DEVELOPER GUIDE

Foxit PDF SDK for Web

Microsoft® Partner
Gold Independent Software Vendor (ISV)

Table of Contents

1 Foxit PDF SDK for Web OverView

By using Foxit PDF SDK for Web, developers can deploy and customize a Foxit PDF SDK for Web that supports viewing PDF documents within a web browser. Integrating a Foxit PDF SDK for Web into a zero-footprint web app allows end users to view PDF documents on desktop and mobile devices without installing anything.

1.1 Why Foxit PDF SDK for Web is your choice

Foxit is an Amazon-invested leading software provider of solutions for reading, editing, creating, organizing, and securing PDF documents. Foxit PDF SDK for Web is a cross-platform solution for PDF online viewing. Foxit PDF SDK for Web enterprise edition has been chosen by many of the world's leading firms for integration into their solutions. Customers choose this product for the following reasons:

Fully customizable
Developers can easily design a unique style for their Foxit PDF SDK for Web interface, and make it consistent to their web applications.

Easy to integrate
Developers can easily reference Foxit PDF SDK for Web by referring to resource files and writing a small amount of code to display and edit PDF files, and have a wealth of interfaces to connect users and user data.

Standard and consistent annotation data
The annotations in Foxit PDF SDK for Web are consistent when viewing and editing in other applications.

Powered by Foxit's high fidelity rendering PDF engine
The core technology of Foxit PDF SDK for Web is based on Foxit's PDF engine, which is trusted by a large number of well-known companies. Foxit's powerful engine makes document viewing fast and consistent in all environments.

In addition, Foxit's products are offered with the full support of our dedicated support engineers if support and maintenance options are purchased. Updates are released on a regular basis. Foxit PDF SDK for Web will be the most cost-effective choice if you want to develop a cross-platform PDF document viewing solution that can control document distribution.

1.2 Audience and Scope

This document is primarily intended for developers who need to integrate the Foxit PDF SDK for Web into their web applications. It includes the direct reference examples as well as custom front-end APIs for customization.

1.3 Your Web Application

Foxit PDF SDK for Web provides a solution that enables a web application to view PDFs seamlessly without any plugins or local applications. Developers should prepare a PDF hosting server like Nginx or XAMP and do the usual configuration before using Foxit PDF SDK for Web.

1.4 Features

Foxit PDF SDK for Web provides the most common PDF viewing features and allows developers to incorporate powerful PDF technology to their applications like viewing, text searching, printing, form filling and annotating PDF documents.

Developers can use the default sample Reader interface in the package or develop a custom interface

Features	Descriptions
PDF View	Go to page, Zoom in/out, Fit width, Bookmark, Thumbnail, Print, Rotate
PDF Search	Quick search and Advanced search
Annotation	Show/Hide all existing annotations， Import/Export annotations 22 annotation tools for editing (Highlight, Area Highlight, Underline, Squiggly, Strikeout, Replace Text, Insert Text, Note, Typewriter, Callout, Textbox, Rectangle, Oval, Line, Arrow, Polygon, Polyline, Cloud, Pencil, Stamp, Distance, Add Image).
PDF Editing	Text selection and copy, Text Content Editing, Shape creating and editing.
PDF Pages	Insert pages, Flatten pages.
PDF Layers	Show and hide layers.
Signature	Ink Signature
Form	Form filling, Export/Import PDF form data
Security	Open Password, Document Restrictions, User Watermarks, and Redaction

Rendering engine	JavaScript Rendering in local browser.

1.5 Evaluation

Foxit PDF SDK for Web allows users to download the trial version to evaluate the SDK. The trial version is the same as the standard version except for the 30-day limitation for free trial and the trial watermarks in the generated pages. After the evaluation period expires, customers should contact the Foxit sales team and purchase licenses to continue using Foxit PDF SDK for Web.

1.6 License

Developers are required to purchase licenses to use Foxit PDF SDK for Web in their solutions. Licenses grant users permission to release their applications based on Foxit PDF SDK for Web. However, users are prohibited to distribute any documents, sample codes, or source codes in the released packages of Foxit PDF SDK for Web to any third party without the permission from Foxit Software Incorporated.

2 Getting Started

2.1 Understanding the Package Structure

docs	API reference, tutorials and samples.
images	Image resources for web viewer.
pc	Wrapper page for Foxit DRM.
scripts	SDK Library files and JavaScript Demo files
styles	CSS Style files
index.html	The PC HTML entry file for the basic demo Viewer.
Legal.txt	Legal and copyright information
mobile.html	The mobile HTML entry file for the basic demo Viewer
offline.appcache	Cache file for app offline.
ReleaseNotes.txt	Release Information
translation-en-US.json	The default English language file.
translation-zh-CN.json	The Chinese localization file.

2.2 Quickly run Foxit PDF SDK for Web Demo

You can use http -server to start a simple local web server, and then run Foxit PDF SDK for Web Demo.

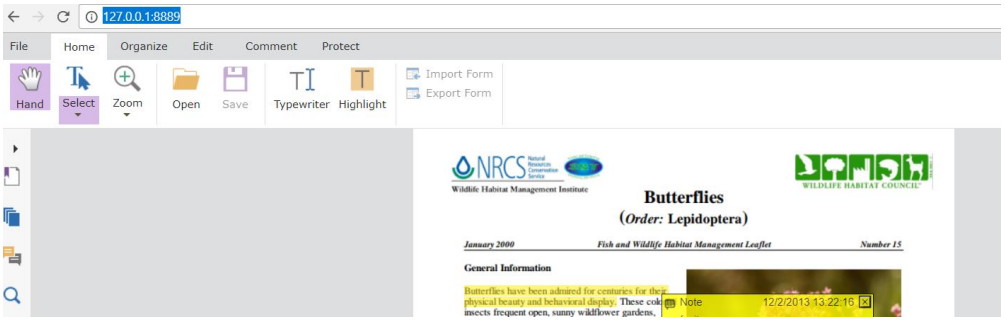
In the following working example, npm package is assumed available on your system already.

1. Download FoxitPDFSDKForWeb.zip.
2. Unzip it to create a folder 'lib' for example to your server.
3. Enter into the above built folder, right-click to start a command window, input follows in the command line

```
http-server -p 8889
```

```
Starting up http-server, serving ./
Available on:
  http://10.203.23.91:8889
  http://127.0.0.1:8889
```

4. As the above example shows, the local http server is available on port 8889. Now you can run the Demo at http://127.0.0.1:8889.



2.3 Integrate to your project and create your own web page

1. Create a new directory as a project folder, such as D:/www.
2. Download FoxitPDFSDKForWeb.zip, unzip it to D:/www/lib for example.
3. Under the 'www' folder, create an HTML file. The directory structure is:

```
www
    +-- lib
    |    +-- <SDK package files ...>
    +-- index.html
```

If you are on port 8080 for example. You can access the Foxit PDF SDK for Web Demo at http://localhost:8080/index.html after setting up the HTML page. Now Let's add elements to the html page step by step.

1. Add a style to the <head> tag of the HTML page.

```
<link rel="stylesheet"
href="//fonts.googleapis.com/css?family=Open+Sans:300,400,600,700|PT+Sans+Narrow|Source+
Sans+Pro:200,300,400,600,700,900&amp;subset=all">
<link rel="stylesheet" href="./styles/webpdf.full.mini.css">
```

2. In the HTML <body> tag, add the elements as the web viewer container, and add the dependencies of JavaScript files.

```
<div id="docViewer" style="background: #dddedf;"></div>

    <script type="text/javascript" src="./scripts/jquery-3.3.1.min.js"></script>

    <script type="text/javascript" src="./scripts/jquery-migrate-
3.0.1.min.js"></script>

    <script type="text/javascript" src="./scripts/jquery-ui.min.js"></script>

    <script type="text/javascript" src="./scripts/jquery.form.min.js"></script>

    <script type="text/javascript"
src="./scripts/release/webpdf.full.mini.js"></script>

    <script type="text/javascript" src="./scripts/control/toolbar/toolbar-
config.bundle.js"></script>

    <script type="text/javascript" src="./scripts/demo-license-key.js"></script>

    <script type="text/javascript" src="./scripts/control/pc/demo.js"></script>
```

- **jquery.js:** Required by Foxit PDF SDK for Web, so it must be included.
- **webpdf.full.mini.js:** SDK core library file.
- **toolbar-config.bundle.js:** Contains the initial toolbar layout and settings. If not referenced here, viewer will be loaded without toolbar.
- **demo-license-key.js:** Store the license information with a global variable window.demoLicenseKey. If not referenced here, demo will not be licensed.

3. Add the following script to create a new instance of Foxit PDF SDK for Web and load a document. The `'doc'` should be the path to one of your PDF files.

```
<script>
    var docViewerId = 'docViewer';
        $(document).ready(function(){
            var optionsParams = {
                language: window.getLanguage(),
                toolbarConfig: window.toolbarConfig,
                fontUrl: 'http://webpdf.foxitsoftware.com/webfonts/',
                licenseKey: window.demoLicenseKey,
            };
            WebPDF.ready(docViewerId, optionsParams, false).then(function(){
                openDocument();
            });
            function openDocument(){
                var fileurl = WebPDF.baseUrl + 'doc';
                var openFileParams = {
                    url: fileurl,
                    fileId: fileurl
                };
WebPDF.ViewerInstance.openFileByUri(openFileParams).catch(function(error) {
                    console.log(error);
                });}
                });
       </script>
```

2.4 **Activating** Foxit PDF SDK for Web

A license string is included in the 'sdkkey.txt' that you will receive via Email once purchasing the SDK. Find and open the 'sdkkey.txt', copy this string, open "..\scripts\demo-license-key.js" and paste it in the required place as shown in the following code.

```
window.demoLicenseKey ='paste your license string here'
```

2.5 Web Server Configuration Sample

2.5.1 **Nginx** Working **Sample**

1. Create a Nginx configuration file, for example FoxitWebPDFjs.conf.
2. Set up a virtual server and configure a location. The example below demonstrates the access to your own HTML page. The 'index.html' is your web page, and 'D:/www/' is your built project directory where the SDK folder resides in.

```
server {
    listen 8080;
    server_name 127.0.0.1;

    location / {
        index index.html;
        charset utf8;
        root "D:/www/";
    }}
```

The example below demonstrates how to run SDK index.html directly. The 'D:/WWW/Lib' is the path to the SDK.

```
server {
    listen 8080;
    server_name 127.0.0.1;
        location / {
        alias D:/WWW/Lib/;
        charset utf8;
    }}
```

3. In the browser, you may access the page at: http://localhost:8080/index.html

2.5.2 XAMPP Working Sample

Assuming XAMPP was installed on D:\xampp.

1. Download FoxitPDFSDKForWeb.zip.

2. Unzip it into 'D:\xampp\htdocs\FoxitPDFSDKForWeb.

3. Restart Apache service.

4. In the browser, you may access the page at: http://localhost/FoxitPDFSDKForWeb/index.html .

2.6 Understanding Demo.js

Demo.js is the JavaScript sample and configuration file corresponding to the entry index.html of the default demo viewer. This file shows how to initialize the viewer and use the API under WebPDF.ready for actions such as open files, import user data and user settings and bind events.

You can define your customization on this file, or create your own JavaScript file and reference it in your HTML page.

2.7 Understanding WebPDF.Ready

WebPDF.ready is the SDK entry point. In the API reference, except for the version, onReady, and ready in the namesapce 'WebPDF', all the other namespaces and variables that will be mounted after the .ready call.

WebPDF.ready(docViewerId, optionParams)

The first parameter docViewerId receives is the DOM id name of the web viewer.

The second parameter supports a series of initializing options. Except for the licenseKey (which is mandatory), these parameters are optional.

2.8 Initializing **Options**

Before you initiate the SDK, you can set options once the DOM is located. Only the option licenseKey is obligatory for authenticating Web SDK, the others are set according to your needs. The following code snippet demonstrates we want web viewer to use the browser's top language, load the default toolbar configuration, use the font from the Foxit font server to render the PDF and load license key for SDK. For more options, refer to WebPDF in the API References.

```
var docViewerId = 'docViewer';
    $(document).ready(function(){
            var optionsParams = {
                language: window.getLanguage(),
                toolbarConfig: window.toolbarConfig,
                fontUrl: 'http://webpdf.foxitsoftware.com/webfonts/',
                licenseKey: window.demoLicenseKey,
                // i18nPath:'',
                // toolbarConfig:window.toolbarConfig,
                // isInitI18n: false
                // appName: '',
                }});
WebPDF.ready(docViewerId, optionsParams)
```

2.9 Document Loading

2.9.1 **Loading Document**

On the default demo viewer, you can click on **Open** to open a local PDF or open a PDF by URL.

You can also programmatically open a PDF by using openFileByUri to open a PDF file from a file hosting system. Or using openFileByStream to open a PDF from stream. The following code snippet demonstrates how to load the sample document "sample.pdf" from stream.

```
   $function openDocument(){
    var url = 'http://10.103.4.154:63051/doc/sample/sample.pdf';
    var xhr = new XMLHttpRequest();
    xhr.open('GET', url, true);
    xhr.responseType = 'arraybuffer';
    xhr.send();
    xhr.onload = function(e) {
    var responseArray = new Uint8Array(this.response);
    var fileId = "Wed, 29 Dec 2018 09:31:20 GMT";
    WebPDF.ViewerInstance.openFileByStream(responseArray, fileId,
"sample.pdf").catch(function(error) {
                    console.log(error);
                });}
      }
```

2.9.2 Asynchronous Loading

When opening a large file, the best practice is to enable asynchronous loading mode in Foxit PDF SDK for Web and enable Range request header in the file hosting server. In the async mode, the viewer will make bytes range requests to download portions of content and render instead of loading the entire file before rendering. In async mode, the document is read-only.

// when a file reaches 400mb, use async mode to open

```
WebPDF.ViewerInstance.openFileByUri({
    fileId: 'file ID',
    url: '', // file download url,
    fileSize: 400r, // When a file reaches this size, start async loading.
    isAsyncMode: true // enable async loading mode
 });
```

About Range Request

The RFC2616 specification defines the range protocol, which gives a rule that allows the client to download only a portion of the complete file at a time. This allows the client to download a file while multithreading is enabled, where each thread only download parts of the file, then assemble into one complete file. Range also supports breakpoint transmission. As long as the client records the downloaded file offset, the client can request the server to send the file on the breakpoint.

// Accept-Ranges on the server

```
Accept-Ranges     = "Accept-Ranges" ":" acceptable-ranges
 acceptable-ranges = 1#range-unit
```

2.10 Loading and Exporting annotations

Foxit PDF SDK for Web is a browser-based application. It doesn't implement the saving annotation feature in the viewer demo. As such, to save the annotation changes in demo, you can either click Download to save the changes to current PDF and save as a local copy or click Export to save annotation in a type of data format. You can also use "export/import" APIs to implement programmatically annotations loading and exporting. For more information, refer to Importing and Exporting.

You can use one of the following methods to import your annotations:

WebPDF.ViewerInstance.importAnnotsFromFDF(params)

WebPDF.ViewerInstance.importAnnotsFromJson(annotsJson)

WebPDF.ViewerInstance.importAnnotsFromXFDF(bufferArray)

Use one of the following methods to export your annotations:

WebPDF.ViewerInstance.exportAnnotsToJson()//You can customize the storage database

WebPDF.ViewerInstance.exportAnnotsToXFDF()//Export and download to the local machine.

WebPDF.ViewerInstance.exportAnnotsToXFDFStream(callback)//You can customize the storage database

If you need to clear annotations, you can use import method to implement it, for example:

WebPDF.ViewerInstance.importAnnotsFromJson(

{ n:'annotation-name', del:1 })

Use the following method to get annotations data of a specific page in Json format:

WebPDF.ViewerInstance.getPageAnnots(pageIndex, callback)

2.11 Importing and Exporting

Basically, you can click on import/export button on the toolbar to import or export annotations forms data. You can also programmatically fulfill the task. The supported data formats are: fdf, xfdf, Json and xml (form only). The code sample below demonstrates methods to import annotations from xfdf and export annotations to Json.

Use XMLHttpRequest(); to get xfdf stream and import annotations data

```
WebPDF.ViewerInstance.importAnnotsFromXFDF(bufferArray)
 var url = 'http://localhost:8080/docs/sample/Annot_all.xfdf';
    var xhr = new XMLHttpRequest();
    xhr.open('GET', url, true);
```

```
    xhr.responseType = 'arraybuffer';
    xhr.send();
    xhr.onload = function(e) {
        var responseArray = new Uint8Array(this.response);
        WebPDF.ViewerInstance.importAnnotsFromXFDF(responseArray);
    }
```

Use FileReader to get xfdf file stream and import annotations

```
//use FileReader to get xfdf file steam and import and import annotations
var fileReader = new FileReader();
    fileReader.readAsArrayBuffer(file);
    fileReader.onload = function (e) {
      var responseArray = new Uint8Array(this.result);
          WebPDF.ViewerInstance.importAnnotsFromXFDF(responseArray);}
```

Export annotations to Json file

```
//export annotations to Json data
WebPDF.ViewerInstance.exportAnnotsToJson([
    {
        "number": 0,
        "annots": [
            {
                "bs": 0,
                "c": "Butterflies have been admired for centuries for their
physical beauty and behavioural display.",
                "ca": 1,
                "cd": "1384916316",
                "cl": "#ffff00",
                "f": 28,
                "md": "1385961736",
                "n": "c97db9f7-5645-4d06-b5f4-6f9bc91231c2",
                "pop": {
                    "f": 28,
                    "md": "1384887516",
                    "n": "7f53f835-1c8b-4a2e-80c7-992f9469b58b",
                    "op": 1,
                    "rc": [
                        281,
                        606,
                        460,
                        486
                    ]
```

```
            },
            "r": 0,
            "rc": [
                335.69,
                490.19,
                355.69,
                470.19
            ],
            "subj": "Note",
            "subty": "Text",
            "tit": "foxit",
            "txtn": "Comment",
            "ty": "Markup"
        }]}])
```

2.12 Localization

Foxit PDF SDK for Web includes two international language files in the package, the default 'translation-en-US.json' and the localization file 'translation-zh-CN'. If you want to use your localization language 'abc-Lng' for example, the simplest way is to:

1. Duplicate 'translation-en-US.json' and rename as 'translation-abc-Lng.json', where abc-Lng is your language. Translate the resources with your language.
2. Remove or rename other language files. Your 'translation-abc-Lng.json' file will be used by SDK.

If you want to keep the other languages but call your built language, you can set options in the WebPDF.ready.

```
WebPDF.ready('docViewer', {
    licensekey : 'window.demoLicenseKey'
    language:  'abc-Lng'})
```

2.13 Offline

To access the web viewer on your server in a browser, the user is required a network connection. What if they are offline but still want to access Viewer? You can achieve that using Foxit PDF SDK for Web by referencing 'offline.appcache' in your HTML file:

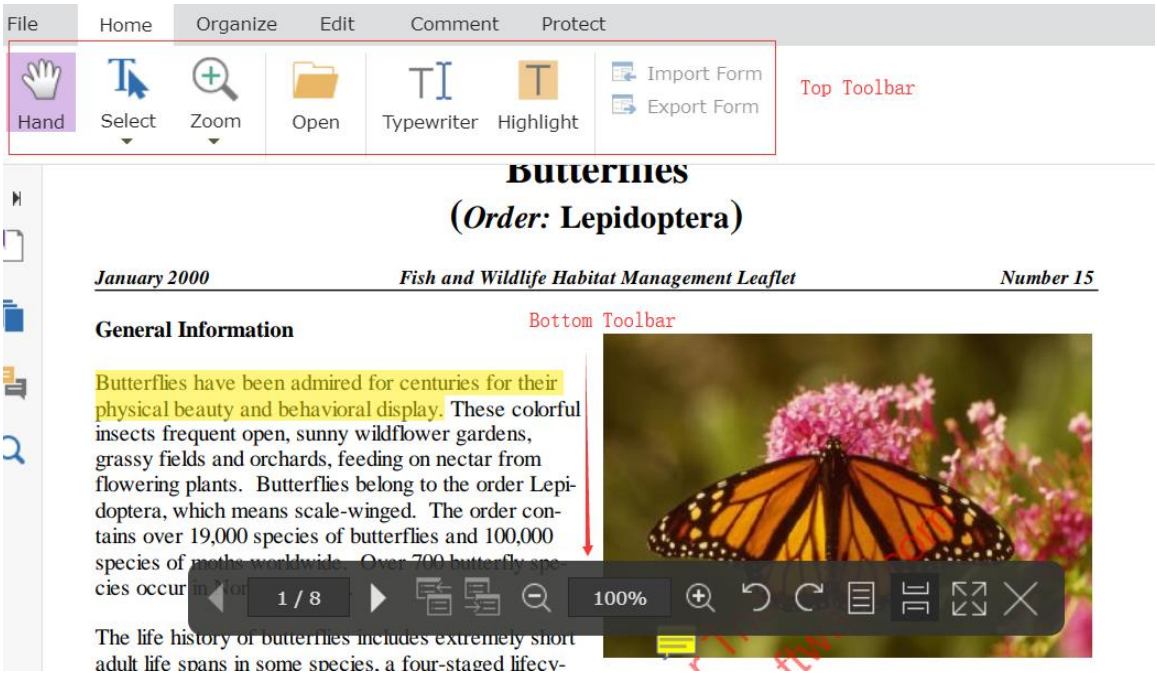
<html lang="" manifest="./offline.appcache">

The default demo viewer will be read-only once it is in offline mode. User operations involving network resources reference like font loading will suffer failures.

3 Configuring Toolbar

3.1 Introduce to Toolbar

Foxit PDF SDK for Web Demo provides a default toolbar global configuration file called '..\scripts\control\toolbar\toolbar-config.bundle.js', through which you can customize both the top toolbar and bottom toolbar.



You can modify the toolbar-config.bundle.js to redefine or add a button. You can also create a new toolbar.config.js file with your own configuration. You may configure the global variable object in the ' toolbar-config.bundle.js'', and declare this file in the entry HTML file, then pass the toolbar-config.bundle.js to WebPDF.ready to see the toolbar configuration take effect. For configuration samples, refer to .. \docs\tutorial\toolbar_en.html.

Introduce the default 'toolbar-config.bundle.js' file and pass the configuration to

```
WebPDF.ready('id', {
    toolbarConfig: window.toolbarConfig
    })
```

You don't need to explicitly call the toolbar initialization API *new WebPDF.Toolbar(window.toolbarConfig, docViewerId).initialize();* otherwise, you will call

```
WebPDF.ready('id', {}).then(function(){
    new WebPDF.Toolbar(window.toolbarConfig, 'id').initialize()
    });
```

3.2 Customize your own toolbar layout

Don't want the default tab toolbar mode? Then you can disable the default one design your won toolbar layout. Click HERE for code samples and result presentation.

3.3 General Procedures to add a new button

To add a new button, you will go through the following steps:

1) In the 'toolbar-config.bundle.js'

   // locate the tab section a button be added.

```
sub: [{
    type: 'custom_button', // custom component's name
    handler: 'myswitcher', // custom controller's name
    text: 'Hand',
    params: {
        toolname: 'Hand'
    }}
```

2) In the demo.js after WebPDF.ready:

   // register the component by calling WebPDF.Toolbar.getRegistry() . See Custom Component Samples in the 'toolbar_en.html'.

   // register the controller by calling WebPDF.Toolbar.getRegistry() . See Custom Controller Samples in the 'toolbar_en.html'.

   // initialize toolbar by calling *new WebPDF.Toolbar(window.toolbarConfig).initialize();*

NOTE:

- For detailed guide samples, refers to 'toolbar_en.html'.
- Click HERE to see samples of configuring a button and handler.

3.4 Activate Save button for annotation

In the default demo viewer, the save annotation button is deactivated. To activate it in the demo viewer, you must go through programmatic steps blew:

1) In the 'toolbar-config.bundle.js':

// configure the save and controller:

{
extend: 'save',
handler: 'saveToDatabase'
}

2) In the Demo.js:

// register the controller calling the getRegistry() function

var registry = WebPDF.Toolbar.getRegistry();

// initialize the toolbar by calling the initialize() function in the Toolbar object

new WebPDF.Toolbar(window.toolbarConfig).initialize();

// modify the saveuserdata data to ensure that the data source is consistent with the save location.

exportAnnotsToJson()

importAnnotsFromJson()

3.5 View Rotate

With the default demo viewer, you can rotate the page view 90 degrees by right-clicking on a page. To programmatically rotate a page view, call the "WebPDF.ViewerInstance.rotate()" after the WebPDF.ready.

```
// rotate all pages in 90 degrees clockwise
WebPDF.ViewerInstance.rotate('right');
// rotate all pages in 90 degrees counterclockwise
WebPDF.ViewerInstance.rotate('left');
//sample to listen for the related event
WebPDF.
ViewerInstance.on(WebPDF.EventList.DOCVIEW_ROTATE_CHANGED,function(event,data){
*       alert('Old rotate:' + data.oldRotate + ', New rotate:' + data.newRotate);
})
```

3.6 ZOOM

The Zoom group includes tools like Zoom in, Zoom Out, Fit Width, Fit Page and zoom levels. In the default demo viewer, you can click on the plus and minus buttons to zoom in and out of the document, or select a zoom level by choosing a specific value from the drop-down list. To programmatically implement zoom, call the "WebPDF.ViewerInstance.zoom()" after the WebPDF.ready.

```
//zoom in twice
WebPDF.ViewerInstance.zoomTo(2);
//sample to listen for the related event
WebPDF. ViewerInstance.on(WebPDF.EventList.DOCVIEW_ZOOM_CHANGED,function(event,data){
*      alert('Old scale:' + data.oldScale + ', New scale:' + data.newScale);
})
```

Zoom Value Description:

Zoom Value	Description
Number	Zoom Ratio
WebPDF.ZOOM_IN	Enlarge the page according to the array value returned by WebPDF.ViewerInstance.getZoomLevels(), and will not continue to zoom in if the current zoom ratio is the maximum.
WebPDF.ZOOM_OUT	Shrink the page according to the array value returned by WebPDF.ViewerInstance.getZoomLevels (), and will not continue to zoom out if the current zoom ratio is the minimum.
WebPDF.ZOOM_FIT_WIDTH	Fit to width, zoom ratio is adaptive to browser width changes.
WebPDF.ZOOM_FIT_PAGE	Fit to the page, zoom ratio is adaptive to browser height and width changes

3.7 Download

Foxit PDF SDK for Web implements Download function. At some cases you may need to disable it. The following shows a way of disabling the Download.

In the ..\scripts\control\toolbar\file-tab.html

*Find and delete :<li data-action="download" data-i18n="[html]PCLng.ToolBar.SaveTo.local" class="disabled">Download
*

In the ..\scripts\control\toolbar\file

replace 'template: require$$0,' with templateUrl:'<IndexPath>/scripts/contrl/toolbar/file-tab.html'

You can also gray out the **Download** function through User Permission .

3.8 Print

Foxit PDF SDK for Web implements the Print function. At some cases you may want to disable it. The following shows a way of disabling print.

In the ..\scripts\control\toolbar\file-tab.html

*Find and delete <li data-action="print" data-i18n="[html]PCLng.ToolBar.Print" class="disabled">Print
*

In the ..\scripts\control\toolbar\file,

Replace 'template: require$$0,' with templateUrl:'<IndexPath>/scripts/contrl/toolbar/file-tab.html'

You can also gray out the **Print** function through User Permission .

4 User Settings

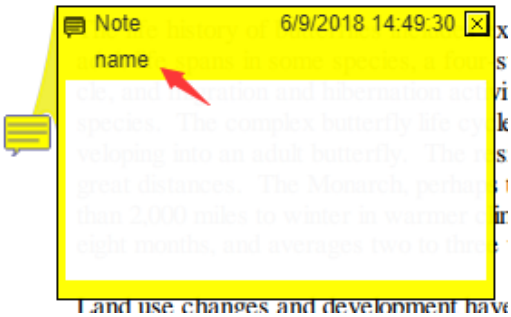
If you have your own user system, you can programmatically configure user settings for specific users or for all users by the user-related APIs in SDK.

4.1 Set User Name

//Set user name after WebPDF.ready

WebPDF.AccountInstance.setUserAccount('name');

The set name would appear as an author of an annotation as shown below:



4.2 Set User Config

Programmatically set the default attributes for annotations. For more information about configuration properties, refer to WebPDF.Config.UserData in the API References.

//an example of setting default attributes for highlight.

```
WebPDF. AccountInstance.setUserConfig({
    annot: {
    highLight: {
    clr: ‘#aaaaaa’,
    opacity: 0.5
    }}})
```

4.3 Set User Permission

The user permission is different from the document permission, which is just for display presentation, not modifying the document. For more permission options, refer to API References.

// assign users all permissions

WebPDF.ViewerInstance.setUserPermission(-1).

//assign users read-only permission, all editing features, download and print button on the toolbar will be disabled

WebPDF.ViewerInstance.setUserPermission(1)

OR

WebPDF.ViewerInstance.setUserPermission(WebPDF.UserPermission.DOCUMENT_VIEW)

// assign users read and modify annotation permissions, the user's permission value settings are cumulative, the print button on the toolbar will be disabled.

WebPDF.ViewerInstance.setUserPermission(17)

OR

WebPDF.ViewerInstance.setUserPermission(1 + 16)

OR

WebPDF.ViewerInstance.setUserPermission(WebPDF.UserPermission.DOCUMENT_VIEW + WebPDF.UserPermission.COMMENT)

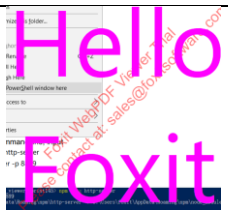
4.4 Set User Watermark

The user watermark is different from the document watermark, which is just for displaying presentation, not modifying the document. The settings will be saved in the memory and applied when a document is re-loaded.

//add Hello Foxit on the PDF page	*Output*

```
watermarkInfo = {
    "add":true,
    "content":"Hello Foxit",
    "rotation": 0,
    "scale": 4.0,
    "fontFamily":"'Segoe UI', sans-serif",
    "fontSize" : 50,
    "color": "0xff00ff",
    "isDynamic": false,
    };
 WebPDF.AccountInstance.setWatermarkInfo(watermarkInfo)
```

Output



// add dynamic user information on the PDF page

```
watermarkInfo = {
    "add":true,
    //"content":"Hello Foxit",
    "rotation": 0,
    "scale": 4.0,
    "fontFamily":"'Segoe UI', sans-serif",
    "fontSize" : 50,
    "color": "0xff00ff",
    "isDynamic": true,
    "userName": true,
    "ip":"127.0.0.1",
    "openTime": true
    };

WebPDF.AccountInstance.setWatermarkInfo(watermarkInfo)
```

5 Pages

5.1 Insert Pages from stream

You can programmatically insert pages into the current document from another document stream. Once the pages were successfully inserted, the modified new document is opened. For the supported parameters to insert pages, refer to WebPDF. Viewerinstance in the API Reference.

// Extract 7-9 pages from a PDF document with a password protection and insert into current document

```
var url = 'http://IPadddress:8080/samplepath/FoxitSample.pdf';
var xhr = new XMLHttpRequest();
```

```
xhr.open('GET', url, true);
xhr.responseType = 'arraybuffer';
xhr.send();
xhr.onload = function(e)
{
var responseArray = new Uint8Array(this.response);
var fileName = "FoxitSample_insert.pdf";
WebPDF.ViewerInstance.insertPagesFromSteam(responseArray, "password", "7-9",-
1,fileName);
}
```

// Insert a page before page 2 of current document with a source file name and reload the file

```
var url = 'http://IPadddress:8080/samplepath/FoxitSample.pdf';
var xhr = new XMLHttpRequest();
xhr.open('GET', url, true);
xhr.responseType = 'arraybuffer';
xhr.send();
xhr.onload = function(e)
{
var responseArray = new Uint8Array(this.response);
var fileName = "FoxitSample_insert.pdf";
WebPDF.ViewerInstance.insertPagesFromSteam(responseArray, "", "1",2,fileName);
}
```

5.2 Flatten Pages

Flatten annotations and form fields by the specified range and return the new file stream after the file is processed.

//flatten page 6-7 of the current document including new changes and save to it to database

```
WebPDF.ViewerInstance.flattenPages(null,'6-7',0).then(function (buffer) {
    //call opening file API or save buffer to database
});
```

5.3 Split Pages

You can programmatically split pages by page ranges or specific page. Below are examples.

Extract each page of current document as a PDF.
WebPDF.ViewerInstance.splitPages("-1")

Extract specified pages as one PDF. The example below will output 1 PDF containing 4 pages.

WebPDF.ViewerInstance.splitPages ("1,3,5,7")

Extract each specified page or page ranges as a PDF. The example below will output 3 separate PDFs.

WebPDF.ViewerInstance.splitPages("1-3;8-10;1,3,5,7")

Support

Foxit Support:

http://www.foxitsoftware.com/support/

Sales Contact:

Phone: 1-866-680-3668

Email: sales@foxitsoftware.com

Support & General:

Phone: 1-866-MYFOXIT or 1-866-693-6948

Email: support@foxitsoftware.com