Switching to previous pages

PDFKit normally flushes pages to the output file immediately when a new page is created, making it impossible to jump back and add content to previous pages. This is normally not an issue, but in some circumstances it can be useful to add content to pages after the whole document, or a part of the document, has been created already. Examples include adding page numbers, or filling in other parts of information you don't have until the rest of the document has been created.

PDFKit has a **`bufferPages`** option in versions v0.7.0 and later that allows you to control when pages are flushed to the output file yourself rather than letting PDFKit handle that for you. To use it, just pass **`bufferPages: true`** as an option to the **`PDFDocument`** constructor. Then, you can call **`doc.switchToPage(pageNumber)`** to switch to a previous page (page numbers start at 0).

When you're ready to flush the buffered pages to the output file, call **`flushPages`**. This method is automatically called by **`doc.end()`**, so if you just want to buffer all pages in the document, you never need to call it. Finally, there is a **`bufferedPageRange`** method, which returns the range of pages that are currently buffered. Here is a small example that shows how you might add page numbers to a document.

```
// create a document, and enable bufferPages mode
let i;
let end;
const doc = new PDFDocument({
  bufferPages: true});

// add some content...
doc.addPage();
// ...
doc.addPage();

// see the range of buffered pages
const range = doc.bufferedPageRange(); // => { start: 0, count: 2 }

for (i = range.start, end = range.start + range.count, range.start <= end; i < end; i+
+;) {
  doc.switchToPage(i);
  doc.text(`Page ${i + 1} of ${range.count}`);
}

// manually flush pages that have been buffered
doc.flushPages();

// or, if you are at the end of the document anyway,
// doc.end() will call it for you automatically.
doc.end();
```