# Render PDFs on the browser using JavaScript

07 March 2016 **3 Comments (/render-pdfs-on-the-browser-using-javascript/#disqus_thread)** Posted in JavaScript (/tag/javascript/), web development (/tag/web-development/)

Most browsers these days support PDF rendering. In some cases, you may also need a PDF plugin to be installed. But what happens if you need to render PDFs on a closed system with no access to 3rd party plugins? How do you ensure that your solution can work consistently cross-browser? I recently worked on a project where we had to integrate a PDF viewer on a “closed” web application and this presented a new challenge. And who doesn’t love a good challenge? The requirements for the PDF viewer were as follows:

- Should work cross-browser
- Should support modern(ish) platforms
- It shouldn’t rely on a PDF add-in to be installed.

In my search for the right solution, I came up across PDF.js. This is a tool created by the Mozilla team and provides native support for rendering PDFs within your page using pure vanilla JavaScript. The project is open source and you can check the source code here: https://mozilla.github.io/pdf.js/ (https://mozilla.github.io/pdf.js/). Since the tool is all written in JavaScript, the solution will work without Adobe PDF Reader or any other PDF Readers installed on the target machine. The code to instantiate and display PDFs using pdf.js is surprisingly simple. However, as with everything else, there are some limitations/caveats:

- It doesn’t support **CORS**
- The documentation is bad/extremely limited

- It doesn’t provide any support for navigation or other tooling around the PDF (printing, saving etc)

The first one means that you can only retrieve and display pdf documents that are on the same domain. If you site is running under http://mysite.com (http://mysite.com), you can only get documents under that domain and trying to access a PDF document served at http://myothersite.com (http://myothersite.com) will fail. And the second is self-explanatory. Trying to implement the app takes a bit of an effort and you may need to poke into the source code. But we are developers so nothing new there. Finally, PDF.js may not provide any tooling, but it does come with a nice PDFViewer.html page which can be used instead to achieve the required results.
Show me the code:
To add PDF.js to your application, you’ll need to execute the following steps:

- Download the code
- Add the JavaScript code to instantiate the PDF.js library and pull the files
- Add a progress handler and navigation (optional)
- Done

## DOWNLOAD THE CODE

You can get the latest code in a zip file from this location: https://mozilla.github.io/pdf.js/getting_started/#download (https://mozilla.github.io/pdf.js/getting_started/#download) Unzip the folder. If you’re only interested in rendering the PDF and creating all the controls yourself, all you need to add is references to the following files:
- Pdf.js (in the `/build/` folder) - Pdf.worker.js (in the `/build/` folder) - Compatibility.js (in the `/web/` folder) However, if you wish to take advantage of the pdfViewr, then you’ll need to add the whole (unzipped) folder into your application.

## ADD THE JAVASCRIPT CODE

If you’ve decided to go with option one (render stuff and control the PDF rendering yourself), you’ll need to write some code to retrieve and render the document on a canvas HTML element. The following sample instantiates a renders a local PDF file to the page. I’ve also added an event handler for the progress and a reference to the total pages. You can use these as a starting point to add your own implementations for navigation and progress status.

```
var progressHandler = function (progress){
    console.info('Loaded ' + progress.loaded + ' of ' + progress.total);
}

PDFJS.getDocument('pdf.pdf').then(function(pdf){

    console.info('Document contains ' + pdf.pdfInfo.numPages + ' pages');

    //assign the progress event handler
    pdf.loadingTask.onProgress = progressHandler;

    // Fetch the page.
    pdf.getPage(1).then(function (page) {

        // Prepare canvas using PDF page dimensions.
        var containerColumn = document.getElementById('pdfContainer');
        var canvas = document.getElementById('canvas');
        var context = canvas.getContext('2d');
        canvas.height = containerColumn.clientHeight;
        canvas.width = containerColumn.clientWidth;

        var unscaledViewport = page.getViewport(1);
        var scale = Math.min((canvas.height / unscaledViewport.height), (canvas.width / unscaledViewport.width)
        var viewport = page.getViewport(scale);

        // Render PDF page into canvas context.
        var renderContext = {
        canvasContext: context,
        viewport: viewport
        };
        page.render(renderContext);
    });
});
```

PdfJsSample.js view raw (https://gist.github.com/cmatskas/084f41c13faf95568049/raw/e584609a16f62aa70fef9a5f5bd92326be2891e1/PdfJsSample.js)
(https://gist.github.com/cmatskas/084f41c13faf95568049#file-pdfjssample-js) hosted with ❤ by GitHub (https://github.com)

The following URL shows an example using controls to navigate the PDF Document: https://mozilla.github.io/pdf.js/examples/learning/prevnext.html (https://mozilla.github.io/pdf.js/examples/learning/prevnext.html)

### THE EASY WAY OUT

If you need a more complete solution for displaying your pdfs, you can use the following method once you've added the full, unzipped PDF.js folder to your solution. Create an iFrame or anything else that can use a URL as a source and supply a URL http:///web/viewer.html?file= => http://localhost/web/viewer.html?file=http://localhost/pdf.pdf (http://localhost/web/viewer.html?file=http://localhost/pdf.pdf)

## Node.js & NPM

For all you Node.js developers/fans, rest assured that you're covered as well. There is a nice npm package which will download all the necessary files to your solution. You can find details about the package here: https://www.npmjs.com/package/pdfjs-dist-for-node (https://www.npmjs.com/package/pdfjs-dist-for-node). However, if you're after the **TL;DR** version, then here you go:

```
npm install pdfjs-dist-for-node
```

## SAMPLE CODE TO DOWNLOAD

Get in Github: https://github.com/cmatskas/PdfJsSample/ (https://github.com/cmatskas/PdfJsSample/)
I hope this gives you a way to display PDFs in a consistent way that works across all browsers.

***P.S. Make sure you follow me on Twitter @christosmatskas (https://twitter.com/christosmatskas) for more up-to-date news, articles and tips.***

---

Share this post on
🐦 (https://twitter.com/share?text=Render%20PDFs%20on%20the%20browser%20using%20JavaScript&url=https://cmatskas.com/render-pdfs-on-the-browser-using-javascript/)
f (https://www.facebook.com/sharer/sharer.php?u=https://cmatskas.com/render-pdfs-on-the-browser-using-javascript/)
in (http://www.linkedin.com/shareArticle?mini=true&url=https://cmatskas.com/render-pdfs-on-the-browser-using-javascript/&title=Render%20PDFs%20on%20the%20browser%20using%20JavaScript&source=cmatskas.com)
g+ (https://plus.google.com/share?url=https://cmatskas.com/render-pdfs-on-the-browser-using-javascript/)

← Joining Microsoft - a new beginning (/joining-microsoft-a-new-beginning/)

Blog Home (https://cmatskas.com)

PowerShell for the Gui People or FTGP → (/powershell-for-the-gui-people-or-ftgp/)

3 Comments cmatskas

1 Login

Recommend Share

Sort by Best

Join the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

**meatlover** • 2 years ago

any example in jdfiddle?

Reply • Share ›

**Christos Matskas** Mod meatlover • 2 years ago

Hi @meatlover could you be a bit more specific so that I can see how I can help you? Thanks

Reply • Share ›

**meatlover** Christos Matskas • 2 years ago

I was trying to follow your example but somehow it ain't working

Reply • Share ›

Subscribe Add Disqus to your siteAdd DisqusAdd Disqus' Privacy PolicyPrivacy PolicyPrivacy