Every option, rendered

For each option that changes what comes out: a small page, the C# that sets the option, and the pages Weblitz printed, beside the default where that shows the difference. Made on 2026-11-23 on macOS; the pages are drawn from the PDFs, which are downloads. What the engine noted is shown as it came back, including what it does not do yet.

Paper size and orientation
Margins
Scale
Header and footer templates
Backgrounds
Choosing pages
The page's own @page size
Scripts and the virtual clock
The network
Pages behind a login
PDF compression
Passwords and permissions
Title, author and PDF/A
Limits
Notes: what could not be done
The browser the page sees
Locale, time zone and clock
Installed fonts
Lists that load as you scroll
Scrolling to a section first
Printing a part of the page
Chrome's experimental features
Printing documentation without its menus
Without the consent boxes
Without advertisements and trackers

Paper size and orientation

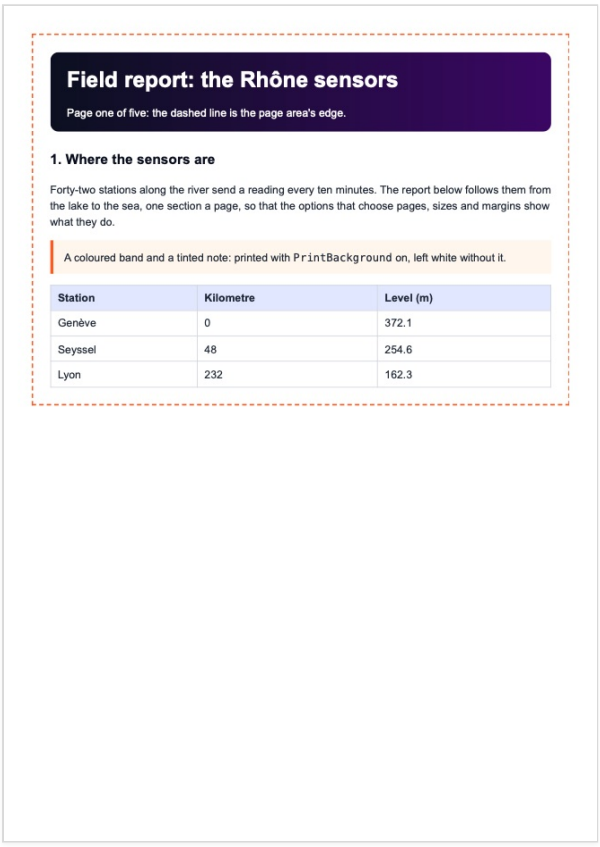
The page area changes with the paper: the dashed frame is the area's edge.

Reference: `HtmlPdfOptions.Format`, `HtmlPdfOptions.Landscape`, `HtmlPdfOptions.Width`, `HtmlPdfOptions.Height`

▶ The page printed: `article.html`

Default: A4, portrait

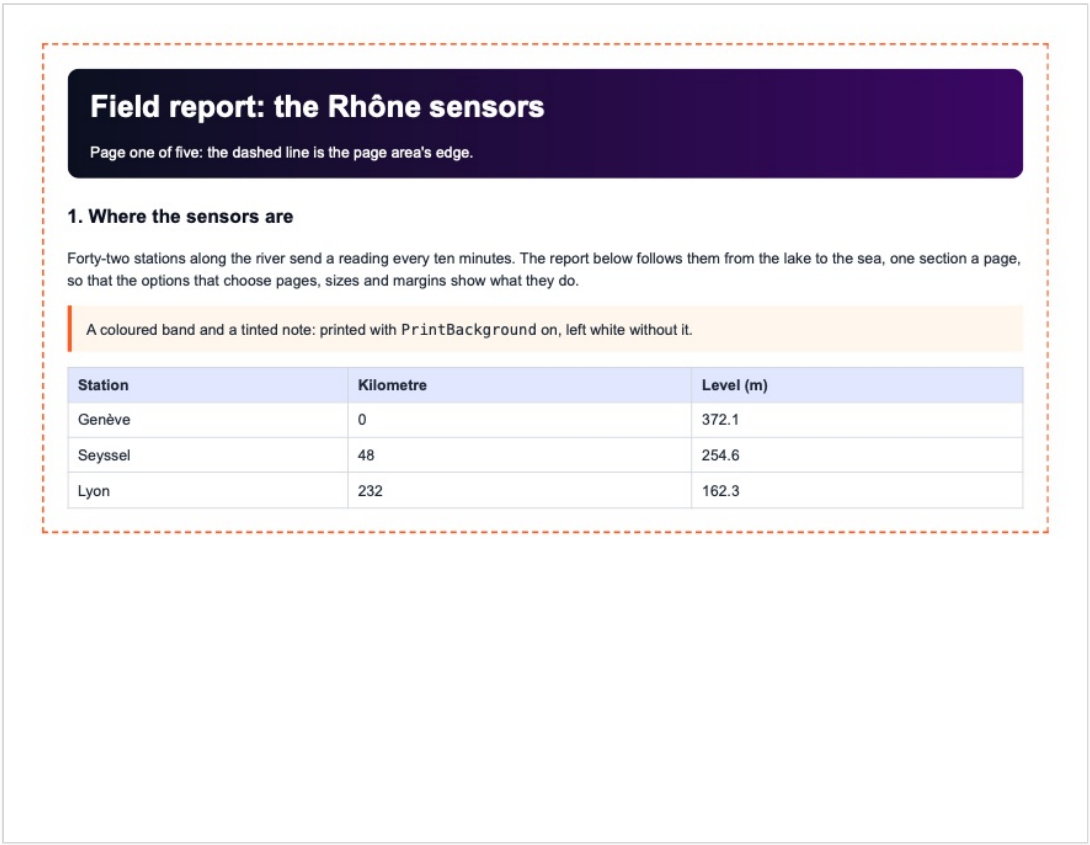
```
var result = HtmlToPdf.Render(html);
```



Pages 5
File 95,383 bytes (93.1 KB) · the PDF

Letter, landscape

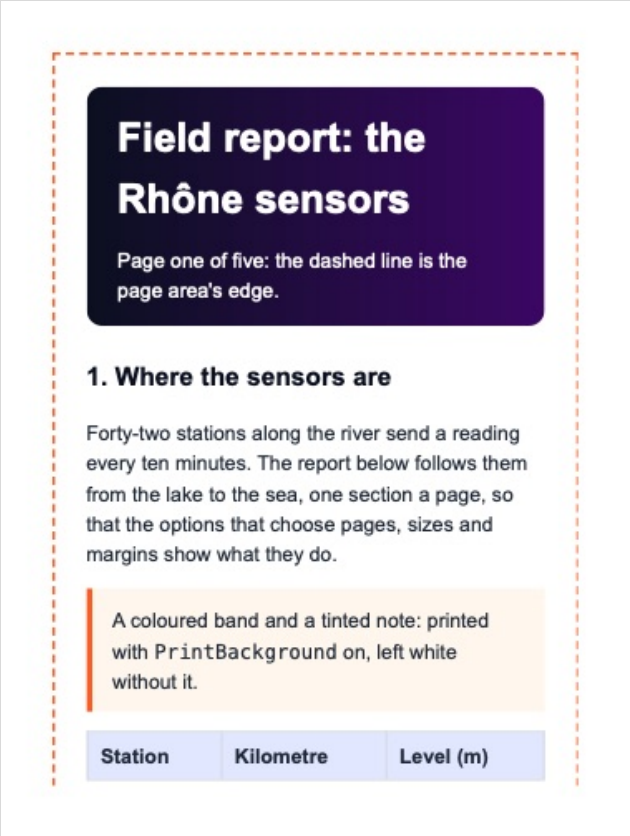
```
var result = HtmlToPdf.Render(html, new HtmlPdfOptions { Format = "Letter", Landscape = true });
```



Pages 5
File 99,598 bytes (97.3 KB) · the PDF

Width and height: 120 × 160 mm

```
var result = HtmlToPdf.Render(html, new HtmlPdfOptions { Width = "120mm", Height =
"160mm" });
```



Pages 6
File 87,157 bytes (85.1 KB) · the PDF

Margins

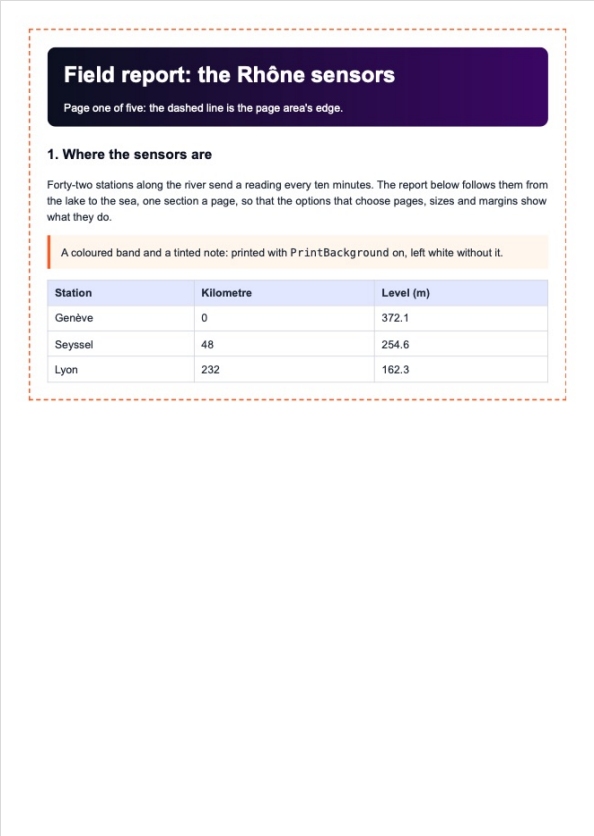
Without margins in the options or the page's @page rule, Chrome's default margins apply.

Reference: `HtmlPdfOptions.Margin`, `HtmlMargins`

▶ The page printed: `article.html`

Default

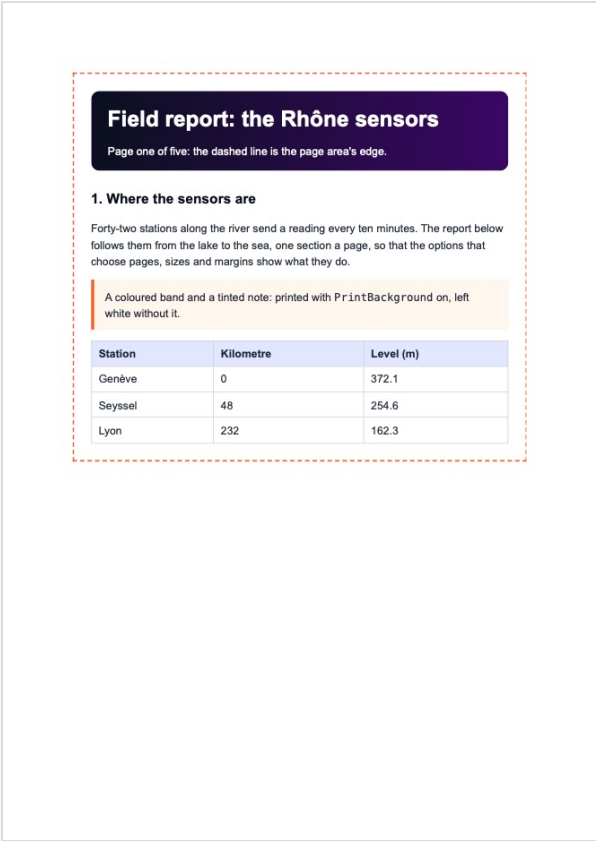
```
var result = HtmlToPdf.Render(html);
```



Pages 5
File 95,383 bytes (93.1 KB) · the PDF

25 mm all round

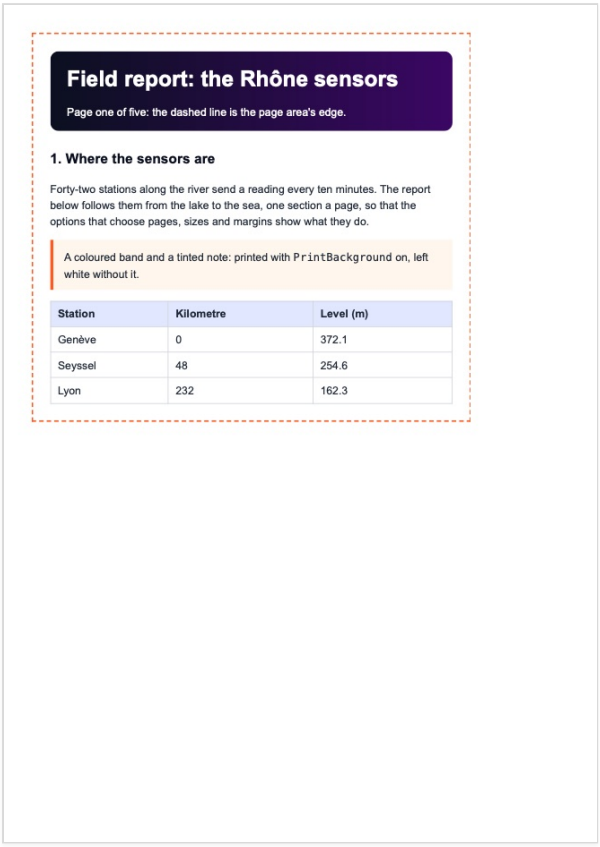
```
var result = HtmlToPdf.Render(html, new HtmlPdfOptions { Margin =
HtmlMargins.All("25mm") });
```



Pages 5
File 91,048 bytes (88.9 KB) · the PDF

Top, right, bottom, left

```
var result = HtmlToPdf.Render(html, new HtmlPdfOptions { Margin = new
HtmlMargins("10mm", "45mm", "10mm", "10mm") });
```



Pages 5
File 91,538 bytes (89.4 KB) · the PDF

Scale

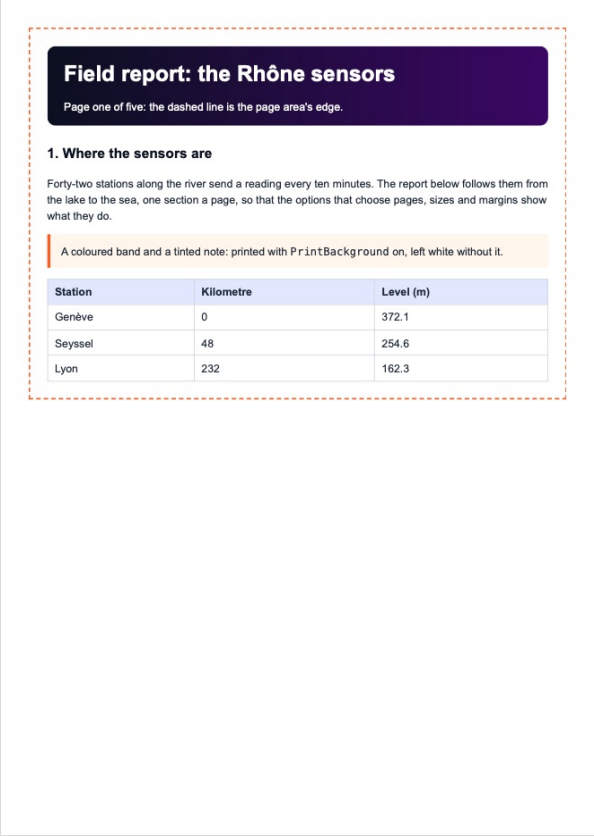
The page is laid out as if the paper were larger or smaller, then printed on it: 0.1 to 2.

Reference: `HtmlPdfOptions.Scale`

▶ The page printed: `article.html`

1 (default)

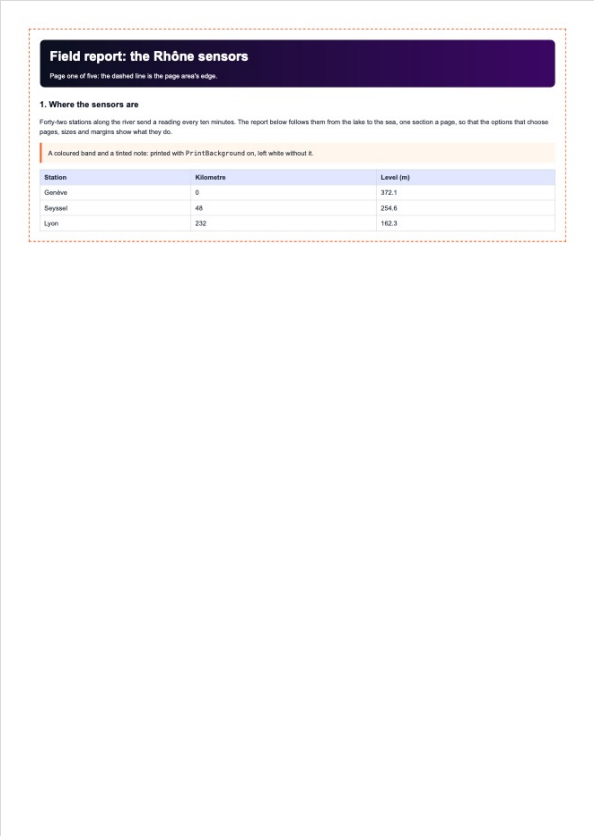
```
var result = HtmlToPdf.Render(html);
```



Pages 5
File 95,383 bytes (93.1 KB) · the PDF

0.6

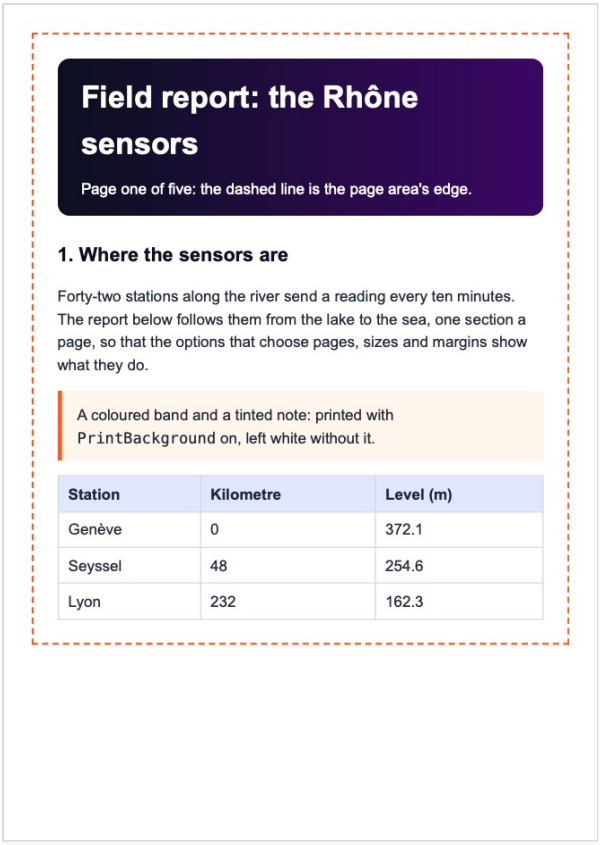
```
var result = HtmlToPdf.Render(html, new HtmlPdfOptions { Scale = 0.6 });
```



Pages 5
File 102,606 bytes (100.2 KB) · the PDF

1.4

```
var result = HtmlToPdf.Render(html, new HtmlPdfOptions { Scale = 1.4 });
```



Pages 5
File 92,984 bytes (90.8 KB) · the PDF

Header and footer templates

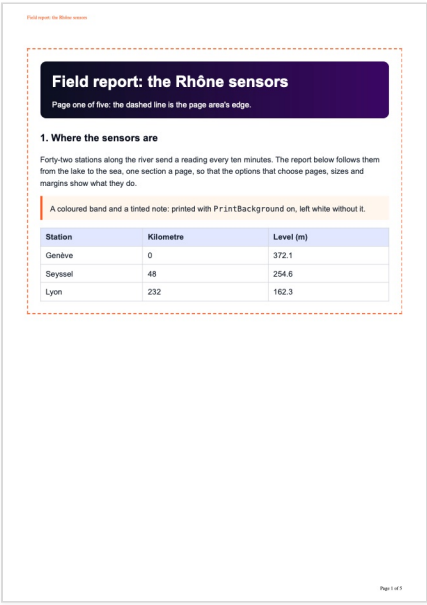
The templates are HTML printed in the top and bottom margins of every page, as Chrome prints them; elements with the classes pageNumber, totalPages, title, url and date are filled in. A template that sets no font size gets Chrome's tiny default: set one. Give the margins room for them.

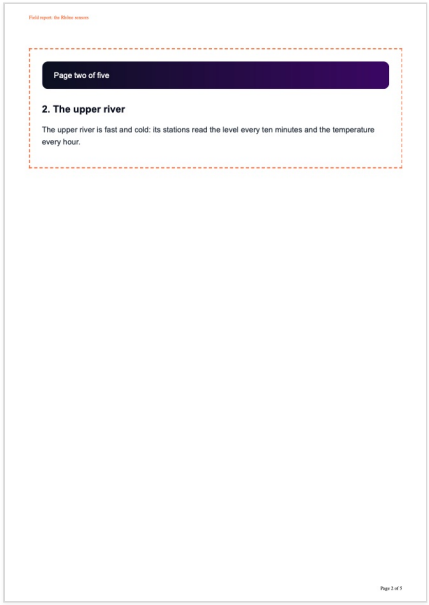
Reference: HtmlPdfOptions.DisplayHeaderFooter, HtmlPdfOptions.HeaderTemplate, HtmlPdfOptions.FooterTemplate

▶ The page printed: `article.html`

Header and footer

```
var result = HtmlToPdf.Render(html, new HtmlPdfOptions
{
    DisplayHeaderFooter = true,
    Margin = new HtmlMargins("22mm", "12mm", "20mm", "12mm"),
    HeaderTemplate = "<div style='font-size:9px; width:100%; padding:0 12mm;
color:#ff5a1f'><span class='title'></span></div>",
    FooterTemplate = "<div style='font-size:9px; width:100%; padding:0 12mm; text-
align:right'>Page <span class='pageNumber'></span> of <span class='totalPages'></span>
</div>",
});
```





Pages 5
File 108,127 bytes (105.6 KB) · the PDF

Backgrounds

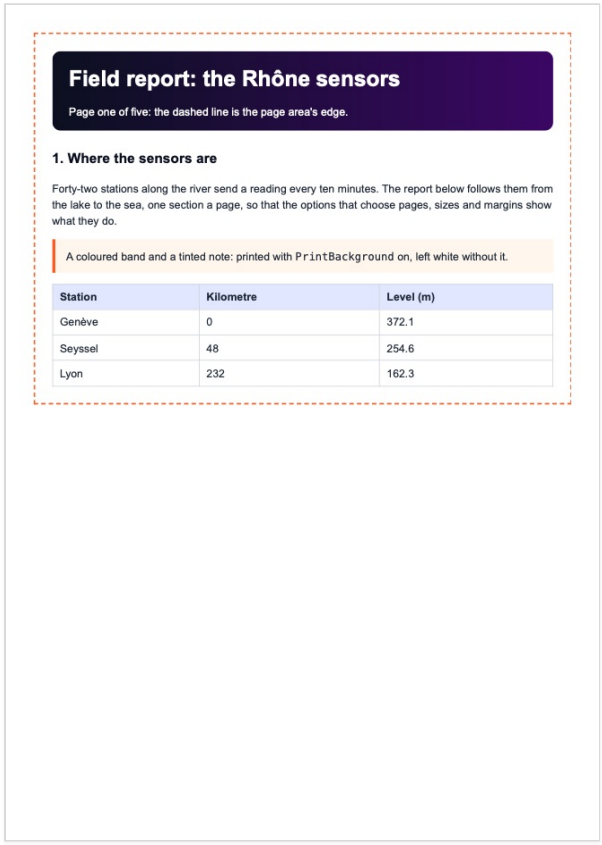
Background colours, gradients and pictures are printed by default; off, they are left out, as Chrome's "Background graphics" box unticked.

Reference: `HtmlPdfOptions.PrintBackground`

▶ The page printed: `article.html`

true (default)

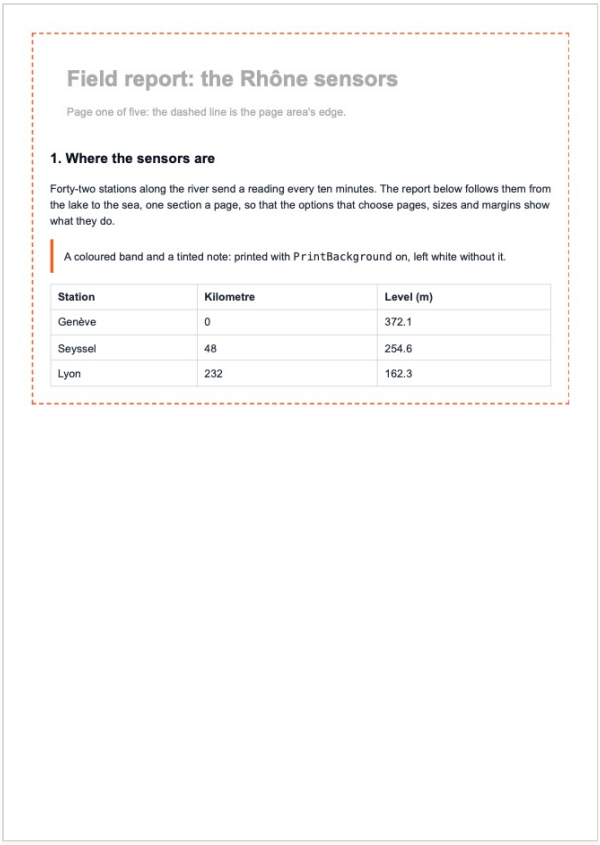
```
var result = HtmlToPdf.Render(html);
```



Pages 5
File 95,383 bytes (93.1 KB) · the PDF

false

```
var result = HtmlToPdf.Render(html, new HtmlPdfOptions { PrintBackground = false });
```



Pages 5
File 87,547 bytes (85.5 KB) · the PDF

Choosing pages

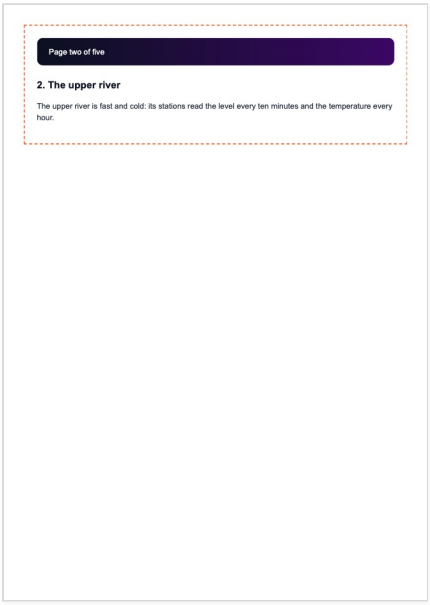
The article has five pages; the PDF holds the ones asked for.

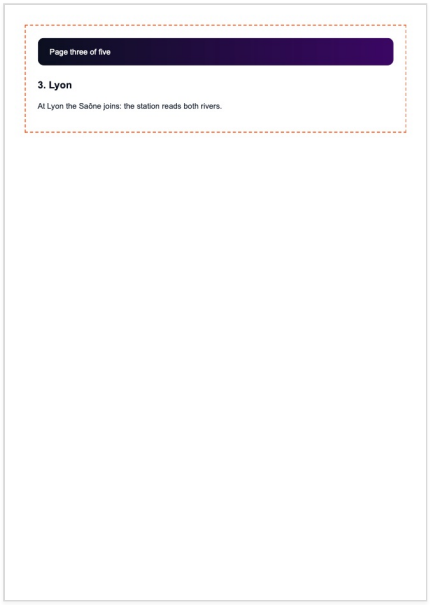
Reference: `HtmlPdfOptions.PageRanges`

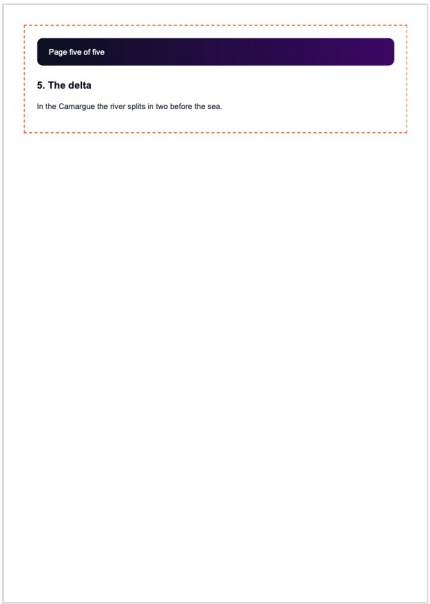
▶ The page printed: `article.html`

"2-3, 5"

```
var result = HtmlToPdf.Render(html, new HtmlPdfOptions { PageRanges = "2-3, 5" });
```







Pages	3
File	59,266 bytes (57.9 KB) · the PDF

The page's own @page size

A page that asks for its paper size gets it by default; with PreferCssPageSize false, the options' Format wins.

Reference: `HtmlPdfOptions.PreferCssPageSize`

▶ The page printed: `css-page-size.html`

true (default): A5 landscape, as the page asks

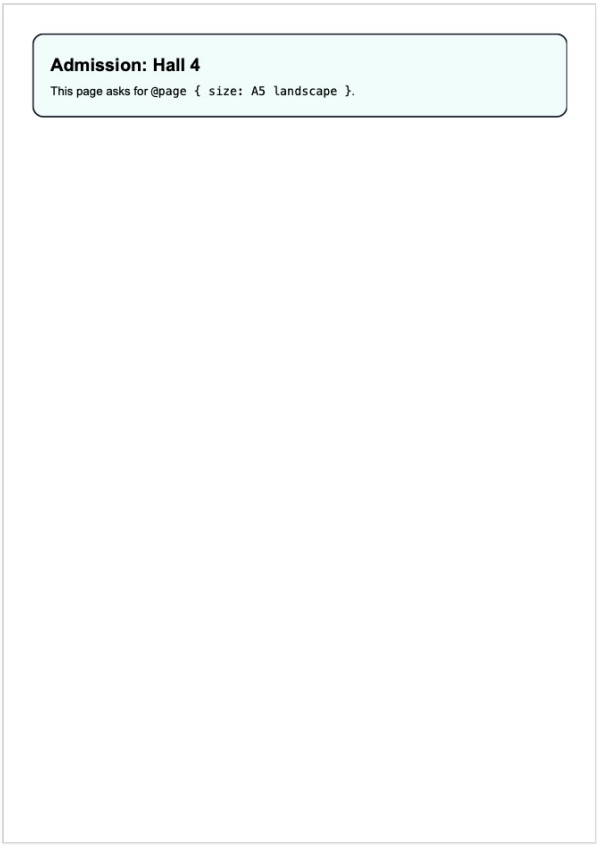
```
var result = HtmlToPdf.Render(html);
```



Pages 1
File 39,665 bytes (38.7 KB) · the PDF

false: the options' A4

```
var result = HtmlToPdf.Render(html, new HtmlPdfOptions { PreferCssPageSize = false,
Format = "A4" });
```



Pages 1
File 39,680 bytes (38.8 KB) · the PDF

Scripts and the virtual clock

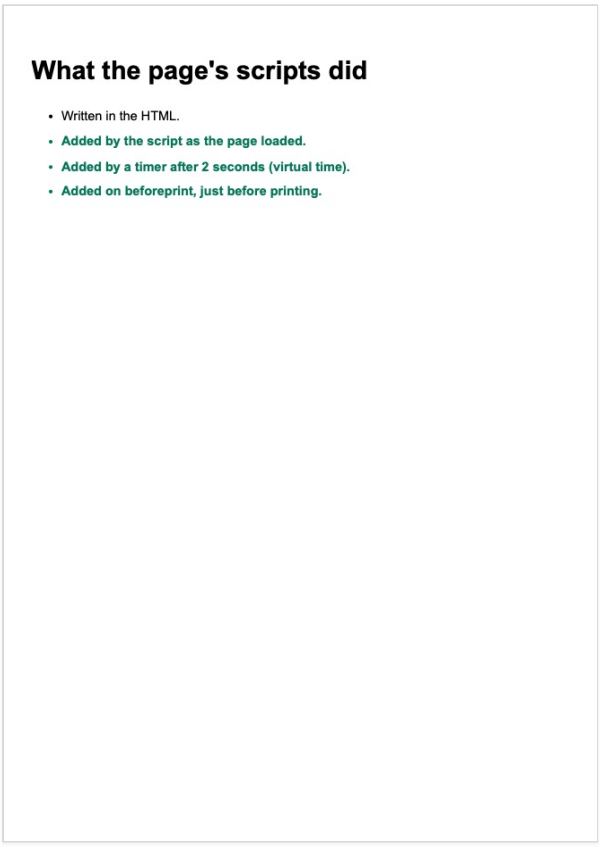
Scripts run once before printing; timers run in virtual time for WaitFor (5 seconds by default), so a page's 2-second timer fires and its 8-second one only with a longer wait. beforeprint runs just before the PDF is made.

Reference: `HtmlPdfOptions.Scripts`, `HtmlScripts`

▶ The page printed: `scripts.html`

Default: scripts on, 5 s of virtual time

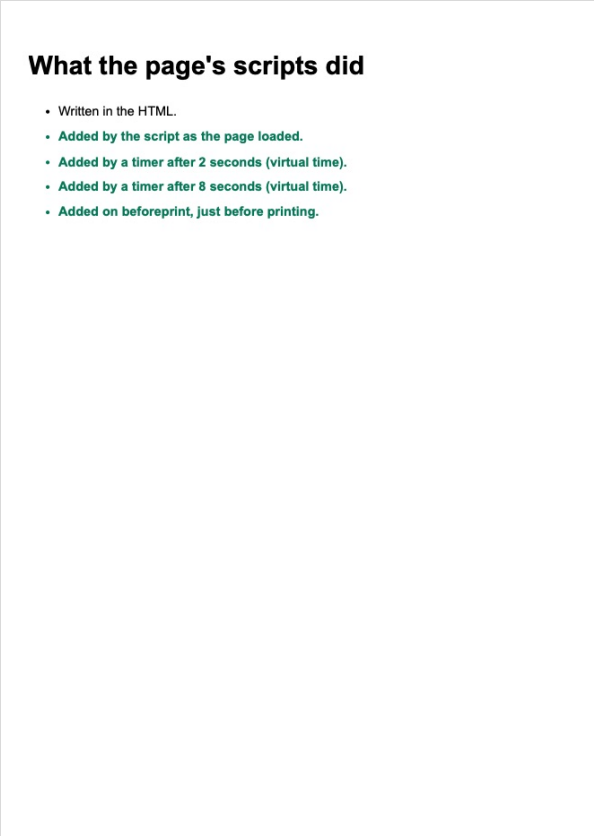
```
var result = HtmlToPdf.Render(html);
```



Pages	1
File	36,218 bytes (35.4 KB) · the PDF
Notes	`Info scriptsRan`: The page's scripts ran (2017 ms of virtual time, 5 tasks; 0 canvases drawn).

WaitFor 10 s

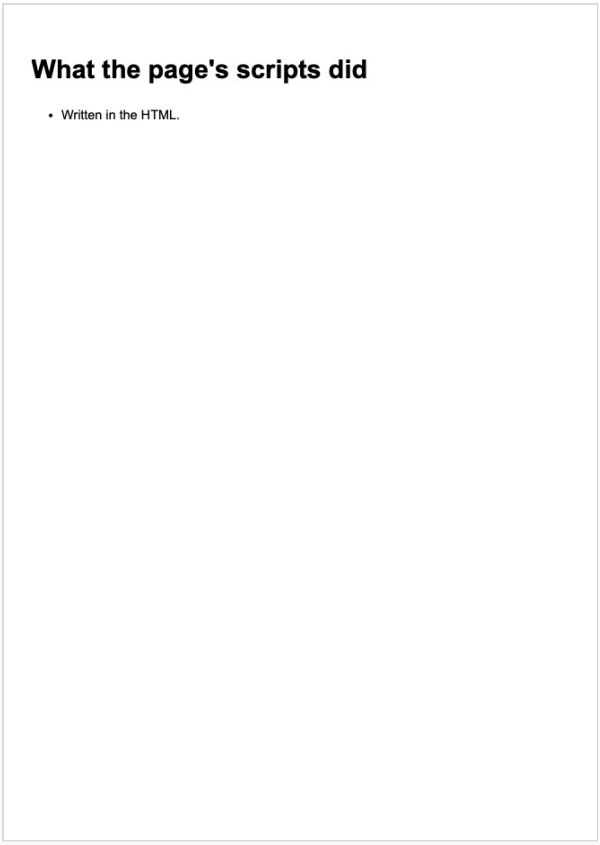
```
var result = HtmlToPdf.Render(html, new HtmlPdfOptions { Scripts = HtmlScripts.Default
with { WaitFor = TimeSpan.FromSeconds(10) } });
```



Pages	1
File	37,010 bytes (36.1 KB) · the PDF
Notes	`Info scriptsRan`: The page's scripts ran (8017 ms of virtual time, 8 tasks; 0 canvases drawn).

Scripts off

```
var result = HtmlToPdf.Render(html, new HtmlPdfOptions { Scripts = HtmlScripts.Off });
```



Pages 1
File 30,676 bytes (30.0 KB) · the PDF

The network

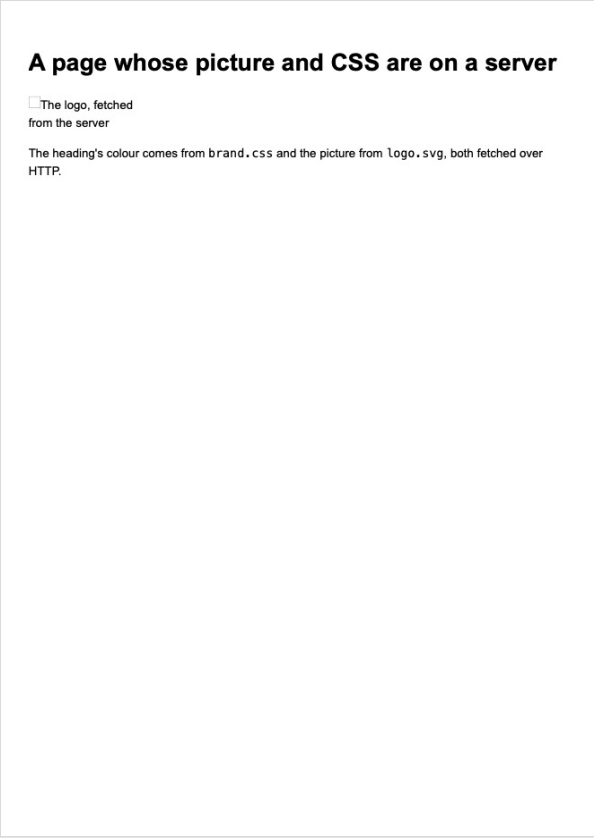
The network is off by default: a page's web addresses are not fetched and noted instead. Allow the hosts a page may load from (`assets` is a server the generator starts on this machine).

Reference: `HtmlPdfOptions.Network`, `HtmlNetwork`, `HtmlPdfOptions.BaseUrl`

▶ The page printed: `network.html`

Off (default)

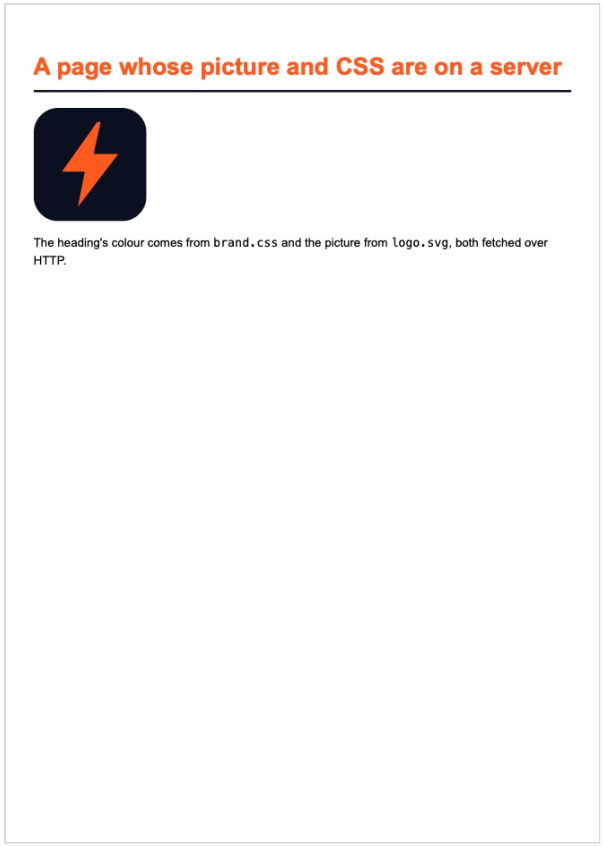
```
var result = HtmlToPdf.Render(html, new HtmlPdfOptions { BaseUrl = assets });
```



Pages	1
File	44,119 bytes (43.1 KB) · the PDF
Notes	`Info resourceRefused`: StyleSheet http://127.0.0.1:18431/brand.css: no resource loader was given (only data: URLs load). `Info resourceRefused`: Image http://127.0.0.1:18431/logo.svg: no resource loader was given (only data: URLs load). `Info notImplemented`: A picture that did not load shows its alternative text; Chrome's broken-image icon beside it is laid out but not drawn.

Allow the assets' host

```
var result = HtmlToPdf.Render(html, new HtmlPdfOptions { BaseUrl = assets, Network = HtmlNetwork.Allow(assets.Host) });
```



Pages 1
File 45,952 bytes (44.9 KB) · the PDF

Pages behind a login

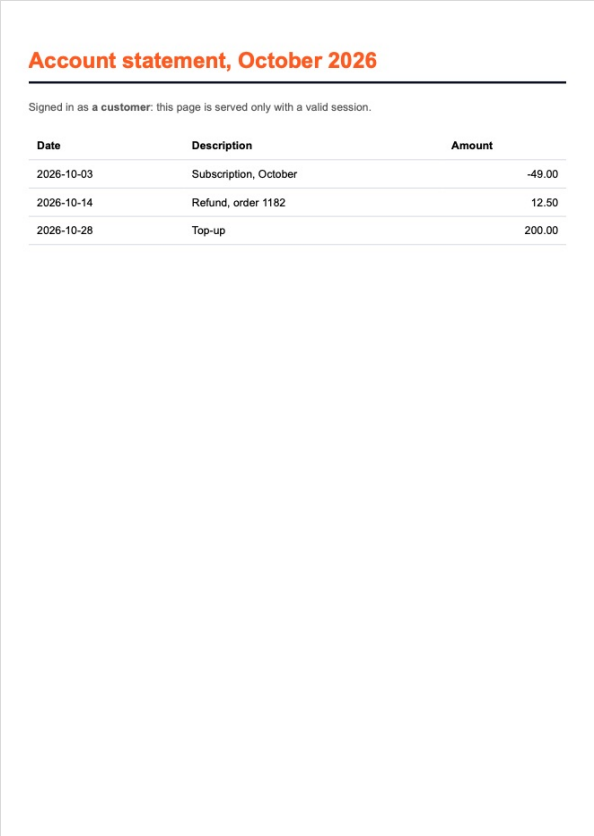
A page your server shows only to signed-in users prints with what a browser would send: a bearer token, a session cookie, or a password for HTTP authentication. Each goes only to the hosts it names, by default the document's own (`assets` is a server the generator starts on this machine; its account pages need one of them).

Reference: `HtmlNetwork.Headers`, `HtmlNetwork.Cookies`, `HtmlNetwork.AcceptCookies`, `HtmlNetwork.Credentials`, `HtmlCookie`

▶ The page printed: `assets/account/statement.html`

A bearer token

```
var result = HtmlToPdf.RenderUrl(new Uri(assets, "account/statement.html"), new
HtmlPdfOptions
{
    Network = HtmlNetwork.Allow(assets.Host).WithBearerToken(accessToken),
});
```



Account statement, October 2026

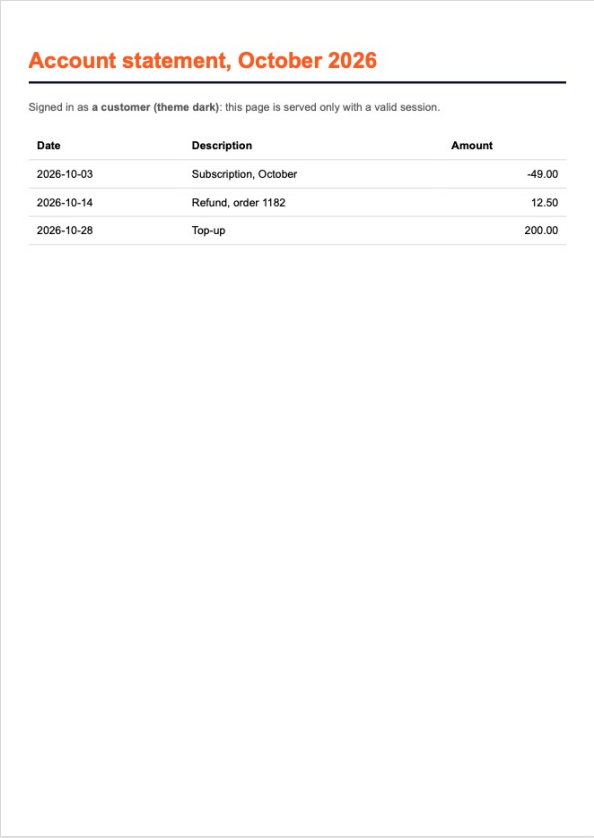
Signed in as **a customer**: this page is served only with a valid session.

Date	Description	Amount
2026-10-03	Subscription, October	-49.00
2026-10-14	Refund, order 1182	12.50
2026-10-28	Top-up	200.00

Pages 1
File 43,239 bytes (42.2 KB) · the PDF
Notes `Info scriptsRan`: The page's scripts ran (17 ms of virtual time, 2 tasks; 0 canvases drawn).

A session cookie

```
var result = HtmlToPdf.RenderUrl(new Uri(assets, "account/statement.html"), new
HtmlPdfOptions
{
    Network = HtmlNetwork.Allow(assets.Host)
        .WithCookie(new HtmlCookie("session", sessionId) { HttpOnly = true })
        .WithCookie(new HtmlCookie("theme", "dark")),
});
```



Account statement, October 2026

Signed in as **a customer (theme dark)**: this page is served only with a valid session.

Date	Description	Amount
2026-10-03	Subscription, October	-49.00
2026-10-14	Refund, order 1182	12.50
2026-10-28	Top-up	200.00

Pages 1
File 44,446 bytes (43.4 KB) · the PDF
Notes `Info scriptsRan`: The page's scripts ran (17 ms of virtual time, 2 tasks; 0 canvases drawn).

PDF compression

The same pages, deflated more or less: the files' sizes below, measured when these pages were made.

Reference: `HtmlPdfOptions.Compression`, `HtmlPdfCompression`

▶ The page printed: `article.html`

None

```
var result = HtmlToPdf.Render(html, new HtmlPdfOptions { Compression =
HtmlPdfCompression.None });
```

Pages 5
File 345,843 bytes (337.7 KB) · the PDF

Fastest (default)

```
var result = HtmlToPdf.Render(html);
```

Pages 5
File 95,383 bytes (93.1 KB) · the PDF

Default

```
var result = HtmlToPdf.Render(html, new HtmlPdfOptions { Compression =
HtmlPdfCompression.Default });
```

Pages 5
File 74,765 bytes (73.0 KB) · the PDF

Smallest

```
var result = HtmlToPdf.Render(html, new HtmlPdfOptions { Compression =
HtmlPdfCompression.Smallest });
```

Pages 5
File 72,832 bytes (71.1 KB) · the PDF

Passwords and permissions

The PDF is encrypted (AES-256): a reader asks for the user password, and without the owner's it may not copy (here).

Reference: `HtmlPdfOptions.Security`, `HtmlPdfSecurity`

▶ The page printed: `article.html`

A user password, no copying

```
var result = HtmlToPdf.Render(html, new HtmlPdfOptions
{
    Security = new HtmlPdfSecurity { UserPassword = "open-me", OwnerPassword = "owner-secret", PermitCopying = false },
});
```

Pages	5
File	98,636 bytes (96.3 KB), encrypted · the PDF

Title, author and PDF/A

What the PDF says of itself; PDF/A-2b for archives.

Reference: `HtmlPdfOptions.Title`, `HtmlPdfOptions.Author`, `HtmlPdfOptions.PdfA`

▶ The page printed: `article.html`

Title, author, PDF/A

```
var result = HtmlToPdf.Render(html, new HtmlPdfOptions { Title = "Rhône field report",
Author = "Hydrology team", PdfA = true });
```

Pages	5
File	99,689 bytes (97.4 KB), PDF/A · the PDF

Limits

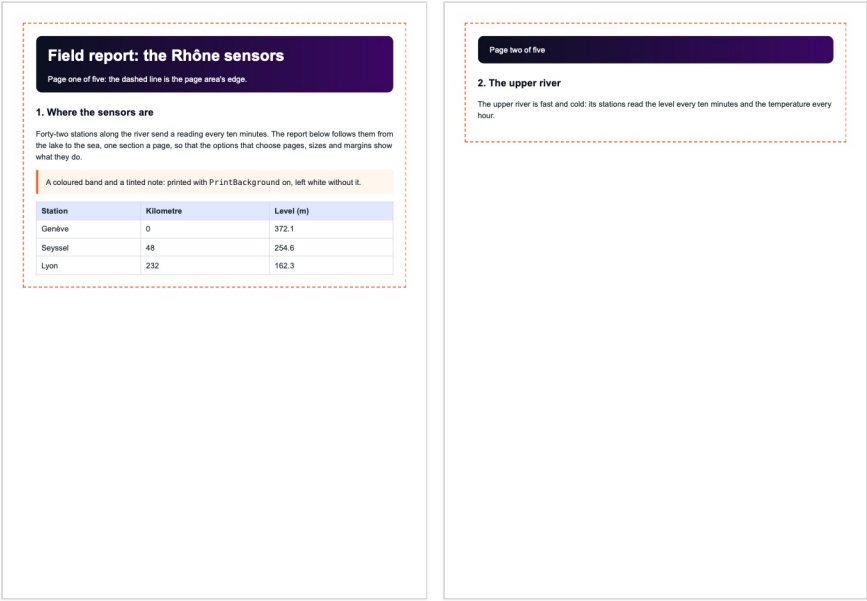
A page held to its limits: here two pages at most. The render stops there, prints what it has, and says so: Completed is false and a note tells which limit.

Reference: `HtmlPdfOptions.Limits`, `HtmlPdfLimits`, `HtmlPdfResult.Completed`

▶ The page printed: `article.html`

MaxPages = 2

```
var result = HtmlToPdf.Render(html, new HtmlPdfOptions { Limits =
HtmlPdfLimits.Default with { MaxPages = 2 } });
Console.WriteLine($"{result.PageCount} pages, completed: {result.Completed}");
```



Pages	2, stopped early (Completed false)
File	74,993 bytes (73.2 KB) · the PDF
Notes	`Warning limitReached`: the document was cut after 2 pages.

Notes: what could not be done

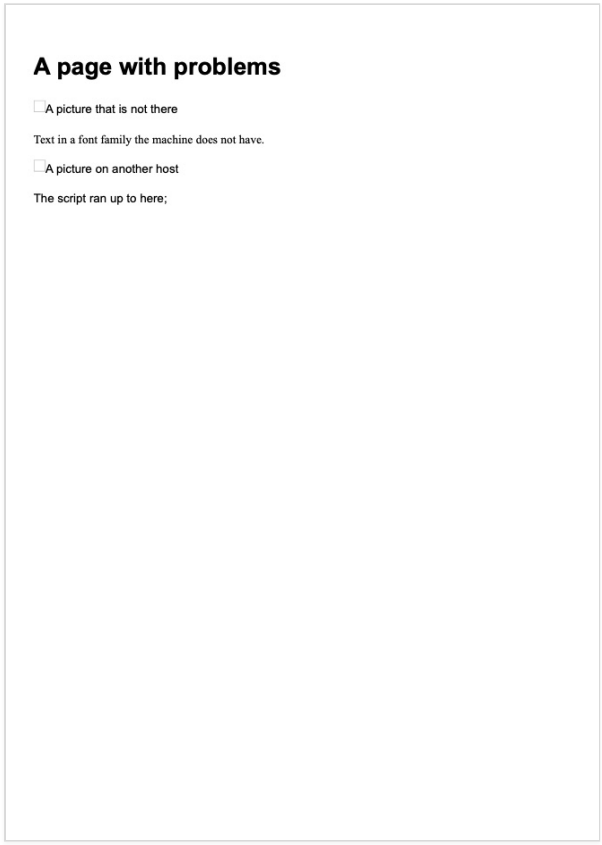
A page's own problems never throw: a picture missing, a host not allowed, a font the machine lacks, a script that throws. Each is a note.

Reference: `HtmlPdfResult.Notes`, `HtmlPdfNote`, `HtmlPdfNoteSeverity`

▶ The page printed: `notes.html`

A page with problems

```
var result = HtmlToPdf.Render(html);
foreach (var note in result.Notes)
    Console.WriteLine($"{note.Severity} {note.Code}: {note.Message}");
```



Pages	1
File	56,265 bytes (54.9 KB) · the PDF
Notes	`Warning scriptError`: ReferenceError: undefinedFunction is not defined | at about:blank#script-25:11:3 | at HostDelegate.<anonymous> (<anonymous>) | at Object.hostCall (ssrs2-dom.js:15562:11) `Warning resourceFailed`: Image "missing-picture.png" is not a URL that resolves against no base URL. `Info resourceRefused`: Image https://cdn.example.com/banner.png: no resource loader was given (only data: URLs load). `Info scriptsRan`: The page's scripts ran (17 ms of virtual time, 4 tasks; 0 canvases drawn). `Info notImplemented`: A picture that did not load shows its alternative text; Chrome's broken-image icon beside it is laid out but not drawn. `Info fontFamilyMissing`: The font family "Some Font Nobody Has" is not on this machine: its text goes to the next family.

The browser the page sees

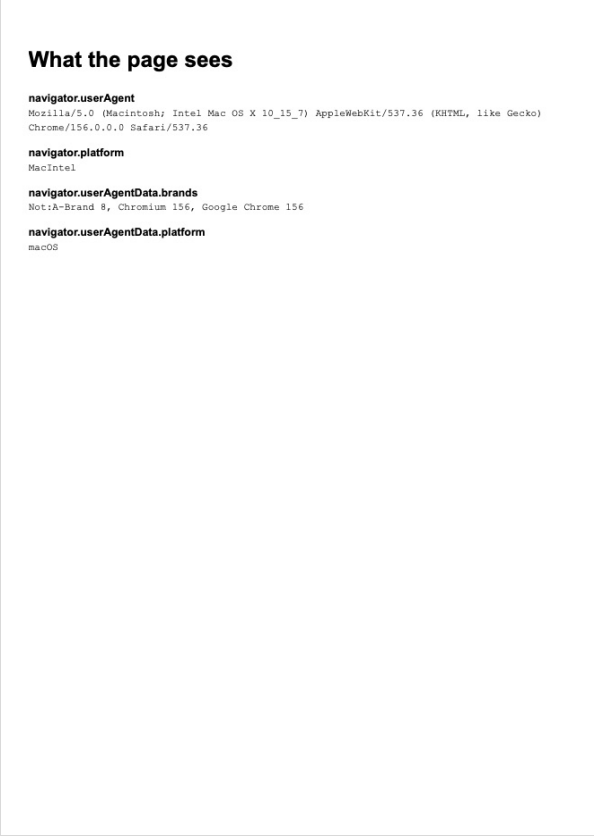
By default a page sees desktop Chrome 156 on the machine's own OS. These pages were made on macOS.

Reference: `HtmlPdfOptions.UserAgent`, `HtmlPdfOptions.UserAgentMetadata`, `HtmlUserAgentMetadata`, `HtmlUserAgentBrand`

▶ The page printed: `user-agent.html`

Default

```
var result = HtmlToPdf.Render(html);
```



Pages	1
File	44,873 bytes (43.8 KB) · the PDF
Notes	`Info scriptsRan`: The page's scripts ran (17 ms of virtual time, 2 tasks; 0 canvases drawn).

A User-Agent and its metadata

```
var result = HtmlToPdf.Render(html, new HtmlPdfOptions
{
    UserAgent = "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/156.0.0.0 Safari/537.36 InvoiceBot/2.1",
    UserAgentMetadata = HtmlUserAgentMetadata.ChromeOnThisMachine with
    {
        Platform = "Windows",
        Brands = [new("InvoiceBot", "2"), new("Chromium", "156")],
    },
});
```



Pages 1
File 45,030 bytes (44.0 KB) · the PDF
Notes `Info scriptsRan`: The page's scripts ran (17 ms of virtual time, 2 tasks; 0 canvases drawn).

Locale, time zone and clock

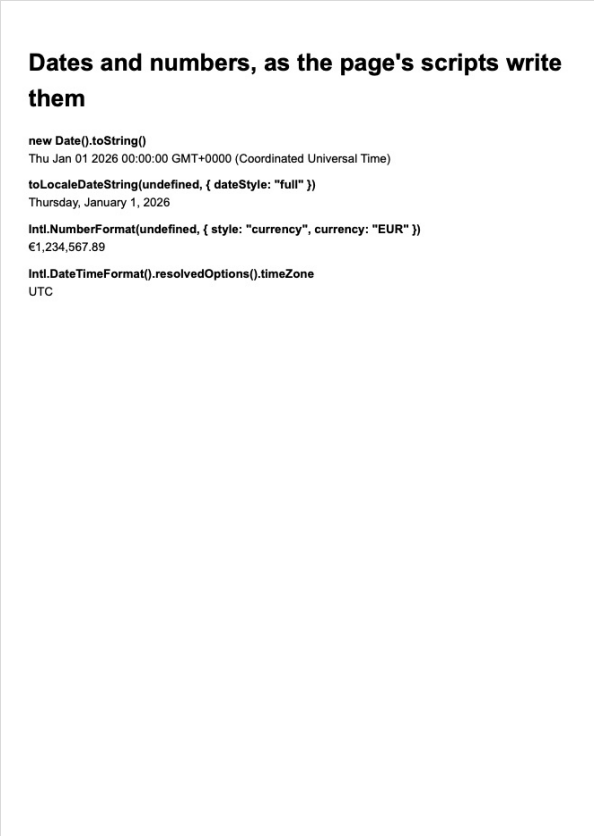
What the page's scripts see of the world: the same values every run, so the same input gives the same PDF.

Reference: `HtmlPdfOptions.Locale`, `HtmlPdfOptions.TimeZone`, `HtmlPdfOptions.Now`

▶ The page printed: `locale.html`

Default

```
var result = HtmlToPdf.Render(html);
```



Dates and numbers, as the page's scripts write them

new Date().toString()
Thu Jan 01 2026 00:00:00 GMT+0000 (Coordinated Universal Time)

toLocaleDateString(undefined, { dateStyle: "full" })
Thursday, January 1, 2026

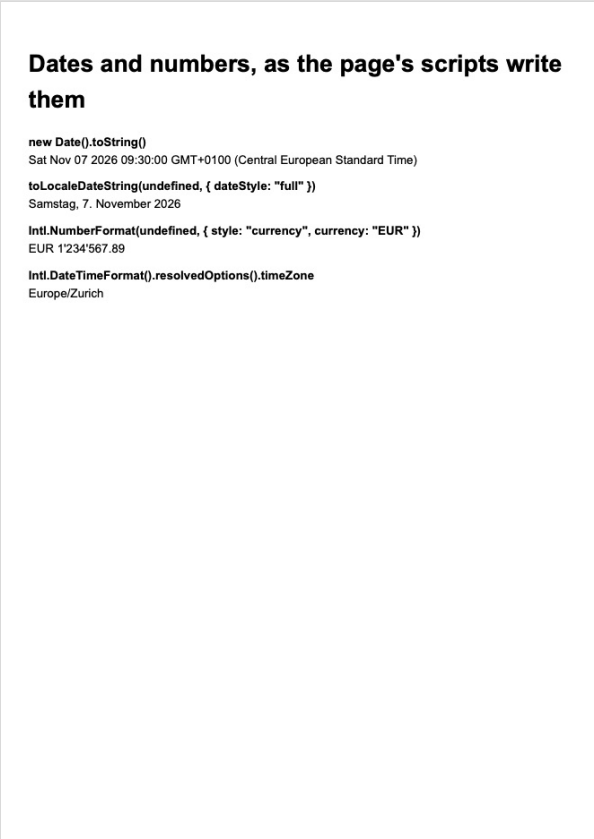
Intl.NumberFormat(undefined, { style: "currency", currency: "EUR" })
€1,234,567.89

Intl.DateTimeFormat().resolvedOptions().timeZone
UTC

Pages	1
File	48,120 bytes (47.0 KB) · the PDF
Notes	`Info scriptsRan`: The page's scripts ran (17 ms of virtual time, 2 tasks; 0 canvases drawn).

de-CH, Europe/Zurich, a fixed clock

```
var result = HtmlToPdf.Render(html, new HtmlPdfOptions
{
    Locale = "de-CH",
    TimeZone = "Europe/Zurich",
    Now = new DateTimeOffset(2026, 11, 7, 9, 30, 0, TimeSpan.FromHours(1)),
});
```



Dates and numbers, as the page's scripts write them

new Date().toString()
Sat Nov 07 2026 09:30:00 GMT+0100 (Central European Standard Time)

toLocaleDateString(undefined, { dateStyle: "full" })
Samstag, 7. November 2026

Intl.NumberFormat(undefined, { style: "currency", currency: "EUR" })
EUR 1'234'567.89

Intl.DateTimeFormat().resolvedOptions().timeZone
Europe/Zurich

Pages 1
File 50,717 bytes (49.5 KB) · the PDF
Notes `Info scripts`Ran: The page's scripts ran (17 ms of virtual time, 2 tasks; 0 canvases drawn).

Installed fonts

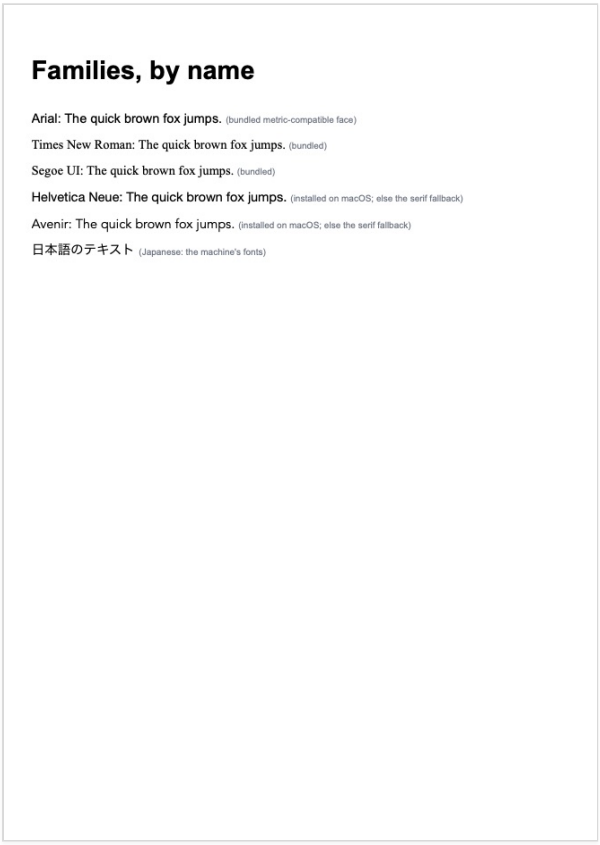
By default the machine's fonts are used after the page's own @font-face fonts; off, only the bundled faces. These pages were made on macOS.

Reference: `HtmlPdfOptions.UseInstalledFonts`

▶ The page printed: `fonts.html`

true (default)

```
var result = HtmlToPdf.Render(html);
```



Families, by name

Arial: The quick brown fox jumps. (bundled metric-compatible face)

Times New Roman: The quick brown fox jumps. (bundled)

Segoe UI: The quick brown fox jumps. (bundled)

Helvetica Neue: The quick brown fox jumps. (installed on macOS; else the serif fallback)

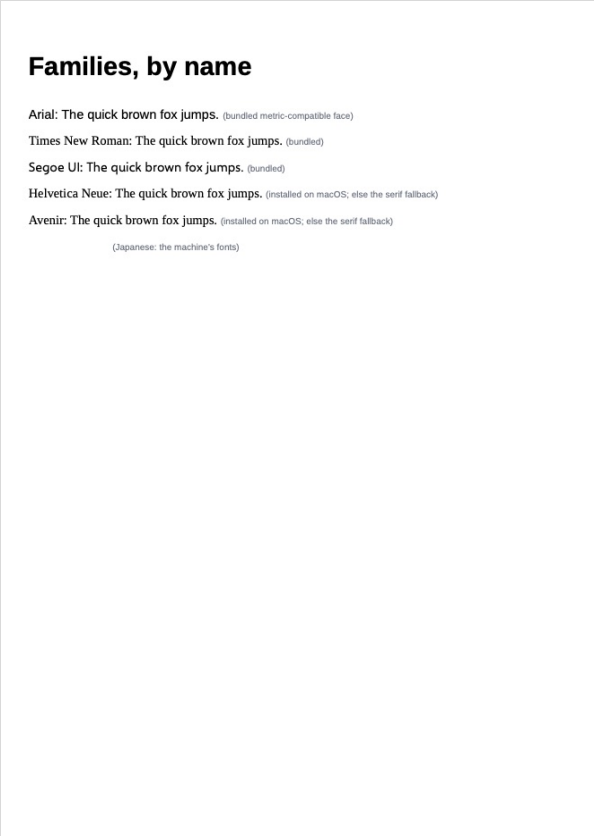
Avenir: The quick brown fox jumps. (installed on macOS; else the serif fallback)

日本語のテキスト (Japanese: the machine's fonts)

Pages	1
File	104,493 bytes (102.0 KB) · the PDF
Notes	`Info fontFamilyMissing`: The font family "Segoe UI" is not on this machine: its text goes to the next family.

false: the bundled faces only

```
var result = HtmlToPdf.Render(html, new HtmlPdfOptions { UseInstalledFonts = false });
```



Pages	1
File	36,473 bytes (35.6 KB) · the PDF
Notes	`Info fontFamilyMissing`: The font family "Helvetica Neue" is not in the font profile "reference": its text goes to the next family. `Info fontFamilyMissing`: The font family "Avenir" is not in the font profile "reference": its text goes to the next family. `Info fontFamilyMissing`: The font family "Hiragino Sans" is not in the font profile "reference": its text goes to the next family. `Warning missingGlyph`: No font of 'Hiragino Sans, serif' nor the fallback has a glyph for a character: drawn as .notdef.

Lists that load as you scroll

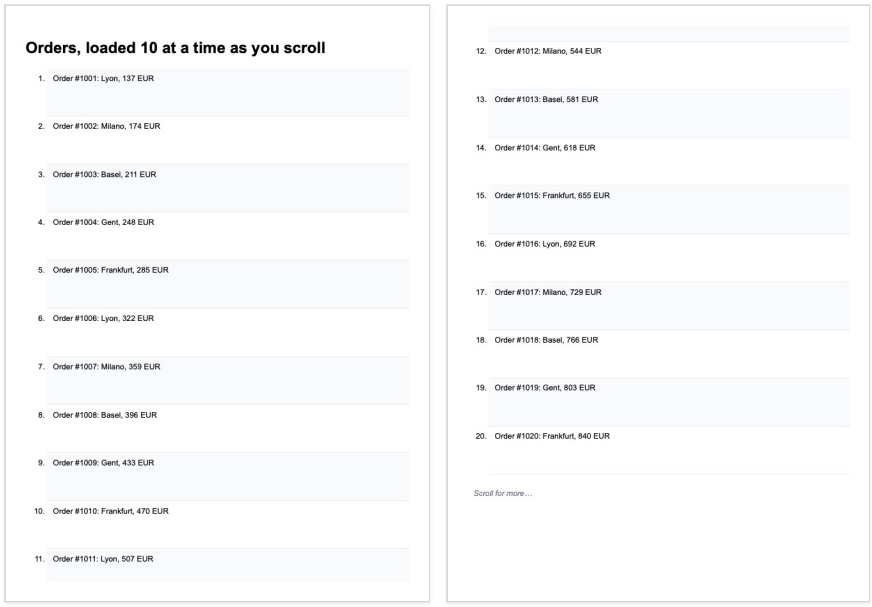
An infinite list loads its next ten orders each time its bottom comes into view. Without a scroll option, Weblitz's IntersectionObserver reports the bottom visible once, so the PDF holds twenty; ScrollThrough scrolls the window down a window height at a time, each step waiting ScrollStepWait of virtual time, so the list loads to its end and all sixty are printed.

Reference: `HtmlPdfOptions.ScrollThrough`, `HtmlPdfOptions.ScrollStepWait`

▶ The page printed: `lazy-list.html`

Default: not scrolled

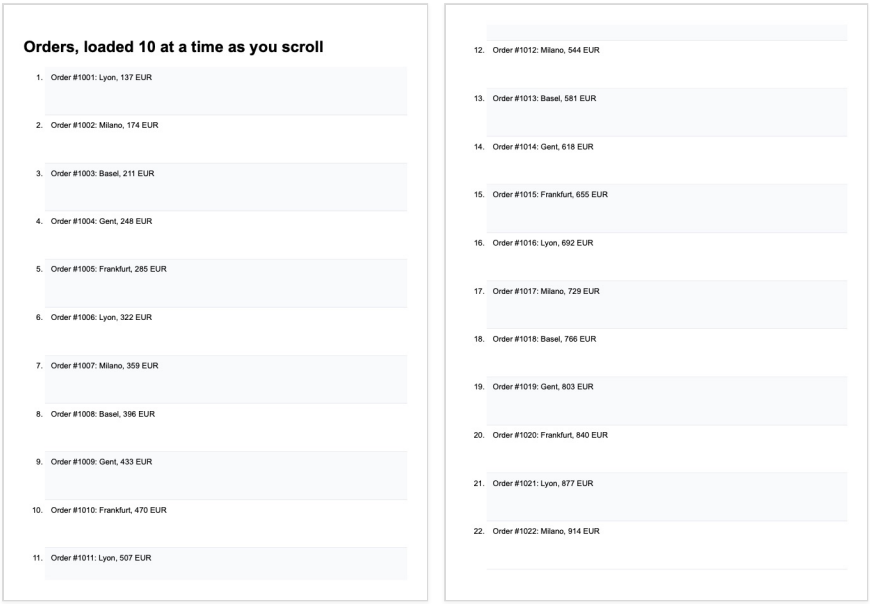
```
var result = HtmlToPdf.Render(html);
```



Pages	2
File	53,524 bytes (52.3 KB) · the PDF
Notes	`Info scriptsRan`: The page's scripts ran (133 ms of virtual time, 5 tasks; 0 canvases drawn).

ScrollThrough

```
var result = HtmlToPdf.Render(html, new HtmlPdfOptions { ScrollThrough = true });
```



Pages 6
File 58,677 bytes (57.3 KB) · the PDF
Notes `Info scriptsRan`: The page's scripts ran (883 ms of virtual time, 40 tasks; 0 canvases drawn).

Scrolling to a section first

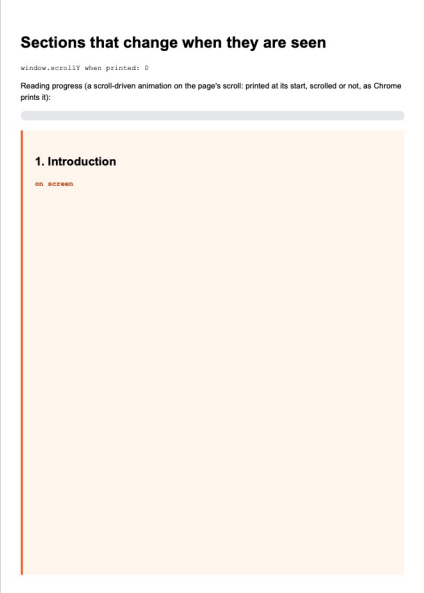
ScrollTo scrolls the window before printing, as a user would before pressing Print: the page's scroll events and IntersectionObserver callbacks see it (here the sections that went off screen, and scrollY). A scroll timeline on the page's own scroll prints at its start either way, as Chrome prints it. The PDF still holds the whole document from its top.

Reference: `HtmlPdfOptions.ScrollTo`

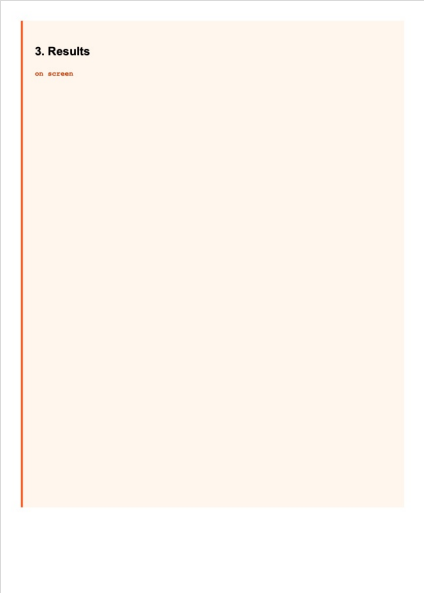
▶ The page printed: `scroll-effects.html`

Default: not scrolled

```
var result = HtmlToPdf.Render(html);
```







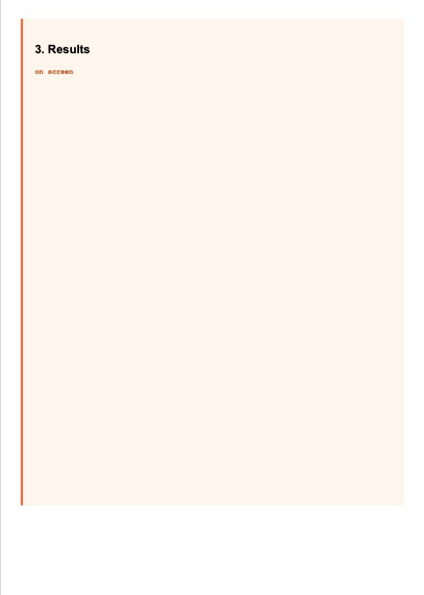
Pages	3
File	69,331 bytes (67.7 KB) · the PDF
Notes	`Info scriptsRan`: The page's scripts ran (17 ms of virtual time, 2 tasks; 0 canvases drawn).

ScrollTo "#results"

```
var result = HtmlToPdf.Render(html, new HtmlPdfOptions { ScrollTo = "#results" });
```







Pages	3
File	70,553 bytes (68.9 KB) · the PDF
Notes	`Info scriptsRan`: The page's scripts ran (33 ms of virtual time, 4 tasks; 0 canvases drawn).

Printing a part of the page

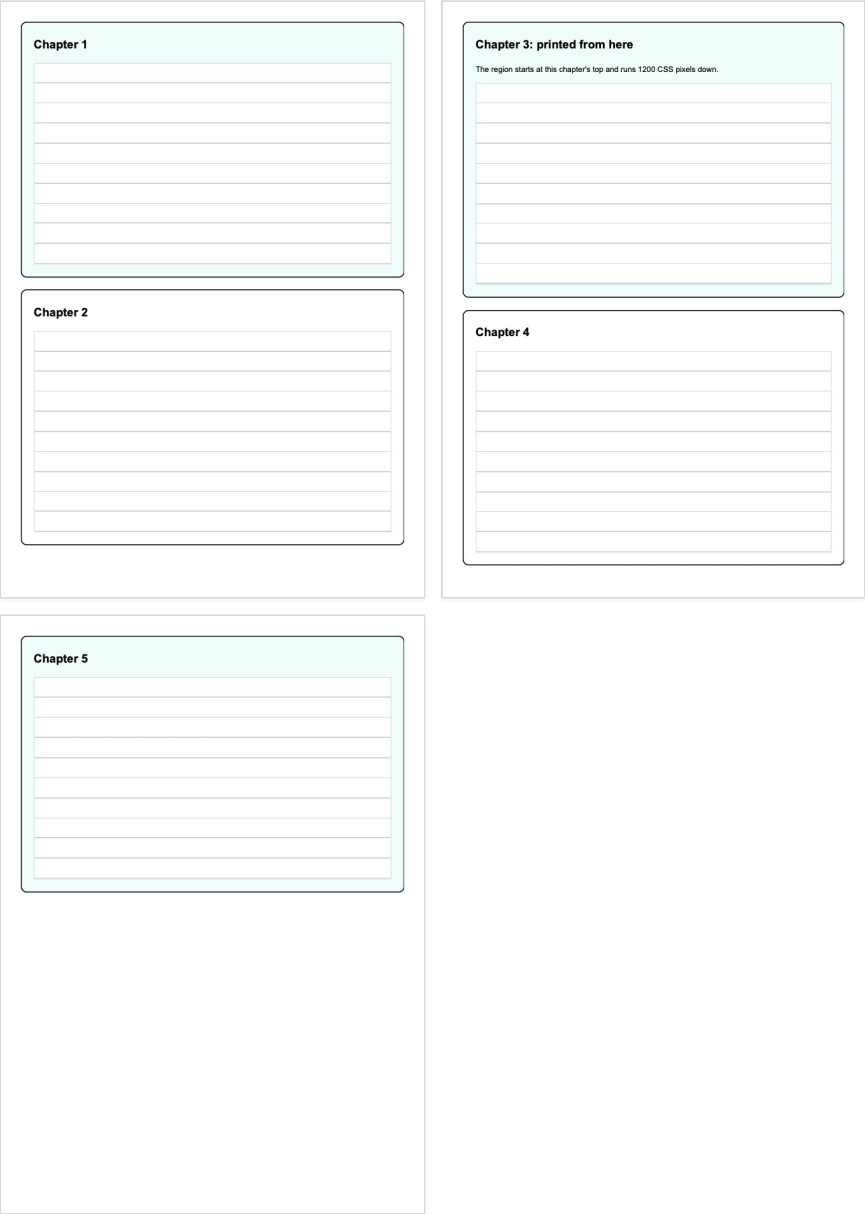
Region prints a part of the document, from a start (the scroll position, a length, an element) for a height; what lies above and below is left out, and the part is laid out on pages as a whole document is.

Reference: `HtmlPdfOptions.Region`, `HtmlPrintRegion`

▶ The page printed: `chapters.html`

Default: the whole document

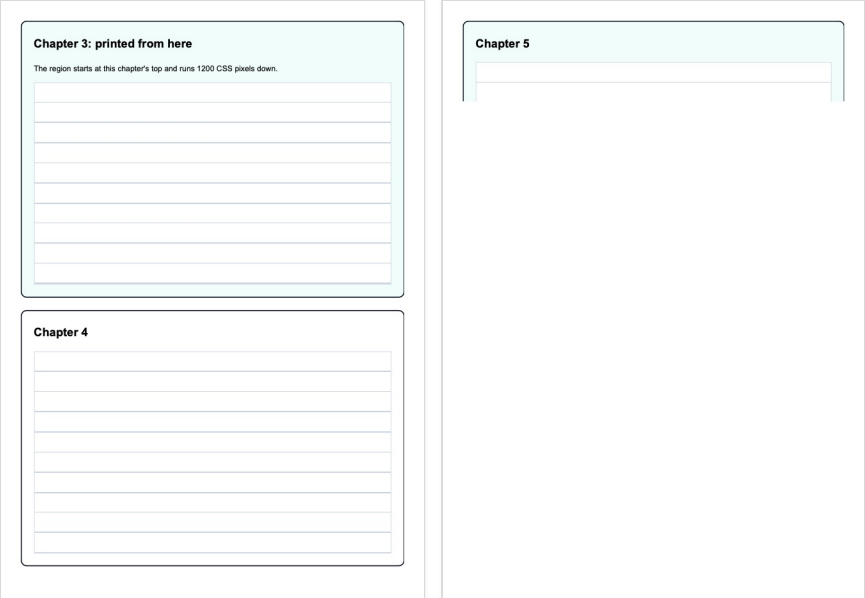
```
var result = HtmlToPdf.Render(html);
```



Pages 3
File 46,933 bytes (45.8 KB) · the PDF

1200 px from #chapter-3

```
var result = HtmlToPdf.Render(html, new HtmlPdfOptions { Region =
HtmlPrintRegion.Of("#chapter-3", "1200px") });
```



Pages 2
File 56,560 bytes (55.2 KB) · the PDF

Chrome's experimental features

Off by default, as stable Chrome prints; on, the features Chrome 156 has behind its "Experimental Web Platform features" flag that Weblitz has built, grid lanes among them.

Reference: `HtmlPdfOptions.ExperimentalWebFeatures`

▶ The page printed: `grid-lanes.html`

Default: as stable Chrome

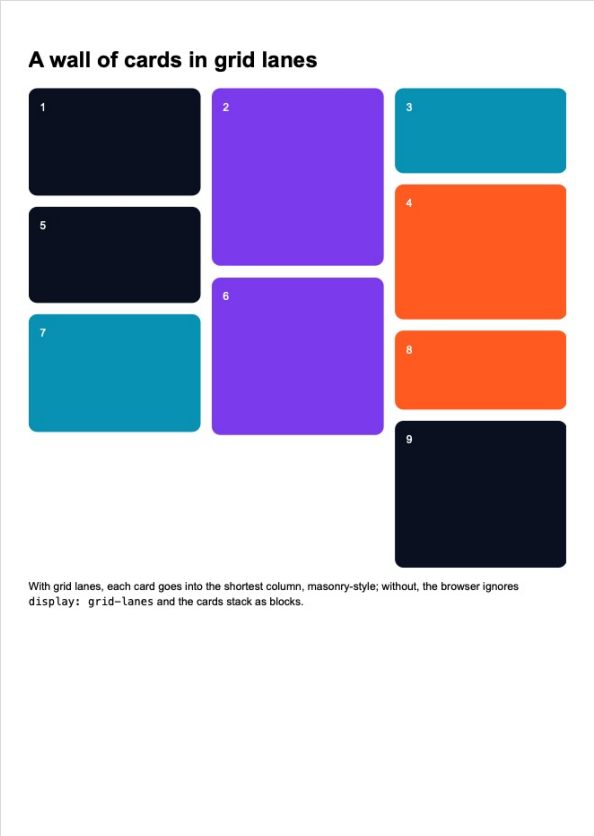
```
var result = HtmlToPdf.Render(html);
```



Pages 2
File 55,360 bytes (54.1 KB) · the PDF

ExperimentalWebFeatures = true

```
var result = HtmlToPdf.Render(html, new HtmlPdfOptions { ExperimentalWebFeatures = true });
```



Pages 1
File 55,515 bytes (54.2 KB) · the PDF

Printing documentation without its menus

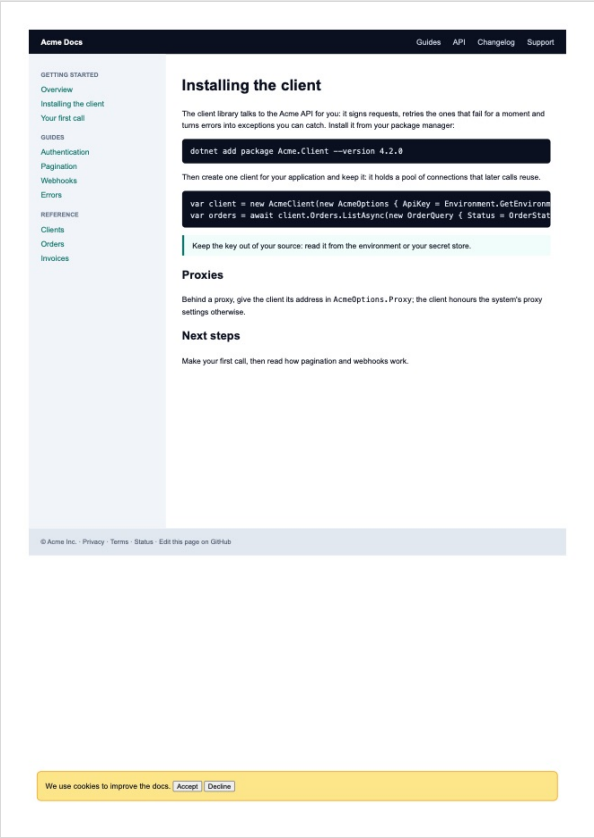
A documentation page without print styles of its own prints its top bar, sidebar, footer and cookie banner. Hide leaves elements out by CSS selector (each a display: none !important rule); Styles adds CSS of your own, here a print style that lets the text take the page's width and wraps long code lines. Both are added as Puppeteer's page.addStyleTag adds CSS before page.pdf(): once the page's scripts have settled, before beforeprint.

Reference: `HtmlPdfOptions.Hide`, `HtmlPdfOptions.Styles`

▶ The page printed: `docs-page.html`

Default: the page as it is

```
var result = HtmlToPdf.Render(html);
```



Pages	1
File	67,355 bytes (65.8 KB) · the PDF
Notes	`Info contentShrunk`: the document is 1052 px wide on a 720 px page: printed at 68.44% (laid out 1052 px wide). `Info wideElement`: reaches 1052 px, past the 720 px page.

Hide and Styles

```
var result = HtmlToPdf.Render(html, new HtmlPdfOptions
{
    Hide = ["nav", ".sidebar", "footer", "#cookie-banner"],
    Styles = ["@media print { main { max-width: none } pre { white-space: pre-wrap }
}"],
});
```



Installing the client

The client library talks to the Acme API for you: it signs requests, retries the ones that fail for a moment and turns errors into exceptions you can catch. Install it from your package manager:

```
dotnet add package Acme.Client --version 4.2.0
```

Then create one client for your application and keep it: it holds a pool of connections that later calls reuse.

```
var client = new AcmeClient(new AcmeOptions { ApiKey =
Environment.GetEnvironmentVariable("ACME_KEY") });
var orders = await client.Orders.ListAsync(new OrderQuery { Status =
OrderStatus.Open, PageSize = 50 });
```

> Keep the key out of your source: read it from the environment or your secret store.

Proxies

Behind a proxy, give the client its address in `AcmeOptions.Proxy`; the client honours the system's proxy settings otherwise.

Next steps

Make your first call, then read how pagination and webhooks work.

Pages 1
File 58,225 bytes (56.9 KB) · the PDF

Without the consent boxes

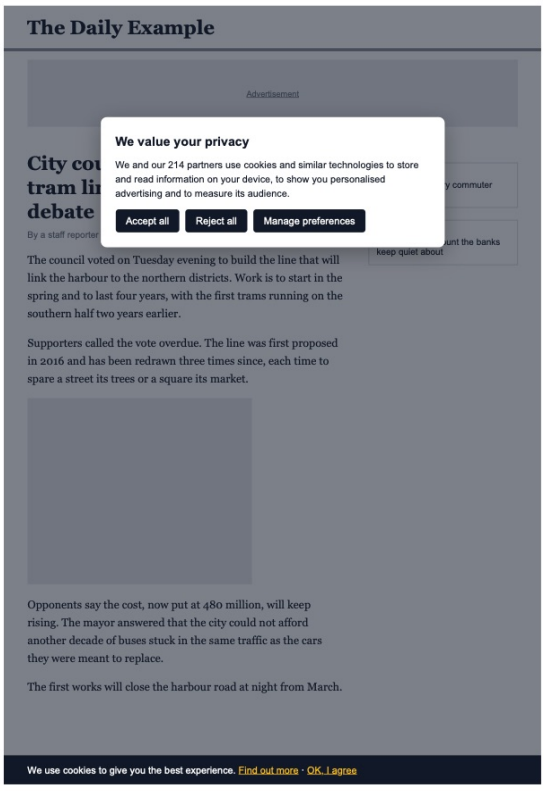
A page from the web often comes with a consent dialog over it and a cookie bar under it. Consent = Hide leaves them out of the print without clicking anything: the page and its consent platform run as they come, no answer is given in the reader's name. It hides the consent platforms' own boxes, the boxes the EasyList Cookie List names, and a box of the page's own (fixed to the window, or a dialog, that speaks of cookies, consent or privacy and holds a button to answer with), with the veil behind it and the lock it puts on the page's scrolling. A consentHidden note says what was hidden.

Reference: `HtmlPdfOptions.Consent`

▶ The page printed: `news-page.html`

Default: the page as it comes

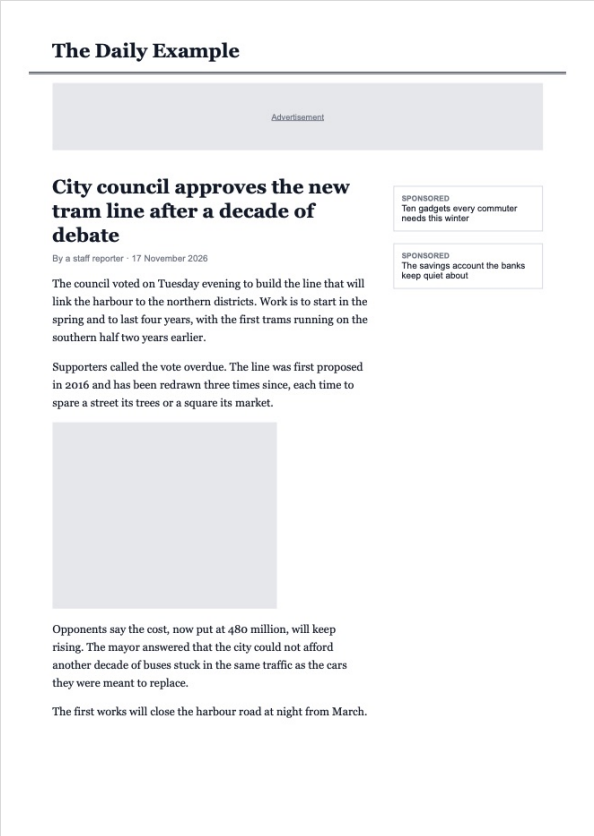
```
var result = HtmlToPdf.Render(html);
```



Pages	1
File	100,219 bytes (97.9 KB) · the PDF
Notes	`Info resourceRefused`: Script https://www.google-analytics.com/analytics.js: no resource loader was given (only data: URLs load). `Info resourceRefused`: Frame https://securepubads.g.doubleclick.net/gampad/adx?slot=inline: no resource loader was given (only data: URLs load). `Info scriptsRan`: The page's scripts ran (17 ms of virtual time, 2 tasks; 0 canvases drawn).

Consent = Hide

```
var result = HtmlToPdf.Render(html, new HtmlPdfOptions { Consent = HtmlConsent.Hide
});
```



Pages	1
File	70,593 bytes (68.9 KB) · the PDF
Notes	`Info resourceRefused`: Script https://www.google-analytics.com/analytics.js: no resource loader was given (only data: URLs load). `Info resourceRefused`: Frame https://securepubads.g.doubleclick.net/gampad/adx?slot=inline: no resource loader was given (only data: URLs load). `Info consentHidden`: Consent boxes are not printed (Consent = Hide): the page's own div#consent, the page's own div#cookie-bar. `Info scriptsRan`: The page's scripts ran (33 ms of virtual time, 6 tasks; 0 canvases drawn).

Without advertisements and trackers

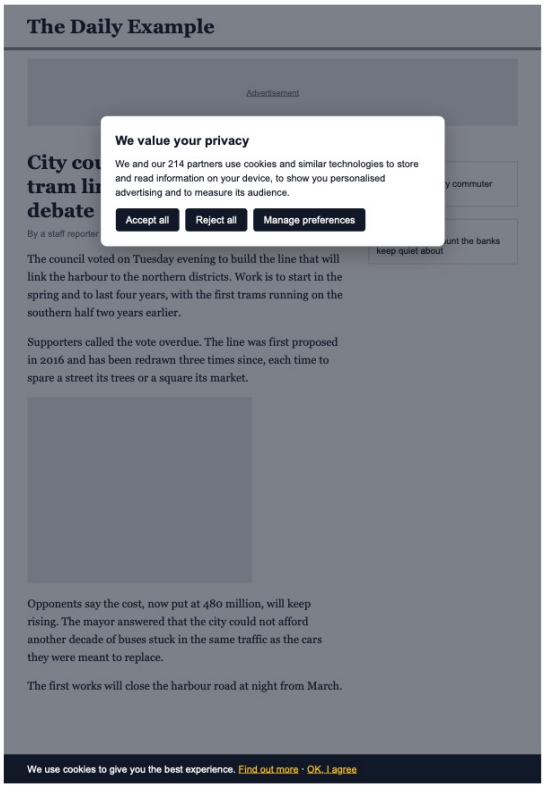
Ads = Block prints a page as an ad blocker shows it, by the filter lists EasyList and EasyPrivacy (the work of the EasyList authors, a snapshot of which the package carries): a request the lists block is not made, an element they hide is not printed, and a frame whose request was refused leaves no empty box. AdFilters adds rules of your own in the same syntax, here for the page's sponsored boxes. An adsBlocked note counts what was left out. Here with Consent = Hide as well.

Reference: `HtmlPdfOptions.Ads`, `HtmlPdfOptions.AdFilters`, `HtmlToPdf.UpdateFilterListsAsync`

▶ The page printed: `news-page.html`

Default: the page as it comes

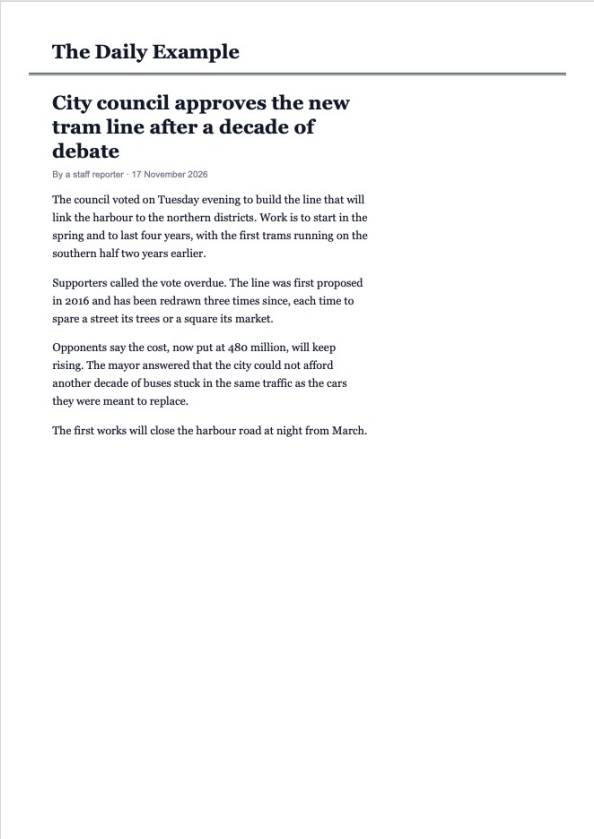
```
var result = HtmlToPdf.Render(html);
```



Pages	1
File	100,219 bytes (97.9 KB) · the PDF
Notes	`Info resourceRefused`: Script https://www.google-analytics.com/analytics.js: no resource loader was given (only data: URLs load). `Info resourceRefused`: Frame https://securepubads.g.doubleclick.net/gampad/adx?slot=inline: no resource loader was given (only data: URLs load). `Info scriptsRan`: The page's scripts ran (17 ms of virtual time, 2 tasks; 0 canvases drawn).

Ads = Block, with a rule of our own

```
var result = HtmlToPdf.Render(html, new HtmlPdfOptions
{
    Ads = HtmlAds.Block,
    AdFilters = ["##.sponsored"],
    Consent = HtmlConsent.Hide,
});
```



The Daily Example

City council approves the new tram line after a decade of debate

By a staff reporter · 17 November 2026

The council voted on Tuesday evening to build the line that will link the harbour to the northern districts. Work is to start in the spring and to last four years, with the first trams running on the southern half two years earlier.

Supporters called the vote overdue. The line was first proposed in 2016 and has been redrawn three times since, each time to spare a street its trees or a square its market.

Opponents say the cost, now put at 480 million, will keep rising. The mayor answered that the city could not afford another decade of buses stuck in the same traffic as the cars they were meant to replace.

The first works will close the harbour road at night from March.

Pages	1
File	52,292 bytes (51.1 KB) · the PDF
Notes	`Info consentHidden`: Consent boxes are not printed (Consent = Hide): the page's own div#consent, the page's own div#cookie-bar. `Info scriptsRan`: The page's scripts ran (33 ms of virtual time, 6 tasks; 0 canvases drawn). `Info adsBlocked`: Advertisements and trackers are left out (Ads = Block): 2 requests not made, 168 of the lists' hiding rules given to the page.