```
Uncaught ReferenceError: addStudent is not defined
at HTMLButtonElement.onclick ((index):14:36)Understand this errorAI
(index):28 Uncaught ReferenceError: changeFont is not defined
at HTMLButtonElement.onclick ((index):28:36)Understand this errorAI
:5173/favicon.ico:1


Failed to load resource: the server responded with a status of 404 (Not Found)Understand this errorAI
3(index):14 Uncaught ReferenceError: addStudent is not defined
at HTMLButtonElement.onclick ((index):14:36)Understand this errorAI
8(index):28 Uncaught ReferenceError: changeFont is not defined
at HTMLButtonElement.onclick ((index):28:36)Understand this errorAI
localhost/:1 Uncaught (in promise) Error: A listener indicated an asynchronous response by returning true, but the message channel closed before a response was receivedUnderstand this errorAI
4localhost/:1 Uncaught (in promise) Error: Could not establish connection. Receiving end does not exist.Understand this errorAI
client:913 [vite] server connection lost. Polling for restart...
```

It looks like you're encountering several errors in your web application. I'll break down each error and provide a brief explanation:

1. Uncaught ReferenceError: `addStudent` is not defined

- **Error Message:** `Uncaught ReferenceError: addStudent is not defined`
- **Location:** `HTMLButtonElement.onclick ((index):14:36)`
- **Explanation:** This error occurs when the JavaScript function `addStudent` is not defined or is not accessible within the scope of the button's `onclick` event.
- **Solution:** Ensure that the `addStudent` function is defined and accessible. Check for any typos or scope-related issues.

2. Uncaught ReferenceError: `changeFont` is not defined

- **Error Message:** `Uncaught ReferenceError: changeFont is not defined`
- **Location:** `HTMLButtonElement.onclick ((index):28:36)`
- **Explanation:** Similar to the previous error, this occurs when the JavaScript function `changeFont` is not defined or is not accessible within the scope of the button's `onclick` event.
- **Solution:** Ensure that the `changeFont` function is defined and accessible. Check for any typos or scope-related issues.

3. Failed to load resource: 404 (Not Found)

- **Error Message:** `Failed to load resource: the server responded with a status of 404 (Not Found)`
- **Location:** `/favicon.ico:1`

- **Explanation:** This error occurs when the browser is unable to find the favicon.ico file at the specified location. The favicon is usually displayed in the browser's address bar or bookmarks.
- **Solution:** Ensure that the favicon.ico file exists in the root directory of your website or update the link to point to the correct location.

4. Uncaught (in promise) Error: A listener indicated an asynchronous response...

- **Error Message:** `Uncaught (in promise) Error: A listener indicated an asynchronous response by returning true, but the message channel closed before a response was received`
- **Location:** `localhost/:1`
- **Explanation:** This error is related to asynchronous communication between the main thread and a worker or another context. It indicates that a listener returned `true` to indicate an asynchronous response, but the message channel was closed before the response was received.
- **Solution:** Ensure that the message channel remains open until the asynchronous response is received. Check your code for any issues related to asynchronous communication.

5. Uncaught (in promise) Error: Could not establish connection. Receiving end does not exist.

- **Error Message:** `Uncaught (in promise) Error: Could not establish connection. Receiving end does not exist.`
- **Location:** `localhost/:1`
- **Explanation:** This error occurs when trying to establish a connection between two contexts (e.g., main thread and worker) using the `postMessage` API. The error indicates that the receiving end does not exist or is not listening for messages.
- **Solution:** Ensure that the receiving end is properly set up and listening for messages. Check your code for any issues related to establishing connections between contexts.

6. [vite] server connection lost. Polling for restart...

- **Error Message:** `[vite] server connection lost. Polling for restart...`
- **Location:** `client:913`
- **Explanation:** This error occurs when the development server connection is lost. Vite will attempt to reconnect to the server.
- **Solution:** This error is usually temporary and resolves itself when the connection is re-established. If the issue persists, try restarting the development server.

okay i fixed these errors but the button does not execute the function anymore

Now that the errors are fixed, let's troubleshoot why the button isn't executing the function.

Possible Causes:

1. **Typo in function name**: Double-check that the function name in the button's `onclick` attribute matches the actual function name in your JavaScript code.
2. **Function scope**: Ensure that the function is defined in the global scope or is accessible within the scope of the button's `onclick` event.
3. **Button element**: Verify that the button element is correctly defined in your HTML code and that it has the correct `id` or `class` attribute.
4. **Event listener**: Instead of using the `onclick` attribute, try adding an event listener to the button element using JavaScript.

Debugging Steps:

1. **Check the browser console**: Look for any error messages in the browser console that might indicate why the function isn't being executed.
2. **Add a console log**: Temporarily add a `console.log` statement inside the function to see if it's being executed. If the log statement doesn't appear in the console, it indicates that the function isn't being called.
3. **Inspect the button element**: Use the browser's developer tools to inspect the button element and verify that it has the correct `onclick` attribute or event listener attached.

Example Code:

If you're still having trouble, please provide more details about your code, including the HTML, JavaScript, and any relevant CSS. This will help me better understand your issue and provide a more accurate solution.

Here's an example of how you might define a button and attach a click event listener using JavaScript:

HTML

```html
<!-- index.html -->
<button id="myButton">Click me!</button>
```

JavaScript

```javascript
// script.js
const button = document.getElementById('myButton');

button.addEventListener('click', function() {
  console.log('Button clicked!');
  // Add your function code here
});
```

```
import studentData from './students.json' assert { type: "json" };
import './style.scss'
// Render the student table
function renderTable(students) {
const tableBody =
document.getElementById("studentTable").getElementsByTagName("tbody")[0];
```

```
tableBody.innerHTML = ""; // Clear existing rows
students.forEach(student => {
const newRow = tableBody.insertRow();
newRow.insertCell(0).innerText = student.id;
newRow.insertCell(1).innerText = student.name;
newRow.insertCell(2).innerText = student.age;
newRow.insertCell(3).innerText = student.grade;
});

}

// Add a student
export function addStudent() {
const id = prompt("Enter Student ID:");
const name = prompt("Enter Student Name:");
const age = prompt("Enter Student Age:");
const grade = prompt("Enter Student Grade:");
if (!id || !name || !age || !grade) {
alert("All fields are required!");
return;
}
const tableBody = document.querySelector("#studentTable tbody");
const newRow = document.createElement("tr");
newRow.innerHTML = `
<td>${id}</td>
<td>${name}</td>
<td>${age}</td>
<td>${grade}</td>
`;
newRow.classList.add("highlight");
tableBody.appendChild(newRow);
}

// Change font
export function changeFont() {
const fontUrl = "https://fonts.googleapis.com/css2?family=Roboto:wght@400;700&display=swap";
const fontLink = document.createElement("link");
fontLink.rel = "stylesheet";
fontLink.href = fontUrl;
document.head.appendChild(fontLink);
document.querySelector("table").style.fontFamily = "'Roboto', sans-serif";
}

// Display browser info
function displayBrowserInfo() {
const browserNameEl = document.querySelector("#browserName");
const browserVersionEl = document.querySelector("#browserVersion");
const windowDimensionsEl =
document.querySelector("#windowDimensions");
const userAgentEl = document.querySelector("#userAgent");
browserNameEl.textContent = navigator.appName;
browserVersionEl.textContent = navigator.appVersion;
windowDimensionsEl.textContent =
`${window.innerWidth}x${window.innerHeight}`;
userAgentEl.textContent = navigator.userAgent;
}

// Initialize the app
document.addEventListener("DOMContentLoaded", () => {
renderTable(studentData);
displayBrowserInfo();
});
```

modify this script

Here's a modified version of your script with some improvements:

JavaScript

```javascript
// Import student data from JSON file
import studentData from './students.json' assert { type: "json" };

// Import stylesheet
import './style.scss';

// Function to render student table
function renderTable(students) {
  const tableBody = document.querySelector("#studentTable tbody");
  tableBody.innerHTML = ""; // Clear existing rows

  students.forEach((student) => {
    const newRow = tableBody.insertRow();
    newRow.insertCell(0).textContent = student.id;
    newRow.insertCell(1).textContent = student.name;
    newRow.insertCell(2).textContent = student.age;
    newRow.insertCell(3).textContent = student.grade;
  });
}

// Function to add a new student
export function addStudent() {
  const id = prompt("Enter Student ID:");
  const name = prompt("Enter Student Name:");
  const age = prompt("Enter Student Age:");
  const grade = prompt("Enter Student Grade:");

  if (!id || !name || !age || !grade) {
    alert("All fields are required!");
    return;
  }
  const newStudent = {
    id,
    name,
    age,
    grade,
  };

  studentData.push(newStudent);
  renderTable(studentData);
}

// Function to change font
export function changeFont() {
  const fontUrl = "https://fonts.googleapis.com/css2?family=Roboto:wght@400;700&display=
  const fontLink = document.createElement("link");
  fontLink.rel = "stylesheet";
  fontLink.href = fontUrl;
  document.head.appendChild(fontLink);
  document.querySelector("table").style.fontFamily = "'Roboto', sans-serif";
}

// Function to display browser info
function displayBrowserInfo() {
  const browserNameEl = document.querySelector("#browserName");
  const browserVersionEl = document.querySelector("#browserVersion");
  const windowDimensionsEl = document.querySelector("#windowDimensions");
  const userAgentEl = document.querySelector("#userAgent");

  browserNameEl.textContent = navigator.appName;
  browserVersionEl.textContent = navigator.appVersion;
  windowDimensionsEl.textContent = `${window.innerWidth}x${window.innerHeight}`;
  userAgentEl.textContent = navigator.userAgent;
}

// Initialize the app
document.addEventListener("DOMContentLoaded", () => {
  renderTable(studentData);
  displayBrowserInfo();
});
```

I made the following changes:

- Replaced `getElementsByTagName` with `querySelector` for better performance and readability.
- Used `textContent` instead of `innerText` for better compatibility and performance.
- In the `addStudent` function, I created a new student object and pushed it to the `studentData` array. Then, I called `renderTable` to update the table with the new student data.
- Removed the `highlight` class addition in the `addStudent` function, as it was not defined in the provided code. If you need to highlight the newly added row, you can add the CSS class and style it accordingly.
- The rest of the code remains the same, as it was already well-structured and functional.

I will give you an html convert that into this js format

```
\\ what you will give me
import './style.css'
import javascriptLogo from './javascript.svg'
import viteLogo from '/vite.svg'
import { setupCounter } from './counter.js'

document.querySelector('#app').innerHTML = `
<div>
<a href="https://vite.dev" target="_blank">
<img src="${viteLogo}" class="logo" alt="Vite logo" />
</a>
<a href="https://developer.mozilla.org/en-US/docs/Web/JavaScript"
target="_blank">
<img src="${javascriptLogo}" class="logo vanilla" alt="JavaScript logo" />
</a>
<h1>Hello Vite!</h1>
<div class="card">
<button id="counter" type="button"></button>
</div>
<p class="read-the-docs">
Click on the Vite logo to learn more
</p>
</div>
`

setupCounter(document.querySelector('#counter'))
```

I'm ready to help. Please provide the HTML code you'd like me to convert into JavaScript format.

```
<!DOCTYPE html>
<html lang="en">

<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Student Management Dashboard</title>
</head>

<body>
<h1>Student Management Dashboard</h1>
<button type="button" (click)="addStudent()">Add Student</button>
<table id="studentTable">
<thead>
<tr>
```

```
<th>ID</th>
<th>Name</th>
<th>Age</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<!-- Rows will be dynamically added here -->
</tbody>
</table>
<button type="button" (click)="changeFont">Change Font</button>
<div id="browserInfo">
<p><strong>Browser:</strong> <span id="browserName"></span></p>
<p><strong>Version:</strong> <span id="browserVersion"></span></p>
<p><strong>Window Dimensions:</strong> <span id="windowDimensions"></span></p>
<p><strong>User Agent:</strong> <span id="userAgent"></span></p>
</div>
<script type="module" src="./src/main.js"></script>
</body>

</html>
```

do it with this

Here is the JavaScript code that generates the same HTML structure:

JavaScript

```javascript
import './style.css';

document.querySelector('#app').innerHTML = `
  <h1>Student Management Dashboard</h1>
  <button type="button" id="addStudentBtn">Add Student</button>
  <table id="studentTable">
    <thead>
      <tr>
        <th>ID</th>
        <th>Name</th>
        <th>Age</th>
        <th>Grade</th>
      </tr>
    </thead>
    <tbody id="studentTableBody">
      <!-- Rows will be dynamically added here -->
    </tbody>
  </table>
  <button type="button" id="changeFontBtn">Change Font</button>
  <div id="browserInfo">
    <p><strong>Browser:</strong> <span id="browserName"></span></p>
    <p><strong>Version:</strong> <span id="browserVersion"></span></p>
    <p><strong>Window Dimensions:</strong> <span id="windowDimensions"></span></p>
```

```
      <p><strong>User Agent:</strong> <span id="userAgent"></span></p>
    </div>
  `;

  // Add event listeners to buttons
  document.querySelector('#addStudentBtn').addEventListener('click', addStudent);
  document.querySelector('#changeFontBtn').addEventListener('click', changeFont);
```

Note that I removed the `(click)` event binding from the buttons, as it's not a standard HTML attribute. Instead, I added event listeners to the buttons using JavaScript.

Also, I assumed that the `addStudent` and `changeFont` functions are defined elsewhere in your code. If not, you'll need to define them accordingly.

so what shoudl i put in the index.html

Since you're using JavaScript to generate the HTML content, your `index.html` file can be very minimal.

Here's what you can put in your `index.html` file:

HTML

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Student Management Dashboard</title>
  <div id="app"></div>
  <script type="module" src="./main.js"></script>
</head>
<body>
</body>
</html>
```

In this code:

- We create a basic HTML structure with a `head` and `body` section.
- We add a `div` element with an `id` of "app". This is where our JavaScript code will render the HTML content.
- We link our `main.js` file to the HTML file using a `script` tag.

That's it! Your `index.html` file is now ready.

```
// Add a student to the table with user input

function addStudentDummy() {
const table =
document.getElementById('studentTable').getElementsByTagName('tbod
y')[0];
const newRow = table.insertRow();

// Insert cells and populate them
newRow.insertCell(0).innerText = 2; // ID
newRow.insertCell(1).innerText = 'Jane Smith'; // Name
newRow.insertCell(2).innerText = 19; // Age
newRow.insertCell(3).innerText = 'B'; // Grade

// Highlight the newly added row
newRow.classList.add('highlight');
}

// Change font of the table using Google Fonts
function changeFont() {
const link = document.getElementById('google-font-link');
link.href = 'https://fonts.googleapis.com/css2?family=Varela+Round&display=swap';
document.getElementById('studentTable').style.fontFamily = "'Varela Round', sans-serif";
}

function addStudent() {
// Prompt the user for student details
const id = prompt("Enter Student ID:");
const name = prompt("Enter Student Name:");
const age = prompt("Enter Student Age:");
const grade = prompt("Enter Student Grade:");

// Validate input
if (!id || !name || !age || !grade) {
alert("All fields are required!");
return;
}

// Get the student table body
const tableBody = document.querySelector("#studentTable tbody");

// Create a new table row
const newRow = document.createElement("tr");

// Add student data to the row
newRow.innerHTML = `
<td>${id}</td>
<td>${name}</td>
<td>${age}</td>
<td>${grade}</td>
`;
```

```
// Add the highlight class
newRow.classList.add("highlight");

// Append the new row to the table
tableBody.appendChild(newRow);
}


// Change the font of the table
function changeFont() {
// Specify the Google Font
const fontUrl = "https://fonts.googleapis.com/css2?
family=Roboto:wght@400;700&display=swap";

// Get the Google Font link element
const fontLink = document.querySelector("#google-font-link");

// Set the href to the Google Font URL
fontLink.href = fontUrl;

// Apply the font to the table
const table = document.querySelector("table");
table.style.fontFamily = "'Roboto', sans-serif";
table.style.fontSize = "1.9rem"
table.style.color = "navy"
}

// Display browser information
function displayBrowserInfo() {
// Get browser info elements
const browserNameEl = document.querySelector("#browserName");
const browserVersionEl = document.querySelector("#browserVersion");
const windowDimensionsEl =
document.querySelector("#windowDimensions");
const userAgentEl = document.querySelector("#userAgent");

// Get browser details
const userAgent = navigator.userAgent;
const browserName = navigator.appName;
const browserVersion = navigator.appVersion;
const windowWidth = window.innerWidth;
const windowHeight = window.innerHeight;

// Set the content of the elements
browserNameEl.textContent = browserName;
browserVersionEl.textContent = browserVersion;
windowDimensionsEl.textContent = `${windowWidth}x${windowHeight}`;
userAgentEl.textContent = userAgent;
}
```

// Call displayBrowserInfo on page load
window.onload = displayBrowserInfo;

now give me this script I will import these function from script.js

Here's the modified JavaScript code that you can put in your `script.js` file:

JavaScript

```javascript
// Add a student to the table with user input
export function addStudent() {
  // Prompt the user for student details
  const id = prompt("Enter Student ID:");
  const name = prompt("Enter Student Name:");
  const age = prompt("Enter Student Age:");
  const grade = prompt("Enter Student Grade:");

  // Validate input
  if (!id || !name || !age || !grade) {
    alert("All fields are required!");
    return;
  }

  // Get the student table body
  const tableBody = document.querySelector("#studentTable tbody");

  // Create a new table row
  const newRow = document.createElement("tr");

  // Add student data to the row
  newRow.innerHTML = `
    <td>${id}</td>
    <td>${name}</td>
    <td>${age}</td>
    <td>${grade}</td>
  `;

  // Add the highlight class
  newRow.classList.add("highlight");

  // Append the new row to the table
  tableBody.appendChild(newRow);
}

// Change the font of the table
export function changeFont() {
  // Specify the Google Font
  const fontUrl = "https://fonts.googleapis.com/css2?family=Roboto:wght@400;700&display=

  // Get the Google Font link element
  const fontLink = document.querySelector("#google-font-link");

  // Set the href to the Google Font URL
  fontLink.href = fontUrl;
  // Apply the font to the table
  const table = document.querySelector("table");
  table.style.fontFamily = "'Roboto', sans-serif";
  table.style.fontSize = "1.9rem";
  table.style.color = "navy";
}

// Display browser information
export function displayBrowserInfo() {
  // Get browser info elements
  const browserNameEl = document.querySelector("#browserName");
  const browserVersionEl = document.querySelector("#browserVersion");
  const windowDimensionsEl = document.querySelector("#windowDimensions");
  const userAgentEl = document.querySelector("#userAgent");

  // Get browser details
  const userAgent = navigator.userAgent;
  const browserName = navigator.appName;
  const browserVersion = navigator.appVersion;
  const windowWidth = window.innerWidth;
  const windowHeight = window.innerHeight;

  // Set the content of the elements
  browserNameEl.textContent = browserName;
  browserVersionEl.textContent = browserVersion;
  windowDimensionsEl.textContent = `${windowWidth}x${windowHeight}`;
  userAgentEl.textContent = userAgent;
}
```

Now you can import and use these functions in your main JavaScript file.

so how will the main.js look like now is there any difference

Here's how your `main.js` file will look like:

JavaScript

```
import './style.css';
import { addStudent, changeFont, displayBrowserInfo } from './script.js';

document.querySelector('#app').innerHTML = `
  <h1>Student Management Dashboard</h1>
  <button type="button" id="addStudentBtn">Add Student</button>
  <table id="studentTable">
    <thead>
      <tr>
        <th>ID</th>
        <th>Name</th>
        <th>Age</th>
        <th>Grade</th>
      </tr>
    </thead>
    <tbody id="studentTableBody">
      <!-- Rows will be dynamically added here -->
    </tbody>
  </table>
  <button type="button" id="changeFontBtn">Change Font</button>
  <div id="browserInfo">
    <p><strong>Browser:</strong> <span id="browserName"></span></p>
    <p><strong>Version:</strong> <span id="browserVersion"></span></p>
    <p><strong>Window Dimensions:</strong> <span id="windowDimensions"></span></p>
    <p><strong>User Agent:</strong> <span id="userAgent"></span></p>
  </div>
`;

// Add event listeners to buttons
document.querySelector('#addStudentBtn').addEventListener('click', addStudent);
document.querySelector('#changeFontBtn').addEventListener('click', changeFont);

// Display browser information
displayBrowserInfo();
```

The main differences are:

- We import the `addStudent`, `changeFont`, and `displayBrowserInfo` functions from `script.js`.
- We call these functions from `main.js` instead of defining them here.
- We removed the `addStudentDummy` function as it's no longer needed.

```
// Add a student to the table with user input

function addStudentDummy() {
const table =
document.getElementById('studentTable').getElementsByTagName('tbody')[0];
const newRow = table.insertRow();

// Insert cells and populate them
newRow.insertCell(0).innerText = 2; // ID
newRow.insertCell(1).innerText = 'Jane Smith'; // Name
newRow.insertCell(2).innerText = 19; // Age
newRow.insertCell(3).innerText = 'B'; // Grade

// Highlight the newly added row
newRow.classList.add('highlight');
}
```

also create a button add dummy student and then also put this function in the script.js.

Here's the updated `script.js` file:

JavaScript

```
// ... (other functions remain the same)
// Add a dummy student to the table
export function addStudentDummy() {
  const table = document.getElementById('studentTable').getElementsByTagName('tbody')[0
  const newRow = table.insertRow();

  // Insert cells and populate them
  newRow.insertCell(0).innerText = 2;  // ID
  newRow.insertCell(1).innerText = 'Jane Smith';  // Name
  newRow.insertCell(2).innerText = 19;  // Age
  newRow.insertCell(3).innerText = 'B';  // Grade

  // Highlight the newly added row
  newRow.classList.add('highlight');
}
```

And here's the updated `main.js` file:

JavaScript

```javascript
import './style.css';
import { addStudent, changeFont, displayBrowserInfo, addStudentDummy } from './script.j

document.querySelector('#app').innerHTML = `
  <h1>Student Management Dashboard</h1>
  <button type="button" id="addStudentBtn">Add Student</button>
  <button type="button" id="addDummyStudentBtn">Add Dummy Student</button>
  <table id="studentTable">
    <thead>
      <tr>
        <th>ID</th>
        <th>Name</th>
        <th>Age</th>
        <th>Grade</th>
      </tr>
    </thead>
    <tbody id="studentTableBody">
      <!-- Rows will be dynamically added here -->
    </tbody>
  </table>
  <button type="button" id="changeFontBtn">Change Font</button>
  <div id="browserInfo">
    <p><strong>Browser:</strong> <span id="browserName"></span></p>
    <p><strong>Version:</strong> <span id="browserVersion"></span></p>
    <p><strong>Window Dimensions:</strong> <span id="windowDimensions"></span></p>
    <p><strong>User Agent:</strong> <span id="userAgent"></span></p>
  </div>
`;

// Add event listeners to buttons
document.querySelector('#addStudentBtn').addEventListener('click', addStudent);
document.querySelector('#addDummyStudentBtn').addEventListener('click', addStudentDummy
document.querySelector('#changeFontBtn').addEventListener('click', changeFont);

// Display browser information
displayBrowserInfo();
```

```
Uncaught TypeError: Cannot set properties of null (setting 'href')
at HTMLButtonElement.changeFont (script.js:59:19)
```

This error is caused by the `changeFont` function trying to set the `href` property of an element that doesn't exist.

The problematic line of code is:

JavaScript

```javascript
const fontLink = document.querySelector("#google-font-link");
fontLink.href = fontUrl;
```

The error message indicates that `fontLink` is `null`, which means that the `#google-font-link` element doesn't exist in the HTML.

To fix this issue, you need to add the `#google-font-link` element to your HTML. You can do this by adding a `link` element to the `head` section of your HTML:

HTML

```html
<head>
  <link id="google-font-link" rel="stylesheet">
  <!-- Other head elements -->
</head>
```

Alternatively, you can create the `link` element dynamically in the `changeFont` function:

JavaScript

```javascript
function changeFont() {
  // Specify the Google Font
  const fontUrl = "https://fonts.googleapis.com/css2?family=Roboto:wght@400;700&display=

  // Create the link element dynamically
  const fontLink = document.createElement("link");
  fontLink.id = "google-font-link";
  fontLink.rel = "stylesheet";
  fontLink.href = fontUrl;

  // Add the link element to the head
  document.head.appendChild(fontLink);

  // Apply the font to the table
  const table = document.querySelector("table");
  table.style.fontFamily = "'Roboto', sans-serif";
  table.style.fontSize = "1.9rem";
  table.style.color = "navy";
}
```

Define a CSS file with classes like m-1, m-2 until m-8. Each m-n is n*2px.

Now do the same in SCSS, with loops. See how you can do it with vite (it supports scss).

do this task now then

Here are the solutions:

CSS Solution

Create a new file called `styles.css` and add the following code:

CSS

```css
.m-1 {
  margin: 2px;
}

.m-2 {
  margin: 4px;
}

.m-3 {
  margin: 6px;
}

.m-4 {
  margin: 8px;
}

.m-5 {
  margin: 10px;
}

.m-6 {
  margin: 12px;
}

.m-7 {
  margin: 14px;
}

.m-8 {
  margin: 16px;
}
```

SCSS Solution with Loops

Create a new file called `styles.scss` and add the following code:

Scss

```scss
@for $i from 1 through 8 {
  .m-#{$i} {
    margin: ($i * 2)px;
  }
}
```

To use SCSS with Vite, you need to install the `sass` package:

Bash

```
npm install sass
```

Then, you can import your SCSS file in your JavaScript code:

JavaScript

```
import './styles.scss';
```

Vite will automatically compile your SCSS code into CSS.

can you give me a fragment of html (for the javascript) that has classes with these elements

Here's an example HTML fragment that uses the margin classes:

HTML

```
<div id="app">
  <h1>Margin Classes Example</h1>
  <div class="m-1">Margin 2px</div>
  <div class="m-2">Margin 4px</div>
  <div class="m-3">Margin 6px</div>
  <div class="m-4">Margin 8px</div>
  <div class="m-5">Margin 10px</div>
  <div class="m-6">Margin 12px</div>
  <div class="m-7">Margin 14px</div>
  <div class="m-8">Margin 16px</div>
</div>
```

does this work with our javascript function ? like make it visible that its working

To make it visible that the margin classes are working with our JavaScript functions, we can modify the HTML to include buttons that trigger the `addStudent` and `changeFont` functions.

Here's an updated HTML fragment:

HTML

```
<div id="app">
  <h1>Student Management Dashboard</h1>
  <button type="button" id="addStudentBtn">Add Student</button>
  <button type="button" id="addDummyStudentBtn">Add Dummy Student</button>
  <button type="button" id="changeFontBtn">Change Font</button>
```

```
    <table id="studentTable">
      <thead>
        <tr>
          <th>ID</th>
          <th>Name</th>
          <th>Age</th>
          <th>Grade</th>
        </tr>
      </thead>
      <tbody id="studentTableBody">
        <!-- Rows will be dynamically added here -->
      </tbody>
    </table>
    <div id="browserInfo">
      <p><strong>Browser:</strong> <span id="browserName"></span></p>
      <p><strong>Version:</strong> <span id="browserVersion"></span></p>
      <p><strong>Window Dimensions:</strong> <span id="windowDimensions"></span></p>
      <p><strong>User Agent:</strong> <span id="userAgent"></span></p>
    </div>
    <div class="m-1">Margin 2px</div>
    <div class="m-2">Margin 4px</div>
    <div class="m-3">Margin 6px</div>
    <div class="m-4">Margin 8px</div>
    <div class="m-5">Margin 10px</div>
    <div class="m-6">Margin 12px</div>
    <div class="m-7">Margin 14px</div>
    <div class="m-8">Margin 16px</div>
  </div>
```

Now, when you run the application, you should see the margin classes applied to the div elements at the bottom of the page.

we need that here

```
import './style.scss';
import { addStudent, changeFont, displayBrowserInfo,
addStudentDummy } from './script.js';

document.querySelector('#app').innerHTML = `
<h1>Student Management Dashboard</h1>
<button type="button" id="addStudentBtn">Add Student</button>
<button type="button" id="addDummyStudentBtn">Add Dummy
Student</button>
<table id="studentTable">
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Age</th>
<th>Grade</th>
</tr>
</thead>
<tbody id="studentTableBody">
<!-- Rows will be dynamically added here -->
```

```
</tbody>
</table>
<button type="button" id="changeFontBtn">Change Font</button>
<div id="browserInfo">
<p><strong>Browser:</strong> <span id="browserName"></span></p>
<p><strong>Version:</strong> <span id="browserVersion"></span></p>
<p><strong>Window Dimensions:</strong> <span id="windowDimensions"></span></p>
<p><strong>User Agent:</strong> <span id="userAgent"></span></p>
</div>

`;

// Add event listeners to buttons
document.querySelector('#addStudentBtn').addEventListener('click', addStudent);
document.querySelector('#addDummyStudentBtn').addEventListener('click', addStudentDummy);
document.querySelector('#changeFontBtn').addEventListener('click', changeFont);

// Display browser information
displayBrowserInfo();
```

Here is the revised JavaScript code with the additional HTML elements that demonstrate the margin classes:

JavaScript

```javascript
import './style.scss';
import { addStudent, changeFont, displayBrowserInfo, addStudentDummy } from './script.j

document.querySelector('#app').innerHTML = `
  <h1>Student Management Dashboard</h1>
  <button type="button" id="addStudentBtn">Add Student</button>
  <button type="button" id="addDummyStudentBtn">Add Dummy Student</button>
  <table id="studentTable">
    <thead>
      <tr>
        <th>ID</th>
        <th>Name</th>
        <th>Age</th>
        <th>Grade</th>
      </tr>
    </thead>
    <tbody id="studentTableBody">
      <!-- Rows will be dynamically added here -->
    </tbody>
  </table>
  <button type="button" id="changeFontBtn">Change Font</button>
  <div id="browserInfo">
    <p><strong>Browser:</strong> <span id="browserName"></span></p>
    <p><strong>Version:</strong> <span id="browserVersion"></span></p>
```

```
      <p><strong>Window Dimensions:</strong> <span id="windowDimensions"></span></p>
      <p><strong>User Agent:</strong> <span id="userAgent"></span></p>
    </div>
    <div class="m-1">Margin 2px</div>
    <div class="m-2">Margin 4px</div>
    <div class="m-3">Margin 6px</div>
    <div class="m-4">Margin 8px</div>
    <div class="m-5">Margin 10px</div>
    <div class="m-6">Margin 12px</div>
    <div class="m-7">Margin 14px</div>
    <div class="m-8">Margin 16px</div>
  `;

  // Add event listeners to buttons
  document.querySelector('#addStudentBtn').addEventListener('click', addStudent);
  document.querySelector('#addDummyStudentBtn').addEventListener('click', addStudentDummy
  document.querySelector('#changeFontBtn').addEventListener('click', changeFont);

  // Display browser information
  displayBrowserInfo();
```

okay that didint work. how sould i approach this

To troubleshoot the issue, let's break it down step by step:

1. **Verify SCSS compilation**: Make sure that your SCSS file is being compiled correctly. Check your terminal or console output for any errors related to SCSS compilation.
2. **Check CSS output**: Inspect the generated CSS file (usually `styles.css`) to ensure that the margin classes (`.m-1` to `.m-8`) are being generated correctly.
3. **Verify HTML structure**: Ensure that the HTML elements with the margin classes are being rendered correctly. Use your browser's developer tools to inspect the HTML elements.
4. **Check CSS styles**: Use your browser's developer tools to inspect the CSS styles applied to the elements with the margin classes. Verify that the margin styles are being applied correctly.
5. **Test with a simple example**: Create a simple HTML file with a single element that uses one of the margin classes (e.g., `.m-1`). Verify that the margin style is being applied correctly.

If none of these steps help, please provide more details about your project, including:

- Your SCSS code
- Your HTML code
- Any error messages you're seeing
- Your project's file structure

This will help me better understand your issue and provide a more accurate solution.

create a div fragment with a box and then put these elements inside there, then apply the scss

Here's an example of how you can create a `div` fragment with a box and apply the SCSS styles:

HTML

```
<div class="box">
  <div class="m-1">Margin 2px</div>
  <div class="m-2">Margin 4px</div>
  <div class="m-3">Margin 6px</div>
  <div class="m-4">Margin 8px</div>
  <div class="m-5">Margin 10px</div>
  <div class="m-6">Margin 12px</div>
  <div class="m-7">Margin 14px</div>
  <div class="m-8">Margin 16px</div>
</div>
```

And here's an example of how you can define the `.box` class in your SCSS file:

Scss

```
.box {
  width: 300px;
  height: 200px;
  border: 1px solid #ccc;
  padding: 20px;
  background-color: #f9f9f9;

  .m-1 {
    margin: 2px;
    background-color: #ffe6e6;
  }

  .m-2 {
    margin: 4px;
    background-color: #ffe6cc;
  }

  .m-3 {
    margin: 6px;
    background-color: #ffffcc;
  }

  .m-4 {
    margin: 8px;
    background-color: #ccffe6;
  }

  .m-5 {
    margin: 10px;
    background-color: #cce6ff;
  }
```

```
    .m-6 {
      margin: 12px;
      background-color: #e6ccff;
    }

    .m-7 {
      margin: 14px;
      background-color: #ffccff;
    }

    .m-8 {
      margin: 16px;
      background-color: #ff99cc;
    }
  }
```

You can then apply the `.box` class to the `div` element that contains the margin elements:

JavaScript

```
document.querySelector('#app').innerHTML = `
  <div class="box">
    <div class="m-1">Margin 2px</div>
    <div class="m-2">Margin 4px</div>
    <div class="m-3">Margin 6px</div>
    <div class="m-4">Margin 8px</div>
    <div class="m-5">Margin 10px</div>
    <div class="m-6">Margin 12px</div>
    <div class="m-7">Margin 14px</div>
    <div class="m-8">Margin 16px</div>
  </div>
`;
```

```
import './style.scss';
import { addStudent, changeFont, displayBrowserInfo,
addStudentDummy } from './script.js';

document.querySelector('#app').innerHTML = `
<h1>Student Management Dashboard</h1>
<button type="button" id="addStudentBtn">Add Student</button>
<button type="button" id="addDummyStudentBtn">Add Dummy
Student</button>
<table id="studentTable">
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Age</th>
<th>Grade</th>
</tr>
</thead>
<tbody id="studentTableBody">
<!-- Rows will be dynamically added here -->
```

```
</tbody>
</table>
<button type="button" id="changeFontBtn">Change Font</button>
<div id="browserInfo">
<p><strong>Browser:</strong> <span id="browserName"></span></p>
<p><strong>Version:</strong> <span id="browserVersion"></span></p>
<p><strong>Window Dimensions:</strong> <span id="windowDimensions"></span></p>
<p><strong>User Agent:</strong> <span id="userAgent"></span></p>
</div>

`;

// Add event listeners to buttons
document.querySelector('#addStudentBtn').addEventListener('click', addStudent);
document.querySelector('#addDummyStudentBtn').addEventListener('click', addStudentDummy);
document.querySelector('#changeFontBtn').addEventListener('click', changeFont);

// Display browser information
displayBrowserInfo();
```

inside this just add the box below it

Here is the revised JavaScript code:

JavaScript

```javascript
import './style.scss';
import { addStudent, changeFont, displayBrowserInfo, addStudentDummy } from './script.j

document.querySelector('#app').innerHTML = `
  <h1>Student Management Dashboard</h1>
  <button type="button" id="addStudentBtn">Add Student</button>
  <button type="button" id="addDummyStudentBtn">Add Dummy Student</button>
  <table id="studentTable">
    <thead>
      <tr>
        <th>ID</th>
        <th>Name</th>
        <th>Age</th>
        <th>Grade</th>
      </tr>
    </thead>
    <tbody id="studentTableBody">
      <!-- Rows will be dynamically added here -->
    </tbody>
  </table>
  <button type="button" id="changeFontBtn">Change Font</button>
  <div id="browserInfo">
    <p><strong>Browser:</strong> <span id="browserName"></span></p>
    <p><strong>Version:</strong> <span id="browserVersion"></span></p>
    <p><strong>Window Dimensions:</strong> <span id="windowDimensions"></span></p>
    <p><strong>User Agent:</strong> <span id="userAgent"></span></p>
  </div>
  <div class="box">
    <div class="m-1">Margin 2px</div>
    <div class="m-2">Margin 4px</div>
    <div class="m-3">Margin 6px</div>
    <div class="m-4">Margin 8px</div>
    <div class="m-5">Margin 10px</div>
    <div class="m-6">Margin 12px</div>
    <div class="m-7">Margin 14px</div>
    <div class="m-8">Margin 16px</div>
  </div>
`;

// Add event listeners to buttons
document.querySelector('#addStudentBtn').addEventListener('click', addStudent);
document.querySelector('#addDummyStudentBtn').addEventListener('click', addStudentDummy
document.querySelector('#changeFontBtn').addEventListener('click', changeFont);

// Display browser information
displayBrowserInfo();
```

@for $i from 1 through 8 {
.m-#{$i} {
margin: ($i * 2)px;
}
}

is this correct

Your SCSS code is correct:

Scss

```
@for $i from 1 through 8 {
  .m-#{$i} {
    margin: ($i * 2)px;
  }
}
```

This code will generate the following CSS:

CSS

```
.m-1 {
  margin: 2px;
}
```

```
  .m-2 {
    margin: 4px;
  }

  .m-3 {
    margin: 6px;
  }

  .m-4 {
    margin: 8px;
  }

  .m-5 {
    margin: 10px;
  }

  .m-6 {
    margin: 12px;
  }

  .m-7 {
    margin: 14px;
  }

  .m-8 {
    margin: 16px;
  }
```

Well done!

plain css is working but scss is not.

If your plain CSS is working but your SCSS is not, there could be a few reasons for this:

1. **SCSS compilation issue**: Make sure that your SCSS file is being compiled correctly. Check your terminal or console output for any errors related to SCSS compilation.
2. **File naming or location issue**: Ensure that your SCSS file is named correctly (e.g., `styles.scss`) and is located in the correct directory.
3. **Import statement issue**: Verify that you are importing the SCSS file correctly in your JavaScript file. The import statement should look something like this: `import './styles.scss';`
4. **CSS class naming issue**: Double-check that the CSS class names in your SCSS file match the class names you are using in your HTML.

To troubleshoot the issue, you can try the following:

1. Check the browser's developer tools to see if the SCSS styles are being applied.
2. Inspect the HTML elements to see if the correct CSS classes are being applied.
3. Verify that the SCSS file is being compiled correctly by checking the terminal or console output.

If none of these steps help, please provide more details about your project, including:

- Your SCSS code
- Your HTML code
- Your JavaScript code
- Any error messages you're seeing

This will help me better understand your issue and provide a more accurate solution.

how to check if scss compilation is working properly

Here are some ways to check if SCSS compilation is working properly:

1. Check the Terminal or Console Output

When you run your build process (e.g., `npm start` or `yarn start`), check the terminal or console output for any errors related to SCSS compilation.

2. Verify the Generated CSS File

Check if a CSS file is being generated in your project's directory. The file name should match the name of your SCSS file, but with a `.css` extension instead of `.scss`.

3. Inspect the Browser's Developer Tools

Open your browser's developer tools and inspect the HTML elements that should be styled by your SCSS code. Check if the styles are being applied correctly.

4. Use a Code Editor with SCSS Support

If you're using a code editor like Visual Studio Code, Sublime Text, or Atom, make sure you have an SCSS plugin or extension installed. These plugins often provide features like syntax highlighting, code completion, and error detection.

5. Check the Build Configuration

Verify that your build configuration (e.g., `webpack.config.js` or `rollup.config.js`) is set up correctly to compile SCSS files.

Some popular build tools for SCSS compilation include:

- Webpack with the `sass-loader` plugin
- Rollup with the `rollup-plugin-sass` plugin
- Gulp with the `gulp-sass` plugin

By checking these areas, you should be able to determine if SCSS compilation is working properly in your project.

in the browser it says invild property

That's a good clue!

When the browser says "invalid property", it usually means that there's a syntax error in your CSS code.
In the case of SCSS, it's possible that the compilation process is not working correctly, resulting in invalid CSS code.

Here are a few things you can try:

1. **Check your SCSS code**: Double-check your SCSS code for any syntax errors. Make sure that you're using the correct syntax for variables, mixins, and other SCSS features.
2. **Check the compiled CSS code**: Inspect the compiled CSS code in the browser's developer tools. Look for any syntax errors or invalid properties.
3. **Try a different SCSS compiler**: If you're using a build tool like Webpack or Rollup, try switching to a different SCSS compiler to see if the issue persists.

Some common SCSS syntax errors that can cause invalid property errors include:

- Missing or mismatched brackets (`{}`) or parentheses (`()`)
- Incorrect use of variables or mixins
- Missing or invalid semicolons (`;`) at the end of statements

By checking your SCSS code and the compiled CSS code, you should be able to identify and fix the syntax error that's causing the invalid property error.

```
@for $i from 1 through 8 {
.m-#{$i} {
margin: ($i * 2)px;
}
}
```

Your SCSS code looks correct!

The `@for` loop is properly defined, and the syntax for generating the CSS classes and properties looks good.

Here's the compiled CSS output:

css

```css
.m-1 {
  margin: 2px;
}

.m-2 {
  margin: 4px;
}

.m-3 {
  margin: 6px;
}

.m-4 {
  margin: 8px;
}

.m-5 {
  margin: 10px;
}

.m-6 {
  margin: 12px;
}

.m-7 {
  margin: 14px;
}

.m-8 {
  margin: 16px;
}
```

If you're still seeing an "invalid property" error, the issue might be elsewhere in your code. Can you please provide more context or details about the error message?

it seems the issue is that there is a white space b/w 2 and px

The whitespace between the number and the unit (`px`) is indeed the culprit!

In SCSS, when using interpolation (`#{$variable}`), the resulting value is treated as a string. When you concatenate this string with a unit (`px`), the whitespace between them is preserved.

To fix this issue, you can simply remove the whitespace between the number and the unit:

Scss

```
@for $i from 1 through 8 {
  .m-#{$i} {
    margin: #{$i * 2}px;
  }
}
```

By removing the whitespace, the resulting CSS will be correct:

CSS

```
.m-1 {
  margin: 2px;
}

.m-2 {
  margin: 4px;
}

.m-3 {
  margin: 6px;
}

.m-4 {
  margin: 8px;
}

.m-5 {
  margin: 10px;
}

.m-6 {
  margin: 12px;
}

.m-7 {
  margin: 14px;
}

.m-8 {
  margin: 16px;
}
```

```
import './style.scss';
import { addStudent, changeFont, displayBrowserInfo, addStudentDummy } from './script.js';

document.querySelector('#app').innerHTML = `
<h1>Student Management Dashboard</h1>
<button type="button" id="addStudentBtn">Add Student</button>
<button type="button" id="addDummyStudentBtn">Add Dummy Student</button>
<table id="studentTable">
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Age</th>
<th>Grade</th>
</tr>
</thead>
<tbody id="studentTableBody">
<!-- Rows will be dynamically added here -->
</tbody>
</table>
<button type="button" id="changeFontBtn">Change Font</button>
<div id="browserInfo">
<p><strong>Browser:</strong> <span id="browserName"></span></p>
<p><strong>Version:</strong> <span id="browserVersion"></span></p>
<p><strong>Window Dimensions:</strong> <span id="windowDimensions"></span></p>
<p><strong>User Agent:</strong> <span id="userAgent"></span></p>
</div>
```

```
<div class="box">
<div class="m-1">Margin 2px</div>
<div class="m-2">Margin 4px</div>
<div class="m-3">Margin 6px</div>
<div class="m-4">Margin 8px</div>
<div class="m-5">Margin 10px</div>
<div class="m-6">Margin 12px</div>
<div class="m-7">Margin 14px</div>
<div class="m-8">Margin 16px</div>
</div>
`;

// Add event listeners to buttons
document.querySelector('#addStudentBtn').addEventListener('click', addStudent);
document.querySelector('#addDummyStudentBtn').addEventListener('click', addStudentDummy);
document.querySelector('#changeFontBtn').addEventListener('click', changeFont);

// Display browser information
displayBrowserInfo();
```

now i am wroking on TS instead of JS lets port the work. first start with this

Here's the equivalent code in TypeScript:

TypeScript

```
import './style.scss';
import { addStudent, changeFont, displayBrowserInfo, addStudentDummy } from './script.t

document.querySelector('#app')!.innerHTML = `
  <h1>Student Management Dashboard</h1>
  <button type="button" id="addStudentBtn">Add Student</button>
  <button type="button" id="addDummyStudentBtn">Add Dummy Student</button>
  <table id="studentTable">
    <thead>
      <tr>
        <th>ID</th>
        <th>Name</th>
        <th>Age</th>
        <th>Grade</th>
      </tr>
    </thead>
    <tbody id="studentTableBody">
      <!-- Rows will be dynamically added here -->
    </tbody>
  </table>
  <button type="button" id="changeFontBtn">Change Font</button>
  <div id="browserInfo">
    <p><strong>Browser:</strong> <span id="browserName"></span></p>
    <p><strong>Version:</strong> <span id="browserVersion"></span></p>
    <p><strong>Window Dimensions:</strong> <span id="windowDimensions"></span></p>
    <p><strong>User Agent:</strong> <span id="userAgent"></span></p>
  </div>
  <div class="box">
    <div class="m-1">Margin 2px</div>
    <div class="m-2">Margin 4px</div>
    <div class="m-3">Margin 6px</div>
    <div class="m-4">Margin 8px</div>
    <div class="m-5">Margin 10px</div>
    <div class="m-6">Margin 12px</div>
    <div class="m-7">Margin 14px</div>
    <div class="m-8">Margin 16px</div>
  </div>
`;

// Add event listeners to buttons
document.querySelector('#addStudentBtn')!.addEventListener('click', addStudent);
document.querySelector('#addDummyStudentBtn')!.addEventListener('click', addStudentDumm
document.querySelector('#changeFontBtn')!.addEventListener('click', changeFont);

// Display browser information
displayBrowserInfo();
```

Note the following changes:

- The file extension of the imported script has been changed to `.ts`.
- The `!` operator has been added after `document.querySelector` to assert that the element will not be null. This is a TypeScript feature called the "non-null assertion operator".

```
// Add a student to the table with user input
export function addStudent() {
// Prompt the user for student details
const id = prompt("Enter Student ID:");
const name = prompt("Enter Student Name:");
const age = prompt("Enter Student Age:");
const grade = prompt("Enter Student Grade:");

// Validate input
if (!id || !name || !age || !grade) {
alert("All fields are required!");
return;
}

// Get the student table body
const tableBody = document.querySelector("#studentTable tbody");

// Create a new table row
const newRow = document.createElement("tr");

// Add student data to the row
```

```
newRow.innerHTML = `
<td>${id}</td>
<td>${name}</td>
<td>${age}</td>
<td>${grade}</td>
`;

// Add the highlight class
newRow.classList.add("highlight");

// Append the new row to the table
tableBody.appendChild(newRow);
}

export function addStudentDummy() {
const table =
document.getElementById('studentTable').getElementsByTagName('tbody')[0];
const newRow = table.insertRow();

// Insert cells and populate them
newRow.insertCell(0).innerText = 2; // ID
newRow.insertCell(1).innerText = 'Jane Smith'; // Name
newRow.insertCell(2).innerText = 19; // Age
newRow.insertCell(3).innerText = 'B'; // Grade

// Highlight the newly added row
newRow.classList.add('highlight');
}

// Change the font of the table
export function changeFont() {
// Specify the Google Font
const fontUrl = "https://fonts.googleapis.com/css2?
family=Roboto:wght@400;700&display=swap";

// Create the link element dynamically
const fontLink = document.createElement("link");
fontLink.id = "google-font-link";
fontLink.rel = "stylesheet";
fontLink.href = fontUrl;

// Add the link element to the head
document.head.appendChild(fontLink);

// Apply the font to the table
const table = document.querySelector("table");
table.style.fontFamily = "'Roboto', sans-serif";
table.style.fontSize = "1.9rem";
table.style.color = "navy";
}

// Display browser information
export function displayBrowserInfo() {
// Get browser info elements
const browserNameEl = document.querySelector("#browserName");
const browserVersionEl = document.querySelector("#browserVersion");
const windowDimensionsEl =
document.querySelector("#windowDimensions");
const userAgentEl = document.querySelector("#userAgent");

// Get browser details
const userAgent = navigator.userAgent;
const browserName = navigator.appName;
const browserVersion = navigator.appVersion;
const windowWidth = window.innerWidth;
const windowHeight = window.innerHeight;

// Set the content of the elements
browserNameEl.textContent = browserName;
browserVersionEl.textContent = browserVersion;
windowDimensionsEl.textContent = `${windowWidth}x${windowHeight}`;
userAgentEl.textContent = userAgent;
}
```

now port this

Here is the equivalent code in TypeScript:

TypeScript

```typescript
// Add a student to the table with user input
export function addStudent(): void {
  // Prompt the user for student details
  const id: string = prompt("Enter Student ID:")!;
  const name: string = prompt("Enter Student Name:")!;
  const age: string = prompt("Enter Student Age:")!;
  const grade: string = prompt("Enter Student Grade:")!;

  // Validate input
  if (!id || !name || !age || !grade) {
    alert("All fields are required!");
    return;
  }

  // Get the student table body
  const tableBody: HTMLTableSectionElement | null = document.querySelector("#studentTab

  if (tableBody) {
    // Create a new table row
    const newRow: HTMLTableRowElement = document.createElement("tr");
    // Add student data to the row
    newRow.innerHTML = `
      <td>${id}</td>
      <td>${name}</td>
      <td>${age}</td>
      <td>${grade}</td>
    `;

    // Add the highlight class
    newRow.classList.add("highlight");

    // Append the new row to the table
    tableBody.appendChild(newRow);
  }
}

export function addStudentDummy(): void {
  const table: HTMLTableElement | null = document.getElementById('studentTable');
  if (table) {
    const tableBody: HTMLTableSectionElement | null = table.getElementsByTagName('tbody
    if (tableBody) {
      const newRow: HTMLTableRowElement = tableBody.insertRow();

      // Insert cells and populate them
      newRow.insertCell(0).innerText = '2';  // ID
      newRow.insertCell(1).innerText = 'Jane Smith';  // Name
      newRow.insertCell(2).innerText = '19';  // Age
      newRow.insertCell(3).innerText = 'B';  // Grade

      // Highlight the newly added row
      newRow.classList.add('highlight');
    }
  }
}

// Change the font of the table
export function changeFont(): void {
  // Specify the Google Font
  const fontUrl: string = "https://fonts.googleapis.com/css2?family=Roboto:wght@400;700

  // Create the link element dynamically
  const fontLink: HTMLLinkElement = document.createElement("link");
  fontLink.id = "google-font-link";
  fontLink.rel = "stylesheet";
  fontLink.href = fontUrl;

  // Add the link element to the head
  document.head.appendChild(fontLink);

  // Apply the font to the table
  const table: HTMLTableElement | null = document.querySelector("table");
  if (table) {
    table.style.fontFamily = "'Roboto', sans-serif";
    table.style.fontSize = "1.9rem";
    table.style.color = "navy";
  }
}

// Display browser information
export function displayBrowserInfo(): void {
  // Get browser info elements
  const browserNameEl: HTMLElement | null = document.querySelector("#browserName");
  const browserVersionEl: HTMLElement | null = document.querySelector("#browserVersion"
  const windowDimensionsEl: HTMLElement | null = document.querySelector("#windowDimensi
  const userAgentEl: HTMLElement | null = document.querySelector("#userAgent");

  // Get browser details
  const userAgent: string = navigator.userAgent;
  const browserName: string = navigator.appName;
  const browserVersion: string = navigator.appVersion;
  const windowWidth: number = window.innerWidth;
  const windowHeight: number = window.innerHeight;

  // Set the content of the elements
  if (browserNameEl) browserNameEl.textContent = browserName;
  if (browserVersionEl) browserVersionEl.textContent = browserVersion;
  if (windowDimensionsEl) windowDimensionsEl.textContent = `${windowWidth}x${windowHeig
  if (userAgentEl) userAgentEl.textContent = userAgent;
}
```

I have two errors.

Cannot find module './script.ts' or its corresponding type declarations.ts(2307)

Type 'HTMLElement | null' is not assignable to type 'HTMLTableElement | null'.
Type 'HTMLElement' is missing the following properties from type 'HTMLTableElement': align, bgColor, border, caption, and 19 more.ts(2322)

Let's address these errors one by one:

This error is occurring because the TypeScript compiler is unable to find the `script.ts` file. Make sure that:

- The file `script.ts` exists in the same directory as the file that's importing it.
- The file `script.ts` is correctly spelled and cased.
- There are no typos in the import statement.

If you're still facing issues, try restarting your TypeScript compiler or IDE.

Error 2: Type 'HTMLElement | null' is not assignable to type 'HTMLTableElement | null'.

This error is occurring because the `document.querySelector` method returns an `HTMLElement | null` type, but you're trying to assign it to a variable that's expecting an `HTMLTableElement | null` type.

To fix this error, you can use the `as` keyword to cast the result of `document.querySelector` to the correct type:

TypeScript

```
const table: HTMLTableElement | null = document.querySelector("table") as HTMLTableEleme
```

Alternatively, you can use the `querySelector` method on the `document.body` property, which returns an `HTMLElement | null` type:

TypeScript

```
const table: HTMLTableElement | null = document.body.querySelector("table");
```

Cannot find module './script.ts' or its corresponding type declarations.ts(2307)

It seems like the issue is with the import statement.

Since you've ported your JavaScript code to TypeScript, you should import the functions from the same file.

Here's the corrected import statement:

TypeScript

```
import { addStudent, changeFont, displayBrowserInfo, addStudentDummy } from './script';
```

Note that I've removed the `.ts` extension from the import statement. This is because TypeScript can automatically resolve the correct file extension.