Title – Date

FILE: copyDirectoryCLI (to text).js

```
#!/usr/bin/env node

const fs = require("fs");
const path = require("path");
require("dotenv").config();

const rootDir = path.resolve(process.env.ROOT_DIR || "./");
const excludeDirs = (
  process.env.EXCLUDE_DIRS || "node_modules,.git,dist"
).split(",");
const allowedExtensions = (process.env.ALLOWED_EXTENSIONS || ".js,
  ","
);
const maxFileSize = parseInt(process.env.MAX_FILE_SIZE_MB || "1")
const outputPrefix = process.env.OUTPUT_PREFIX || "output";

let outputFileIndex = 1;
let currentSize = 0;
let writeStream = null;

function getOutputFilePath() {
  return path.join(process.cwd(), `${outputPrefix}-${outputFileInd
}

function openNewWriteStream() {
  if (writeStream) {
    writeStream.end();
  }
  const filePath = getOutputFilePath();
  writeStream = fs.createWriteStream(filePath, { flags: "w" });
  currentSize = 0;
  console.log(`📄 Writing to ${filePath}`);
}

function shouldExclude(filePath) {
  return excludeDirs.some((exclude) =>
    filePath.includes(path.sep + exclude + path.sep)
  );
}

function isHiddenOrInvalid(filePath) {
  const baseName = path.basename(filePath);
  const ext = path.extname(filePath);
  return baseName.startsWith(".") || !allowedExtensions.includes(e
}

function writeWithSizeCheck(data) {
  const buffer = Buffer.from(data, "utf-8");

  if (currentSize + buffer.length > maxFileSize) {
    outputFileIndex++;
    openNewWriteStream();
  }
```

```
  writeStream.write(buffer);
  currentSize += buffer.length;
}

function processFile(filePath) {
  if (isHiddenOrInvalid(filePath)) return;

  const relativePath = path.relative(rootDir, filePath);
  const content = fs.readFileSync(filePath, "utf-8");

  const header = `--- FILE: ${relativePath} ---\n`;
  const fullContent = `${header}${content}\n\n`;

  writeWithSizeCheck(fullContent);
}

function traverseDirectory(dir) {
  const entries = fs.readdirSync(dir, { withFileTypes: true });

  for (const entry of entries) {
    const fullPath = path.join(dir, entry.name);

    if (entry.name.startsWith(".")) continue;

    if (entry.isDirectory()) {
      if (!shouldExclude(fullPath)) {
        traverseDirectory(fullPath);
      }
    } else {
      processFile(fullPath);
    }
  }
}

function main() {
  openNewWriteStream();
  traverseDirectory(rootDir);
  if (writeStream) writeStream.end(() => console.log("✅ All done
}

main();
```

FILE: copyDirectoryCLI.js

```
#!/usr/bin/env node

const fs = require("fs");
const path = require("path");
const puppeteer = require("puppeteer");
require("dotenv").config();

// Load config from .env or use defaults
const rootDir = path.resolve(process.env.ROOT_DIR || "./");
const excludeDirs = (
  process.env.EXCLUDE_DIRS || "node_modules,.git,dist"
).split(",");
const allowedExtensions = (process.env.ALLOWED_EXTENSIONS || ".js,
  ","
);
const maxFileSize = parseInt(process.env.MAX_FILE_SIZE_MB || "1")
const outputPrefix = process.env.OUTPUT_PREFIX || "output";

// State variables
let outputFileIndex = 1;
let currentSize = 0;
let fileContents = [];
let outputFiles = [];

// Escapes HTML special characters
function escapeHtml(text) {
  return text
    .replace(/&/g, "&amp;")
    .replace(/</g, "&lt;")
    .replace(/>/g, "&gt;");
}

// Builds a styled HTML block per file
function addToContent(relativePath, content) {
  const htmlSection = `
    <section style="margin-bottom: 40px; page-break-inside: avoid
      <h2 style="font-family: monospace; font-size: 16px; backgrou
        FILE: ${relativePath}
      </h2>
<pre><code class="language-javascript">${escapeHtml(content)}</coc
    </section>
  `;
  fileContents.push(htmlSection);
}

// Chunking based on total buffer size
function writeWithSizeCheck(relativePath, content) {
  const bufferSize = Buffer.byteLength(content, "utf-8");

  if (currentSize + bufferSize > maxFileSize) {
    outputFiles.push([...fileContents]);
    fileContents = [];
    outputFileIndex++;
```

```
    currentSize = 0;
  }

  currentSize += bufferSize;
  addToContent(relativePath, content);
}

// Check if a path should be excluded
function shouldExclude(filePath) {
  return excludeDirs.some((exclude) =>
    filePath.includes(path.sep + exclude + path.sep)
  );
}

// Validate file based on extension and hidden status
function isHiddenOrInvalid(filePath) {
  const baseName = path.basename(filePath);
  const ext = path.extname(filePath);
  return baseName.startsWith(".") || !allowedExtensions.includes(e
}

// Process and collect file content
function processFile(filePath) {
  if (isHiddenOrInvalid(filePath)) return;

  const relativePath = path.relative(rootDir, filePath);
  const content = fs.readFileSync(filePath, "utf-8");
  writeWithSizeCheck(relativePath, content);
}

// Recursively walk through directories
function traverseDirectory(dir) {
  const entries = fs.readdirSync(dir, { withFileTypes: true });

  for (const entry of entries) {
    const fullPath = path.join(dir, entry.name);

    if (entry.name.startsWith(".")) continue;

    if (entry.isDirectory()) {
      if (!shouldExclude(fullPath)) {
        traverseDirectory(fullPath);
      }
    } else {
      processFile(fullPath);
    }
  }
}

// Save collected HTML as a styled PDF
async function saveAsPDF(index, htmlContent) {
  const browser = await puppeteer.launch();
  const page = await browser.newPage();

  const html = `
    <html>
    <head>
  <meta charset="UTF-8">
```

```
  <title>Output</title>

  <!-- Font -->
  <link href="https://fonts.googleapis.com/css2?family=Roboto+Mono

  <!-- Highlight.js Arduino Light Theme -->
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/l

  <style>
    body {
      font-family: 'Roboto Mono', monospace;
      font-size: 12px;
      color: #333;
      padding: 40px;
    }
    pre {
      background: #f8f8f8;
      padding: 12px;
      border-radius: 4px;
      overflow-x: auto;
    }
    code {
      font-family: 'Roboto Mono', monospace;
      font-size: 12px;
    }
  </style>
</head>

    <body>
    <span class="title">Title</span> – <span class="date">Date</sp

      ${htmlContent}
      <script src="https://cdnjs.cloudflare.com/ajax/libs/highligh
<script>hljs.highlightAll();</script>

    </body>
    </html>
  `;

  await page.setContent(html, { waitUntil: "domcontentloaded" });

  const pdfPath = path.join(process.cwd(), `${outputPrefix}-${inde

  await page.pdf({
    path: pdfPath,
    format: "A4",
    printBackground: true,
    margin: {
      top: "1in",
      bottom: "1in",
      left: "1in",
      right: "1in",
    },
    displayHeaderFooter: true,
    headerTemplate: `
      <div style="font-family: Roboto Mono, monospace; font-size:
        <span>${outputPrefix}-${index}.pdf</span>
      </div>
```

```
    `,
    footerTemplate: `
      <div style="font-family: Roboto Mono, monospace; font-size:
        Page <span class="pageNumber"></span> of <span class="tota
      </div>
    `,
  });

  await browser.close();
  console.log(`📄 PDF saved: ${pdfPath}`);
}

// Entry point
async function main() {
  console.log(`📂 Scanning: ${rootDir}`);
  traverseDirectory(rootDir);

  // Push last batch
  if (fileContents.length) {
    outputFiles.push([...fileContents]);
  }

  // Generate PDFs
  for (let i = 0; i < outputFiles.length; i++) {
    const htmlContent = outputFiles[i].join("\n");
    await saveAsPDF(i + 1, htmlContent);
  }

  console.log("✅ All done!");
}

main();
```

FILE: package.json

```
{
  "name": "copy-directory-cli",
  "version": "1.0.0",
  "description": "",
  "main": "copyDirectoryCLI.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "packageManager": "pnpm@10.8.1",
  "dependencies": {
    "dotenv": "^16.5.0",
    "puppeteer": "^24.8.2"
  },
  "bin": {
    "copydir": "./copyDirectoryCLI.js"
  }
}
```