html2pdf

An auxiliary tool to increase the efficiency of mental work

Agenda

- **html2pdf** description
- **pdf2knt** description
- External alternatives for the **html2pdf**
- Internal alternatives to the design solution
- The main challenges of the **html2pdf** development process
- Tools for prototyping of **html2pdf**
- **html2pdf** development process
- **html2pdf** technologies stack
- **html2pdf**: Minimum Viable Product
- **html2pdf**: Our team

html2pdf description

- A tool for converting web-based manuals and books to local PDF-file for enabling full-text search
- It's an auxiliary tool for rapid processing of large amounts of information
- It's an intellectual tool that can significantly reduce the time spent searching for information in web-books and web-manuals
- That is, it actually means converting the desired **website to a pdf book**, therefore the project should be renamed something like **ws2pb**, **web2pdf, WebPDFer** or **Web2pdfBk**.
- The complexity of implementing this project is comparable to developing a one window web browser with pdf-book-view. So development of it should be based on the WebKit web browser engine or Chromium web browser project

pdf2knt description

- The technological stack of **html2pdf** is an important part of the development of the pdf2knowlegeTree (**pdf2knt**) project as another more powerful intellectual support tool.
- **pdf2knt** is needed to build a tree of knowledge based on top of the one or more pdf-files.
- **pdf2knt** should automatically create a dictionary of available words and then on its basis to help build a hierarchy of concepts, their definitions, explanations, and examples of use.
- After building a knowledge tree, **pdf2knt** should allow you to find all the available information on a particular topic in a matter of seconds based on all the sources involved.

External alternatives for the **html2pdf**

It should be noted that today there are no open-source or well-known software analogs that would effectively solve the problem of converting a particular website with online documentation into a pdf-book.

Known analogs can only convert a single HTML file or a separate HTML page, with a specific URL address, in the corresponding pdf file.

This is predictable, as there are a number of technically complex tasks, without which it is impossible to implement such a project in practice.

External alternatives for the **html2pdf**

1	wkgtkprinter https://github.com/gnudles/wkgtkprinter HTML to PDF conversion library (with one simple function) using WebkitGtk (similar to wkhtmltopdf, but without the Qt part**).** A simple snippet (glue code) written in C to demonstrate the conversion process of HTML to PDF using Gtk+ and WebkitGtk. You can use this code to generate invoices from command line or from your software. you can use it from any programming language using libffi. Very minimalistic code, and easy to tweak, embed and understand. only one function does the conversion in synchronous way.

External alternatives for the **html2pdf**

2	**Doctron** https://github.com/lampnick/doctron Doctron is a Docker-powered,serverless,sample,fast,high quality document convert tool.Supply html convert to pdf(html2pdf), html convert to image(html2image like jpeg,png),which using chrome(Chromium) kernel, add watermarks to pdf, convert pdf to images etc. Features • Html convert to pdf/image using chrome kernel to guarantee what you see is what you get. • Easy deployment.(Using docker,kubernetes.) • Rich transformation parameters. • Customize page size from html convert to pdf or image. • Serverless supported.

External alternatives for the **html2pdf**

3	**Puppeteer** https://github.com/puppeteer/puppeteer Puppeteer is a Node library which provides a high-level API to control Chrome or Chromium over the DevTools Protocol. Puppeteer runs headless by default, but can be configured to run full (non-headless) Chrome or Chromium. Most things that you can do manually in the browser can be done using Puppeteer! Here are a few examples to get you started: • Generate screenshots and PDFs of pages. • Crawl a SPA (Single-Page Application) and generate pre-rendered content (i.e. "SSR" (Server-Side Rendering)). • Automate form submission, UI testing, keyboard input, etc. • Create an up-to-date, automated testing environment. Run your tests directly in the latest version of Chrome using the latest JavaScript and browser features. • Capture a timeline trace of your site to help diagnose performance issues. • Test Chrome Extensions.

Internal alternatives to the design solution

	An web-app that performs the necessary search on all documents to which the links from the table of contents points	
	pros	**cons**
1	The easiest way to programmatically implement the solution: • There is no need to download the all content of the website locally • There is no need to convert it to pdf-format with keeping the style of information formatting	It is an unacceptable long waiting for a search results, due to the need to perform a complete bypass of the entire tree of articles on the site for each search

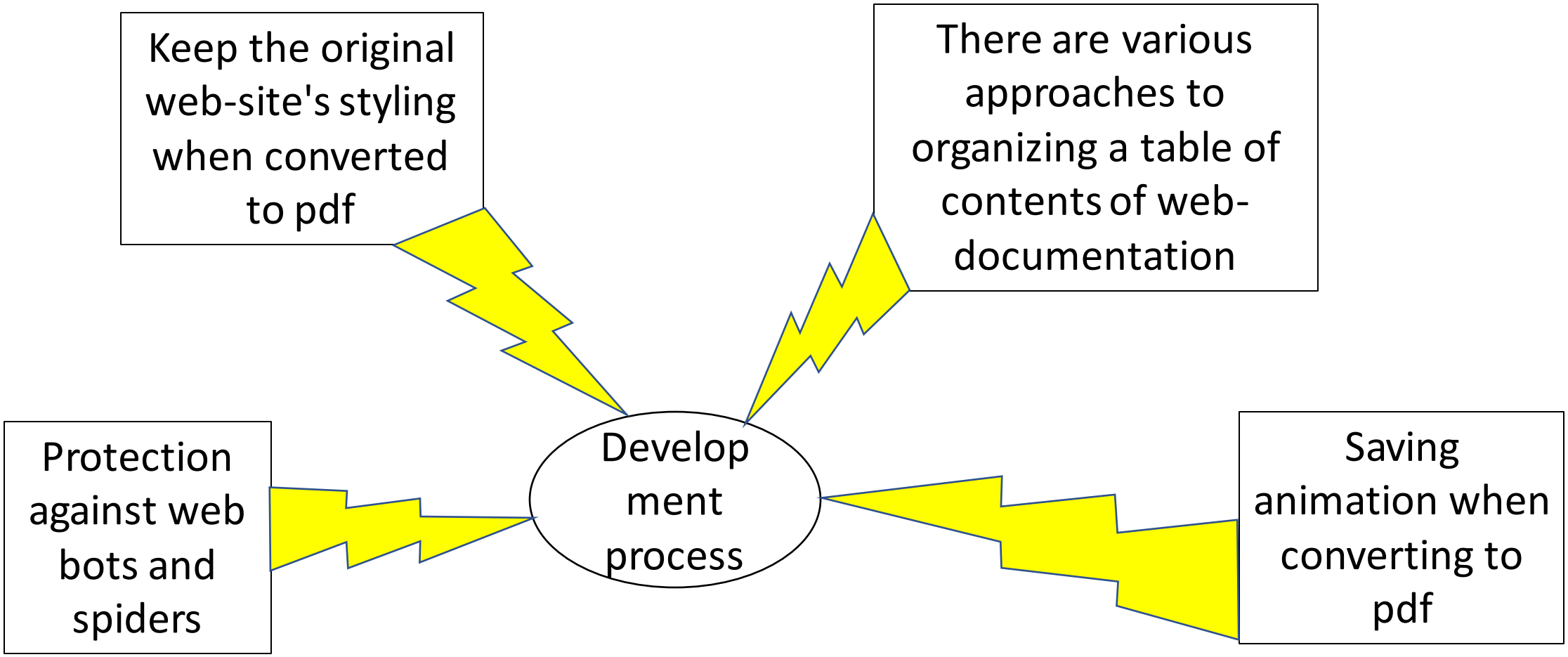
Internal alternatives to the design solution

2	An console application that convert any web-manual or web-book to the one page HTML file	
	pros	**cons**
	This is an easier way to programmatically implement the solution: • There is no need to convert web-site content to pdf-format with keeping the style of information formatting	Typically, a web-browser spends significantly more time to process a local HTML file than a PDF-viewer process a PDF file with a similar content.

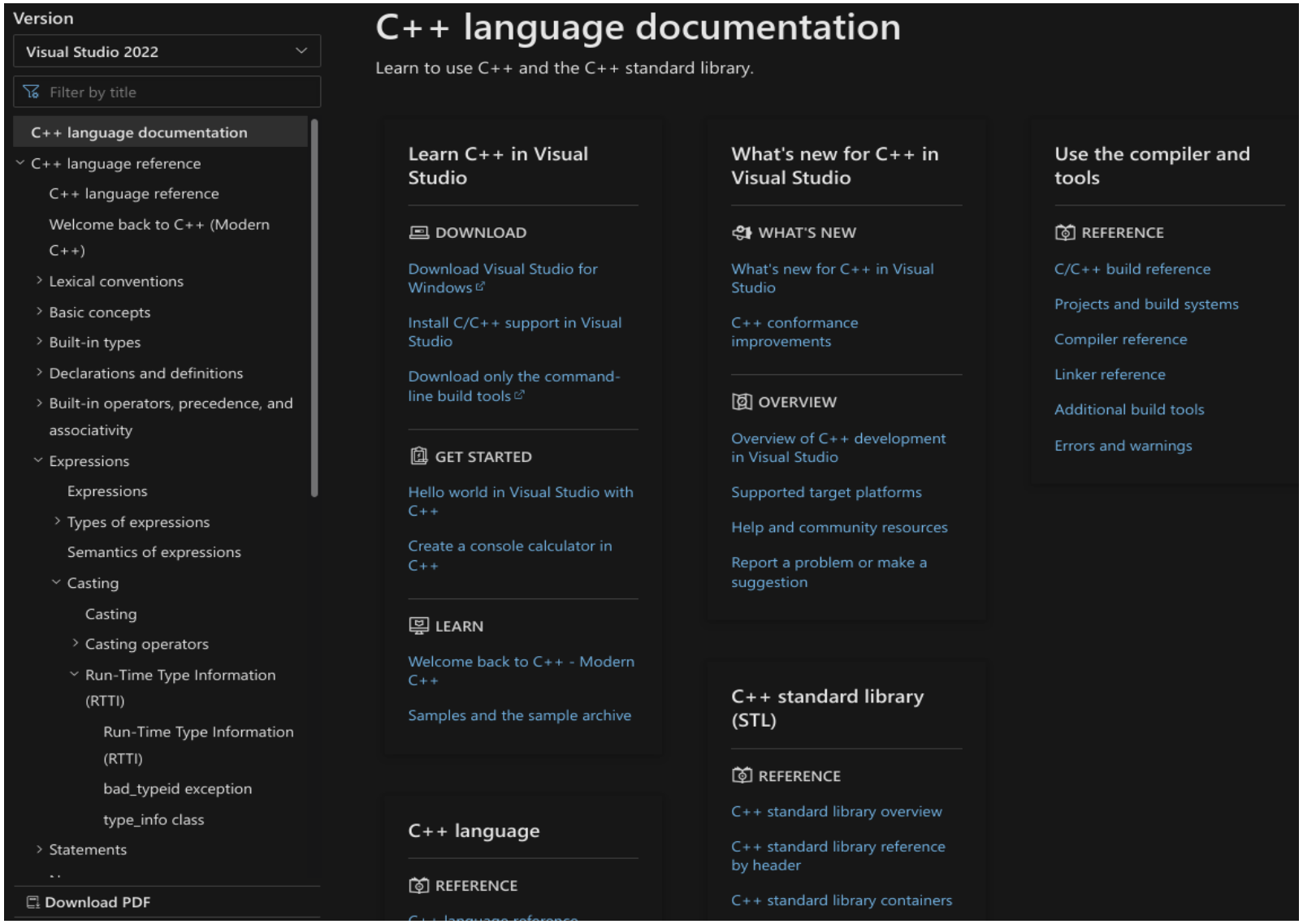
Internal alternatives to the design solution

3	html2pdf	
	pros	**cons**
	The most convenient and fastest way to process the content of a website is by converting it to a PDF file	This is the most complicated way to programmatically implement the solution: • Need to download all website content • Need to convert it to pdf-format with keeping the style of information formatting

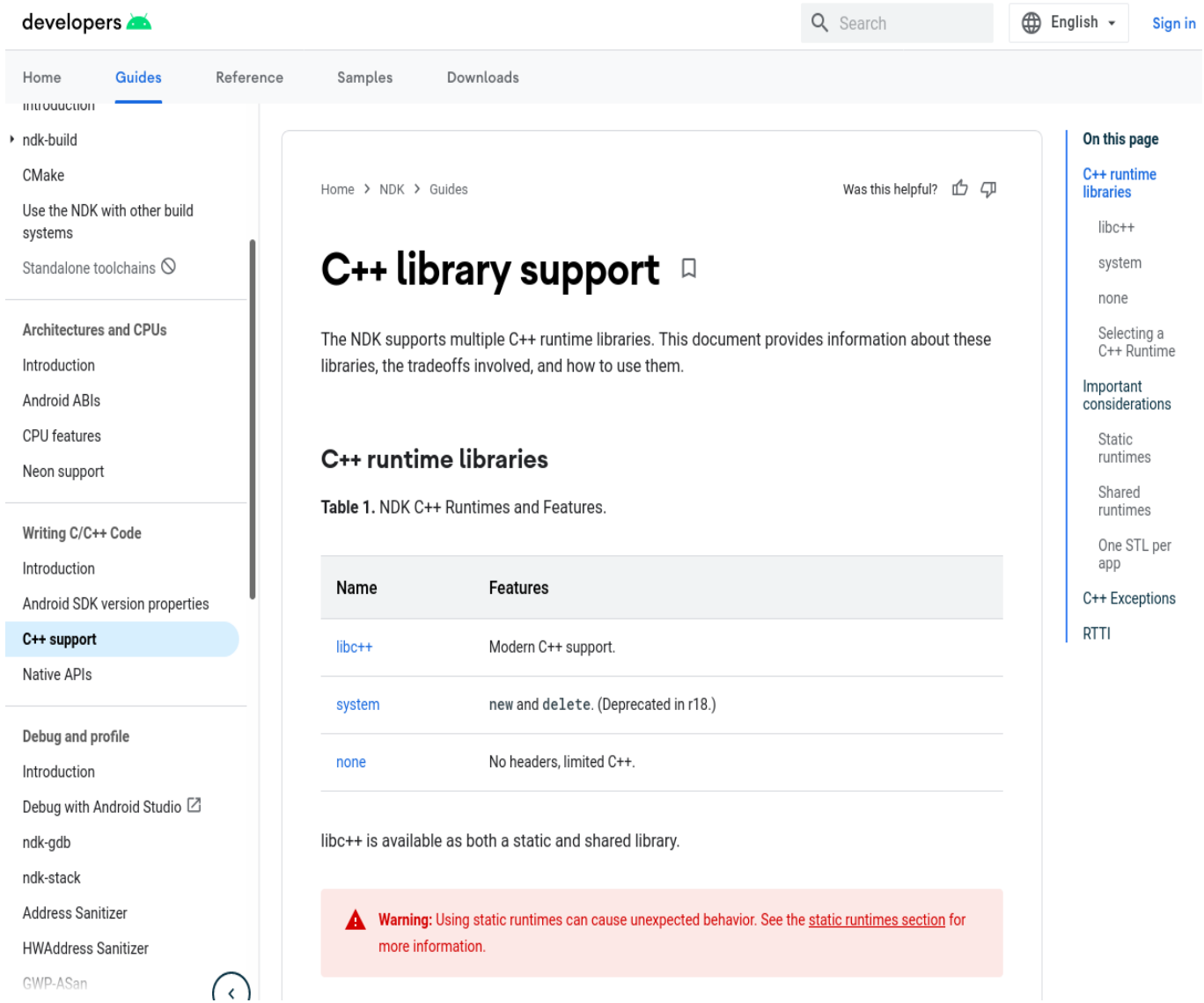
The main challenges of the **html2pdf** development process



Approaches to organizing a table of contents



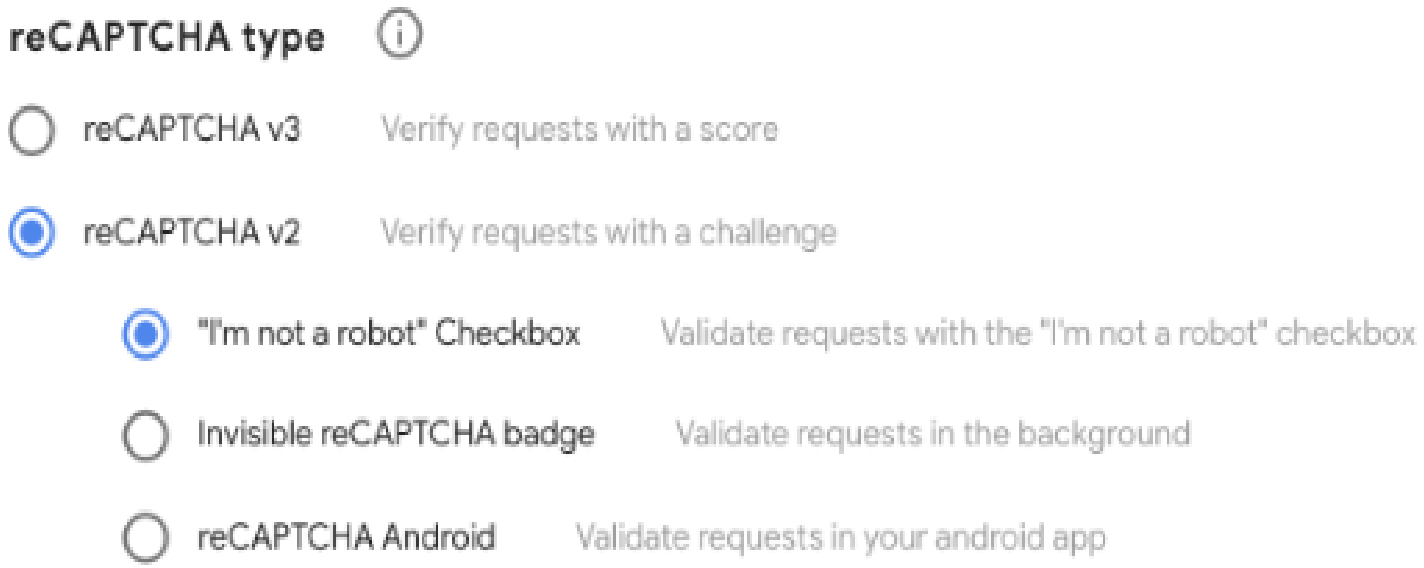
Approaches to organizing a table of contents



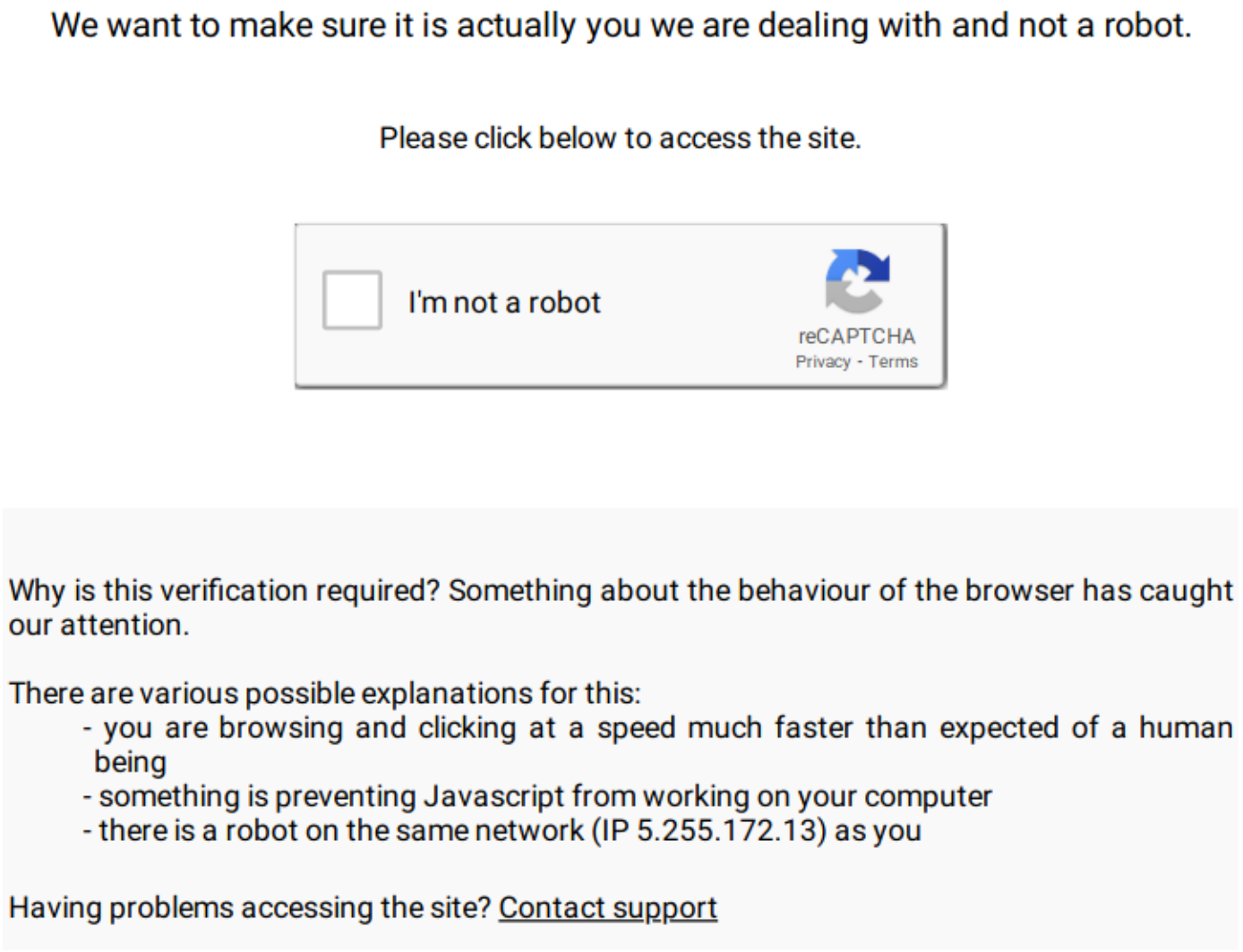
Protection against web bots and spiders

Choosing the type of reCAPTCHA

Choosing the type of reCAPTCHA There are four types of reCAPTCHA to choose from when creating a new site.



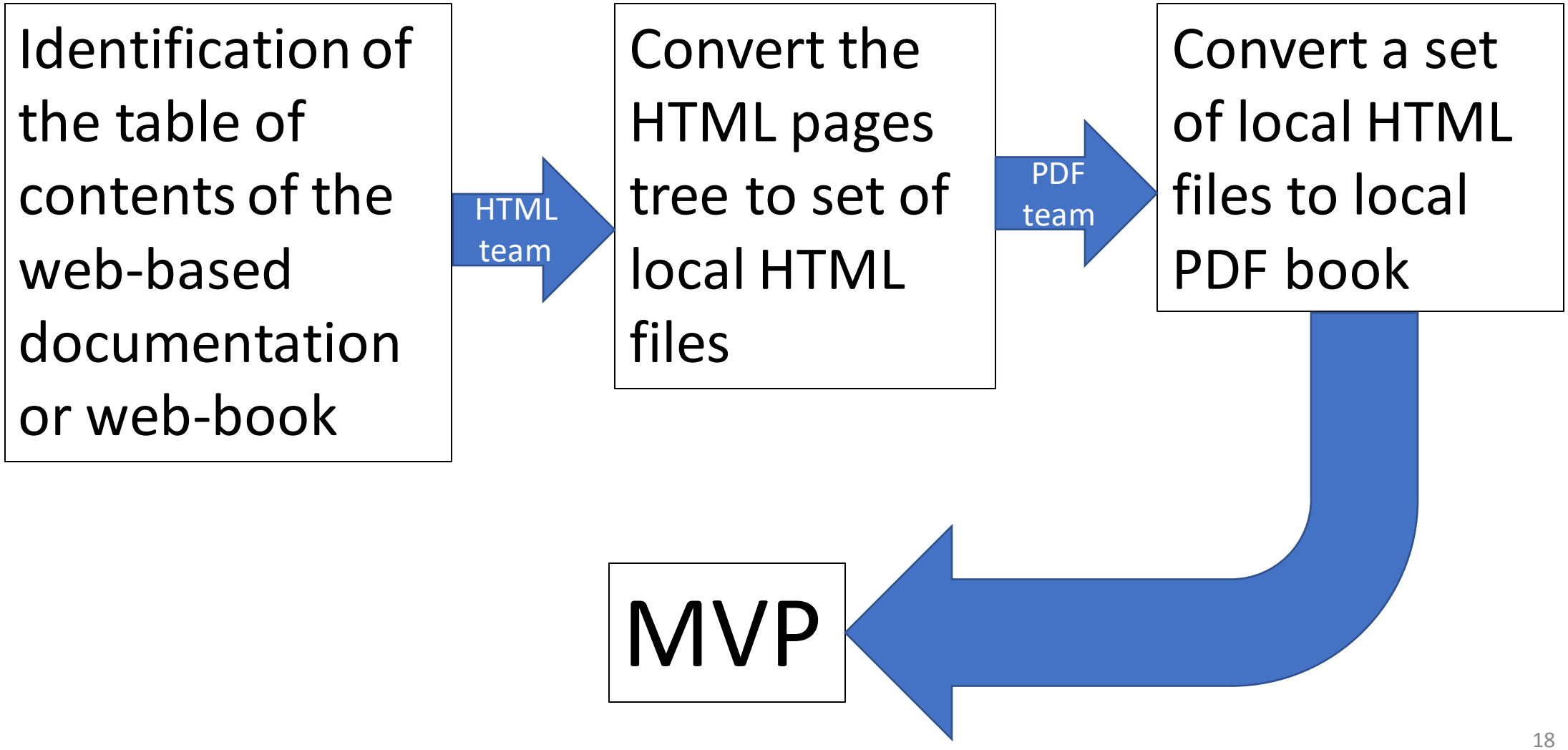
Protection against web bots and spiders



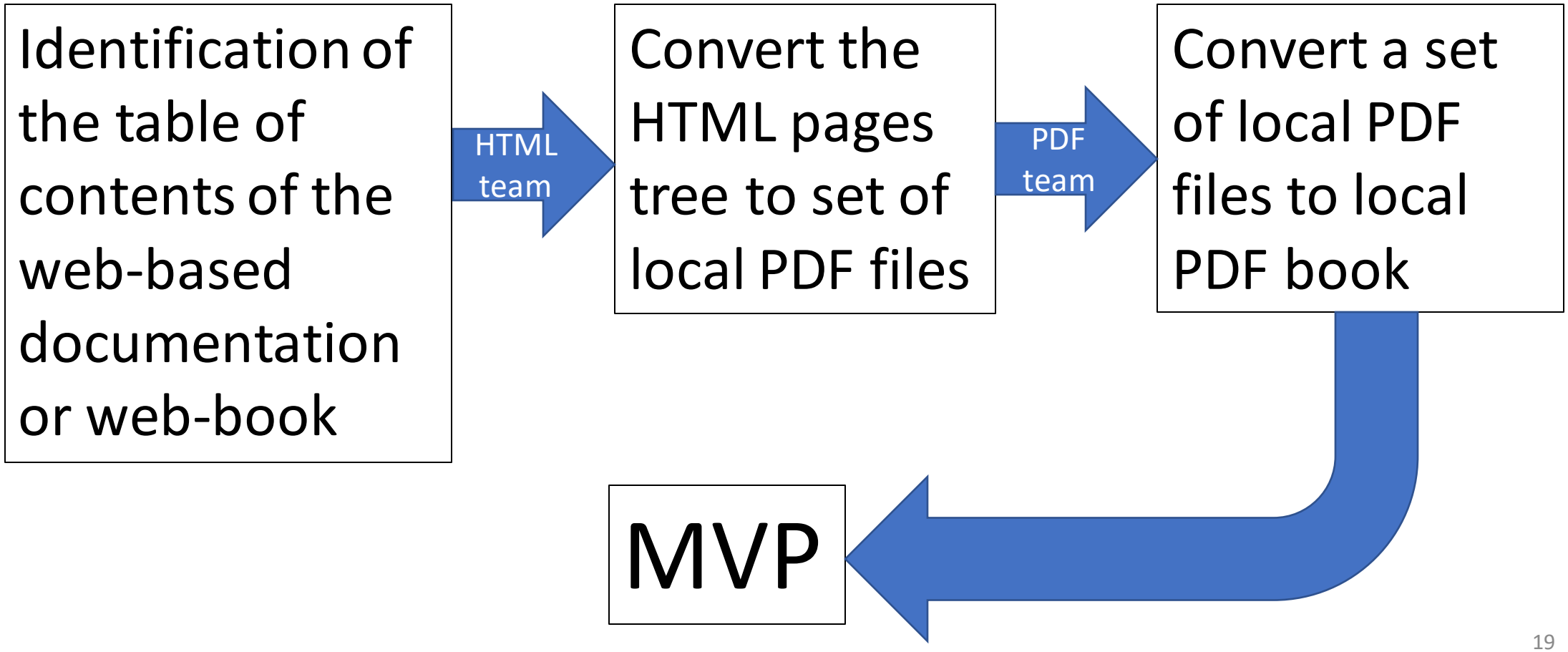
Tools for prototyping **html2pdf**

- Google-chrome in headless mode.
- Puppeteer. It's a Node library which provides a high-level API to control Chrome or Chromium over the DevTools Protocol. Puppeteer runs headless by default, but can be configured to run full (non-headless) Chrome or Chromium.

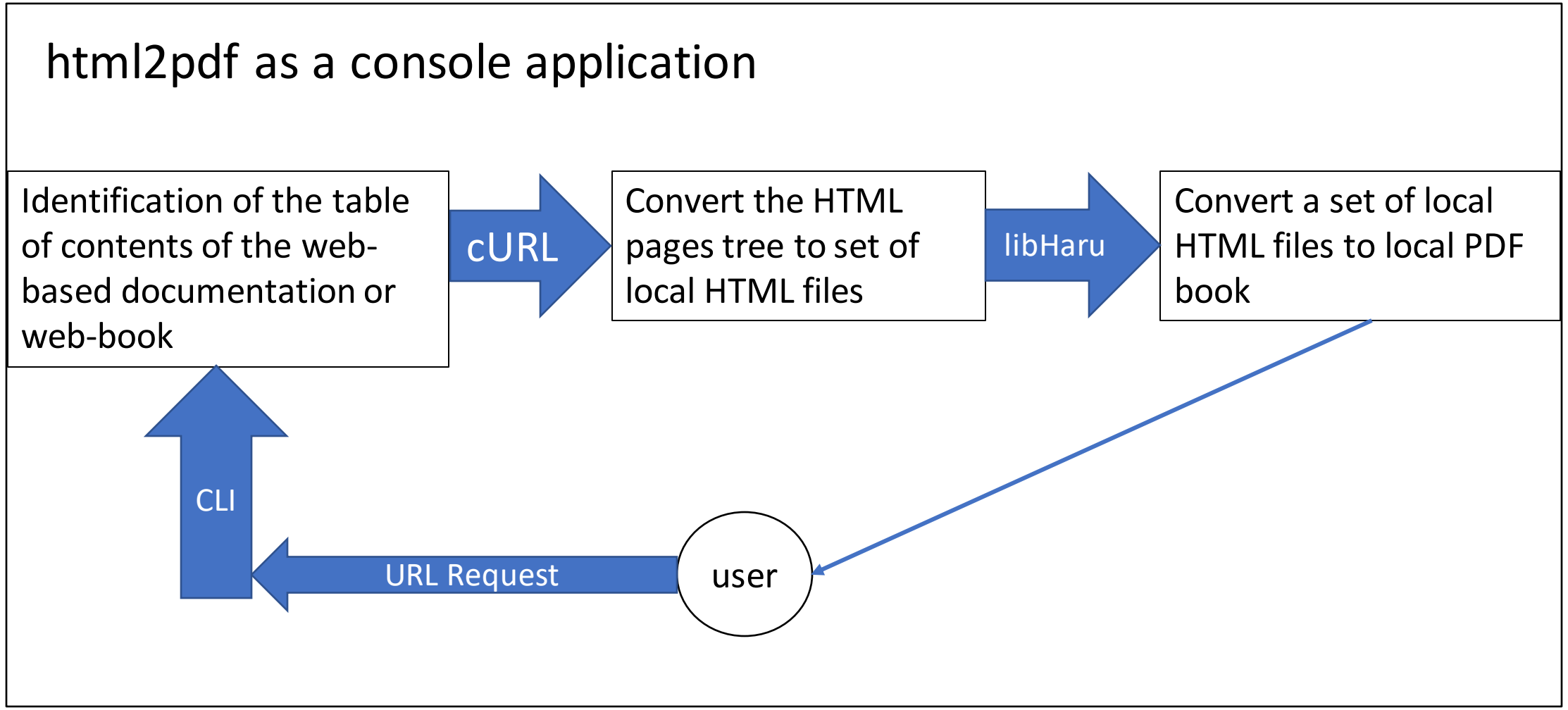
html2pdf development process (chosen path)



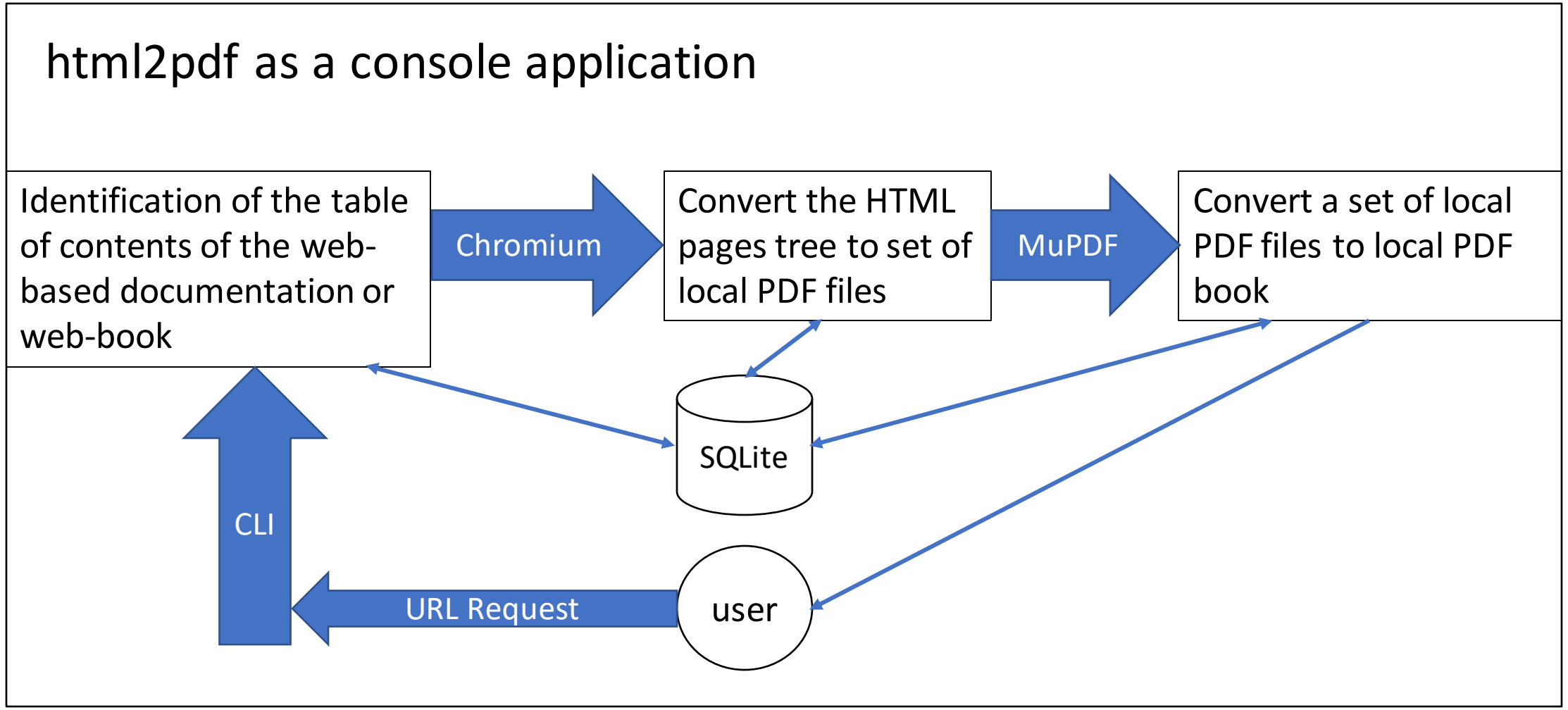
html2pdf development process based on Puppeteer project



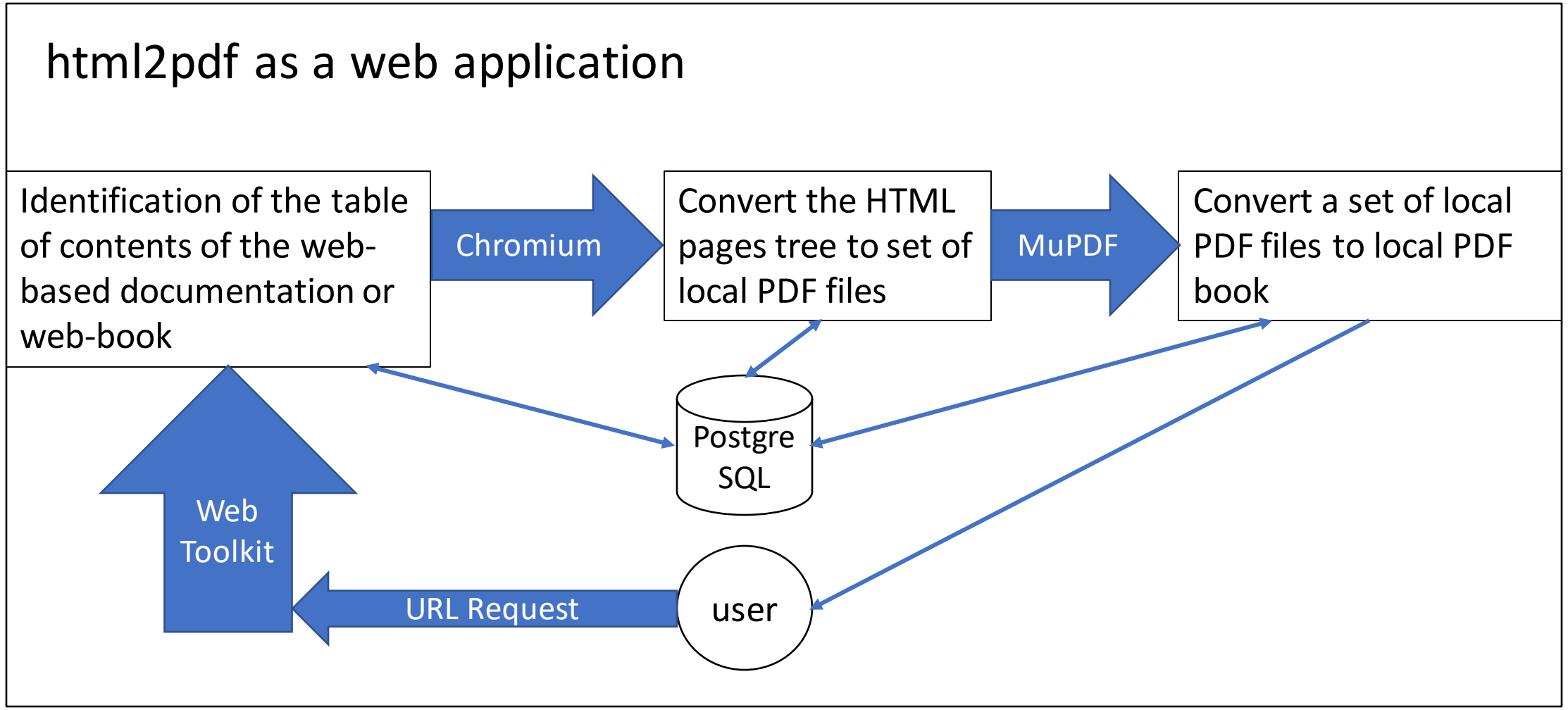
html2pdf technologies set (chosen path)



html2pdf technologies set (right way)



html2pdf technologies set



html2pdf Minimum Viable Product

- html2pdf is a console application
- html2pdf should get the table of contents from the https://www.learncpp.com/ home page and put it into ***learncpp.pdf***, where all positions of the content are links to the relevant pages of the site

html2pdf Minimum Viable Product2

- **html2pdf** is a console application
- **html2pdf** should get the table of contents from the https://www.learncpp.com/ home page and put it into ***learncpp.pdf***.
- **html2pdf** should take turns downloading HTML files from the website according to the links of the table of contents and put them into ***learncpp.pdf***, after the table of contents.
- All positions of the table of contents should be links to the relevant pages of the web-site inside the ***learncpp.pdf***.

html2pdf Minimum Viable Product3

- **html2pdf** is a console application
- **html2pdf** should get the table of contents from the https://www.learncpp.com/ home page and put it into ***learncpp.pdf***.
- **html2pdf** should take turns downloading HTML files from the website according to the links of the table of contents and put them into ***learncpp.pdf***, after the table of contents.
- All positions of the table of contents should be links to the relevant pages of the site inside ***learncpp.pdf***.
- **html2pdf** should save design style of the https://www.learncpp.com/ in resulting ***learncpp.pdf***.

html2pdf Minimum Viable Product4

- **html2pdf** is a console application
- **html2pdf** should get the table of contents from the any web site (without web scraping protection) home page and put it into ***web_domain.pdf***.
- **html2pdf** should take turns downloading HTML files from the website according to the links of the table of contents and put them into ***web_domain.pdf***, after the table of contents.
- All positions of the table of contents should be the links to the relevant pages of the site inside ***web_domain.pdf***.
- **html2pdf** should save the design style of any website in the resulting ***web_domain.pdf***.

html2pdf Minimum Viable Product5

- **html2pdf** is a web application.
- In the interactive mode, **html2pdf** should circumvent any web scraping protection.
- **html2pdf** should get the table of contents from the any web site home page and put it into ***web_domain.pdf***.
- **html2pdf** should take turns downloading HTML files from the website according to the links of the table of contents and put them into ***web_domain.pdf***, after the table of contents.
- All positions of the table of contents should be the links to the relevant pages of the site inside ***web_domain.pdf***.
- **html2pdf** should save the design style of any website in the resulting ***web_domain.pdf***.

html2pdf: Our team

21q3	
HTML sub-team	Artem Pidgornyi
HTML sub-team	Dmytro Lytvynenko
Mentor	Mykhailo Lohachov
Mentor	Oleksandr Yemets
PDF sub-team	Andrii Bulak
PDF sub-team	Dmytro Kovalenko
Scrum Master	Vitalii Kaplenko
22q1	