PDF.JS EXPRESS
Buy vs Build

PDF.JS EXPRESS

PDF.JS EXPRESS - Buy vs Build

In this article, we outline some key considerations when deploying a PDF.js viewer into your application. The three top considerations are:

1. Determining if customizing your PDF.js viewer is required
2. Understanding the cost of developing customizations
3. The costs of maintaining your viewer

A need for customization

When Mozilla introduced PDF.js in 2011, it was designed to be Firefox's PDF Viewer -- allowing users to render PDFs in their web browsers. Out-of-the-box PDF.js comes with three layers:

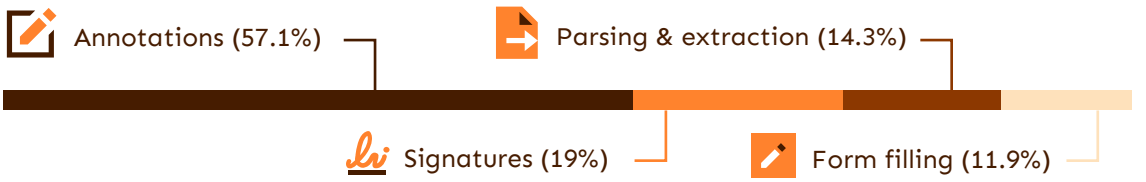
- A core layer to interpret the binary format of the PDF
- A display interface to render a PDF page into a canvas element
- A ready-to-use PDF viewer that includes basic features like thumbnails, search, rotate, print and more.

Initially PDF.js was not intended for commercial use but contributions from the open-source community have extended its functionality. Some simple PDF.js customizations you can quickly deploy include:

- Additional display modes
- Page transitions
- Different page layouts
- Document outlines and bookmarks
- Document printing

Adding additional features to PDF.js

In a recent survey, 57 unique organizations stated the most common feature they wanted to add to PDF.js were Annotations (57.1%), Signatures (19%) , Parsing & extraction (14.3%), Form filling (11.9%)



Of these respondents, **71.4% tried to add some of these PDF.js features themselves** but ultimately found it too difficult or time-intensive. If your requirements include extending a PDF.js viewer with this type of functionality, an important first step is to spend time understanding all the costs involved in adding these features.

The costs of customizing PDF.js

If you are considering adding these features to your PDF.js Viewer, there are 3 key costs to investigate:

Development cost

the time it will take to learn, code and deploy features in your viewer

Maintenance cost

the time it will take to maintain and support your custom viewer

Opportunity cost

resources spent developing & maintaining your viewer that are not spent improving your core differentiator

Developing with PDF.js Documentation

When customizing PDF.js there is a risk that documentation for the feature you want to add is non-existent, stale or incomplete. If no documentation exists, development occurs in uncharted territory, making it difficult to predict amount of developer resources needed and the turnaround time. A good first step is to read through closed PDF.js Github issues & open PDF.js Github issues to identify if the features you are interested in adding have been successfully deployed or if documentation is incomplete.

Complex PDF Specification

Depending on the functionality you hope to add, you might need to spend time getting acquainted with the PDF specification (over 1000 pages not including extensions and supplements). For example, if you plan to add annotations to a viewer, you will need to learn how to handle basic rendering instructions, including how to convert PDF annotation coordinates to canvas coordinates.

PDF.js does not have APIs

No API exists for adding features to the PDF.js Viewer UI. If you are planning on adding features like eSignatures or annotations to your viewer's UI, you will need to account for the time it takes to familiarize yourself with the PDF.js code base.

Support during development

Another important consideration is determining how often you rely on support during development. With an open-source solution, there is some uncertainty about response time when opening a new issue. Overall, PDF.js has many active contributors who typically respond within a day or two, particularly for simple issues. For more complex issues (e.g., adding annotation, signature, form filling, etc.) responses can take longer. And if your request falls outside the scope of the project, you are largely on your own.

Unsupported Features	Reason
Form filling	Open Issue for 3 years
Direct annotation (Add, edit, and remove)	Out of scope
Signatures	Open for 7 years
Toggleable visual layers (via OCGs)	Open 8 years
Pinch zoom for mobile	Open 6 years
Night mode	Open 7 years

Cost of maintaining custom PDF.js project

PDF.js has over 6,000 forks and commits typically occur several times a week. Depending on the extent of your customization, these changes might have an impact on your project. Some of the customers we've worked with have encountered situations where their customizations break after PDF.js is updated. The best way to deal with this is to assign a developer to periodically monitor and test your PDF.js viewer customizations.

> "Can someone please fix this issue in pdf js. This issue has ruined our experience in an application that is in production. If we don't get a fix we will have to abandon pdf.js and move to a different product. I don't want to do that as I find pdf.js really simple in its UI and dependencies."

Opportunity cost of PDF.js

When assessing whether to undertake a custom pdf.js project, the first thing to look at is the overall cost. Once you've defined how much time will need to be invested by developer resources, this assessment ends up being a straightforward calculation:

Custom PDF.js cost assessment:

- $45 per hour for developer salary
- 80 to 160 hours of development time
- $900 to $1,800 maintenance cost per year
- 20 to 40 hours for yearly maintenance
- $3,600 to $7,200 to setup viewer

If the cost of using internal resources is lower than a commercial solution, then it would seem obvious to develop in-house. But there is one more cost to consider - the cost of using developer resources to work on something that is not your core differentiator - a cost known as opportunity cost.

> "Opportunity cost is the loss of potential gain from other alternatives when one alternative is chosen."

Opportunity cost is most commonly used when deciding how best to allocate resources to maximize gains. A simple example would be deciding how to allocate developer resources in your product roadmaps:

Option #1:	Option #2:
develop feature 1 to increase new users by 2%	develop feature 2 to increase new users by 1%

Your approach to opportunity cost will likely be driven by your business strategy & philosophy. Some strategies focus resources on their key differentiators and outline the types of work they will not do. Other strategies emphasize developing software internally to avoid reliance on third-party software providers. Dropbox is an example of the latter strategy - they developed their own custom PDF.js viewer but eventually abandoned the project for another open-source solution:

> "Integrating PDF.js with Dropbox was quite difficult, if not downright hacky. PDF.js was designed to be Firefox's integrated PDF viewer, rather than a component of another product." ~Senior Developer, Dropbox"

Opportunity cost of long-term maintenance is another important consideration. If an unexpected bug is discovered or a major update occurs, a developer will need to be pulled off their current project to fix the issue. These unplanned delays can have a direct impact on product release schedules. It can also demotivate and defocus developers who need to jump from their current priority back to supporting the PDF.js project.

Next steps

If you require a simple viewer without features like annotations, signatures or form filling, a PDF.js viewer will be a good fit out-of-the-box. If you plan on

adding custom features to your viewer, however, it's important to understand all the different costs you will encounter. Having an understanding of these costs will help you determine if it makes sense to build your viewer internally or whether to get started with a commercial pdf.js viewer like PDF.js Express.