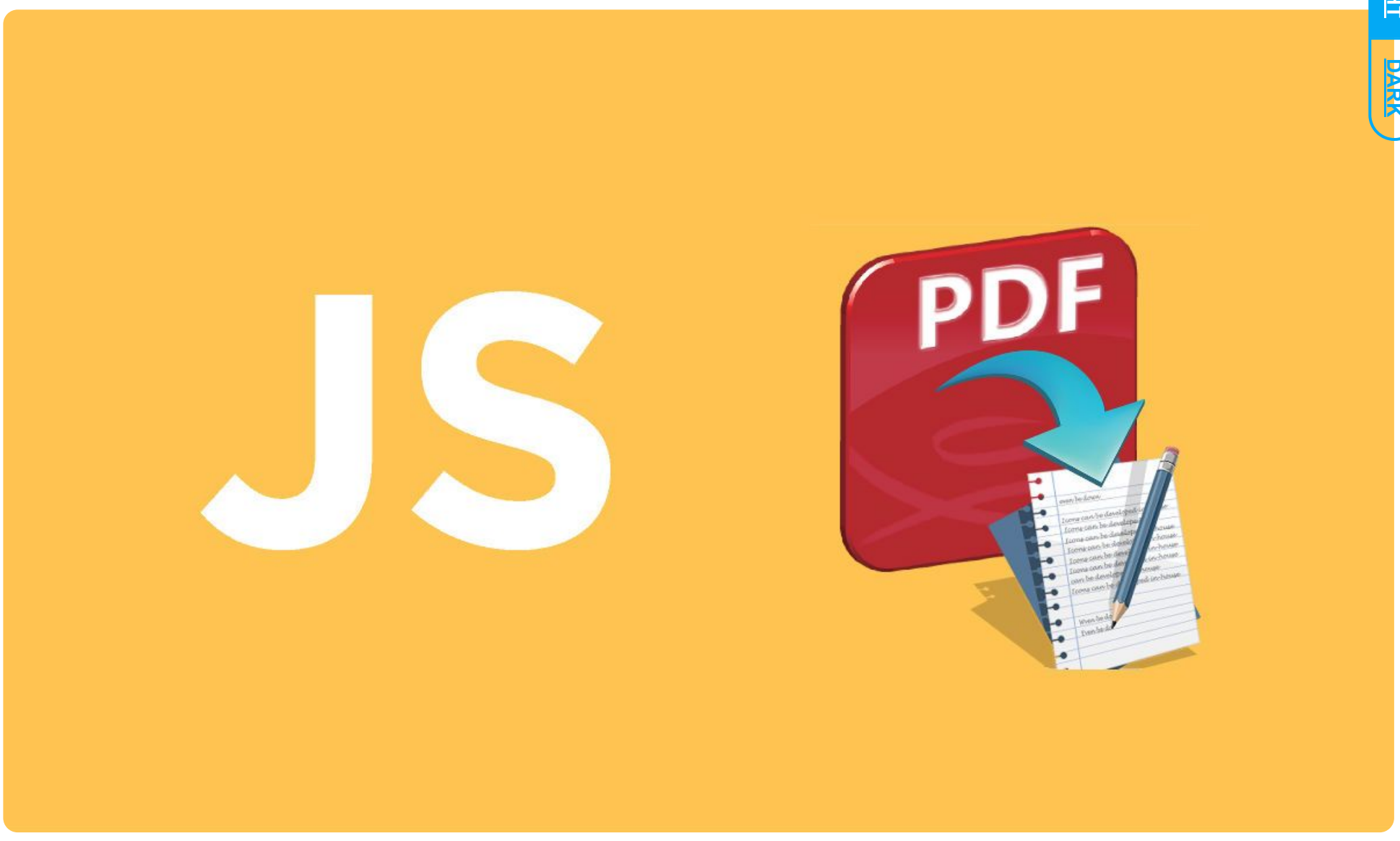
Javascript

How to convert PDF to Text (extract text from PDF) with JavaScript

Carlos Delgado
March 05, 2017 • 107K views

Learn how to extract the text from a PDF with JavaScript using pdf.js



While dealing with Portable Document Format files (PDFs), the user may want to extract all the text from a PDF file. So the user doesn't have to select all the text of a PDF with the mouse and then do something with it.

In this article you will learn how to extract the text from a PDF with Javascript using pdf.js. This library is a general-purpose, web standards-based platform for parsing and rendering PDFs. This project uses different layers, we are going to use specifically 2, the core and the display layer. PDF.js heavily relies on the use of Promises. If promises are new to you, it's recommended you become familiar with them before continuing on. PDF.js is community-driven and supported by Mozilla Labs.

Having said that, let's get started !

Requirements

- Availability of WebWorkers in the browser (see compatibility table).
- The Promises API is frequently used in almost all the methods of pdf.js (you can use a polyfill to provide support for outdated browsers).
- Obviously, a copy of pdf.js (download from the website here).

For more information about pdf.js, please visit the official Github repository here.

1. Include required files

In order to extract the text from a PDF **you will require at least 3 files (2 of them asynchronously loaded)**. As previously mentioned we are going to use `pdf.js`. The Prebuilt of this library is based in 2 files namely `pdf.js` and `pdf.worker.js`. The `pdf.js` file should be included though a script tag:

LIGHT DARK

```
<script src="/path/to/pdf.js"></script>
```

And the `pdf.worker.js` should be loaded through the `workerSrc` method, that expects the URL and loads it automatically. You need to store the URL of the PDF that you want to convert in a variable that will be used later:

```
<script>
    // Path to PDF file
    var PDF_URL = '/path/to/example.pdf';
    // Specify the path to the worker
    PDFJS.workerSrc = '/path/to/pdf.worker.js';
</script>
```

With the required scripts, you can proceed to extract the text of a PDF following the next steps.

2. Load PDF

Proceed to import the PDF that you want to convert into text using the `getDocument` method of `PDFJS` (exposed globally once the pdf.js script is loaded in the document). The object structure of PDF.js loosely follows the structure of an actual PDF. At the top level there is a document object. From the document, more information and individual pages can be fetched. Use the following code to get the PDF document:

Note

To prevent CORS problems, the PDF needs to be served from the same domain of the web document (e.g www.yourdomain.com/pdf-to-test.html and www.yourdomain.com/pdffile.pdf). Besides, you can load the PDF document through base64 directly in the document without make any request (read the docs).

```
var PDF_URL  = '/path/to/example.pdf';

PDFJS.getDocument(PDF_URL).then(function (PDFDocumentInstance) {

    // Use the PDFDocumentInstance To extract the text later

}, function (reason) {
    // PDF loading error
    console.error(reason);
});
```

The `PDFDocumentInstance` is an object that contains useful methods that we are going to use to extract the text from the PDF.

3. Extracting text from a single page

The `PDFDocumentInstance` object retrieven from the `getDocument` method (previous step) allows you to explore the PDF through an useful method, namely `getPage`. This method expects as first argument the number of the page of the PDF that should be processed, then it returns (when the promise is fulfilled) as a variable the `pdfPage`. From the `pdfPage`, to achieve our goal of extracting the text from a PDF, we are going to rely on the `getTextContent` method. The `getTextContent` method of a pdf page is a promise based method that returns an object with 2 properties:

LIGHT DARK

- `items`: Array[X]
- `styles`: Object

We are insterested in the objects stored in the items array. This array contains multiple objects (or just one according to the content of the PDF) that have the following structure:

```
{
    "dir":"ltr",
    "fontName": "g_d0_f2",
    "height": 8.9664,
    "width": "227.1458",
    "str": "When a trace call returns blabla bla ..."
}
```

Do you see something of interest? That's right ! the object contains a `str` property that has the text that should be drawn into the PDF. To obtain all the text of the page you just need to concatenate all the `str` properties of all the objects. That's what the following method does, a simple promise based method that returns the concatenated text of the page when it's solved:

Important anti-hater note

Before you start in the comments to say that instead of concatenating strings using `+=` should be avoided and instead do something like store the strings within an array and then join them, you should know that, based on benchmarks at JSPerf that using `+=` is the fastest method, though not necessarily in every browser. Read more about it here, and in case you don't like it, then modify it as you want.

```
/**
 * Retrieves the text of a specif page within a PDF Document obtained through pdf.js
 *
 * @param {Integer} pageNum Specifies the number of the page
 * @param {PDFDocument} PDFDocumentInstance The PDF document obtained
 **/
function getPageText(pageNum, PDFDocumentInstance) {
    // Return a Promise that is solved once the text of the page is retrieven
    return new Promise(function (resolve, reject) {
        PDFDocumentInstance.getPage(pageNum).then(function (pdfPage) {
            // The main trick to obtain the text of the PDF page, use the getTextContent method
            pdfPage.getTextContent().then(function (textContent) {
                var textItems = textContent.items;
                var finalString = "";

                // Concatenate the string of the item to the final string
                for (var i = 0; i < textItems.length; i++) {
                    var item = textItems[i];

                    finalString += item.str + " ";
                }

                // Solve promise with the text retrieven from the page
                resolve(finalString);
            });
        });
    });
}
```

Pretty simple isn't? Now you just need to write the code previously described:

```
var PDF_URL  = '/path/to/example.pdf';

PDFJS.getDocument(PDF_URL).then(function (PDFDocumentInstance) {

    var totalPages = PDFDocumentInstance.numPages;
    var pageNumber = 1;

    // Extract the text
    getPageText(pageNumber , PDFDocumentInstance).then(function(textPage){
        // Show the text of the page in the console
        console.log(textPage);
    });

}, function (reason) {
    // PDF loading error
    console.error(reason);
});
```

And the text (if there's any) of the first page of the PDF should be shown in the console. Awesome !

4. Extracting text from multiple pages

To extract the text of many pages simultaneously, we are going to use the same `getPageText` method created in the previous step that returns a promise when the content of a page is extracted. As the asynchrony could lead to very problematic misunderstandings, and in order to retrieve the text correctly we are going to trigger multiple promises at

time with `Promise.all` that allows you to solve multiple promises at time in the same order that they were providen as argument (that will help to control the problem of the promises that are executed first than others) and respectively retrieve the results in an array with the same order:

```
var PDF_URL = '/path/to/example.pdf';

PDFJS.getDocument(PDF_URL).then(function (PDFDocumentInstance) {

    var pdfDocument = pdf;
    // Create an array that will contain our promises
    var pagesPromises = [];

    for (var i = 0; i < pdf.numPages; i++) {
        // Required to prevent that i is always the total of pages
        (function (pageNumber) {
            // Store the promise of getPageText that returns the text of a page
            pagesPromises.push(getPageText(pageNumber, pdfDocument));
        })(i + 1);
    }

    // Execute all the promises
    Promise.all(pagesPromises).then(function (pagesText) {

        // Display text of all the pages in the console
        // e.g ["Text content page 1", "Text content page 2", "Text content page 3" ... ]
        console.log(pagesText);
    });

}, function (reason) {
    // PDF loading error
    console.error(reason);
});
```

LIGHT DARK

Live example

Play with the following fiddle, it extracts the content of all the pages of this PDF and append them as text to the DOM (go to the Result tab):

Не удаётся установить соединение с сайтом.

Превышено время ожидания ответа от сайта **jsfiddle.net**.

Попробуйте сделать следующее:

- Проверить, есть ли подключение к интернету.
- Проверить настройки прокси-сервера и брандмауэра.

Перезагрузить

Подробнее

Example

The following document contains a very simple example that will display the content of every page of a PDF in the console. You just need to implement it on a http server, add the pdf.js and pdf.worker.js, a PDF to test and that's it:

```html
<!DOCTYPE html>
<html lang="en">

<head>
    <title></title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
</head>

<body>
    <h1>PDF.js</h1>

    <script src="/path/to/pdf.js"></script>
    <script>
        var urlPDF = '/path/to/example.pdf';
        PDFJS.workerSrc = '/path/to/pdf.worker.js';

        PDFJS.getDocument(urlPDF).then(function (pdf) {
            var pdfDocument = pdf;
            var pagesPromises = [];

            for (var i = 0; i < pdf.numPages; i++) {
                // Required to prevent that i is always the total of pages
                (function (pageNumber) {
                    pagesPromises.push(getPageText(pageNumber, pdfDocument));
                })(i + 1);
            }

            Promise.all(pagesPromises).then(function (pagesText) {

                // Display text of all the pages in the console
                console.log(pagesText);
            });

        }, function (reason) {
            // PDF loading error
            console.error(reason);
        });


        /**
         * Retrieves the text of a specif page within a PDF Document obtained through pdf.js
         *
         * @param {Integer} pageNum Specifies the number of the page
         * @param {PDFDocument} PDFDocumentInstance The PDF document obtained
         **/
        function getPageText(pageNum, PDFDocumentInstance) {
            // Return a Promise that is solved once the text of the page is retrieven
            return new Promise(function (resolve, reject) {
                PDFDocumentInstance.getPage(pageNum).then(function (pdfPage) {
                    // The main trick to obtain the text of the PDF page, use the getTextContent method
                    pdfPage.getTextContent().then(function (textContent) {
                        var textItems = textContent.items;
                        var finalString = "";

                        // Concatenate the string of the item to the final string
                        for (var i = 0; i < textItems.length; i++) {
                            var item = textItems[i];

                            finalString += item.str + " ";
                        }
```

LIGHT DARK

```
                    // Solve promise with the text retrieven from the page
                    resolve(finalString);
                });
            });
        });
    }
</script>
</body>

</html>
```

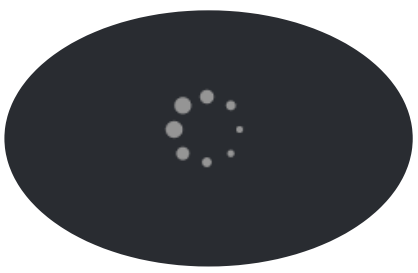
Text isn't being retrieved

If you already tried the code and not any kind of text is being obtained, **is because your pdf probably doesn't has any**. Probably the PDF text that you can't see is not text but an image, then the process explained in this process won't help you. You can use another approaches like the Optical Character Recognition (OCR), however this isn't recommended to do in the client side but in the server side (see a Node.js usage of OCR or with PHP in Symfony).

Happy coding !

javascript pdf pdf to text extract text pdf text get text

Share this article



Carlos Delgado

Author

Senior Software Engineer at EPAM Anywhere. Interested in programming since he was 14 years old, Carlos is a self-taught programmer and founder and author of most of the articles at Our Code World.

LIGHT DARK

15 Comments

Add Your Comment

Become a more social person

Search

Search

Search

Related Articles

How to convert PDF to Text (extract text from PDF) with PHP in Symfony 3

April 05, 2017 • 40.2K views

[How to convert a PDF to an image with PHP in Symfony 3]

August 25, 2017 • 17.7K views

[How to convert a Word file to PDF (docx to pdf) in LibreOffice with the CLI in Ubuntu 20.04]

April 15, 2021 • 31.3K views

[How to convert images to text with pure JavaScript using Tesseract.js]

December 25, 2018 • 60.6K views

[How to trigger the direct download of a PDF with JavaScript]

August 19, 2017 • 92.6K views

[How to convert text to speech (speech synthesis) in Cordova]

February 02, 2017 • 18.2K views

Follow Us

LIGHT DARK