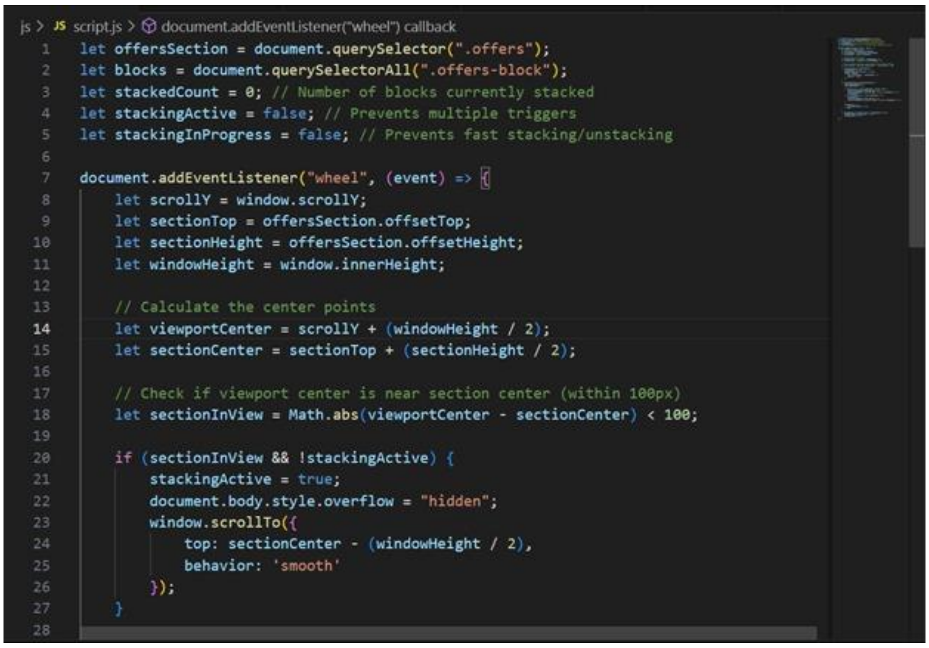
Stacking cards

Initial version

(see first iteration video button at LO3) ‘

I wanted to have a stacking card animation for my portfolio website to showcase the 5 Learning outcomes and what they represent.



js > script.js > document.addEventListener("wheel") callback

```javascript
let offersSection = document.querySelector(".offers");
let blocks = document.querySelectorAll(".offers-block");
let stackedCount = 0; // Number of blocks currently stacked
let stackingActive = false; // Prevents multiple triggers
let stackingInProgress = false; // Prevents fast stacking/unstacking

document.addEventListener("wheel", (event) => {
    let scrollY = window.scrollY;
    let sectionTop = offersSection.offsetTop;
    let sectionHeight = offersSection.offsetHeight;
    let windowHeight = window.innerHeight;

    // Calculate the center points
    let viewportCenter = scrollY + (windowHeight / 2);
    let sectionCenter = sectionTop + (sectionHeight / 2);

    // Check if viewport center is near section center (within 100px)
    let sectionInView = Math.abs(viewportCenter - sectionCenter) < 100;

    if (sectionInView && !stackingActive) {
        stackingActive = true;
        document.body.style.overflow = "hidden";
        window.scrollTo({
            top: sectionCenter - (windowHeight / 2),
            behavior: 'smooth'
        });
    }

```

Figure 1 Initial version

Feedback

The cards are currently lagging while scrolling and the user gets stuck when they want to scroll back. The animation also doesn’t completely work on a MacBook.

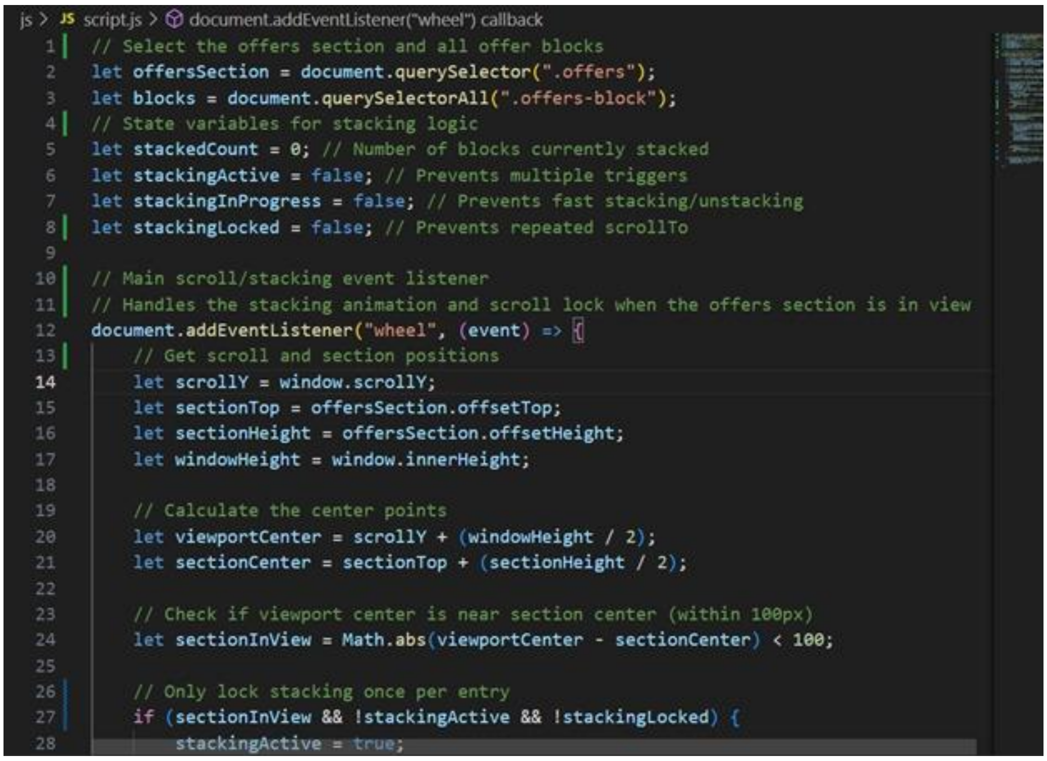
What I plan to do

- Fix the lagging
- Fix it so it does work on a MacBook
- Make sure the user doesn’t get stuck while scrolling back

Second version

(See second version video button)

This version is less laggy and gives the user an overall smoother experience.


js > script.js > document.addEventListener("wheel") callback

```
// Select the offers section and all offer blocks
let offersSection = document.querySelector(".offers");
let blocks = document.querySelectorAll(".offers-block");
// State variables for stacking logic
let stackedCount = 0; // Number of blocks currently stacked
let stackingActive = false; // Prevents multiple triggers
let stackingInProgress = false; // Prevents fast stacking/unstacking
let stackingLocked = false; // Prevents repeated scrollTo

// Main scroll/stacking event listener
// Handles the stacking animation and scroll lock when the offers section is in view
document.addEventListener("wheel", (event) => {
    // Get scroll and section positions
    let scrollY = window.scrollY;
    let sectionTop = offersSection.offsetTop;
    let sectionHeight = offersSection.offsetHeight;
    let windowHeight = window.innerHeight;

    // Calculate the center points
    let viewportCenter = scrollY + (windowHeight / 2);
    let sectionCenter = sectionTop + (sectionHeight / 2);

    // Check if viewport center is near section center (within 100px)
    let sectionInView = Math.abs(viewportCenter - sectionCenter) < 100;

    // Only lock stacking once per entry
    if (sectionInView && !stackingActive && !stackingLocked) {
        stackingActive = true;
```

Figure 2 Second version

Feedback

When scrolling back you don’t get stuck anymore but it still lags. This is especially noticeable on MacBooks.

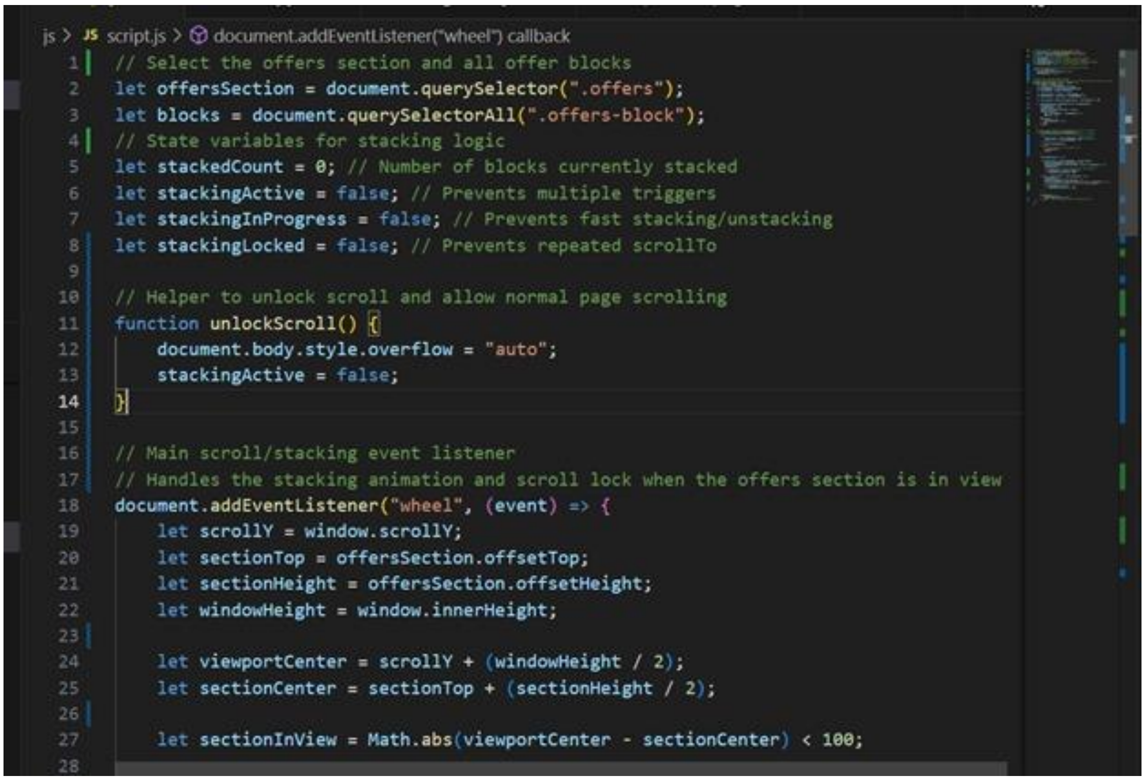
What I plan to do

- Fix the issues between the website and MacBook
- Make sure the scrolling back doesn’t lag anymore.

Final version

(see final version button video)

In this version it doesn’t lag anymore, the code is now compatible with MacBook and should not be lagging anymore there. This is because the part in the code that says ‘unlockScroll’.


js > script.js > document.addEventListener("wheel") callback

```javascript
// Select the offers section and all offer blocks
let offersSection = document.querySelector(".offers");
let blocks = document.querySelectorAll(".offers-block");
// State variables for stacking logic
let stackedCount = 0; // Number of blocks currently stacked
let stackingActive = false; // Prevents multiple triggers
let stackingInProgress = false; // Prevents fast stacking/unstacking
let stackingLocked = false; // Prevents repeated scrollTo

// Helper to unlock scroll and allow normal page scrolling
function unlockScroll() {
    document.body.style.overflow = "auto";
    stackingActive = false;
}

// Main scroll/stacking event listener
// Handles the stacking animation and scroll lock when the offers section is in view
document.addEventListener("wheel", (event) => {
    let scrollY = window.scrollY;
    let sectionTop = offersSection.offsetTop;
    let sectionHeight = offersSection.offsetHeight;
    let windowHeight = window.innerHeight;

    let viewportCenter = scrollY + (windowHeight / 2);
    let sectionCenter = sectionTop + (sectionHeight / 2);

    let sectionInView = Math.abs(viewportCenter - sectionCenter) < 100;
```

Figure 3 Final version

Reflection

For this project, I built a stacking card animation to show my learning outcomes. The first version had lag issues that were especially noticeable on MacBooks because of this ` users got stuck when scrolling back. Through multiple iterations, I improved performance and fixed compatibility issues.

In the final version, I solved the lag by adjusting the scroll-lock code, making the animation smooth and usable across devices. This helped me learn how to debug animations, improve performance, and test across different platforms.