📄 Nimipdf

Nimipdf is a nimibex extension that adds a PDF backend for nimib. This allows you to generate PDFs with nimib (in fact, you may be reading one right now!).

🔽 Installation

Install nimipdf with Nimble:

```
nimble install nimipdf
```

🦵 Limitations

Emoji's don't work properly, so Twemoji is used instead of your OS's native emoji set.

In addition, images *might* require the absolute path to the image.

📝 How Nimipdf Works!

PDFs are generated through the use of libwkhtmltox Nim bindings. This allows for quick and simple PDF generation while also providing easy customizaton.

Nimipdf provides two main templates, so lets talk about those.

- `nbInitPdf`:
    - When calling `nbInitPdf`, libwkhtmltox (and nimib) are initalized, and the converter, global settings, and object settings are created and attached onto the injected `nbPdf` variable.
- `nbSavePdf`:
    - When saving via `nbSavePdf`, nimipdf takes the rendered html output of your nimib document and converts it into a PDF using libwkhtmltox.

      Before this, however, the `out` and `documentTitle` global settings are set to the nimib document's filename (`nb.filename`) and title (`doc.context["title"]`) respectively.

      Libwkhtmltopdf is also de-initialized when calling `nbSavePdf`.

🐝 API

- `nbInitPdf` injects a `nbPdf` variable
    - `nbPdf.converter` contains the wkhtmltopdf converter used to generate the PDF.
    - `nbPdf.globalSettings` contains the wkhtmltopdf global settings. A list of available settings to set can be found here, and can be set via `nbPdf.setGlobalSetting(name, value)`
    - `nbPdf.objectSettings` contains the wkhtmltopdf object settings. A list of available settings to set can be also found here, and can be set via a similar way to global settings (`setObjectSetting`)
- use `nbPageBreak` to insert a page break
- use `nbSavePdf` to save document

The `pdf` submodule from nimwkhtmltox is exported, so you can also use functions and methods from wkhtmltox aswell to further modify nimipdf's behavior (for example, adding callback functions to `nbPdf.converter`).

⁉️ How to Use

- Initialize nimipdf (and wkhtmltopdf) with `nbInitPdf`
- Write your usual nimib code (nbText, nbCode, etc.)
- Save the PDF using `nbSavePdf` (this also de-inits wkhtmltopdf)

There is a more "hands-on" example in the following section.

👋 Get Started/Example

To start, firstly import the library and nimib (of course...)

```
import nimipdf
import nimib
```

Initialize nimipdf (and nimib) using `nbInitPdf`.

```
nbInitPdf
```

Next, add your usual nimib code (nbCode, nbText, etc.)

Here, we'll use hello.nim:

> *The code block is too long, skip down to the next page...*

```
import strformat, strutils

nbText: """
  ## Secret talk with a computer
  Let me show you how to talk with the computer like a [real hacker](https://mango.pdf.zone/)
  and incidentally you might learn the basics of [nimib](https://github.com/pietroppeter/nimib).
  ### A secret message
  Inside this document is hidden a secret message. I will ask the computer to spit it out:
  """

let secret = [104, 101, 108, 108, 111, 44, 32, 119, 111, 114, 108, 100]

nbCode:
  echo secret

nbText: fmt"""
  what does this integer sequence mean?
  Am I supposed to [recognize it](https://oeis.org/search?q={secret.join("%2C+")}&language=english&go=Search)?

  ### A cryptoanalytic weapon
  Luckily I happen to have a [nim](https://nim-lang.org/) implementation of
  a recently declassified top-secret cryptoanalytic weapon:"""

nbCode:
  func decode(secret: openArray[int]): string =
    ## classified by NSA as <strong>TOP SECRET</strong>
    for c in secret:
      result.add char(c)

nbText: """
    ### The great revelation
    Now I can just apply it to my secret message and
    finally decrypt what the computer wants to tell me:"""

nbCode:
  let msg = decode secret
  echo msg  # what will it say?

nbText:
  fmt"_Hey_, there must be a bug somewhere, the message (`{msg}`) is not even addressed to me!"
```

Finally, save the PDF using `nbSavePdf`.

```
nbSavePdf
```

The generated pdf file can be found here
