Scrolling and printing a part

Lazy content prints, with no option to set. Without a scroll option (neither ScrollTo nor ScrollThrough), a page's IntersectionObserver reports every element it observes visible, once, wherever it sits on the page. So what a page loads as it comes into view, lazy pictures, charts drawn when they are reached, items its scripts add, is loaded and printed: the whole page is in the PDF. Chrome reports only what is in its window, so with Puppeteer such a page prints its first screen unless a script scrolls it first.

To print as Chrome does, with only the first screen's lazy content loaded, set a scroll option: `new HtmlPdfOptions { ScrollTo = "top" }`. The observer then reports real intersections with the window, and what lies below the first screen prints as the page left it, as in Chrome's "Save as PDF" or Puppeteer's `page.pdf()`. This is a choice, not a fidelity setting: either way the page is laid out and drawn as Chrome draws it.

Some pages want real scrolling: a list that loads more each time its bottom comes into view, effects that follow the scroll, a page opened at a section. Four options scroll the window before printing, as a user would before pressing Print, or print a part of the page. With a scroll option set, Weblitz reports real intersections with the window, as Chrome does.

- ScrollThrough: scroll from the top to the end, a window height at a time, so what loads on the way is printed (200 steps at most).
- ScrollTo: scroll to `"top"`, `"bottom"`, an element (a CSS selector) or a length, once the scripts have settled; the page's scroll events and IntersectionObserver callbacks see it. A scroll timeline on the page's own scroll prints at its start, scrolled or not, as Chrome prints it.
- ScrollStepWait: the virtual time each scroll gives the page to react (0.5 s by default, 0 to 60 s).
- Region: print a part of the document (HtmlPrintRegion), from the scroll position, a length or an element, for a height; laid out on pages as a whole document is.

Scrolling alone never crops: the PDF holds the whole document from its top, as Chrome's print of a scrolled page does. Only Region leaves parts out.

The examples below show both: the infinite list prints twenty orders by default, as its observer reports its bottom once, and all sixty with ScrollThrough.

Lists that load as you scroll

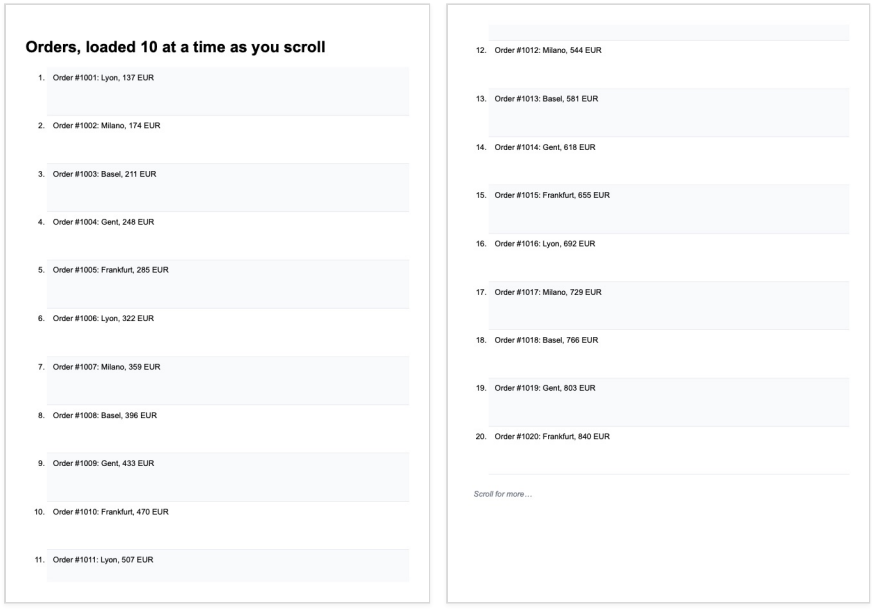
An infinite list loads its next ten orders each time its bottom comes into view. Without a scroll option, Weblitz's IntersectionObserver reports the bottom visible once, so the PDF holds twenty; ScrollThrough scrolls the window down a window height at a time, each step waiting ScrollStepWait of virtual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

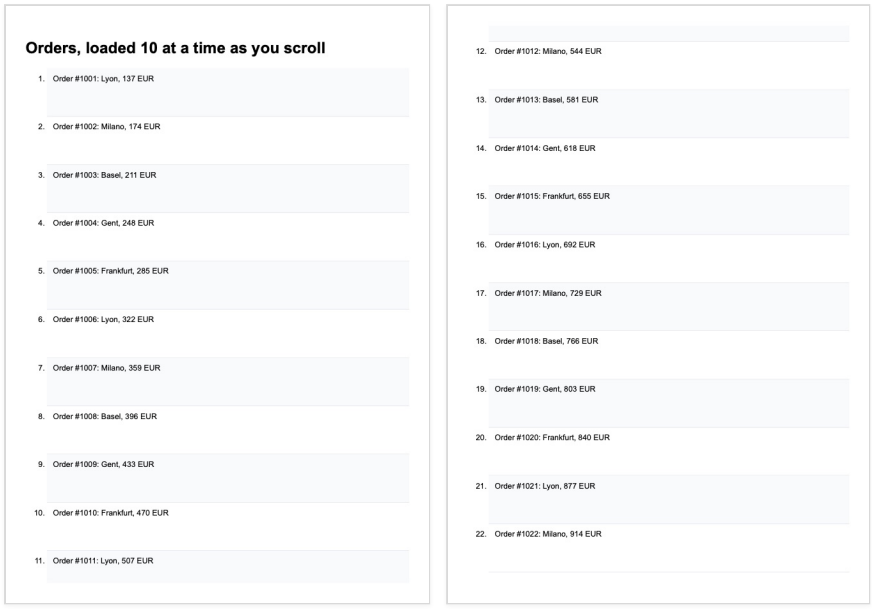
```
var result = HtmlToPdf.Render(html);
```



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 });
```



Orders, loaded 10 at a time as you scroll

1. Order #1001: Lyon, 137 EUR
2. Order #1002: Milano, 174 EUR
3. Order #1003: Basel, 211 EUR
4. Order #1004: Gent, 248 EUR
5. Order #1005: Frankfurt, 285 EUR
6. Order #1006: Lyon, 322 EUR
7. Order #1007: Milano, 359 EUR
8. Order #1008: Basel, 396 EUR
9. Order #1009: Gent, 433 EUR
10. Order #1010: Frankfurt, 470 EUR
11. Order #1011: Lyon, 507 EUR
12. Order #1012: Milano, 544 EUR
13. Order #1013: Basel, 581 EUR
14. Order #1014: Gent, 618 EUR
15. Order #1015: Frankfurt, 655 EUR
16. Order #1016: Lyon, 692 EUR
17. Order #1017: Milano, 729 EUR
18. Order #1018: Basel, 766 EUR
19. Order #1019: Gent, 803 EUR
20. Order #1020: Frankfurt, 840 EUR
21. Order #1021: Lyon, 877 EUR
22. Order #1022: Milano, 914 EUR

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

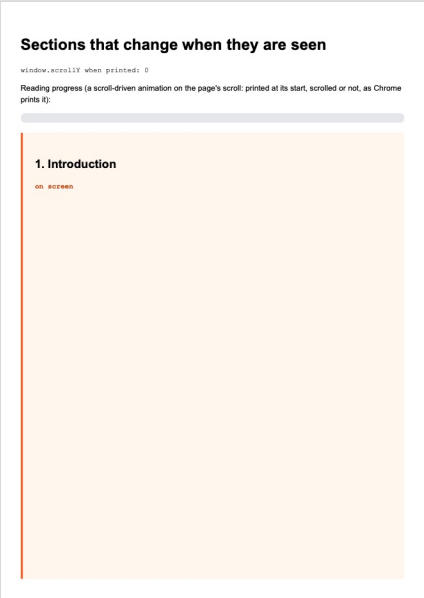
ScrollTo scrolls the window before printing, as a user would before pressing Print: the page's scroll events and IntersectionObserver callbacks see it (here the sections that went off screen, and scrollY). A scroll timeline on the page's own scroll prints at its start either way, as Chrome prints it.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

Reference: `HtmlPdfOptions.ScrollTo`

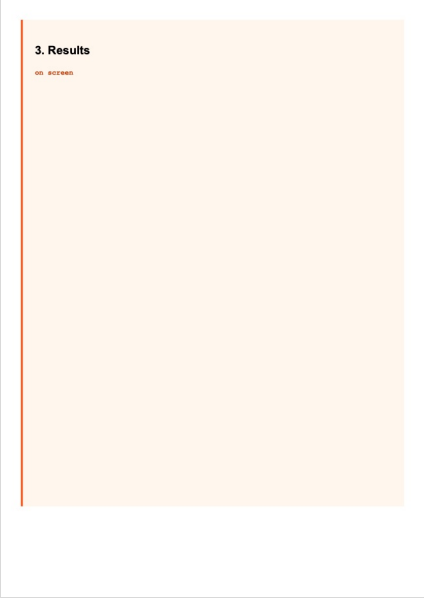
▶ The page printed: `scroll-effects.html`

Default: not scrolled

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







Pages 3

File 69,331 bytes (67.7 KB) · the PDF

Notes

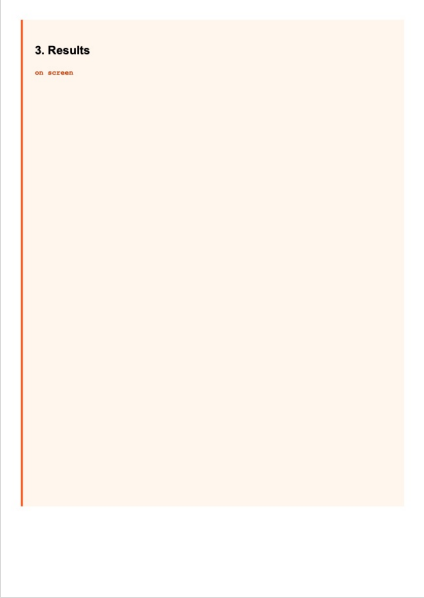
- `Info scriptsRan`: The page's scripts ran (17 ms of virtual time, 2 tasks; 0 canvases drawn).

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

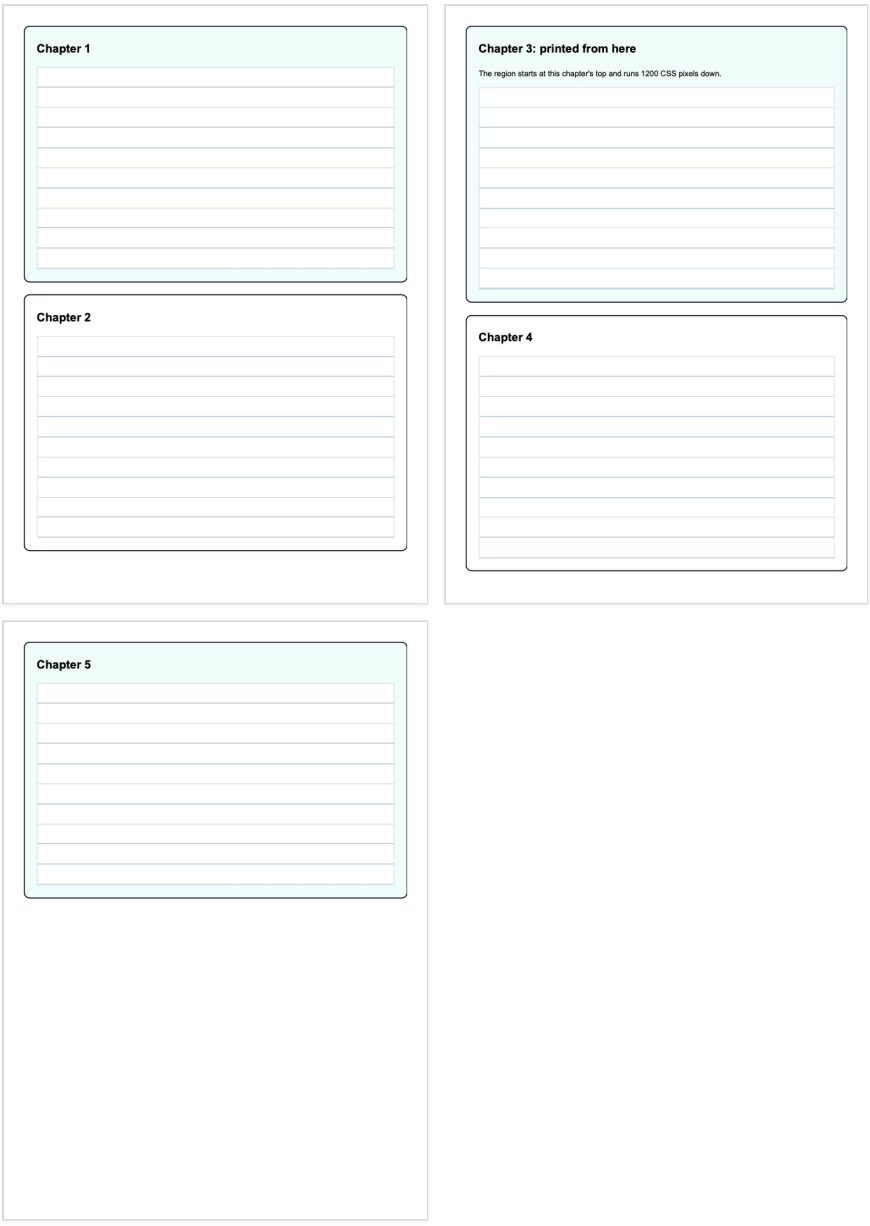
Region prints a part of the document, from a start (the scroll position, a length, an element) for a height; what lies above and below is left out, and the part is laid out on pages as a whole document is.

Reference: `HtmlPdfOptions.Region`, `HtmlPrintRegion`

▶ The page printed: `chapters.html`

Default: the whole document

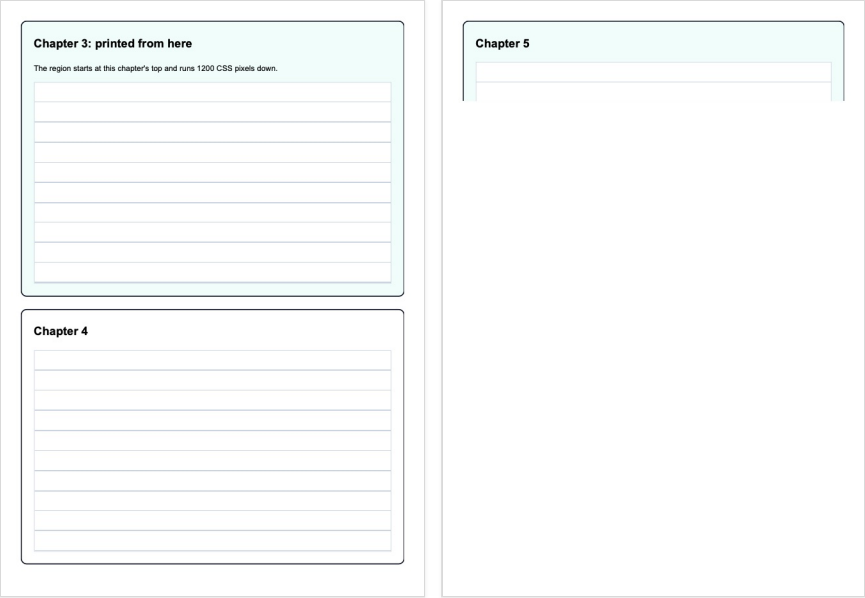
```
var result = HtmlToPdf.Render(html);
```



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