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, Apache or the HTTP server in Node.js platform and do the usual configuration before using Foxit PDF SDK for Web.

1.4 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 15-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.

1.5 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

Foxit PDF SDK for Web provides multi-level and rich features and interfaces which include the API without UI, simple UI rendering, UI customization, as well as application of complex UI interactions. This following sections introduces the package structure, integration, demos, and how to run the demos.

2.1 Understanding the Package Structure

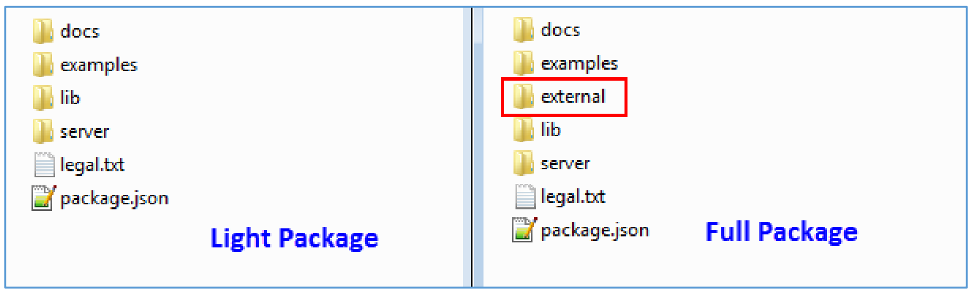
2.1.1 Package Introduction

Foxit PDF SDK for Web provides two packages as follows:

- Light package: FoxitPDFSDKForWeb_7_1_0.zip (without font resources)
- Full package: FoxitPDFSDKForWeb_7_1_0_Full.zip (with font resources)

If you already have the font resources or only want to use online fonts, you can choose the light package. If you don't want to make any change on the font library and don't care the package size, the full package is your first choice.

The following image shows what folders or files are included in the two packages. They have virtually same folders except the "**external**" folder in the full package.



The package contains:

docs:	A folder containing API references, developer guide.
examples:	A folder containing demos for Foxit PDF SDK for Web.
lib:	A folder containing the source packages for SDK core libraries.
external	A folder containing the font resources (only for full package).

server	A folder containing http-server and the Node.js scripts for snapshot server.
legal.txt:	Legal and copyright information.
package.json:	Project description file.

In the "lib" folder, it contains the files as follows:

jr-engine	A folder containing front-end rendering engine.
locales	A folder containing the internationalized entries data. The entries for different languages is placed in different directories by name.
PDFViewCtrl	A folder containing the plugins for PDFViewCtrl.
stamps	Stamps templates.
uix-addons	A folder containing the plugins for UIExtension.
PDFViewCtrl.css	The CSS of the simple UI for SDK library.
PDFViewCtrl.full.js	The SDK library with simple UI.
PDFViewCtrl.js	The PDFViewCtrl library without third-party libraries.
PDFViewCtrl.vendor.js	The third-party libraries used by PDFViewCtrl (See the lists later).
PDFViewCtrl.polyfills.js	The script files that PDFViewCtrl relies on, used to be compliant with different browsers.
UIExtension.css	The CSS of the UI for all the features.
UIExtension.full.js	The full-featured SDK library.
UIExtension.js	The UIExtension library without third-party libraries
UIExtension.vendor.js	The third-party libraries used by UIExtension (See the lists later)
UIExtension.polyfills.js	The script files that UIExtension relies on, used to be compliant with different browsers.
WebPDFJRWorker.js	The script files running in the Web Worker, which are used for calling the front-end rendering engine.

2.1.2 Package.json

Foxit PDF SDK for Web provides a package.json file to help developers to quickly experience the SDK product, and make it easier to be integrated into their project. The content is as follows:

```json
{
    "name": "foxit-pdf-sdk-for-web",
    "version": "7.1.0",
    "description": "Foxit pdf sdk for web.",
    "author": "Foxit Software Inc.",
    "main": "./lib/PDFViewCtrl.full.js",
    "scripts": {
        "start": "concurrently --kill-others \"npm run start-http-server\" \"npm run start-
snapshot-server\"",
        "start-snapshot-server": "node ./server/snapshot/src/index -p 3002",
        "start-http-server": "node ./server/index"
    },
    "devDependencies": {
        "boxen": "^4.1.0",
        "chalk": "^2.4.1",
        "concurrently": "^4.1.0",
        "http-proxy-middleware": "^0.19.1",
        "koa": "^2.7.0",
        "koa-body": "^4.0.4",
        "koa-body-parser": "^1.1.2",
        "koa-router": "^7.4.0",
        "koa2-connect": "^1.0.2",
        "lru-cache": "^4.1.3",
        "raw-body": "^2.3.3",
        "require-dir": "^1.0.0",
        "serve-handler": "^6.0.2"
    },
    "serve": {
        "port": 8080,
        "public": "/",
        "proxy": {
            "target": "http://127.0.0.1:3002",
            "changeOrigin": true
        }
    }
}
```

2.1.3 The third-party libraries used in Foxit PDF SDK for Web

Foxit PDF SDK for Web provides two formats of JS packages: the JS package with the third-party libraries, and the JS package without the third-party libraries. If your project has already used the same third-party libraries, you don't need to add it for avoiding duplication.

For *PDFViewCtrl.full.js* package:

PDFViewCtrl.full.js	The SDK library with simple UI.
PDFViewCtrl.polyfills.js	The script files that PDFViewCtrl relies on, used to be compliant with different browsers.
PDFViewCtrl.vendor.js	The third-party libraries used by PDFViewCtrl (See the lists later).
PDFViewCtrl.js	The PDFViewCtrl library without third-party libraries.

So that, **(PDFViewCtrl.polyfills.js** + **PDFViewCtrl.vendor.js** + **PDFViewCtrl.js)** = **PDFViewCtrl.full.js**.

The following two forms are equivalent:

```
1)  <script src="../FoxitPDFSDKForWeb/lib/PDFViewCtrl.full.js"></script>
```

```
2)  <script src="../FoxitPDFSDKForWeb/lib/ PDFViewCtrl.polyfills.js"></script>
    <script src="../FoxitPDFSDKForWeb/lib/PDFViewCtrl.vendor.js"></script>
    <script src="../FoxitPDFSDKForWeb/lib/PDFViewCtrl.js"></script>
```

The **PDFViewCtrl.vendor.js** contains the following third-party libraries:

jquery
i18next
i18next-chained-backend
i18next-localstorage-backend
i18next-xhr-backend
jquery-contextmenu
dialog-polyfill
hammerjs
eventemitter3

For *UIExtension.full.js* package:

UIExtension.full.js	The full-featured SDK library
UIExtension.polyfills.js	The script files that UIExtension relies on, used to be compliant with different browsers.
UIExtension.js	The UIExtension library without third-party libraries
UIExtension.vendor.js	The third-party libraries used by UIExtension (See the lists later)

So that, **(UIExtension.polyfills.js** + **UIExtension.vendor.js** + **UIExtension.js) = UIExtension.full.js**.

The following two forms are equivalent:

```
1)  <script src="../FoxitPDFSDKForWeb/lib/UIExtension.full.js"></script>

2)  <script src="../FoxitPDFSDKForWeb/lib/UIExtension.polyfills.js"></script>
    <script src="../FoxitPDFSDKForWeb/lib/UIExtension.vendor.js"></script>
    <script src="../FoxitPDFSDKForWeb/lib/UIExtension.js"></script>
```

The **UIExtension.vendor.js** contains the following third-party libraries:

jquery
i18next
i18next-chained-backend
i18next-localstorage-backend
i18next-xhr-backend
dialog-polyfill
hammerjs

eventemitter3

spectrum-colorpicker

file-saver

2.2 Integration

This section will introduce how to integrate Foxit PDF SDK for Web to your project.

2.2.1 Integrate as a Global Variable

You can integrate the Foxit PDF SDK for Web to your project as a global variable:

```
<script src="./lib/PDFViewCtrl.full.js"></script>
var PDFViewer = PDFViewCtrl.PDFViewer;
var pdfViewer = new PDFViewer(…)
```

2.2.2 Integrate as module

You can integrate the Foxit PDF SDK for Web to your project by module.

2.2.2.1 ES Modules (ESM)

Configure babelrc:

```
{
  "presets": ["env", "stage-0"]
}
```

Configure webpack loader:

```
     rules: [
      {
        test: /\.js$/,
        loader: 'babel-loader'
      },
    ],
```

Import the library:

```
import {PDFViewer} from './FoxitPDFSDKForWeb/lib/PDFViewCtrl.full';
var pdfViewer = new PDFViewer(…)
```

2.2.2.2 CommonJS

Import the library:

```
var PDFViewCtrl = require('./FoxitPDFSDKForWeb/lib/PDFViewCtrl.full');
var pdfviewer = new PDFViewCtrl.PDFViewer(...);
```

The import way of CommonJS is simple, so that it doesn't need additional configuration of webpack.

2.2.2.3 AMD

```
define('your-module-name', ['./FoxitPDFSDKForWeb/lib/PDFViewCtrl.full'],
function(PDFViewerModule) {
    var pdfviewer = new PDFViewCtrl.PDFViewer(...);
});
```

webpack configuration:

```
module.exports = {
    context: __dirname + 'point to the dir your app code is in',

    entry: 'your entry file path',
    output: {
        path: 'The path for the output built files',
        // ...
    },
    // It is used to support the older versions of jquery that you might not need.
    amd: {
        jQuery: true
    },
    module: {
        // loaders
    },
    resolve: {
        root: [],
        alias: {
        }
    }
};
```

2.2.2.4 Convenient dependency import

The above methods all use the relative path when integrating PDFViewCtrl module, but it is not easy to use the relative path to import multiple files. In this case, convenient dependency import might be a good choice. Please follow the steps below.

1) Add npm package declaration "package.json" for the unzipped package of Foxit PDF SDK for Web

lib/package.json:

```
{
  "name": "foxitwebsdk",
  "version": "7.0.0",
  "description": "",
  "main": "PDFViewCtrl.full.js", // Point to the Foxit PDF SDK for Web library that you need
to reference.
  "scripts": {
  },
}
```

2) Add the Foxit PDF SDK for Web dependencies in the package.json file of your project

```
"dependencies": {
    "foxitweb": "file:./FoxitPDFSDKForWeb/lib"
  },
```

3) Run "npm install" to install the dependencies.

4) Import and use.

```
// EMS
import {PDFViewer} from 'foxitweb';
var pdfViewer = new PDFViewer(…);

// or CommonJS
var PDFViewCtrl = require('foxitweb');
var pdfviewer = new PDFViewCtrl.PDFViewer(...);
```

2.3 Quickly run Foxit PDF SDK for Web Demos

To run Foxit PDF SDK for Web Demos, you should prepare a web server at first. In this guide, we will introduce the demos and then take the Nginx and Node.js servers as examples to show you how to quickly run Foxit PDF SDK for Web demos.

2.3.1 Introduction to the Demos

In the "examples" folder, it provides multiple demos as references for users. After starting the http server, you can access and experience the demos in the browser by typing the corresponding address.

- **A full-featured PDF viewer demo**

It is a ready-to-go demo that you can integrate it into your project with all of the featured provided by Foxit PDF SDK for Web. This demo uses the full-featured package "UIExtension.full.js" (in the "lib" folder) including the PDF view and document parsing. You can get the source code of this demo from "examples/UIExtension/advanced_webViewer/index.html".

In the browser, quickly access the demo at
http://localhost:{port}/examples/UIExtension/advanced_webViewer/index.html

- **A simple UI demo**

It is a simple UI demo that demonstrates how to call Foxit PDF SDK for Web API to load a PDF document, and zoom in/out the document. This demo uses the "PDFViewCtrl.full.js" package in the "lib" folder. You can get the source code of this demo from "examples/PDFViewCtrl/basic_webViewer/index.html".

In the browser, quickly access the demo at
http://localhost:{port}/examples/PDFViewCtrl/basic_webViewer/index.html

- **Offline demo**

This demo is a sample that demonstrates how to register the "service-worker.js" found in the "examples/PDFViewCtrl/service-worker" folder to better cache the core dependency files "gsdk.js" and font files in a browser supported by the service worker, in order to speed up the file secondary opening or use the offline mode. You can get the source code of this demo from "examples/PDFViewCtrl/service-worker/cache.html ".

In the browser, quickly access the demo at http://localhost:{port}/examples/PDFViewCtrl/service-worker/cache.html

- **Inline DIV application demo**

This demo renders the simple UI of Foxit PDF SDK for Web to a div container with a specified size. You can get the source code of this demo from "examples/PDFViewCtrl/div/index.html".

In the browser, quickly access the demo at http://localhost:{port}/examples/PDFViewCtrl/div/index.html .

2.3.2 Nginx working sample

Using Windows as an example, assume that Nginx was installed on your system already. When you have Nginx server running, you can directly modify the 'nginx.conf' in the conf directory, here we directly write a configuration file to make Foxit PDF SDK for Web Demo run. Please follows the steps below:

1) Download FoxitPDFSDKForWeb_7_1_0.zip.
2) Unzip it to a new folder like "FoxitPDFSDKForWeb" in 'D:/' directory for example.
3) Create a Nginx configuration file in a location. For example, create a new 'webpdf.conf' file inside 'D:/FoxitPDFSDKForWeb'.
4) Set up a virtual server and configure a location. The following configuration is an example. The 'D:/FoxitPDFSDKForWeb/' is the path to the SDK.

```
server {
    listen 8080;
    server_name 127.0.0.1;

    location / {
        alias "D:/FoxitPDFSDKForWeb/";
        charset utf8;
        index index.html;
    }
}
```

5) Locate to the installed path of Nginx, find the 'nginx.conf' in the conf directory, use **include** directive to reference the contents of the new configuration file.

```
include D:/FoxitPDFSDKForWeb/webpdf.conf;
```

6) For changes to the configuration file to take effect, you need to restart the nginx server or use 'nginx -s reload' to upgrade the configuration without interrupting the processing of current requests.

7) Access the demos in the browser.
   For the full-featured PDF viewer demo, please access the page at http://127.0.0.1:8080/examples/UIExtension/advanced_webViewer/index.html.
   For the simple UI demo, please access the page at http://127.0.0.1:8080/examples/PDFViewCtrl/basic_webViewer/index.html.

Note*: You can run the demo according to the above configuration, but at that time the snapshot feature cannot work correctly. The snapshot cannot be cached to clipboard, so that you cannot paste it to the location as you wish. Please follow the steps below to build the snapshot server:*

a) Install node.js 9.0 or higher, if it is already installed, skip it.
b) In a command prompt, navigate to the root directory ("D:/FoxitPDFSDKForWeb"), type "npm install" to install the required dependencies, and then type "npm run start-snapshot-server" to start the snapshot server (the default port is 3002).

   Note*: If you want to specify the port for snapshot server as desired, you can change it in the "****server/snapshot/package.json****" file of Foxit PDF SDK for Web package. To find the default port 3002 as below, and change it as you wish:*

```
"start": "node src/index -p 3002"
```

c) Configure Nginx reverse proxy in the 'webpdf.conf' file under 'D:/FoxitPDFSDKForWeb' folder.

```
server {
    listen 8080;
    server_name 127.0.0.1;

    location / {
        alias "D:/FoxitPDFSDKForWeb/";
        charset utf8;
        index index.html;
    }

    location ~ ^/snapshot/(.+)$ {
        proxy_pass http://127.0.0.1:3002/snapshot/$1$is_args$args;
        proxy_redirect off;

        proxy_request_buffering on;

        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    }
}
```

d) Restart Nginx server, and refresh your browser, the snapshot feature can work correctly.

2.3.3 Node.js working sample

Assume that Node.js 9.0 or higher is available on your system already. Then follow the steps below to run the Foxit PDF SDK for Web demos:

1) Download FoxitPDFSDKForWeb.zip.
2) Unzip it to a new folder like "FoxitPDFSDKForWeb" in 'D:/' directory for example.
3) In the command prompt, navigate to the above unzipped folder ("D:/FoxitPDFSDKForWeb"), type "npm install" to install the required dependencies, and then type "npm start" to start http-server.
4) Access the demos in the browser.
   For the full-featured PDF viewer demo, please access the page at
   http://127.0.0.1:8080/examples/UIExtension/advanced_webViewer/index.html.
   For the simple UI demo, please access the page at
   http://127.0.0.1:8080/examples/PDFViewCtrl/basic_webViewer/index.html.

***Note**: Using this method, you do not need to configure the proxy, the snapshot feature can be used normally. If you want to specify the ports for http-server and snapshot server as desired, you can change the two ports in the "**package.json**" file of Foxit PDF SDK for Web package.*

To change the port for http-server, find the default port 8080 as below, and change it as you wish:

```
"serve": {
    "port": 8080,
    "public": "/",
    "proxy": {
        "target": "http://127.0.0.1:3002",
        "changeOrigin": true
    }
}
```

To change the port for snapshot server, find the default port 3002 as below, and change it as you wish: (there are two places that need to be changed.)

```
"start-snapshot-server": "node ./server/snapshot/src/index -p 3002",
```

```
"serve": {
    "port": 8080,
    "public": "/",
    "proxy": {
        "target": "http://127.0.0.1:3002",
        "changeOrigin": true
    }
}
```

3 Building a new web project

This section will help you to quickly get started with using Foxit PDF SDK for Web to build a simple PDF viewer and a full-featured PDF viewer with step-by-step instructions provided.

3.1 Preparations

3.1.1 Create a new web project

a) Create a new directory as a project folder, such as "D:/test_web".

b) Copy the "lib", "server" folders and the "package.json" file from Foxit PDF SDK for Web package to "D:/test_web".

c) Copy a PDF file (for example, the demo guide in the "docs" folder) to "D:/test_web".

d) Create an html file (index.html) in the "D:/test_web" folder. Then the directory structure is:

```
test_web

    +-- lib          (copy from the Foxit PDF SDK for Web package)
    +-- server       (copy from the Foxit PDF SDK for Web package)
    +-- package.json (copy from the Foxit PDF SDK for Web package)
    +-- index.html
```

The whole content of the index.html is:

```
<html>
<head>
    <meta charset="utf-8">
    <style>
        .fv__ui-tab-nav li span {
            color: #636363;
        }
        .flex-row {
            display: flex;
            flex-direction: row;
        }
    </style>
    <!-- ignore other unimportant code -->
</head>
<body>
</body>
</html>
```

3.1.2 Start Http-Server

You can refer to section 2.3.2 "Nginx working sample" or section 2.3.3 "Node.js working sample" to start the http-server. In this project, we use the default port 8080.

Then, access http://127.0.0.1:8080/index.html in the browser, and you will see a blank page, which indicates that the preparations have been completed.

3.2 Integrate the simple demo using PDFViewCtrl

This section will describe how to integrate the simple demo using PDFViewCtrl based on the above create project. Just follow the steps below:

a) Add a style (/lib/PDFViewCtrl.css) to the <head> tag of the HTML page.

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

b) Import the "PDFViewCtrl.full.js" library found in the "lib" folder:

```
<script src="./lib/PDFViewCtrl.full.js"></script>
```

c) In the HTML <body> tag, add the <div> elements as the web viewer container:

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

d) Initialize the Foxit PDF SDK for Web:

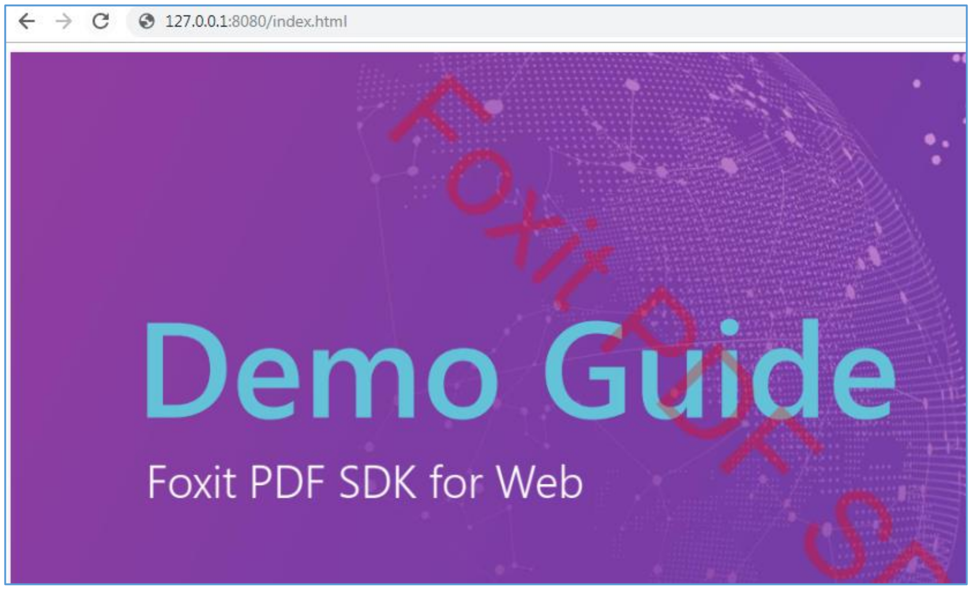
```
<script>
     var licenseSN = "Your license SN";
     var licenseKey = "Your license Key";
</script>
<script>
     var PDFViewer = PDFViewCtrl.PDFViewer;
     var pdfViewer = new PDFViewer({
         libPath: './lib', // the library path of Web SDK.
         jr: {
             licenseSN: licenseSN,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

***Note**: The trial values of **licenseSN** and **licenseKey** can be found in the "example/license-key.js" file of Foxit PDF SDK for Web package.*

e) Open a PDF document:

```
// modify the file path as your need.
fetch('/FoxitPDFSDKforWeb_DemoGuide.pdf').then(function(response) {
    response.arrayBuffer().then(function(buffer) {
        pdfViewer.openPDFByFile(buffer);
    })
})
```

f) The above steps are the key points of integrating the simple demo to your created project using PDFViewCtrl. After finishing it, refresh your browser (http://127.0.0.1:8080/index.html), and then you can see the demo guide as follows:



Now, it is a simple web PDF viewer, you can zoom in/out the PDF document by right-clicking anywhere on the page to select the zoom in or zoom out options.

The whole content of the **index.html** is:

```
<html>
<head>
    <meta charset="utf-8">
    <link rel="stylesheet" type="text/css" href="./lib/PDFViewCtrl.css">

    <!-- You can delete the following style because it doesn't work in this project -->
    <style>
        .fv__ui-tab-nav li span {
            color: #636363;
        }
        .flex-row {
            display: flex;
            flex-direction: row;
        }
    </style>
    <!-- ignore other unimportant code -->
</head>
<body>
    <div id="pdf-viewer"></div>
    <script src="./lib/PDFViewCtrl.full.js"></script>
    <script>
        var licenseSN = "Your license SN";
        var licenseKey = "Your license Key";
    </script>
    <script>
        var PDFViewer = PDFViewCtrl.PDFViewer;
        var 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")

        // modify the file path as your need.
        fetch('/FoxitPDFSDKforWeb_DemoGuide.pdf').then(function (response) {
            response.arrayBuffer().then(function (buffer) {
                pdfViewer.openPDFByFile(buffer);
            })
        })

    </script>
</body>
</html>
```

3.3 Integrate the full-featured PDF viewer using UIExtension

The previous section integrates the simple demo using PDFViewCtrl, which is just a simple web PDF viewer. In this section, we will show you how to integrate the full-featured PDF viewer using UIExtension based on the created project in section 3.3.1. Just follow the steps below:

a) Add a style (/lib/UIExtension.css) to the <head> tag of the HTML page.

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

b) Import the "UIExtension.full.js" library found in the "lib" folder:

```
<script src="./lib/UIExtension.full.js"></script>
```

c) In the HTML <body> tag, add the <div> elements as the web viewer container:

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

d) Initialize the Foxit PDF SDK for Web:

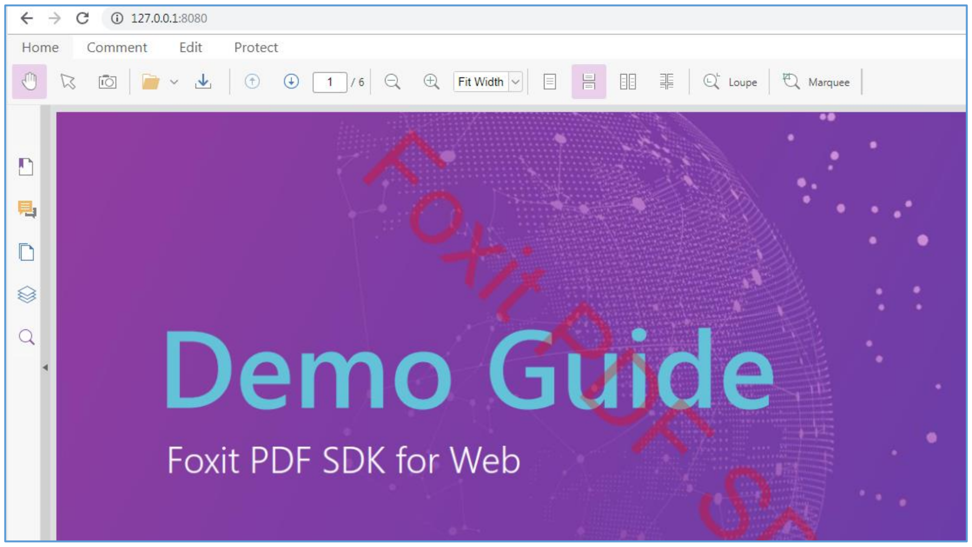
```
<script>
     var licenseSN = "Your license SN";
     var licenseKey = "Your license Key";
</script>
<script>
    var pdfui = new UIExtension.PDFUI({
        viewerOptions: {
            libPath: './lib', // the library path of web sdk.
            jr: {
                licenseSN: licenseSN,
   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

***Note**: The trial values of **licenseSN** and **licenseKey** can be found in the "example/license-key.js" file of Foxit PDF SDK for Web package.*

e) Open a PDF document:

```
// modify the file path as your need.
fetch('/FoxitPDFSDKforWeb_DemoGuide.pdf').then(function(response) {
    response.arrayBuffer().then(function(buffer) {
        pdfui.openPDFByFile(buffer);
    })
})
```

f) The above steps are the key points of integrating the full-featured PDF viewer to your created project using UIExtension. After finishing it, refresh your browser (http://127.0.0.1:8080/index.html), and then you can see the demo guide as follows:



Now, it is a full-featured web PDF viewer, you can view/edit/comment/protect the PDF document as desired.

The whole content of the **index.html** is:

```
<html>
<head>
    <meta charset="utf-8">
    <link rel="stylesheet" type="text/css" href="./lib/UIExtension.css">
    <style>
        .fv__ui-tab-nav li span {
            color: #636363;
        }
        .flex-row {
            display: flex;
            flex-direction: row;
        }
    </style>
    <!-- ignore other unimportant code -->
</head>
<body>
    <div id="pdf-ui"></div>
    <script src="./lib/UIExtension.full.js"></script>
    <script>
        var licenseSN = "Your license SN";
        var licenseKey = "Your license Key";
    </script>
    <script>
        var 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").
        });

        // modify the file path as your need.
        fetch('/FoxitPDFSDKforWeb_DemoGuide.pdf').then(function (response) {
            response.arrayBuffer().then(function (buffer) {
                pdfui.openPDFByFile(buffer);
            })
        })

    </script>
</body>
</html>
```

3.4 Integrate the add-ons of UIExtension

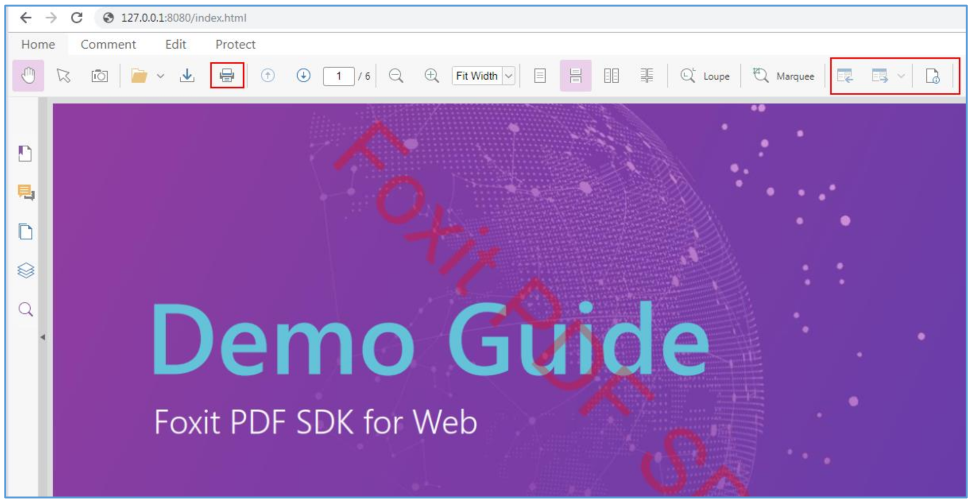
Foxit PDF SDK for Web provides some extension features for UIExtension, such as file property, multi-media, password protection, redaction, PDF path objects, print, text object, importing/exporting form, and full screen. To integrate these add-ons, you should add the code snippets below:

```
addons: [
    './lib/uix-addons/file-property/', // should use the relative path.
    './lib/uix-addons/multi-media/',
    './lib/uix-addons/password-protect/',
    './lib/uix-addons/redaction/',
    './lib/uix-addons/path-objects/',
    './lib/uix-addons/print/',
    './lib/uix-addons/text-object',
    './lib/uix-addons/import-form',
    './lib/uix-addons/export-form',
    './lib/uix-addons/full-screen'
    ]
});
```

Based on the full-featured PDF viewer project in section 3.3, update the index.html as follows:

```
<html>
<head>
    <meta charset="utf-8">
    <link rel="stylesheet" type="text/css" href="./lib/UIExtension.css">
    <style>
        .fv__ui-tab-nav li span {
            color: #636363;
        }
        .flex-row {
            display: flex;
            flex-direction: row;
        }
    </style>
    <!-- ignore other unimportant code -->
</head>
<body>
    <div id="pdf-ui"></div>
    <script src="./lib/UIExtension.full.js"></script>
    <script>
        var licenseSN = "Your license SN";
        var licenseKey = "Your license Key";
    </script>
    <script>
        var 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").

            addons: [
                './lib/uix-addons/file-property/',  // should use the relative path.
                './lib/uix-addons/multi-media/',
                './lib/uix-addons/password-protect/',
                './lib/uix-addons/redaction/',
                './lib/uix-addons/path-objects/',
                './lib/uix-addons/print/',
                './lib/uix-addons/text-object',
                './lib/uix-addons/import-form',
                './lib/uix-addons/export-form',
                './lib/uix-addons/full-screen'
            ]
        });

        // modify the file path as your need.
        fetch('/FoxitPDFSDKforWeb_DemoGuide.pdf').then(function (response) {
            response.arrayBuffer().then(function (buffer) {
                pdfui.openPDFByFile(buffer);
            })
        })

    </script>
</body>
</html>
```

Refresh your browser (http://127.0.0.1:8080/index.html), and then you can see a web PDF viewer with the extension features. For example, for the Home tab, the file property, print, and import/export form features have been integrated as follows:



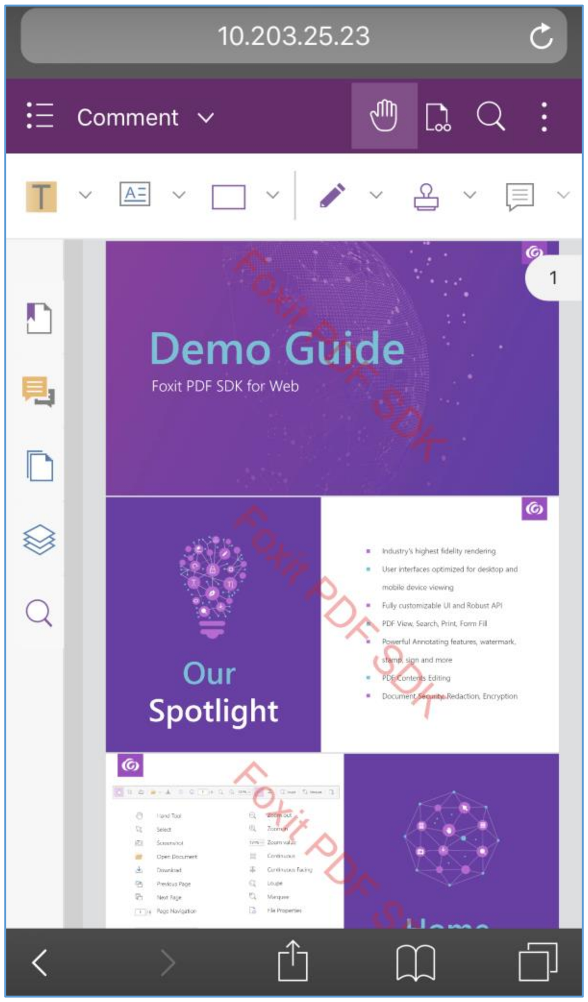
3.5 Make the project adaptive to the device

Foxit PDF SDK for Web can allow the project to be adaptive to the device (desktop or mobile), which means if you access the project in your desktop browser, it will be present using the desktop UI layout, and if you access the project in your mobile browser, it will be present using the mobile UI layout. To make the project adaptive to the device, based on the Integrate the add-ons of UIExtension in section 3.4, update the index.html as follows:

```
<html>
<head>
    <meta charset="utf-8">
    <link rel="stylesheet" type="text/css" href="./lib/UIExtension.css">

    <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">

    <meta name="viewport" content="width=device-width,initial-scale=1,minimum-scale=1,maximum-
scale=1,user-scalable=no">
    <!-- default(white), black, black-translucent -->
    <meta name="apple-mobile-web-app-status-bar-style" content="default" />
    <meta name="apple-mobile-web-app-capable" content="yes" />
    <meta name="apple-touch-fullscreen" content="yes" />
    <meta name="apple-mobile-web-app-title" content="Foxit PDF SDK for Web">
    <meta name="App-Config" content="fullscreen=yes,useHistoryState=no,transition=no">
    <meta name="format-detaction" content="telephone=no,email=no">
    <meta http-equiv="Cache-Control" content="no-siteapp" />
    <meta name="HandheldFriendly" content="true">
    <meta name="MobileOptimized" content="750">
    <meta name="screen-orientation" content="portrait">
    <meta name="x5-orientation" content="portrait">
    <meta name="full-screen" content="yes">
    <meta name="x5-fullscreen" content="true">
    <meta name="browsermode" content="application">
    <meta name="x5-page-mode" content="app">
    <meta name="msapplication-tap-highlight" content="no">
    <meta name="renderer" content="webkit">

    <!-- Add the adaptive.js -->
    <script src="./lib/adaptive.js"></script>

    <style>
        .fv__ui-tab-nav li span {
            color: #636363;
        }
        .flex-row {
            display: flex;
            flex-direction: row;
        }
    </style>
</head>

<body>
    <div id="pdf-ui"></div>
    <script src="./lib/UIExtension.full.js"></script>
    <script>
        var licenseSN = "Your license SN";
        var licenseKey = "Your license Key";
    </script>
    <script>

        var 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").
            appearance: UIExtension.appearances.adaptive,
            addons: [
                './lib/uix-addons/file-property/',  // should use the relative path.
                './lib/uix-addons/multi-media/',
                './lib/uix-addons/password-protect/',
                './lib/uix-addons/redaction/',
                './lib/uix-addons/path-objects/',
                './lib/uix-addons/print/',
                './lib/uix-addons/text-object',
                './lib/uix-addons/import-form',
                './lib/uix-addons/export-form',
                './lib/uix-addons/full-screen'
            ]
        });

        // modify the file path as your need.
        fetch('/FoxitPDFSDKforWeb_DemoGuide.pdf').then(function (response) {
            response.arrayBuffer().then(function (buffer) {
                pdfui.openPDFByFile(buffer);
            })
        })

    </script>
```

```
</body>
</html>
```

For example, assume the IP address of your computer is 10.203.25.23, so that access the project in mobile browser (http://10.203.25.23:8080/index.html), and then you can see a mobile PDF viewer as follows:



4 Customizing the UI

From version 7.0, Foxit PDF SDK for Web comes with built-in UI design including the feature modules UI, which are implemented using Foxit PDF SDK for Web and are shipped in the UIExtension.js. Furthermore, customizing UI is straightforward. Foxit PDF SDK for Web provides a rich set of APIs for developers to customize and style the appearance of your web viewer.

The user interface of UIExtension consists of two parts: template and fragments. Template is equivalent to the extension of HTML, the components in the template should be declared by tags. Template is used to customize the UI layout (css style, icon, text, and so on) without interactions. Fragments are a set of UI snippets, which can be used to customize the configuration items and interaction logic of the components in the template. Each snippets has an operation type "action" that specifies the action mode (append, prepend, before, after, ext, replace, insert, and remove, the default is ext.) of the snippets. Through these action modes, you can insert, delete, replace and modify the components in the template.

For all the built-in components used in the following examples, please refer to the section 4.4 "Understanding the built-in components".

This section provides two standard types of code for the examples: ES6 and ES5. If you want to run the examples in IE browser, please refer to the ES5 code.

4.1 Customize the UI layout using template

Template is mainly used to customize the UI layout of the components. Following will list some examples to demonstrate the usage of template. Please note that all the examples are based on the full-featured PDF viewer project in section 3.3.

4.1.1 Create a simple template

A simplest template is as follows:

For **ES6**:

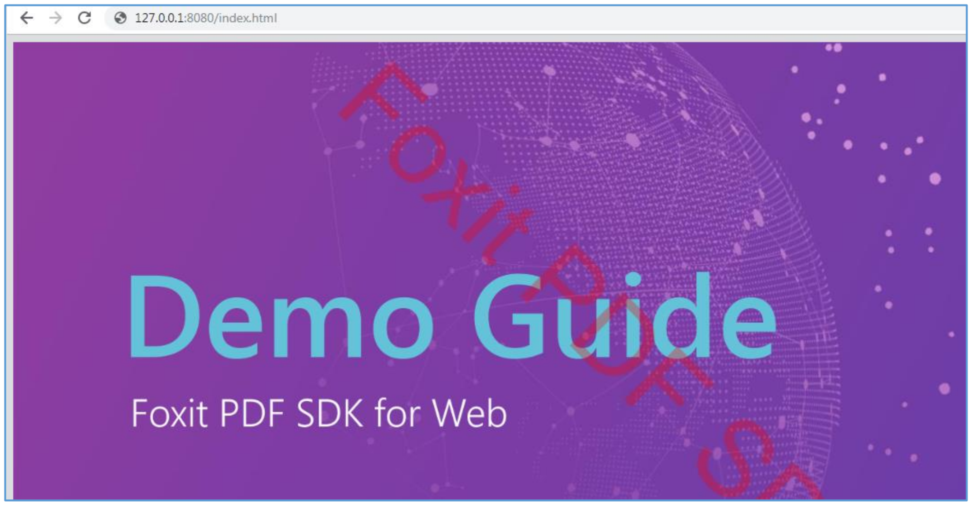
```
let pdfui = new UIExtension.PDFUI({
    viewerOptions: {
        libPath: './lib', // the library path of web sdk.
        jr: {
            licenseSN: licenseSN,
            licenseKey: licenseKey
        }
    },
    renderTo: '#pdf-ui', // the div (id="pdf-ui").
    template: `
        <webpdf>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

For **ES5**:

```
var 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").
    template: [
        '<webpdf>',
        '    <viewer></viewer>',
        '</webpdf>'
        ].join('')
});
```

- <webpdf> tag listens the document state (open or not) to control enabling or disabling the UI components. If you need these features, you can use other html tags or custom component tags to replace it.
- <viewer> tag is where the PDF contents are rendered. Each template must have a <viewer> tag. It can be placed anywhere you want, please refer to the examples in the following sections.

Refresh your browser (http://127.0.0.1:8080/index.html), and then you can see a simple PDF web viewer as follows:



4.1.2 Add a new toolbar button

Use <toolbar> tag to add a new toolbar button.

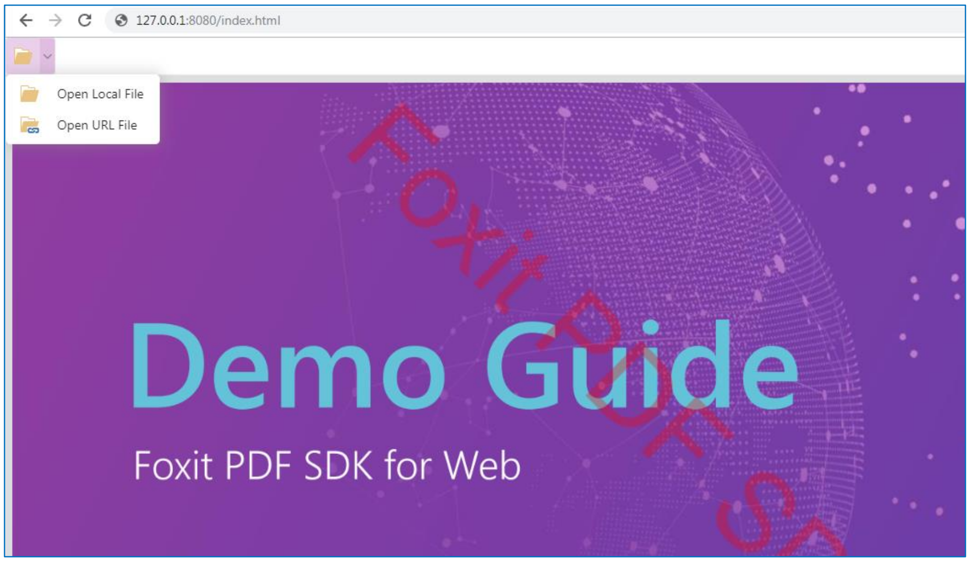
For **ES6**:

```
let 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").
    template: `
    <webpdf>
        <toolbar>
            <open-file-dropdown></open-file-dropdown>
        </toolbar>
        <viewer></viewer>
    </webpdf>
        `
});
```

For **ES5**:

```
var 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").
    template: [
    '<webpdf>',
    '    <toolbar>',
    '        <open-file-dropdown></open-file-dropdown>',
    '    </toolbar>',
    '    <viewer></viewer>',
    '</webpdf>'
        ].join('')
});
```

Refresh your browser (http://127.0.0.1:8080/index.html), and then you can see the new toolbar button as follows:



4.1.3 Add a new tab page

Use <tabs> and <tab> tags to add a new tab page.

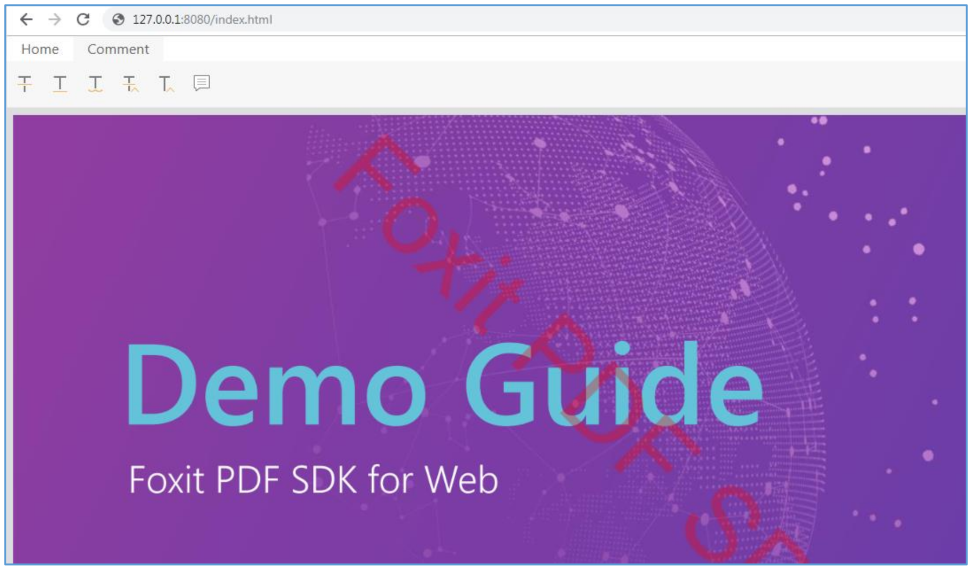
For **ES6**:

```
let 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").
    template: `
<webpdf>
    <toolbar>
        <tabs>
            <tab title="Home">
                <open-file-dropdown></open-file-dropdown>
            </tab>
            <tab title="Comment">
                <div class="flex-row">
                    <create-strikeout-button></create-strikeout-button>
                    <create-underline-button></create-underline-button>
                    <create-squiggly-button></create-squiggly-button>
                    <create-replace-button></create-replace-button>
                    <create-caret-button></create-caret-button>
                    <create-note-button></create-note-button>
                </div>
            </tab>
        </tabs>
    </toolbar>
    <viewer></viewer>
</webpdf>
`
});
```

For **ES5**:

```
var 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").
    template: [
'<webpdf>',
'    <toolbar>',
'        <tabs>',
'            <tab title="Home">',
'                <open-file-dropdown></open-file-dropdown>',
'            </tab>',
'            <tab title="Comment">',
'                <div class="flex-row">',
'                    <create-strikeout-button></create-strikeout-button>',
'                    <create-underline-button></create-underline-button>',
'                    <create-squiggly-button></create-squiggly-button>',
'                    <create-replace-button></create-replace-button>',
'                    <create-caret-button></create-caret-button>',
'                    <create-note-button></create-note-button>',
'                </div>',
'            </tab>',
'        </tabs>',
'    </toolbar>',
'    <viewer></viewer>',
'</webpdf>'
].join('')
});
```

Refresh your browser (http://127.0.0.1:8080/index.html), and then you can see the new tab page as follows:



4.1.4 Add a sidebar button

Use <sidebar> tag to add a sidebar button.

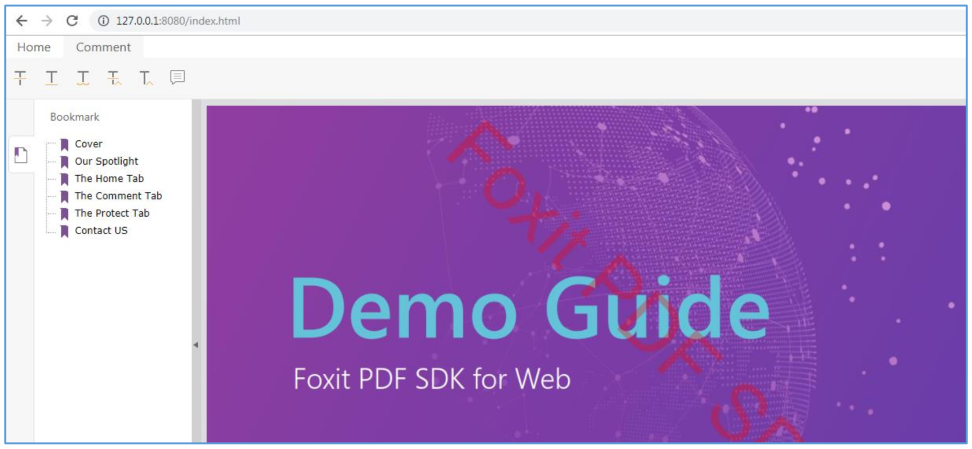
For **ES6**:

```
let 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").
    template: `
<webpdf>
    <toolbar>
        <tabs>
            <tab title="Home">
                <open-file-dropdown></open-file-dropdown>
            </tab>
            <tab title="Comment">
                <div class="flex-row">
                    <create-strikeout-button></create-strikeout-button>
                    <create-underline-button></create-underline-button>
                    <create-squiggly-button></create-squiggly-button>
                    <create-replace-button></create-replace-button>
                    <create-caret-button></create-caret-button>
                    <create-note-button></create-note-button>
                </div>
            </tab>
        </tabs>
    </toolbar>
    <div class="flex-row">
        <sidebar>
            <bookmark-sidebar-panel></bookmark-sidebar-panel>
        </sidebar>
        <viewer></viewer>
    </div>
</webpdf>
`
});
```

For **ES5**:

```
var 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").
    template: [
'<webpdf>',
'    <toolbar>',
'        <tabs>',
'            <tab title="Home">',
'                <open-file-dropdown></open-file-dropdown>',
'            </tab>',
'            <tab title="Comment">',
'                <div class="flex-row">',
'                    <create-strikeout-button></create-strikeout-button>',
'                    <create-underline-button></create-underline-button>',
'                    <create-squiggly-button></create-squiggly-button>',
'                    <create-replace-button></create-replace-button>',
'                    <create-caret-button></create-caret-button>',
'                    <create-note-button></create-note-button>',
'                </div>',
'            </tab>',
'        </tabs>',
'    </toolbar>',
'    <div class="flex-row">',
'        <sidebar>',
'            <bookmark-sidebar-panel></bookmark-sidebar-panel>',
'        </sidebar>',
'        <viewer></viewer>',
'    </div>',
'</webpdf>'
].join('')

});
```

Refresh your browser (http://127.0.0.1:8080/index.html), and then you can see the bookmark sidebar as follows:



4.1.5 Built-in layout template

The built-in layout template for Foxit PDF SDK for Web is shown below. You can modify this template directly to customize the UI layout as desired.

```
<webpdf>
    <toolbar name="toolbar" class="fv__ui-toolbar-scrollable">
        <tabs name="toolbar-tabs">
            <tab title="toolbar.tabs.home.title" name="home-tab">
                <group-list name="home-toolbar-group-list">
                    <group name="home-tab-group-hand" retain-count="3">
                        <hand-button></hand-button>
                        <selection-button></selection-button>
                        <snapshot-button></snapshot-button>
                    </group>
                    <group name="home-tab-group-io" retain-count="1" shrink-
title="toolbar.more.document.title">
                        <open-file-dropdown></open-file-dropdown>
                        <download-file-button></download-file-button>
                        <print:print-button></print:print-button>
                    </group>
                    <group name="home-tab-group-nav" retain-count="3">
                        <goto-prev-page-button></goto-prev-page-button>
                        <goto-next-page-button></goto-next-page-button>
                        <goto-page-input></goto-page-input>
                    </group>
                    <group name="home-tab-group-zoom">
                        <zoom-out-button></zoom-out-button>
                        <zoom-in-button></zoom-in-button>
                        <editable-zoom-dropdown></editable-zoom-dropdown>
                    </group>
                    <group name="home-tab-group-page" retain-count="1">
                        <single-page-button></single-page-button>
                        <continuous-page-button></continuous-page-button>
                        <facing-page-button></facing-page-button>
                        <continuous-facing-page-button></continuous-facing-page-button>
                    </group>
                    <group name="home-tab-group-magnifier">
                        <loupe-tool-button></loupe-tool-button>
                    </group>
                    <group name="home-tab-group-marquee">
                        <marquee-tool-button></marquee-tool-button>
                    </group>
                    <group name="home-tab-group-form" retain-count="2">
                        <import-form-module:import-form-button></import-form-module:import-
form-button>

                        <export-form-module:export-form-dropdown></export-form-module:export-
form-dropdown>
                    </group>
                    <group name="file-property" @require-module="fpmodule">
                        <fpmodule:file-property-button></fpmodule:file-property-button>
                    </group>
                </group-list>
            </tab>
            <tab title="toolbar.tabs.comment.title" name="comment-tab">
                <group-list name="comment-toolbar-group-list">
                    <group name="comment-tab-group-hand" retain-count="3">
                        <hand-button></hand-button>
                        <selection-button></selection-button>
                        <zoom-dropdown></zoom-dropdown>
                    </group>
                    <group name="comment-tab-group-note" retain-count="3">
                        <create-note-button></create-note-button>
                    </group>
                    <group name="comment-tab-group-mark">
                        <create-text-highlight-button></create-text-highlight-button>
                        <create-strikeout-button></create-strikeout-button>
                        <create-underline-button></create-underline-button>
                        <create-squiggly-button></create-squiggly-button>
                        <create-replace-button></create-replace-button>
                        <create-caret-button></create-caret-button>
                    </group>
                    <group name="comment-tab-group-text">
                        <create-typewriter-button></create-typewriter-button>
                        <create-callout-button></create-callout-button>
                        <create-textbox-button></create-textbox-button>
                    </group>
                    <group name="comment-tab-group-drawing" retain-count="2">
                        <create-drawings-dropdown></create-drawings-dropdown>
                        <create-area-highlight-button></create-area-highlight-button>
                    </group>
                    <group name="comment-tab-group-pencil" retain-count="2">
                        <create-pencil-button></create-pencil-button>
                        <eraser-button></eraser-button>
                    </group>
                    <group name="comment-tab-group-stamp">
                        <stamp-dropdown></stamp-dropdown>
                    </group>
                    <group name="comment-tab-group-measurement">
                        <create-distance-button></create-distance-button>
                    </group>
                    <group name="comment-tab-group-media">
                        <create-attachment-button></create-attachment-button>
                        <create-image-button></create-image-button>
                        <create-link-button></create-link-button>
                        <multi-media:multi-media-button></multi-media:multi-media-button>
                    </group>
                    <group name="comment-tab-group-inksign" visible='false'></group>
                    <group name="comment-tab-group-other" visible='false'></group>
                </group-list>
            </tab>
            <tab title="toolbar.tabs.edit.title" name="edit-
tab" visible="true" @device="desktop">
                <group-list name="edit-toolbar-group-list">
                    <group name="edit-tab-group-hand" retain-count="3">
                        <hand-button></hand-button>
                        <selection-button></selection-button>
                        <zoom-dropdown></zoom-dropdown>
                    </group>
                    <group name="edit-tab-group-mode" retain-count="3">
                        <edit-pageobjects:edit-all-objects-button></edit-pageobjects:edit-all-
objects-button>
                        <add-image-button></add-image-button>
                        <edit-pageobjects:path-objects-dropdown></edit-pageobjects:path-
objects-dropdown>
                        <!--<edit-image-button></edit-image-button>-->
                        <edit-text-object:add-text-button></edit-text-object:add-text-button>
                    </group>
                    <group name="edit-tab-group-font" retain-count="5">
                        <edit-text-object:text-bold-style-button></edit-text-object:text-bold-
style-button>
                        <edit-text-object:text-italic-style-button></edit-text-object:text-
italic-style-button>
                        <edit-text-object:font-color-picker></edit-text-object:font-color-
picker>
                        <edit-text-object:font-style-dropdown></edit-text-object:font-style-
dropdown>
                    </group>
                    <group name="edit-tab-group-layer" visible="false"></group>
                    <group name="edit-tab-group-redact" visible="false"></group>
                </group-list>
            </tab>
            <tab title="toolbar.tabs.protect.title" name="protect-tab" visible="true">
                <group-list name="protect-toolbar-group-list">
                    <group name="protect-tab-group-hand" retain-count="4">
                        <hand-button></hand-button>
                        <selection-button></selection-button>
                        <zoom-dropdown></zoom-dropdown>
                    </group>
                    <group name="protect-tab-group-sign" retain-count="4">
                        <ink-sign-dropdown></ink-sign-dropdown>
                    </group>
                    <group name="password-protect-group" retain-count="2">
                        <password-protect:password-protect-button></password-protect:password-
protect-button>
                        <password-protect:remove-protect-button></password-protect:remove-
protect-button>
                    </group>
                    <group name="redaction">
                        <redaction:create-redactions-dropdown></redaction:create-redactions-
dropdown>
                        <redaction:apply-redactions-button></redaction:apply-redactions-
button>
                        <redaction:redaction-search-button></redaction:redaction-search-
button>
                    </group>
                </group-list>
            </tab>
        </tabs>
    </toolbar>
    <div class="fv__ui-body">
        <sidebar name="sidebar" @controller="sidebar:SidebarController">
            <bookmark-sidebar-panel></bookmark-sidebar-panel>
            <commentlist-sidebar-panel></commentlist-sidebar-panel>
            <thumbnail-sidebar-panel></thumbnail-sidebar-panel>
            <layer-sidebar-panel></layer-sidebar-panel>
            <search-sidebar-panel></search-sidebar-panel>
            <attachment-sidebar-panel></attachment-sidebar-panel>
        </sidebar>
        <distance:ruler-container name="pdf-viewer-container-with-ruler">
            <slot>
                <viewer @zoom-on-wheel @touch-to-scroll></viewer>
            </slot>
        </distance:ruler-container>
    </div>
    <template name="template-container">
        <create-stamp-dialog></create-stamp-dialog>
        <print:print-dialog></print:print-dialog>
        <loupe-tool-dialog></loupe-tool-dialog>
        <create-ink-sign-dialog></create-ink-sign-dialog>
        <measurement-popup></measurement-popup>
        <fpmodule:file-property-dialog></fpmodule:file-property-dialog>
        <redaction:redaction-page-dialog></redaction:redaction-page-dialog>
        <!-- contextmenus -->
        <page-contextmenu></page-contextmenu>
        <default-annot-contextmenu></default-annot-contextmenu>
        <markup-contextmenu></markup-contextmenu>
        <default-annot-contextmenu name="fv--caret-contextmenu"></default-annot-contextmenu>
        <textmarkup-contextmenu name="fv--highlight-contextmenu"></textmarkup-contextmenu>
        <textmarkup-contextmenu name="fv--strikeout-contextmenu"></textmarkup-contextmenu>
        <textmarkup-contextmenu name="fv--underline-contextmenu"></textmarkup-contextmenu>
        <textmarkup-contextmenu name="fv--squiggly-contextmenu"></textmarkup-contextmenu>
        <freetext-contextmenu name="fv--typewriter-contextmenu"></freetext-contextmenu>
        <freetext-contextmenu name="fv--callout-contextmenu"></freetext-contextmenu>
        <freetext-contextmenu name="fv--textbox-contextmenu"></freetext-contextmenu>
        <action-annot-contextmenu name="fv--image-contextmenu"></action-annot-contextmenu>
        <action-annot-contextmenu name="fv--link-contextmenu"></action-annot-contextmenu>
        <comment-card-contextmenu></comment-card-contextmenu>
        <fileattachment-contextmenu></fileattachment-contextmenu>
        <media-contextmenu></media-contextmenu>
        <redact-contextmenu></redact-contextmenu>
    </template>
</webpdf>
```

4.2 Customize the UI using fragments

Fragments are a set of UI snippets. It can be used to customize the configuration items and interaction logic of the components in the template.

4.2.1 Create a new drop-down menu item

The sample code below creates a new drop-down menu including two drop-down buttons, and append it to the end of the list of children of the group component with the name of "home-tab-group-hand".

To get the name of a target component (only for widget), you can right-click the component in the browser, choose "Inspect", and then find the value of the "component-name" attribute in the corresponding <a> tag. For a container component, for example "`target: 'home-tab-group-hand',`", you can right-click one of the subcomponents in the browser, choose "Inspect", and then find the value of the "component-name" attribute in the related <div> tag.

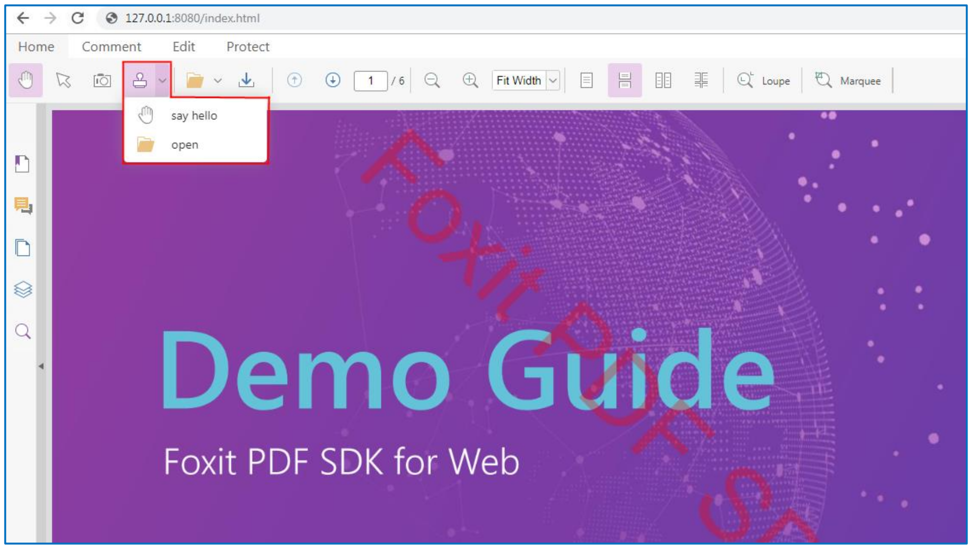
For **ES6**:

```
var 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").

    fragments: [{
        // Add a component to the end of the list of children of a specified target component.
        action: UIExtension.UIConsts.FRAGMENT_ACTION.APPEND,
        // Specify the name of the target component that the new components defined in the
above template will be appended to. All the target names of fragments are defined in the
layout template.
        target: 'home-tab-group-hand',
        // Define the properties of the added component, such as icon, text, and css style.
        template: `
        <dropdown icon-class="fv__icon-toolbar-stamp">
            <dropdown-button name="show-hello-button" icon-class="fv__icon-toolbar-hand">say
hello</dropdown-button>
            <dropdown-button name="select-pdf-file-button" accept=".pdf" file-selector icon-
class="fv__icon-toolbar-open">open</dropdown-button>
        </dropdown>
        `,
        // Define the interaction logic of the added component.
        config: [{
            // specify the component in the above template that the configuration will be 
applied to.
            // For example, the configuration will be applied to the component with the name 
of "show-hello-button".
            target: 'show-hello-button',
            callback: function () {
                alert('hello');
            }
        },
        {
            // The configuration will be applied to the component with the name of "select-
pdf-file-button" which is defined in the above template of fragments.
            target: 'select-pdf-file-button',
            // Extend Controller, and implement the handle function.
            callback: class extends UIExtension.Controller {
                // the lifecycle method: mounted. It will be called after appending the 
component into the DOM tree.
                mounted() {
                    // Get the component mounted by the current Controller instance
                    console.info(this.component, 'mounted');
                }
                handle(selectedFile) {
                    alert(selectedFile.name);
                }
            }
        }]
    }]
});
```

For **ES5**:

```
function CustomController() {
    UIExtension.Controller.apply(this, arguments);
}
CustomController.prototype = Object.create(UIExtension.Controller.prototype, {
    constructor: {
        value: CustomController
    }
});
Object.assign(CustomController.prototype, {
    mounted: function () {
        console.info(this.component, 'mounted');
    },
    handle: function (selectedFile) {
        alert(selectedFile.name);
    }
});

var 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").
    fragments: [{
        // Add a component to the end of the list of children of a specified target component.
        action: UIExtension.UIConsts.FRAGMENT_ACTION.APPEND,
        // Specify the name of the target component that the new components defined in the
above template will be appended to. All the target names of fragments are defined in the
layout template.
        target: 'home-tab-group-hand',
        // Define the properties of the added component, such as icon, text, and css style.
        template: [
            '<dropdown icon-class="fv__icon-toolbar-stamp">',
            '    <dropdown-button name="show-hello-button" icon-class="fv__icon-toolbar-
hand">say hello</dropdown-button>',
            '    <dropdown-button name="select-pdf-file-button" accept=".pdf" file-selector
icon-class="fv__icon-toolbar-open">open</dropdown-button>',
            '</dropdown>'
        ].join(''),
        // Define the interaction logic of the added component.
        config: [{
            // specify the component in the above template that the configuration will be
applied to.
            // For example, the configuration will be applied to the component with the name
of "show-hello-button".
            target: 'show-hello-button',
            callback: function () {
                alert('hello');
            }
        },
        {
            // The configuration will be applied to the component with the name of "select-
pdf-file-button" which is defined in the above template of fragments.
            target: 'select-pdf-file-button',
            // Extend Controller, and implement the handle function.
            callback: CustomController
        }]
    }]

});
```

Refresh your browser (http://127.0.0.1:8080/index.html), and then you can see a new created drop-down menu as follows:

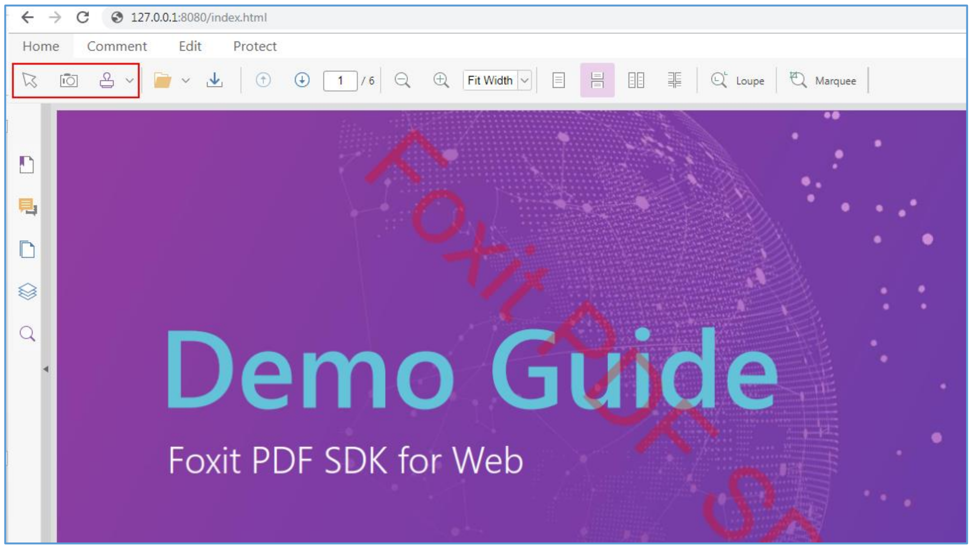


4.2.2 Delete a toolbar button

To delete a toolbar button through fragment is fairly simple. For example, to delete the Hand tool, you only need to add a new object to the fragment. Based on the above example in section 4.2.1, add the code below:

```
{
    target: 'hand-tool',
    action: UIExtension.UIConsts.FRAGMENT_ACTION.REMOVE
}
```

Refresh your browser (http://127.0.0.1:8080/index.html), and then you can see the Hand tool has been removed from the group component as follows:



4.2.3 Modify a toolbar button

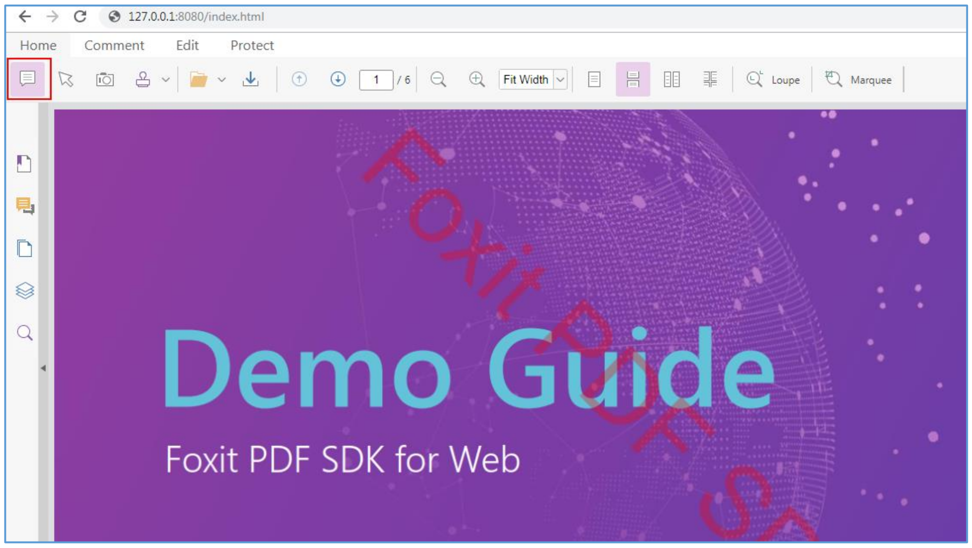
To modify a toolbar button through fragment is also fairly simple. Just like Delete a toolbar button, you only need to add a new object to the fragment.

4.2.3.1 Change icon of a button

For example, to change the icon of the Hand tool, just add the code below based on the example in section 4.2.1:

```
{
    target: 'hand-tool',
    config: {
        iconCls: 'fv__icon-toolbar-note' // your custom icon.
    }
}
```

Refresh your browser (http://127.0.0.1:8080/index.html), and then you can see the icon of the Hand tool has been changed as follows:

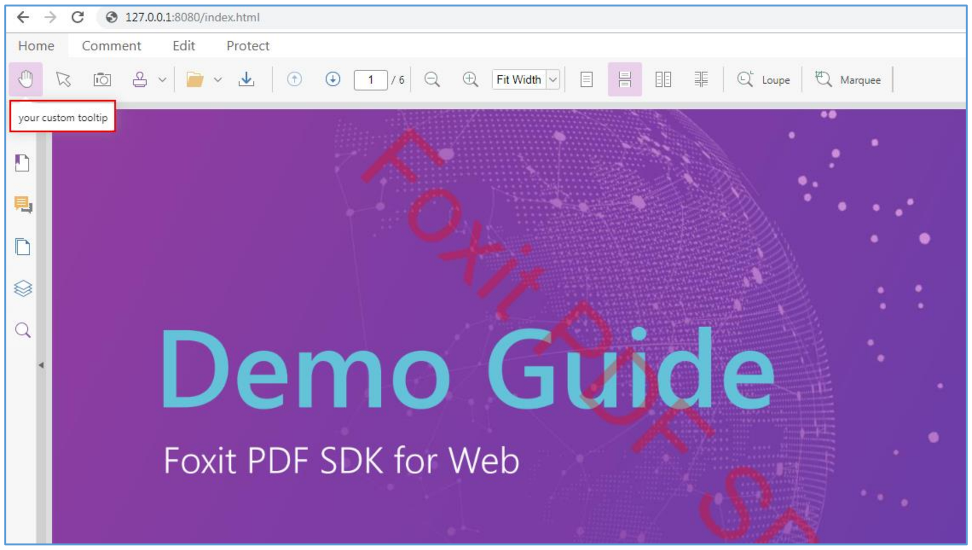


4.2.3.2 Change the tooltip of a button

For example, to change the tooltip of the Hand tool, just add the code below based on the example in section 4.2.1:

```
{
    target: 'hand-tool',
    config: {
        tooltip: {
            title: 'your custom tooltip'
        }
    }
}
```

Refresh your browser (http://127.0.0.1:8080/index.html), and then you can see the tooltip of the Hand tool has been changed to "your custom tooltip" as follows:



4.2.3.3 *Change the event of a button*

To change the event of a button, there are two ways:

a) Overwrite the built-in event. For example:

```
{
    target: 'hand-tool',
    config: {
        callback: function() {
            alert('your click event handler');
        }
    }
}
```

b) Add custom behavior based on the built-in event (original logic). You can configure the button using the method as follows:

For **ES6**:

```
{
    target: 'hand-tool',
    config: {
        callback: {
            before: function(...handleMethodArguments) {
                console.info('called before handle callback with arguments: ', 
handleMethodArguments);
            },
            after: function(value, ...handleMethodArguments) {
                console.info('called after handle callback with returning value and 
arguments: ',value, args );
```

```
            }
        }
    }
}
```

For **ES5**:

```
{
    target: 'hand-tool',
    config: {
        callback: {
            before: function () {
                var handleMethodArguments = Array.prototype.slice.call(arguments);
                console.info('called before handle callback with arguments: ',
handleMethodArguments);
            },
            after: function (value) {
                var handleMethodArguments = Array.prototype.slice.call(arguments, 1);
                console.info('called after handle callback with returning value and
arguments: ', value, handleMethodArguments);
            }
        }
    }
}
```

4.3 Modularization

In order to differentiate the built-in components and user-defined components to avoid conflicts, UIExtension provides modularization feature, which allows you to register the components in different modules separately. Then, you only need to add the prefix of the module name when you declare the components in the template.

4.3.1 Create your own custom module

If you want to define and use a custom component which has the same name with the built-in component, you can create a custom module and then register your custom component in the custom module.

For example, in the built-in component, there is already an existing component with the name of "**dropdown**", if you also want to define a custom component called "**dropdown**", you can refer to the simple code below:

Create a new module "my-widgets", and registers a user-defined component in this module:

For **ES6**:

```
// Create a new module. Please note that the second parameter must be an array if you create a
new module.
let module = UIExtension.modular.module('my-widgets', []);

class UserDefinedDropdownComponent extends UIExtension.Component {
    static getName(){
        return 'dropdown'; // Declare the tag name of the component. There is already an
existing component with the same name of 'dropdown' in the built-in component.
```

```
    }
    render() {
        super.render();
        this.element.innerText = 'User-defined dropdown component';
    }
}
module.registerComponent(UserDefinedDropdownComponent);
```

For **ES5**:

```
// Create a new module. Please note that the second parameter must be an array if you create a
new module.
var module = UIExtension.modular.module('my-widgets', []);

function UserDefinedDropdownComponent() {
    UIExtension.Component.apply(this, arguments);
}
UserDefinedDropdownComponent.getName = function() {
   return 'dropdown'; // Declare the tag name of the component. There is already an existing
component with the same name of 'dropdown' in the built-in component.
}
UserDefinedDropdownComponent.prototype.constructor = UIExtension.Component;
UserDefinedDropdownComponent.prototype.render = function() {
UIExtension.Component.prototype.render.call(this);
       this.element.innerText = 'User-defined dropdown component';
}
module.registerComponent(UserDefinedDropdownComponent);
```

Then, the built-in dropdown and user-defined dropdown can be differentiated in the following way:

```
<!-- built-in dropdown -->
<dropdown></dropdown>
<!-- user-defined dropdown -->
<my-widgets:dropdown></my-widgets:dropdown>
```

For example, use the user-defined dropdown component:

```
var pdfui = new UIExtension.PDFUI({
    // Omit other parameters.
    fragments: [{
        action: UIExtension.UIConsts.FRAGMENT_ACTION.APPEND,
        target: 'home-tab-group-hand',
        template: '<my-widgets:dropdown></my-widgets:dropdown>' // use a colon to separate the
module name and component name in the template.

    }]
});
```

4.4 Understanding the built-in components

In simple terms, a tag defined in the template is a component. Container and Widget also belong to the component. The different is that container can hold subcomponents, but widget cannot. All of the native HTML tags are defined as container components, so you can freely nest them.

Foxit PDF SDK for Web has already wrapped a rich set of built-in components for developers to quickly build a web viewer, and customize it as desired. This section will list and introduce the basic components and business components.

4.4.1 Basic components

Type	Name	Description
container	toolbar	Toolbar components, like an empty div (just has bottom border)
	tabs	Tab components, which are the parent components of tab and used to manage the tab pages. ***Note**: Tab must exist as the subcomponents of the tabs component.*
	tab	Tab pages.
	group-list	Group list, which is used to manage group components.
	group	Group components, which must exist as the subcomponents of group-list.
	sidebar	Sidebar, which managers the sidebar panels.
	sidebar-panel	Sidebar panel, which allow you to specify the icon, title and content of the panel. For example, the *sidebar-search*, and *sidebar-bookmark* in the demo page.
	layer	Floating box component. It supports translucent backdrop and modal box, which can be used to implement dialogs.
	layer-header	A component for the head of a dialog, which can be set with a title and a close button.
	layer-toolbar	Equivalent to a div with 1.5em height and flex layout. It is usually used to store buttons or other components.
	layer-view	Equivalent to a div with *fv_ui-layer-panel*. It is usually used to store the contents of dialogs.
	dropdown	Dropdown component.
	dropdown-item	Dropdown items.
	contextmenu	Context menu component. The subcomponents must be contextmenu-item or contextmenu-separator. To show context menu component, you need to call **showAt** method.
widget	dropdown-button	A type of dropdown items. It will trigger the controller's handle method and click event after clicking. If set to file selector mode, it will trigger the change event after clicking to select a file, and also pass the selected file to the controller's handle method.
container	accordion	Accordion component.
	accordion-card	The card of accordion component. It can be set with a title, and the content can be shrunk. It must be present as the subcomponent of the accordion.
	webpdf	The root component of the default template. It will listen the document open and close event of pdfviewer object, and control to disable/enable all the components depending on the document state.
widget	viewer	A component that renders PDF files. Each template file must be configured with a viewer component.
	text	It will be parsed into a text node in the DOM tree. The content of the text component will be localized by i18next library
	xbutton	Button component. It supports setting the icon and text. After clicking, it will trigger the click event and call the controller's handle method.
	file-selector	File selection component. It supports setting icon and text. After clicking and selecting a file, it will trigger the change event, pass the selected file to the controller's handle method and call it.
	number	Number input box. You can set the number range and step size. The default range is infinite, and it has no step size by default. After setting the step size, if the minimum

Type	Name	Description
		value is not equal to infinitely small, then the value of the input box must be (the minimum value + step size* integer).
	contextmenu-item	Context menu item. It must be as a subcomponent of contextmenu. Other configurations and usages are the same as xbutton.
	contextmenu-separator	The separator line of context menu. It has no other special effects.
slot	slot	Slot can be used to implement the requirement for inserting subcomponents to the different locations of the complex container components. ***Note**: If "for="slotName", slotName" is not supported, it will report the "slot does not exist" error. In this case, it depends on how the corresponding component of the slot's parent node handle the value, please refer to the Component#appendSlot interface.*

4.4.2 Business components

Type	Name	Inherited Component	Description
widget	hand-button	xbutton	Hand tool button
widget	selection-button	xbutton	Text/Annotation selection tool button
widget	snapshot-button	xbutton	Snapshot tool button
widget	open-file-dropdown	dropdown	Dropdown menu for opening document
widget	download-file-button	xbutton	Download file button
widget	goto-prev-page-button	xbutton	Go to the previous page
widget	goto-next-page-button	xbutton	Go to the next page
widget	goto-page-input	number	The input box for the page number that you want to jump to.
widget	zoom-out-button	xbutton	Zoom out page button
widget	zoom-in-button	xbutton	Zoom in page button
widget	editable-zoom-dropdown	dropdown	The editable zoom dropdown menu
widget	zoom-dropdown	dropdown	Zoom dropdown menu
widget	continuous-page-button	xbutton	A button used to enable continuous page mode which views pages continuously with scrolling enabled
widget	continuous-facing-page-button	xbutton	A button used to enable continuous-facing page mode which views pages side-by-side with continuous scrolling enabled
widget	loupe-tool-button	xbutton	Loupe tool button
widget	marquee-tool-button	xbutton	Marquee tool button
widget	create-text-highlight-button	xbutton	Text highlight tool button
widget	create-strikeout-button	xbutton	Text strikeout tool button

Type	**Name**	**Inherited Component**	**Description**
widget	create-underline-button	xbutton	Text underline tool button
widget	create-squiggly-button	xbutton	Text squiggly tool button
widget	create-replace-button	xbutton	Text replace tool button
widget	create-caret-button	xbutton	Caret tool button
widget	create-note-button	xbutton	Note tool button
widget	create-typewriter-button	xbutton	Typewriter tool button
widget	create-callout-button	xbutton	Callout tool button
widget	create-textbox-button	xbutton	Textbox tool button
widget	create-drawings-dropdown	dropdown	Shapes tool dropdown
widget	create-pencil-button	xbutton	Pencil tool button
widget	eraser-button	xbutton	Eraser tool button
widget	create-area-highlight-button	xbutton	Area highlight tool button
widget	create-distance-button	xbutton	Distance tool button
widget	create-attachment-button	xbutton	Attachment tool button
widget	create-image-button	xbutton	Image tool button
widget	create-link-button	xbutton	Link tool button
widget	stamp-dropdown	dropdown	Stamp tool dropdown list
widget	add-image-button	xbutton	Add image tool button
widget	ink-sign-dropdown	dropdown	Ink signature dropdown list
widget	create-stamp-dialog	layer	Create stamp dialog
widget	loupe-tool-dialog	layer	Loupe tool dialog
widget	create-ink-sign-dialog	layer	Create ink signature dialog
widget	bookmark-sidebar-panel	sidebar-panel	Bookmark panel in the sidebar
widget	commentlist-sidebar-panel	sidebar-panel	Comment list panel in the sidebar
widget	thumbnail-sidebar-panel	sidebar-panel	Thumbnail panel in the sidebar
widget	layer-sidebar-panel	sidebar-panel	PDF layers panel in the sidebar

Type	Name	Inherited Component	Description
widget	search-sidebar-panel	sidebar-panel	Search panel in the sidebar
widget	attachment-sidebar-panel	sidebar-panel	Attachments panel in the sidebar
container	page-contextmenu	contextmenu	Page context menu
container	annot-contextmenu	contextmenu	Annotation context menu
container	action-annot-contextmenu	contextmenu	Image and link annotation context menu
container	default-markup-contextmenu	contextmenu	Markup annotation context menu
container	fileattachment-contextmenu	contextmenu	Attachment annotation context menu
container	media-contextmenu	contextmenu	video/audio annotation context menu
container	redact-contextmenu	contextmenu	Redaction context menu
container	textmarkup-contextmenu	contextmenu	Caret/highlight/strikeout/underline/squiggly/typewriter/callout/textbox context menu
widget	contextmenu-item-annot-actions	contextmenu-item	Show action settings dialog
widget	contextmenu-item-apply-all	contextmenu-item	Apply all redaction annotations
widget	contextmenu-item-apply	contextmenu-item	Apply current redaction annotation
widget	contextmenu-item-copy-text	contextmenu-item	Copy the content data of the current annotation
widget	contextmenu-item-delete-annot	contextmenu-item	Delete current annotation
widget	contextmenu-item-hand-tool	contextmenu-item	Switch to hand tool
widget	contextmenu-item-marquee-zoom	contextmenu-item	Switch to loupe tool
widget	contextmenu-item-media-download	contextmenu-item	Download current multimedia data streams
widget	contextmenu-item-place	contextmenu-item	Apply current redaction annotation to other page
widget	contextmenu-item-properties	contextmenu-item	Show the properties dialog of current annotation
widget	contextmenu-item-reply	contextmenu-item	Reply current annotation
widget	contextmenu-item-search	contextmenu-item	Show search sidebar
widget	contextmenu-item-select-tool	contextmenu-item	Switch to selection tool

Type	Name	Inherited Component	Description
widget	contextmenu-item-zoom-actual-size	contextmenu-item	Zoom to 100%
widget	contextmenu-item-fitpage	contextmenu-item	Fit page (resize the page to fit entirely in the document pane)
widget	contextmenu-item-fitwidth	contextmenu-item	Fit width (resize the page to fit the width of the window. Part of the page may be out of view)
widget	contextmenu-item-zoomin	contextmenu-item	Zoom in
widget	contextmenu-item-zoomout	contextmenu-item	Zoom out

4.4.3 Business components in the add-on modules

The business components in the add-on modules are valid only when your project has already integrated the related add-on modules, please refer to section 3.4 "Integrate the add-ons of UIExtension".

Type	Module:Name	Inherited Component	Add-on Module	Description
widget	print:print-button	xbutton	print	Open the print dialog
widget	print:print-dialog	layer	print	Print dialog
widget	import-form-module:import-form-button	xbutton	import-form	Import form data
widget	export-form-module:import-form-dropdown	xbutton	export-form	Export form data
widget	fpmodule:file-property-button	xbutton	file-property	Show current document information dialog
widget	fpmodule:file-property-dialog	layer	file-property	Document information dialog
widget	multi-media:multi-media-button	xbutton	multi-media	Switch to the multimedia Annotation creation tool
widget	edit-pageobjects:edit-all-objects-button	xbutton	path-objects	Switch to the page objects (image/path/text) editing tool
container	edit-pageobjects:page-objects-dropdown	dropdown	path-objects	Path objects creation tool dropdown list
widget	edit-text-object:add-text-button	xbutton	text-object	Text editing
widget	edit-text-object:text-bold-style-button	xbutton	text-object	Make text bold
widget	edit-text-object:text-italic-style-button	xbutton	text-object	Italicize text
container	edit-text-object:font-color-picker	dropdown	text-object	Set the color of text

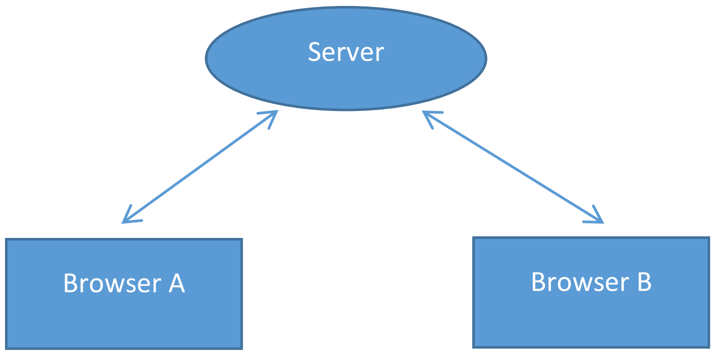
Type	Module:Name	Inherited Component	Add-on Module	Description
container	edit-text-object:font-style-dropdown	dropdown	text-object	Set font and font size of text
widget	password-protect:password-protect-button	xbutton	password-protect	Open the password protect dialog
widget	password-protect:remove-protect-button	xbutton	password-protect	Remove password protect
container	redaction:create-redactions-dropdown	dropdown	redaction	Redaction options dropdown list
widget	redaction:apply-redactions-button	xbutton	redaction	Apply redaction
widget	redaction:redaction-search-button	xbutton	redaction	Open the left search panel
container	redaction:redaction-page-dialog	layer	redaction	Mark page to redact dialog

5 Implementing Annotation Real-time collaboration

Foxit PDF SDK for Web supports annotation real-time collaboration which means when two browsers are opening a same PDF document, if one browser adds an annotation event, the event will be added synchronization to the other browser. This section will introduce how to implement annotation real-time collaboration using Foxit PDF SDK for Web.

5.1 Device requirement

- Two browser terminals
- A host with http-server (to handle network events and distribution)
- Foxit PDF SDK for Web 7.0



5.2 Development process for implementing annotation real-time collaboration

To implement annotation real-time collaboration, please refer to the steps below:

1) Initialize the Foxit PDF SDK for Web. Please refer to the step c) in the section "Integrate the simple UI demo to your project".

2) Add the "annotation-add" events as below:

   For **ES6**:

```
pdfViewer.eventEmitter.on('annotation-add', (annotations) => {
    // annotations : the newly added annotations
    let xmlHttp = new XMLHttpRequest();
    let url = 'url/to/post/added/annots'
    xmlHttp.open('POST', url, true);

    let formData = new FormData();

    let annotJsons = []
    annotations.map(ele => {
        annotJsons.push(ele.exportToJSON());
    })
    formData.append('annots', annotJsons)
    xmlHttp.send(formData);
    // handle loaded event.
})
```

For **ES5**:

```
pdfViewer.eventEmitter.on('annotation-add',function(annotations) {
    // annotations : the newly added annotations
    var xmlHttp = new XMLHttpRequest();
    var url = 'url/to/post/added/annots';
    xmlHttp.open('POST', url, true);

    var formData = new FormData();

    var annotJsons = [];
    annotations.map(function(ele) {
        annotJsons.push(ele.exportToJSON());
    });
    formData.append('annots',annotJsons);
    xmlHttp.send(formData);
    // handle loaded event.
});
```

3) Pull the server distribution event at a regular time.

For **ES6**:

```
setInterval(() => {
    let xmlHttp = new XMLHttpRequest();
    let url = 'url/to/get/added/annots'
    xmlHttp.open('GET', url, true);
    xmlHttp.send(null);

    xmlHttp.onload = (e) => {
        if (xhr.readyState === 4) {
            if (xhr.status === 200) {
                // Assuming that directly returns the JSON data of the annotations.
                let annotsJson = xmlHttp.response;

pdfViewer.getCurrentPDFDoc().getPageByIndex(annotsJson.pageIndex).then(page => {
                    annotsJson.annots.map(annotJSON => {
                        page.addAnnot(annotJSON);
                    })
                })
            }
        }
    }
}, 1000);
```

For **ES5**:

```
setInterval(function() {

    var xmlHttp = new XMLHttpRequest();
    var url = 'url/to/get/added/annots';
    xmlHttp.open('GET', url, true);
    xmlHttp.send(null);

    xmlHttp.onload = function(e) {
        if (xhr.readyState === 4) {
            if (xhr.status === 200) {
                // Assuming that directly returns the JSON data of the annotations.
                var annotsJson = xmlHttp.response;

pdfViewer.getCurrentPDFDoc().getPageByIndex(annotsJson.pageIndex).then(function(page)
{
                    annotsJson.annots.map(function(annotJSON) {
                        page.addAnnot(annotJSON);
                    })
                })
            }
        }
    }
}, 1000);
```

4) Restart the server, open a PDF document in two browsers, add an annotation event to one of the browser, then you will find that the event will be added synchronization to the other browser.

6 Technical Support

Reporting Problems

Foxit offers 24/7 support for its products and are fully supported by the PDF industry's largest development team of support engineers. If you encounter any technical questions or bug issues when using Foxit PDF SDK for Web, please submit the problem report to the Foxit support team at http://tickets.foxitsoftware.com/create.php. In order to better help you solve the problem, please provide the following information:

- Contact details
- Foxit PDF SDK product and version
- Your Operating System and IDE version
- Detailed description of the problem
- Any other related information, such as log file or error screenshot

Contact Information

You can contact Foxit directly, please use the contact information as follows:

Foxit Support:

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

Sales Contact:

- Phone: 1-866-680-3668
- Email: sales@foxitsoftware.com

Support & General Contact:

- Phone: 1-866-MYFOXIT or 1-866-693-6948
- Email: support@foxitsoftware.com