PDFKit

A JavaScript PDF generation library for Node and the browser.

Description

PDFKit is a PDF document generation library for Node and the browser that makes creating complex, multi-page, printable documents easy. The API embraces chainability, and includes both low level functions as well as abstractions for higher level functionality. The PDFKit API is designed to be simple, so generating complex documents is often as simple as a few function calls.

Check out some of the documentation and examples to see for yourself! You can also read the guide as a self-generated PDF with example output displayed inline. If you'd like to see how it was generated, check out the README in the docs folder.

You can also try out an interactive in-browser demo of PDFKit here.

Installation

Installation uses the npm package manager. Just type the following command after installing npm.

```
npm install pdfkit
```

Features

- Vector graphics
  - HTML5 canvas-like API
  - Path operations
  - SVG path parser for easy path creation
  - Transformations
  - Linear and radial gradients
- Text
  - Line wrapping
  - Text alignments
  - Bulleted lists
- Font embedding
  - Supports TrueType (.ttf), OpenType (.otf), WOFF, WOFF2, TrueType Collections (.ttc), and Datafork TrueType (.dfont) fonts
  - Font subsetting
  - See fontkit for more details on advanced glyph layout support.
- Image embedding
  - Supports JPEG and PNG files (including indexed PNGs, and PNGs with transparency)
- Annotations
  - Links
  - Notes
  - Highlights
  - Underlines
  - etc.
- AcroForms
- Outlines
- PDF security
  - Encryption
  - Access privileges (printing, copying, modifying, annotating, form filling, content accessibility, document assembly)
- Accessibility support (marked content, logical structure, Tagged PDF, PDF/UA)

Coming soon!

- Patterns fills
- Higher level APIs for creating tables and laying out content
- More performance optimizations
- Even more awesomeness, perhaps written by you! Please fork this repository and send me pull requests.

Example

```
const PDFDocument = require('pdfkit');
const fs = require('fs');

// Create a document
const doc = new PDFDocument();

// Pipe its output somewhere, like to a file or HTTP response
// See below for browser usage
doc.pipe(fs.createWriteStream('output.pdf'));

// Embed a font, set the font size, and render some text
doc
  .font('fonts/PalatinoBold.ttf')
  .fontSize(25)
  .text('Some text with an embedded font!', 100, 100);

// Add an image, constrain it to a given size, and center it vertically and horizontally
doc.image('path/to/image.png', {
  fit: [250, 300],
  align: 'center',
  valign: 'center'
});

// Add another page
doc
  .addPage()
  .fontSize(25)
  .text('Here is some vector graphics...', 100, 100);

// Draw a triangle
doc
  .save()
  .moveTo(100, 150)
  .lineTo(100, 250)
  .lineTo(200, 250)
  .fill('#FF3300');

// Apply some transforms and render an SVG path with the 'even-odd' fill rule
doc
  .scale(0.6)
  .translate(470, -380)
  .path('M 250,75 L 323,301 131,161 369,161 177,301 z')
  .fill('red', 'even-odd')
  .restore();

// Add some text with annotations
doc
  .addPage()
  .fillColor('blue')
  .text('Here is a link!', 100, 100)
  .underline(100, 100, 160, 27, { color: '#0000FF' })
  .link(100, 100, 160, 27, 'http://google.com/');

// Finalize PDF file
doc.end();
```

The PDF output from this example (with a few additions) shows the power of PDFKit — producing complex documents with a very small amount of code. For more, see the **demo** folder and the PDFKit programming guide.

Browser Usage

There are three ways to use PDFKit in the browser:

- Use Browserify. See demo source code and build script
- Use webpack. See complete example.
- Use prebuilt version. Distributed as `pdfkit.standalone.js` file in the releases or in the package `js` folder.

In addition to PDFKit, you'll need somewhere to stream the output to. HTML5 has a Blob object which can be used to store binary data, and get URLs to this data in order to display PDF output inside an iframe, or upload to a server, etc. In order to get a Blob from the output of PDFKit, you can use the blob-stream module.

The following example uses Browserify or webpack to load `PDFKit` and `blob-stream`. See here and here for examples of prebuilt version usage.

```
// 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 `devDependency` 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.

License

Next