Using PDFKit in the browser

PDFKit can be used in the browser as well as in Node! There are two ways to use PDFKit in the browser. The first is to create an app using an module bundler like Browserify or Webpack. The second is to create a standalone pdfkit script as explained here.

Using PDFKit in the browser is exactly the same as using it in Node, except you'll want to pipe the output to a destination supported in the browser, such as a Blob. Blobs can be used to generate a URL to allow display of generated PDFs directly in the browser via an **`iframe`**, or they can be used to upload the PDF to a server, or trigger a download in the user's browser.

To get a Blob from a **`PDFDocument`**, you should pipe it to a blob-stream, which is a module that generates a Blob from any Node-style stream. The following example uses Browserify to load **`PDFKit`** and **`blob-stream`**, but if you're not using Browserify, you can load them in whatever way you'd like (e.g. script tags).

```
// require dependencies
const PDFDocument = require('pdfkit');
const blobStream  = require('blob-stream');

// create a document the same way as above
const doc = new PDFDocument;

// pipe the document to a blob
const stream = doc.pipe(blobStream());

// add your content to the document here, as usual

// get a blob when you're done
doc.end();
stream.on('finish', function() {
  // get a blob you can do whatever you like with
  const blob = stream.toBlob('application/pdf');

  // or get a blob URL for display in the browser
  const url = stream.toBlobURL('application/pdf');
  iframe.src = url;
});
```

You can see an interactive in-browser demo of PDFKit here.

Note that in order to Browserify a project using PDFKit, you need to install the **`brfs`** module with npm, which is used to load built-in font data into the package. It is listed as a **`devDependencies`** in PDFKit's **`package.json`**, so it isn't installed by default for Node users. If you forget to install it, Browserify will print an error message.