Software Project PDF Generator

A JavaScript tool to generate beautiful PDF documents for your projects.

Built on:

HTML CSS JS chrome

React GitHub Sketch Weasy Print

Software Project PDF Generator

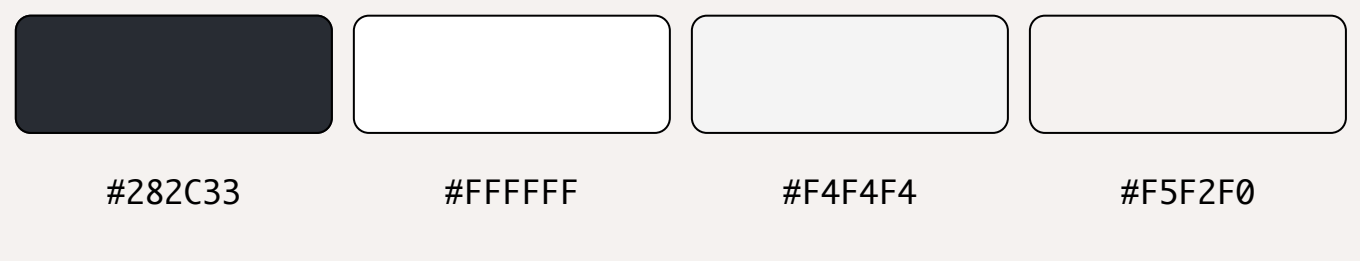
1. Introduction

This tool was created to enable developers to showcase their projects and ideas to others. The user can enter information about their project into the tool, and then export the information in a standardised PDF format that can be easily shared with others. It allows developers to easily showcase their work, so that they can focus on what they love - producing great software.

This very document was created using the tool, and is intended to act as an example of the tool's funtionality.

Body colors

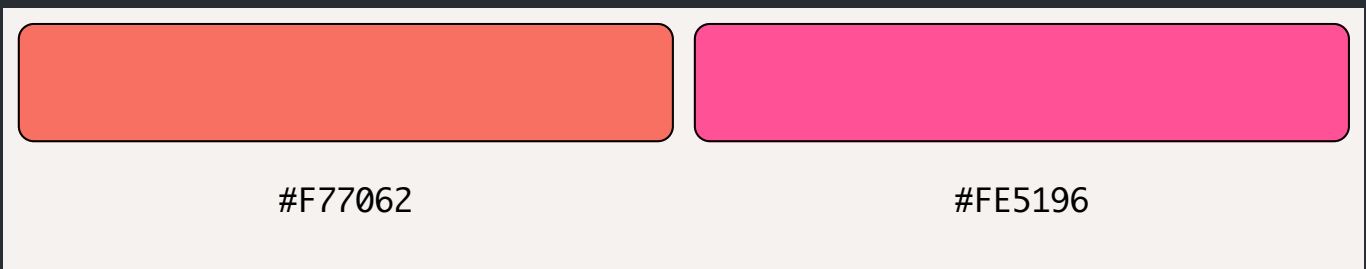
The colors used throughout the body of the PDF are simple and high contrasting in order to keep attention focused on what matters most - the content.



Yup, pretty simple - just shades of white and black!

***Title page* and *final page* colors**

The *title page* and the *final page* were given a little more color in order to catch the eye.



2. Approach

Key tools

1. HTML5 - used to structure the document.
2. CSS3 - used to style the document.
3. JavaScript - used to implement the tool's logic and receive input from the user.
4. Headless Chrome - used to execute the JavaScript and convert the output to HTML before reaching WeasyPrint, as WeasyPrint does not handle JavaScript.
5. WeasyPrint - used to convert the final HTML and CSS into a PDF document.

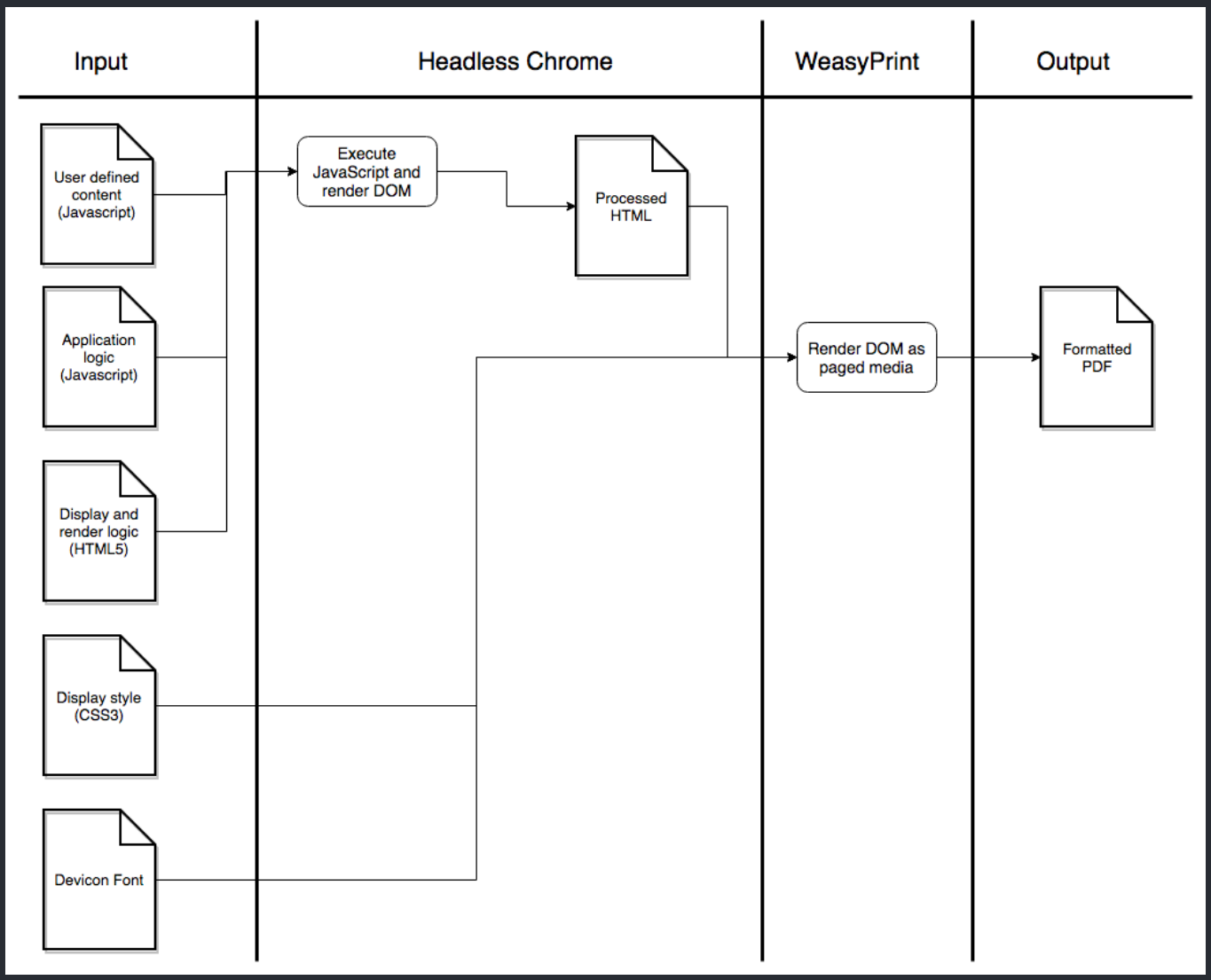


Figure 1 - Dataflow diagram

3. Usage

To use the tool, first open `content.js`. Then, use the `createToolDisplay` function at the top of the page to list the tools used to build your app. The Devicon package has been used to provide the logos, which is great because almost every tool you'll use for development is covered within this package.

To add HTML5, CSS3, and JavaScript, for example, you'll use:

```JavaScript
createToolDisplay([
  'devicon-html5-plain-wordmark',
  'devicon-css3-plain-wordmark',
  'devicon-javascript-plain',
]);
```

Next, we want to start adding content to our document. We do this by creating sections, using the `Section` class:

```JavaScript
const exampleSection = new Section("Example Section");
```

A new H1 header is added when a new section is created.

We can then begin adding content to our new section by using the following methods:

1. addParagraph()
2. addImage()
3. addImageGrid()
4. addH2Header()
5. addOrderedList()
6. addUnorderedList
7. addCodeBlock()
8. addColorPallete()

The image grid is useful to show multiple, related images at once, for example:

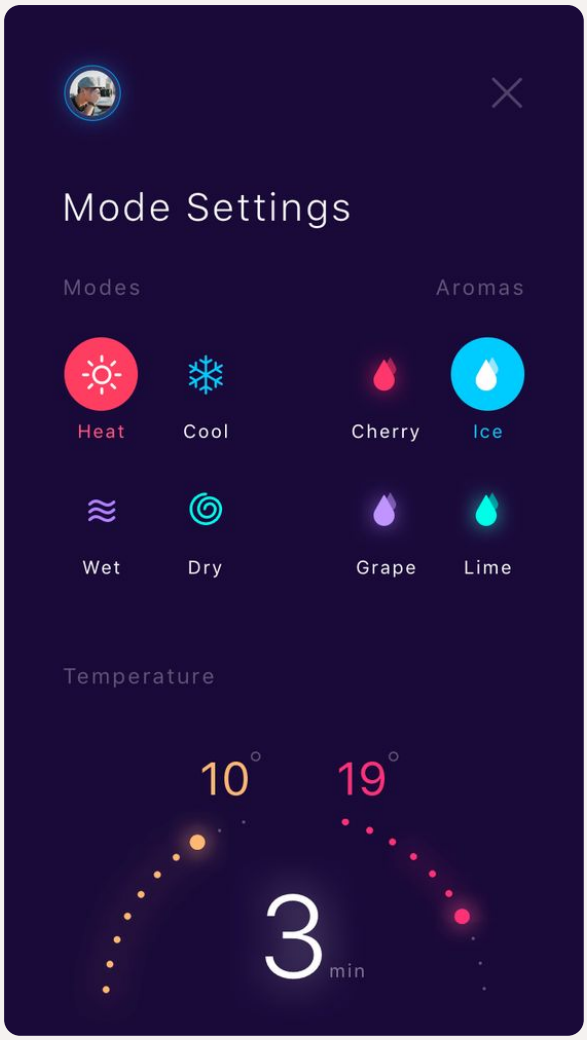


Figure 4 - UI Design

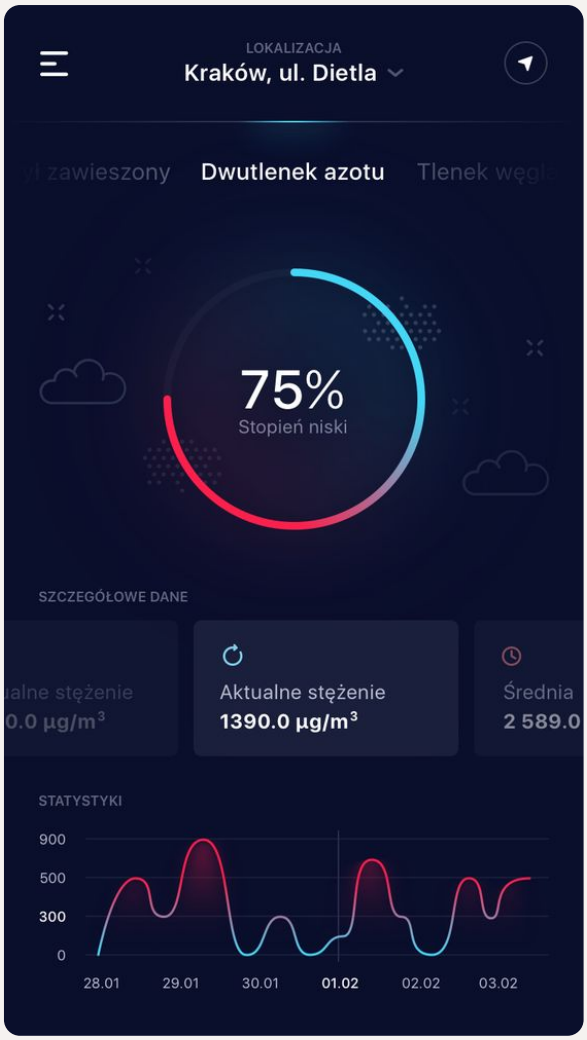


Figure 5 - UI Design

Matt Doyle

M: 0407 417 405
E: hello@matt.fyi