How to Perform Full-Text Search in PDFs Using JavaScript

Introduction

Full-Text Search is a method of searching a document or a collection of document using its metadata or part or the original document such as titles or a selected section. A search engine examines every word in the document and tries to match each word with the user's query.

What we'll cover in this article:

- Why does full-text search on PDFs often fail?
- What is Foxit?
- How does Foxit SDK let you search text in a PDF?
- Building the APP
- Setting up a new JavaScript web app
- How to allow users to search a PDF with the Web SDK

Why does full-text search on PDFs often fail?

PDF (Portable Document Format) files are made to keep the originality of a document, like layouts, fonts and graphics. you can view a PDF file on any computer or device that has PDF support and it'll still maintain its originality so for a text search to be successful, the texts needs to be extracted from the PDF which can be difficult without the use of software. In this Article, i will show you how to use Foxit SDK and JavaScript to perform a successful full text search.

What is Foxit

Foxit is a software that offers various PDF solutions. The software can be used to create, edit, sign, merge, annotate, protect, scan PDF files and do alot more. Foxit also provides easy to use collaboration features like filling out forms and sharing information with friends and colleagues. It
renders PDF files relatively quick no matter how large the file is and uses a very little memory which is important.
The company also provides SDKs and Plugins for developers to plug into their apps which i will be talking about in this article.
Foxit is available on almost all platforms visit https://www.foxit.com/ for more.

How does Foxit SDK let you search text in a PDF

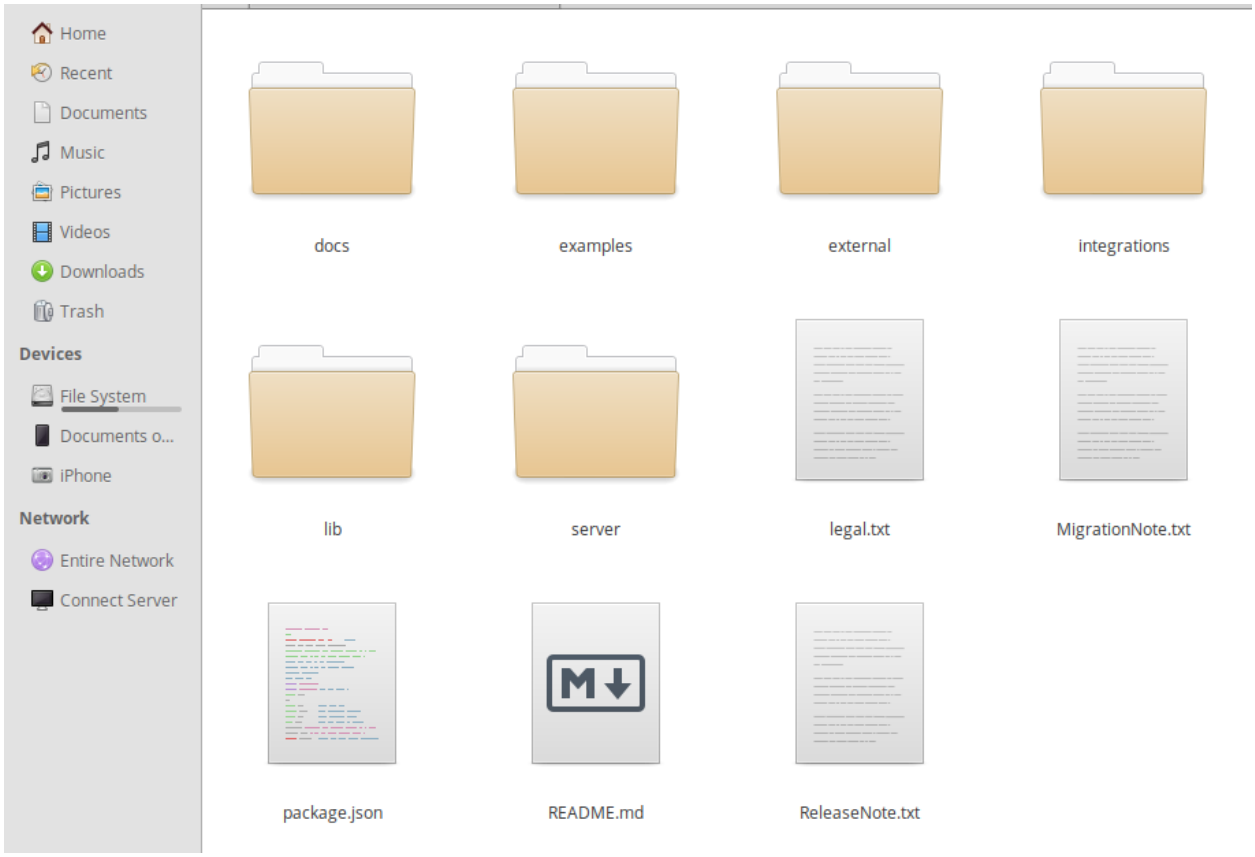
The problem with finding a text in a PDF is the way PDF format organizes text and objects. The SDK take notes of these objects (or characters) based on the location, size or rotation angle to be displayed. This makes finding words in your document easy as the SDK lets you customize the search engine to account for common occurrences. It applies to every text in the PDF according to the index of the text, overcoming document encoding type and language. The software uses SQLite to check the document which is returns a quick response.

Building The App

To get started, first you'll need to have the following

- Foxit Web SDK (https://developers.foxit.com/pdf-sdk/download/)
- Node and NPM

Step 1.
Download SDK the Zip and extract it. The folder structure should look like this



Step 2
After extracting the app, you'll find a package.json file that contains all packages used. You'll need to install the packages
Open your command prompt or terminal, change directory to the folder that contains the extracted files.
run
npm install

To install packages

Step 3
After installing the packages next step is to start the local server
run
npm start

Step 4
open your browser and access the following address

Complete webviewer : http://localhost:8080/examples/UIExtension/complete_webViewer/
Basic Webviewer: http://localhost:8080/examples/UIExtension/basic_webViewer/

Setting Up a new JavaScript WebApp With Foxit

We will be using the Foxit PDF SDK to build a web app that has a full PDF viewer feature. Just follow the instructions below. To get started,

- Create a new folder for the project
- From the SDK you downloaded earlier, Copy the **`lib`**, **`server`**, and **`external`** folders and also the " **package.json**" file into the new folder you created. (only copy the external folder if you want to use font resources).
- Add a PDF file to the new folder also (we'll be using this to test).
- Lastly Create an index.html file in the new folder.

Now this is what your file structure should look like:

```
newFolder
   +-- lib          (copied from the Foxit_PDF_SDK)
   +-- server       (copied from the Foxit_PDF_SDK)
   +-- package.json (copied from the Foxit_PDF_SDK)
   +-- index.html   (You created this file)
   +-- youOwn.pdf   (sample pdf you added to the folder)
   +-- external     (optional file from Foxit_PDF_SDK for font resources)
```

Lets Start writing codes. Open your Code editor and add the following code snippet in your index.html

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Foxit Web SDK Practice</title>
    <style>
      .fv__ui-tab-nav li span {
        color: #636363;
      }

      .flex-row {
        display: flex;
        flex-direction: row;
      }
```

```
  </style>
 </head>

 <body>

 </body>

</html>
```

lets import styles from the `lib` file we copied. We'll be adding it to the `<head>` tag

```
<link rel="stylesheet" href="./lib/PDFViewCtrl.css">
```

also Import the script library from the `lib` folder:

```
<script src="./lib/PDFViewCtrl.full.js" charset="utf-8"></script>
```

Add a `<div>` element between the `<body>` tag, this would be the webViewer container:

```
<div id="pdf-viewer"></div>
```

Initialize PDFViewer before the closing body tag:

```
 <script>
  const licenseSN = "Your license SN";
  const licenseKey = "Your license Key";

  const PDFViewer = PDFViewCtrl.PDFViewer;
  const pdfViewer = new PDFViewer({
   libPath: './lib', // the library path of Web SDK.
   jr: {
    licenseSN: licenseSN,
    licenseKey: licenseKey,
   }
  });
  pdfViewer.init('#pdf-viewer'); // the div (id="pdf-viewer")
 </script>
```

you can get the trail license key and license SN from the `license-key.js` file in the examples folder, from the SDK folder

get the PDF document:

```
  fetch('./JavaScript-for-Kids.pdf').then( (res) => {
    //modify the path to get your pdf
    res.arrayBuffer().then( (buffer) => {
      pdfViewer.openPDFByFile(buffer);
    })
  })
```

These are the key settings we need to set up foxit, now you can view from the browser. open your `index.html` file. The complete html file should look like this

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Foxit Web SDK Practice</title>
  <link rel="stylesheet" href="./lib/PDFViewCtrl.css">
  <script src="./lib/PDFViewCtrl.full.js" charset="utf-8"></script>
  <style>
    .fv__ui-tab-nav li span {
      color: #636363;
    }
    .flex-row {
      display: flex;
      flex-direction: row;
    }
  </style>
</head>
<body>
  <div id="pdf-viewer"></div>


  <script>
    const licenseSN = "Your license SN";
    const licenseKey = "Your license Key";
    const PDFViewer = PDFViewCtrl.PDFViewer;
    const pdfViewer = new PDFViewer({
      libPath: './lib', // the library path of Web SDK.
      jr: {
          licenseSN: licenseSN,
          licenseKey: licenseKey,
```

```
    }
  });
  pdfViewer.init('#pdf-viewer'); // the div (id="pdf-viewer")

  fetch('./JavaScript-for-Kids.pdf').then((res) => {
    // modify the path to get your pdf
    res.arrayBuffer().then(function (buffer) {
      pdfViewer.openPDFByFile(buffer);
    })
  })
 </script>
</body>
</html>
```

Integrating the complete Web Viewer package

We just finished setting up the Basic package… let move to the complete web-view package

import the styles

```
<link rel="stylesheet"  href="./lib/UIExtension.css">
```

import the script

```
<script src="./lib/UIExtension.full.js" charset="utf-8"></script>
```

In the body tag add a div tag

```
<div id="pdf-ui"></div>
```

Initialize the Complete package extension

```
  const pdfui = new UIExtension.PDFUI({
    viewerOptions: {
      libPath: './lib', // the library path of web sdk.
      jr: {
        licenseSN: licenseSN,
        licenseKey: licenseKey
      }
    },
    renderTo: '#pdf-ui' // the div (id="pdf-ui").
  });
```

Lunch the PDF

```
fetch('./JavaScript-for-Kids.pdf').then((res) => {
  // modify the path to get your pdf
  res.arrayBuffer().then((buffer) => {
    pdfui.openPDFByFile(buffer);
  })
})
```

How Users can Search a PDF with the Web SDK

If the complete web-view SDK is plugged into your app users can easier navigate the sidebar and find the search bar icon
However you can also implement custom controls for your WebApp, the SDK gives you the freedom to magnify it to suite your project.