

Building Elements for Personal Websites

Welcome to JavaScript Games: Programming Fundamentals!

This tutorial looks at how you build your BabylonJS elements **separately** for integration into your personal websites. This will ensure that we can bring **each element individually** and ensure they will run on your website independently. Please see the following tutorial in this tutorial guiding you through this process and also **how to upload your entire website to GitHub for submission**.

Please note: a new working solution with the Havok Physics has been edited into the lab sheet. Go through the process and ensure you follow each step CLEARLY. I have provided a shortcut for moving the HavokPhysics.wasm too. This can be found in the new section 'HavokPhysics.wasm Shortcut Script'.

To begin, in your BabylonJS directory in Visual Studio Code, we have each of our element folders but in our babylonProj (or whatever you have named this), you have a package.json file. Open this. **Please ensure it's the package.json INSIDE your babylonProj**. It should look similar as below.

```
{
  "name": "testproj",
  "private": true,
  "version": "0.0.0",
  "type": "module",
  "scripts": {
    "dev": "vite",
    "build": "tsc && vite build",
    "preview": "vite preview"
  },
  "devDependencies": {
    "typescript": "^5.0.2",
    "vite": "^4.4.0"
  }
}
```

We are going to edit this for each of our elements. Now for my elements, my folders are called 'element01', 'element02' and so on. With the next step ensure you have the folder name followed by the keyword. See below.

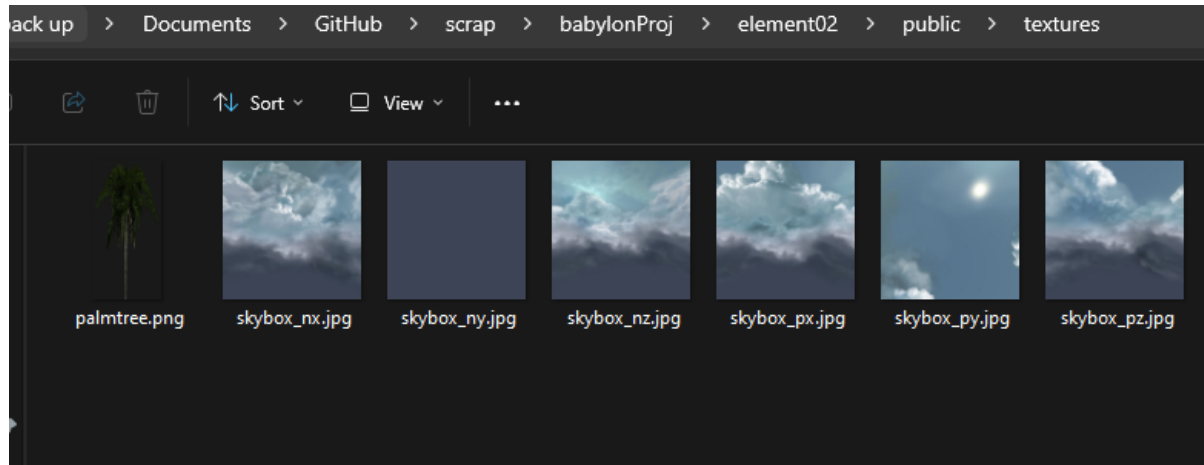
```
babylonProj > {} package.json > {} scripts
1  {
2    "name": "testproj",
3    "private": true,
4    "version": "0.0.0",
5    "type": "module",
6    "scripts": {
7      "dev": "vite",
8      "build": "tsc && vite build",
9      "preview": "vite preview",
10     "element01build": "tsc && vite build element01",
11     "element01preview": "vite preview element01",
12     "element02build": "tsc && vite build element02",
13     "element02preview": "vite preview element02",
14     "element03build": "tsc && vite build element03",
15     "element03preview": "vite preview element03",
16     "element04build": "tsc && vite build element04",
17     "element04preview": "vite preview element04",
18     "element05build": "tsc && vite build element05",
19     "element05preview": "vite preview element05"
20   },
21   "devDependencies": {
22     "typescript": "^5.0.2",
23     "vite": "^4.4.5"
24   }
25 }
26
```

As you can see here under “scripts”, we have added builds and previews for each element. You can copy this the name but when you specify “**element01build**”, the part in **bold** should be replaced with the name of your folder that **your element is in**.

The next step is building your element. For this example, I will build Element 2 on my demonstration project. Press ‘CTRL + C’ and ensure you have stopped the terminal and any development servers from running.

Important! Before building: notes here around building each elements. If you have necessary files to support such as textures folder, you are going to have to add these to another folder too for building purposes. **Copy the folders DO NOT remove from your main folder. You will have a set in your main folder and another in your ‘public’ folder.**

In each of your elements, create a ‘public’ folder and insert any folders you have. This may be models, textures, etc. I will be copying the ‘textures’ folder for the purpose of this demonstration.



As you can see in the image above, I have created a 'public' folder inside my 'element02' folder and put the 'textures' folder **inside the 'public' folder**.

Havok Physics scenes (element 3, 4 and 5): If you are preparing an element with Havok Physics, add the HavokPhysics.wasm file inside the 'public' folder too.

Next, in your element, you are going to right click 'elementXX' folder and add a new file called 'vite.config.js'. This should create below your index.html. Open this file and enter the following code.

```
import { defineConfig } from "vite";

export default defineConfig({
  esbuild: {
    supported: {
      'top-level-await': true //browsers can handle top-level-await features
    },
  },
})
```

Once you have done this, you can build an element. I will be doing Element 2, as mentioned. Ensure you are inside your 'babylonProj' folder in the **terminal** in Visual Studio Code and then run 'npm run **element02build**'. You will change the **bold** text to your represented element you wish to build. **Remember this is referring to the script we added and yours may be differently named.**

This will take a minute, if not more to build. Please be patient. **Please note at this point in a scene if you are working with any Havok Physics in your scene and not did the necessary steps state for Havok Physics then it will not build.**

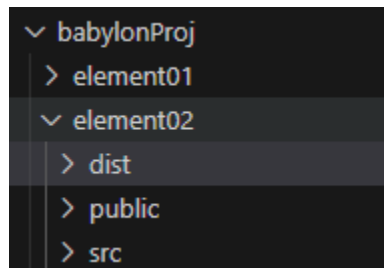
```
node →/workspaces/scrapp/babylonProj $ npm run element02build

> testproj@0.0.0 element02build
> tsc && vite build element02

vite v4.5.0 building for production...
✓ 1585 modules transformed.
dist/index.html                0.28 kB | gzip:    0.21 kB
dist/assets/index-d6d7b775.css 0.10 kB | gzip:    0.10 kB
dist/assets/index-d1135aae.js  9,888.76 kB | gzip: 2,412.50 kB

(!) Some chunks are larger than 500 kB after minification. Consider:
- Using dynamic import() to code-split the application
- Use build.rollupOptions.output.manualChunks to improve chunking: https://rollupjs.org/configuration-options/#output-manualchunks
- Adjust chunk size limit for this warning via build.chunkSizeWarningLimit.
✓ built in 52.81s
node →/workspaces/scrapp/babylonProj $
```

Once the build is finished, you should see a 'dist' folder appear in your directory under the specific element.



Copy the 'dist' folder and for the time being paste it somewhere safe. I have copied it to my desktop for the time being. Open your website in the file explorer and also the 'dist' folder.

In the 'dist' folder rename 'index.html' to the name of your element. I have named it 'element2.html'.

My current website project hierarchy looks like this:

Name	Date modified	Type	Size
assets	28/09/2023 10:35	File folder	
CSS	28/09/2023 10:36	File folder	
JS	28/09/2023 10:36	File folder	
design.html	22/11/2023 13:06	Opera GX Web Do...	3 KB
index.html	22/11/2023 13:06	Opera GX Web Do...	9 KB

If you have your website in a 'public' folder. You will have to take all content out the public folder and keep in the main folder to get this working. We are going to add our HTML to this area. I am currently using our Bootstrap demonstration website from Week 4 called 'JSGF-Week4Bootstrap'.

Name	Date modified	Type	Size
assets	28/09/2023 10:35	File folder	
CSS	28/09/2023 10:36	File folder	
JS	28/09/2023 10:36	File folder	
design.html	22/11/2023 13:06	Opera GX Web Do...	3 KB
element2.html	22/11/2023 13:10	Opera GX Web Do...	1 KB
index.html	22/11/2023 13:06	Opera GX Web Do...	9 KB

We want all our HTML files to be in the same location. Next copy your textures folder and place in with the other folders. It should be as below. You will want to copy any other folders you have. **If you ever have any folders with same names you will have to combine all assets into one folder, this is fine.**

assets	22/11/2023 13:11	File folder	
CSS	28/09/2023 10:36	File folder	
JS	28/09/2023 10:36	File folder	
textures	22/11/2023 13:12	File folder	
design.html	22/11/2023 13:13	Opera GX Web Do...	3 KB
element2.html	22/11/2023 13:10	Opera GX Web Do...	1 KB
index.html	22/11/2023 13:13	Opera GX Web Do...	9 KB

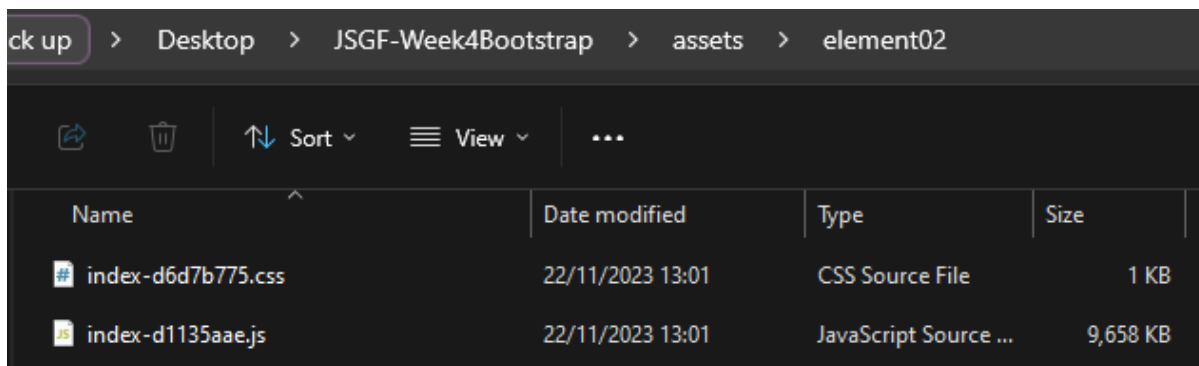
Next, go into the assets folder. As you remember from before, when we built our project, you can see the HTML file, assets and textures. The assets folder for us contains all the JavaScript and CSS files needed to run our BabylonJS game. In your website, if you already have an assets folder good, if not copy in your assets folder from the dist project.

We are next going to move the two files within it. This is a great way to keep it organised for your Elements. You will want to create a folder for each element inside your assets folder and your desired JavaScript and CSS files will go inside here. **It is important to know that your elementX.html file needs to find these so you will have to add some extra code to the elementX.html file in order to find it.** You should initially see as below.

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="UTF-8">
5     <title>Element 2</title>
6     <script type="module" crossorigin src="/assets/index-d1135aae.js"></script>
7     <link rel="stylesheet" href="/assets/index-d6d7b775.css">
8   </head>
9   <body> </body>
10 </html>
11
```

After the '/assets/...' you want to add the desired folder in. I will add element02 as I have created a folder within. See both code and corresponding folder structure.

```
<> element2.html > ...
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="UTF-8">
5     <title>Element 2</title>
6     <script type="module" crossorigin src="/assets/element02/index-d1135aae.js"></script>
7     <link rel="stylesheet" href="/assets/element02/index-d6d7b775.css">
8   </head>
9   <body> </body>
10 </html>
11
```



ck up > Desktop > JSGF-Week4Bootstrap > assets > element02				
Sort View ...				
Name	Date modified	Type	Size	
index-d6d7b775.css	22/11/2023 13:01	CSS Source File	1 KB	
index-d1135aae.js	22/11/2023 13:01	JavaScript Source ...	9,658 KB	

Now, any time you direct to your Element HTMLs, they should load no problem. **Remember, you will have to link your HTML up with your website. You may want to add a nav bar into your Element HTML files so you can return.**

HavokPhysics.wasm Shortcut Script

In previous labs, when working with the BabylonJS Havok Physics, if there was ever an issue with this, we had to move the 'HavokPhysics.wasm' file into the '.vite/deps' folder in the node_modules. Now I have a shortcut for this to assist in the process instead of having to drag and drop every time.

In your 'babylonProj' package.json file, we are going to add a script. This is the same 'package.json' that we added our build and preview statements to.

We are going to add the following line within the 'scripts' section.

```
"havok": "cp /workspaces/jspf-babylonjs-2023/node_modules/@babylonjs/havok/lib/esm/HavokPhysics.wasm /workspaces/jspf-babylonjs-2023/initialisation/node_modules/.vite/deps",
```

To get the path right click on the file or folder and 'Copy Path'. Replace both your location with your desired path. This is the path from my project.

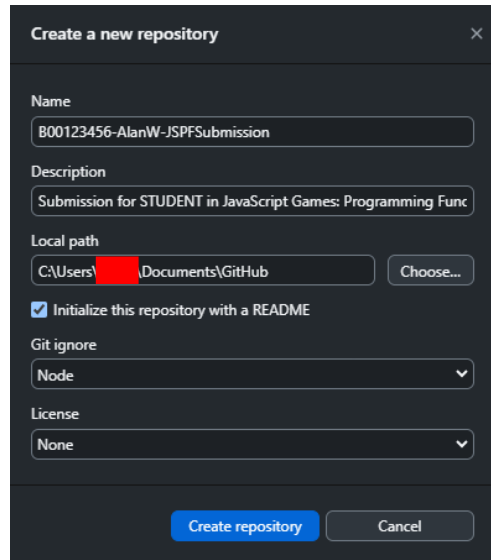
I have added this before our 'element01build', 'element01preview', 'element02build' etc. If you ever get the previous error we are aware of with the .wasm file, you can go to the terminal. Make sure you are in the 'babylonProj' folder and run 'npm run havok'.

Adding Your Existing Website to GitHub

You should be aware of this process partially since we completed it last time for the setup of Docker, Vite and BabylonJS. For this, I will set out how you can put your website on GitHub ready for submission. Let's begin.

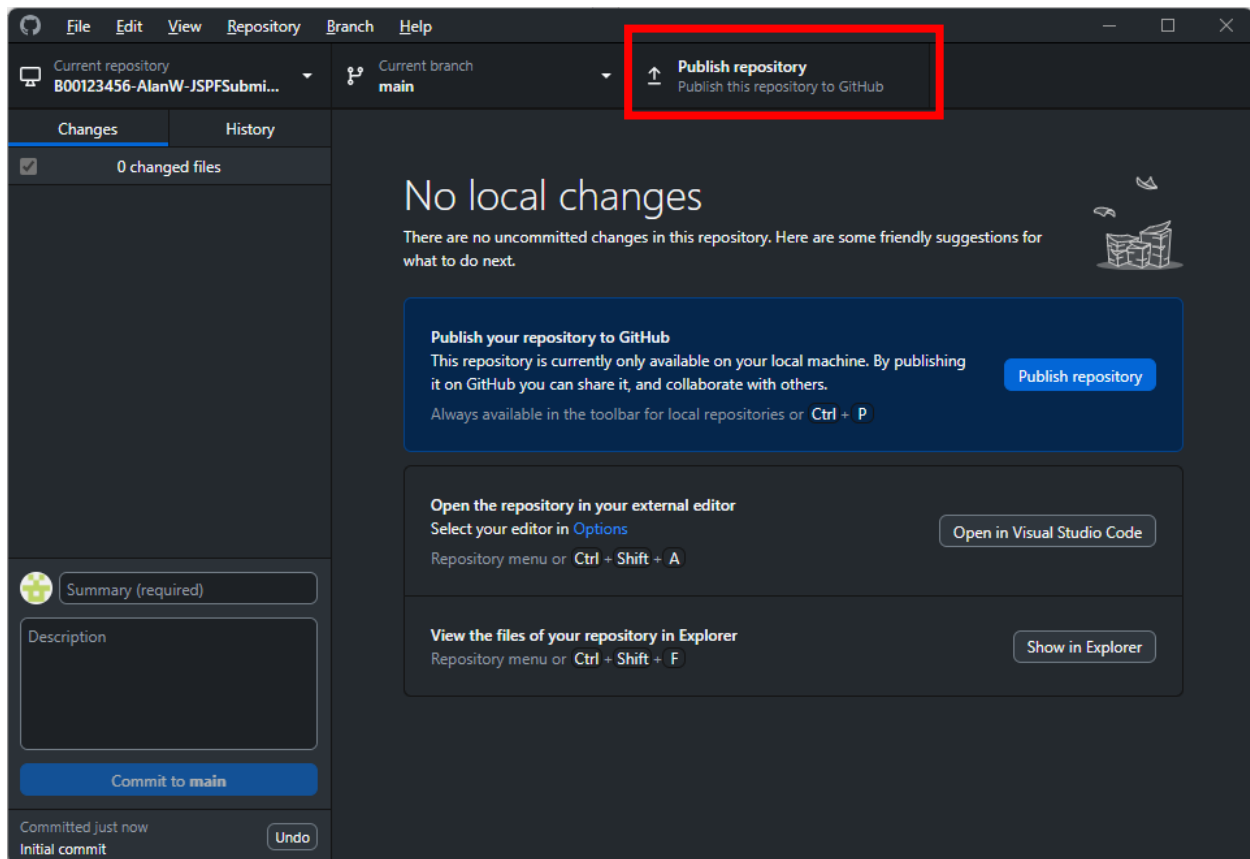
Login to your GitHub Desktop. You will already be using GitHub Desktop for our Docker, Vite, and BabylonJS. We are going to create a new repository. You can access this by going to 'File' > 'New repository...'. **Name your repository whatever you wish.** We will drag all files from your current website into this location.

You can either do this after you have finished everything or now. I will demonstrate this during a demonstration in Week 12.

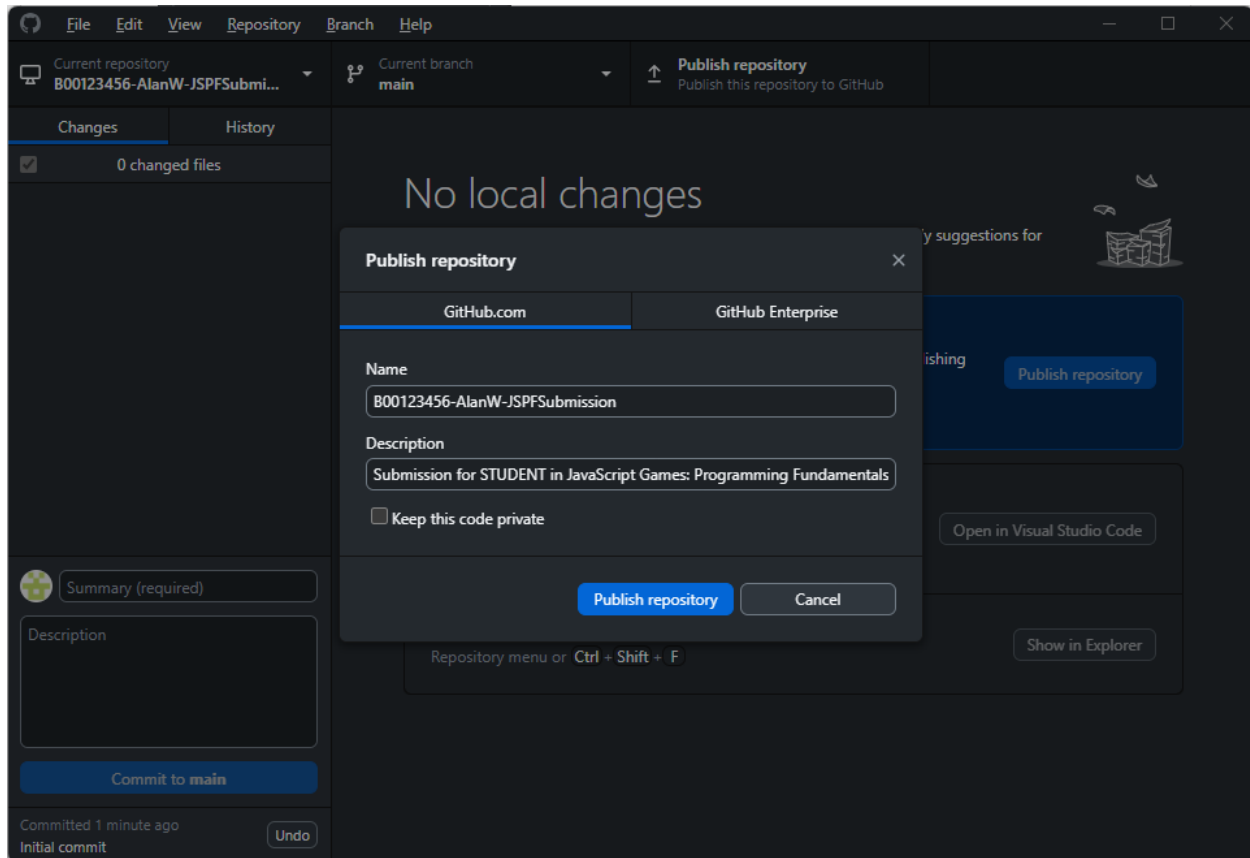


The screenshot shows the 'Create a new repository' dialog box. The 'Name' field contains 'B00123456-AlanW-JSPFSubmission'. The 'Description' field contains 'Submission for STUDENT in JavaScript Games: Programming Func'. The 'Local path' field contains 'C:\Users\ [redacted] \Documents\GitHub'. The 'Initialize this repository with a README' checkbox is checked. The 'Git ignore' dropdown is set to 'Node'. The 'License' dropdown is set to 'None'. At the bottom, there are 'Create repository' and 'Cancel' buttons.

Name your repository name as the format shown, replacing your Banner ID with your own and your name. You don't need to include a description. **Tick the box for a README. Add a Git ignore for Node.** The Git ignore for this may not be needed but will be added as any precautions. Press 'Create repository'.

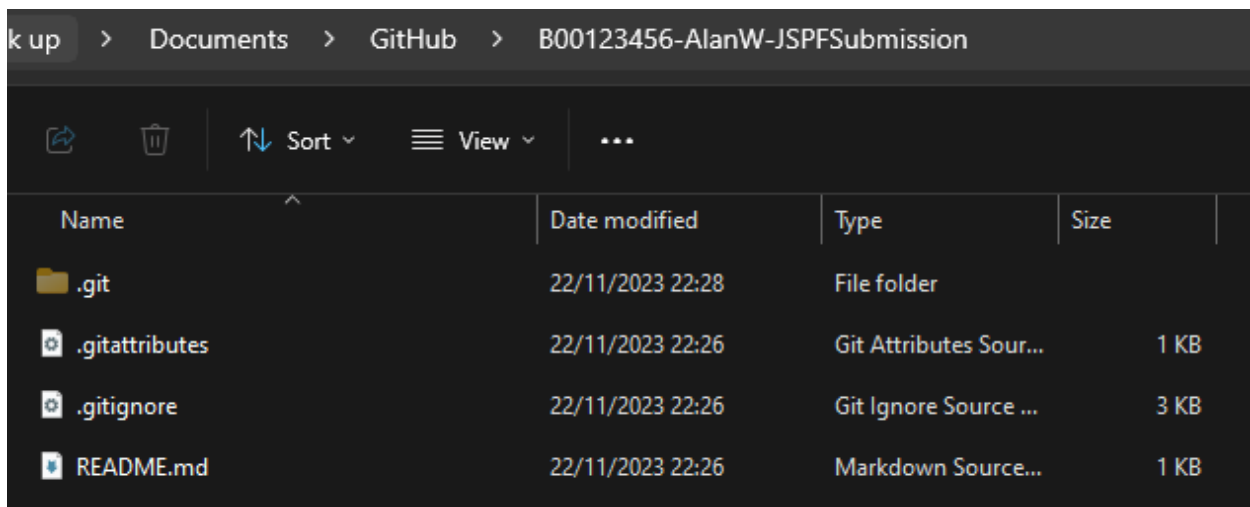


Click 'Publish Repository' as highlighted above.

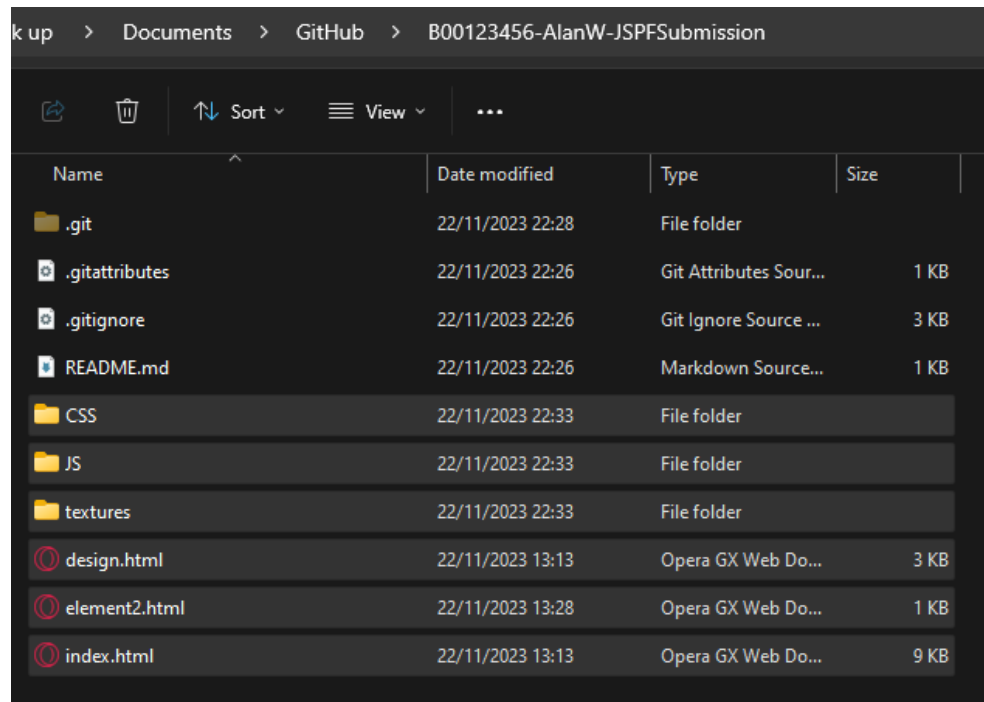
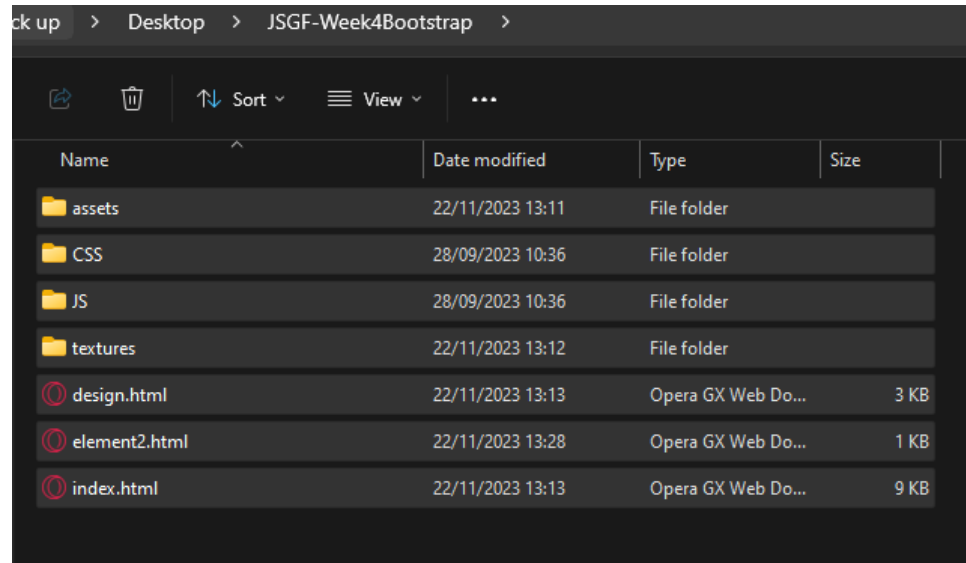


The following prompt will appear. You can untick 'Keep this code private' to keep things easier. Then click 'Publish repository'. This will create the repository in your saved location.

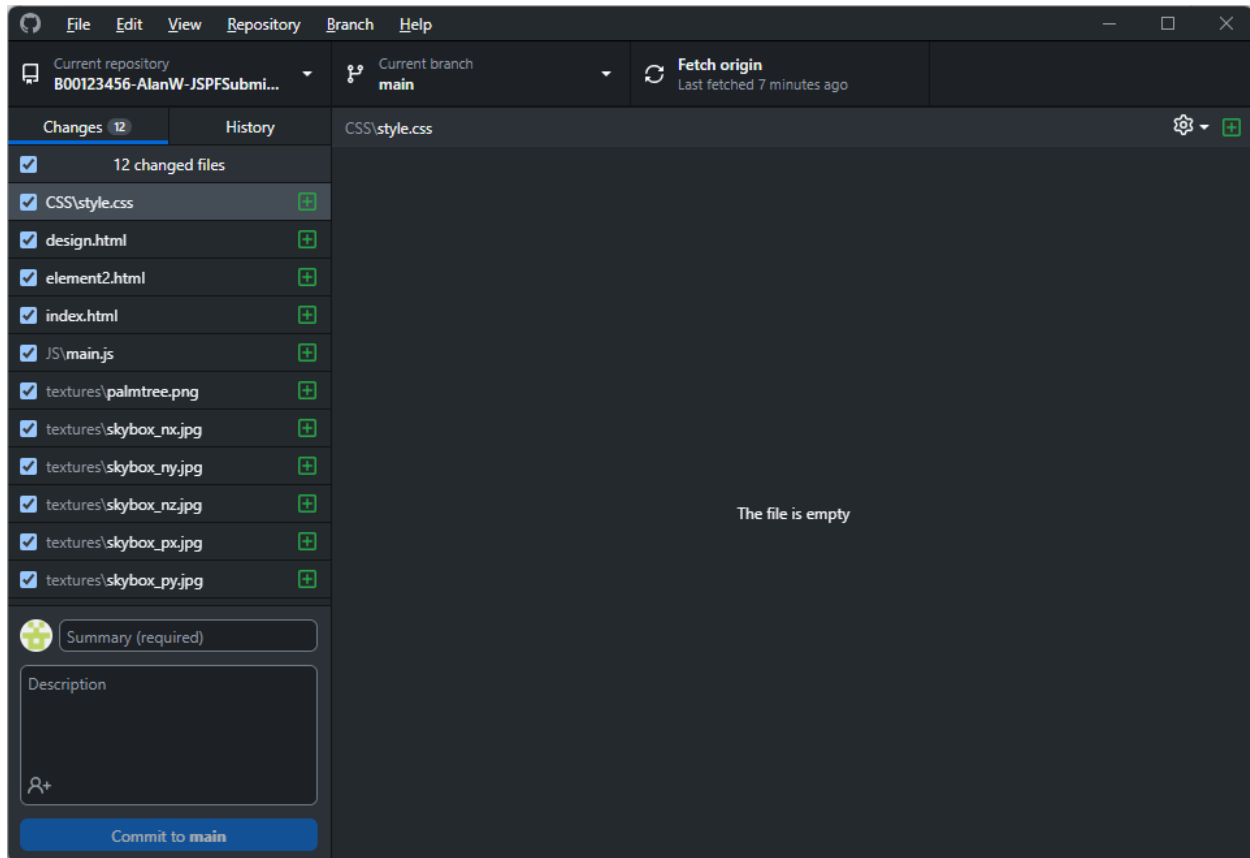
Next, go to this GitHub repository location and open the folder in your file explorer.



It should look as follows including the following files. To transfer your website, locate it in your file explorer. Open the main folder. Select all files and **copy and paste** into your GitHub folder. See process below.



Once you have added in, if you return to GitHub Desktop, you should see your changes added.



Next, do the usual process that you should be aware of, add a Summary, Description if needed and push to your repository.

There you have it. That is as simple as that. To share your GitHub link, you can go to [GitHub.com](https://github.com), login, access your repository under 'Your Repositories', select your repository and copy the link once selected.

Any questions related to this, please contact your lecturer(s).