index.js

```
const path = require('path');
const puppeteer = require('puppeteer');
const readline = require('readline');

const {
  newFileName,
  extractExtension,
  makeHtml,
  validExtensions,
} = require('./utils');

const pageStyles = `body {font-family: sans-serif}`;

const rl = readline.createInterface({
  input: process.stdin,
});

let outputFile = newFileName();
if (process.argv[2]) {
  outputFile = process.argv[2];
}

const waitForAll = [];
rl.on('line', function (l) {
  console.error('file: ', l);
  if (!validExtensions.includes(extractExtension(l))) return;
  waitForAll.push(makeHtml(l));
});

rl.on('close', async () => {
  const content = (await Promise.all(waitForAll)).join('\n');
  const browser = await puppeteer.launch({headless: outputFile !== 'preview'});
  const page = await browser.newPage();
  await page.setContent(content);
  const prismPath = path.dirname(require.resolve('prismjs'));
  await page.addStyleTag({path: path.join(prismPath, 'themes/prism.css')});
  await page.addStyleTag({content: pageStyles});
  await page.addStyleTag({
    path: path.join(prismPath, 'plugins/line-numbers/prism-line-numbers.css'),
  });
  if (outputFile === 'preview') return;
  await page.pdf({path: outputFile, format: 'a4', scale: 0.5});
  await browser.close();
});
```

---

utils.js

```
const {readFile} = require('fs/promises');
const Prism = require('prismjs');

const newFileName = () => {
  const d = new Date();
  const adjustedISO = new Date(d.getTime() - d.getTimezoneOffset() * 60 * 1000)
    .toISOString()
    .substr(0, 19)
    .replace('T', ' ');
  return `code2pdf_${adjustedISO}.pdf`;
};

const extractExtension = (path) => {
  const fileExtMatch = path.match(/\.([^.]+)/);
  if (!fileExtMatch) return;
  return fileExtMatch[1];
};

const validExtensions = ['js', 'c', 'cpp', 'cs', 'css', 'html', 'xml', 'svg'];

const getLanguage = (path) => {
  const extension = extractExtension(path);
  let language = 'none';
  switch (extension) {
    case 'js':
      language = 'javascript';
      break;
    case 'c':
    case 'cpp':
    case 'cs':
      language = 'clike';
      break;
    case 'css':
      language = 'css';
      break;
    case 'html':
    case 'xml':
    case 'svg':
      language = 'markup';
  }
  return language;
```

```
};

const makeHtml = async (path) => {
  const lang = getLanguage(path);
  const code = await readFile(path, 'utf8');
  // Static generation of line numbers
  // https://stackoverflow.com/a/59577306/1615721
  // https://github.com/PrismJS/prism/blob/master/plugins/line-numbers/prism-line-numbers.js#L109
  const NEW_LINE_EXP = /\n(?!$)/g;
  let lineNumbersWrapper;

  Prism.hooks.add('after-tokenize', function (env) {
    const match = env.code.match(NEW_LINE_EXP);
    const linesNum = match ? match.length + 1 : 1;
    const lines = new Array(linesNum + 1).join('<span></span>');
    lineNumbersWrapper = `<span aria-hidden="true" class="line-numbers-rows">${lines}</span>`;
  });

  let formatted = Prism.highlight(code, Prism.languages[lang], lang);
  if (lineNumbersWrapper) formatted += lineNumbersWrapper;

  return `<h2>${path}</h2><pre class='line-numbers language-${lang}'><code class='language-${lang}'>${formatted}</code></pre><hr />`;
};

exports.extractExtension = extractExtension;
exports.validExtensions = validExtensions;
exports.newFileName = newFileName;
exports.makeHtml = makeHtml;
```