}
```

The metadata of the document can be overridden from the API:

```
pdfReactor.setAuthor("John Doe");
pdfReactor.setTitle("Architecture of the World Wide Web, Volume One");
pdfReactor.setSubject("Architecture of the world wide web");
pdfReactor.setKeywords("w3c, www");
```

The code above creates metadata as shown in the screenshot below:

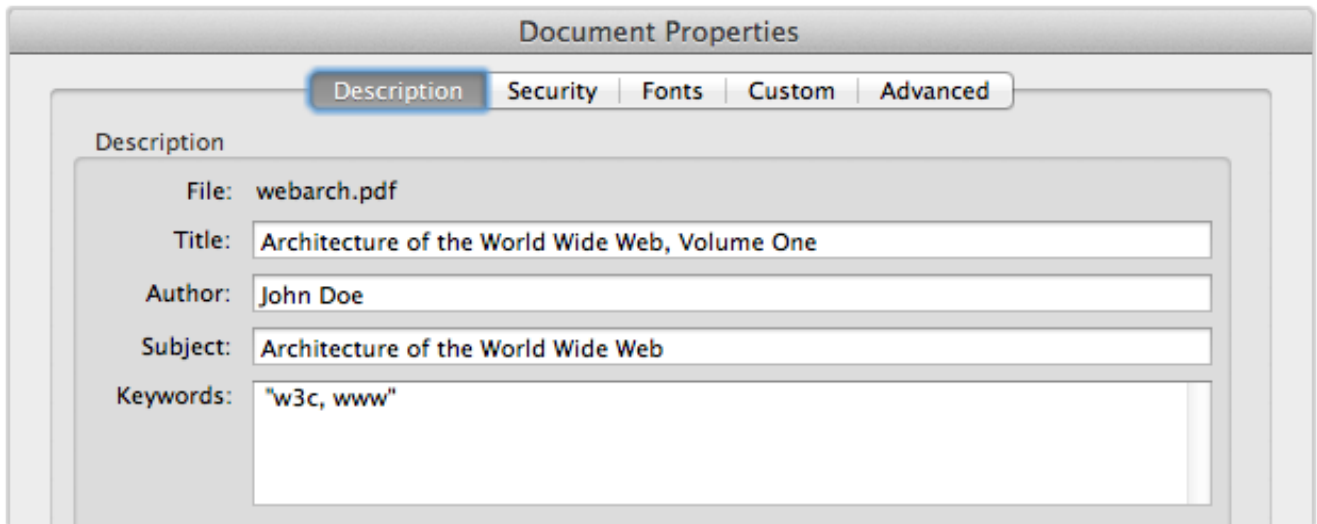


Fig. 2: Document properties dialog of Adobe Reader

Custom Properties

You can also add custom properties to the documents, for which you can define the name and value, e.g.

```
pdfReactor.addCustomDocumentProperty("feedback address", "peter@miller.com");
```

4.1.5 Interactive PDF Forms

HTML forms are automatically rendered by PDFReactor. In addition, you can also convert HTML forms to fully functional interactive PDF forms (sometimes referred to as AcroForms) using the proprietary CSS property `-ro-pdf-format` (p. 149). This property must be specified for the forms you wish to convert to an interactive PDF form.

Example form:

```
<form id="credentials">
  First Name: <input type="text" value="firstname" />
  Last Name: <input type="text" value="lastname" />
  <input type="submit" />
</form>
```

To convert the form with the ID "credentials" to an AcroForm, you can use this style declaration:

```
#credentials, #credentials > input { -ro-pdf-format: pdf; }
```

Using this style declaration, only the form with the ID "credentials" and the input fields contained in this form are converted to an AcroForm when the PDF is rendered. Only the forms and form elements having this CSS style are converted. You can convert all forms and input fields using this CSS code:

```
form, form input { -ro-pdf-format: pdf; }
```

4.1.6 Tagged PDF

Tagged PDF files contain information about the structure of the document. The information about the structure is transported via so-called "PDF tags". Tagging a PDF usually makes it more accessible to screen readers, handhelds and similar devices.

Using the `setAddTags` API method, you can add PDF tags to the PDF documents generated with PDFReactor. If you are generating a PDF from HTML documents, the HTML elements are automatically mapped to the corresponding PDF tags, so all you have to do is setting this property to enable this feature:

```
pdfReactor.setAddTags(true);
```

If you are generating a PDF from another XML language (e.g. DocBook), the elements of this XML language are not mapped to PDF tag types. Instead, they are translated to PDF tags using the XML element name. Thus, the element "para" would be mapped to the PDF tag "para", while the PDF tag type "P" may be more appropriate for this element.

However you can manually map XML elements to PDF tag types using style properties. These style properties are "-ro-pdf-tag-type" and "-ro-alt-text". "-ro-pdf-tag-type" is used to map an element of the XML language you are using to a PDF tag, for example:

```
para { -ro-pdf-tag-type: "P"; }
```

If you were using DocBook, this would map the DocBook element "para" to the PDF tag "P".

The property `-ro-alt-text` (p. 101) is used to specify an alternative description for an XML element. Example:

```
img {
  -ro-pdf-tag-type: "Figure";
}
img[alt] {
  -ro-alt-text: ro-attr(alt);
}
```

The example above maps the HTML element `` to the PDF tag "Figure", and the content of its `alt` attribute to an alternative description for this tag.

If you want to define from which element or attribute in the document the names of the form elements are adopted to a generated PDF, you can use the property `-ro-formelement-name` (p. 133). By default, the names are adopted from the `value` attribute of the form element.

Using the `-ro-radiobuttonelement-group` (p. 152), the name for radio button groups can be adopted in the same way. By default, it will be adopted from the `name` attribute of the radio button element.

4.1.7 PDF/A Conformance

PDFReactor supports the creation of PDF/A-1a²⁸ or PDF/A-3a²⁹ conformant files.

PDF/A³⁰ is a family of ISO³¹ standards for long-term archiving of documents. The goal of these standards is to ensure the reproduction of the visual appearance as well as the inclusion of the document's structure. All information necessary for displaying the document in the same way every time is embedded in the file. Dependencies on external resources is not permitted.

Many companies and government organizations worldwide require PDF/A conformant documents.

PDF/A-1a is the most strict PDF/A standard while the newer PDF/A-3a is more lenient, e.g. allowing transparency and attachments.

²⁸ Synonym for ISO 19005-1 Level a (<http://www.pdfa.org/>)

²⁹ Synonym for ISO 19005-3 Level a (<http://www.pdfa.org/>)

³⁰ Synonym for the ISO 19005 family of standards (<http://www.pdfa.org/>)

³¹ International Organization for Standardization (<http://www.iso.org/iso/home.htm>)

Common PDF/A conformance requirements

PDF/A restriction	PDFReactor actions
All used fonts are embedded.	PDFReactor ignores the option to disable font embedding.
All images are embedded.	Images are always automatically embedded by PDFReactor.
Multi-media content is forbidden.	Embedding objects is automatically prevented by PDFReactor, when PDF/A conformance is set.
JavaScript is prohibited.	No JavaScript is embedded, when PDF/A conformance is set. (This does not prohibit JavaScript in the source HTML document to be processed during conversions)
Encryption is disallowed.	This is automatically prevented, when PDF/A conformance is set.
The PDF must be tagged.	This is automatically done by PDFReactor, when PDF/A conformance is set.
Metadata included in the PDF is required to be standard-based XMP ³² .	This is automatically done by PDFReactor, when PDF/A conformance is set.
Colors are specified in a device-independent manner.	In PDFReactor colors are defined either as RGB or CMYK. When PDF/A conformance is set, one of these color spaces has to be set in conjunction with a color space profile. CMYK requires an ICC profile to be set, RGB colors use a default sRGB ³³ profile, if no other is set.

PDF/A-1a specific conformance requirements

PDF/A-1a restriction	PDFReactor actions
Transparency is disallowed	PDFReactor will ignore certain kinds of transparency of images. Other occurrences of transparency will cause an exception to be thrown.
Attachments are disallowed	This is automatically prevented, when PDF/A-1a conformance is set.

To create a PDF/A conformant document, the method `setConformance` can be used in the PDFReactor integration:

```
pdfReactor.setConformance(PDFReactor.CONFORMANCE_PDFA1A);
```

or

```
pdfReactor.setConformance(PDFReactor.CONFORMANCE_PDFA3A);
```

If CMYK³⁴ colors are used in a document to be converted into a PDF/A-conformant file, an Output Intent has to be set. It is possible to use the following API methods:

```
pdfReactor.setOutputIntentFromURL(String identifier, String profileUrl);
```

or

```
pdfReactor.setOutputIntentFromByteArray(String identifier, byte[] profileData);
```

The first parameter is a string identifying the intended output device or production condition in human- or machine-readable form. The second parameter points to an ICC³⁵ profile file or contains data of such a profile.

³² Extensible Metadata Platform, standard for processing and embedding metadata (<http://www.adobe.com/products/xmp/index.html>)

³³ a standard RGB profile (<http://www.w3.org/Graphics/Color/sRGB>)

³⁴ Cyan Magenta Yellow Key, subtractive color model, consisting of the color components cyan, magenta, yellow and black.

Note:

When PDF/A conformance is set, encryption, restrictions, comments and other non-PDF/A-conformant features are automatically overwritten, regardless of their own settings.

4.1.8 Print Dialog Prompt

PDFReactor can be configured to immediately display a print dialog when a PDF file created with PDFReactor is opened. To do so, the `setPrintDialogPrompt` method of the PDFReactor class must be used. Example:

```
pdfReactor.setPrintDialogPrompt(true);
```

4.1.9 PDF Compression

Using the API method `setFullCompression`, PDF files can be generated with full compression, thus reducing the file size of the resulting PDF document.

Example usage:

```
pdfReactor.setFullCompression(true);
```

Your PDF reader needs to support Adobe PDF version 1.5 in order to be able to display PDF documents created with full compression enabled.

4.1.10 Encryption and Restrictions

PDFReactor can protect generated PDF documents via 40 or 128 bit encryption.

To encrypt the output PDF, set the encryption strength to a value other than `ENCRYPTION_NONE`:

```
pdfReactor.setEncryption(PDFReactor.ENCRYPTION_128);
```

When the PDF document is opened, the user has to supply the user password in order to view the content. When no user password is set, the PDF can be viewed by any user. In either case, certain restrictions are imposed. These can be suspended by supplying the owner password. You can set the passwords as follows:

```
pdfReactor.setUserPassword("upasswd");
pdfReactor.setOwnerPassword("opasswd");
```

Both passwords are optional, but recommended for security reasons.

By default, all restrictions are imposed on the PDF document. You can however exclude selected ones by using the following methods:

List of methods to disable restrictions

Method name	Allows ...
<code>setAllowPrinting</code>	printing
<code>setAllowCopy</code>	copying or otherwise extracting content
<code>setAllowAnnotations</code>	adding or modifying annotations and interactive form fields
<code>setAllowModifyContents</code>	modifying the content of the document

Method name	Allows ...
setAllowDegradedPrinting	printing (same as setAllowPrinting, however with a limited resolution) (128 bit encryption only)
setAllowFillIn	filling in form fields (128 bit encryption only)
setAllowAssembly	inserting, removing and rotating pages and adding bookmarks (128 bit encryption only)
setAllowScreenReaders	extracting content for use by accessibility devices (128 bit encryption only)

See:

API docs for further information.

4.1.11 Page Preview Images

Using the `setAddPreviewImages` API method, you can use PDFReactor to add page preview images to your PDF file. This is recommended if the PDF reader you are using does not offer an automatic preview of pages or if you are rendering highly complex documents containing many graphical elements.

Example usage:

```
pdfReactor.setAddPreviewImages(true);
```

The Adobe Reader starting from version 5.0 supports an automatic preview of pages. Furthermore, adding page preview images to the rendered PDF increases file size.

4.1.12 Viewer Preferences

You can configure the initial presentation of the document in the viewer by setting viewer preferences. Setting viewer preferences will activate / deactivate certain options of the viewer, for example it allows to hide the viewer's toolbar when the document is opened.

Note that these preferences are not enforced, i.e. if you decide to set the `HIDE_TOOLBAR` preference, the user can still display the toolbar again when viewing this PDF if he decides to do so. Setting this preference only affects the default state of the toolbar when the document is opened, but does not enforce this state.

Some viewer preferences also influence the default settings of the print dialog of the viewer.

You can set viewer preferences by using the method `setViewerPreferences`. Multiple preferences can be combined using the OR operator, e.g.:

```
pdfReactor.setViewerPreferences(
    PDFReactor.VIEWER_PREFERENCES_PAGE_LAYOUT_SINGLE_PAGE |
    PDFReactor.VIEWER_PREFERENCES_DISPLAY_DOC_TITLE);
```

PDFReactor supports the following viewer preferences:

List of Viewer Preferences

Viewer Preference	Effect
PAGE_LAYOUT_SINGLE_PAGE	Display one page at a time. (default)
PAGE_LAYOUT_ONE_COLUMN	Display the pages in one column.

Viewer Preference	Effect
PAGE_LAYOUT_TWO_COLUMN_LEFT	Display the pages in two columns, with odd numbered pages on the left.
PAGE_LAYOUT_TWO_COLUMN_RIGHT	Display the pages in two columns, with odd numbered pages on the right.
PAGE_LAYOUT_TWO_PAGE_LEFT	Display two pages at a time, with odd numbered pages on the left.
PAGE_LAYOUT_TWO_PAGE_RIGHT	Display two pages at a time, with odd numbered pages on the right.
PAGE_MODE_USE_NONE	Show no panel on startup.
PAGE_MODE_USE_OUTLINES	Show document outline panel on startup.
PAGE_MODE_USE_THUMBS	Show thumbnail images panel on startup.
PAGE_MODE_FULLSCREEN	Switch to fullscreen mode on startup.
PAGE_MODE_USE_OC	Show optional content group panel on startup.
PAGE_MODE_USE_ATTACHMENTS	Show attachments panel on startup.
HIDE_TOOLBAR	Hide the viewer application's tool bars when the document is active.
HIDE_MENUBAR	Hide the viewer application's menu bar when the document is active.
HIDE_WINDOW_UI	Hide user interface elements in the document's window.
FIT_WINDOW	Resize the document's window to fit the size of the first displayed page
CENTER_WINDOW	Position the document's window in the center of the screen.
DISPLAY_DOC_TITLE	Display the document's title in the top bar.
NON_FULLSCREEN_PAGE_MODE_USE_NONE	Show document outline panel on exiting full-screen mode. Has to be combined with PageModeFullScreen.
NON_FULLSCREEN_PAGE_MODE_USE_OUTLINES	Show thumbnail images panel on exiting full-screen mode. Has to be combined with PageModeFullScreen.
NON_FULLSCREEN_PAGE_MODE_USE_THUMBS	Show optional content group panel on exiting full-screen mode. Has to be combined with PageModeFullScreen.
NON_FULLSCREEN_PAGE_MODE_USE_OC	Show attachments panel on exiting full-screen mode. Has to be combined with PageModeFullScreen.
DIRECTION_L2R	Position pages in ascending order from left to right.
DIRECTION_R2L	Position pages in ascending order from right to left.
PRINTSCALING_NONE	Print dialog default setting: disabled scaling
PRINTSCALING_APPDEFAULT	Print dialog default setting: set scaling to application default value
DUPLEX_SIMPLEX	Print dialog default setting: simplex
DUPLEX_FLIP_SHORT_EDGE	Print dialog default setting: duplex (short edge)
DUPLEX_FLIP_LONG_EDGE	Print dialog default setting: duplex (long edge)

Viewer Preference	Effect
PICKTRAYBYPDFSIZE_FALSE	Print dialog default setting: do not pick tray by PDF size
PICKTRAYBYPDFSIZE_TRUE	Print dialog default setting: pick tray by PDF size

4.1.13 Merging PDFs

A generated PDF can easily be merged with existing ones. To merge with a single PDF use the `setMergeURL(String)` method that declares the path to the PDF file.

```
pdfReactor.setMergeURL("http://www.myserver.com/overlaid.pdf");
```

To merge multiple PDFs use the methods `setMergeURLs` or `setMergeByteArrays`.

```
String[] urls = {"http://www.myserver.com/overlaid1.pdf",
    "http://www.myserver.com/overlaid2.pdf"};
pdfReactor.setMergeURLs(urls);
```

Whether the existing PDFs are appended to the generated PDF or whether the generated PDF is laid over them depends on the general type of merge:

- Concatenation
- Overlay

Concatenation merges append PDFs before or after the generated PDF. The following sample shows how to append an existing PDF after the generated one:

```
pdfReactor.setMergeURL("http://www.myserver.com/appendDoc.pdf");
pdfReactor.setMergeMode(PDFReactor.MERGE_MODE_APPEND);
```

Overlay merges add the generated PDF above or below existing PDFs. The following sample shows how to overlay an existing PDF:

```
pdfReactor.setMergeURL("http://www.myserver.com/overlaid.pdf");
pdfReactor.setMergeMode(PDFReactor.MERGE_MODE_OVERLAY);
```

PDFReactor allows to repeat the pages of PDFs with less pages than other PDFs involved in the merger. The API method `setOverlayRepeat` offers different options to do this:

- repeat only the last page
- repeat all pages of the PDF

In the following example, all pages are repeated:

```
pdfReactor.setOverlayRepeat(PDFReactor.OVERLAY_REPEAT_ALL_PAGES);
```

The default merge behavior of PDFReactor is a Concatenation after the pages of the generated PDF.

4.1.14 Digital Signing

PDFReactor is able to sign the PDFs it creates. This allows to validate the identity of the creator of the document. A self-signed certificate may be used. A keystore file in which the certificate is included, is required.

The keystore type is required to be one of the following formats:

- "pkcs12"
- "jks"

PDFReactor supports the following three modes to sign a PDF:

- Self-signed
- Windows certificate
- VeriSign

To sign a PDF digitally use the API method `setSignPDF`.

Note:

If a PDF is signed via the VeriSign signing mode, a plugin for the PDF viewer is required to show the signature.

4.1.15 Font Embedding

By default, PDFReactor automatically embeds the required subsets of all fonts used in the document. This can be disabled using the method `setDisableFontEmbedding`.

```
pdfReactor.setDisableFontEmbedding(true)
```

Doing so reduces the file size of the resulting PDF documents. However, these documents are likely to not look the same on all systems. Therefore this method should only be used when necessary.

4.1.16 Overprinting

Overprinting means that one color is printed on top of another color. As this is a feature for printing it should be used with CMYK colors.

PDFReactor can set the values of the PDF graphics state parameters `overprint` and `overprint mode` via CSS. This can be enabled using the API method `setAddOverprint`:

```
pdfReactor.setAddOverprint(true)
```

Using the styles `-ro-pdf-overprint` (p. 150) and `-ro-pdf-overprint-content` (p. 150) you can specify the overprint properties of elements and their content to either `none` (default), `mode0` or `mode1` (nonzero overprint mode).

`-ro-pdf-overprint` affects the entire element, while `-ro-pdf-overprint-content` only affects the content of the element (not its borders and backgrounds). In both cases the children of the element are affected entirely, unless overprint styles are applied to them as well.

The following example sets small text on solid background to overprint, without enabling overprinting for the background of either the paragraphs or the highlighting spans:

```
p.infobox {
  border: 1pt solid black;
  background-color: lightgrey;
  color: black;
  font-size: 8pt;
  -ro-pdf-overprint-content: mode1;
}
p.infobox span.marked {
  background-color: yellow;
  -ro-pdf-overprint: none;
  -ro-pdf-overprint-content: mode1;
}
```

Note:

When having small text with a background, overprinting can be very helpful to avoid white lines around the text, if the printing registration is imperfect.

4.1.17 Attachments

Alternatively to linking to external URLs (see [Links \(p. 34\)](#)) PDFReactor also allows embedding their content into the PDF.

Attachments can be defined via CSS, which can be enabled by the API method `setAddAttachments`:

```
pdfReactor.setAddAttachments(true)
```

The following styles can be used to specify attachments:

- `-ro-pdf-attachment-url` (p. 149):

A URL pointing to the file to be embedded. This URL can be relative. Specifying "#" will embed the source document.

- `-ro-pdf-attachment-name` (p. 149):

The file name associated with the attachment. It is recommended to specify the correct file extension. If this is not specified the name is derived from the URL.

- `-ro-pdf-attachment-description` (p. 148):

The description of the attachment. If this is not specified the name is used.

- `-ro-pdf-attachment-location` (p. 148):

- `element` (default): The attachment is related to the area of the element. Viewers may show a marker near that area.
- `document`: The file is attached to the document with no relation to the element.

Attachments can be specified for specific elements as follows:

```
#downloadReport {
  -ro-pdf-attachment-url: "../resources/0412/report.doc";
  -ro-pdf-attachment-name: "report-2012-04.doc";
  -ro-pdf-attachment-description: "Report for April of 2012";
}
```

Strings can be dynamically read from the document using the CSS functions `ro-attr` and `content` (p. 126), that read specified attributes or the text content of the element respectively. Using those, certain a-tags can be changed from links to attachments:

```
.downloadReports a[href] {
  -ro-link: none;
  -ro-pdf-attachment-url: ro-attr(href);
  -ro-pdf-attachment-description: content() " (" ro-attr(href) ")";
}
```

Attachments can also be set via the API method `addAttachment`. This method also allows specifying the content of the attachment as a byte array instead of an URL, so dynamically created data can be attached:

```
pr.addAttachment("sample attachment text".getBytes(), null,
  "sample.txt", "a dynamically created attachment containing text");
pr.addAttachment(null, "../resources/0412/report.doc",
  "report-2012-04.doc", "Report for April of 2012");
```

4.2 Images

In addition to PDF, PDFReactor supports the following image output formats:

- PNG (optionally with transparent background)
- JPEG
- GIF
- TIFF (only multi-page image output format; can use the following compression methods: LZW, PackBits, Uncompressed, CCITT 1D, CCITT Group 3 & CCITT Group 4)
- BMP

These can be selected using the method `setOutputFormat`, e.g.:

```
pdfreactor.setOutputFormat(PDFReactor.OUTPUT_FORMAT_PNG, 512, -1)
```

The later two parameters set the width and height of the resulting images in pixels. If either of these is set to a value of less than 1 it is computed from the other value and the aspect ratio of the page.

4.2.1 Selecting a page

All image output formats, except for the TIFF formats, create an image of a single page. By default, this is the first page. A different page can be selected using the method `setPageOrder`, e.g.:

```
pdfReactor.setPageOrder("5");
```

4.2.2 Converting a Document Into Multiple Images

The methods `renderDocumentFromURLAsArray`, `renderDocumentFromContentAsArray` and `renderDocumentFromByteArrayAsArray` convert a document, returning an array of byte-arrays each containing an image representing one page of the document.

4.2.3 Continuous Output

The method `setContinuousOutput` sets PDFReactor to continuous mode. In this mode each document is converted into one image. Also screen styles will be used and print styles will be ignored, resulting in a very browser-like look for the output image.

```
pdfReactor.setContinuousOutput(1024, 768);
```

The first parameter sets the width of the layout. This has the same effect as the width of a browser window. This only changes the layout. The result will still be scaled to the width specified by `setOutputFormat`

The second parameter sets the height. This has the same effect as the height of a browser window, i.e. it will cut off the image or increase its height. Values of less than 1 cause the full height of the laid out document to be used.

5. LAYOUT DOCUMENTS

This chapter provides information on how to lay out a document.

The document layout mostly depends on CSS but there are PDFReactor API methods and JavaScript functionality that may be of use too to achieve the desired results.

Basic knowledge about CSS is recommended.

5.1 Basic Page Layout

PDFReactor renders HTML and XML documents on pages. The rules to achieve that are provided by CSS.

The document content is laid out page by page, whenever there is no more space left on a page, PDFReactor automatically breaks text and boxes to the next.

Note:

Basic page styles are provided for HTML. Page style for XML documents need to be created based on the document language.

5.1.1 Page Selectors

To create an individual page layout pages need to be selected with CSS. In principle it works the same way as selecting an element, but the selector is different.

To select all pages of the document, the `@page` rule is used instead of the usual element selector.

Example: A one inch wide page margin on all pages.

```
@page {
    margin: 1in
}
```

Using the pseudo-classes `:first`, `:left` and `:right` makes it possible to select the first, all left or all right pages. This allows to define different style for certain pages, e.g. for a title page.

Example: Definition of inside and outside margins on left and right pages.

```
@page{
    margin-top: 2in;
    margin-bottom: 6in
}
@page:left{
    margin-left: 4in;
    margin-right: 3in
}
@page:right{
    margin-left: 3in;
    margin-right: 4in
}
```

5.1.2 Page Size & Orientation

The size and orientation of a page can be set with the `size` (p. 155) property. PDFReactor supports many different page sizes, see Appendix [Supported Page Size Formats](#) (p. 90).

Example: All pages in format 'letter' and portrait orientation.

```
@page{
  size: letter portrait
}
```

To set a page to landscape orientation, "portrait" is replaced by "landscape":

Example: All pages in format 'letter' and landscape orientation.

```
@page{
  size: letter landscape
}
```

Instead of setting fixed page formats with a specified orientation it is also possible to set two length values. These then define page size and orientation.

Example: A page size of 4.25 inches by 6.75 inches for all pages.

```
@page{
  size: 4.25in 6.75in
}
```

5.1.3 Named Pages

With Named Pages an element is able to create and appear on a special page that has a name. This name can be used as part of a Page Selector to attach additional style properties to all pages of that name.

To create a Named Page, an element receives the `page` (p. 146) property with a page name as identifier.

Example: All HTML `<table>` elements have to appear on pages with the name `pageName`.

```
table{
  page: pageName
}
```

A page break will be inserted before an element that has the `page` property set. Another page break will be inserted for the next element that defines a different page name (or none) to ensure the Named Page only contains elements that specify its name.

To attach style to a named page, the page name is added to the `@page` rule. The page name is not a pseudo-class as `:first` is. There is a space between `@page` and the page name, not a colon.

Example: A Named Page with 'letter' format and landscape orientation.

```
@page pageName{
  size: letter landscape
}
```

5.2 Breaking Text

Text is broken whenever there is not enough space left, e.g. inside the line or on the page.

5.2.1 Automatic Hyphenation

Automatic Hyphenation allows breaking words in a way appropriate for the language of the word.

To use Automatic Hyphenation two requirements must be met:

- The text to hyphenate requires a language set in the document.
- The language set for the hyphenated text is supported by PDFReactor (see Appendix [Hyphenation Dictionaries](#) (p. 91) for more information)

The `lang` attribute in HTML or the `xml:lang` attribute in XML allow defining a language for the document and on individual elements, in case they deviate from the document language.

Example: An entire HTML document in English language.

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

Hyphenation is enabled or disabled via CSS with the [hyphens](#) (p. 135) property:

Example: Hyphenation enabled for an entire document except for paragraphs of the `noHyphenation` class.

```
html {
  hyphens: auto
}
p.noHyphenation {
  hyphens: none
}
```

In addition it is possible to specify the number of minimum letters before or after which text can be broken within a word. This is done with the [hyphenate-before](#) (p. 135) and [hyphenate-after](#) (p. 134) properties.

5.2.2 Widows & Orphans

Definition: Widow

If the last line of a paragraph is also the first line of a page it is called a widow.

Definition: Orphan

If the first line of a paragraph is also the last line of a page it is called an orphan.

By default, PDFReactor avoids widows and orphans by adding a page break before the paragraph. This behaviour can be changed with the CSS properties [widows](#) (p. 165) and [orphans](#) (p. 144).

Example: Widows & Orphans set to an amount of two lines.

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

Changing the value to 1 will allow widows and orphans. Changing it to higher integer values will prevent even multiple line widows and orphans. (e.g.: `orphans: 5` means that if the first 4 lines of a paragraph are the last 4 lines of a page these lines are considered an orphan.)

5.3 Generated Content

Generated content does not originate from the document. It is created by CSS during the rendering process and appears in the rendered result as if it was part of the document.

The pseudo-elements `::before` and `::after` are used to generate content before or after an element. The actual content is created with the `content` (p. 126) property.

5.3.1 Generated Text

To create generated text, set a String as value of the `content` property.

Example: Generated Text on an HTML `<div>` element.

HTML:

```
<div>This is a note.</div>
```

CSS:

```
div::before{
  /* Adds the text "Note:" at the start of the element. */
  content: "Note: ";
  padding-right: 0.1in;
  font-weight: bold
}
div{
  border: 1px solid black;
  background-color: palegoldenrod;
  padding: 0.1in
}
```

As a result, the `<div>` would look like this:

Note: This is a note.

Sometimes it is necessary to add an explicit line break to generated text. To create such a line break, a `"\A"` needs to be added to the String and the `white-space` (p. 164) property needs to be set to either `pre`, `pre-wrap` or `pre-line`.

Example: An explicit line break inside Generated Text.

```
div::before{
  content: "RealObjects\APDFReactor";
  white-space: pre
}
```

The result would look like this:

RealObjects
PDFReactor

If the first character after the line break is an HTML entity, add an additional space between the `"\A"` and the entity.

5.3.2 Generated Images

A generated image can be created with the image's URL set as value of the content property.

Example: A Generated Image with an SVG image as source.

```
h1::before{
  content: url(http://mydomain/pictures/image.svg)
}
```

5.3.3 Counters

Counters can be used to count elements or pages and then add the value of the Counter to generated text.

A Counter needs to be defined either with the `counter-reset` (p. 127) or the `counter-increment` (p. 127) property. Its value is read with the `counter()` function as value of the content property.

A common use-case for Counters are numbered headings. The chapter heading of a document is intended to display a number in front of its text that increases with each chapter.

Example: A chapter heading for HTML `<h1>` elements using Counters and Generated Text.

```
h1{
  /* increases the counter "heading1" by 1 on each <h1> element */
  counter-increment: heading1 1
}
h1::before{
  /* Adds the current value of "heading1" before the <h1> element's
  text as decimal number */
  content: counter(heading1, decimal)
}
```

Subchapter headings, work the same way, with a simple addition. The number of each subchapter is intended to be reset whenever a new chapter begins. To restart numbering, the `counter-reset` property is used.

Example: Subchapter headings with Counters reset every chapter.

```
h1{
  /* resets the value of counter "heading2" to 0 on every <h1> element */
  counter-reset: heading2 0
}
h2{
  counter-increment: heading2 1
}
h2::before{
  /* Shows the current value of "heading1" and "heading2", separated by a
  generated text ".", the value of "heading2" is shown as lower-case
  letter */
  content: counter(heading1, decimal) "." counter(heading2, lower-alpha)
}
```

5.4 Page Header & Footer

5.4.1 Header, Footer & Page Side Boxes

It is possible to add Generated Content to a page within the page margin. The page margin is the space between the document content and the edges of a sheet. It is defined on a page using [Page Selectors](#) (p. 49) and the `margin` (p. 140) property.

Each page provides sixteen Page Margin Boxes that can display Generated Content much like a pseudo-element. To add Generated Content to a page, add a Page Margin Box declaration to an existing `@page` rule and set the Generated Content to the `content` (p. 126) property as usual.

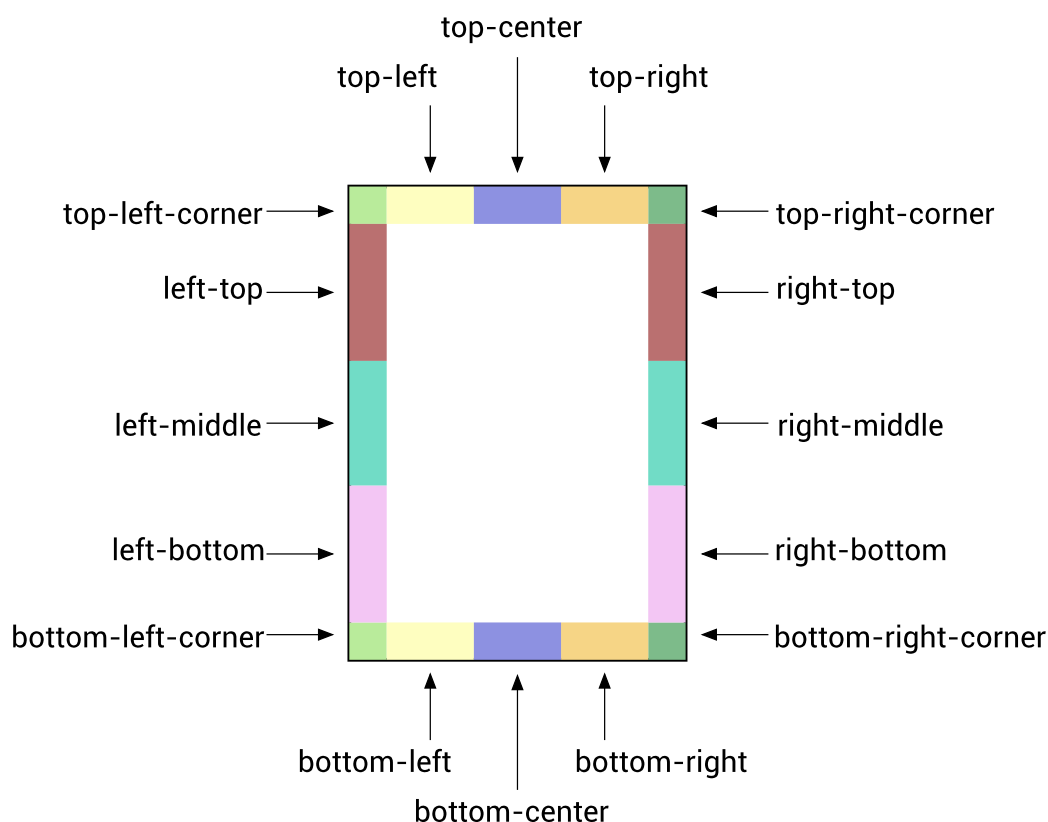


Fig. 3: Page Margin Boxes

A Page Margin Box declaration consists of an "@" character followed by the name of the Page Margin Box.

Example: Page Header with Generated Text on the left and right side.

```
@top-left{
  content: "RealObjects PDFReactor(R)";
}
@top-right{
  content: "copyright 2015 by RealObjects";
}
```

5.4.2 Running Elements

Running Elements are elements inside the document that are not rendered inside the document content but inside Page Margin Boxes.

They are useful whenever the content of a Page Margin Box needs to be more complex than Generated Content (e.g. a table) or parts of it need to be styled individually.

Note

In case the document does not provide elements to use Running Elements and Generated Content does not suffice, it is possible to add elements to the document with JavaScript to be able to use Running Elements.

To create a Running Element, an element needs to be positioned as "running", using the `running()` function with an identifier for the element as argument. The function is set as value of the `position` (p. 151) property. This removes the element from the document content.

To display a Running Element inside a Page Margin Box, set the `element()` function as value of the `content` property. The argument of the function is the same identifier used to in the `running()` function of the Running Element.

Example: An HTML `<footer>` element at the start of the document used as page footer in all pages.

HTML:

```
<body>
  <footer>...</footer>
  ...
</body>
```

CSS:

```
footer{
  position: running(footerIdentifier)
}
@page{
  @bottom-center{
    content: element(footerIdentifier)
  }
}
```

The `<footer>` needs to be at the beginning of the HTML document to guarantee, that it will appear on every page of the document.

The reason for that is, that running elements stay anchored to the location they would appear in if they were not Running Elements.

The original position of the running element inside the document plays a key role when designing a document, it provides document designers with additional options.

First of all it is possible to have running elements of the same name, which makes it possible to change the content of a Page Margin Box over the course of the document.

Example: Two Running Elements at the start of the document with the same name. The first appears on page one, the second on every page thereafter because it is the latest Running Element of the name.

HTML:

```
<body>
  <header id="titlePageHeader">...</header>
  <header id="pageHeader">...</header>
  <!-- first page content -->
  ...
  <!-- second page content -->
  ...
</body>
```

CSS:

```
#titlePageHeader, #pageHeader{
  position: running(headerIdentifier)
}
@page{
  @top-center{
    content: element(headerIdentifier)
  }
}
```

Second of all it is possible to have running elements appear for the first time later in the document than on the first page.

Example: An HTML `<footer>` element at the end of the document is as Running Element. The page footer displays it in the last page only, as it is not available earlier.

HTML:

```
<body>
  ...
  <footer>...</footer>
</body>
```

CSS:

```
footer{
  position: running(footerIdentifier)
}
@page{
  @bottom-center{
    content: element(footerIdentifier)
  }
}
```

Notice how the style does not differ from the one used in the first example of this chapter. This shows how much influence the position of a Running Element inside the document has.

It is possible that more than one Running Element of the same name would anchor on the same page. Sometimes, it may not be the first Running Element on a page that should be used for that page. For that case it is possible to add one of these identifiers as second argument to the `element()` function:

- `start`
 - Retrieves the latest Running Element of the name from previous pages.
 - If there is none, nothing is displayed.
- `first`
 - Retrieves the first Running Element of the name on the page.
 - If there is none, it falls back to the behavior of `start`.
 - This is the default behavior if no argument is given.
- `last`
 - Retrieves the last Running Element of the name on the page.
 - If there is none, it falls back to the behavior of `start`.
 - This keyword is useful in case a Running Element is displayed as footer throughout the document but the last page should receive a different Running Element, which is placed at the end of the document.
- `first-except`
 - If a Running Element of the name is on the page, nothing is displayed.
 - If there is none, it falls back to the behavior of `start`.
 - This keyword is useful on chapter title pages where the chapter name is already displayed.

Note

If a Running Element or its contents define Generated Content that contains [Counters \(p. 53\)](#) (or [Named Strings \(p. 58\)](#)) their value will be the same as if they were defined as content of the Page Margin Box the Running Element is used in.

5.4.3 Running Documents

In case [Generated Content \(p. 52\)](#) does not suffice and [Running Elements \(p. 54\)](#) are not an option, it is possible to use Running Documents inside Page Margin Boxes.

A Running Document is a String containing an HTML document or document fragment or a URL that references a document as argument of the `xhtml()` function.

Note

The `xhtml()` function is a proprietary extension of CSS and will only work for RealObjects products.

Example: Variants of `xhtml()` function declarations:

```
/* document fragment */
content: xhtml("<table>...</table>");
/* complete document */
content: xhtml("<html><head>...</head><body>...</body></html>");
/* external document */
content: xhtml(url("header.html"));
```

The document is loaded independantly inside the Page Margin Box but styles from the document are passed down to it. This can be an advantage as the same style is used throughout all documents. In some cases though this behavior is not desired as this style may break the layout of the document inside the Page Margin Box. To prevent passing down style the `-ro-passdown-styles` ([p. 148](#)) property is used.

Important

When using the `xhtml()` function in non-HTML5 documents (e.g. XHTML inside the head in a `<style>` element) the entire CSS needs to be wrapped in an XML comment.

```
<!--
@page {
  @top-center{
    content: xhtml("<table>...</table>");
  }
}
-->
```

Note

Running Documents have access to [Counters \(p. 53\)](#) and [Named Strings \(p. 58\)](#) from their embedding document and may display them, but cannot influence them.

Counters and Named Strings created inside Running Documents have no effect outside of the Running Document.

5.5 Generated Content for Pages

Additional features for [Generated Content \(p. 52\)](#) are available within [Page Margin Boxes \(p. 53\)](#).

5.5.1 Page Counters

To add page numbers to a document, Page Counters are used. Page Counters work like [Counters \(p. 53\)](#), but are defined on the [Page \(p. 49\)](#) and accessed in [Page Margin Boxes \(p. 53\)](#).

The following example creates a Page Counter and a simple page footer that shows the page number on the bottom right of the page.

Example: A Page Counter used at the bottom right of the page to display the page number.

```
@page{
  counter-increment: page 1
  @bottom-right{
    content: counter(page)
  }
}
```

To retrieve the total number of pages PDFReactor provides the Total Pages Counter. The Total Pages Counter is a counter provided on application level, meaning it can be accessed without defining it first by the name `pages`.

Example: A declaration of the Page Counter and the Total Pages Counter.

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

Note:

When the output of PDFReactor is appended after an existing PDF document, the Page Counter of the output generated by PDFReactor would not continue with the number of pages of the PDF document it will be appended to by default.

Setting the value of the Page Counter to `applicationValue("com/realobjects/pdfreactor/start-page-number")` function on the first page ensures the page numbering is continued when appending the output of PDFReactor to an existing PDF document.

Example:

```
@media print{
  @page:first {
    counter-reset: page
    applicationValue("com/realobjects/pdfreactor/start-page-number");
  }
  @page{
    counter-increment: page;
  }
}
```

Note:

To reset a Counter to a specific number from within the content, you can use the property `-ro-counter-set` (p. 127).

The following sample shows how to reset the Page Counter to 1 for every `h1` element:

```
h1 {
  -ro-counter-set: page 1;
}
```

5.5.2 Named Strings

Named Strings allow to store the text of an element and its [Generated Content](#) (p. 52) as String for use in [Page Margin Boxes](#) (p. 53).

A Named String is defined very similar to a [Counter \(p. 53\)](#) and is used in a similar way. To create a Named String the property `string-set` ([p. 156](#)) is used, which requires an identifier and a definition of the contents of the String. To read a Named String the `string()` function is used as value of the content property.

Example: A NamedString "headingString" created from the heading's text with the keyword `self` and read with the `string()` function from the page header.

```
h1 {
  string-set: headingString self;
}
@page{
  @top-left{
    content: string(headingString);
  }
}
```

The content of a named String is very flexible and can take a combination of Strings, `counter()` functions and Named String keywords.

Example: Variations of Named String declarations.

```
/* Creates a Named String in the form of "Chapter [chapter number]: [chapter title]". */
h1{
  string-set: headingString "Chapter " before ": " self
}
/* Retrieves the first letter of an address element, useful as part of a page header
   for a sorted list of addresses */
address{
  string-set: addressEntry first-letter;
}
```

Named Strings are similar to [Running Elements \(p. 54\)](#) in that they can occur multiple times on the same page and are accessed from Page Margin Boxes. Similar to the `element()` function, the `string()` function allows to add a second argument to specify which Named String inside a page should be used:

- `start`
 - Retrieves the latest Named String of the name from previous pages.
 - If there is none, nothing is displayed.
- `first`
 - Retrieves the first Named String of the name on the page.
 - If there is none, it falls back to the behavior of `start`.
 - This is the default behavior if no argument is given.
- `last`
 - Retrieves the last Named String of the name on the page.
 - If there is none, it falls back to the behavior of `start`.
- `last-except`
 - If a Named String of the name is on the page, nothing is displayed.
 - If there is none, it falls back to the behavior of `start`.

5.5.3 Cross-references

A Cross-reference is a piece of text that references another location in the document in order to establish a thematic relationship to that location.

Although it is perfectly possible to add such references by hand, this approach is prone to error when creating and modifying the document. After a change the numbering and page numbers might not match the numbering from when the cross-reference was first defined. The same could happen to the reference text if it includes the chapter title.

To automatically keep the reference up-to-date with the referenced location, CSS provides the `target-counter()` and `target-text()` functions to automatically retrieve the exact numbering, title or page number of the referenced location.

Note:

PDFReactor only resolves internal links referring to an anchor in the same input document, see the chapter [Links \(p. 34\)](#) for more information.

Counter Cross-references

The `target-counter()` function is used inside the `content` property the same way a `counter()` function would be used. It receives a URL to the referenced location and the name of the counter as identifier. It may receive an optional third argument to define the output style of the counter, just like the `counter()` function.

Example: Cross-references created from an HTML hyperlink to a chapter heading with a numbering. The Cross-reference is declared with generated text and `target-counter()` functions to retrieve the page and chapter numbers.

HTML:

```
...
<p>For more information <a href="#chapter">see</a>.
...
<h1 id="chapter">Cross-references</h1>
...
```

CSS:

```
@page{
  counter-increment: pageCounter;
  @bottom-right{
    content: counter(pageCounter);
  }
}
h1{
  counter-increment: chapterCounter;
}
h1::before{
  content: counter(chapterCounter, upper-roman);
}
a[href]::after{
  content: "Chapter " target-counter(ro-attr(href url), chapterCounter, upper-roman)
          " on page " target-counter(ro-attr(href url), pageCounter);
}
```

Assuming the referenced chapter would render on page 5 as the third chapter, the cross-reference would read:

For more information, [see Chapter III on page 5.](#)

Text Cross-references

The `target-text()` function is used inside the content property in a similar way as the `target-counter()` function is used. It receives a URL to the referenced location and takes one of these four keywords to specify the text to retrieve:

- `content` - Retrieves the textual content of the element. This is the default keyword if no keyword is present.
- `first-letter` - Retrieves the first letter of the element's textual content.
- `before` - Retrieves the before **Generated Content** (p. 52) of an element.
- `after` - Retrieves the after **Generated Content** (p. 52) of an element.

The following example shows a cross-reference that references a heading and shows its before Generated Content and text:

Example: A Cross-reference that references a heading and shows the heading's before Generated Content and text:

```
a[href]{
  content: target-text(ro-attr(href url), before) " "
          target-text(ro-attr(href url), content);
}
```

Note:

`target-text()` makes it easy to retrieve the before Generated Content of an element, which may include its numbering. This method does not require any knowledge about how this before Generated Content is created but it also does not allow to rebuild it into something different.

If the before Generated Content of an element is "2.1" and the page header should be "Chapter 2, Section 1" the `target-counter()` (p. 60) function provides the necessary means to retrieve all the **Counters** (p. 53) individually.

5.5.4 Footnotes

A footnote is a text note placed on the bottom of a page. It references a specific part of the main content of the document, giving further explanations or information about a citation. A footnote is marked by a defined symbol both in the main content of the page and in the footnote area at the bottom of the page, to show which parts belong together.

For content that is required to have a footnote, the following style can be applied:

```
float: footnote
```

The text content of the element that the style applied to, will appear in the footnote area at the bottom of the page. Content in the footnote area can be styled via CSS using the `footnote` rule.

Example: Defining a footnote for an element and styling the footnote area.

```
.footnote {
  float: footnote;
}
@page {
  @footnote {
    border-top: solid black 1px;
  }
}
```

By defining a footnote, a footnote call is left behind in the main content. Its content and style can be influenced by the `footnote-call` pseudo-element.

For every footnote element, there is also a `footnote-marker` pseudo-element added. Usually this contains the same number or symbol as the footnote-call it belongs to.

Example: Styling the footnote-call and footnote-marker:

```
.footnote::footnote-call {
  content: counter(footnote, decimal)
}
.footnote::footnote-marker {
  content: counter(footnote, decimal);
}
```

By default, the footnote counter is available and is automatically incremented for every element with the style:

```
float: footnote
```

By default, this counter numbers the footnotes sequentially for the entire document. To number footnotes on a per-page basis, the counter has to be reset on every page, using the following style:

```
@page {
  counter-reset: footnote;
}
```

Note:

PDFReactor currently does not support Footnotes inside [Multi-column layouts \(p. 63\)](#).

5.6 Transforms

5.6.1 2D Transforms

PDFReactor is capable of transforming elements with the `transform` ([p. 160](#)) property, which makes moving, rotating and scaling document content possible.

Note:

2D Transforms do not have an impact on the document layout, e.g. content with scaled up size will not push other content away to prevent overlapping.

Reduce Table Width with Rotated Table Headers

[awesomizr.js \(p. 32\)](#) is able to automatically reduce the width of table headers with 2D transforms.

The `rotateTableHeaders()` function transforms and rotates a table header, in order to reduce its width. If there is no table header, the first line is converted to one.

This function takes two parameters:

- `table`: The HTML node of the table
- `params`: An object of *optional* parameters

Options

Key	Description	Default
angle	The angle in degrees at which the header will be rotated. Should be between -90 and 90	45

Key	Description	Default
width	The width that the header cells should have after the transformation, e.g. "20pt".	"auto"
firstCol	Whether to prevent the first column from being transformed.	false
lastCol	Whether to prevent the last column from being transformed.	false
footer	Whether to automatically create a <code><tfoot></code> element from the last row in the table. Has no effect if the table already contains a <code><tfoot></code> .	false

5.7 Multi-column Layout

The content of a document can be arranged in columns with elements like images or titles spanning through all columns if desired. Elements are laid out in a way similar to pages, text and boxes will break whenever no space is left in a column.

Multi-column layout is often used in print products like newspapers or magazines, it is intended to reduce the line width to make text easier to read.

The following box shows how text flows in a three-column layout. The paragraphs are numbered to better visualize the effect of multi-column layout.

[1] Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla in libero turpis. Sed sed dolor diam, eu dapibus quam. Quisque ut nulla purus, iaculis sollicitudin erat. Nullam dictum suscipit porttitor.	[2] Aliquam aliquam elementum elementum. Donec vel odio nec diam ullamcorper ultricies vel sit amet elit. Cras non aliquet lectus. [3] Donec sollicitudin lorem placerat est condimentum rutrum. Fusce tempor cursus	rutrum. Duis mattis mattis sapien. Phasellus tempus iaculis tellus sed vestibulum. [4] Etiam faucibus consectetur augue, sit amet interdum elit dapibus at.
---	---	---

To create a multi-column layout inside an element add either the property `column-count` (p. 116) or `column-width` (p. 121) or both. By adding them the element becomes a multi-column element.

The `column-count` property defines the number of columns inside the element. Any number greater than 1 will create a multi-column layout. The `column-count` property is especially useful if the actual width of the columns is not as important as the number of columns.

The `column-width` property is used to control how wide columns inside the element should be. The number of columns is computed from that value. Therefore the actual width of the columns may be wider or narrower than the specified width. This property is useful if the general width of the columns is more important than the number of columns.

If both properties are used the resulting layout tries to honor both values. `column-count` will provide the maximum number of columns in most cases.

```
/* define two columns */
div.twoColumns{ column-count: 2 }

/* define columns with a width of 2in */
div.twoInchColumns { column-width: 2in }
```

Note:

PDFreactor currently does not support **Footnotes** (p. 61) inside Multi-column layouts.

By default, PDFReactor aims to balance the content of columns so that the content of all individual columns is equally long, if possible. This has the effect of keeping the height of each column at the possible minimum, which automatically determines the height of the multi-column element as a whole if it wasn't defined by a height property or attribute.

This behavior can also be modified to fill columns sequentially. In this case, the columns are filled until no more space is available in one column and the rest of the content needs to be moved to the next column. With this behavior a multi-column element whose height is not restricted will take up all the remaining space inside the multi-column-element, up to the remaining space available on the page until it breaks to another column.

The filling behavior can be controlled with the `column-fill` (p. 117) property:

```
/* sequential filling behavior */
div.sequentialFill{ column-fill: auto }

/* balanced filling behavior */
div.balancedFill{ column-fill: balance }
```

A defined height on the multi-column element will be used for an element, regardless of the filling behavior. If there is less content than there is space inside the multi-column-element a balanced filling behavior will create smaller columns, leaving space at the bottom of the multi-column element. Sequential filling behavior may not have enough content to fill all the columns. If there is more content than there is space inside the multi-column element, the multi-column element will create a page break and continue on the next page, at the first column.

Usually elements inside a multi-column element are laid out one after another in columns automatically defined by the filling behavior. Some elements however may require a certain behavior when inside columns.

There are elements that are required to span all columns inside the multi-column element instead of only one. Headings, pictures or tables are the most common examples. To have an element span all columns the `column-span` (p. 120) property is used.

```
/* a heading that spans all columns */
h1{ column-span: all }

/* a table in a single column */
table{ column-span: none }
```

To add some visual appeal to the multi-column element borders, backgrounds and padding can be used. Beside these standard styles multi-column elements can also receive additional styles for the space between columns.

To visually separate columns it is possible to define the gap width. Gaps can be considered as padding between columns. To define the gap width for a multi-column element the `column-gap` (p. 118) property is used.

```
/* a gap of 0.25in */
div.multiColumn{ column-gap: 0.25in }
```


In addition to the gap a rule can be added between the columns as additional visual aid for separating columns. To define rules for a multi-column element the property either the `column-rule` (p. 118) shorthand or the individual properties `column-rule-width` (p. 120), `column-rule-style` (p. 119) or `column-rule-color` (p. 119) can be used.

```
/* a solid black rule with 0.1in width*/
div.multiColumn{
  column-rule-width: 0.1in;
  column-rule-style: solid;
  column-rule-color: black
}

/* the same definition as shorthand */
div.multiColumn{ column-rule: 0.1in solid black }
```

Note

A Multi-column layout with justified text looks best when the text is laid out with [Automatic Hyphenation](#) (p. 51) enabled.

5.8 Region Layout

Regions are containers for document content similar to pages or [columns](#) (p. 63), but they can be positioned individually. In contrast to automatically created pages and columns, regions are based on block elements from the document, which presents them with more styling options.

Regions belong to a region chain, that connects them and tells how their contents flows from one to another. The content of a region chain is called the named flow and elements can be added to a named flow to be displayed in regions.

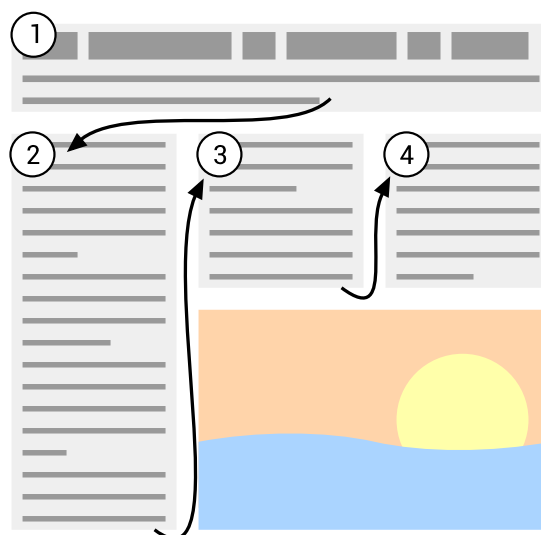


Fig. 4: A named flow flows through a region chain.

5.8.1 Adding Regions to Region Chains

Most block elements can be defined as a region. They are not required to be of the same size nor are they required to be the same node name.

To create a region from a block element, the `-ro-flow-from` (p. 131) property is used. It receives an identifier. A region chain contains all regions of the same identifier in document order. The identifier is also the name of the named flow these regions will display.

Note

A region element will not have its subtree rendered. It either displays content from a named flow or nothing.

Example: A chain of two regions defined for two HTML `div` elements with IDs `"region1"` and `"region2"`.

```
#region1, #region2{
    -ro-flow-from: regionChainName;
}
```

PDFreactor automatically lays out content inside regions and breaks text and boxes where no space is left. The number of regions inside a region chain is limited by the number of associated Region elements though and it is possible that the content of a named flow occupies more space than is available inside the regions of a region chain. In that case content from the named flow overflows the last region inside the region chain.

Note

A region does not influence the style of the content it contains. No style is inherited from a region into the displayed named flow and style that would influence the content of an element has no effect on a region's content.

5.8.2 Adding Content to a Named Flow

The `-ro-flow-into` (p. 131) property adds document content to a named flow. The content may consist of content from one or more elements. Content assigned to a named flow is not rendered at its position inside the document but inside one of the regions inside the region chain.

The property receives an identifier which is the name of the named flow the content belongs to. An optional keyword defines what part of the styled element should be taken into the named flow:

- `element`
 - Adds the entire element to the named flow.
 - If no keyword is given, this is the default behavior.
- `content`
 - Adds the element's content to the named flow.

Example: Creation of a named flow for two HTML `<article>` elements while an HTML `<section>` element from one of the articles is moved to a different named flow.

HTML:

```
<article>...</article>
<article>
  ...
  <section id="info">...</section>
</article>
```

CSS:

```
article{
  -ro-flow-into: articleNamedFlowName;
}
section#info{
  -ro-flow-into: infoNamedFlowName;
}
```

Note

The content of a named flow may be rendered inside regions, but it still inherits style and computes its style the same way it would as if it did not appear inside a region.

5.8.3 Region Generated Content

A region element can have before and after [Generated Content](#) (p. 52) just like any other element. This generated content is rendered above or below the region's content and is not moved to the next region due to lack of space. Instead the available space inside a region is reduced. If there is still not enough space left, the region's content flows over.

5.9 Breaking Boxes

Although PDFReactor performs automatic breaks between boxes for [pages](#) (p. 49), [columns](#) (p. 63) and [regions](#) (p. 65), it is often necessary to add explicit breaks in certain situations or breaks should be avoided to keep content together where it belongs together. This chapter explains how both can be achieved.

Note

PDFReactor provides style for HTML that influences the break behavior for certain elements like headings and lists. Break Styles for XML documents need to be created based on the document language.

5.9.1 Breaking Around Boxes

To manipulate the break behavior before and after boxes, the [break-before](#) (p. 112) and [break-after](#) (p. 112) properties are used. They provide keywords to force or avoid page, column and region breaks.

Example: A manual page break before an HTML `<h1>` element, used to make a chapter start on top of a new page.

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

Example: A manual page break before an HTML `<h1>` element, that makes the chapter start on a right page.

```
h1{
  break-before: right;
}
```

This style creates a page break before the `h1` and moves it to the next page. In case this is a left page another page break is performed, to move it to a right page again.

Example: Avoiding breaks after HTML heading elements.

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

Note

PDFReactor also supports the CSS 2.1 properties `page-break-before` and `page-break-after`. They are resolved as shorthands for `break-before` and `break-after`.

5.9.2 Avoid Breaking Inside Boxes

To manipulate the break behavior inside a box, the property `break-inside` (p. 113) is used. It specifies whether breaking should be avoided inside the box or not.

Example: Avoid breaks inside an HTML `<div>` element.

```
div{
  break-inside: avoid;
}
```

Note

PDFReactor also accepts the CSS 2.1 property `page-break-inside` and resolves it as shorthand for `break-inside`.

5.9.3 Adaptive Page Breaks

`awesomizr.js` (p. 32) is able to automatically add page breaks depending on the amount of space left below an element with the help of the `applyAdaptivePageBreaks()` function.

A possible use case is to prevent a new section from beginning at the bottom of a page.

The function also prevents large whitespaces that occur when in situations where only a couple of sentences from a previous section are followed by a page break as the next section begins.

The function takes two parameters:

- `selector`: (optional) The CSS selector for the elements that may require a new page break. *Default value:* `"h1, h2"`
- `threshold`: (optional) If an element is below this percentage of the page height, a page break is inserted. *Default value:* 67

5.10 Advanced Page Layout

PDFReactor provides additional means for professional printing that allow to specify oversized pages, a bleed area and marks for cutting sheets to the final page size and color proofing.

5.10.1 PDF Page Boxes

Page boxes are used to specify the page geometry. PDFReactor supports the TrimBox, MediaBox, BleedBox, CropBox and ArtBox.

TrimBox

The TrimBox defines the size of the final print result, the final page. It contains the page content.

The size of the TrimBox is defined equivalent to the page size, as mentioned in chapter [Page Size & Orientation](#) (p. 50), using the `size` (p. 155) property.

Example: The value of the `size` property also automatically specifies the TrimBox.

```
size: A4 portrait;
```

MediaBox

In prepress, a printed document can contain more information than just the actual content in the TrimBox (e.g. bleed or [Printer Marks](#) (p. 70)).

As this information does not belong to the print result and instead needs to be printed around it, a print sheet larger than the print result is needed. The MediaBox defines the size of the print sheet.

Special oversize formats are used as print sheet in such cases. For DIN ³⁶ standard-based formats, the matching oversize formats to the A series are the DIN-RA and DIN-SRA formats. An overview of all supported page sizes can be found in the Appendix [Supported Page Size Formats](#) (p. 90)

The property `-ro-media-size` (p. 143) is used to specify the media size.

Example: The document should be printed in DIN-SRA4 and the MediaBox is set to this size.

```
-ro-media-size: SRA4;
```

The MediaBox is the largest of all 5 page boxes and contains all others which can be smaller or equal than this box.

BleedBox

The BleedBox contains the TrimBox and is slightly larger. Content from the TrimBox may "bleed" into the BleedBox where it is still painted.

This is necessary for content that should reach to the edge of the print result. The "bleeded" content prevents having unprinted areas due to unprecise cutting out the print result from the oversized print sheet.

The size of the BleedBox is defined as a width that adds up on the TrimBox' size. Common bleed values are 3-5 mm (Europe) or 1/8 inch (USA/UK).

Setting the bleed size can be achieved by using the property `-ro-bleed-width` (p. 107).

Example: A bleed width of 3mm around the print result. The Bleed Box determines it's size from the TrimBox and this width.

```
-ro-bleed-width: 3mm;
```

³⁶ Deutsches Institut für Normung, in English: German Institute for Standardization, Germany's ISO member body.

CropBox

The CropBox defines the complete area of the document that should be displayed on screen or printed out.

The crop size can be defined using the property `-ro-crop-size` (p. 127).

The crop size can be set to a specific page size format (like setting the trim size) or to one of the page boxes. It is not set by default.

Example: The CropBox is set to match the MediaBox.

```
-ro-crop-size: media;
```

ArtBox

The ArtBox is used to define a specific area inside which the page's content is located.

Using the property `-ro-art-size` (p. 102), the ArtBox can be set to a specific page size or one of the page boxes. It is not set by default.

Note:

When generating a PDF/A conformant file (see [PDF/A conformance](#) (p. 39)), the ArtBox is required not to be defined.

5.10.2 Printer Marks

Printer Marks are special pieces of information located outside of the actual print result. They are used to prove the correctness of the result in prepress printing and are placed outside the [TrimBox](#) (p. 69).

Cutting out the print result of the print sheet is done inside the bleed area. Trim and bleed marks indicate where this area starts and ends. Both types of marks are displayed as hairlines in the corner of the print sheet.

Registration marks show whether the printer's colors are aligned properly. They are printed as crosshair-shaped objects located on each side of the print sheet.

Color bars show if the colors of the print result meet the expected result. They consist of a variety of colors that can be checked individually.

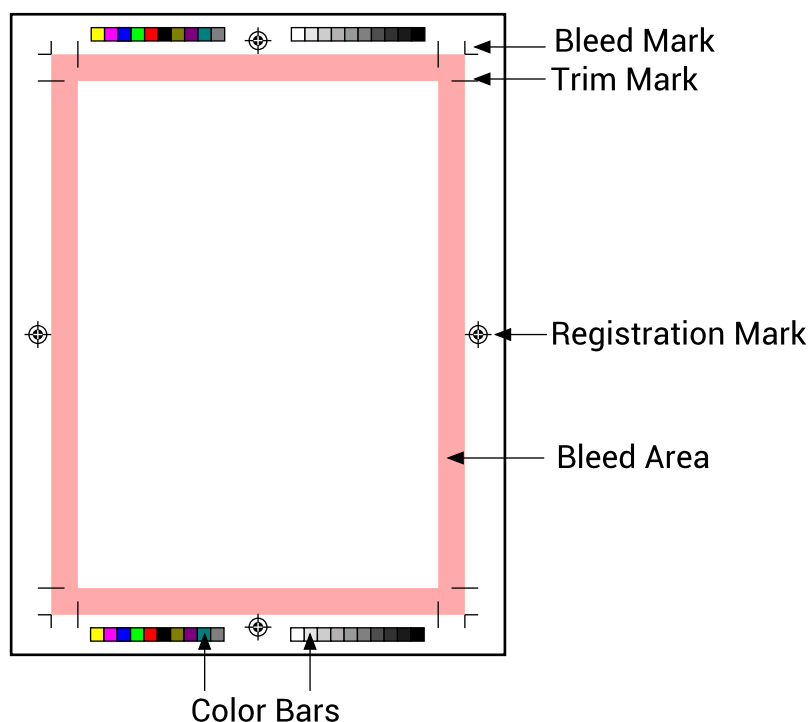


Fig. 5: Printer Marks

The property `-ro-marks` (p. 141) is used to add trim, bleed and registration marks. The property `-ro-marks-width` (p. 141) sets the width of the mark lines, `-ro-marks-color` (p. 141) sets their color.

Example: Setting printer marks

```
-ro-marks: trim bleed registration;
-ro-marks-width: 1pt;
-ro-marks-color: red;
```

Setting one of the `-ro-colorbar-*` (p. 116) properties defines where a color bar is added to the document.

Example: Setting color bars at the bottom left and right.

```
-ro-colorbar-bottom-left: gradient-tint;
-ro-colorbar-bottom-right: progressive-color;
```

5.11 Leaders

Leaders are often used to draw a visual connection between an entry in a table of contents or similar structures, and a corresponding value.

In CSS, drawing leaders is accomplished via the use of the `leader()` function. This function accepts the following values:

- dotted
- solid
- space
- <string>

A leader may be added using the content property, and can be freely combined with other generated content such as counters.

Example: Adding leaders to the entries in a table of contents.

```
a.toc_ah2::after{
    content: leader(dotted) " " target-counter(ro-attr(href url), page);
}
```

This may result in a display such as:

Table of Contents	
1. Introduction.....	2
2. The Cyclone.....	2
3. The Council with the Munchkins.....	4
4. How Dorothy Saved the Scarecrow.....	7

5.12 Table of Contents

A table of contents can be automatically inserted into a document to generate a list of the chapters or other important sections in the document.

This feature is usually used together with cross-references to add links to a table of contents. With the addition of counters, it can be complemented with the page numbers of the linked chapters.

The `createTableOfContents()` function provided by [awesomizr.js](#) (p. 32) allows to insert a table of contents that is generated from given elements.

Note:

The table of contents requires certain styles to work properly. These styles are included in the `awesomizr.css` and should be added either to the document or by using the `addUserStyleSheet()` method of the PDFreactor API.

The table of contents is inserted as an HTML `div` element with the class `ro-toc`. Inside this `div` can be two headings (document title and a heading for the table of contents with the class `ro-toc-heading`) and the `div` elements with links to the pages and a class depending on the level of the referenced element (`ro-toc-heading1`, `ro-toc-heading2`, ...)

The level of a TOC entry is determined by the position of its selector in the `elements` array.

```
Awesomizr.createTableOfContents({elements: ["h1", "h2", "h3"]});
```

The function's optional parameter is an object with several options:

Values of the option object

Key	Type	Description	Default
<code>insertiontarget</code>	string	CSS selector string of the element where the table of contents should be inserted.	"body"

Key	Type	Description	Default
insertiontype	string	Specifies where exactly the table of contents should be inserted: <ul style="list-style-type: none"> • "beforebegin": Before the element • "afterbegin": As new first-child • "beforeend": As new last-child • "afterend": After the element 	"afterbegin"
elements	array	An array of the CSS selector strings of elements that should be added to the table of contents. Each TOC entry gets a class name based on the index of the corresponding selector in this array, e.g. by default the h2 entries have the class ro-toc-level-2.	["h1", "h2"]
toctitle	string	The title of the table of contents. If an empty string is set, no title is inserted.	"Table of Contents"
disableddocumenttitle	boolean	Whether the document title should NOT be inserted before the table of contents.	false
text	function	By default, the text for the entries of the TOC is the text content of the element matching the specified selector. Alternatively, you can specify a function, the return value of which will be used as text for the respective entry. The element representing the entry is passed as an argument to the function. Returning false will skip the entry entirely and not include it in the TOC.	null

Example: Simple table of contents created with Awesomizr based on HTML <h2> elements

```
<link href="css/awesomizr.css" type="text/css" rel="stylesheet" />
<script type="text/javascript" src="awesomizr.js"></script>
...
<body onload="Awesomizr.createTableOfContents({elements: ['h2']});">
```

Example: List of figures with custom text content for the entries

```
Awesomizr.createTableOfContents({
  elements: ['img'],
  text: function(elem) {
    // the entry text should be the image's alt text
    var txt = elem.alt;

    if (txt) {
      return txt;
    }

    // skip images without alt text
    return false;
  }
});
```

Note:

Alternatively, a table of contents can also be created by using XSLT. The two samples for table of contents demonstrate both approaches.

5.13 Shrink-to-Fit

For some documents parts of the content are too wide to fit the pages. In most cases this is caused by HTML documents containing fixed widths intended for screens, e.g. 1024px for the main container element.

While the best solution is adding a print style sheet to override the critical styles with relative widths, such content can also be shrunk automatically without changing the source document or adding specific styles.

There are two different shrink-to-fit functionalities available in PDFreactor, `setPixelsPerInchShrinkToFit` and `-ro-scale-content` (p. 155). These are non-exclusive and are applied in the aforementioned order.

5.13.1 The method `setPixelsPerInchShrinkToFit`

This method adapts the "pixels per inch" value used for laying out the document, i.e. it only scales lengths set as px including such set via HTML attributes. It does not cause gaps at the bottom of pages.

Example: Shrink-to-fit using the `setPixelsPerInchShrinkToFit` API method

```
pdfReactor.setPixelsPerInchShrinkToFit(true)
```

The `pixels per inch` (p. 76) can also be specified manually.

5.13.2 The property `-ro-scale-content`

This property must be part of the `@page` rule and allows the following values:

- A percental value which is treated as a scaling factor for the content.
- The value `none` causes no scaling.
- The value `auto` enables the automatic scaling of the content to fit the size of the page.

Example: Shrink-to-fit using the `-ro-scale-content` CSS property

```
@page {
  -ro-scale-content: auto;
}
```

Note:

This functionality scales down entire pages, which can cause gaps at the bottom of pages.

5.14 Page Order

Usually, the page order of a PDF is only determined by its input document. However, using the API method `"setPageOrder"`, the page order can be set by providing a string parameter.

For ease of use the following constants are available for the most common cases of page orders:

- `PAGE_ORDER_REVERSE` — The page order is reversed.
- `PAGE_ORDER_EVEN` — All even pages are moved before all odd pages.
- `PAGE_ORDER_ODD` — All even pages are moved before all even pages.
- `PAGE_ORDER_BOOKLET` — All pages are ordered as in a booklet.
- `PAGE_ORDER_BOOKLET_RTL` — All pages are in right-to-left booklet order.

Instead of using a predefined order the parameter can also provide a custom order as comma-separated list of page numbers and ranges:

- "x,y,z" — New page order x, y, z
- "x-y" — All consecutive pages from x to y
- "x*n" — The page x is repeated n times

Example: Setting the page order

```
pdfReactor.setPageOrder("2,5,6*2,8-10");
```

The page order shown above results in a PDF having the following page numbers from the original document: 2, 5, 6, 6, 8, 9, 10

5.15 Pages Per Sheet

Instead of containing only one page of the input document per PDF page, multiple pages of the input document can be displayed on one sheet.

The pages will be arranged in a grid on the sheet. The number of columns and rows of this grid are user-defined.

To utilize Pages Per Sheet use the API method `setPagesPerSheetProperties`:

```
pdfReactor.setPagesPerSheetProperties(int cols, int rows, String sheetSize,
    String sheetMargin, String spacing, int direction);
```

The parameters `rows` and `cols` define the corresponding number of pages that get laid out on a single page. Their values are required. The values for `sheetSize`, `sheetMargin` and `spacing` can be set as CSS width values. `direction` defines in which way the single pages are ordered.

There are the following options to set a direction:

- `PAGES_PER_SHEET_DIRECTION_RIGHT_DOWN` — The single pages are ordered from left to right and top to bottom. This is the default value.
- `PAGES_PER_SHEET_DIRECTION_RIGHT_UP` — The single pages are ordered from left to right and bottom to top.
- `PAGES_PER_SHEET_DIRECTION_LEFT_DOWN` — The single pages are ordered from right to left and top to bottom.
- `PAGES_PER_SHEET_DIRECTION_LEFT_UP` — The single pages are ordered from left to right and bottom to top.
- `PAGES_PER_SHEET_DIRECTION_UP_RIGHT` — The single pages are ordered from bottom to top and left to right.
- `PAGES_PER_SHEET_DIRECTION_UP_LEFT` — The single pages are ordered from bottom to top and right to left.
- `PAGES_PER_SHEET_DIRECTION_DOWN_RIGHT` — The single pages are ordered from top to bottom and left to right.

- `PAGES_PER_SHEET_DIRECTION_DOWN_LEFT` — The single pages are ordered from top to bottom and right to left.

Example: Arranging 4 pages per sheet

```
pdfReactor.setPagesPerSheetProperties(2, 2, "A4 landscape",
    "2.5cm", "2cm", PDFReactor.PAGES_PER_SHEET_DIRECTION_RIGHT_UP);
```

5.16 Booklet

A Booklet is a set of folded pages meant to be read like a book. PDFReactor supports creating Booklets by combining the [Pages Per Sheet \(p. 75\)](#) functionality with the [Page Order \(p. 74\)](#) feature.

It orders the pages in booklet or rtl booklet page order and places two of these pages on each sheet, rotated by 90 degrees and side-to-side.

An API method allows to configure the page size and margins of the container page as well as to use the default booklet page order or a reversed order:

```
pdfReactor.setBookletMode("A4 landscape", "1cm", PDFReactor.PAGE_ORDER_BOOKLET_DEFAULT);
```

5.17 Pixels per Inch

By default, lengths specified in pixels (i.e. via the CSS unit `px` or HTML attributes) are converted to physical lengths at a rate of 96 pixels per inch. With the method `setPixelsPerInch` this can be changed, e.g.:

```
pdfReactor.setPixelsPerInch(120)
```

Increasing the pixels per inch can be used to shrink documents that would be too wide for pages due to fixed widths originally intended for screens.

Finding the optimum value can be automated using [shrink to fit \(p. 74\)](#).

6. PDFREACTOR COOKBOOK

This chapter will guide you through some of the topics that will most frequently arise when using PDFReactor, and will give you hands-on advice in each case.

6.1 How Do I Create Running Table Headers?

If a page break occurs in a table with running table headers, the table headers are repeated for each page the table runs over. To ensure that the table headers are repeated, all you have to do is using the corresponding page markup.

Example:

```
<table>
  <thead>
    <tr>
      <td>A Simple Heading</td>
    </tr>
  </thead>
  <tr>
    <td>Row 1</td>
  </tr>
  <tr>
    <td>Row 2</td>
  </tr>
</table>
```

6.2 How Do I Set CSS & XSLT Stylesheets?

You can set CSS style sheets either by referencing them in your document, setting or adding them using an API method, or inline in your document.

Defining a CSS style sheet in the "style" Section of the Document:

```
<head><style type="text/css">p { color: red }</style></head>
```

Referencing an external CSS style sheet using the <link> Element:

```
<link href="http://someServer/css/layout.css"
      rel="stylesheet" type="text/css">...</link>
```

Defining CSS Styles Inline:

```
<table style="color: red">...</table>
```

Adding a CSS style sheet Using an API Method:

```
Java:
  pdfReactor.addUserStyleSheet("", "", "", "http://server/layout.css")
PHP:
  pdfReactor->addUserStyleSheet("", "", "", "http://server/layout.css")
.NET:
  pdfReactor.AddUserStyleSheet("", "", "", "http://server/layout.css")
CLI:
  -s "http://some/layout.css"
```

```

Java: pdfReactor.addUserStyleSheet("p { color: red }", "", "", "")
PHP: pdfReactor->addUserStyleSheet("p { color: red }", "", "", "")
.NET: pdfReactor.AddUserStyleSheet("p { color: red }", "", "", "")
CLI: -u "p { color: red }"

```

XSLT style sheets can be set either using an API method, or by referencing them in the document. They can not be specified directly inline as CSS style sheets can be.

Note:

XSLT style sheets are applied in a pre-processing step, before the document is layed out and CSS or JavaScript is processed.

Adding an XSLT style sheet using an API method:

```

CLI: java -jar pdfreactor.jar -a links bookmarks -s
      file:///C:/style.css -X file:///C:/xsl-style.xsl sample.html
      C:/sample.pdf
PHP: $pdfReactor->addXSLTStyleSheet(null, "style.xsl");
Java: pdfReactor.addXSLTStyleSheet(null, "style.xsl");
.Net: pdfReactor.AddXSLTStyleSheet(null, "style.xsl");

```

Referencing an external XSLT style sheet via the <link> element:

```
<link href="wizard0f0z.css" type="text/css" rel="stylesheet"/>
```

6.3 How Do I Handle Relative References?

For documents including relative resources, like

```
...</img>
```

```
<a href="/english/index.html">...</a>
```

```
<link href="../css/layout.css"
      rel="stylesheet" type="text/css">...</link>
```

PDFreactor needs a base URL³⁷ to resolve these resources. If your source is a URL the base URL will be set automatically. In all other cases you have to set it yourself:

```

Java: pdfReactor.setBaseURL("http://someServer/public/")
PHP: pdfReactor->setBaseURL("http://someServer/public/")
.NET: pdfReactor.SetBaseURL("http://someServer/public/")
CLI: -b "http://someServer/public/"

```

It is also possible to specify file URLs:

```

Java: pdfReactor.setBaseURL("file:///directory/")
PHP: pdfReactor->setBaseURL("file:///directory/")
.NET: pdfReactor.SetBaseURL("file:///directory/")
CLI: -b "file://someServer/file"

```

³⁷ Uniform Resource Locator (<http://www.w3.org/Addressing/>)

6.4 How Do I Set Styles for Print or Screen Only?

All styles inside this block will only affect print media:

```
@media print{...}
```

All styles inside this block will only affect screen media:

```
@media screen{...}
```

6.5 Automatic Resizing of Form Controls

When HTML form controls such as input fields and textareas are rendered on screen, they usually have a fixed size determined by their attributes or by the browser. If the content of the form control is larger than the form control itself, the browser usually adds scroll bars to the control or allows navigation using a caret.

This, of course, is not possible on print or in a paged environment. To overcome this, PDFReactor supports some style properties which allow the automatic resizing of form controls according to their content. If these properties are set, the form controls' size automatically adjusts according to its content.

These properties are: `-ro-width` (p. 166) and `-ro-height` (p. 134).

`-ro-width` automatically adjusts the width of a form control if the contents' width exceeds the width defined for the form control.

`-ro-height` automatically adjusts the height of a form control if the contents' height exceeds the height defined for the form control.

Example usage of these properties:

```
input[type="text"] {
    -ro-width: auto;
}
textarea {
    -ro-height: auto;
}
```

6.6 How Do I Set Colors in CSS?

How do I set RGB colors?

In CSS you can specify RGB³⁸ colors in the following ways:

- # followed by a 6 digit RGB value in hexadecimal notation, e.g. `#00ff00` for perfect green.

You can abbreviate this notation by using only 3 digits which will be expanded internally, e.g. `#0f5` equals `#00ff55`.

- Using the function `rgb`. It takes the 3 RGB component values as parameters in decimal or procentual notation, e.g. `rgb(0,255,0)` or `rgb(0%,100%,0%)` for perfect green.

³⁸ **Red Green Blue**, additive color model, consisting of the color components red, blue and green.

How do I set RGBA colors?

RGBA³⁹ colors are also supported and can be specified by using the function `rgba`. It takes the 3 RGB component values as well as 1 alpha component value as parameters in decimal or procentual notation, e.g. `rgba(0,0,255,0.5)` or `rgba(0%,100%,0%,50%)` for semi-translucent blue.

While it is currently possible to set RGBA colors on any CSS border, complex border settings (e.g. table cells borders) or border styles other than "solid" are not yet supported and may cause unexpected visual outcome.

How do I set CMYK colors?

Besides `rgb` and `rgba` PDFreactor also supports the non-standard function `cmyk`. It takes the 4 CMYK component values as parameters in decimal or percent notation, e.g. `cmyk(0,0,1,0)` or `cmyk(0%,0%,100%,0%)` for perfect yellow.

Color key words can be converted automatically into CMYK using the `setDefaultColorSpace` API method:

```
pdfReactor.setDefaultColorSpace(PDFreactor.COLOR_SPACE_CMYK);
```

CMYK colors are also supported in [SVGs \(p. 24\)](#).

How do I set HSL colors?

HSL⁴⁰ is another representation of the RGB colorspace. The hue value is in the range of 0 to 360, the saturation and lightness values range between 0 and 1. It is possible to set HSL colors using the function `hsl`. It takes the 3 HSL component values as parameters in decimal or percent notation, e.g. `hsl(240,0,0)` or `hsl(66%,0%,0%)` for blue.

How do I use color key words?

Instead of using color functions or the hexadecimal notation a single human readable key word can be used. For more information which key words are supported by PDFreactor see the [CSS Color Keywords table \(p. 92\)](#). The key words are internally converted into the user-set color space. By default, they are converted into RGB colors.

6.7 How Do I Resize Background Images?

You can use the `background-size` property to resize background images:

```
background-size: 100px 50px /* set size to 100 x 50 pixels */
background-size: 100% 100% /* set size to 100% of the size
                             of the containing element */
background-size: 50% /* set width to 50% of the width of
                      the containing element and keeps
                      the aspect ratio of the image */
background-size: auto 80px /* set height to 80 pixels and keeps
                           the aspect ratio of the image */
```

³⁹ Red Green Blue Alpha, a color model similar to RGB, with extra information about the translucency.

⁴⁰ Hue Saturation Lightness, alternative representation of colors of the RGB color model.

6.8 How Do I Create Rounded Corners?

To create rounded corners for borders, you can use the property `border-radius` (p. 109), e.g.:

```
border-radius: 0.2cm;
```

6.9 How Do I Place an Image in the Header?

Adding images as generated content is explained in [Generated Images](#) (p. 53). This also works for the content of [Page Margin Boxes](#) (p. 53).

Example: An image as content of a Page Margin Box.

```
@page{
  @top-left{
    content: url(http://mydomain/pictures/image.svg)
  }
}
```

6.10 How Do I Use HTML in Headers and Footers?

There are two options to add HTML to header and footer boxes, either [Running Elements](#) (p. 54) or [Running Documents](#) (p. 57).

6.11 How Do I Create a Document With a Text Direction of Right-to-Left?

PDFReactor automatically analyzes the document to handle both left-to-right and right-to-left text correctly.

The base direction of the document defaults to left-to-right. You can set it to right-to-left by specifying the `dir` attribute of the root element as in the following example:

```
<html dir="rtl">
```

You can also override the base direction specifically for certain elements via the property `direction` (p. 129):

```
div.english {
  direction: rtl;
}
```

You can override the automatically selected text direction by combining `direction` with the property `unicode-bidi` (p. 163):

```
span.forcertl {
  unicode-bidi: bidi-override;
  direction: ltr;
}
```

6.12 How Do I Save Memory if a Document Refers to Many or Very Large Image Files?

To reduce the memory consumption caused by converting documents referencing many or large images, set the processing preference `PROCESSING_PREFERENCES_SAVE_MEMORY_IMAGES`:

```
pdfReactor.setProcessingPreferences(
    PDFreactor.PROCESSING_PREFERENCES_SAVE_MEMORY_IMAGES);
```

This setting will have an impact on the performance and should therefore only be used when necessary.

6.13 How can I determine programatically that content fits into its pages

Content that does not fit into its pages can be logged as well as programatically analyzed. This functionality is enabled and configured by the method `setLogExceedingContent`, which takes two arguments:

The first one specifies what to analyze:

Constant	Description
NONE	Disable this functionality (default)
CONTENT	analyze content (text and images) only
CONTENT_AND_BOXES	analyze content as well as boxes. (catches exceeding borders and backgrounds)
CONTENT_AND_STATIC_BOXES	analyze content as well as boxes, except for those with absolute or relative positioning

The second one specifies how to analyze:

Constant	Description
NONE	Disable this functionality (default)
PAGE_BORDERS	Find content exceeding the actual edges of the page
PAGE_CONTENT	Find content exceeding the page content area. (avoids content extending into the page margins)
PARENT	Find content exceeding its parent (i.e. any visible overflow)

For example:

```
pdfReactor.setLogExceedingContent(
    PDFreactor.EXCEEDING_CONTENT_ANALYZE_CONTENT_AND_STATIC_BOXES,
    PDFreactor.EXCEEDING_CONTENT_AGAINST_PAGE_CONTENT);
```

To programatically process the results you can get an array of `ExceedingContent` objects using the method `getExceedingContents`. Please see the API documentation for details on this class.

6.14 How Can I Retrieve the Number of Pages of a Converted Document Programmatically?

After converting a document you can use the method `getNumberOfPages` of the same instance of `PDFReactor` to retrieve the number of pages of the resulting PDF.

```
pdfReactor.renderDocumentFromContent(content);
int numberOfPages = pdfReactor.getNumberOfPages(true);
```

The boolean parameter specifies whether to return the number of pages of the final PDF (`true`) or of the layed out input document without any postprocessing (`false`)

6.15 How Do I Access Resources That Are Secured Via Basic or Digest Authentication?

Documents or other resources that are secured via Basic or Digest authentication can be accessed by setting authentication credentials for `PDFReactor` using the `setAuthenticationCredentials` API method:

```
pdfReactor.setAuthenticationCredentials("user", "password");
```

The credentials are set for all outgoing HTTP connections.

6.16 How Can I Set Request Headers And Cookies For The Outgoing Connections of PDFReactor?

Using the method `setRequestHeader`, you can set request headers for all outgoing HTTP connections of `PDFReactor`, used to load the document and its resources like images and style sheets. Similarly you can set cookies using the method `setCookie`.

Both expect a key-value-pair as parameters and can be called multiple times to set multiple headers or cookies. Existing keys will be overwritten.

```
pdfReactor.setRequestHeader("User-Agent", "PDFReactor");
pdfReactor.setCookie("name", "Peter");
```

This functionality can be used to pass a session ID from the integration to `PDFReactor`.

6.17 How Can I Add a Smooth Color Transition to the Background of an Element?

A color transition of two and more colors can be added to elements using CSS gradients. CSS gradients are dynamically generated images that can be used as background and list style images. Following an example that generates a background with a fine blue gradient.

```
background-image: linear-gradient(skyblue, cornflowerblue);
```

6.18 How Can I Rotate Text by 90 Degrees?

Text can be rotated and transformed via [Transforms \(p. 62\)](#).

Example: An element with the `label` class is rotated by -90 degrees and moved to the left side.

```
.label {  
  transform-origin: 0px;  
  transform: rotate(-90deg) translateY(-100%);  
}
```

APPENDIX A: FONTS

To be able to display text PDFReactor requires font data. This font data must be in TTF⁴¹ or in OTF⁴² format and may come from different types of sources (see [Font Sources \(p. 85\)](#)).

Important:

Using OpenType fonts requires Oracle Java SE 7 or higher.

It is recommended for a number of fonts to be available on the system PDFReactor runs on (see [Recommended Fonts \(p. 87\)](#)).

A.1 Font Sources

The font data of PDFReactor may come from different types of sources.

A.1.1 Core Fonts Pack

PDFReactor contains fonts that will be used for the [Default Font Mapping \(p. 88\)](#) when no other fonts could be registered on the system, e.g. because of insufficient file permissions or due to the fact that there are no fonts available.

These fonts are distributed by RealObjects and licensed by their respective authors under the SIL Open Font License⁴³. The packaged core fonts are:

- Roboto
- Tinos
- Anonymous Pro

A.1.2 System and JVM Font Directories

The main sources PDFReactor uses to retrieve font data are:

- fonts registered with the Java VM
- fonts located in the system font folder

Both provide fonts physically available to PDFReactor.

Java VM fonts are usually located in "JAVA_HOME/jre/lib/fonts". The location of the system font folder is platform dependent. PDFReactor registers fonts from these sources automatically.

⁴¹ *TrueType Font*

⁴² *OpenType Font*

⁴³ A free and open source license designed for fonts (http://scripts.sil.org/cms/scripts/page.php?id=OFL_web)

If PDFReactor was unable to retrieve any font data, fonts from the Core Fonts Pack will be used. (see [Core Fonts Pack \(p. 85\)](#)).

Note:

PDFReactor can be configured to ignore all system fonts and only use fonts that either have been specifically added via API method or that are webfonts from the document:

```
pdfReactor.setDisableSystemFonts(true);
```

A.1.3 Additional Fonts & Font Directories

PDFReactor allows setting additional fonts that are neither located in the system font directory nor the font directory of the Java VM. These fonts still need to be physically available to PDFReactor.

To register these fonts with PDFReactor via the Java API, use the following methods:

- `addFontDirectory` — Adds a directory PDFReactor will search for fonts.
- `addFont` — Adds an additional font from a specified source URL.

A font cache is created for each directory added by the `addFontDirectory` method. Should the contents of these directories change, please delete the font cache files before running PDFReactor. See the Chapter [The Font Cache Mechanism \(p. 87\)](#) for more information about the font cache.

A.1.4 CSS Defined Fonts

PDFReactor is capable of using fonts defined in CSS via the `@font-face` rule. These fonts are retrieved by PDFReactor along with other resources of the document (i.e. images) and will only be used to render the document they belong to.

Example: Defining a custom font

```
@font-face {
  font-family: "My Font";
  src: url("http://www.my-server.com/fonts/my-font.ttf");
}
```

A.2 Font Installation

This chapter describes how to install fonts on various operating systems.

Note:

To make use of new fonts please delete the font cache file before running PDFReactor. See the Chapter [The Font Cache Mechanism \(p. 87\)](#) for more information about the font cache.

Note:

If the PDFReactor Web Service is used, the Java VM should be restarted after installing new fonts.

A.2.1 Recommended Fonts

PDFReactor recommends the following fonts to be installed for the [Default Font Mapping \(p. 88\)](#). These fonts should be installed either within the system font directory or the font directory of the Java VM depending on the operating system:

- Arial
- Times New Roman
- Courier New

For a description on how to install the recommended fonts see the following chapters.

A.2.2 Font Installation on Windows

Windows should provide the recommended fonts by default, no extra installation should be required.

A.2.3 Font Installation on Linux

Linux does not provide the fonts recommended for PDFReactor by default.

To install these fonts download Microsoft's TrueType Core Font package (available at <http://corefonts.sourceforge.net/>). Unpack the fonts and move them into the directory:

```
"$JAVA_HOME/jre/lib/fonts"
```

The file permissions must allow the user running PDFReactor access to the *.ttf files in this directory.

A.2.4 Font Installation on Mac OS X

Mac OS X usually provides the recommended fonts in a proprietary format.

On recent versions of Mac OS X PDFReactor is not able to extract the TrueType font data from these fonts, due to a change within the font format. In this case the TTF fonts need to be installed separately.

To install these fonts download Microsoft's True Type Core Font package (available at <http://corefonts.sourceforge.net/>). Unpack the fonts and move them into the directory:

```
"/System/Library/Frameworks/JavaVM.framework/Version/1.6.0/Home/lib/fonts/"
```

The file permissions must allow the user running PDFReactor access to the *.ttf files in this directory.

A.3 The Font Cache Mechanism

One of the steps PDFReactor performs on startup is registering fonts. The first time this is done will take some time since every font inside the font directories available to PDFReactor will be identified and registered.

At the end of this step PDFReactor creates a font cache file that will be used on subsequent starts to significantly reduce its startup time. The font cache ensures the rendering process will start as soon as possible.

If the font cache file is present new fonts put into the font directories available to PDFReactor will be ignored by PDFReactor unless the font cache file has been deleted. Then PDFReactor will create a new font cache file on startup as it would on its first startup.

To delete the font cache file, visit the "user.home/.PDFReactor" directory and delete all files inside it.

A.3.1 Controlling the Font Registration and Caching Mechanism

It is possible to customize the registration and caching of fonts via the Java API:

- `setFontCachePath` — Specifies the location where the font cache file should be stored.
- `setCacheFonts` — Activates or deactivates the fontcache.
- `setDisableFontRegistration` — Specifies whether fonts are registered with PDFReactor
- `setDisableSystemFonts` — If set to true, PDFReactor will neither register system fonts, nor use the respective font cache.

A.4 Font Matching

A.4.1 Matching CSS Font Families

The default CSS font families are mapped as follows:

Default Font Mapping

CSS Font Family	Used Font	Core Font ⁴⁴
sans-serif	Arial	Roboto
serif	Times New Roman	Tinos
monospace	Courier New	Anonymous Pro

A.4.2 Font Alias Names

It is possible to add a font alias name for a font available in the system font directory or the font directory of the Java VM. The font alias name allows referencing to a font using a different name.

Authors can thus use a font alias name as the font-family value in CSS instead of the actual font name. Exchanging the font in all these documents can be done by changing the actual font behind the alias.

To define a font alias name via the Java API use the following method:

- `addFontAlias` — Adds an alias family for a registered font.

A.4.3 Automatic Font Fallback

Whenever the current font cannot be used to display a certain character, an automatic font fallback is used to find a replacement font for this character. To do so fonts are iterated according to the following parameters:

- The font-family property of the current element
- The method `setFontFallback`
- An internal list of recommended fonts
- All fonts on the system, starting with those with the most glyphs

⁴⁴ Used in cases where PDFReactor could not register any other fonts (see *Core Fonts Pack* (p. 85)).

APPENDIX B: CSS SUPPORT

B.1 CSS Pseudo-elements and -classes

PDFReactor supports the following pseudo-elements and -classes:

Supported Pseudo-elements/-classes

Pseudo-element/-class	Meaning	CSS Level
::after	Generated content after an element	CSS 2.1
::before	Generated content before an element	CSS 2.1
:empty	An element without children (including text nodes)	CSS 3
:first-child	An element, first child of its parent	CSS 2.1
:footnote-call	Generated content replacing elements that are moved to the footnote area. Usually shares its value with the corresponding footnote marker.	CSS 3
:footnote-marker	Generated content preceding footnotes. Usually shares its value with the corresponding footnote call.	CSS 3
:last-child	An element, last child of its parent	CSS 3
:nth-child(n)	An element, n^{th} child of its parent	CSS 3
:nth-last-child(n)	An element, n^{th} child of its parent	CSS 3
:first-of-type	An element, first sibling of its type.	CSS 3
:last-of-type	An element, last sibling of its type.	CSS 3
:only-of-type	An element, only sibling of its type.	CSS 3
:nth-of-type(n)	An element, n^{th} sibling of its type.	CSS 3
:nth-last-of-type(n)	An element, n^{th} last sibling of its type.	CSS 3
:not(s)	An element that does not match selector s	CSS 3

B.2 CSS Attribute Selector

PDFReactor supports the following CSS selectors which select elements that have certain attributes:

Supported attribute selectors

Attribute selector	Meaning	CSS Level
Elem[attr]	An Elem element with a attr attribute.	CSS 2.1
Elem[attr="val"]	An Elem element whose attr attribute value is exactly equal to "val".	CSS 2.1
Elem[attr~="val"]	An Elem element whose attr attribute value is a list of whitespace-separated values, one of which is exactly equal to "val".	CSS 2.1

Attribute selector	Meaning	CSS Level
Elem[attr^="val"]	An Elem element whose attr attribute value begins exactly with the string "val".	CSS 3
Elem[attr\$="val"]	An Elem element whose attr attribute value ends exactly with the string "val".	CSS 3
Elem[attr*="val"]	An Elem element whose attr attribute value contains the substring "val".	CSS 3

B.3 CSS At-Rules

Supported at-rules

Name	Syntax	Notes
@page	@page [name][:first :left :right] { <i>page ruleset</i> }	Page selectors, see Basic Page Layout (p. 49) for more information.
@charset	@charset "encoding"	The character encoding that is used. Please see the restrictions (p. 90) .
@font-face	@font-face { <i>font descriptors</i> }	A custom font. See CSS Defined Fonts (p. 86) .
@media	@media <i>media type</i> ,... { <i>ruleset</i> }	The specific media types to which this style sheet will apply. Also see How Do I Set Styles for Print or Screen Only? (p. 79) .
@import	@import {url} [<i>media type</i> ,...];	Imports another style sheet into this one.
@namespace	@namespace [<i>prefix</i>] <i>uri</i>	Declares an XML namespace, usually with a prefix.

Note:

The at-rule @charset does not work for a style sheet that is imported via @import.

B.4 Supported Page Size Formats

Key words for the supported A series formats, based on DIN 476/ISO 216, and their corresponding oversize formats

A series	Size [mm]	RA oversizes	Size [mm]	SRA oversizes	Size [mm]
A0	841 x 1189	RA0	860 x 1220	SRA0	900 x 1280
A1	594 x 841	RA1	610 x 860	SRA1	640 x 900
A2	420 x 594	RA2	430 x 610	SRA2	450 x 640
A3	297 x 420	RA3	305 x 430	SRA3	320 x 450
A4	210 x 297	RA4	215 x 305	SRA4	225 x 320
A5	148 x 210	RA5	152 x 215	SRA5	160 x 225
A6	105 x 148	RA6	107 x 152	SRA6	112 x 160
A7	74 x 105	RA7	76 x 107	SRA7	80 x 112

A series	Size [mm]	RA oversizes	Size [mm]	SRA oversizes	Size [mm]
A8	52 x 74	RA8	53 x 76	SRA8	56 x 80
A9	37 x 52				
A10	26 x 37				

CSS Key words for the supported B series formats

B series	Size [mm]
B1	707 x 1000
B2	500 x 707
B3	353 x 500
B4	250 x 353
B5	176 x 250
B6	125 x 176
B7	88 x 125
B8	62 x 88
B9	44 x 62
B10	31 x 44

Key words for the supported C series formats

C series	Size [mm]
C1	648 x 917
C2	458 x 648
C3	324 x 458
C4	229 x 324
C5	162 x 229
C6	114 x 162
C7	81 x 114
C8	57 x 81
C9	40 x 57
C10	28 x 40

Key words for supported international page formats

Page format	Size [in]
Letter	8.5 x 11
Legal	8.5 x 14
Ledger	11 x 17
Invoice	5.5 x 8
Executive	7.25 x 10.5
Broadsheet	17 x 22

B.5 Hyphenation Dictionaries

Supported hyphenation dictionaries

ISO 639-1 Language code	Language name
bg	Bulgarian
ca	Catalan
da	Danish
de	New German
de-1901	German traditional
el	Greek, Modern
en	English (US)
en-GB	English (GB)
et	Estonian
gl	Galician

ISO 639-1 Language code	Language name
ia	Interlingua
id	Indonesian (Bahasa Indonesia)
is	Icelandic
it	Italian
kmr	Kurmanji (Northern Kurdish)
la	Latin
nl	Dutch
pl	Polish
ru	Russian
sv	Swedish


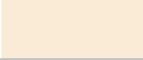


B.6 Supported length units


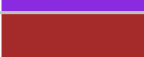







Supported length units



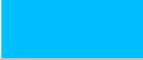

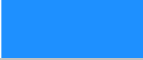

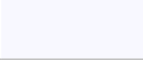



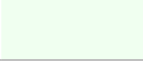

Length unit	Description
mm	milimeters
cm	centimeters
in	inches
pt	points
px	pixels
pc	pica
%	percent
em	Defines the proportion of the letter width and height with respect to the point size of the current font.




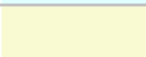
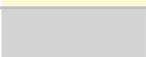

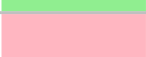


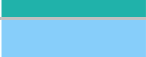
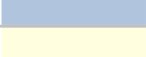













B.7 CSS Color Keywords



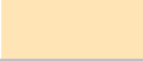


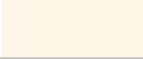
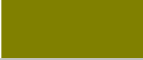




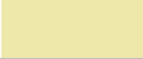

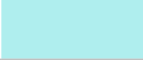

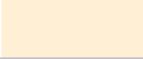
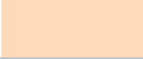



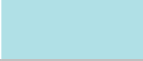



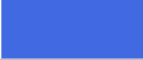




Supported Color Keywords



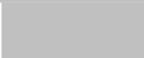





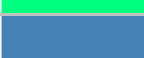






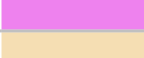
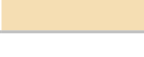

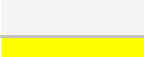




Color name	Color	hex RGB	Decimal
aliceblue		#F0F8FF	240,248,255
antiquewhite		#FAEBD7	250,235,215
aqua		#00FFFF	0,255,255
aquamarine		#7FFFD4	127,255,212

Color name	Color	hex RGB	Decimal
azure		#F0FFFF	240,255,255
beige		#F5F5DC	245,245,220
bisque		#FFE4C4	255,228,196
black		#000000	0,0,0
blanchedalmond		#FFEBCD	255,235,205
blue		#0000FF	0,0,255
blueviolet		#8A2BE2	138,43,226
brown		#A52A2A	165,42,42
burlywood		#DEB887	222,184,135
cadetblue		#5F9EA0	95,158,160
chartreuse		#7FFF00	127,255,0
chocolate		#D2691E	210,105,30
coral		#FF7F50	255,127,80
cornflowerblue		#6495ED	100,149,237
cornsilk		#FFF8DC	255,248,220
crimson		#DC143C	220,20,60
cyan		#00FFFF	0,255,255
darkblue		#00008B	0,0,139
darkcyan		#008B8B	0,139,139
darkgoldenrod		#B8860B	184,134,11
darkgray/darkgrey		#A9A9A9	169,169,169
darkgreen		#006400	0,100,0
darkkhaki		#BDB76B	189,183,107
darkmagenta		#8B008B	139,0,139
darkolivegreen		#556B2F	85,107,47
darkorange		#FF8C00	255,140,0
darkorchid		#9932CC	153,50,204
darkred		#8B0000	139,0,0
darksalmon		#E9967A	233,150,122

Color name	Color	hex RGB	Decimal
darkseagreen		#8FBC8F	143,188,143
darkslateblue		#483D8B	72,61,139
darkslategray/darkslategrey		#2F4F4F	47,79,79
darkturquoise		#00CED1	0,206,209
darkviolet		#9400D3	148,0,211
deeppink		#FF1493	255,20,147
deepskyblue		#00BFFF	0,191,255
dimgray/dimgrey		#696969	105,105,105
dodgerblue		#1E90FF	30,144,255
firebrick		#B22222	178,34,34
floralwhite		#FFFAF0	255,250,240
forestgreen		#228B22	34,139,34
fuchsia		#FF00FF	255,0,255
gainsboro		#DCDCDC	220,220,220
ghostwhite		#F8F8FF	248,248,255
gold		#FFD700	255,215,0
goldenrod		#DAA520	218,165,32
gray/grey		#808080	128,128,128
green		#008000	0,128,0
greenyellow		#ADFF2F	173,255,47
honeydew		#F0FFF0	240,255,240
hotpink		#FF69B4	255,105,180
indianred		#CD5C5C	205,92,92
indigo		#4B0082	75,0,130
ivory		#FFFFF0	255,255,240
khaki		#F0E68C	240,230,140
lavender		#E6E6FA	230,230,250
lavenderblush		#FFF0F5	255,240,245
lawngreen		#7CFC00	124,252,0

Color name	Color	hex RGB	Decimal
lemonchiffon		#FFFACD	255,250,205
lightblue		#ADD8E6	173,216,230
lightcoral		#F08080	240,128,128
lightcyan		#E0FFFF	224,255,255
lightgoldenrodyellow		#FAFAD2	250,250,210
lightgray/lightgrey		#D3D3D3	211,211,211
lightgreen		#90EE90	144,238,144
lightpink		#FFB6C1	255,182,193
lightsalmon		#FFA07A	255,160,122
lightseagreen		#20B2AA	32,178,170
lightskyblue		#87CEFA	135,206,250
lightslategray/lightslategrey		#778899	119,136,153
lightsteelblue		#B0C4DE	176,196,222
lightyellow		#FFFFE0	255,255,224
lime		#00FF00	0,255,0
limegreen		#32CD32	50,205,50
linen		#FAF0E6	250,240,230
magenta		#FF00FF	255,0,255
maroon		#800000	128,0,0
mediumaquamarine		#66CDAA	102,205,170
mediumblue		#0000CD	0,0,205
mediumorchid		#BA55D3	186,85,211
mediumpurple		#9370DB	147,112,219
mediumseagreen		#3CB371	60,179,113
mediumslateblue		#7B68EE	123,104,238
mediumspringgreen		#00FA9A	0,250,154
mediumturquoise		#48D1CC	72,209,204
mediumvioletred		#C71585	199,21,133
midnightblue		#191970	25,25,112

Color name	Color	hex RGB	Decimal
mintcream		#F5FFFA	245,255,250
mistyrose		#FFE4E1	255,228,225
moccasin		#FFE4B5	255,228,181
navajowhite		#FFDEAD	255,222,173
navy		#000080	0,0,128
oldlace		#FDF5E6	253,245,230
olive		#808000	128,128,0
olivedrab		#6B8E23	107,142,35
orange		#FFA500	255,165,0
orangered		#FF4500	255,69,0
orchid		#DA70D6	218,112,214
palegoldenrod		#EEE8AA	238,232,170
palegreen		#98FB98	152,251,152
paleturquoise		#AFEEEE	175,238,238
palevioletred		#DB7093	219,112,147
papayawhip		#FFEFD5	255,239,213
peachpuff		#FFDAB9	255,218,185
peru		#CD853F	205,133,63
pink		#FFC0CB	255,192,203
plum		#DDA0DD	221,160,221
powderblue		#B0E0E6	176,224,230
purple		#800080	128,0,128
red		#FF0000	255,0,0
rosybrown		#BC8F8F	188,143,143
royalblue		#4169E1	65,105,225
saddlebrown		#8B4513	139,69,19
salmon		#FA8072	250,128,114
sandybrown		#F4A460	244,164,96
seagreen		#2E8B57	46,139,87

Color name	Color	hex RGB	Decimal
seashell		#FFF5EE	255,245,238
sienna		#A0522D	160,82,45
silver		#C0C0C0	192,192,192
skyblue		#87CEEB	135,206,235
slateblue		#6A5ACD	106,90,205
slategray/slategrey		#708090	112,128,144
snow		#FFFAFA	255,250,250
springgreen		#00FF7F	0,255,127
steelblue		#4682B4	70,130,180
tan		#D2B48C	210,180,140
teal		#008080	0,128,128
thistle		#D8BFD8	216,191,216
tomato		#FF6347	255,99,71
turquoise		#40E0D0	64,224,208
violet		#EE82EE	238,130,238
wheat		#F5DEB3	245,222,179
white		#FFFFFF	255,255,255
whitesmoke		#F5F5F5	245,245,245
yellow		#FFFF00	255,255,0
yellowgreen		#9ACD32	154,205,50
ro-comment-highlight		#FFFF0B	255,255,11
ro-comment-underline		#23FF06	35,255,6
ro-comment-strikeout		#FB0007	251,0,7

B.8 Counter and Ordered List Style Types

Supported counter and ordered list style types

Counter style name	1	12	123	1234
decimal	1.	12.	123.	1234.

Counter style name	1	12	123	1234
decimal-leading-zero	01.	12.	123.	1234.
super-decimal	¹ .	¹² .	¹²³ .	¹²³⁴ .
upper-hexadecimal	1.	C.	7B.	4D2.
lower-hexadecimal	1.	c.	7b.	4d2.
octal	1.	14.	173.	2322.
binary	1.	1100.	1111011.	10011010010.
upper-roman	I.	XII.	CXXIII.	MCCXXXIV.
lower-roman	i.	xii.	cxxiii.	mccxxxiv.
upper-alpha	A.	L.	DS.	AUL.
lower-alpha	a.	l.	ds.	aul.
arabic-indic	۱.	۱۲.	۱۲۳.	۱۲۳۴.
armenian	Ա.	ԺԲ.	ՃԻԳ.	ՌՄՂԴ.
upper-armenian	Ա.	ԺԲ.	ՃԻԳ.	ՌՄՂԴ.
lower-armenian	ա.	ժբ.	ճիգ.	ռմղդ.
bengali	১.	১২.	১২৩.	১২৩৪.
cambodian	១.	១២.	១២៣.	១២៣៤.
devanagari	१.	१२.	१२३.	१२३४.
georgian	ა.	იბ.	ჩკგ.	ჩსლდ.
upper-greek	A.	M.	ZI.	ΓΗΣ.
lower-greek	α.	μ.	εθ.	βηο.
gujarati	૧.	૧૨.	૧૨૩.	૧૨૩૪.

Counter style name	1	12	123	1234
gurmukhi	ੴ.	ੴ੨.	ੴ੨੩.	ੴ੨੩੪.
hiragana	あ.	し.	いひ.	のめ.
hiragana-iroha	い.	を.	ろや.	のを.
japanese-formal	壹.	拾貳.	壹佰貳拾參.	壹仟貳佰參拾肆.
japanese-informal	一.	十二.	一百二十三.	一千二百三十四.
kannada	೦.	೦೨.	೦೨೩.	೦೨೩೪.
katakana	ア.	シ.	イヒ.	ノメ.
katakana-iroha	イ.	ヲ.	ロヤ.	ノヲ.
khmer	១.	១២.	១២៣.	១២៣៤.
lao	໐.	໐໒.	໐໒໓.	໐໒໓໔.
upper-latin	A.	L.	DS.	AUL.
lower-latin	a.	l.	ds.	aul.
malayalam	൧.	൧൨.	൧൨൩.	൧൨൩൪.
mongolian	᠐.	᠐᠒.	᠐᠒ᠢ.	᠐᠒ᠢᠶ.
myanmar	၁.	၁၂.	၁၂၃.	၁၂၃၄.
oriya	୧.	୧୨.	୧୨୩.	୧୨୩୪.
persian	۱.	۱۲.	۱۲۳.	۱۲۳۴.
simp-chinese-formal	壹.	拾貳.	壹佰貳拾參.	壹仟貳佰參拾肆.
simp-chinese-informal	一.	十二.	一百二十三.	一千二百三十四.
telugu	౦.	౦౨.	౦౨౩.	౦౨౩౪.
thai	๑.	๑๒.	๑๒๓.	๑๒๓๔.

Counter style name	1	12	123	1234
tibetan	༡.	༡༢.	༡༢༣.	༡༢༣༤.
urdu	۱.	۱۲.	۱۲۳.	۱۲۳۴.
ro-spelled-out-en	one	twelve	one hundred twenty-three	one thousand two hundred thirty-four
ro-spelled-out-en-ordinal	first	twelfth	one hundred twenty-third	one thousand two hundred thirty-fourth
ro-spelled-out-de	eins	zwölf	ehundertdreißig	eintausendzweihundertvierunddreißig

B.9 CSS Property Documentation

PDFreactor supports the following CSS properties:

-ro-align-content

Aligns the content of a block-level element inside the element. The property has no effect if the height of the content is larger than the block element's height.

Value:	auto start center end
Initial:	auto
Applies To:	block-level elements
Inherited:	No

auto

The content is positioned as usual inside the block.

start

Content is positioned at the top of the block.

center

Content is vertically centered inside the block.

end

Content is positioned at the bottom of the block.

■ *Source:* CSS3, Proprietary

-ro-alt-text

The property -ro-alt-text is used to specify an alternative description for an element for use in PDF tags.

Value:	none <string> ro-attr(<attribute>)
Initial:	
Inherited:	No

none

The element receives no alternate text.

<string>

Defines alternate text for the element.

ro-attr(<attribute>)

Defines alternate text for the element from an attribute of the element.

■ *Source:* Proprietary

■ *More information:* [Tagged PDF \(p. 38\)](#)

–ro-anchor

This property allows to define an anchor via style.

Value:	none <identifier> ro-attr(<attribute> ro-ident)
Initial:	none
Inherited:	No

none

The element is not an anchor.

<identifier>

The element is an anchor with the given identifier.

ro-attr(<attribute> ro-ident)

The element is an anchor with the given identifier resolved from the ro-attr() function.

- **Source:** Proprietary
- **More information:** [Links](#) (p. 34)

–ro-art-size

Specifies the size of the ArtBox, one of the PDF page boxes.

Value:	none <length>{1,2} [<page-size> [portrait landscape]] media trim crop
Initial:	none
Applies To:	page context
Inherited:	No

none

The element does not specify an ArtBox.

media

The ArtBox is specified with the same dimensions as the MediaBox.

trim

The ArtBox is specified with the same dimensions as the TrimBox.

crop

The ArtBox is specified with the same dimensions as the CropBox.

- **Source:** Proprietary
- **More information:** [Page Boxes](#) (p. 69)

–ro-author

Sets the author in the metadata of the PDF document. Multiple values are concatenated to one string. (When applied to multiple elements the values are concatenated, separated by a comma.)

Value:	none [<string> content() ro-attr(<attribute>)]+
Initial:	none
Applies To:	all elements
Inherited:	No

none

Does not set a author.

<string>

Sets the specified string as author.

content()

Sets the author from the content of the element.

ro-attr(<attribute>)

Sets the author from the specified attribute of the element.

- **Source:** Proprietary
- **See also:** [-ro-keywords](#) (p. 136), [-ro-subject](#) (p. 157), [-ro-title](#) (p. 159)
- **More information:** [Metadata](#) (p. 37)

background

This property is a shorthand property for setting most background properties at the same place in the style sheet.

Value:	<bg-image> <position> [/ <bg-size>]? <repeat-style> <attachment> <box> <box> <'background-color'>
Initial:	see individual properties
Inherited:	No

- **Source:** CSS3
- **See also:** [background-attachment](#) (p. 103), [background-color](#) (p. 104), [background-image](#) (p. 104), [background-origin](#) (p. 105), [background-position](#) (p. 105), [background-repeat](#) (p. 106), [background-size](#) (p. 106)

background-attachment

If background images are specified, this property specifies whether they are fixed with regard to the viewport ('fixed') or scroll along with the element ('scroll').

Value:	scroll fixed inherit
Initial:	scroll
Inherited:	No

scroll

This keyword means that the background is fixed with regard to the element itself and does not scroll with its contents. (It is effectively attached to the element's border.)

fixed

This keyword means that the background is fixed with regard to the viewport. Even if an element has a scrolling mechanism, a 'fixed' background doesn't move with the element.

- **Source:** CSS 2.1

background-clip

Determines the background painting area, which determines the area within which the background is painted.

Value:	border-box padding-box content-box
Initial:	border-box
Inherited:	No

border-box

The background is painted within (clipped to) the border box.

padding-box

The background is painted within (clipped to) the padding box.

content-box

The background is painted within (clipped to) the content box.

■ *Source:* CSS3

background-color

This property sets the background color of an element. The color is drawn behind any background images.

Value:	<color> transparent inherit
Initial:	transparent
Inherited:	No

<color>

Is a CSS <color> that describes the uniform color of the background. Even if one or several background-image are defined, this color can be affect the rendering, by transparency if the images aren't opaque.

■ *Source:* CSS 2.1

■ *More information:* [Color Keywords \(p. 92\)](#)

background-image

This property sets the background image of an element. When setting a background image, authors should also specify a background color that will be used when the image is unavailable. When the image is available, it is rendered on top of the background color. (Thus, the color is visible in the transparent parts of the image).

Value:	<uri> none
Initial:	none
Inherited:	No

<uri>

The format of a URI value is 'url(' followed by optional white space followed by an optional single quote (') or double quote (") character followed by the URI itself, followed by an optional single quote (') or double quote (") character followed by optional white space followed by ')'. The two quote characters must be the same.

■ *Source:* CSS 2.1

background-origin

For elements rendered as a single box, specifies the background positioning area. For elements rendered as multiple boxes (e.g., inline boxes on several lines, boxes on several pages), specifies which boxes 'box-decoration-break' operates on to determine the background positioning area(s).

Value:	border-box padding-box content-box
Initial:	padding-box
Inherited:	No

padding-box

The position is relative to the padding box. (For single boxes '0 0' is the upper left corner of the padding edge, '100% 100%' is the lower right corner.)

border-box

The position is relative to the border box.

content-box

The position is relative to the content box.

■ *Source:* CSS3

background-position

If a background image has been specified, this property specifies its initial position. If only one value is specified, the second value is assumed to be 'center'. If at least one value is not a keyword, then the first value represents the horizontal position and the second represents the vertical position. Negative <percentage> and <length> values are allowed.

Value:	[[<percentage> left center right] [<percentage> <length> top center bottom]?] [[left center right] [top center bottom]] inherit
Initial:	0% 0%
Inherited:	No

<percentage>

A percentage X aligns the point X% across (for horizontal) or down (for vertical) the image with the point X% across (for horizontal) or down (for vertical) the element's padding box. For example, with a value pair of '0% 0%', the upper left corner of the image is aligned with the upper left corner of the padding box. A value pair of '100% 100%' places the lower right corner of the image in the lower right corner of the padding box. With a value pair of '14% 84%', the point 14% across and 84% down the image is to be placed at the point 14% across and 84% down the padding box.

<length>

A length L aligns the top left corner of the image a distance L to the right of (for horizontal) or below (for vertical) the top left corner of the element's padding box. For example, with a value pair of '2cm 1cm', the upper left corner of the image is placed 2cm to the right and 1cm below the upper left corner of the padding box.

top

Equivalent to '0%' for the vertical position.

right

Equivalent to '100%' for the horizontal position.

bottom

Equivalent to '100%' for the vertical position.

left

Equivalent to '0%' for the horizontal position.

center

Equivalent to '50%' for the horizontal position if it is not otherwise given, or '50%' for the vertical position if it is.

■ Source: CSS 2.1

background-repeat

If a background image is specified, this property specifies whether the image is repeated (tiled), and how. All tiling covers the content, padding and border areas of a box.

Value:	repeat repeat-x repeat-y no-repeat inherit
Initial:	repeat
Inherited:	No

repeat

The image is repeated both horizontally and vertically.

repeat-x

The image is repeated horizontally only.

repeat-y

The image is repeated vertically only.

no-repeat

The image is not repeated: only one copy of the image is drawn.

■ Source: CSS 2.1

background-size

Specifies the size of the background images.

Value:	[<length> <percentage> auto]{1,2}
Initial:	auto
Inherited:	No

[<length> | <percentage> | auto]{1,2}

The first value gives the width of the corresponding image, the second value its height. If only one value is given the second is assumed to be 'auto'. A percentage is relative to the background positioning area. An 'auto' value for one dimension is resolved by using the image's intrinsic ratio and the size of the other dimension, or failing that, using the image's intrinsic size, or failing that, treating it as 100%. If both values are 'auto' then the intrinsic width and/or height of the image should be used, if any, the missing dimension (if any) behaving as 'auto' as described above. If the image has neither an intrinsic width nor an intrinsic height, its size is determined as for 'contain'. Negative values are not allowed.

■ Source: CSS3

-ro-bleed-width

Specifies the width of the bleed area around the TrimBox. This implicitly defines the size of the BleedBox. Twice the bleed width added up on the width and height of the TrimBox' (twice for both sides of the TrimBox).

Value:	none <length>
Initial:	none
Applies To:	page context
Inherited:	No

none

There is no bleed area round the TrimBox.

<length>

The length of the bleed area on each side of the TrimBox.

- *Source:* Proprietary
- *See also:* [size \(p. 155\)](#)
- *More information:* [Page Boxes \(p. 69\)](#)

-ro-bookmarks-enabled

This property allows to enable or disable PDF bookmarks for the content inside an iframe. If the iframe is seamless, this property is set to true by default.

Value:	true false
Initial:	false
Applies To:	iframe
Inherited:	No

- *Source:* Proprietary
- *More information:* [iFrames \(p. 26\)](#)

border

This property is a shorthand property for setting the same width, color, and style for all four borders of a box.

Value:	<line-width> <line-style> <color>
Initial:	see individual properties
Inherited:	No

border-bottom-left-radius**border-bottom-right-radius****border-top-left-radius****border-top-right-radius**

The two length or percentage values of the 'border-*-radius' properties define the radii of a quarter ellipse that defines the shape of the corner of the outer border edge. The first value is the horizontal radius, the second the vertical radius. If the second value is omitted it is copied from the first. If either length is zero, the corner is square, not rounded. Percentages for the horizontal radius refer to the width of the border box, whereas percentages for the vertical radius refer to the height of the border box. Negative values are not allowed.

Value:	[<length> <percentage>]{1,2}
Initial:	0
Applies To:	all elements (but see prose)
Inherited:	No

■ Source: CSS3

border-collapse

This property selects a table's border model. The value 'separate' selects the separated borders border model. The value 'collapse' selects the collapsing borders model.

Value:	collapse separate inherit
Initial:	separate
Applies To:	'table' and 'inline-table' elements
Inherited:	Yes

■ Source: CSS 2.1

border-color

The 'border-color' property sets the color of the four borders. The 'border-color' property can have from one to four component values, and the values are set on the different sides as for 'border-width'.

Value:	[<color> transparent]{1,4} inherit
Initial:	see individual properties
Inherited:	No

<color>

Specifies a color value.

transparent

The border is transparent (though it may have width).

■ Source: CSS 2.1

■ See also: [border-*-color](#) (p. 110)

■ More information: [Color Keywords](#) (p. 92)

border-radius

The 'border-radius' shorthand sets all four 'border-*-radius' properties. If values are given before and after the slash, then the values before the slash set the horizontal radius and the values after the slash set the vertical radius. If there is no slash, then the values set both radii equally. The four values for each radii are given in the order top-left, top-right, bottom-right, bottom-left. If bottom-left is omitted it is the same as top-right. If bottom-right is omitted it is the same as top-left. If top-right is omitted it is the same as top-left.

Value:	[<length> <percentage>]{1,4} [/ [<length> <percentage>]{1,4}]?
Initial:	see individual properties
Applies To:	all elements (but see prose)
Inherited:	No

- Source: CSS3
- See also: [border-*-radius](#) (p. 108)

border-spacing

The lengths specify the distance that separates adjoining cell borders. If one length is specified, it gives both the horizontal and vertical spacing. If two are specified, the first gives the horizontal spacing and the second the vertical spacing. Lengths may not be negative. The distance between the table border and the borders of the cells on the edge of the table is the table's padding for that side, plus the relevant border spacing distance. For example, on the right hand side, the distance is padding-right + horizontal border-spacing.

Value:	<length> <length>? inherit
Initial:	0
Applies To:	'table' and 'inline-table' elements
Inherited:	Yes

- Source: CSS 2.1

border-style

The 'border-style' property sets the style of the four borders. It can have from one to four component values, and the values are set on the different sides as for 'border-width'.

Value:	<border-style>{1,4} inherit
Initial:	see individual properties
Inherited:	No

<border-style>

see individual properties

- Source: CSS 2.1
- See also: [border-*-style](#) (p. 110)

border-top

border-right

border-bottom

border-left

This is a shorthand property for setting the width, style, and color of the top, right, bottom, and left border of a box.

Value:	[<border-width> <border-style> 'border-top-color'] inherit
Initial:	see individual properties
Inherited:	No

■ **Source:** CSS 2.1

■ **See also:** [border-*-color](#) (p. 110), [border-*-style](#) (p. 110), [border-*-width](#) (p. 111)

border-top-color

border-right-color

border-bottom-color

border-left-color

The 'border-*-color' properties set the color of the specified border.

Value:	<color> transparent inherit
Initial:	the value of the 'color' property
Inherited:	No

<color>

Specifies a color value.

transparent

The border is transparent (though it may have width).

■ **Source:** CSS 2.1

border-top-style

border-right-style

border-bottom-style

border-left-style

The border style properties specify the line style of a box's border (solid, double, dashed, etc.). The properties defined in this section refer to the <border-style> value type, which may take one of the following values:

Value:	<border-style> inherit
Initial:	none
Inherited:	No

none

No border; the computed border width is zero.

hidden

Same as 'none', except in terms of border conflict resolution for table elements.

dotted

The border is a series of dots.

dashed

The border is a series of short line segments.

solid

The border is a single line segment.

double

The border is two solid lines. The sum of the two lines and the space between them equals the value of 'border-width'.

groove

The border looks as though it were carved into the canvas.

ridge

The opposite of 'groove': the border looks as though it were coming out of the canvas.

inset

The border makes the box look as though it were embedded in the canvas.

outset

The opposite of 'inset': the border makes the box look as though it were coming out of the canvas.

■ *Source:* CSS 2.1

border-top-width**border-right-width****border-bottom-width****border-left-width**

The border width properties specify the width of the border area. The properties defined in this section refer to the <border-width> value type, which may take one of the following values:

Value:	<border-width> inherit
Initial:	medium
Inherited:	No

thin

A thin border.

medium

A medium border.

thick

A thick border.

<length>

The border's thickness has an explicit value. Explicit border widths cannot be negative.

■ *Source:* CSS 2.1

border-width

This property is a shorthand property for setting 'border-top-width', 'border-right-width', 'border-bottom-width', and 'border-left-width' at the same place in the style sheet. If there is only one component value, it applies to all sides. If there are two values, the top and bottom borders are set to the first value and the right and left are set to the second. If there are three values, the top is set to the first value, the left and right are set to the second, and the bottom is set to the third. If there are four values, they apply to the top, right, bottom, and left, respectively.

Value:	<border-width>{1,4} inherit
Initial:	see individual properties
Inherited:	No

- Source: CSS 2.1
- See also: [border-*-width](#) (p. 111)

bottom

Like 'top', but specifies how far a box's bottom margin edge is offset above the bottom of the box's containing block. For relatively positioned boxes, the offset is with respect to the bottom edge of the box itself.

Value:	<length> <percentage> auto inherit
Initial:	auto
Applies To:	positioned elements
Inherited:	No

<length>

The offset is a fixed distance from the reference edge. Negative values are allowed.

<percentage>

The offset is a percentage of the containing block's height. Negative values are allowed.

auto

For non-replaced elements, the effect of this value depends on which of related properties have the value 'auto' as well. For replaced elements, the effect of this value depends on the intrinsic dimensions of the replaced content.

- Source: CSS 2.1

break-before

break-after

These properties describe page/column/region break behavior before/after the element's box.

Value:	auto always avoid left right page column region avoid-page avoid-column avoid-region
Initial:	auto
Applies To:	block-level elements
Inherited:	No

auto

Neither force nor forbid a page/column/region break before/after the box.

always

Always force a page break before/after the box.

avoid

Avoid a page/column/region break before/after the box.

left

Force one or two page breaks before/after the box so that the next page is formatted as a left page.

right

Force one or two page breaks before/after the box so that the next page is formatted as a right page.

page

Always force a page break before (after) the box.

column

Always force a column break before/after the box.

region

Always force a region break before/after the box.

avoid-page

Avoid a page break before/after the box.

avoid-column

Avoid a column break before/after the box.

avoid-region

Avoid a region break before/after the box.

■ *Source:* CSS3, Experimental

■ *More information:* [Breaking Boxes \(p. 67\)](#)

break-inside

This property describes the page/column/region break behavior inside the element's box.

Value:	auto avoid avoid-page avoid-column avoid-region
Initial:	auto
Applies To:	block-level elements
Inherited:	No

auto

Neither force nor forbid a page break inside the generated box.

avoid

Avoid any break inside the generated box.

avoid-page

Avoid a page/column/region break inside the generated box.

avoid-column

Avoid a column break inside the generated box.

avoid-region

Avoid a region break inside the generated box.

- *Source:* CSS3, Experimental
- *More information:* [Breaking Boxes \(p. 67\)](#)

caption-side

This property specifies the position of the caption box with respect to the table box.

Value:	top bottom inherit
Initial:	top
Applies To:	'table-caption' elements
Inherited:	Yes

top

Positions the caption box above the table box.

bottom

Positions the caption box below the table box.

- *Source:* CSS 2.1

clear

This property indicates which sides of an element's box(es) may not be adjacent to an earlier floating box. The 'clear' property does not consider floats inside the element itself or in other block formatting contexts.

Value:	none left right both inherit
Initial:	none
Inherited:	No

left

Requires that the top border edge of the box be below the bottom outer edge of any left-floating boxes that resulted from elements earlier in the source document.

right

Requires that the top border edge of the box be below the bottom outer edge of any right-floating boxes that resulted from elements earlier in the source document.

both

Requires that the top border edge of the box be below the bottom outer edge of any right-floating and left-floating boxes that resulted from elements earlier in the source document.

none

No constraint on the box's position with respect to floats.

- *Source:* CSS 2.1

clip

A clipping region defines what portion of an element's border box is visible. By default, the element is not clipped. However, the clipping region may be explicitly set with the 'clip' property.

Value:	<shape> auto inherit
Initial:	auto
Applies To:	absolutely positioned elements
Inherited:	No

auto

The element does not clip.

<shape>

In CSS 2.1, the only valid <shape> value is: rect(<top>, <right>, <bottom>, <left>) where <top> and <bottom> specify offsets from the top border edge of the box, and <right>, and <left> specify offsets from the left border edge of the box. Authors should separate offset values with commas. <top>, <right>, <bottom>, and <left> may either have a <length> value or 'auto'. Negative lengths are permitted. The value 'auto' means that a given edge of the clipping region will be the same as the edge of the element's generated border box (i.e., 'auto' means the same as '0' for <top> and <left>, the same as the used value of the height plus the sum of vertical padding and border widths for <bottom>, and the same as the used value of the width plus the sum of the horizontal padding and border widths for <right>, such that four 'auto' values result in the clipping region being the same as the element's border box).

■ *Source:* CSS 2.1

color

This property describes the foreground color of an element's text content.

Value:	<color> inherit
Initial:	depends on user agent
Inherited:	Yes

<color>

Specifies the foreground color.

■ *Source:* CSS 2.1

■ *More information:* [Color Keywords \(p. 92\)](#)

`-ro-colorbar-top-left`
`-ro-colorbar-top-right`
`-ro-colorbar-bottom-left`
`-ro-colorbar-bottom-right`
`-ro-colorbar-left-top`
`-ro-colorbar-left-bottom`
`-ro-colorbar-right-top`
`-ro-colorbar-right-bottom`

Color bars for print layout in oversized pages.

Value:	gradient-tint progressive-color [<color>]+ none
Initial:	none
Applies To:	page context
Inherited:	No

gradient-tint

Defines a set of 11 grayscale colors, starting with a CMYK value of 0% each and raising the cyan, magenta and yellow values by 10% on every step.

progressive-color

Defines a set including solid process colors (cyan, magenta, yellow, black), solid overprint colors (cyan & magenta, cyan & yellow, magenta & yellow) and a 50% tint of each of the process colors.

[<color>]+

One or more colors which will be sequentially painted from left to right or from top to bottom respectively.

- Source: Proprietary
- More information: [Printer Marks \(p. 70\)](#)

`-ro-column-break-before`
`-ro-column-break-after`

Deprecated! These properties describe column break behavior before/after the element's box.

Value:	auto always avoid inherit
Initial:	auto
Applies To:	block-level elements
Inherited:	No

- Source: Proprietary
- **Deprecated!** Use `break-before`, `break-after` (p. 112) instead.

column-count

This property describes the number of columns of a multicol element.

Value:	<integer> auto
Initial:	auto
Applies To:	non-replaced block-level elements (except table elements), table cells, and inline-block elements
Inherited:	No

auto

means that the number of columns will be determined by other properties (e.g., 'column-width', if it has a non-auto value).

<integer>

describes the optimal number of columns into which the content of the element will be flowed. Values must be greater than 0. If both 'column-width' and 'column-count' have non-auto values, the integer value describes the maximum number of columns.

- *Source:* CSS3
- *More information:* [Multi-column Layout \(p. 63\)](#)

-ro-column-count

Deprecated! This property describes the number of columns of a multicol element.

Value:	<integer> auto
Initial:	auto
Inherited:	No

- *Source:* CSS3, Experimental
- **Deprecated!** Use [column-count \(p. 116\)](#) instead.

column-fill

In continuous media, this property will only be consulted if the length of columns has been constrained. Otherwise, columns will automatically be balanced.

Value:	balance auto
Initial:	balance
Applies To:	multicol elements
Inherited:	No

balance

Balance content equally between columns, if possible.

auto

Fills columns sequentially.

- *Source:* CSS3
- *More information:* [Multi-column Layout \(p. 63\)](#)

-ro-column-fill

Deprecated! In continuous media, this property will only be consulted if the length of columns has been constrained. Otherwise, columns will automatically be balanced.

Value:	balance auto
Initial:	balance
Inherited:	No

- *Source:* CSS3, Experimental
- **Deprecated!** Use [column-fill \(p. 117\)](#) instead.

column-gap

The 'column-gap' property sets the gap between columns. If there is a column rule between columns, it will appear in the middle of the gap.

Value:	<length> normal
Initial:	normal
Applies To:	multicol elements
Inherited:	No

normal

The 'normal' value is UA-specific.

<length>

Specifies the width of the gap. Column gaps cannot be negative.

- *Source:* CSS3
- *More information:* [Multi-column Layout \(p. 63\)](#)

-ro-column-gap

Deprecated! The 'column-gap' property sets the gap between columns. If there is a column rule between columns, it will appear in the middle of the gap.

Value:	<length> normal
Initial:	normal
Inherited:	No

- *Source:* CSS3, Experimental
- *Deprecated!* Use [column-gap \(p. 118\)](#) instead.

column-rule

This property is a shorthand for setting 'column-rule-width', 'column-rule-style', and 'column-rule-color' at the same place in the style sheet. Omitted values are set to their initial values.

Value:	<color> <border-style> <border-width> <percentage> none
Initial:	see individual properties
Applies To:	multicol elements
Inherited:	No

- *Source:* CSS3
- *More information:* [Multi-column Layout \(p. 63\)](#)

`-ro-column-rule`

Deprecated! This property is a shorthand for setting 'column-rule-width', 'column-rule-style', and 'column-rule-color' at the same place in the style sheet. Omitted values are set to their initial values.

Value:	<code><color> <border-style> <border-width> <percentage> none</code>
Initial:	see individual properties
Inherited:	No

- Source: CSS3, Experimental
- **Deprecated!** Use `column-rule` (p. 118) instead.

`column-rule-color`

This property sets the color of the column rule.

Value:	<code><color> none</code>
Initial:	same as for 'color' property
Applies To:	multicol elements
Inherited:	No

- Source: CSS3
- More information: [Multi-column Layout](#) (p. 63), [Color Keywords](#) (p. 92)

`-ro-column-rule-color`

Deprecated! This property sets the color of the column rule.

Value:	<code><color> none</code>
Initial:	same as for 'color' property
Inherited:	No

- Source: CSS3, Experimental
- **Deprecated!** Use `column-rule-color` (p. 119) instead.

`column-rule-style`

The 'column-rule-style' property sets the style of the rule between columns of an element. The `<border-style>` values are defined in CSS2.1 and the values are interpreted as in the the collapsing border model.

Value:	<code><border-style> none</code>
Initial:	none
Applies To:	multicol elements
Inherited:	No

- Source: CSS3
- More information: [Multi-column Layout](#) (p. 63)

`-ro-column-rule-style`

Deprecated! The 'column-rule-style' property sets the style of the rule between columns of an element. The <border-style> values are defined in CSS2.1 and the values are interpreted as in the the collapsing border model.

Value:	<border-style> none
Initial:	none
Inherited:	No

- Source: CSS3, Experimental
- **Deprecated!** Use `column-rule-style` (p. 119) instead.

`column-rule-width`

This property sets the width of the rule between columns. Negative values are not allowed.

Value:	<border-width> <percentage> none
Initial:	medium
Applies To:	multicol elements
Inherited:	No

- Source: CSS3
- More information: [Multi-column Layout](#) (p. 63)

`-ro-column-rule-width`

Deprecated!

Value:	<border-width> <percentage> none
Initial:	
Inherited:	No

- Source: CSS3, Experimental
- **Deprecated!** Use `column-rule-width` (p. 120) instead.

`column-span`

This property describes how many columns an element spans across.

Value:	none all
Initial:	none
Applies To:	block-level elements, except floating and absolutely positioned elements
Inherited:	No

none

The element does not span multiple columns.

all

The element spans across all columns. Content in the normal flow that appears before the element is automatically balanced across all columns before the element appears. The element establishes a new block formatting context.

- *Source:* CSS3
- *More information:* [Multi-column Layout \(p. 63\)](#)

-ro-column-span

Deprecated! This property describes how many columns an element spans across.

Value:	none all
Initial:	none
Inherited:	No

- *Source:* CSS3, Experimental
- *Deprecated!* Use [column-span \(p. 120\)](#) instead.

column-width

This property describes the width of columns in multicol elements.

Value:	<length> auto
Initial:	auto
Applies To:	non-replaced block-level elements (except table elements), table cells, and inline-block elements
Inherited:	No

auto

means that the column width will be determined by other properties (e.g., 'column-count', if it has a non-auto value).

<length>

describes the optimal column width. The actual column width may be wider (to fill the available space), or narrower (only if the available space is smaller than the specified column width). Specified values must be greater than 0.

- *Source:* CSS3
- *More information:* [Multi-column Layout \(p. 63\)](#)

-ro-column-width

Deprecated! This property describes the width of columns in multicol elements.

Value:	<length> auto
Initial:	auto
Inherited:	No

- *Source:* CSS3, Experimental
- *Deprecated!* Use [-ro-column-width \(p. 121\)](#) instead.

columns

This is a shorthand property for setting 'column-width' and 'column-count'. Omitted values are set to their initial values.

Value:	<integer> <length> auto
Initial:	see individual properties
Applies To:	non-replaced block-level elements (except table elements), table cells, and inline-block elements
Inherited:	No

- Source: CSS3
- More information: [Multi-column Layout \(p. 63\)](#)

-ro-columns

Deprecated! This is a shorthand property for setting 'column-width' and 'column-count'. Omitted values are set to their initial values.

Value:	<integer> <length> auto
Initial:	see individual properties
Inherited:	No

- Source: CSS3, Experimental
- **Deprecated!** Use [columns \(p. 122\)](#) instead.

-ro-comment-color

Specifies the color of the comment. The initial value of this property depends on the value of the '-ro-comment-style' property: 'ro-comment-highlight' for 'note' and 'highlight', 'ro-comment-underline' for 'underline' and 'squiggly', 'ro-comment-strikeout' for 'strikeout'

Value:	<color> currentColor
Initial:	see text
Inherited:	No

<color>

The color of the comment.

currentColor

Sets the color of the comment to the same value as the computed value of the "color" CSS property.

- Source: Proprietary
- More information: [Comments \(p. 34\)](#), [Color Keywords \(p. 92\)](#)

-ro-comment-content

Specifies the content of a comment.

Value:	none [<string> content() ro-attr(<attribute>)]+
Initial:	none
Inherited:	No

none

The comment receives no content.

<string>

Defines the content of the comment.

content()

Defines the content of the comment from the content of the element.

ro-attr(<attribute>)

Defines the content of the comment from an attribute of the element.

- Source: Proprietary
- More information: [Comments \(p. 34\)](#)

-ro-comment-date

Specifies the date of the comment which will be formatted according the the value of the "-ro-comment-dateformat" property. If no date is specified, the current date will be used.

Value:	<string>
Initial:	see text
Inherited:	No

<string>

The date of the comment.

- Source: Proprietary
- More information: [Comments \(p. 34\)](#)

-ro-comment-dateformat

The format wich is applied to the string value of the "-ro-comment-date" property. The format of this value is similar to the Java SimpleDateFormat class. The initial value is the ISO date format.

Value:	<string>
Initial:	"yyyy-MM-dd'T'HH:mm:ss"
Inherited:	No

<string>

The date format for the comment.

- Source: Proprietary
- More information: [Comments \(p. 34\)](#)

-ro-comment-position

The position of the note icon of the comment. This property is only applicable when the value of the property "-ro-comment-style" is set to note.

Value:	auto page-left page-right
Initial:	auto
Inherited:	No

page-left

Shifts the note icon to the left side of the page.

page-right

Shifts the note icon to the right side of the page.

- **Source:** Proprietary
- **More information:** [Comments \(p. 34\)](#)

-ro-comment-start**-ro-comment-end**

Specifies the start or end elements which encompass commented text. Both properties have to be specified in the respective element to link the start element of the comment with the end element.

Value:	<string> ro-attr(<attribute>) [<string>]
Initial:	see text
Inherited:	No

<string>

A unique identifier which links start and end element.

ro-attr(<attribute>)

A unique identifier from an attribute of the element which links start and end element.

[<string>]

An optional second identifier to link start and end properties. This should only be used if the unique identifier is not unique for all elements but only for certain elements.

- **Source:** Proprietary
- **More information:** [Comments \(p. 34\)](#)

-ro-comment-state

The initial state of the comment bubbles displayed by the viewer. This property only affects certain PDF viewers.

Value:	open closed
Initial:	closed
Inherited:	No

open

All comment bubbles will be opened and displayed when the document is opened in the PDF viewer.

closed

All comment bubbles will be closed when the document is opened in the PDF viewer.

- *Source:* Proprietary
- *More information:* [Comments \(p. 34\)](#)

-ro-comment-style

Specifies the style of the comment.

Value:	note highlight underline strikeout squiggly invisible
Initial:	note
Inherited:	No

note

Displays the comment as a note icon.

highlight

Highlights the background of the comment area in a certain color.

underline

Underlines the text of the comment area with a straight line.

strikeout

Strikes out the text of the comment area.

squiggly

Underlines the text of the comment area with a squiggly line.

invisible

Does not visualize the comment in any way.

- *Source:* Proprietary
- *More information:* [Comments \(p. 34\)](#)

-ro-comment-title

Specifies the title or author of the comment.

Value:	none <string> ro-attr(<attribute>)
Initial:	none
Inherited:	No

none

The comment receives no title.

<string>

Defines the title of the comment.

ro-attr(<attribute>)

Defines the content of the comment from an attribute of the element.

- *Source:* Proprietary
- *More information:* [Comments \(p. 34\)](#)

content

This property is used with the `:before` and `:after` pseudo-elements to generate content in a document.

Value:	<code>normal</code> <code>none</code> [<code><string></code> <code><counter></code> <code>attr(<identifier>)</code> <code><target-counter></code> <code><target-text></code> <code><named-string></code> <code><leader></code>]+ <code><running-element></code> <code>inherit</code>
Initial:	<code>normal</code>
Applies To:	<code>:before</code> and <code>:after</code> pseudo-elements
Inherited:	No

none

The pseudo-element is not generated.

normal

Computes to 'none' for the `:before` and `:after` pseudo-elements.

<string>

Text content.

<counter>

Counters may be specified with two different functions: `'counter()'` or `'counters()'`.

attr(<attribute>)

This function returns as a string the value of attribute `<attribute>` for the subject of the selector.

<target-counter>

Target counters may be specified with two different functions: `'target-counter()'` or `'target-counters()'`.

<target-text>

Target text may be specified with the function: `'target-text()'`.

<named-string>

Named strings may be specified with the function: `'string()'`. The string function has two arguments. The name of the named string as identifier and the location on the page (which is optional).

<leader>

Leaders may be specified with the function: `'leader()'`.

<running-element>

Running Elements may be specified with the function: `'element()'` from a position property. The element function has two arguments. The name of the running element as identifier and the location on the page (which is optional).

■ *Source:* CSS 2.1, CSS3

■ *More information:* [Generated Content \(p. 52\)](#), [Page Header & Footer \(p. 53\)](#), [Generated Content for Pages \(p. 57\)](#)

counter-increment

The 'counter-increment' property accepts one or more names of counters (identifiers), each one optionally followed by an integer. The integer indicates by how much the counter is incremented for every occurrence of the element. The default increment is 1. Zero and negative integers are allowed.

Value:	none [<identifier> <integer>?]+ inherit
Initial:	none
Inherited:	No

- *Source:* CSS 2.1
- *More information:* [Counters \(p. 53\)](#), [Page Counters \(p. 57\)](#)

counter-reset

The 'counter-reset' property contains a list of one or more names of counters, each one optionally followed by an integer. The integer gives the value that the counter is set to on each occurrence of the element. The default is 0.

Value:	none [<identifier> <integer>?]+ inherit
Initial:	none
Inherited:	No

- *Source:* CSS 2.1
- *More information:* [Counters \(p. 53\)](#), [Page Counters \(p. 57\)](#)

-ro-counter-set

The '-ro-counter-set' property contains a list of one or more names of counters, each one optionally followed by an integer. The integer gives the value that the counter is set to on each occurrence of the element. The default is 0. The difference to the 'counter-reset' property is, that '-ro-counter-set' does not create a new instance of a counter if an existing counter is present. This allows '-ro-counter-set' to reset an existing counter from anywhere inside the document.

Value:	none [<identifier> <integer>?]+ inherit
Initial:	none
Inherited:	No

- *Source:* Proprietary
- *More information:* [Page Counters \(p. 57\)](#)

-ro-crop-size

Specifies the size of the CropBox, one of the PDF page boxes.

Value:	none <length>{1,2} [<page-size> [portrait landscape]] media trim art
Initial:	none
Applies To:	page context
Inherited:	No

none

The element does not specify a CropBox.

media

The CropBox is specified with the same dimensions as the MediaBox.

trim

The CropBox is specified with the same dimensions as the TrimBox.

art

The CropBox is specified with the same dimensions as the ArtBox.

- **Source:** Proprietary
- **More information:** [Page Boxes \(p. 69\)](#)

cursor

This property specifies the type of cursor to be displayed for the pointing device.

Value:	<code>[[<uri> ,]* [auto crosshair default pointer move e-resize ne-resize nw-resize n-resize se-resize sw-resize s-resize w-resize text wait help progress]] inherit</code>
Initial:	auto
Inherited:	Yes

auto

The UA determines the cursor to display based on the current context.

crosshair

A simple crosshair (e.g., short line segments resembling a "+" sign).

default

The platform-dependent default cursor. Often rendered as an arrow.

pointer

The cursor is a pointer that indicates a link.

move

Indicates something is to be moved.

e-resize, ne-resize, nw-resize, n-resize, se-resize, sw-resize, s-resize, w-resize

Indicate that some edge is to be moved. For example, the 'se-resize' cursor is used when the movement starts from the south-east corner of the box.

text

Indicates text that may be selected. Often rendered as an I-beam.

wait

Indicates that the program is busy and the user should wait. Often rendered as a watch or hourglass.

progress

A progress indicator. The program is performing some processing, but is different from 'wait' in that the user may still interact with the program. Often rendered as a spinning beach ball, or an arrow with a watch or hourglass.

help

Help is available for the object under the cursor. Often rendered as a question mark or a balloon.

<uri>

The user agent retrieves the cursor from the resource designated by the URI. If the user agent cannot handle the first cursor of a list of cursors, it should attempt to handle the second, etc. If the user agent cannot handle any user-defined cursor, it must use the generic cursor at the end of the list. Intrinsic sizes for cursors are calculated as for background images, except that a UA-defined rectangle is used in place of the rectangle that establishes the coordinate system for the 'background-image' property. This UA-defined rectangle should be based on the size of a typical cursor on the UA's operating system. If the resulting cursor size does not fit within this rectangle, the UA may proportionally scale the resulting cursor down until it fits within the rectangle.

■ *Source:* CSS 2.1

direction

This property specifies the base writing direction of blocks and the direction of embeddings and overrides (see 'unicode-bidi') for the Unicode bidirectional algorithm. In addition, it specifies such things as the direction of table column layout, the direction of horizontal overflow, the position of an incomplete last line in a block in case of 'text-align: justify'.

Value:	ltr rtl inherit
Initial:	ltr
Applies To:	all elements (but see prose)
Inherited:	Yes

ltr

Left-to-right direction.

rtl

Right-to-left direction.

■ *Source:* CSS 2.1

■ *See also:* [unicode-bidi](#) (p. 163)

■ *More information:* [Right-to-Left](#) (p. 81)

display

The computed value is the same as the specified value, except for positioned and floating elements (see Relationships between 'display', 'position', and 'float') and for the root element. For the root element, the computed value is changed as described in the section on the relationships between 'display', 'position', and 'float'. Note that although the initial value of 'display' is 'inline', rules in the user agent's default style sheet may override this value. See the sample style sheet for HTML 4 in the appendix.

Value:	inline block list-item inline-block table inline-table table-row-group table-row table-cell none inherit
Initial:	inline
Inherited:	No

block

This value causes an element to generate a block box.

inline-block

This value causes an element to generate an inline-level block container. The inside of an inline-block is formatted as a block box, and the element itself is formatted as an atomic inline-level box.

inline

This value causes an element to generate one or more inline boxes.

list-item

This value causes an element (e.g., LI in HTML) to generate a principal block box and a marker box. For information about lists and examples of list formatting, please consult the section on lists.

none

This value causes an element to not appear in the formatting structure (i.e., in visual media the element generates no boxes and has no effect on layout). Descendant elements do not generate any boxes either; the element and its content are removed from the formatting structure entirely. This behavior cannot be overridden by setting the 'display' property on the descendants. Please note that a display of 'none' does not create an invisible box; it creates no box at all. CSS includes mechanisms that enable an element to generate boxes in the formatting structure that affect formatting but are not visible themselves. Please consult the section on visibility for details.

table, inline-table, table-row-group, table-column, table-column-group, table-header-group, table-footer-group, table-row, table-cell, and table-caption

These values cause an element to behave like a table element (subject to restrictions described in the chapter on tables).

■ Source: CSS 2.1

empty-cells

In the separated borders model, this property controls the rendering of borders and backgrounds around cells that have no visible content. Empty cells and cells with the 'visibility' property set to 'hidden' are considered to have no visible content.

Value:	show hide inherit
Initial:	show
Applies To:	'table-cell' elements
Inherited:	Yes

show

When this property has the value 'show', borders and backgrounds are drawn around/behind empty cells (like normal cells).

hide

A value of 'hide' means that no borders or backgrounds are drawn around/behind empty cells. Furthermore, if all the cells in a row have a value of 'hide' and have no visible content, then the row has zero height and there is vertical border-spacing on only one side of the row.

■ Source: CSS 2.1

float

This property specifies whether a box should float to the left, right, or not at all. It may be set for any element, but only applies to elements that generate boxes that are not absolutely positioned.

Value:	left right footnote none inherit
Initial:	none
Inherited:	No

left

The element generates a block box that is floated to the left. Content flows on the right side of the box, starting at the top (subject to the 'clear' property).

right

Similar to 'left', except the box is floated to the right, and content flows on the left side of the box, starting at the top.

none

The box is not floated.

- *Source:* CSS 2.1
- *See also:* [position](#) (p. 151)
- *More information:* [Footnotes](#) (p. 61)

-ro-flow-from

The 'flow-from' property makes a block container a region and associates it with a named flow.

Value:	none <identifier> inherit
Initial:	
Applies To:	Non-replaced block containers.
Inherited:	No

none

The block container is not a CSS Region.

<identifier>

The block container becomes a CSS Region, and is ordered in a region chain according to its document order.

- *Source:* CSS3, Experimental
- *More information:* [Region Layout](#) (p. 65)

-ro-flow-into

The 'flow-into' property can place an element or its contents into a named flow. Content that belongs to the same flow is laid out in regions associated with that flow. The 'flow-into' property neither affects the CSS cascade and inheritance nor the DOM position of an element or its contents. A named flow needs to be associated with one or more regions to be displayed.

Value:	none <identifier> [element content]?
Initial:	none
Applies To:	All elements, but not pseudo-elements such as ::first-line, ::first-letter, ::before or ::after.
Inherited:	No

none

The element is not moved to a named flow and normal CSS processing takes place.

<identifier>

If the keyword 'element' or neither keyword is present, the element is taken out of its parent's flow and placed into the named flow '<identifier>'. If the keyword 'content' is present, then only the element's contents is placed into the named flow. The values 'none', 'inherit', 'default', 'auto' and 'initial' are invalid flow names.

- *Source:* CSS3, Experimental
- *More information:* [Region Layout](#) (p. 65)

font

The 'font' property is, except as described below, a shorthand property for setting 'font-style', 'font-variant', 'font-weight', 'font-size', 'line-height' and 'font-family' at the same place in the style sheet. The syntax of this property is based on a traditional typographical shorthand notation to set multiple properties related to fonts. All font-related properties are first reset to their initial values, including those listed in the preceding paragraph. Then, those properties that are given explicit values in the 'font' shorthand are set to those values.

Value:	[['font-style' 'font-weight']? 'font-size' [/ 'line-height']? 'font-family'] inherit
Initial:	see individual properties
Inherited:	Yes

■ **Source:** CSS 2.1

■ **See also:** [font-family](#) (p. 132), [font-size](#) (p. 132), [font-style](#) (p. 133), [font-weight](#) (p. 133), [line-height](#) (p. 137)

font-family

The property value is a prioritized list of font family names and/or generic family names. Unlike most other CSS properties, component values are separated by a comma to indicate that they are alternatives.

Value:	[[<family-name> <generic-family>] [, <family-name> <generic-family>]*] inherit
Initial:	depends on user agent
Inherited:	Yes

<family-name>

The name of a font family of choice.

<generic-family>

The following generic families are defined: 'serif' (e.g., Times) 'sans-serif' (e.g., Helvetica) 'cursive' (e.g., Zapf-Chancery) 'fantasy' (e.g., Western) 'monospace' (e.g., Courier) Style sheet designers are encouraged to offer a generic font family as a last alternative. Generic font family names are keywords and must NOT be quoted.

■ **Source:** CSS 2.1

font-size

The font size corresponds to the em square, a concept used in typography. Note that certain glyphs may bleed outside their em squares.

Value:	<absolute-size> <relative-size> <length> <percentage> inherit
Initial:	medium
Inherited:	Yes

<absolute-size>

An <absolute-size> keyword is an index to a table of font sizes computed and kept by the UA. Possible values are: [xx-small | x-small | small | medium | large | x-large | xx-large]

<relative-size>

A <relative-size> keyword is interpreted relative to the table of font sizes and the font size of the parent element. Possible values are: [larger | smaller]. For example, if the parent element has a font size of 'medium', a value of 'larger' will make the font size of the current element be 'large'.

■ **Source:** CSS 2.1

font-style

The 'font-style' property selects between normal (sometimes referred to as "roman" or "upright"), italic and oblique faces within a font family.

Value:	normal italic oblique inherit
Initial:	normal
Inherited:	Yes

■ Source: CSS 2.1

font-variant

Another type of variation within a font family is the small-caps. In a small-caps font the lower case letters look similar to the uppercase ones, but in a smaller size and with slightly different proportions. The 'font-variant' property selects that font.

Value:	normal small-caps inherit
Initial:	normal
Inherited:	Yes

■ Source: CSS 2.1

font-weight

The 'font-weight' property selects the weight of the font. The values '100' to '900' form an ordered sequence, where each number indicates a weight that is at least as dark as its predecessor. The keyword 'normal' is synonymous with '400', and 'bold' is synonymous with '700'. The 'bolder' and 'lighter' values select font weights that are relative to the weight inherited from the parent.

Value:	normal bold bolder lighter 100 200 300 400 500 600 700 800 900 inherit
Initial:	normal
Inherited:	Yes

■ Source: CSS 2.1

-ro-formelement-name

Defines from which element or attribute in the document the names of the form elements are adopted to a generated PDF.

Value:	none <string> ro-attr(<attribute>)
Initial:	none
Applies To:	Form elements
Inherited:	No

■ Source: Proprietary

■ More information: [Tagged PDF \(p. 38\)](#)

height

This property specifies the content height of boxes. This property does not apply to non-replaced inline elements. Negative values for 'height' are illegal.

Value:	<length> <percentage> auto inherit
Initial:	auto
Applies To:	all elements but non-replaced inline elements, table columns, and column groups
Inherited:	No

<length>

Specifies the height of the content area using a length value.

<percentage>

Specifies a percentage height. The percentage is calculated with respect to the height of the generated box's containing block. If the height of the containing block is not specified explicitly (i.e., it depends on content height), and this element is not absolutely positioned, the value computes to 'auto'. A percentage height on the root element is relative to the initial containing block.

auto

The height depends on the values of other properties.

■ **Source:** CSS 2.1

height

This property allows the automatic resizing of form controls according to their content. If this property is set to auto, the form controls' height automatically adjusts according to its content.

Value:	auto none
Initial:	none
Applies To:	Form elements
Inherited:	No

auto

automatically adjusts the height of a form control if the contents' height exceeds the height defined for the form control.

■ **Source:** Proprietary

■ **More information:** [Automatic Resizing of Form Controls \(p. 79\)](#)

hyphenate-after

This property specifies the minimum number of characters in a hyphenated word after the hyphenation character. The 'auto' value means that the UA chooses a value that adapts to the current layout.

Value:	<integer> auto
Initial:	auto
Inherited:	Yes

■ **Source:** CSS3

■ **More information:** [Automatic Hyphenation \(p. 51\)](#)

hyphenate-before

This property specifies the minimum number of characters in a hyphenated word before the hyphenation character. The 'auto' value means that the UA chooses a value that adapts to the current layout.

Value:	<integer> auto
Initial:	auto
Inherited:	Yes

- *Source:* CSS3
- *More information:* [Automatic Hyphenation \(p. 51\)](#)

hyphenate-character

This property specifies a string that is shown when a hyphenate-break occurs. The 'auto' value means that the user agent should find an appropriate value.

Value:	<string> auto
Initial:	auto
Inherited:	Yes

- *Source:* CSS3
- *More information:* [Automatic Hyphenation \(p. 51\)](#)

hyphens

This property controls whether hyphenation is allowed to create more soft wrap opportunities within a line of text.

Value:	none manual auto
Initial:	manual
Inherited:	Yes

none

Words are not hyphenated, even if characters inside the word explicitly define hyphenation opportunities.

manual

Words are only hyphenated where there are characters inside the word that explicitly suggest hyphenation opportunities.

auto

Words may be broken at appropriate hyphenation points either as determined by hyphenation characters inside the word or as determined automatically by a language-appropriate hyphenation resource. Conditional hyphenation characters inside a word, if present, take priority over automatic resources when determining hyphenation opportunities within the word.

- *Source:* CSS3
- *More information:* [Automatic Hyphenation \(p. 51\)](#)

-ro-keywords

Sets the keywords in the metadata of the PDF document. Multiple values are concatenated to one string. (When applied to multiple elements the values are concatenated, separated by a comma.)

Value:	none [<string> content() ro-attr(<attribute>)]+
Initial:	none
Applies To:	all elements
Inherited:	No

none

Does not set a keywords.

<string>

Sets the specified string as keywords.

content()

Sets the keywords from the content of the element.

ro-attr(<attribute>)

Sets the keywords from the specified attribute of the element.

- *Source:* Proprietary
- *See also:* [-ro-author](#) (p. 102), [-ro-subject](#) (p. 157), [-ro-title](#) (p. 159)
- *More information:* [Metadata](#) (p. 37)

left

Like 'top', but specifies how far a box's left margin edge is offset to the right of the left edge of the box's containing block. For relatively positioned boxes, the offset is with respect to the left edge of the box itself.

Value:	<length> <percentage> auto inherit
Initial:	auto
Applies To:	positioned elements
Inherited:	No

<length>

The offset is a fixed distance from the reference edge. Negative values are allowed.

<percentage>

The offset is a percentage of the containing block's width. Negative values are allowed.

auto

For non-replaced elements, the effect of this value depends on which of related properties have the value 'auto' as well. For replaced elements, the effect of this value depends on the intrinsic dimensions of the replaced content.

- *Source:* CSS 2.1

letter-spacing

This property specifies spacing behavior between text characters.

Value:	normal <length> inherit
Initial:	normal
Inherited:	Yes

normal

The spacing is the normal spacing for the current font. This value allows the user agent to alter the space between characters in order to justify text.

<length>

This value indicates inter-character space in addition to the default space between characters.

■ *Source:* CSS 2.1

line-height

On a block container element whose content is composed of inline-level elements, 'line-height' specifies the minimal height of line boxes within the element. The minimum height consists of a minimum height above the baseline and a minimum depth below it, exactly as if each line box starts with a zero-width inline box with the element's font and line height properties.

Value:	normal <number> <length> <percentage> inherit
Initial:	normal
Inherited:	Yes

normal

Tells user agents to set the used value to a "reasonable" value based on the font of the element. The value has the same meaning as <number>. The computed value is 'normal'.

<length>

The specified length is used in the calculation of the line box height. Negative values are illegal.

<number>

The used value of the property is this number multiplied by the element's font size. Negative values are illegal. The computed value is the same as the specified value.

<percentage>

The computed value of the property is this percentage multiplied by the element's computed font size. Negative values are illegal.

■ *Source:* CSS 2.1

-ro-link

This property allows to define hyperlinks via style. Multiple values are concatenated to one URL.

Value:	none [<string> ro-attr(<attribute>)]+
Initial:	none
Applies To:	all elements
Inherited:	No

none

The element is not a hyperlink.

<string>

The element is a hyperlink to the URL the <string> contains.

ro-attr(<attribute>)

The element is a hyperlink to the URL the <string> resolved from the ro-attr() function.

- *Source:* Proprietary
- *See also:* [-ro-link-area](#) (p. 138)
- *More information:* [Links](#) (p. 34)

-ro-link-area

This property can be used to specify how the 'clickable' areas of a link are determined.

Value:	content block content-block
Initial:	content
Inherited:	No

content

For block elements there is one clickable area for each piece of content (text, image or empty block). For inline elements there is one clickable area for each part.

block

For block elements there is one clickable area for the whole block. For inline elements there is one clickable area for the bounding rectangle of all parts.

content-block

For block elements there is one clickable area for the bounding rectangle of the content. For inline elements there is one clickable area for the bounding rectangle of all parts.

- *Source:* Proprietary
- *See also:* [-ro-link](#) (p. 137)
- *More information:* [Links](#) (p. 34)

list-style

The 'list-style' property is a shorthand notation for setting the three properties 'list-style-type', 'list-style-image', and 'list-style-position' at the same place in the style sheet.

Value:	['list-style-type' 'list-style-position' 'list-style-image'] inherit
Initial:	see individual properties
Applies To:	elements with 'display: list-item'
Inherited:	Yes

- *Source:* CSS 2.1
- *See also:* [list-style-image](#) (p. 139), [list-style-position](#) (p. 139), [list-style-type](#) (p. 139)

list-style-image

This property sets the image that will be used as the list item marker. When the image is available, it will replace the marker set with the 'list-style-type' marker.

Value:	<uri> none inherit
Initial:	none
Applies To:	elements with 'display: list-item'
Inherited:	Yes

■ Source: CSS 2.1

list-style-position

This property specifies the position of the marker box with respect to the principal block box.

Value:	inside outside inherit
Initial:	outside
Applies To:	elements with 'display: list-item'
Inherited:	Yes

outside

The marker box is outside the principal block box. The position of the list-item marker adjacent to floats is undefined in CSS 2.1. CSS 2.1 does not specify the precise location of the marker box or its position in the painting order, but does require that for list items whose 'direction' property is 'ltr' the marker box be on the left side of the content and for elements whose 'direction' property is 'rtl' the marker box be on the right side of the content. The marker box is fixed with respect to the principal block box's border and does not scroll with the principal block box's content.

inside

The marker box is placed as the first inline box in the principal block box, before the element's content and before any :before pseudo-elements.

■ Source: CSS 2.1

list-style-type

This property specifies appearance of the list item marker if 'list-style-image' has the value 'none' or if the image pointed to by the URI cannot be displayed. The value 'none' specifies no marker, otherwise there are three types of marker: glyphs, numbering systems, and alphabetic systems. Glyphs are specified with disc, circle, and square.

Value:	<counter-style> none inherit
Initial:	disc
Applies To:	elements with 'display: list-item'
Inherited:	Yes

<counter-style>

The list item marker is formatted according to the given counter style. Unordered types are: box, check, circle, diamond, disc, dash, square. Ordered types are for example lower-alpha, lower-greek or upper-roman.

■ Source: CSS 2.1, CSS3

■ More information: [List Style Types \(p. 97\)](#)

-ro-listitem-value

The name of the property to determine the start number of a list item. The content contains the number a numbered itemized list starts with.

Value:	<integer>
Initial:	1
Applies To:	list-item
Inherited:	No

<integer>

The starting number of the current list item.

- Source: Proprietary

margin

The 'margin' property is a shorthand property for setting 'margin-top', 'margin-right', 'margin-bottom', and 'margin-left' at the same place in the style sheet. If there is only one component value, it applies to all sides. If there are two values, the top and bottom margins are set to the first value and the right and left margins are set to the second. If there are three values, the top is set to the first value, the left and right are set to the second, and the bottom is set to the third. If there are four values, they apply to the top, right, bottom, and left, respectively.

Value:	<margin-width>{1,4} inherit
Initial:	see individual properties
Applies To:	all elements except elements with table display types other than table-caption, table and inline-table
Inherited:	No

- Source: CSS 2.1
- See also: [margin-*](#) (p. 140)

margin-top**margin-right****margin-bottom****margin-left**

These properties set the top, right, bottom, and left margin of a box.

Value:	<margin-width> inherit
Initial:	0
Applies To:	all elements except elements with table display types other than table-caption, table and inline-table
Inherited:	No

- Source: CSS 2.1

-ro-marks

Adds the specified printer marks inside the page's MediaBox.

Value:	none [trim bleed registration]
Initial:	none
Applies To:	page context
Inherited:	No

none

No marks are added to the page.

trim

Adds trim line marks to the four corners of the page.

bleed

Adds bleed line marks to the four corners of the page.

registration

Adds registration marks to the four sides of the page.

- *Source:* Proprietary
- *See also:* [-ro-marks-color](#) (p. 141), [-ro-marks-width](#) (p. 141), [-ro-media-size](#) (p. 143)
- *More information:* [Printer Marks](#) (p. 70)

-ro-marks-color

Sets the color of the printer marks.

Value:	<color>
Initial:	The color "black".
Applies To:	page context
Inherited:	Yes

- *Source:* Proprietary
- *See also:* [-ro-marks](#) (p. 141)
- *More information:* [Printer Marks](#) (p. 70)

-ro-marks-width

Sets the width of the printer marks.

Value:	none <length>
Initial:	A length of "0.5pt"
Applies To:	page context
Inherited:	No

- *Source:* Proprietary
- *See also:* [-ro-marks](#) (p. 141)
- *More information:* [Printer Marks](#) (p. 70)

max-height

This property allows authors to limit box heights.

Value:	<length> <percentage> none inherit
Initial:	none
Applies To:	all elements but non-replaced inline elements, table columns, and column groups
Inherited:	No

<length>

Specifies a fixed maximum computed height.

<percentage>

Specifies a percentage for determining the used value. The percentage is calculated with respect to the height of the generated box's containing block. If the height of the containing block is not specified explicitly (i.e., it depends on content height), and this element is not absolutely positioned, the percentage value is treated as 'none'.

none

No limit on the height of the box.

- *Source:* CSS 2.1
- *See also:* [min-height](#) (p. 143)

max-width

This property allows authors to constrain content widths to a maximum.

Value:	<length> <percentage> none inherit
Initial:	none
Applies To:	all elements but non-replaced inline elements, table rows, and row groups
Inherited:	No

<length>

Specifies a fixed maximum used width.

<percentage>

Specifies a percentage for determining the used value. The percentage is calculated with respect to the width of the generated box's containing block. If the containing block's width is negative, the used value is zero. If the containing block's width depends on this element's width, then the resulting layout is undefined in CSS 2.1.

none

No limit on the width of the box.

- *Source:* CSS 2.1
- *See also:* [min-width](#) (p. 143)

`-ro-media-size`

Specifies the size of the MediaBox, one of the PDF page boxes. The MediaBox defines an oversized paper sheet that allows to add a bleed area, marks and color bars around the normal page content. This property works the same way as the `size` property does.

Value:	<code>none</code> <code><length>{1,2}</code> <code>auto</code> [<code><page-size></code> [<code>portrait</code> <code>landscape</code>]]
Initial:	<code>none</code>
Applies To:	page context
Inherited:	No

- **Source:** Proprietary
- **See also:** `-ro-bleed-width` (p. 107), `-ro-colorbar-*` (p. 116), `-ro-marks` (p. 141), `size` (p. 155)
- **More information:** [Page Boxes](#) (p. 69)

`min-height`

This property allows authors to set a minimum box height.

Value:	<code><length></code> <code><percentage></code> <code>inherit</code>
Initial:	<code>0</code>
Applies To:	all elements but non-replaced inline elements, table columns, and column groups
Inherited:	No

`<length>`

Specifies a fixed minimum computed height.

`<percentage>`

Specifies a percentage for determining the used value. The percentage is calculated with respect to the height of the generated box's containing block. If the height of the containing block is not specified explicitly (i.e., it depends on content height), and this element is not absolutely positioned, the percentage value is treated as '0'.

- **Source:** CSS 2.1
- **See also:** `max-height` (p. 142)

`min-width`

This property allows authors to constrain content widths to a minimum value.

Value:	<code><length></code> <code><percentage></code> <code>inherit</code>
Initial:	<code>0</code>
Applies To:	all elements but non-replaced inline elements, table rows, and row groups
Inherited:	No

`<length>`

Specifies a fixed minimum used width.

`<percentage>`

Specifies a percentage for determining the used value. The percentage is calculated with respect to the width of the generated box's containing block. If the containing block's width is negative, the used value is zero. If the containing block's width depends on this element's width, then the resulting layout is undefined in CSS 2.1.

- **Source:** CSS 2.1
- **See also:** `max-width` (p. 142)

orphans

The 'orphans' property specifies the minimum number of lines in a block container that must be left at the bottom of a page. Only positive values are allowed.

Value:	<integer> inherit
Initial:	2
Applies To:	block container elements
Inherited:	Yes

- Source: CSS 2.1
- More information: [Widows & Orphans \(p. 51\)](#)

outline

The 'outline' property is a shorthand property, and sets all three of 'outline-style', 'outline-width', and 'outline-color'.

Value:	['outline-color' 'outline-style' 'outline-width'] inherit
Initial:	see individual properties
Inherited:	No

- Source: CSS 2.1
- See also: [outline-color \(p. 144\)](#), [outline-style \(p. 144\)](#), [outline-width \(p. 145\)](#)

outline-color

The 'outline-color' accepts all colors, as well as the keyword 'invert'. 'Invert' is expected to perform a color inversion on the pixels on the screen. This is a common trick to ensure the focus border is visible, regardless of color background.

Value:	<color> invert inherit
Initial:	invert
Inherited:	No

- Source: CSS 2.1
- More information: [Color Keywords \(p. 92\)](#)

outline-style

The 'outline-style' property accepts the same values as 'border-style', except that 'hidden' is not a legal outline style.

Value:	<border-style> inherit
Initial:	none
Inherited:	No

- Source: CSS 2.1

outline-width

The 'outline-width' property accepts the same values as 'border-width'.

Value:	<border-width> inherit
Initial:	medium
Inherited:	No

■ *Source:* CSS 2.1

overflow

This property specifies whether content of a block container element is clipped when it overflows the element's box. It affects the clipping of all of the element's content except any descendant elements (and their respective content and descendants) whose containing block is the viewport or an ancestor of the element.

Value:	visible hidden
Initial:	visible
Applies To:	block containers
Inherited:	No

visible

This value indicates that content is not clipped, i.e., it may be rendered outside the block box.

hidden

This value indicates that the content is clipped and that no scrolling user interface should be provided to view the content outside the clipping region.

■ *Source:* CSS 2.1

overflow-wrap

word-wrap

This property specifies whether the UA may arbitrarily break within a word to prevent overflow when an otherwise-unbreakable string is too long to fit within the line box. It only has an effect when 'white-space' allows wrapping.

Value:	normal break-word
Initial:	normal
Inherited:	Yes

normal

Lines may break only at allowed break points.

break-word

An unbreakable "word" may be broken at an arbitrary point if there are no otherwise-acceptable break points in the line. Shaping characters are still shaped as if the word were not broken, and grapheme clusters must together stay as one unit. No hyphenation character is inserted at the break point.

■ *Source:* CSS3

padding

The 'padding' property is a shorthand property for setting 'padding-top', 'padding-right', 'padding-bottom', and 'padding-left' at the same place in the style sheet.

Value:	<padding-width>{1,4} inherit
Initial:	see individual properties
Applies To:	all elements except table-row-group, table-header-group, table-footer-group, table-row, table-column-group and table-column
Inherited:	No

- *Source:* CSS 2.1
- *See also:* [padding-*](#) (p. 146)

padding-top padding-right padding-bottom padding-left

These properties set the top, right, bottom, and left padding of a box.

Value:	<padding-width> inherit
Initial:	0
Applies To:	all elements except table-row-group, table-header-group, table-footer-group, table-row, table-column-group and table-column
Inherited:	No

- *Source:* CSS 2.1

page

This property is used to specify a particular type of page (called a named page) on which an element must be displayed. If necessary, a forced page break is introduced and a new page generated of the specified type.

Value:	auto <identifier>
Initial:	auto
Applies To:	boxes that create class 1 break points
Inherited:	No

<identifier>

The name of a particular page type. Page type names are case-sensitive identifiers.

- *Source:* CSS3
- *More information:* [Named Pages](#) (p. 50)

page-break-before

page-break-after

Shorthand for the 'break-before' and 'break-after' properties.

Value:	auto always avoid left right inherit
Initial:	auto
Applies To:	block-level elements
Inherited:	No

auto

Neither force nor forbid a page break before (after) the generated box.

always

Always force a page break before (after) the generated box.

avoid

Avoid a page break before (after) the generated box.

left

Force one or two page breaks before (after) the generated box so that the next page is formatted as a left page.

right

Force one or two page breaks before (after) the generated box so that the next page is formatted as a right page.

- **Source:** CSS 2.1
- **See also:** [break-before](#), [break-after](#) (p. 112)

page-break-inside

Shorthand for the 'break-inside' property.

Value:	avoid auto inherit
Initial:	auto
Applies To:	block-level elements
Inherited:	No

auto

Neither force nor forbid a page break inside the generated box.

avoid

Avoid a page break inside the generated box.

- **Source:** CSS 2.1
- **See also:** [break-inside](#) (p. 113)

-ro-passdown-styles

The `-ro-passdown-styles` property controls how style is passed down from an embedding document to an embedded document. Counters or Named Strings from the embedding document will remain available to the embedded document, independent of the value set

Value:	all stylesheets-only none
Initial:	all
Applies To:	iframe, @page
Inherited:	No

all

Default value, all inheritable inline styles and all style sheets passed down to the embedded document.

stylesheets-only

Styles that have been set via the `style`-attribute (inline styles) are ignored, but the style sheets of the embedding document are passed down.

none

Styles are not passed down to the embedded document.

- Source: Proprietary
- More information: [iframes \(p. 26\)](#), [Running Documents0 \(p. 57\)](#)

-ro-pdf-attachment-description

The description of the attachment. If this is not specified the name is used.

Value:	none <string> ro-attr(<attribute>)
Initial:	none
Inherited:	No

- Source: Proprietary
- More information: [Attachments \(p. 46\)](#)

-ro-pdf-attachment-location

Specifies whether the attachment is related to the area of the element.

Value:	element document
Initial:	element
Inherited:	No

element

The attachment is related to the area of the element. Viewers may show a marker near that area.

document

The file is attached to the document with no relation to the element.

- Source: Proprietary
- More information: [Attachments \(p. 46\)](#)

`-ro-pdf-attachment-name`

The file name associated with the attachment. It is recommended to specify the correct file extension. If this is not specified the name is derived from the URL.

Value:	none <string> ro-attr(<attribute>)
Initial:	none
Inherited:	No

- *Source:* Proprietary
- *More information:* [Attachments \(p. 46\)](#)

`-ro-pdf-attachment-url`

A URL pointing to the file to be embedded. This URL can be relative. Specifying "#" will embed the source document.

Value:	none <url> ro-attr(<attribute>)
Initial:	none
Inherited:	No

- *Source:* Proprietary
- *More information:* [Attachments \(p. 46\)](#)

`-ro-pdf-bookmark-level`

Using this property, one can structure the specified elements within the bookmark view of the PDF viewer. The elements are ordered in ascending order. The element with the lowest bookmark level is on top of the bookmark hierarchy (similar to HTML headlines).

Value:	none <integer>
Initial:	none
Inherited:	No

- *Source:* Proprietary
- *More information:* [Bookmarks \(p. 33\)](#)

`-ro-pdf-format`

This property converts form elements to interactive PDF forms.

Value:	none pdf
Initial:	none
Applies To:	Form elements
Inherited:	No

none

The form element is not converted.

pdf

The form element is converted to an AcroForm.

- **Source:** Proprietary
- **More information:** [Interactive PDF Forms \(p. 38\)](#)

-ro-pdf-overprint**-ro-pdf-overprint-content**

Using the properties `-ro-pdf-overprint` and `-ro-pdf-overprint-content` you can specify the overprint properties of elements and their content to either `none` (default), `mode0` or `mode1` (zero overprint mode). `-ro-pdf-overprint` affects the entire element, while `-ro-pdf-overprint-content` only affects the content of the element (not its borders and backgrounds). In both cases the children of the element are affected entirely, unless overprint styles are applied to them as well.

Value:	none mode0 mode1
Initial:	none
Inherited:	No

none

Disables overprinting. Painting a new color, no matter in which color space, causes unspecified colorants to be erased at the corresponding positions. This means that in any area, only the color that was painted last is visible.

mode0

Standard overprint mode, also known as "OPM 0". In this mode source color component values replace values that have been previously painted for the corresponding device colorants, regardless what the new values are.

mode1

Illustrator overprint mode, also known as "OPM 1" or "nonzero overprint mode". When the overprint mode is 1, tint values of 0.0 for source color components do not change the corresponding components of previously painted colors.

- **Source:** Proprietary
- **More information:** [Overprinting \(p. 45\)](#)

-ro-pdf-tag-type

This property makes it possible to map an element to a PDF tag type.

Value:	none <string>
Initial:	none
Inherited:	No

none

The element does not map to a PDF tag.

<string>

The name of the PDF tag to map the element to.

- **Source:** Proprietary
- **More information:** [Tagged PDF \(p. 38\)](#)

position

The 'position' and 'float' properties determine which of the positioning algorithms is used to calculate the position of a box.

Value:	static relative absolute fixed running(<identifier>) inherit
Initial:	static
Inherited:	No

running(<identifier>)

Moves the element out of the normal flow and into a page margin box as a running header or footer. The page margin box needs to specify the element function with the same <identifier> used for the running element to display it.

- *Source:* CSS 2.1, CSS3
- *See also:* [float](#) (p. 130)
- *More information:* [Running Elements](#) (p. 54)

-ro-qr-code-error-correction-level

Sets the error correction level of the QR code.

Value:	L M Q H
Initial:	L
Applies To:	QR Code elements
Inherited:	No

L

Low level error correction. Up to 7% damaged data can be restored.

M

Medium level error correction. Up to 15% damaged data can be restored.

Q

Quartile level error correction. Up to 25% damaged data can be restored.

H

High level error correction. Up to 30% damaged data can be restored.

- *Source:* Proprietary
- *More information:* [QR Code](#) (p. 25)

-ro-qr-code-forced-colors

Defines whether the colors of the QR code are black and white or based on the text color and the background.

Value:	normal none
Initial:	normal
Applies To:	QR Code elements
Inherited:	No

normal

QR code is black on white.

none

Instead of black, the value of the CSS property color is used to paint the squares. The background is visible instead of the white squares.

- Source: Proprietary
- More information: [QR Code \(p. 25\)](#)

-ro-qr-code-quality

By default, The QR code is built from multiple squares. This method is fast and looks correct in print. However, in PDF viewers on screen the edges of neighboring squares may be visible.

Value:	normal high
Initial:	normal
Applies To:	QR Code elements
Inherited:	No

normal

The QR code is built from multiple squares.

high

The squares are combined into one object, ensuring a seamless look, at the cost of performance.

- Source: Proprietary
- More information: [QR Code \(p. 25\)](#)

-ro-qr-code-quietzone-size

Sets the size of the quiet (empty) zone around the QR code in modules (QR code "square" widths).

Value:	<integer>
Initial:	1
Applies To:	QR Code elements
Inherited:	No

<integer>

Possible values are 0 (no quiet zone) and positive integers.

- Source: Proprietary
- More information: [QR Code \(p. 25\)](#)

-ro-radio-button-element-group

Defines from which element or attribute in the document the names of the radio button groups are adopted to a generated PDF.

Value:	none <string> ro-attr(<attribute>)
Initial:	none
Applies To:	Form elements
Inherited:	No

- Source: Proprietary
- More information: [Tagged PDF \(p. 38\)](#)

-ro-rasterization

This property configures in which cases SVGs and Canvas elements should be rasterized. It may disable some functionalities of those elements to avoid that. (Canvas shadows are converted into separate images, not affecting other parts of the Canvas, for both 'fallback' and 'avoid')

Value:	fallback avoid always
Initial:	fallback
Applies To:	SVG and Canvas elements
Inherited:	No

fallback

The SVG or Canvas is only rasterized when it uses features that are not supported by PDF vector graphics: masks, filters or non-default composites for SVG; non-default composites and ImageData access for Canvas.

avoid

Avoids rasterization of the entire SVG or Canvas by disabling functionality that is not supported by PDF vector graphics.

always

Rasterizes the Canvas in any case. (does not apply to SVG)

- **Source:** Proprietary
- **More information:** [SVG \(p. 24\)](#), [Canvas Element \(p. 28\)](#)

-ro-rasterization-supersampling

This property configures the resolution of the rasterization of SVGs and Canvas elements. Higher resolution factors increase the quality of the image, but also increase the conversion time and the size of the output documents.

Value:	<integer>
Initial:	2
Applies To:	SVG and Canvas elements
Inherited:	No

<integer>

The resolution of the rasterization is 96dpi multiplied by this factor. For example, a value of 2 means 192dpi. Accepted values are all positive integers, however canvas will clip values larger than 4.

- **Source:** Proprietary
- **More information:** [SVG \(p. 24\)](#), [Canvas Element \(p. 28\)](#)

-ro-replacedelement

Turns an element into a so called 'replaced element' that displays an image or other external or embedded content.

Value:	none image barcode qrcode embedded-svg embedded-mathml
Initial:	none
Inherited:	No

image

Creates an image replaced element. Used in combination with -ro-source.

barcode

Creates a barcode replaced element from embedded Barcode XML content.

qrcode

Creates an QR code replaced element. The QR code is read from an existing "href" attribute or the text content of the element.

embedded-svg

Creates an SVG replaced element from embedded SVG content.

embedded-mathml

Creates a MathML replaced element from embedded MathML content.

- *Source:* Proprietary
- *See also:* [-ro-source](#) (p. 156)
- *More information:* [Compound Formats](#) (p. 23)

right

Like 'top', but specifies how far a box's right margin edge is offset to the left of the right edge of the box's containing block. For relatively positioned boxes, the offset is with respect to the right edge of the box itself.

Value:	<length> <percentage> auto inherit
Initial:	auto
Applies To:	positioned elements
Inherited:	No

<length>

The offset is a fixed distance from the reference edge. Negative values are allowed.

<percentage>

The offset is a percentage of the containing block's width. Negative values are allowed.

auto

For non-replaced elements, the effect of this value depends on which of related properties have the value 'auto' as well. For replaced elements, the effect of this value depends on the intrinsic dimensions of the replaced content.

- *Source:* CSS 2.1

-ro-rounding-mode

Specifies the method that is used for rounding lengths to full pixels.

Value:	round floor
Initial:	round
Applies To:	root element
Inherited:	No

round

Use normal rounding function: if the value is smaller than .5 it is rounded down else it is rounded up.

floor

Always round down.

- *Source:* Proprietary

–ro–rowspan

The property to determine the row span of a cell. The content contains the number of rows spanned by this cell.

Value:	<integer>
Initial:	1
Applies To:	table-cell elements
Inherited:	No

■ *Source:* Proprietary

–ro–scale–content

This property enables shrink-to-fit functionality. This functionality scales down entire pages, which can cause gaps at the bottom of pages.

Value:	<length> <percentage> auto none
Initial:	none
Applies To:	page context
Inherited:	No

<length> | <percentage>

A percental value which is treated as a scaling factor for the content.

auto

Enables the automatic scaling of the content to fit the size of the page.

■ *Source:* Proprietary

■ *More information:* [Shrink-to-Fit \(p. 74\)](#)

size

This property specifies the target size and orientation of the page box's containing block. In the general case, where one page box is rendered onto one page sheet, the 'size' property also indicates the size of the destination page sheet.

Value:	auto <length>{1,2} [<page-size> [portrait landscape]]
Initial:	auto
Applies To:	page context
Inherited:	No

auto

The page box will be set to a size and orientation chosen by the UA. In the usual case, the page box size and orientation is chosen to match the target media sheet.

landscape

Specifies that the page's content be printed in landscape orientation. The longer sides of the page box are horizontal. If a '<page-size>' is not specified, the size of the page sheet is chosen by the UA.

portrait

Specifies that the page's content be printed in portrait orientation. The shorter sides of the page box are horizontal. If a '<page-size>' is not specified, the size of the page sheet is chosen by the UA.

<length>

The page box will be set to the given absolute dimension(s). If only one length value is specified, it sets both the width and height of the page box (i.e., the box is a square). If two length values are specified, the first establishes the page box width, and the second the page box height. Values in units of 'em' and 'ex' refer to the page context's font. Negative lengths are illegal.

<page-size>

A page size can be specified using one of the following media names. This is the equivalent of specifying the '<page-size>' using length values. The definition of the media names comes from Media Standardized Names. A5, A4, A3, B5, B4, letter, legal, ledger

- Source: CSS3
- More information: [Supported Page Sizes \(p. 90\)](#), [Page Boxes \(p. 69\)](#)

-ro-source

Specifies the URL of an image. Used in combination with -ro-replacdelement.

Value:	<string> ro-attr(<attribute>)
Initial:	
Inherited:	No

- Source: Proprietary
- See also: [-ro-replacdelement \(p. 153\)](#)
- More information: [Images \(p. 23\)](#)

string-set

The 'string-set' property accepts a comma-separated list of named strings. Each named string is followed by a content list that specifies which text to copy into the named string. Whenever an element with value of 'string-set' different from 'none' is encountered, the named strings are assigned their respective value.

Value:	[[<identifier> <content-list>] [, <identifier> <content-list>]*] none
Initial:	none
Inherited:	No

<string>

a string, e.g. "foo"

<counter>

counter() or counters() function

<content>

the 'content()' function returns the content of elements and pseudo-elements.

- Source: CSS3
- More information: [Named Strings \(p. 58\)](#)

-ro-subject

Sets the subject in the metadata of the PDF document. Multiple values are concatenated to one string. (When applied to multiple elements the values are concatenated, separated by a comma.)

Value:	none [<string> content() ro-attr(<attribute>)]+
Initial:	none
Applies To:	all elements
Inherited:	No

none

Does not set a subject.

<string>

Sets the specified string as subject.

content()

Sets the subject from the content of the element.

ro-attr(<attribute>)

Sets the subject from the specified attribute of the element.

- *Source:* Proprietary
- *See also:* [-ro-author](#) (p. 102), [-ro-keywords](#) (p. 136), [-ro-title](#) (p. 159)
- *More information:* [Metadata](#) (p. 37)

table-layout

The 'table-layout' property controls the algorithm used to lay out the table cells, rows, and columns.

Value:	auto fixed inherit
Initial:	auto
Applies To:	'table' and 'inline-table' elements
Inherited:	No

fixed

Use the fixed table layout algorithm

auto

Use any automatic table layout algorithm

- *Source:* CSS 2.1

text-align

This property describes how the inline-level content of a block is aligned along the inline axis if the content does not completely fill the line box.

Value:	start end left right center justify inherit
Initial:	start
Applies To:	block containers
Inherited:	Yes

start

Inline-level content is aligned to the start edge of the line box.

end

Inline-level content is aligned to the end edge of the line box.

left

Inline-level content is aligned to the line left edge of the line box.

right

Inline-level content is aligned to the line right edge of the line box.

center

Inline-level content is centered within the line box.

justify

Inline-level content is justified within the line box, except the last one.

- Source: CSS 2.1
- See also: `-ro-text-align-last` (p. 158)

`-ro-text-align-last`

This property describes how the last line of a block or a line right before a forced line break is aligned when 'text-align' is 'justify'. If a line is also the first line of the block or the first line after a forced line break, then 'text-align-last' takes precedence over 'text-align'.

Value:	auto start end left right center justify
Initial:	
Applies To:	block containers
Inherited:	Yes

- Source: CSS3, Experimental
- See also: `text-align` (p. 157)

`text-decoration`

This property describes decorations that are added to the text of an element using the element's color.

Value:	[underline line-through]
Initial:	none
Inherited:	No

none

Produces no text decoration.

underline

Each line of text is underlined.

line-through

Each line of text has a line through the middle.

- Source: CSS 2.1

text-indent

This property specifies the indentation of the first line of text in a block container.

Value:	<length> <percentage> inherit
Initial:	0
Applies To:	block containers
Inherited:	Yes

<length>

The indentation is a fixed length.

<percentage>

The indentation is a percentage of the containing block width.

■ Source: CSS 2.1

text-transform

This property controls capitalization effects of an element's text.

Value:	capitalize uppercase lowercase none inherit
Initial:	none
Inherited:	Yes

capitalize

Puts the first character of each word in uppercase; other characters are unaffected.

uppercase

Puts all characters of each word in uppercase.

lowercase

Puts all characters of each word in lowercase.

none

No capitalization effects.

■ Source: CSS 2.1

-ro-title

Sets the title in the metadata of the PDF document. Multiple values are concatenated to one string. (When applied to multiple elements the values are concatenated, separated by a comma.)

Value:	none [<string> content() ro-attr(<attribute>)]+
Initial:	none
Applies To:	all elements
Inherited:	No

none

Does not set a title.

<string>

Sets the specified string as title.

content()

Sets the title from the content of the element.

ro-attr(<attribute>)

Sets the title from the specified attribute of the element.

- **Source:** Proprietary
- **See also:** [-ro-author](#) (p. 102), [-ro-keywords](#) (p. 136), [-ro-subject](#) (p. 157)
- **More information:** [Metadata](#) (p. 37)

top

This property specifies how far an absolutely positioned box's top margin edge is offset below the top edge of the box's containing block. For relatively positioned boxes, the offset is with respect to the top edges of the box itself (i.e., the box is given a position in the normal flow, then offset from that position according to these properties).

Value:	<length> <percentage> auto inherit
Initial:	auto
Applies To:	positioned elements
Inherited:	No

<length>

The offset is a fixed distance from the reference edge. Negative values are allowed.

The offset is a fixed distance from the reference edge. Negative values are allowed. <percentage>

The offset is a percentage of the containing block's height. Negative values are allowed.

auto

For non-replaced elements, the effect of this value depends on which of related properties have the value 'auto' as well. For replaced elements, the effect of this value depends on the intrinsic dimensions of the replaced content.

- **Source:** CSS 2.1

transform

This property contains a list of transform functions. The final transformation value for a coordinate system is obtained by converting each function in the list to its corresponding matrix, then multiplying the matrices. A list of supported <transform-functions> can be found below.

Value:	none <transform-function>+
Initial:	none
Applies To:	transformable elements
Inherited:	No

matrix(<number>[, <number>]{5,5})

specifies a 2D transformation in the form of a transformation matrix of the six values a-f.

translate(<translation-value> [, <translation-value>]?)

specifies a 2D translation by the vector [tx, ty], where tx is the first translation-value parameter and ty is the optional second translation-value parameter. If <ty> is not provided, ty has zero as a value.

translateX(<translation-value>)

specifies a translation by the given amount in the X direction.

translateY(<translation-value>)

specifies a translation by the given amount in the Y direction.

scale(<number> [, <number>]?)

specifies a 2D scale operation by the [sx,sy] scaling vector described by the 2 parameters. If the second parameter is not provided, it takes a value equal to the first. For example, scale(1, 1) would leave an element unchanged, while scale(2, 2) would cause it to appear twice as long in both the X and Y axes, or four times its typical geometric size.

scaleX(<number>)

specifies a 2D scale operation using the [sx,1] scaling vector, where sx is given as the parameter.

scaleY(<number>)

specifies a 2D scale operation using the [1,sy] scaling vector, where sy is given as the parameter.

rotate(<angle>)

specifies a 2D rotation by the angle specified in the parameter about the origin of the element, as defined by the transform-origin property. For example, rotate(90deg) would cause elements to appear rotated one-quarter of a turn in the clockwise direction.

skew(<angle> [, <angle>]?)

specifies a 2D skew by [ax,ay] for X and Y. If the second parameter is not provided, it has a zero value.

skewX(<angle>)

specifies a 2D skew transformation along the X axis by the given angle.

skewY(<angle>)

specifies a 2D skew transformation along the Y axis by the given angle.

■ Source: CSS3

-ro-transform

Deprecated! This property contains a list of transform functions. The final transformation value for a coordinate system is obtained by converting each function in the list to its corresponding matrix, then multiplying the matrices.

Value:	none <transform-function>+
Initial:	none
Inherited:	No

■ Source: CSS3, Experimental

■ **Deprecated!** Use **transform** (p. 160) instead.

transform-origin

This property defines the point of origin of transformations. If only one value is specified, the second value is assumed to be center.

Value:	[left center right top bottom <percentage> <length>] [left center right <percentage> <length>] [top center bottom <percentage> <length>]
Initial:	50% 50%
Applies To:	transformable elements
Inherited:	No

<percentage>

A percentage for the horizontal offset is relative to the width of the bounding box. A percentage for the vertical offset is relative to height of the bounding box. The value for the horizontal and vertical offset represent an offset from the top left corner of the bounding box.

<length>

A length value gives a fixed length as the offset. The value for the horizontal and vertical offset represent an offset from the top left corner of the bounding box.

top

Computes to 0% for the vertical position.

right

Computes to 100% for the horizontal position.

bottom

Computes to 100% for the vertical position.

left

Computes to 0% for the horizontal position.

center

Computes to 50% (left 50%) for the horizontal position if the horizontal position is not otherwise specified, or 50% (top 50%) for the vertical position if it is.

■ Source: CSS3

-ro-transform-origin

Deprecated! This property defines the point of origin of transformations. If only one value is specified, the second value is assumed to be center.

Value:	[left center right top bottom <percentage> <length>] [left center right <percentage> <length>] [top center bottom <percentage> <length>]
Initial:	50% 50%
Inherited:	No

■ Source: CSS3, Experimental

■ **Deprecated!** Use `transform-origin` (p. 162) instead.

unicode-bidi

This property relates to the handling of bidirectional text in a document.

Value:	normal embed bidi-override inherit
Initial:	normal
Inherited:	No

normal

The element does not open an additional level of embedding with respect to the bidirectional algorithm. For inline elements, implicit reordering works across element boundaries.

embed

If the element is inline, this value opens an additional level of embedding with respect to the bidirectional algorithm. The direction of this embedding level is given by the 'direction' property. Inside the element, reordering is done implicitly. This corresponds to adding a LRE (U+202A; for 'direction: ltr') or RLE (U+202B; for 'direction: rtl') at the start of the element and a PDF (U+202C) at the end of the element.

bidi-override

For inline elements this creates an override. For block container elements this creates an override for inline-level descendants not within another block container element. This means that inside the element, reordering is strictly in sequence according to the 'direction' property; the implicit part of the bidirectional algorithm is ignored. This corresponds to adding a LRO (U+202D; for 'direction: ltr') or RLO (U+202E; for 'direction: rtl') at the start of the element or at the start of each anonymous child block box, if any, and a PDF (U+202C) at the end of the element.

- *Source:* CSS 2.1
- *See also:* [direction](#) (p. 129)
- *More information:* [Right-to-Left](#) (p. 81)

vertical-align

This property affects the vertical positioning inside a line box of the boxes generated by an inline-level element.

Value:	baseline sub super top text-top middle bottom text-bottom <percentage> <length> inherit
Initial:	baseline
Applies To:	inline-level and 'table-cell' elements
Inherited:	No

baseline

Align the baseline of the box with the baseline of the parent box. If the box does not have a baseline, align the bottom margin edge with the parent's baseline.

middle

Align the vertical midpoint of the box with the baseline of the parent box plus half the x-height of the parent.

sub

Lower the baseline of the box to the proper position for subscripts of the parent's box. (This value has no effect on the font size of the element's text.)

super

Raise the baseline of the box to the proper position for superscripts of the parent's box. (This value has no effect on the font size of the element's text.)

text-top

Align the top of the box with the top of the parent's content area.

text-bottom

Align the bottom of the box with the bottom of the parent's content area.

<percentage>

Raise (positive value) or lower (negative value) the box by this distance (a percentage of the 'line-height' value). The value '0%' means the same as 'baseline'.

<length>

Raise (positive value) or lower (negative value) the box by this distance. The value '0cm' means the same as 'baseline'.

top

Align the top of the aligned subtree with the top of the line box.

bottom

Align the bottom of the aligned subtree with the bottom of the line box.

■ Source: CSS 2.1

visibility

The 'visibility' property specifies whether the boxes generated by an element are rendered. Invisible boxes still affect layout (set the 'display' property to 'none' to suppress box generation altogether).

Value:	visible hidden collapse inherit
Initial:	visible
Inherited:	Yes

visible

The generated box is visible.

hidden

The generated box is invisible (fully transparent, nothing is drawn), but still affects layout. Furthermore, descendants of the element will be visible if they have 'visibility: visible'.

collapse

For table rows, columns, column groups, and row groups the rows or columns are hidden and do not occupy space, as if display: none were applied to them. If used on elements other than rows, row groups, columns, or column groups, 'collapse' has the same meaning as 'hidden'.

■ Source: CSS 2.1

white-space

This property declares how white space inside the element is handled.

Value:	normal pre nowrap pre-wrap pre-line inherit
Initial:	normal
Inherited:	Yes

normal

This value directs user agents to collapse sequences of white space, and break lines as necessary to fill line boxes.

pre

This value prevents user agents from collapsing sequences of white space. Lines are only broken at preserved newline characters.

nowrap

This value collapses white space as for 'normal', but suppresses line breaks within text.

pre-wrap

This value prevents user agents from collapsing sequences of white space. Lines are broken at preserved newline characters, and as necessary to fill line boxes.

pre-line

This value directs user agents to collapse sequences of white space. Lines are broken at preserved newline characters, and as necessary to fill line boxes.

■ *Source:* CSS 2.1

widows

The 'widows' property specifies the minimum number of lines in a block container that must be left at the top of a page.

Value:	<integer> inherit
Initial:	2
Applies To:	block container elements
Inherited:	Yes

■ *Source:* CSS 2.1

■ *More information:* [Widows & Orphans \(p. 51\)](#)

width

This property specifies the content width of boxes.

Value:	<length> <percentage> auto inherit
Initial:	auto
Applies To:	all elements but non-replaced inline elements, table rows, and row groups
Inherited:	No

<length>

Specifies the width of the content area using a length unit.

<percentage>

Specifies a percentage width. The percentage is calculated with respect to the width of the generated box's containing block.

auto

The width depends on the values of other properties.

■ *Source:* CSS 2.1

-ro-width

This property allows the automatic resizing of form controls according to their content. If this property is set to `auto`, the form controls' width automatically adjusts according to its content.

Value:	auto none
Initial:	none
Applies To:	Form elements
Inherited:	No

auto

Automatically adjusts the width of a form control if the contents' width exceeds the width defined for the form control.

- Source: Proprietary
- More information: [Automatic Resizing of Form Controls \(p. 79\)](#)

z-index

For a positioned box, the 'z-index' property specifies: 1. The stack level of the box in the current stacking context. 2. Whether the box establishes a stacking context.

Value:	<Integer> auto inherit
Initial:	auto
Applies To:	positioned elements
Inherited:	No

<integer>

This integer is the stack level of the generated box in the current stacking context. The box also establishes a new stacking context.

auto

The stack level of the generated box in the current stacking context is 0. The box does not establish a new stacking context unless it is the root element.

- Source: CSS 2.1