Printing in Docs/Sheets/Slides

SHARED EXTERNALLY

luizp@google.com
ikopylov@google.com

Last Updated: October 2019

Overview

Currently Docs/Sheets/Slides do a combination of client-side and server-side printing.

Client printing is preferred over server printing:

1. It's much faster. Server printing requires relayout of the document on the server-side, as well as shipping of all the resources (e.g. images) back down to the client.
2. It ensures that the printed page matches the on-screen layout. This can vary across browser/platform due to font rendering differences.
3. It works offline.

Client-side Printing

Client-side printing is currently only done by Docs on Chrome.

Docs currently renders its content to HTML. In order to print, it uses print media CSS to hide the UI and position the content appropriately. It uses an @page rule to set the page size, remove printer margins, and hide browser header/footers.

Server-side Printing

Printing is always done server-side for Sheets and Slides, which use Java server-side rendering logic to render to PDF. For non-Chrome browsers, printing is also done server-side for Docs.

Ideal Long-Term State

1. *Client rendering on Canvas.* Currently we use a mix of HTML, SVG and Canvas for rendering and would like to unify on Canvas for better cross-platform support, simplified shared logic, and performance.
2. *Client printing via Canvas.* Once there is a consistent rendering surface, ideally it could be used for both local PDF generation and local printing.

Requests for Chrome

[done] P0 Printing via Canvas

Canvas surfaces currently print as an image, which is problematic for client printing:

1. Resolution differences between the screen dpi and printer dpi could cause the content to be blurry.
2. Results in large printed documents, which is particularly problematic when printing to PDF.
3. Text is rendered as an image when printing to PDF and is not selectable/interactive or searchable.
4. Hyperlinks are lost when printing to PDF.
5. Entire document content needs to be rendered to one or more canvases at print time, which has memory / performance considerations.

A fix was implemented in crbug/959357, which saves any canvas commands during beforeprint to a display list, and plays them back to the PDF. This addresses the issue, but an explicit API for producing PDF, would still be the ideal solution.

P1 Printing mixed page size/orientation

We'd like to be able to produce PDFs with different page sizes and orientations. In order to do this, we need some way to represent that in the browser. Could be through @page rules or through the printing API discussed below.

P2 Accessible PDFs

In order to generate accessible PDFs from the printed content, the document needs to be tagged with additional metadata (e.g. alt text for images, document structure tags for paragraphs / tables, etc).

P2 Customization of settings in print dialog

Changes that users make in the Print Dialog can sometimes produce unexpected results. (e.g. crbug/716883).
In the short term, would like to handle client printing in Docs in the same way as PDFs. This would remove some dialog options (e.g. margins, headers/footers) that cause unexpected results. It would also make behavior consistent between Doc/Sheets/Slides, since client and server printing would be treated the same way.
Longer term, would like to either be able to customize the dialog, or print directly from the app.

P3 Print API

Ideally, we could produce PDFs using a canvas-like API. This is sketched out in the appendix below. This would allow PDFs to be produced on demand, and avoid having to control PDF properties indirectly through HTML/CSS.

P3 Offline Download as PDF

Today, downloading a document as PDF from within Docs (and not via browser print) requires a round trip to the server which lays out the document from scratch, produces a PDF and sends it back down to the client. This requires the user to be online and is wasteful in terms of bandwidth and repeated layout work. Ideally, the client would be able to generate a PDF directly, via the same mechanism as printing.

Appendix A: Ideal browser API for Canvas printing

Ideally there would be a print-specific API allowing developers to issue low-level printing instructions. One possibility would be to allow the creation of a print-specific Canvas2D Context with new APIs to start/end pages at custom sizes and to generate hyperlink rects.

iOS and Android have similar solutions via creation of a PDF context for print.[1]

	iOS	**Android**
Initialization	Create/terminate PDF context via UIGraphicsBeginPDFContext and UIGraphicsEndPDFContext	Create a PDFDocument or PrintedPDFDocument
Start/finish pages	Via UIGraphicsBeginPDFPage.	Via PDFDocument#startPage and #finishPage.
Mixed page sizes	Supported	Only supported when output to PDF (not when printing)
PDF context allows same instructions as screen context	Yes, active PDF context is a standard CGContext	Yes, using a standard Canvas from Page#getCanvas
Hyperlink to external URL	Yes, via UIGraphicsSetPDFContextURLForRect	Not supported
Hyperlink to internal position	Yes, via UIGraphicsAddPDFContextDestinationAtPoint and UIGraphicsSetPDFContextDestinationForRect	Not supported

[1] Additionally, both iOS and Android allow for querying of printer capabilities and direct control over print jobs, which may not be an initial requirement for this web print API.