SUSCO (Source Code)

A library created by Joshua Sosso to auto generate a pdf from source code.

Files Included:

- package.json
- README.md
- src/files.ts
- src/logger.ts
- src/main.ts
- src/assets/template.html
- test_files/hello.js
- test_files/hello.py
- test_files/somemodule/hello.rs
- test_files/somemodule/hello_again.rs

---

package.json

```
{
  "name": "susco",
  "version": "0.1.2",
  "description": "Automatically generate a PDF from source code",
  "main": "dist/main.js",
  "files": [
    "dist/**/*"
  ],
  "keywords": [
    "pdf",
    "copyright"
  ],
  "repository": "https://github.com/modiimedia/susco",
  "scripts": {
    "prepare": "npm run build",
    "build": "npm run copyPrism && npm run cleanDist && tsc && npm run copyAssets",
    "cleanDist": "ts-node scripts/cleanDir --project tsconfig.build.json",
    "copyAssets": "ts-node scripts/copyAssets --project tsconfig.build.json",
    "copyPrism": "ts-node scripts/copyPrism --project tsconfig.build.json"
  },
  "author": "Joshua Sosso (joshmossas)",
  "license": "MIT",
  "dependencies": {
    "cli-progress": "^3.11.1",
    "fast-glob": "^3.2.11",
    "fs-extra": "^10.1.0",
    "get-installed-path": "^4.0.8",
    "istextorbinary": "^6.0.0",
    "pdf-merger-js": "^3.4.0",
    "prismjs": "^1.28.0",
    "wkhtmltopdf": "^0.4.0"
  },
  "devDependencies": {
    "@types/cli-progress": "^3.11.0",
    "@types/fs-extra": "^9.0.13",
    "@types/get-installed-path": "^4.0.1",
    "@types/prismjs": "^1.26.0",
    "@types/wkhtmltopdf": "^0.3.4",
    "ts-node": "^10.8.1",
    "typescript": "^4.7.4"
  }
}
```

README.md

```
# SUSCO - Generate a PDF from Source Code

A NodeJS tool made to please the morons at the US Copyright Office. It that will take every text file in a directory and output

See `/example` to see an example PDF generated from this repo using an example script.

## Why SUSCO?

If you've ever tried to submit source code to the US Copyright Office you may have recieved a response like this:

> "It looks like you uploaded a runnable copy of your program. We need the textual source code to register the computer program.

Even though what you submitted IS IN FACT the textual source code. When pressing further you will discover that they want you to

From the copyright.gov website ([https://www.copyright.gov/circs/circ61.pdf](https://www.copyright.gov/circs/circ61.pdf))

> You can upload the source code to the electronic registration system, preferably as a PDF file or other file type accepted by

For this reason, SUSCO (aka "Stupid United States Copyright Office") was created to automate this process.

## Installation

This package requires that you have [wkhtmltopdf](https://wkhtmltopdf.org/) installed on your machine. Ensure that you have it p
```

```
npm install susco
```

Usage

```ts
// both require.js and es6 imports are supported
import { generatePdf } from "susco";
const { generatePdf } = require("susco");

const config = defineConfig();

generatePdf({
  heading: "Some title",
  description: "Some description",
  // accepts an array of glob patterns
  include: ["src/**/*", "scripts/**/*"],
  // accepts an array of glob patterns
  ignore: ["node_modules", "dist", "some-other-lib/**/*.ts"],
  output: "<output-file>",
  disableLogs: false, // false by default
});
```

---

src/files.ts

```ts
import { createWriteStream, readFile } from "fs-extra";
import path from "path";
import htmlToPdf from "wkhtmltopdf";
import Prism from "prismjs";
const loadLanguages = require("prismjs/components/");

loadLanguages([
  "markup",
  "html",
  "xml",
  "svg",
  "rss",
  "css",
  "js",
  "actionscript",
  "ada",
  "bash",
  "shell",
  "c",
  "cpp",
  "csharp",
  "cs",
  "dotnet",
  "cmake",
  "elixir",
  "dart",
  "elm",
  "csv",
  "go",
  "gdscript",
  "git",
  "graphql",
  "haxe",
  "ignore",
  "gitignore",
  "java",
  "json",
  "kotlin",
  "kt",
  "lisp",
  "llvm",
  "lua",
  "nginx",
  "php",
  "perl",
  "powershell",
  "pug",
  "python",
  "py",
  "jsx",
  "tsx",
  "rust",
  "ruby",
  "rb",
  "sass",
  "scss",
  "scala",
  "sql",
  "swift",
  "toml",
  "yaml",
  "typescript",
  "ts",
  "tsconfig",
  "zig",
]);

const replaceHtmlCharacters = (str: string) => {
  let result = str
    .replace(/\&/g, "&amp;")
    .replace(/\</g, "&lt;")
```

```
    .replace(/\>/g, "&gt;");
  return result;
};

const getHtml = async (
  heading: string,
  description: string,
  listOfFiles: string,
  body: string
) => {
  const html = (
    await readFile(path.resolve(__dirname, "./assets/template.html"))
  ).toString();
  const css = (
    await readFile(path.resolve(__dirname, "./assets/prism.css.copy"))
  ).toString();
  return html
    .replace("{{prismcss}}", css)
    .replace("{{heading}}", heading)
    .replace("{{description_content}}", description)
    .replace("{{listoffiles}}", listOfFiles)
    .replace("{{body}}", body);
};

const getPrismGrammar = (file: string) => {
  const ext = file.split(".").pop() ?? "";
  if (ext === "rs") {
    return Prism.languages.rust;
  }
  const grammar = Prism.languages[ext] ?? Prism.languages.markup;
  return grammar;
};

const getPrismLanguage = (file: string) => {
  const ext = file.split(".").pop() ?? "";
  if (ext === "rs") {
    return "rust";
  }
  return ext;
};

export const convertToHtml = async (file: string) => {
  const text = (await readFile(file)).toString();
  const highlightedText = Prism.highlight(
    text,
    getPrismGrammar(file),
    getPrismLanguage(file)
  );
  const html = `
  <div class="code-section">
    <h2>${file}</h2>
    <div class="code-block">
      <pre><code>${highlightedText}</code></pre>
    </div>
  </div>`;
  return html;
};

export const convertToPdf = async (
  heading: string,
  description: string,
  listOfFiles: string[],
  htmlBlocks: string[],
  output: string
): Promise<void> => {
  const html = await getHtml(
    heading,
    description,
    listOfFiles.map((file) => `<div>- ${file}</div>`).join("\n"),
    htmlBlocks.join("\n")
  );
  return new Promise((resolve, reject) => {
    const writeStream = createWriteStream(output);
    htmlToPdf(html, {
      footerFontSize: 8,
      footerSpacing: 8,
      runScript: [],
      footerLeft: `This PDF was generated on ${new Date()} using https://github.com/modiimedia/susco`,
    }).pipe(writeStream);
    writeStream.on("finish", () => resolve());
    writeStream.on("error", (err) => reject(err));
  });
};
```

---

src/logger.ts

```
class Logger {
  logEnabled: boolean;

  constructor(enabled: boolean) {
    this.logEnabled = enabled;
  }

  log(msg: any) {
    if (this.logEnabled) {
      console.log(msg);
    }
  }
```

```
  error(msg: any) {
    if (this.logEnabled) {
      console.error(msg);
    }
  }
}

export default Logger;
```

---

src/main.ts

```
import glob from "fast-glob";
import { isText } from "istextorbinary";
import CliProgress from "cli-progress";
import { convertToHtml, convertToPdf } from "./files";
import Logger from "./logger";

const removeLeadingAndTrailingSlashes = (string: string): string =>
  string.trim().replace(/^\/+|\/+$/g, "");

const getFilePaths = async (includes: string[], ignore: string[]) => {
  const files = await glob(includes, {
    ignore,
  });
  return files.filter((file) => isText(file));
};

export interface Config {
  heading: string;
  description?: string;
  include: string[];
  ignore: string[];
  output: string;
  disableLogs?: boolean;
}

export const defineConfig = (config: Config) => config;

export const generatePdf = async (config: Config) => {
  const logger = new Logger(!config.disableLogs);
  const progressBar = new CliProgress.SingleBar(
    {},
    CliProgress.Presets.shades_classic
  );
  const files = await getFilePaths(config.include, config.ignore);
  logger.log(`\nprocessing ${files.length} files`);
  progressBar.start(files.length, 0);
  const htmlBlocks: string[] = [];
  for (let i = 0; i < files.length; i++) {
    const file = files[i];
    const html = await convertToHtml(file);
    htmlBlocks.push(html);
    progressBar.update(i + 1);
  }
  progressBar.stop();
  logger.log(`\nmerging files into PDF...`);
  await convertToPdf(
    config.heading,
    config.description || "",
    files,
    htmlBlocks,
    config.output
  );
  logger.log(`${config.output} created`);
};
```

---

src/assets/template.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>PDF Template Base</title>
    <style>
      h1 {
        font-size: 40px;
        font-weight: bold;
      }
      h2 {
        font-size: 26px;
      }
      body {
        font-size: 18px;
        font-family: Arial, Helvetica, sans-serif;
      }
      code {
        display: block;
      }
```

```
      .code-section {
        padding-top: 20px;
        padding-bottom: 20px;
        border-bottom: 1px solid #a1a1a1;
      }
      .code-block {
        background: #f3f3f3;
        padding: 20px;
        border-radius: 4px;
        display: block;
      }
    </style>
    <style>
      {{prismcss}}
    </style>
  </head>
  <body>
    <div class="code-section" style="padding-top: 0px">
      <h1>{{heading}}</h1>
      <div>{{description_content}}</div>
    </div>
    <div class="code-section">
      <h2>Files Included:</h2>
      <div>{{listoffiles}}</div>
    </div>
    <div>{{body}}</div>
  </body>
</html>
```

---

test_files/hello.js

```
console.log("hello world");

const sayHello = () => {
  console.log("hello world");
};

const saySomething = (input) => {
  console.log(input);
};

sayHello();
saySomething("Hello Again!");
```

---

test_files/hello.py

```
print("hello world")

def sayHello():
    print("Hello World!")

def saySomething(str):
    print(str)

sayHello()

saySomething("Hello again")
```

---

test_files/somemodule/hello.rs

```
fn main() {
    println!("Hello world")
}
```

---

test_files/somemodule/hello_again.rs

```
fn main() {
    println!("Hello Again")
}
```
