Scripts and the virtual clock

A page's scripts run once, before printing, in V8 (Chrome's JavaScript engine, built into the package): pages built by React or Vue, charts drawn by Chart.js, tables filled from the page's own data. The scripts of the page's frames run with them. HtmlScripts.Off runs none.

The virtual clock

Timers and animations run in virtual time: a page that waits two seconds for a timer does not make the render wait two seconds. The page gets WaitFor of virtual time (5 seconds by default) before it is printed; a timer set for later does not run. Timeout is different: the real processor time scripts may take (30 seconds by default) before they are stopped and the page printed as it is. See scripts and the clock.

```
var options = new HtmlPdfOptions
{
    Scripts = HtmlScripts.Default with { WaitFor = TimeSpan.FromSeconds(10), Timeout =
TimeSpan.FromSeconds(5) },
};
```

Animations and transitions

A print shows one moment of a page, so the frames of an animation in between are work nobody sees. By default (Animations = `HtmlPdfAnimations.Finished`) Weblitz takes each CSS animation, transition and `Element.animate()` at its end while the page's scripts run, and one that never ends, a spinner, at its first moment. The printed page is the one Chrome prints. The events (`transitionend`, `animationend`) are sent at their times and `Animation` objects read their real times, so a page that waits for an animation to end waits as long as in Chrome.

One difference from Chrome is on purpose: a script that reads a style or a size in the middle of an animation reads its end value, where Chrome's reads the value at that frame.

- `HtmlPdfAnimations.FrameByFrame` runs them as Chrome does, for a page whose scripts depend on values read in mid-animation. It takes longer on pages with many animated elements.
- `HtmlPdfAnimations.None` switches CSS animations and transitions off, as the rule `* { animation: none !important; transition: none !important }` many applications inject into Chrome before printing. No animation or transition event is sent, so a page that waits for one to show its content may print without it.

```
var options = new HtmlPdfOptions { Animations = HtmlPdfAnimations.FrameByFrame };
```

The same PDF every time

What scripts read of the world is fixed by the options: Locale (en-US), TimeZone (UTC) and Now; `Date` and `Intl` both read the time zone. Give them, and the same input prints the same PDF: see locale and clock.

Canvas text and web fonts

A chart drawn on a canvas writes its text with the fonts loaded when it draws. Draw after your web fonts are loaded, the weight you use included:

```
Promise.all([document.fonts.load("400 12px 'Your Font'"),
document.fonts.ready]).then(drawCharts);
```

The browser a page sees

A page sees desktop Chrome 156 on the machine's own system (`navigator.userAgent`, `navigator.userAgentData`, and the User-Agent header when the network is on). UserAgent and UserAgentMetadata change it: see the browser a page sees.

The page is also told that a program drives its browser: `navigator.webdriver` is true, as in Chrome under Puppeteer or Playwright. Analytics and trackers that leave automated browsers out then stay off, so a render makes fewer requests, runs less script, and is not counted as a visit. Some sites show an automated browser a check, or less of their page: Automated = `false` prints as an ordinary visitor's browser would. The user agent says desktop Chrome either way.

Since WeblitzPdf 2.0.0-beta.4; in 2.0.0-beta.3 and before, `navigator.webdriver` is false.