Printing like Chrome

Weblitz prints a page the way Chrome's "Save as PDF" does: the page is first loaded as on a screen, its scripts run, then it is printed. Knowing the order explains what a page sees and when.

From HTML to PDF

1. **Load.** The HTML is parsed, its stylesheets, pictures and fonts are loaded (from the file's folder, or the network if you allow it), and the page is laid out in *screen* media.
2. **Scripts.** The page's scripts run once, at the width and height of the printed page area (the paper less its margins), as they would in the window Chrome prints from. Timers and animations run in virtual time, up to HtmlScripts.WaitFor.
3. **beforeprint.** The page's `beforeprint` handlers run: a page can change itself for print there.
4. **Print media.** The media switches to `print` (`@media print` rules apply) and the document is laid out on pages: the paper, the margins, `@page` rules, page breaks.
5. **The PDF** is written, then `afterprint` runs (it changes nothing in the PDF).

The scripts example shows it: text added as the page loads, by a timer, and on `beforeprint`.

Defaults, beside Puppeteer's

The options carry Puppeteer's `page.pdf()` names, so a call moved over from Puppeteer reads the same. Four defaults differ: three chosen to print as the engine is measured against Chrome, and one so that the PDF holds the whole page:

- Format is A4 (Puppeteer's is Letter).
- PrintBackground is true: backgrounds are printed (Puppeteer's default leaves them out).
- PreferCssPageSize is true: a page's own `@page` size wins over Format.

- **Lazy content is printed.** Without a scroll option, a page's IntersectionObserver reports every element it observes visible, so lazy pictures, charts drawn when they come into view and items added as the list is reached are all in the PDF. Chrome, and so Puppeteer, reports only what is in the first screen and prints the rest as the page left it: placeholders, spinners, empty boxes. Set ScrollTo to `"top"` to print as Chrome does (Scrolling).

Set them in the options if your pages expect Puppeteer's.

Videos

A `<video>` prints its poster, or the frame at its current time once it plays, as Chrome prints it. Playback runs on the page's virtual time: a page that plays a video, then prints, shows the frame at that time. VP8, VP9 and AV1 are decoded by Weblitz itself. H.264 and HEVC, the usual codecs of `.mp4` files, are decoded with the machine's own codec, which Weblitz does not ship: macOS has it; Windows desktop editions have it, and Windows Server once its Media Foundation feature is added (`Install-WindowsFeature Server-Media-Foundation`; HEVC needs Microsoft's HEVC Video Extensions); Linux needs FFmpeg's libavcodec (`apt-get install -y libavcodec61`, or your release's version; `apk add ffmpeg-libavcodec` on Alpine) or, for H.264, Cisco's OpenH264. Without one, such a video prints its poster, and the render's notes say which codec to install.

What it is not

Weblitz prints; it does not browse. There is no window to click or type in: content a page shows only after a user's clicks needs to be in its HTML (or made by its scripts) to print. Pages behind a login print with what a browser would send, headers and bearer tokens, cookies, Basic and Digest (pages behind a login), and the scroll options scroll the window before printing (scrolling). Pages are printed once, in their state after the scripts' virtual time.