# index.html

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">    <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>Code to Word</title>    <script src="https://unpkg.com/docx@7.1.0/build/index.js"></script>  <script src="https://cdnjs.cloudflare.com/ajax/libs/FileSaver.js/1.3.8/FileSaver.js"></script>  <style>  body {  font-family: sans-serif;  text-align: center;  }  </style>  </head>    <body>  <h1>Code to Word</h1>  <input type="number" id="heading\_number" name="heading\_number" placeholder="Heading Number: 2">  <br>  <input type="text" id="path\_font" name="path\_font" placeholder="Path Font: Calibri Light">  <br>  <input type="number" id="path\_size" name="path\_size" placeholder="Path Size: 12">  <br>  <input type="number" id="code\_size" name="code\_size" placeholder="Code Size: 9">  <br>  <input type="text" id="code\_font" name="code\_font" placeholder="Code Font: Courier New">  <br>  <textarea rows="1" cols="50" id="exts" name="exts"  placeholder="Extensions: rs py js ts html css scss md txt"></textarea>  <br>  <button type="button" id="start">Start</button>    <script>  /\*\*  \* Process a file  \* @param {FileSystemDirectoryHandle} root  \* @param {FileSystemFileHandle} path  \* @param {number} path\_size  \* @param {number} code\_size  \*  \* @returns {[docx.Paragraph, docx.Table]}  \*/  async function process\_file(root, path, heading\_number, path\_font, path\_size, code\_font, code\_size) {  const paragraph = new docx.Paragraph({  heading: docx.HeadingLevel["HEADING\_" + heading\_number],  pageBreakBefore: true,  children: [  new docx.TextRun({  text: path.name,  font: path\_font,  size: path\_size \* 2,  }),  ],  });    const file = await path.getFile();  const text = await file.text();    const children = text.replace("\r", "").split("\n").map((line) => {  return new docx.Paragraph({  alignment: docx.AlignmentType.JUSTIFIED,  spacing: {  after: 0,  before: 0,  },  children: [  new docx.TextRun({  text: line,  font: code\_font,  size: code\_size \* 2,  }),  ]  })  })    const table = new docx.Table({  rows: [  new docx.TableRow({  children: [  new docx.TableCell({  children,  }),  ],  }),  ],  })    return [paragraph, table]  }    /\*\*  \* Scans the input directory for files.  \* @param {FileSystemDirectoryHandle} root  \* @param {FileSystemDirectoryHandle} input  \* @param {string[]} file\_exts  \* @param {number} path\_size  \* @param {number} code\_size  \* @param {any[]} output  \*  \* @returns {Promise<[docx.Paragraph, docx.Table]>}  \*/  async function scan\_dir(root, input, file\_exts, heading\_number, path\_font, path\_size, code\_font, code\_size, output) {  output = output || [];    // Iterate over the entries  for await (const entry of input.values()) {  if (entry.kind == "file" && file\_exts.includes(entry.name.split(".").pop())) {  const [a, b] = await process\_file(root, entry, heading\_number, path\_font, path\_size, code\_font, code\_size);  output.push(a);  output.push(b);  } else if (entry.kind == "directory") {  await scan\_dir(root, entry, file\_exts, heading\_number, path\_font, path\_size, code\_font, code\_size, output);  }  }    return output  }    document.getElementById("start").addEventListener("click", async \_ => {  /\*\*  \* Grab the heading number  \* @type {number}  \*/  let heading\_number = parseInt(document.getElementById("heading\_number").value);  if (isNaN(heading\_number)) {  heading\_number = 2;  return;  }  console.log(heading\_number);    /\*\*  \* Grab the path size  \* @type {number}  \*/  const path\_size = parseInt(document.getElementById("path\_size").value) || 12;    /\*\*  \* Grab the code size  \* @type {number}  \*/  const code\_size = parseInt(document.getElementById("code\_size").value) || 9;    /\*\*  \* Grab the path font  \* @type {string}  \*/  const path\_font = document.getElementById("path\_font").value || "Calibri Light";    /\*\*  \* Grab the code font  \* @type {string}  \*/  const code\_font = document.getElementById("code\_font").value || "Courier New";    /\*\*  \* Grab the extensions we want to search for  \* @type {string[]}  \*/  let exts = document.getElementById("exts").value.split(" ");  if (exts.length == 1 && exts[0] == "") {  exts = ["rs", "py", "js", "ts", "html", "css", "scss", "md", "txt"];  }    /\*\*  \* Grab the directory we want to search  \* @type {FileSystemDirectoryHandle}  \*/  const dir = await window.showDirectoryPicker();    // Create the document  const doc = new docx.Document({  compatability: {  doNotExpandShiftReturn: true,  },  sections: [{  children: await scan\_dir(dir, dir, exts, heading\_number, path\_font, path\_size, code\_font, code\_size),  }]  })    // Save the document  docx.Packer.toBlob(doc).then((blob) => {  saveAs(blob, "code.docx");  console.log("Document created successfully");  });  }, true)  </script>  </body>    </html> |

# README.md

|  |
| --- |
| # Code to Word (WEB)    A helper tool to export all code in a directory to a word document. Designed for A Level Computer Science students to help with their NEA technical solution write up.    This is a web version. |