You'll recognize the five steps in the PDF creation process discussed in part 1. Now you're also creating a `PdfReader` object and looping over all the pages, getting `PdfImportedPage` instances with the `getImportedPage()` method (as highlighted in bold). What does this method do?

If you browse the API of the `PdfReader` class, you'll discover the `getPageContent()` method, which returns the content stream of a page. This content stream is very similar to what's inside the hero.txt file. In general, such a content stream contains references to external objects, images, and fonts.

In section 3.4.1, for instance, we examined the PDF syntax needed to draw a raster image:

In this snippet, `/img0` referred to a key in the `/Resources` dictionary of the page. The corresponding value was a reference to a stream object containing the bits and bytes of the image. Without the bits and bytes of the image, the PDF syntax referring to `/img0` is

It doesn't make sense to get the content stream of a page from one PDF document, and copy that stream into another PDF *without* copying all the resources that are needed.

The `Hero` example was an exception: the syntax to draw the vector image of Superman was self-contained, and this is very unusual. As soon as there's text involved, you'll have at least a reference to a font. If you don't copy that font, you'll get warnings or errors, such as "Could not find a font in the Resources dictionary." That's why it's never advisable to extract a page from `PdfReader` directly. Instead, you should pass the reader object to the writer class, and ask the writer (not the reader!) to import a page. A `PdfImportedPage` object is returned. Behind the scenes, all the necessary resources (such as images and fonts) are retrieved and copied to the writer.

FAQ *Why are all my links lost when I copy a page with* `PdfImportedPage`*?* It's important to understand the difference between resources needed to render the content of a page and the interactive features of a page. In general, these features are called *annotations*. They include links, text annotations, and form fields. Annotations aren't part of the content stream. They aren't listed in the resources dictionary of the page, but in the annotation dictionary. These interactive features aren't copied when using `PdfImportedPage`, which means that all interactivity is lost when copying a page with the `getImportedPage()` method of the `PdfWriter` class.

The `PdfImportedPage` class extends `PdfTemplate`, but you can't add any new content to it. It's a read-only XObject you can reuse in a document with the method `addTemplate()`; or you can wrap it inside an `Image`. You've already used these techniques in