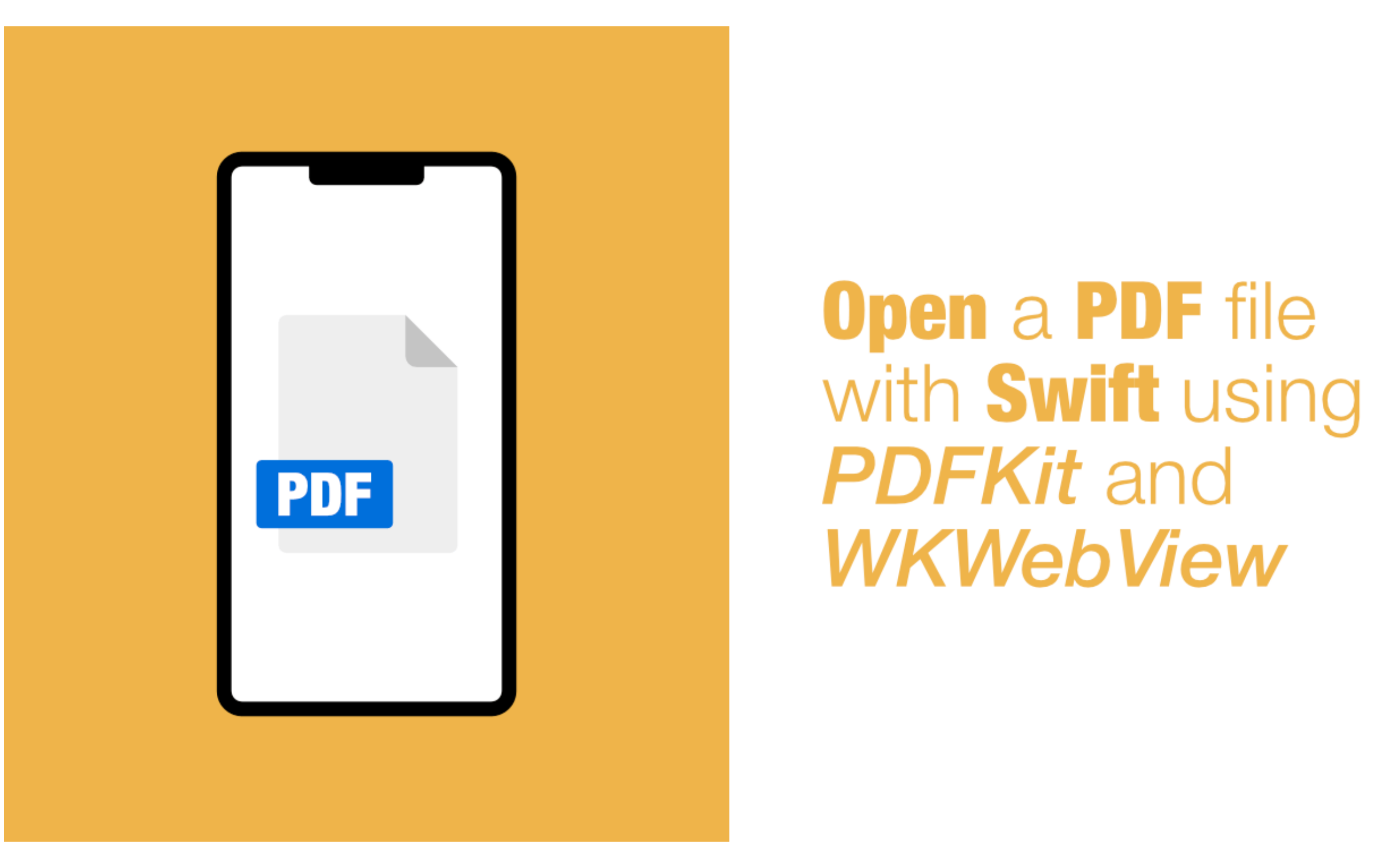
乔纳森·斯威夫特

使用Swift（PDFKit，WKWebView）打开PDF文件

达伦
2020年4月18日 • 7分钟阅读



PDF文件是常见的，随着iOS/iPadOS成为更强大的操作系统，打开PDF文件和其他常规计算机类型任务将变得更加常见。

在本教程中，我们探索了使用Swift打开PDF文件的两种方法。PDF可以是本地文件或托管文件，但我们将使用本地文件。如果您想使用托管文件，它将以基本相同的方式工作。我们今天要使用的两种方法都允许我们在尝试加载PDF时使用 `URL`，因此 `URL` 是否是本地文件并不重要。

尽管本教程不专注于打开远程PDF，但我将在每个部分的末尾添加代码，允许您打开远程PDF文件。您还可以在这里找到本教程的完整源代码。

现在我们知道我们在做什么了，让我们进入代码：

第1步：获取本地PDF的URL

正如我在本教程前面提到的，我们将专注于如何打开本地PDF文件。为了做到这一点，我们需要一个PDF文件，我们可以添加到我们的项目中。为此，我使用了以下PDF文件-https://web.stanford.edu/class/archive/cs/cs161/cs161.1168/lecture4.pdf。我已将此文件命名为 `heaps.pdf`，因此在下面的代码中，您将看到我使用文件名 `heaps`。

我不会将PDF文件包含在回购协议中，因为我不确定它有什么许可证，以及我是否被允许分发它。

由于我们正在将此PDF添加到我们的项目中，我们可以使用 `Bundle.main` 来获取文件的 `URL`。为此，我们需要创建以下方法：

```
private func resourceUrl(forFileName fileName: String) -> URL? {
    if let resourceUrl = Bundle.main.url(forResource: fileName,
                                         withExtension: "pdf") {
        return resourceUrl
    }

    return nil
}
```

此方法将以文件名为参数。如果文件存在，那么我们将返回 `resourceUrl`，如果它不存在，我们将返回 `nil`。

我们将对 `WKWebView` 和 `PDFKit` 都使用此方法。

> *注意：本教程的所有代码都将在我的 `ViewController` 文件中，因为我正在处理一个新项目，这是一个教程。如果这不是教程，我会把这个代码放在一个更合适的地方。*

使用WKWebView打开PDF

现在我们有了 `resourceUrl` 方法，该方法将返回PDF文件的 `URL`，我们可以实现 `createWebView` 方法。在我们这样做之前，我们需要导入 `WebKit`。

将以下导入添加到您的文件中：

```
import WebKit
```

好的，既然我们已经导入了 `WebKit`，我们可以创建我们的新方法。这个新方法将创建一个新的 `WKWebView`，然后它将加载 `URL` 并返回创建的网页视图。但是，只有当PDF存在时，才会发生这种情况，如果没有，那么此方法将返回零。

```
private func createWebView(withFrame frame: CGRect) -> WKWebView? {
    let webView = WKWebView(frame: frame)
    webView.autoresizingMask = [.flexibleWidth, .flexibleHeight]

    if let resourceUrl = self.resourceUrl(forFileName: "heaps") {
        webView.loadFileURL(resourceUrl,
                            allowingReadAccessTo: resourceUrl)

        return webView
    }

    return nil
}
```

上述方法将 `frame` 作为参数，以便我们在初始化时设置 `WKWebView` 帧。

之后，我们用 `.flexibleWidth` 和 `.flexibleHeight` 设置 `autoResizingMask`。

现在我们将使用我们在步骤1中创建的 `resourceUrl` 方法。我们用 `heaps` 调用theresourceUrl方法作为 `fileName` 参数值，这是我的PDF文件的名称，请将此名称更改为您调用的PDF文件。如果 `resourceUrl` 返回 `URL`，我们将使用 `loadFileUrl` 将其加载到网络视图中。如果它返回nil，我们将返回nil，因为webview将是空的。

> *如果你想要一个关于用WKWebView加载本地文件的完整教程，我已经写了一个，你可以在这里找到。*

显示网页视图

现在我们已经创建了一个网页视图来显示PDF，我们需要显示网页视图。为此，我们将创建一个名为 `displayWebView` 的新方法，其外观如下：

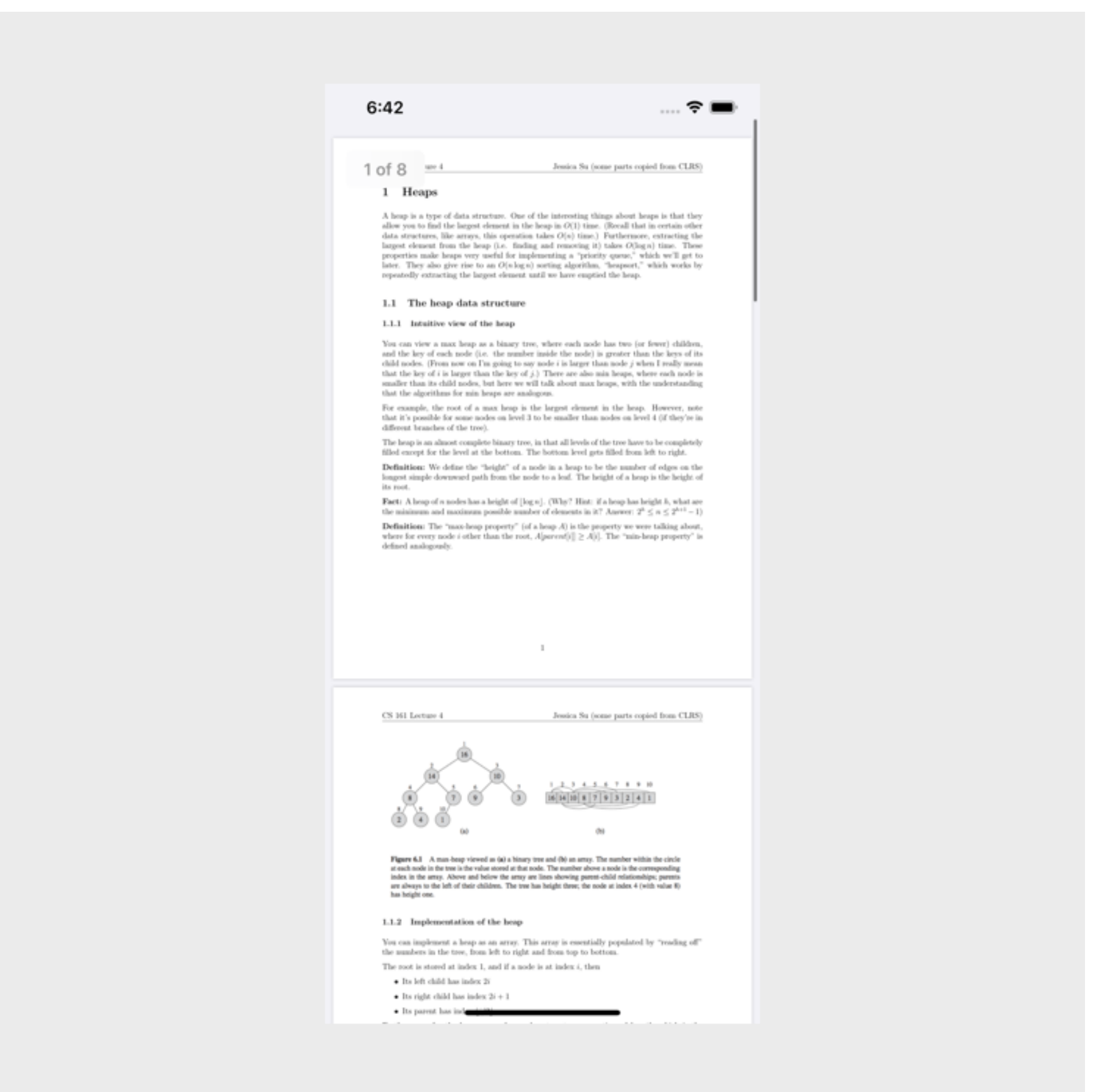
```
private func displayWebView() {
    if let webView = self.createWebView(withFrame: self.view.bounds) {
        self.view.addSubview(webView)
    }
}
```

这里没什么复杂的。我们用视图的边界调用 `createWebView` 方法。如果此方法返回webview，我们将将其添加为subview toself `self.view`，如果它返回 `nil`，则不会发生任何事情。

我们现在可以通过在ourviewDidLoad方法中调用 `displayWebView` 来显示webview。更新您的 `viewDidLoad` 方法，如下所示：

```
override func viewDidLoad() {
    super.viewDidLoad()
    self.displayWebView()
}
```

如果您现在构建并运行该应用程序，您应该会看到以下内容：



使用WKWebView打开远程PDF

将本地PDF文件加载到 `WKWebView` 很容易，但从远程 `URL` 加载一个文件甚至更容易，因为我们不需要使用resourceUrl方法。如果您想从我在步骤1中链接的文件的网址打开PDF，您可以将 `createWebView` 方法更新为以下内容：

```
private func createWebView(withFrame frame: CGRect) -> WKWebView? {
    let webView = WKWebView(frame: frame)
    webView.autoresizingMask = [.flexibleWidth, .flexibleHeight]

    if let resourceUrl = URL(string: "https://web.stanford.edu/class/arch
        let request = URLRequest(url: resourceUrl)
        webView.load(request)

        return webView
    }

    return nil
}
```

我们没有调用 `resourceUrl`，而是使用了 `URL(string: )` 如果我们传递的字符串有效，它将返回aURL。

无法使用 `loadFileURL` 打开远程PDF，因为 `loadFileURL` 仅适用于本地文件。因此，为了加载远程 `URL`，我们需要创建一个 `URLRequest`，并将 `URLRequest` 传递给名为 `load` 的 `WKWebView` 方法。

如果您现在构建并运行该应用程序，您将看到PDF按预期加载，但是，由于需要下载文件，因此需要更长的时间。

使用PDFKit打开PDF

如果您跳过了第1步，请确保在继续之前阅读。在步骤1中，我们创建了 `resourceUrl` 方法，该方法将返回文件 `URL`。

使用 `PDFKit` 时，我们将使用两个主要类。这两个类是 `PDFView` 和 `PDFDocument`。

`PDFView` 是 `UIView` 的一个子类，它将是显示PDF文件的视图。`PDFDocument` 加载PDF文件，并允许我们在 `PDFView` 上设置 `document` 属性。

在使用这些类之前，我们需要导入 `PDFKit`。为此，将以下导入添加到您的文件中：

```
import PDFKit
```

现在我们已经导入了 `PDFKit`，我们可以开始创建允许我们使用它的方法。

我们将创建的第一个方法是 `createPdfView`。这种方法将把一个框架作为参数。当我们初始化 `PDFView` 时，我们将使用它。

将以下代码添加到您的文件中：

```
private func createPdfView(withFrame frame: CGRect) -> PDFView {
    let pdfView = PDFView(frame: frame)
    pdfView.autoresizingMask = [.flexibleWidth, .flexibleHeight]
    pdfView.autoScales = true

    return pdfView
}
```

This method will initialise a `PDFView`, it will set the `autoResizingMask` and then we will set `autoScales` as true. If we don't set the `autoScales` the scale of the PDF document won't be correct. If you want you can set this manually, but for this tutorial using `autoScales` is fine. Once the `PDFView` has be setup we return it.

我们现在可以使用resourceUrl方法并创建一个 `PDFDocument`。为此，请在代码中添加以下方法：

```
private func createPdfDocument(forFileName fileName: String) -> PDFDocumen
    if let resourceUrl = self.resourceUrl(forFileName: fileName) {
        return PDFDocument(url: resourceUrl)
    }

    return nil
}
```

此方法将获取我们想要打开的文件的文件名。我们将该文件名传递给 `resourceUrl`，如果文件存在，我们将创建一个 `PDFDocument`，如果它不存在，我们将返回nil。

所有的辛勤工作现在都完成了。让我们创建我们的显示方法。添加以下代码：

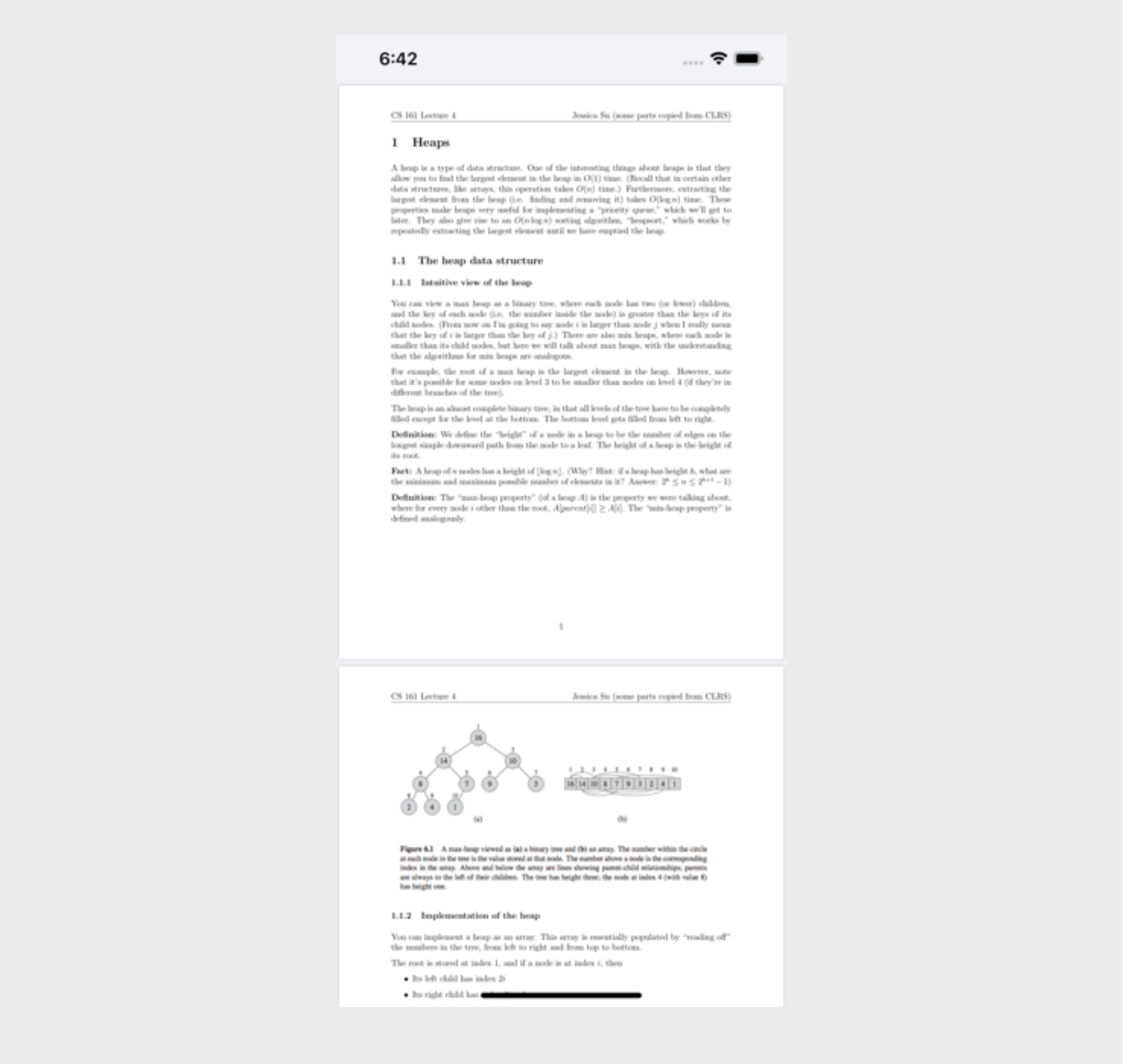
```
private func displayPdf() {
    let pdfView = self.createPdfView(withFrame: self.view.bounds)

    if let pdfDocument = self.createPdfDocument(forFileName: "heaps") {
        self.view.addSubview(pdfView)
        pdfView.document = pdfDocument
    }
}
```

在此方法中，我们调用 `createPdfView`，并传递 `self.view.bounds` 来设置 `PDFView` 的帧。接下来，我们在 `if let` 中调用 `createPdfDocument`。如果可以创建 `PDFDocument`，我们将添加 `pdfView` 作为subview toself `self.view`，并将 `pdfDocument` 分配给 `pdfView.document`，以便当视图显示时，PDF将可见。

我们现在需要做的最后一件事就是从ourviewDidLoad调用 `displayPdf`。将您的 `viewDidLoad` 更新为以下内容：

```
override func viewDidLoad() {
    super.viewDidLoad()
    self.displayPdf()
}
```

如果您现在构建并运行该应用程序，您将看到以下内容：



当将其与webview实现进行比较时，这看起来几乎相同。除了缩放之外，还有一个视觉上的区别。使用 `PDFView` 时，我们在左上角没有页面计数，而网页视图是这样做的。

使用PDFKit打开远程PDF文件

要使用 `PDFKit` 打开远程PDF，我们需要在 `createPdfDocument` 方法中更改一行。

更新 `createPdfDocument` 方法，如下：

```
private func createPdfDocument(forFileName fileName: String) -> PDFDocumen
    if let resourceUrl = URL(string: "https://web.stanford.edu/class/arch
        return PDFDocument(url: resourceUrl)
    }

    return nil
}
```

上述方法的唯一更改是删除调用 `self.resourceUrl` 并将其替换为 `URL(string: )` 如果您现在构建并运行该应用程序，一切都将按预期工作，只是PDF的显示需要更长的时间，因为它需要下载。

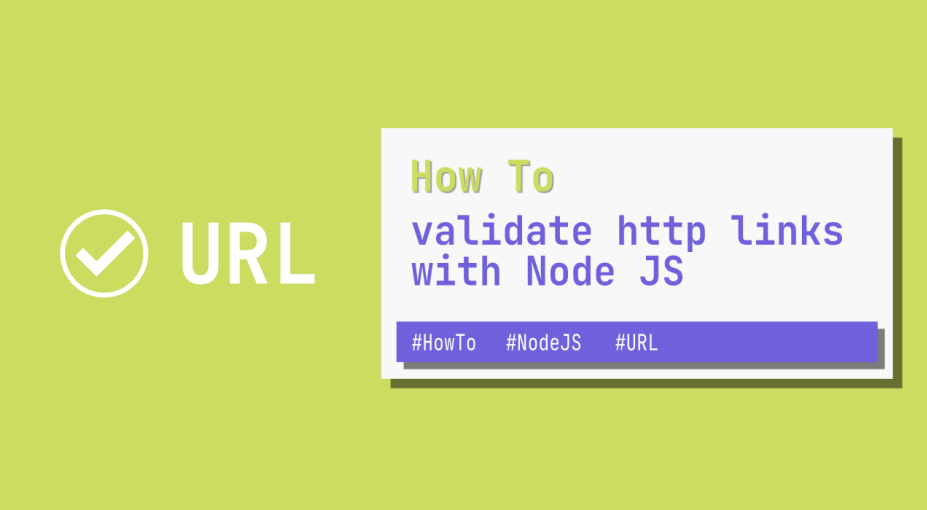
结论：

无论文件是否是设备本地的，这两种方法都可以轻松打开PDF文件。决定使用哪个由你决定。一种方法可能更适合您的需求，这完全取决于您的要求是什么。

如果你想查看完整的来源，你可以在这里找到它。

注册更多像这样的东西。

输入您的电子邮件 订阅



如何使用Node JS验证URL

最近我需要验证与Node的http链接。我认为这应该很容易，而且事实就是如此。幸运的是，Node有一个...

2022年5月17日 · 2分钟阅读

Grokking算法评论

> TLDR：如果您是算法和数据结构的新手，我强烈推荐Grokking算法。这是我读到的第一本关于这个主...

2022年5月12日 · 8分钟阅读

转到：逐行阅读文件

在本教程中，我们将研究如何使用Go逐行读取文件。Go通过使用bufio.NewScanner()使这变得非常容易...

2021年9月19日 · 3分钟阅读