Getting Started with WebViewer.js

The PDFTron WebViewer is a JavaScript wrapper control that encapsulates the XOD and PDF viewer technologies. It provides a single, easy-to-use interface for loading documents and controlling the viewers.

WebViewer will automatically select the appropriate type of viewer depending on web browser support.

Viewer Technologies

HTML5 Viewer
Included under lib/html5/ is the HTML5 ReaderControl, which leverages modern browsers' capabilities such as vector graphics and fonts to deliver the best possible user experience without the need for any plugins.

HTML5 Mobile Viewer
Also included as part of the HTML5 viewer, the Mobile ReaderControl uses the same powerful HTML5 technology, but is optimized for mobile devices and tablets with touch events.

PDFNet.js
Included under lib/html5/pdf is the PDF ReaderControl which extends the capabilities of WebViewer to take advantage of PDFTron's PDFNet in the browser (PDFNet.js) for precise rendering of PDF files. PDF documents do not need to be converted to the XOD format to work with this backend. Note that a license key specificly for PDFNet.js is required for viewing PDF files without a demo stamp.

Create your own WebViewer page

1. Download the WebViewer SDK and unzip the package.
2. Copy the lib/ folder and GettingStarted.xod to a location on your web server.
3. Create an HTML page.
4. Add the necessary scripts to the <head> of the HTML page.

```
<script src="jquery-1.7.2.min.js"></script>
<script src="lib/WebViewer.js"></script>
```

   WebViewer.js depends on jQuery so it must be included. Instead of including WebViewer.js you could include WebViewer.min.js which is the minified version of the file.

5. Create a tag in the HTML <body> and give it an id. This will be the container for the WebViewer.

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

   Add a style for the viewer element so that it takes up a reasonably sized area.

```
<style>
    #viewer {
        width: 1024px;
        height: 600px;
    }
</style>
```

6. Add the following script to create a new WebViewer instance

```
<script>
    $(function() {
        var viewerElement = document.getElementById("viewer");
        var myWebViewer = new PDFTron.WebViewer({
            path: "lib",
            type: "html5",
            initialDoc: "GettingStarted.xod"
        }, viewerElement);
    });
</script>
```

The script above will create and render a WebViewer control on the page as a child of the provided element.

To load a PDF file simply specify the documentType option with a value of "pdf" and make sure that initialDoc points to a PDF file. Note that if you don't specify a (valid) license key then a demo stamp will be applied to the document. For example:

```
var myWebViewer = new PDFTron.WebViewer({
    path: "lib",
    type: "html5",
    l: "Your License Key Here",
    documentType: "pdf",
    initialDoc: "GettingStarted.pdf"
}, viewerElement);
```

Try viewing your HTML page on your web server. See an example here.

Troubleshooting

If the viewer or XOD file didn't load correctly then there could be a few reasons.

- Make sure that the initialDoc option is pointing to the correct file and that the XOD file exists on your server. The path should be relative to the HTML file.
- Make sure that the path option is pointing to the lib folder. It should be relative to the HTML file.
- The .xod file extension might need to be added to your server's allowed mime types list. Please consult your server's manual on how to add the following entry to its mime types list. Extension: .xod, Mime type: application/octet-stream or application/vnd.ms-xpsdocument.

Note
For best performance your server should support HTTP range requests (most servers support this) if you are loading XOD files. If range requests are not supported then the HTML5 viewer will automatically fall back to downloading the entire document up front which can be slow. When range requests are used only the necessary parts of the

file are downloaded as needed, providing much better performance and stability.

When using PDFNet.js to view PDF documents this is not applicable as the entire document needs to be downloaded.

7. It is now possible to use the "myWebViewer" variable to invoke ReaderControl's methods. For example, the following code demonstrates how to jump to page 2:

```
myWebViewer.setCurrentPageNumber(2);
```

Additionally you can use jQuery event bindings to react to changed events. For example the following code shows how to be notified when the page changes.

```
$(viewerElement).on("pageChanged", function(event) {
    alert("Current page is: " + myWebViewer.getCurrentPageNumber());
});
```

Please refer to the API documentation for other methods and events which can be used.