```
#!/bin/node
 //
//    Opens a file or a folder and creates the preetyfied code
//
//    This program is free software: you can redistribute it and/or modify
//    it under the terms of the GNU General Public License as published by
//    the Free Software Foundation, either version 3 of the License, or
//    (at your option) any later version.
//
//    This program is distributed in the hope that it will be useful,
//    but WITHOUT ANY WARRANTY; without even the implied warranty of
//    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
//    GNU General Public License for more details.
//
//    You should have received a copy of the GNU General Public License
//    along with this program.  If not, see <http://www.gnu.org/licenses/>.
const VERSION = "1.0.0";

const child_process = require("child_process");
const detect = require("language-detect");
const fs = require("fs");
const hljs = require("highlight.js");
const markdown = require("markdown-it");
const path = require("path");
const pdf = require("html-pdf");

let args = require('optimist').argv;
let help = `Usage: preetycode [-h | -v | -s style | -e encoding | -l language | -a message | -o output_folder] FILES
Converts code into a preety formated pdf with syntax highlight

Arguments:
  -h, --help        Show this help.
  -v, --version     Shows the software version.
  -s, --style       Choses the stylesheet to use (list below). Default: github.
  -e, --encoding    Selects the encoding. Default: UTF-8.
  -l, --language    Selects the language to apply highlight. Default: Autodetect.
  -a, --add-header  Adds a message to the header of the file. Eg: version number. You can use HTML.
  -f, --add-footer  Adds a message to the footer of the file. Eg: version number. You can use HTML.
  -o, --output      Chooses the output folder. Default: .

Available Stylesheets:
`;
let cssFiles = fs.readdirSync(`${__dirname}/node_modules/highlight.js/styles`);
cssFiles.filter((file => file.endsWith(".css"))).map(file => {
  help += `    *${file.substr(0, file.length-4)}\n`;
});

if ((args.h) || (args.help) || args._.length === 0) {
  console.log(help);
  process.exit(0);
} else if ((args.v) || (args.version)) {
  console.log(VERSION);
  process.exit(0);
}

let style = args.s || args.style || "github";
let encoding = args.e || args.encoding || "utf-8";
let language = args.l || args.language || "";
let header = args.a || args["add-header"] || "";
let footer = args.f || args["add-footer"] || "";
let output = args.o || args.output || ".";

if (!fs.existsSync(output)) {
  child_process.execSync(`mkdir -p ${output}`);
}

var md = markdown({
  html: false,
  xhtmlOut: false,
  breaks: false,
  langPrefix: 'language-',
  linkify: false,
  typographer: false,
  quotes: '“”‘’',
  highlight: function(str, lang) {
    if (lang && hljs.getLanguage(lang)) {
```

```
      try {
        return `<pre class="hljs"><code>${hljs.highlight(lang, str, true).value}</code></pre>`;
      } catch (__) {}
    }
    return `<pre class="hljs"><code>${hljs.highlightAuto(str).value}</code></pre>`;
  }
});

args._.map(file => {
  let fileName = path.basename(file);
  fs.readFile(file, encoding, (err, data) => {
    let fileLanguage = language;
    if (language == "") {
      try {
        language = detect.sync(`${__dirname}/${file}`);
      } catch (error) {
        // console.log(error);
      }
    }
    let code = md.render(`\`\`\`${language}\n${data}\n\`\`\`\n\n`);
    let html = `
    <head>
    <link rel="stylesheet" type="text/css" href="file://${__dirname}/node_modules/highlight.js/styles/${style}.css">
    </head>
    <style>
      body, pre, code {
        font-size: 7px;
        overflow: hidden !important;
      }
    </style>
    <body>
    ${code.trim()}
    </body>`;
    let options = {
      "phantomPath": `${__dirname}/node_modules/phantomjs-prebuilt/bin/phantomjs`,
      "phantomArgs": [],
      "timeout": 30000,
      "border": {
        "top": "1in", // default is 0, units: mm, cm, in, px
        "right": "0.5in",
        "bottom": "1in",
        "left": "0.5in"
      },
    };
    if (header !== "") {
      options.border.top = 0;
      options.header = {
        "height": "1in",
        "contents": `<br/>${header}`
      };
    }
    if (footer !== "") {
      options.border.bottom = 0;
      options.footer = {
        "height": "1in",
        "contents": `${footer}`
      };
    }
    pdf.create(html, options).toFile(`${output}/${fileName}.pdf`, function(err, res) {
      if (err) return console.log(err);
    });
  });
});
```