Online publishing via pdf2htmlEX

Lu Wang and Wanmin Liu

Abstract

The Web has long become an essential part of our lives. While web technologies have been actively developed for years, there is still a large gap between web and traditional paper publishing. For example, the PDF format, the *de facto* standard for publishing, is not supported in the HTML standard; and the most powerful typesetting system, TeX, cannot be integrated perfectly.

Despite of the long history of people trying to convert TeX or PDF into HTML, some are focused on only a small fraction of features, *e.g.* text, formulas or images; some are too old to support new features in the HTML standard such as font embedding or linear transformations (*e.g.* rotation); some display everything in images at the cost of larger sizes.

In this article, while we survey and compare existing methods of publishing TeX or PDF documents online, a new approach is attempted to attack this issue. We introduce an open source program, called pdf2htmlEX, which is a general PDF to HTML converter and publishing tool with high fidelity. It presents PDF elements with corresponding native HTML elements, in order to achieve high accuracy and small size. The flexible design also makes it useful for a variety of use cases in online publishing. Obviously TeX users can immediately benefit with zero learning cost, just like `dvipdf` while people were still using DVI. More information is available at the home page:
`https://github.com/coolwanglu/pdf2htmlex`

1 Introduction

Arguably, for many people the World Wide Web *is* the Internet. Indeed, web technologies have been so actively developed in the past few years, nowadays web pages far surpass plain text and images. HTML5 brings audio, video, 3D graphics and many other rich features; CSS3 defines brand new visual effects, and JavaScript allows different kinds of user interactions. Modern web browsers are literally operating systems, and the boundary between web apps and local software has been blurred. Today, we can access the WWW with all kinds of devices such as watches, phones, tablets, computers and even glasses. It has become an essential part of our lives.

The web technologies provide brand new user experiences compared to traditional media. Taking Wikipedia as an example, it has *rich contents*: inside an article, besides plain text, there are often images, animations, audio and video that are relevant to the topic; it is *well organized*: users may jump to relevant articles by clicking links; it is *interactive*: users may create or edit an article; it is *personalized*: the appearance of the web site respects users' preferences such as language, theme or format; it is *social*: users may leave comments and have discussions regarding an article.

Compared with traditional publishing media, it is more convenient and easier for users to obtain, view and share the contents. While most features in HTML are targeting visual effects, multimedia and rich Internet applications, there is still a large gap between the Web and traditional publishing. Many existing publishing technologies cannot be perfectly integrated online — especially two of them focused on in this article, PDF and TeX, which are the most popular format and typesetting system respectively.

PDF The Portable Document Format, developed by Adobe, is one of the most popular formats for digital documents. PDF is known for its wide support of different types of fonts, encodings, raster images, vector graphics, and many other features from prepress processing to user interaction. It is widely supported in different operating systems and devices. Nowadays, almost all documents can be exported to PDF. Notably, with a virtual PDF printer, any document that can be printed on paper can be converted to PDF. It has become the *de facto* standard for academic articles, technical reports, manuals, newspapers and ebooks. As an example, the final format for *TUGboat* is PDF.

PDF is a print-ready format; it is designed to completely describe a fixed-layout flat document. A PDF file clearly defines the appearance of the document, independent of particular devices or viewers.

PDF is not supported in the HTML standard, but it can be viewed directly in several web browsers. Users of other web browsers usually have to read PDF documents with web browser plugins, or download the files and then read them with a local PDF reader. In all these cases, PDF files are viewed in a closed environment where users cannot utilize most web features.[1]

TeX Designed and written by Professor Donald Knuth, TeX is one of the most powerful typesetting systems in the world.[2] It is well-known for its capability of producing high quality formulas and figures

[1] PDF does include features such as external links and interactive functions within a document, but these are quite limited compared to HTML.

[2] When using 'TeX' in this article, most of the time we will be referring to the whole TeX family.

in many different areas. While it is most popular in academia, it is also used for typesetting books, magazines and sheet music.

TEX is a source format for *authors*. It contains structured contents including text, formulas, figures and possibly cross references between them. Users can define their own concepts by writing macros. Typically the layout of the document must be determined by *compiling* the file with a TEX compiler; different compilers may produce different results from the same TEX source file.

People started trying to connect TEX to the Web nearly since the Web began. There were some early overviews, such as [31, 32], [34, chapter 7], but we are not aware of any recent surveys on the topic. Early works were mainly focusing on correctly displaying formulas produced by TEX. Different methods include using images, Unicode characters, MathML or HTML5. However the power of TEX is far more than formulas, it is also famous for its capability of handling mathematical spacing, hyphenation and justification, which is often ignored in these cases.

In the following sections, we are going to describe and compare some popular existing approaches. We will also introduce a new program, pdf2htmlEX [25], and discuss its advantages and limitations with examples.

2 Preliminaries

The target audience of this article includes those who need to publish both online versions and print versions of their documents at the same time, especially those who want to publish existing documents online.

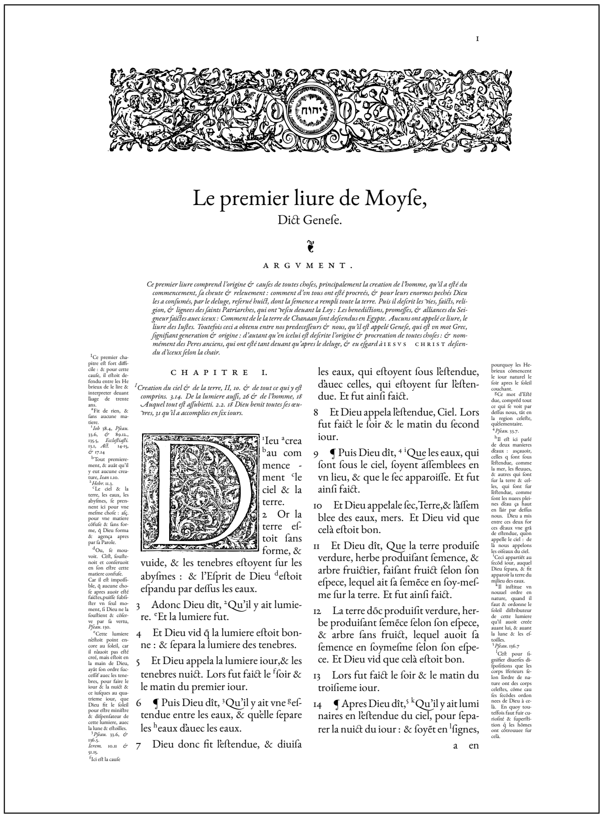
We assume that the existing document is in PDF format. It could be generated from TEX or any other tool. We do not assume that the publisher is the author, *i.e.* the source files may not be available to the publisher.

We believe that the following requirements are essential for most users. They are also the criteria we will use to discuss and compare existing approaches.

Convenience The publishing process should be automated, with minimal manual adjustments involved, such that publishers need focus on only one version, while the other can be generated accordingly.

Consistency Both the online version and the print version should have a consistent appearance, sometimes including the same layout and format.

Evidently the contents should never vary between the two versions, but one may argue that screen and paper are two completely different kinds of media, and so fonts, spacing and even layouts should be optimized individually. For example, users might want text in the document to be reflowed according to the screen size of their mobile devices.



1

Le premier liure de Moyſe,
Dict Geneſe.

ARGVMENT.

Ce premier liure comprend l'origine & cauſes de toutes choſes, principalement la creation de l'homme, qu'il a eſté du commencement, ſa cheute & releuement : comment d'vn tous ont eſté procreés, & pour leurs enormes pechés Dieu les a conſumés, par le deluge, reſerué huict, dont la ſemence a rempli toute la terre. Puis il deſcrit les vies, faicts, religion, & lignees des ſaints Patriarches, qui ont veſcu deuant la Loy : Les benedictions, promeſſes, & alliances du Seigneur faictes auec iceux : Comment de le la terre de Chanaan ſont deſcendus en Egypte. Aucuns ont appelé ce liure, le liure des Iuſtes. Toutefois ceci a obtenu entre nos predeceſſeurs & nous, qu'il eſt appelé Geneſe, qui eſt vn mot Grec, ſignifiant generation & origine : d'autant qu'en icelui eſt deſcrite l'origine & procreation de toutes choſes : & nommément des Peres anciens, qui ont eſté tant deuant qu'apres le deluge, & eu eſgard à IESVS CHRIST *deſcendu d'iceux ſelon la chair.*

CHAPITRE I.

[1] *Creation du ciel & de la terre, II, 10. & de tout ce qui y eſt comprins. 3.14. De la lumiere auſſi, 26 & de l'homme, 18 Auquel tout eſt aſſubietti. 2.2. 18 Dieu benit toutes ſes œuvres, 31 qu'il a accomplies en ſix iours.*

D[1]Ieu [a]crea [b]au commencement [c]le ciel & la terre.

2 Or la terre eſtoit ſans forme, & vuide, & les tenebres eſtoyent ſur les abyſmes : & l'Eſprit de Dieu [d]eſtoit eſpandu par deſſus les eaux.

3 Adonc Dieu dît, [2]Qu'il y ait lumiere. [e]Et la lumiere fut.

4 Et Dieu vid q̃ la lumiere eſtoit bonne : & ſepara la lumiere des tenebres.

5 Et Dieu appela la lumiere iour,& les tenebres nuict. Lors fut faict le [f]ſoir & le matin du premier iour.

6 ¶ Puis Dieu dît, [3]Qu'il y ait vne [g]eſtendue entre les eaux, & qu'elle ſepare les [h]eaux d'auec les eaux.

7 Dieu donc fit l'eſtendue, & diuiſa les eaux, qui eſtoyent ſous l'eſtendue, d'auec celles, qui eſtoyent ſur l'eſtendue. Et fut ainſi faict.

8 Et Dieu appela l'eſtendue, Ciel. Lors fut faict le ſoir & le matin du ſecond iour.

9 ¶ Puis Dieu dît, [4] [i]Que les eaux, qui ſont ſous le ciel, ſoyent aſſemblees en vn lieu, & que le ſec apparoiſſe. Et fut ainſi faict.

10 Et Dieu appelale ſec,Terre,& l'aſſemblee des eaux, mers. Et Dieu vid que celà eſtoit bon.

11 Et Dieu dît, Que la terre produiſe verdure, herbe produiſant ſemence, & arbre fruictier, faiſant fruict ſelon ſon eſpece, lequel ait ſa ſemẽce en ſoy-meſme ſur la terre. Et fut ainſi faict.

12 La terre dõc produiſit verdure, herbe produiſant ſemẽce ſelon ſon eſpece, & arbre ſans fruict, lequel auoit ſa ſemence en ſoymeſme ſelon ſon eſpece. Et Dieu vid que celà eſtoit bon.

13 Lors fut faict le ſoir & le matin du troiſieme iour.

14 ¶ Apres Dieu dît,[5] [k]Qu'il y ait luminaires en l'eſtendue du ciel, pour ſeparer la nuict du iour : & ſoyẽt en [l]ſignes,

a en

[1] Ce premier chapitre eſt fort difficile : & pour cette cauſe, il eſtoit defendu entre les Hebrieux de le lire & interpreter deuant l'aage de trente ans.

[a] Fit de rien, & ſans aucune matiere.

[1] *Iob* 38.4, *Pſeau.* 33.6, & 89.12., 135.5, *Eccleſtiaſti.* 13.1, *Act.* 14-15, & 17.14

[b] Tout premierement, & auãt qu'il y eut aucune creature, *Iean* 1.10.

[2] *Hebr.* 11.3.

[c] Le ciel & la terre, les eaux, les abyſmes, ſe prennent ici pour vne meſme choſe : aſç. pour vne matiere cõfuſe & ſans forme, q̃ Dieu forma & agença apres par ſa Parole.

[d] Ou, ſe mouvoit. C'eſt, ſouſtenoit et conſeruoit en ſon eſtre cette matiere confuſe. Car il eſt impoſſible, q̃ aucune choſe apres auoir eſté faictes,puiſſe ſubſiſter vn ſeul moment, ſi Dieu ne la ſouſtient & cõſerve par ſa vertu, *Pſeau.* 130.

[e] Cette lumiere n'eſtoit point encore au ſoleil, car il n'auoit pas eſté creé, mais eſtoit en la main de Dieu, ayãt ſon ordre ſucceſſif auec les tenebres, pour faire le iour & la nuict & ce iuſques au quatrieme iour, que Dieu fit le ſoleil pour eſtre miniſtre & diſpenſateur de cette lumiere, auec la lune & eſtoilles.

[3] *Pſeau.* 33.6, & 136.5. *Ierem.* 10.11 & 51.15.

[f] Ici eſt la cauſe pourquoy les Hebrieux cõmencent le iour naturel le ſoir apres le ſoleil couchant.

[g] Ce mot d'Eſtẽdue, comprẽd tout ce qui ſe voit par deſſus nous, tãt en la region celeſte, qu'elementaire.

[4] *Pſeau.* 33.7.

[h] Il eſt ici parlé de deux manieres d'eaux : aſçauoir, celles q ſont ſous l'eſtendue, comme la mer, les fleuues, & autres qui ſont ſur la terre & celles, qui ſont ſur l'eſtendue, comme ſont les nuees pleines d'eau ça haut en l'air par deſſus nous. Dieu a mis entre ces deux forces d'eaux vne grãde eſtendue, qu'on appelle le ciel : de là nous appelons les oiſeaux du ciel.

[i] Ceci appartiẽt au ſecõd iour, auquel Dieu ſepara, & fit apparoir la terre du milieu des eaux.

[k] Il inſtitue vn nouuel ordre en nature, quand il faut & ordonne le ſoleil diſtributeur de cette lumiere qu'il auoit creée auant lui, & auant la lune & les eſtoilles.

[5] *Pſeau.* 136.7

[l] C'eſt pour ſignifier diuerſes diſpoſitions que les corps ĩferieurs ſelon l'ordre de nature ont des corps celeſtes, cõme cauſes ſecõdes ordonnees de Dieu à celà. En quoy touteſfois faut fuir curioſité & ſuperſtition q̃ les hõmes ont cõtrouuee ſur celà.

Figure 1: Bible de Genève, 1564 [8], typeset by Raphaël Pinson with XƎTEX. The drop cap, fonts, spacing and layout were carefully tuned in order to duplicate the 16th century French Bible.

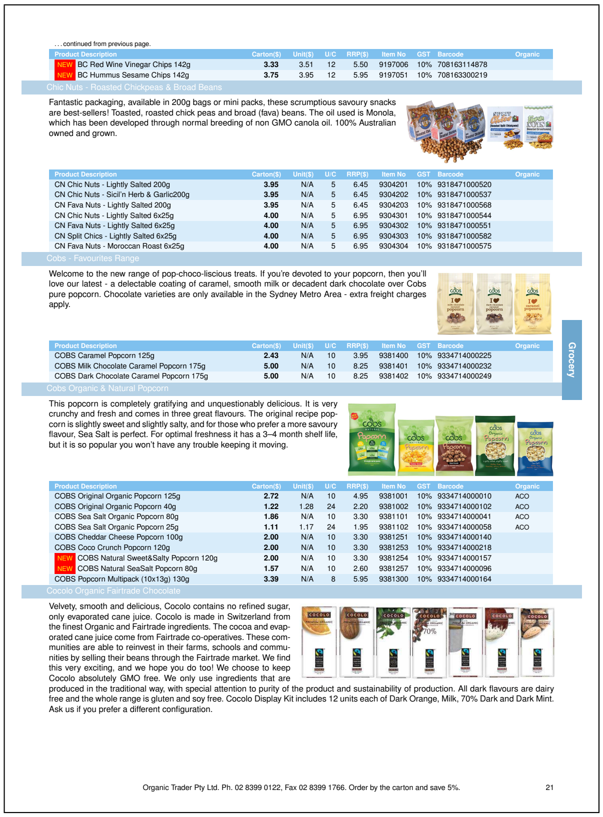
However, in many cases, fonts, spacing and layouts are carefully designed to assist reading, and sometimes they have already become essential parts of the document, see Figures 1 and 2 as examples. In such cases, a complete redesign may be involved in order to optimize for specific media.

In our opinion both situations are important, and we will try to cover both of them in this article.

Flexibility An important purpose of the online version is to provide better services and user experience. This version of the document should be flexible enough for front-end designers to design interactive web pages.

For example, text and other elements in the documents should be accessible such that extra styles or effects can be specified; the whole document should be able to be embedded into existing frameworks with well-defined behaviours and themes applied.

Optimization There are concerns for web services which may not be covered by traditional media: the



. . . continued from previous page.

Product Description	Carton($)	Unit($)	U/C	RRP($)	Item No	GST	Barcode	Organic
NEW BC Red Wine Vinegar Chips 142g	**3.33**	3.51	12	5.50	9197006	10%	708163114878	
NEW BC Hummus Sesame Chips 142g	**3.75**	3.95	12	5.95	9197051	10%	708163300219	

Chic Nuts - Roasted Chickpeas & Broad Beans

Fantastic packaging, available in 200g bags or mini packs, these scrumptious savoury snacks are best-sellers! Toasted, roasted chick peas and broad (fava) beans. The oil used is Monola, which has been developed through normal breeding of non GMO canola oil. 100% Australian owned and grown.

Product Description	Carton($)	Unit($)	U/C	RRP($)	Item No	GST	Barcode	Organic
CN Chic Nuts - Lightly Salted 200g	**3.95**	N/A	5	6.45	9304201	10%	9318471000520	
CN Chic Nuts - Sicil'n Herb & Garlic200g	**3.95**	N/A	5	6.45	9304202	10%	9318471000537	
CN Fava Nuts - Lightly Salted 200g	**3.95**	N/A	5	6.45	9304203	10%	9318471000568	
CN Chic Nuts - Lightly Salted 6x25g	**4.00**	N/A	5	6.95	9304301	10%	9318471000544	
CN Fava Nuts - Lightly Salted 6x25g	**4.00**	N/A	5	6.95	9304302	10%	9318471000551	
CN Split Chics - Lightly Salted 6x25g	**4.00**	N/A	5	6.95	9304303	10%	9318471000582	
CN Fava Nuts - Moroccan Roast 6x25g	**4.00**	N/A	5	6.95	9304304	10%	9318471000575	

Cobs - Favourites Range

Welcome to the new range of pop-choco-liscious treats. If you're devoted to your popcorn, then you'll love our latest - a delectable coating of caramel, smooth milk or decadent dark chocolate over Cobs pure popcorn. Chocolate varieties are only available in the Sydney Metro Area - extra freight charges apply.

Product Description	Carton($)	Unit($)	U/C	RRP($)	Item No	GST	Barcode	Organic
COBS Caramel Popcorn 125g	**2.43**	N/A	10	3.95	9381400	10%	9334714000225	
COBS Milk Chocolate Caramel Popcorn 175g	**5.00**	N/A	10	8.25	9381401	10%	9334714000232	
COBS Dark Chocolate Caramel Popcorn 175g	**5.00**	N/A	10	8.25	9381402	10%	9334714000249	

Cobs Organic & Natural Popcorn

This popcorn is completely gratifying and unquestionably delicious. It is very crunchy and fresh and comes in three great flavours. The original recipe popcorn is slightly sweet and slightly salty, and for those who prefer a more savoury flavour, Sea Salt is perfect. For optimal freshness it has a 3–4 month shelf life, but it is so popular you won't have any trouble keeping it moving.

Product Description	Carton($)	Unit($)	U/C	RRP($)	Item No	GST	Barcode	Organic
COBS Original Organic Popcorn 125g	**2.72**	N/A	10	4.95	9381001	10%	9334714000010	ACO
COBS Original Organic Popcorn 40g	**1.22**	1.28	24	2.20	9381002	10%	9334714000102	ACO
COBS Sea Salt Organic Popcorn 80g	**1.86**	N/A	10	3.30	9381101	10%	9334714000041	ACO
COBS Sea Salt Organic Popcorn 25g	**1.11**	1.17	24	1.95	9381102	10%	9334714000058	ACO
COBS Cheddar Cheese Popcorn 100g	**2.00**	N/A	10	3.30	9381251	10%	9334714000140	
COBS Coco Crunch Popcorn 120g	**2.00**	N/A	10	3.30	9381253	10%	9334714000218	
NEW COBS Natural Sweet&Salty Popcorn 120g	**2.00**	N/A	10	3.30	9381254	10%	9334714000157	
NEW COBS Natural SeaSalt Popcorn 80g	**1.57**	N/A	10	2.60	9381257	10%	9334714000096	
COBS Popcorn Multipack (10x13g) 130g	**3.39**	N/A	8	5.95	9381300	10%	9334714000164	

Cocolo Organic Fairtrade Chocolate

Velvety, smooth and delicious, Cocolo contains no refined sugar, only evaporated cane juice. Cocolo is made in Switzerland from the finest Organic and Fairtrade ingredients. The cocoa and evaporated cane juice come from Fairtrade co-operatives. These communities are able to reinvest in their farms, schools and communities by selling their beans through the Fairtrade market. We find this very exciting, and we hope you do too! We choose to keep Cocolo absolutely GMO free. We only use ingredients that are produced in the traditional way, with special attention to purity of the product and sustainability of production. All dark flavours are dairy free and the whole range is gluten and soy free. Cocolo Display Kit includes 12 units each of Dark Orange, Milk, 70% Dark and Dark Mint. Ask us if you prefer a different configuration.

Grocery

Organic Trader Pty Ltd. Ph. 02 8399 0122, Fax 02 8399 1766. Order by the carton and save 5%. 21

Figure 2: One page from a product catalogue generated with LaTeX. Text paragraphs, images and tables are well-organized for each category. The whole catalogue contains more than 70 pages, including information on 800–1000 products. Courtesy of Jason Lewis [35].

size of the files should be as small as possible in order to save storage space; the cache mechanism of web browsers should be utilized when possible, in order to save bandwidth; the readers should not need to wait long before viewing the first few pages, even if there are thousands of pages in the document.

Therefore special optimizations are necessary when producing an online version from a traditional document.

3 Existing approaches

Quite a number of approaches have been developed to publish TeX or PDF contents online. Possible workflows are shown in Figure 3. It is possible to *compile*[3] a TeX file into HTML; or to *convert*[4] a PDF document into HTML.

Converting a large TeX file with complicated layouts into PDF is usually not a fast process. Because of this, it is a common practice to convert the source format into web pages on the server side, the results can be stored on the servers, and sent to users upon request.

On the other hand, nowadays JavaScript is already powerful enough for many tasks, and it can be *embedded*[5] into HTML, in which case HTML is used as a container — the embedded files are to be parsed and rendered on the client side with JavaScript.

In order to utilize existing technologies, it is also common to introduce intermediate formats, which are to be converted or embedded into HTML. In particular, PDF may be viewed as an intermediate format while compiling TeX to HTML,

In this section we try to describe the most popular approaches and discuss them from different aspects. Although some of them might not be originally designed for publishing, they are still listed here because they can be used to facilitate the process.

3.1 Raster image-based approaches

Approaches of this type render source files into raster images (*e.g.* PNG, JPEG), usually one image per page, which are then embedded into HTML. Popular tools of this type include:

- pdftocairo from Poppler [27]
- ImageMagick [20]
- mathTeX [9]
- "fallback" mode of pdf2htmlEX

Pros Raster images were introduced in a very early stage of HTML, and so are highly compatible with old web browsers. All visual elements can be displayed correctly.[6]

Cons The main disadvantage of this type of approach is that the image sizes are usually huge. It is costly to convert text into images and it is usually not easy to balance quality and size. Large files consume large bandwidth of both server and client, which also cause delays. Another issue is that all semantic information is lost, users can no longer copy text out from the document, nor follow the links.

Raster image-based approaches are "universal", in that they are widely used to publish many different formats, not limited to TeX or PDF. Famous examples include the *Look Inside* feature of SpringerLink [11] and Google Docs Viewer [3].

[3] To determine the layout based on the information from the source.

[4] To transform between two presentation formats, in both of which layout and appearance are clearly defined.

[5] To keep the source format as it is inside the target format.

[6] For TeX and PDF, there are also advanced features like audio, video, animation or annotation, *etc.*, which are beyond the scope of this article.

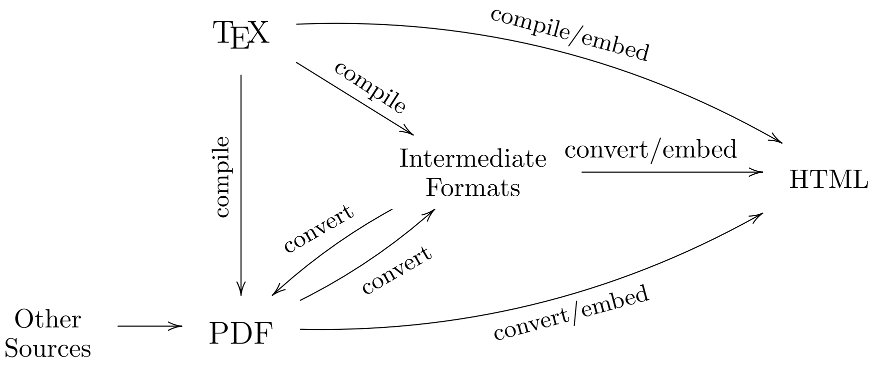


Figure 3: Different approaches to publishing online.

The disadvantages can be compensated for to some extent: A hidden text layer can be overlaid above the images in order to simulate user text selection, however generating this text layer itself actually involves other conversion technologies; for a responsive user experience, the input document may be converted into images with different resolutions, and images with high resolutions can then be split into small blocks. When the document is rendered on the client side, only the block being viewed by the user is needed to transfer. However much more disk storage and network bandwidth is required in this way, which might not be affordable for all publishers, especially individuals.

3.2 SVG-based approaches

Scalable Vector Graphics, developed by W3C, is an XML-based format for presenting 2D graphics. It supports a similar set of features as PDF, including color, gradients, patterns, paintings and raster images. It also supports font definition within SVG as well as external fonts defined in CSS.

Due to the large feature set, most visual elements can be rendered with SVG counterparts. Popular tools in this category include:

- Inkscape [21]
- pdftocairo from Poppler [27]
- pdf2svg [26]
- dvisvgm [17]
- “fallback” mode of pdf2htmlEX

Pros Due to the similar nature between SVG and PDF, it is relatively easy to find an SVG counterpart for each PDF element. SVG is one of the few methods that support advanced layout features such as characters along a curved path and image clipping.

SVG is based on XML, hence it can be easily parsed or edited manually. SVG can be well integrated with HTML/CSS, and it can be easily accessed and manipulated by JavaScript.

Cons Old web browsers do not support SVG, and the degree of support for SVG varies for modern web browsers.

While SVG-based approaches are powerful when integrated with HTML, CSS and JavaScript, most tools in this category do not support such integrations, probably because they were designed as an SVG converter instead of an online publishing tool.

3.3 Semantic HTML-based approaches

Approaches of this type try to find the matching HTML element for each TEX element, for example `\section` and `\textbf` in TEX might become `<h1>` and `<b>` in HTML respectively. Popular tools in this category include

- HEVEA [19]
- LATEX2HTML [1, 37]
- LATEXML [24]
- plasTEX [5, 29]
- TEX4ht [6, 33]
- TEX2page [14]
- TtH [12]

all of which are designed to process general TEX files. There are also programs designed for particular documents, for example

- The Feynman Lectures on Physics [28]
- The Stacks Project [38]
- The TUG Interviews Project [30]

Pros Semantic HTML files are normally expected by most users. Semantic information is retained in an XML-like format, such that they can be read or edited by a human or further processed by other programs.

Basic elements such as colors, font family and sizes, paragraphs, links, images can all be supported.

CSS can be used to specify the layout and appearance. Math formulas may be semantically retained via Unicode characters, MathML or embedded TEX snippets (see Section 3.5).

Approaches of this type can be used when the publisher does not rely on the layout produced by a TEX compiler. They are often used for simple text-based files without complicated layouts. The final layout is determined by the web browser based on the semantic structure and CSS rules.

Cons Approaches of this type usually don't work well for PDF files. In general, PDF files do not contain semantic information, and so *recognition* is inevitable to detect semantic meanings, which is considered to be hard. This is also true for other intermediate files.

On the other hand, TEX users do not necessarily expect the same appearance as compiled by TEX. Most advanced layouts in TEX cannot be used, for example, double columns. Specific layouts might be simulated, but it is hard in general due to the essential differences between the page model of TEX and HTML. While font embedding is possible nowadays, most tools of this kind do not support it.

Furthermore, this type of approach can be considered a re-implementation of TEX, as these tools parse and process TEX syntax in their own engines, and therefore some macros and packages may not work with them, especially those related to drawing, page layout or PDF-specific features. Sometimes the authors have to prepare different versions of TEX files for both HTML and PDF, and HTML knowledge might also be required.

It is possible to achieve HTML documents in rather good quality, while reserving not only semantic information, but also well organized links, elegant styles and MathJaX-based math formulas. Good examples are [28] and [38]. However, most of them employ project-specific tools and lots of engineering work, and there are also limitations or paradigms for authors. Therefore their methods might not work for general documents.

3.4 Presentation HTML-based approaches

Approaches of this type focus on the layout and appearance of the result, utilizing CSS rules to set accurate position and size for each element, mostly text. Non-text elements are usually converted into images, raster or vector, which are embedded in the HTML. They are still significantly different from raster image based approaches or SVG based approaches mentioned above, as they do not convert the whole document into images.

Prior to pdf2htmlEX, the pdftohtml utility from Poppler [27] is probably the best known tool that is freely available to the community. While pdftohtml focuses more on extracting semantic information, pdf2htmlEX focuses on precise layout and appearance. There seems not to be any tools that directly produce presentation HTML files from TEX, because of course TEX users may produce PDF files before further converting it into HTML.

Pros Comparing this output with images, text is now represented with native HTML elements, such that they can be selected by users or easily extracted by programs; the file size is heavily reduced in this way. Also it is easier to apply CSS and JavaScript to tweak the appearance.

Comparing with semantic HTML files, the appearance of presentation HTML output is often closer or even identical to the original document. TEX users are free to use any advanced layout, macro or package, fine-tuning will also be reflected in the output, as the TEX page model is simulated in HTML.

Cons While text is still available, the semantic meanings (*e.g.* title, section, table *etc.*) are likely to be lost. The content may be too complicated to be further processed. Precise layout and appearance rely on advanced CSS features, like font embedding, absolute positioning and linear transformation, which might not be supported by old web browsers.

3.5 JavaScript-based approaches for TEX

While TEX is not directly supported in HTML, modern JavaScript technologies allow us to *embed* these files in HTML, such that they will be parsed and rendered directly in the web browsers. Similar technologies for PDF are covered in Section 3.6.

MathJaX [7] is a JavaScript display engine which parses and renders TEX snippets on web pages with HTML/CSS, SVG or MathML. MathJaX is designed for online communications where users want to directly input formulas in the TEX syntax. Similar projects include jsTeX [2], and jsMath [4].

LATEX2HTML5 [22] is able to produce interactive diagrams from PSTricks macros; it also utilizes MathJaX for rendering math formulas.

Pros This kind of approach is best for dynamic content, especially that intended to be created or modified by users. Any modification to the source can be reflected in the result promptly without any network transmission. It can also handle documents with simple layout, while formatting can be specified with CSS.

Cons Approaches of this kind are usually focusing on specific elements; while they may support a small set of TEX syntax, they are not designed as a JavaScript implementation of TEX. Therefore advanced commands or macros may not be supported, and usually they are not capable of typesetting general documents with complicated layout.

3.6 JavaScript-based approaches for PDF

PDF.js [13] is a JavaScript library for rendering PDF; it is now a part of Mozilla Firefox. It is like the raster image-based approaches except that all the parsing and rendering are done on the client side. Recent web browsers are necessary to support the technologies used by the library.

PDF.js is one of a kind; there are no similar alternatives to the best of our knowledge.

Pros PDF.js renders PDF files into an HTML5 canvas, which is similar to a raster image; most PDF elements can be rendered correctly. Furthermore, it does not suffer from a huge network cost as only the original PDF file need be transferred.

The library can be embedded into web pages, and can be extended by publishers if needed.

Cons PDF.js relies heavily on the computation power on the client side, which might cause performance issues in some environments.

It is designed as a PDF reader, and it does not optimize for online publishing; for example users still have to wait for the entire file to be downloaded before they can read any page.

PDF elements are rendered into an HTML5 canvas, which may not be flexible enough for publishers.

3.7 Plugin-based approaches

Many web browsers support plugins to add new features, especially plugins can be used to display TEX, PDF, or other formats converted from them. Publishers may also develop plugins for their own formats, which are otherwise not supported by web browsers.

Adobe Reader includes plugins to display PDF files within different web browsers. There are also similar third-party plugins based on Adobe Flash. There are also plugins to display math formulas inside web browsers, *e.g.* MathPlayer [23].

Pros Plugins are not limited to web technologies, thus they are usually better in term of rendering quality or supported features. For example, Adobe Reader should be the plugin with the most complete support for PDF features.

Cons The crucial downside is that plugins usually create closed environments, which prevent an interactive user experience on the web sites. Most plugins are not easily customizable, except for a few commercial ones. Due to the active development of HTML5 technologies, plugins nowadays are no longer so popular as before, for security, compatibility and performance reasons.

3.8 Third-party services

Third-party services also exist such that embedded TEX or PDF can be redirected to their servers for processing and rendering, for example:

- QuickLATEX [10]
- mathTEX [9]
- Google Docs [3]
- Crocodoc [15]

This kind of service accepts input uploaded by publishers, converts it internally with their own implementations and redirects the result to users. The technologies behind them, open or proprietary, should still fall into the categories mentioned above.

Pros This kind of service is usually easy to deploy, and convenient for publishing a few simple documents. The conversion process relies only on the computation power of the third party. Some services also provide an API for better integration with publishers' sites.

Cons A crucial issue here that may concern a lot of publishers is that the files must be accessible by the third party. This may not be acceptable for private, confidential or copyrighted materials. In addition, the publisher's service has to depend on the availability of the third party. Further development may also be limited by the API provided or other issues such as different domains.

3.9 Discussion

In this section we categorized a number of popular approaches, among which the most popular three types of output are:

Image is best for publishers who can afford large volume of storage and bandwidth. There is no need to fine-tune or even redesign the document, as layout and format are already accurately preserved in the output. Yet this highly compatible result can be viewed by most users.

Semantic HTML is best for simple TEX source files. All semantic information is preserved in the output, which can be further processed by other tools. In particular, math formulas may be rendered interactively with latest web technologies.

Presentation HTML is best when complicated layout, advanced TEX macros and packages are used. It is also suitable when the source files are not available at all. PDF files produced from other tools can also be supported. Flexibility of semantic HTML and accuracy of image are cleverly balanced.

In general there is no best approach for all situations. Users should carefully choose the best matching one according their specific concerns.

While we tried our best to be complete and accurate, it is quite possible that we have missed or misunderstood some approaches due to the limitations of our knowledge. Please contact us for corrections or suggestions, and thank you.

4 A tour of pdf2htmlEX

In this section we introduce pdf2htmlEX, created and mostly written by Lu Wang, which is an open source PDF to HTML converter. It generates presentation HTML documents, utilizing modern Web technologies such as HTML5, CSS3, JavaScript, *etc.*, such that most PDF features can be retained. Especially fonts, math formulas and images can all be displayed correctly.

Figure 4: Logo of pdf2htmlEX

pdf2htmlEX is not only a *converter*, but also a *publishing tool.* It is designed for many different situations, for example:

Scenario 1 My sister wants to put her resumé on her online homepage. She wants the resumé to be stored into one single file such that it can be easily downloaded by others. She also needs to add JavaScript code to track how many people have read her resumé.

Scenario 2 A book publisher wants to put some sample books online to attract readers. The publisher does not want readers to wait for too long before they can read any page, thus pages are better converted and stored separately. Images and fonts should also be stored in individual files such that users may benefit from web caches.

Scenario 3 A cloud storage service provider wants to provide a PDF preview feature to their service, such that users may read their files online. The service provider needs to design their own viewer to match the theme and behaviour of their web site. They also want to attach advertisements based on the contents of the files. Advanced users may be allowed to leave marks and discuss with others about particular parts of the documents. In this case the service provider needs the finest control — they need to access every single element of the document for their customizations.

We can see that different forms of HTML files are desired in different scenarios, and flexibility is always necessary. pdf2htmlEX is indeed designed for all these scenarios and many others; some features have been requested or implemented by users.

In this section we will introduce a few useful features and explore some internal mechanisms of pdf2htmlEX. More information, including source code and license terms, is at the project home page: `https://github.com/coolwanglu/pdf2htmlEX`

4.1 Quick start

Throughout this section, a sample PDF file is used to demonstrate different features of pdf2htmlEX. The file, `integral.pdf`, contains 4 pages from the book *Differential and Integral II* [16, 36], which consists of Japanese characters, mathematical symbols and formulas, figures, images and delicate layouts. We believe that it reflects common elements used in real use cases.

To start with, we simply execute

```
$ pdf2htmlEX --fit-width 1024 integral.pdf
```

which produces a *single* HTML file `integral.html`. The result is shown in Figure 5,[7] and we challenge the readers to find any evidence or clues that the screenshot shows an HTML file instead of PDF. (Except for the title bar of course.)

The `--fit-width 1024` option specifies that each page should be squeezed or stretched to the width of 1024 pixels. The zoom ratio can be adjusted with similar options: `--fit-height` and `--zoom`.

It is possible to convert only a few pages of a PDF file, for example

```
$ pdf2htmlEX -f 2 -l 3 integral.pdf
```

converts only the second page and the third page.

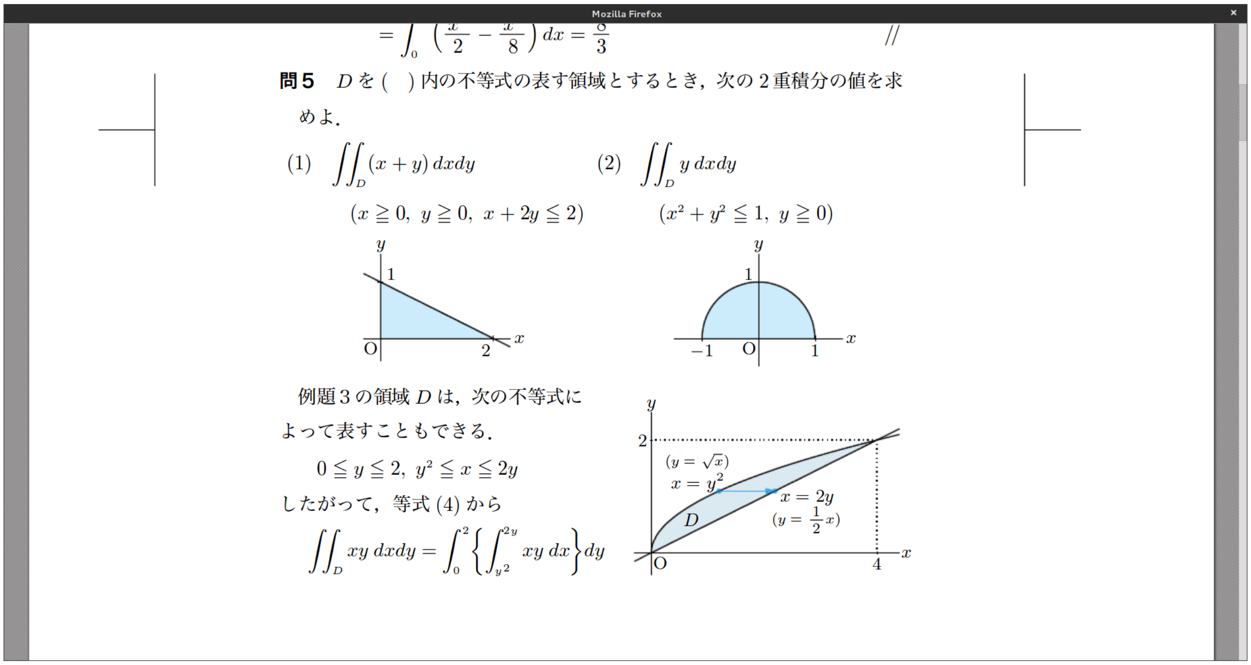
4.2 Separating resource files

By default everything is combined into one single HTML file, which is good for creating archives or performing tests. However, it is not a good practice when publishing HTML documents online; often we want resource files (fonts, CSS, JavaScript, images *etc.*) to be stored separately in order to reduce size and improve efficiency.

With the `--embed` option, we can decide which types of resource files are embedded and which are not. For example,

```
$ pdf2htmlEX --embed fi integral.pdf
```

[7] Mozilla Firefox 24 on Ubuntu 13.04 is used for all the demonstrations.



$$= \int_0 \left(\frac{x}{2} - \frac{x}{8}\right) dx = \frac{8}{3} \qquad //$$

問 5　D を (　) 内の不等式の表す領域とするとき，次の 2 重積分の値を求めよ．

(1) $\displaystyle\iint_D (x+y)\,dxdy$ 　(2) $\displaystyle\iint_D y\,dxdy$

$(x \geqq 0,\ y \geqq 0,\ x + 2y \leqq 2)$ 　$(x^2 + y^2 \leqq 1,\ y \geqq 0)$

例題 3 の領域 D は，次の不等式によって表すこともできる．

$$0 \leqq y \leqq 2,\ y^2 \leqq x \leqq 2y$$

したがって，等式 (4) から

$$\iint_D xy\,dxdy = \int_0^2 \left\{ \int_{y^2}^{2y} xy\,dx \right\} dy$$

Figure 5: An HTML document produced by pdf2htmlEX.

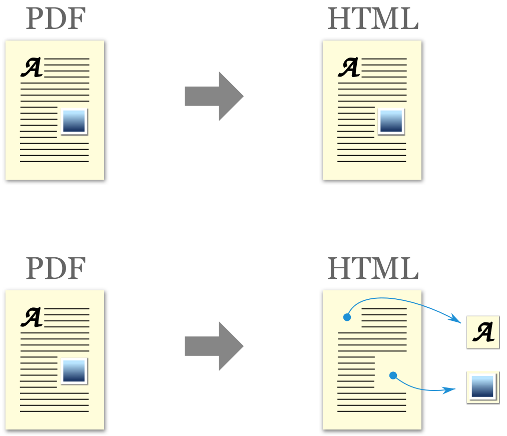


Figure 6: Above: by default all resources are embedded in the HTML file. Below: with the `--embed fi` option, fonts and images are stored into separate files and linked to the HTML file.

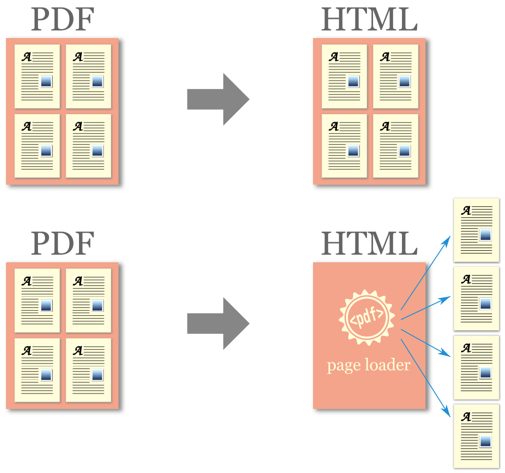


Figure 7: Above: by default all pages are embedded in the HTML file. Below: with `--split-pages 1`, pages are stored in separate HTML snippets, which can be dynamically loaded to the main HTML file.

stores all fonts and images in separate files, as shown in Figure 6. There are also specific options including `--embed-css`, `--embed-font`, `--embed-image`, *etc.*

4.3 Splitting pages

With a large PDF file containing hundreds of pages, often we have to download the whole file even if we want to take a look at only a few pages inside. On the other hand, web pages are usually stored into separate files, such that we just need to download the pages we request.

With the `--split-pages` option, it is possible to store PDF pages into separate HTML *snippets*. In this way, when the main HTML file is loaded on the client side, only necessary pages will be dynamically loaded via Ajax, as shown in Figure 7.

4.4 Image format for backgrounds

For each page, pdf2htmlEX generates a background image to present all non-text elements. By default all images are generated in the PNG format, and different formats can be specified via the `--bg-format` option. For example,

```
$ pdf2htmlEX --bg-format jpg integral.pdf
```

would generate all images in the JPEG format.

Currently pdf2htmlEX supports PNG and JPEG. There is also preliminary support for SVG. Users can also convert the images into other formats.

4.5 Customizing the output

`integral.html` contains a default set of HTML, CSS and JavaScript, which is designed for average use cases. All of them can be found in the so-called `data-dir` (run `pdf2htmlEX -v` to see the location), and they can be tweaked by the users.

HTML template The `manifest` file determines how pages should be combined into an HTML document. It is a template for the output and users may add their own HTML snippets into it. A typical use case is enabling a traffic statistics service on the page.

CSS Quite a number of features of pdf2htmlEX rely on CSS; the default CSS styles determine the correct appearance and behavior of the elements. Advanced users can override existing properties by modifying the CSS files.

JavaScript A simple UI is implemented in the default `pdf2htmlEX.js` file. This also serves as a demonstration of accessing and manipulating HTML elements produced by pdf2htmlEX. It can be a good reference for advanced users who want to implement their own UIs.

4.6 Secrets of pdf2htmlEX

Here we briefly introduce some internal mechanisms of pdf2htmlEX for the curious readers.

`integral.html` consists of two layers: the text layer and the image layer, as shown in Figures 8 and 9. pdf2htmlEX parses `internal.pdf` and extracts elements from it. The elements are then processed and put into one of the layers.

Text Unlike in HTML, in PDF text is set in fixed positions. Text extracted from the PDF is translated into native HTML text elements, and put into the same position in the HTML as they were in PDF. In this way text can be selected and copied by users, while preserving the layout. Many fixed-position text elements in HTML make the file very large in size and very slow to render; to compensate, pdf2htmlEX tries to recognize and merge text lines according to their geometric metrics.

Font Font embedding is one of the most important features of PDF, without which it is nearly impossible to preserve the appearance of PDF in HTML. No similar feature has been supported in the HTML standard until recently. Figure 10 shows a few fonts used in `integral.pdf`. pdf2htmlEX is able to extract all the fonts from PDF and convert them into web fonts via FontForge [18]; converted fonts are then embedded or referred to in the HTML file. All font formats supported in PDF are supported by pdf2htmlEX, and different web font formats can be specified for output.

Encoding Unlike HTML, PDF uses two sets of encodings for text rendering, one for choosing correct glyphs to display, and the other for meaningful text that can be selected and copied by users. pdf2htmlEX is able to combine both sets into one and re-encode the font accordingly, such that text in HTML is correct both visually and meaningfully. This is another essential feature of pdf2htmlEX, like font processing.

Images PDF supports graphical instructions such as drawing and image embedding. Such elements are all rendered into images, and then put into the image layer.

4.7 Future work

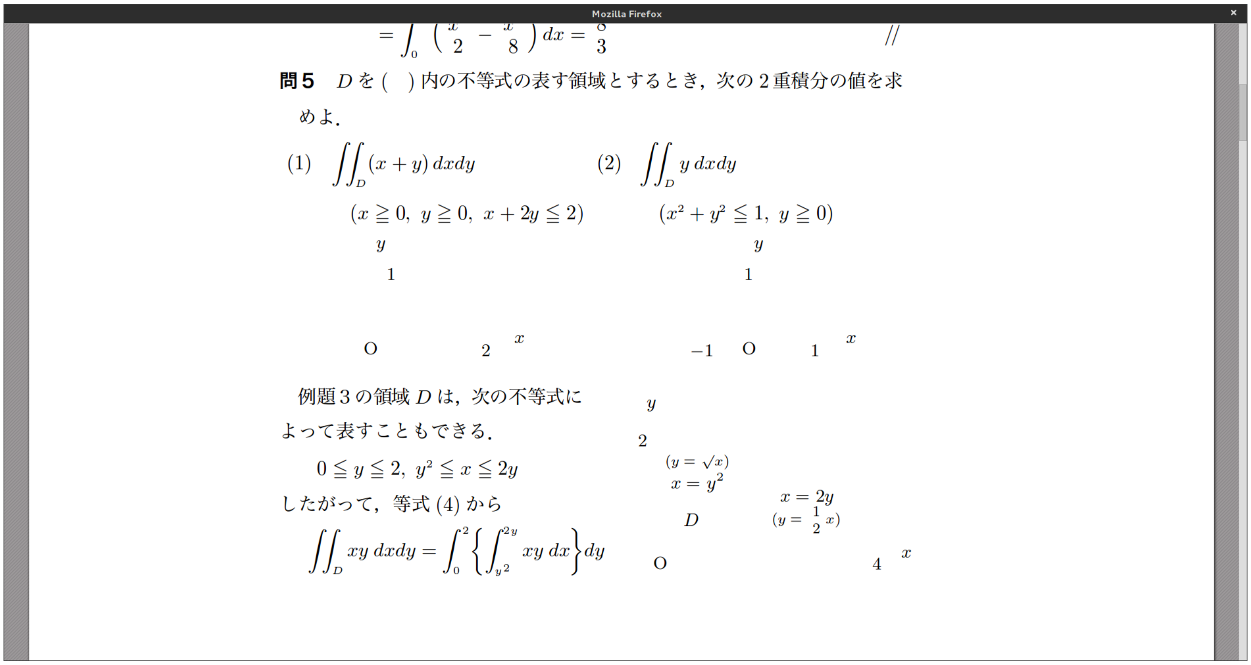
Several features are planned in the future versions of pdf2htmlEX.

Reflowable text Comparing with HTML files directly converted from TeX, text in HTML files generated by pdf2htmlEX is generally not reflowable, *i.e.* the width of paragraphs cannot be self-adapting to the size of the viewer. After all, that information is generally not available in a PDF file, and it is not easy for pdf2htmlEX to recover it.

On the other hand, reflowable text may be extracted for specific document types and layouts. Extracting such information would make it much easier to further process HTML files generated by pdf2htmlEX, such as to edit manually, to embed accessibility information or to convert into other formats like EPUB.

Preserving semantic information While much semantic information is lost in PDF as mentioned above, theoretically it is possible for authors to embed additional information into PDF, such that it may be further recognized and used by pdf2htmlEX. This kind of PDF file is called a *tagged* PDF, which can also be generated with other tools.

Especially for TeX users, it is possible to mark text paragraphs such that text will be reflowable in HTML to some extent; also, mathematical formulas may be marked such that they will be rendered with MathJaX in HTML.



$= \int_0 \left(\frac{x}{2} - \frac{x}{8} \right) dx = \frac{8}{3}$ //

問 5　D を (　) 内の不等式の表す領域とするとき，次の 2 重積分の値を求めよ．

(1) $\displaystyle\iint_D (x+y)\,dxdy$　$(x \geqq 0,\ y \geqq 0,\ x + 2y \leqq 2)$

(2) $\displaystyle\iint_D y\,dxdy$　$(x^2 + y^2 \leqq 1,\ y \geqq 0)$

例題 3 の領域 D は，次の不等式によって表すこともできる．

$0 \leqq y \leqq 2,\ y^2 \leqq x \leqq 2y$

したがって，等式 (4) から

$$\iint_D xy\,dxdy = \int_0^2 \left\{ \int_{y^2}^{2y} xy\,dx \right\} dy$$

Figure 8: The text layer

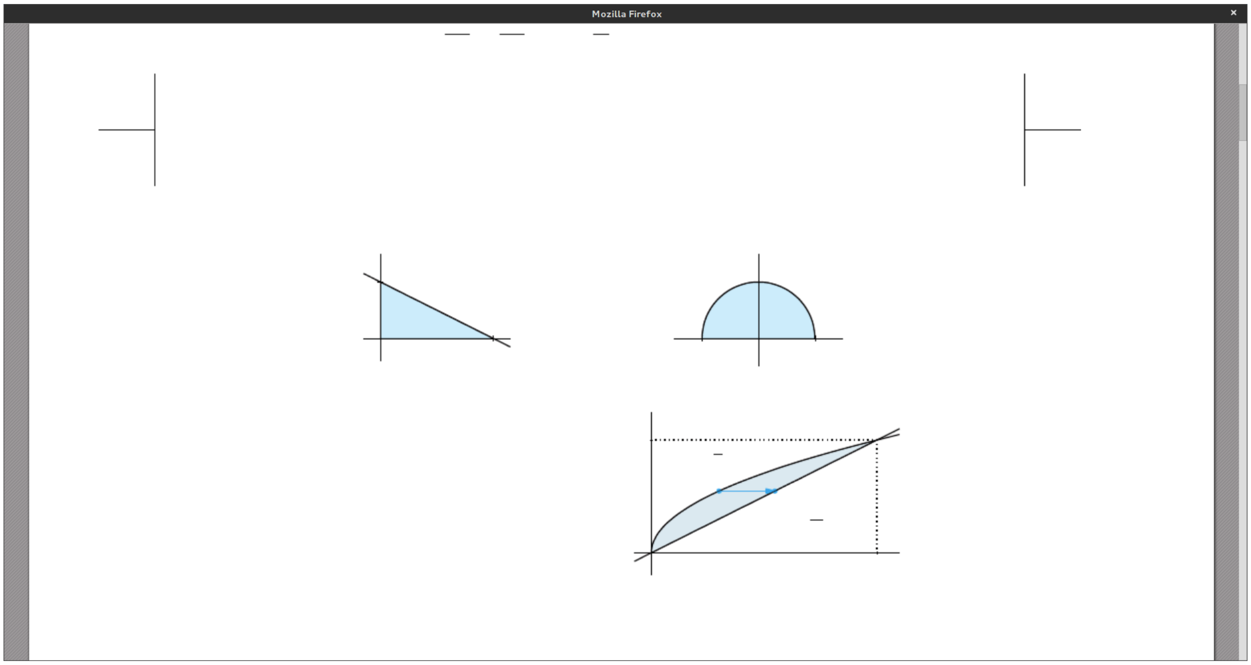


Figure 9: The image layer

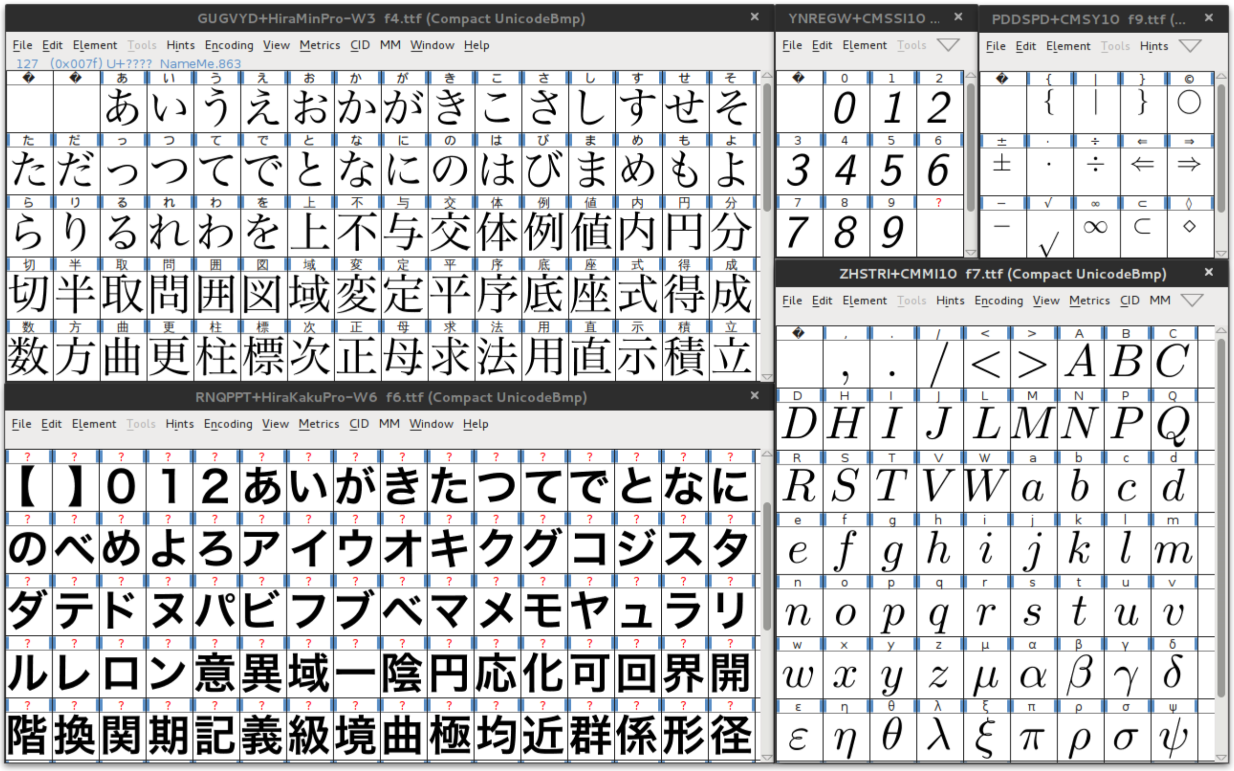

GUGVYD+HiraMinPro-W3 f4.ttf (Compact UnicodeBmp)
File Edit Element Tools Hints Encoding View Metrics CID MM Window Help
127 (0x007f) U+???? NameMe.863
あいうえおかがきこさしすせそ
ただっつてでとなにのはびまめもよ
らりるれわを上不与交体例値内円分
切半取問囲図域変定平序底座式得成
数方曲更柱標次正母求法用直示積立
RNQPPT+HiraKakuPro-W6 f6.ttf (Compact UnicodeBmp)
File Edit Element Tools Hints Encoding View Metrics CID MM Window Help
【】012あいがきたつてでとなに
のべめよろアイウオキクグコジスタ
ダテドヌパビフブベマメモヤュラリ
ルレロン意異域一陰円応化可回界開
階換関期記義級境曲極均近群係形径
YNREGW+CMSSI10 ...
File Edit Element Tools Hints
0 1 2
3 4 5 6
7 8 9
PDDSPD+CMSY10 f9.ttf (...
File Edit Element Tools Hints
{ | } ○
± · ÷ ⇐ ⇒
− √ ∞ ⊂ ◇
ZHSTRI+CMMI10 f7.ttf (Compact UnicodeBmp)
File Edit Element Tools Hints Encoding View Metrics CID MM
, . / < > A B C
D H I J L M N P Q
R S T V W a b c d
e f g h i j k l m
n o p q r s t u v
w x y z μ α β γ δ
ε η θ λ ξ π ρ σ ψ

Figure 10: Fonts embedded in the HTML file

Image overlay For each PDF page, pdf2htmlEX puts all non-text elements into a background image; this image is then put behind all the text in that page. However, it is possible that some text is in fact covered by an image in the PDF, in which case in the corresponding HTML file produced by pdf2htmlEX, the text will be visible due to the superimposition.

We are still looking for efficient solutions for this issue; fortunately, this issue is not common, especially for TEX users. A workaround is to use the *fallback* mode of pdf2htmlEX at the cost of larger file size.

Image optimization When generating the background image, pdf2htmlEX calculates the bounding box of all non-text elements in that page, and renders everything inside. However, if there are only a few images which are far away from each other, most parts in the image are actually blank, which will be a waste of bandwidth. It is possible to recognize and split those small images and pack them into one small image, such that they will be loaded using the CSS sprite technique. In this way significant bandwidth and computation can be saved.

4.8 Discussion

In this section we introduced pdf2htmlEX from several perspectives. Due to space limitations here, we cannot present everything — there are nearly 50 different options in total, and there are also tricky implementations regarding font conversion, text handling and image processing. Interested readers are encouraged to visit the project web site for detailed and up-to-date documentation.

5 Conclusion

In this article we tried to categorize and compare existing methods of publishing TEX or PDF online. We hope that readers may use this article as a guide to choose the proper tool for their specific use cases, or be inspired to create their own implementations.

We also introduced our program pdf2htmlEX, a PDF to HTML converter and publishing tool which is accurate and flexible for many different use cases. We encourage interested users to get involved.

Acknowledgement

We thank Professor Haruhiko Okumura for his help and great advice. We also thank Professor Makasata Kaneko, Mr Raphaël Pinson and Mr Jason Lewis for the nice sample files used in this article.

References

[1] LaTeX2HTML. http://www.latex2html.org, 2001.

[2] jsTeX. http://simile.mit.edu/wiki/JsTeX, 2008.

[3] Google Docs Viewer. https://docs.google.com/viewer, 2009.

[4] jsMath: A Method of Including Mathematics in Web Pages. http://www.math.union.edu/~dpvc/jsmath, 2009.

[5] plasTeX. http://plastex.sourceforge.net, 2009.

[6] TeX4ht: LaTeX and TeX for Hypertext. http://tug.org/tex4ht, 2010.

[7] MathJaX: Beautiful math in all browsers. http://www.mathjax.org, 2011.

[8] Bible de Genève, 1564. https://github.com/raphink/geneve_1564, 2012.

[9] mathTeX. http://www.forkosh.com/mathtex.html, 2012.

[10] QuickLaTeX — advanced LaTeX web rendering service. http://quicklatex.com, 2012.

[11] SpringerLink. http://link.springer.com/, 2012.

[12] TtH: The TeX to HTML translator. http://hutchinson.belmont.ma.us/tth, 2012.

[13] PDF.js. https://github.com/mozilla/pdf.js, 2013.

[14] TeX2page. http://www.ccs.neu.edu/home/dorai/tex2page/index.html, 2013.

[15] Crocodoc: HTML5 Document Embedding. https://crocodoc.com, 2013.

[16] *Differential and Integral II*. Dai-Nippon Tosho Publisher, 2013.

[17] dvisvgm: A DVI to SVG converter. http://dvisvgm.sourceforge.net, 2013.

[18] FontForge: A font editor. http://fontforge.org, 2013.

[19] HEVEA: A LaTeX to HTML translator. http://hevea.inria.fr, 2013.

[20] ImageMagick: Convert, Edit, And Compose Images. http://www.imagemagick.org, 2013.

[21] Inkscape: An open source scalable vector graphics editor. http://inkscape.org, 2013.

[22] LaTeX2HTML5 — interactive math equations and diagrams. http://latex2html5.com, 2013.

[23] MathPlayer: Display MathML in your browser. http://www.dessci.com/en/products/mathplayer, 2013.

[24] LaTeXML: A LaTeX to XML Converter. http://dlmf.nist.gov/LaTeXML, 2013.

[25] pdf2htmlEX: Convert PDF to HTML without losing text or format. https://github.com/coolwanglu/pdf2htmlex, 2013.

[26] pdf2svg. http://www.cityinthesky.co.uk/opensource/pdf2svg, 2013.

[27] Poppler. http://poppler.freedesktop.org, 2013.

[28] The Feynman Lectures on Physics. http://www.feynmanlectures.info, 2013.

[29] Tim Arnold. Getting started with plasTeX. *TUGboat*, 30(2):180–182, 2009. http://tug.org/TUGboat/tb30-2/tb95arnold.pdf.

[30] Karl Berry and David Walden. TeX People: The TUG interviews project and book. *TUGboat*, 30(2):196–202, 2009. http://tug.org/TUGboat/tb30-2/tb95berry-interviews.pdf.

[31] Peter Flynn. LaTeX on the Web. *TUGboat*, 26(1):66–67, 2005. http://tug.org/TUGboat/tb26-1/flynn.pdf.

[32] Stephen A. Fulling. Keynote: TeX and the Web in the higher education of the future: Dreams and difficulties. *TUGboat*, 20(3):371–372, 1999. http://tug.org/TUGboat/tb11-3/tb29fulling.pdf.

[33] Eitan Gurari. TeX4ht: HTML production. *TUGboat*, 25(1):39–47, 2004. http://tug.org/TUGboat/tb25-1/gurari.pdf.

[34] Steven G. Krantz. *Handbook of Typography for Mathematical Sciences*. Chapman and Hall/CRC, 2000.

[35] Jason Lewis. How I use LaTeX to make a product catalogue that doesn't look like a dissertation. *TUGboat*, 34(3):263–267, 2013.

[36] Yoshifumi Maeda and Masataka Kaneko. Making math textbooks and materials with TeX+KETpic+hyperlink. Presentation at TUG 2013.

[37] Ross Moore. Presenting mathematics and languages in Web-pages using LaTeX2HTML. *TUGboat*, 19(2):195–203, 1998. http://tug.org/TUGboat/tb19-2/tb59moore.pdf.

[38] The Stacks Project Authors. The Stacks Project. http://stacks.math.columbia.edu, 2013.

◇ Lu Wang
Department of Computer Science and Engineering
The Hong Kong University of Science and Technology
Hong Kong
coolwanglu (at) gmail dot com
http://coolwanglu.github.io/

◇ Wanmin Liu
Department of Mathematics
The Hong Kong University of Science and Technology
Hong Kong
wanminliu (at) gmail dot com