

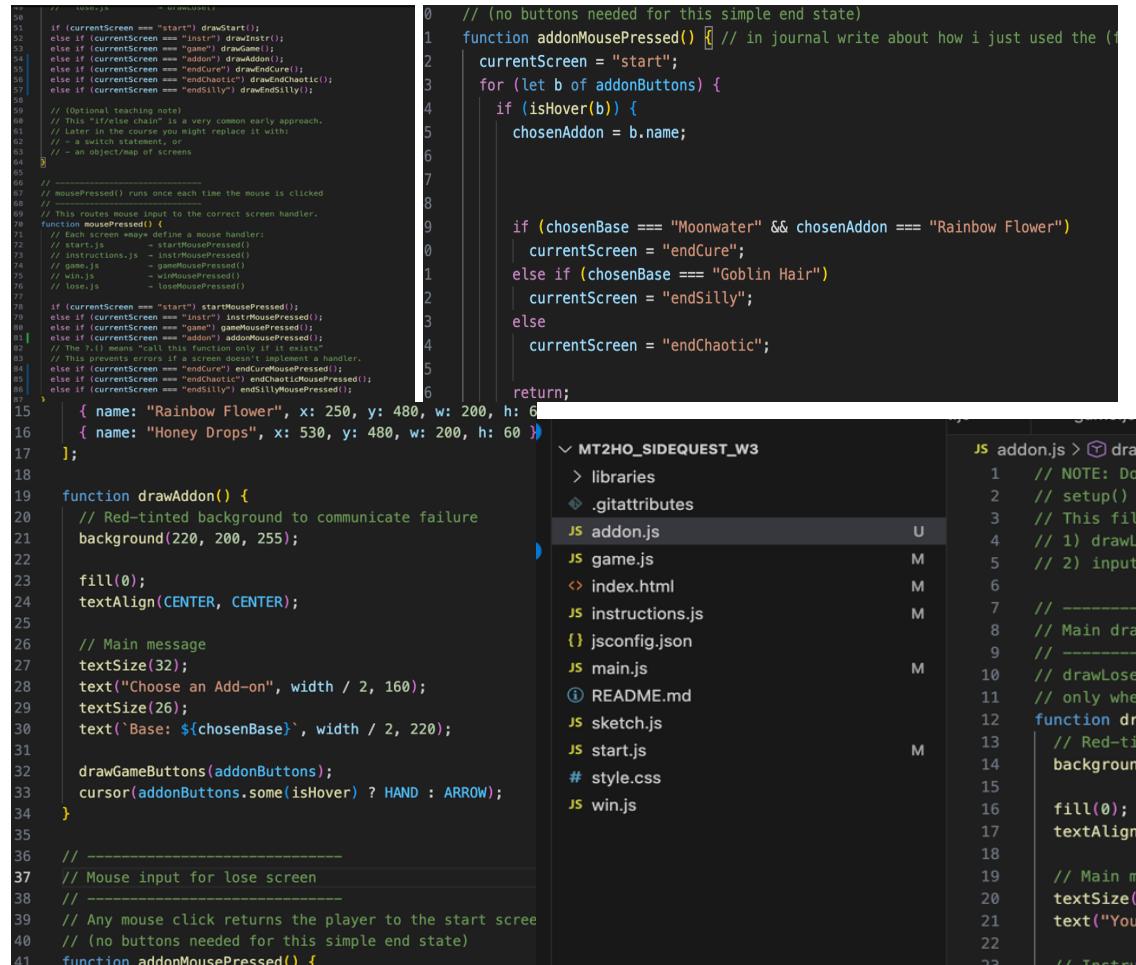
Process & Decision Documentation

Project/Assignment Decisions

Side Quests and A4 (Individual Work)

One key decision I made was that I wanted to initially be able to have several combinations of ingredients and a more interactive story that includes an NPC as the magic assistant, guiding the player. This required a file where all data and options were stored, as well as multiple graphics included on each screen had to be drawn, which was not able to do due to limited time. Instead, I simplified the code so that there were 6 different combinations. I also reduced the amount of endings I reduced the number of screens and states by mapping all outcomes to three shared ending screens (endSilly, endChaotic, endCure). I also used for loops and direct conditional logic (if/else), functions that I have some knowledge of to do the different endings and draw buttons.

Role-Based Process Evidence



```
0 // (no buttons needed for this simple end state)
1 function addonMousePressed() { // in journal write about how i just used the (if
2   currentScreen = "start";
3   for (let b of addonButtons) {
4     if (isHover(b)) {
5       chosenAddon = b.name;
6
7       if (chosenBase === "Moonwater" && chosenAddon === "Rainbow Flower")
8         currentScreen = "endCure";
9       else if (chosenBase === "Goblin Hair")
10         currentScreen = "endSilly";
11       else
12         currentScreen = "endChaotic";
13
14     return;
15   }
16 }
17
18 function drawAddon() {
19   // Red-tinted background to communicate failure
20   background(220, 200, 255);
21
22   fill(0);
23   textAlign(CENTER, CENTER);
24
25   // Main message
26   textSize(32);
27   text("Choose an Add-on", width / 2, 160);
28   textSize(26);
29   text(`Base: ${chosenBase}`, width / 2, 220);
30
31   drawGameButtons(addonButtons);
32   cursor(addonButtons.some(isHover) ? HAND : ARROW);
33 }
34
35
36 // -----
37 // Mouse input for lose screen
38 // -----
39 // Any mouse click returns the player to the start screen
40 // (no buttons needed for this simple end state)
41 function addonMousePressed() {
```

```
JS addon.js > draw
1 // NOTE: Do
2 // setup()
3 // This file
4 // 1) drawL
5 // 2) input
6
7 // -----
8 // Main draw
9 // -----
10 // drawLose
11 // only when
12 function dr
13 // Red-ti
14 background
15
16 fill(0);
17 textAlign
18
19 // Main m
20 textSize(
21 text("You
22 // Testin
```

GenAI Documentation

GenAI was used for this project:

Date Used: Feb 1 2026

Tool Disclosure: ChatGPT 5.2

Purpose of Use: I used ChatGPT to debug and help guide me how to do specific functions/tasks

Summary of Interaction: I prompted ChatGPT to help fix my code, particularly debugging my for loop" as I forgot to assigned something to [i]. I suggested functions to use like the interpolation expression \${chosenBase}. I made ChatGPT explain to me how these methods worked in detailed. Lastly, they helped me generate some pixel art for the ingredients.

Human Decision Point(s): When I needed some to help to debug my code so that the screening change would work from base ingredients to add-on ingredients, they made all the functions and draw happen on the same game.js file. I didn't take this advice, and separated each screen into multiple files to easily transition the screens.

Integrity & Verification Note: I cross checked by going through our coding tutorial slides and checked if the information ChatGPT provided was accurate.

Scope of GenAI Use:

- Gen ai was used as a debugging assistance, specifically to help identify why ingredient buttons were not rendering correctly. This included pointing out missing variable declarations and forgetting to declare buttons with const.) It was also used for general coding guidance, such as clarifying JavaScript syntax or logic when errors occurred.
- GenAI was not used for ② Designing the visual layout or UI structure choosing background colours, typography, or text sizing, creating or positioning buttons and images, structuring the project into multiple files/screens
- Writing the core draw() logic in main.js

Limitations or Misfires: The tool didn't understand that there are some concepts we didn't learn yet, and used overcomplicated codes.

Summary of Process (Human + Tool)

Describe what you did, focusing on process rather than outcome. This may include:

The parts I did not use AI on was changing the background colours, text sizes/fonts, creating the buttons and images layout, as well as creating the different files/screens and coding them under function draw() in the main.js file. I loaded in the images and laid them out. I added in the buttons for base ingredients and add-on ingredients, as well as fixing errors for drawing the ingredient button. AI was interfered here because I forgot to include add const to the buttons.

Decision Points & Trade-offs

Describe one or two key decisions you made:

- I used a for-loop to draw all three buttons because it was easier for me to understand than other methods like (for let) as I have used for loops before
- Initially, I had the images embedded with the const buttons and drawing them directly in the same loop, this causes lots of layering issues. I decided to preload and draw images independently in the screen's main draw function (drawGame() / drawAddon())

Verification & Judgement

I verified the code by replaying the code many times, making sure that the text is readable and there are clear visual cues like keeping the hover, and on-screen instructions, or different colours good or bad endings. I also got peer feedback and explained the code myself so if we both understand it, then it's clear.

Limitations, Dead Ends, or Open Questions

- The ending results were not that varied and used the same text for 3 different endings, but there were around 6 different combinations.
- I wanted to make an ending for each of them, but that most likely required a different approach, and more graphics to be made, which I did not have much time.

Appendix

Me: I'm working on a potion game in p5.js. I stored data in an array of objectives so it's easier for me to call upon and draw 3 of the buttons at the sametime (drawGameButtons(basebuttons)). But for some reason the buttons aren't responding.

```
baseButtons = [
  { name: "Dragon Scale", x: 200, y: 420, w: 180, h: 60 },
  { name: "Moonwater", x: 400, y: 420, w: 180, h: 60 },
  { name: "Goblin Hair", x: 600, y: 420, w: 180, h: 60 }
];
```

ChatGPT: Yes — you're missing a declaration keyword. You should define this with const or let. Without it, JavaScript won't properly scope the variable.

```
const baseButtons = [
  { name: "Dragon Scale", x: 200, y: 420, w: 180, h: 60 },
  { name: "Moonwater", x: 400, y: 420, w: 180, h: 60 },
  { name: "Goblin Hair", x: 600, y: 420, w: 180, h: 60 }
];
```

Me: I realize that the user needs to know what ingredient they picked before, and have it labeled underneath. is there a function that keeps track of what base ingredient the user picks previously every time?

```
function drawAddon() {
  // Red-tinted background to communicate failure
  background(220, 200, 255);

  fill(0);
  textAlign(CENTER, CENTER);

  // Main message
  textSize(32);
  text("Choose an Add-on", width / 2, 160);
```

ChatGPT: You can do that by storing the user's choice in a variable and then displaying it. Here's an example snippet:

```
Chatgpt: textSize(32);
text("Choose an Add-on", width / 2, 180);
textSize(18);
text(`Base: ${chosenBase}` , width / 2, 230);

drawButtons(addonButtons);
cursor(addonButtons.some(isHover) ? HAND : ARROW);
}
```

Me: I tried using a for loop to create the 3 buttons , but the buttons aren't showing up.

```
function drawGameButtons(buttons) {
  rectMode(CENTER);
  textAlign(CENTER, CENTER);

  for (let i = 0; i < buttons.length; i++) {

    fill(isHover(b) ? 200 : 255);
    stroke(0);
    rect(b.x, b.y, b.w, b.h, 12);
```

```

    fill(0);
    noStroke();
    textSize(18);
    text(b.name, b.x, b.y);
}
}

```

ChatGPT: Ahh you're *very* close — this is a classic for-loop + variable mix-up  Your logic is right, but the buttons aren't showing because **you never define b inside the loop**.

Right now you have:

```

for (let i = 0; i < buttons.length; i++) {

    fill(isHover(b) ? 200 : 255);
    rect(b.x, b.y, b.w, b.h, 12);
    text(b.name, b.x, b.y);
}

```

apr

The fix

Inside the loop, you need to grab the current button from the array:

```

function drawGameButtons(buttons) {
    rectMode(CENTER);
    textAlign(CENTER, CENTER);

    for (let i = 0; i < buttons.length; i++) {
        let b = buttons[i]; // ← THIS LINE

        fill(isHover(b) ? 200 : 255);
        stroke(0);
        rect(b.x, b.y, b.w, b.h, 12);

        fill(0);
        noStroke();
        textSize(18);
        text(b.name, b.x, b.y);
    }
}

```

can you generate images game icons of a rainbow flower, dragon scale, goblin hair, honey drops, and moonwater? make them kind of video game style illustrative/ maybe a bit pixelated



me: I tried loading images inside basegame.js like this, but after I did that, my game stopped moving past the start screen.

```
function preloadBaseGameImages() {  
  dragonImg = loadImage("assets/ dragonscale.png");  
}
```

ChatGPT: That's because p5.js only supports **one preload()**, and it needs to be in main.js. Other files can *use* the images, but not preload them.

```
function preload() {  
  
  dragonImg = loadImage("assets/ragonscale.png");  
  
  moonwaterImg = loadImage("assets/moonwater.png");  
  
  goblinImg = loadImage("assets/goblin.png");  
  
}
```

me: That's is exactly what I did, but it still not loading the images, do I need to add images after that?

Chatgpt: Ahhh got it — you're asking about **folder structure**, not code 🤓

Short answer: **no, you don't have to** use assets/images, **but your path MUST match your folders exactly.**

✓ Valid setups (pick ONE)

Option 1: Images directly in `assets/` project-folder/

```
index.html
main.js
start.js
assets/
  dragonscale.png
  moonwater.png
  goblin.png
```

Then in `preload()`:

```
loadImage("assets/dragonscale.png");
```

Option 2: Images inside `assets/images/` project-folder/

```
index.html
main.js
start.js
assets/
  images/
    dragonscale.png
    moonwater.png
    goblin.png
```

Then you **must** do:

```
loadImage("assets/images/dragonscale.png");
```

Me: Nothing shows up still.

ChatGPT: Alright, let's debug this calmly and **surgically**—when images don't show in `p5.js`, it's almost always **one of 4 things**, and we can rule them out fast.

I'll walk you through this in the **exact order that catches the bug 99% of the time**.

✓ 1. First: confirm your folder + path (this is the #1 cause)

You said your images are in:

```
project-folder/
  index.html
```

```
└── sketch.js
└── start.js
└── game.js
└── assets/
    └── images/
        ├── dragon.png
        ├── moonwater.png
        └── goblin.png
```

👉 Then your paths **MUST** be:

"assets/images/dragon.png"

✗ NOT /assets/images/dragon.png
✗ NOT ../assets/images/dragon.png

✓ 2. Images **MUST** be loaded in `preload()` (one place only)

In `sketch.js` (or whatever file has `setup()` and `draw()`):

```
let dragonImg, moonwaterImg, goblinImg;

function preload() {
  dragonImg = loadImage("assets/dragon.png");
  moonwaterImg = loadImage("assets/moonwater.png");
  goblinImg = loadImage("assets/goblin.png");
}
```

⚠ If this is not in the file that actually runs `setup()` → images will be `undefined`.

✓ 3. Buttons must store the IMAGE VARIABLE, not a string

This is a super common silent bug.

✓ Correct

```
const baseButtons = [
  { name: "Dragon Scale", x: 200, y: 420, w: 180, h: 60, img: dragonImg },
  { name: "Moonwater", x: 400, y: 420, w: 180, h: 60, img: moonwaterImg },
  { name: "Goblin Hair", x: 600, y: 420, w: 180, h: 60, img: goblinImg }
];
```

✗ Wrong (won't draw)

```
img: "assets/images/dragon.png"
```

✓ 4. Your draw function is almost right — add a safety check

Update your image loop slightly so we can *see* if the image exists:

```

for (let i = 0; i < buttons.length; i++) {
  let b = buttons[i];

  if (b.img !== undefined) {
    image(b.img, b.x, b.y - 90, 100, 100);
  }
}

```

If `b.img` is undefined, **nothing will draw and no error appears** — p5 fails silently.

Human decision point(me): I ended not using suggestion above because it was made everything extremely buggy, unnecessary, and nothing was loading. I just did it the easy way and loaded the game images individually, and loaded it under function `drawgame()`. I also found out the reason the images weren't loading because I didn't write down the entire folder structure when reloading images, I had to make it assets/images.

```

function drawGame() {
  // Set background colour for the game screen
  background(240, 230, 140);

  // ---- Title and instructions text ----
  fill(0); // black text
  textSize(32);
  textAlign(CENTER, CENTER);
  text("Choose a base ingredient.", width / 2, 160);

  imageMode(CENTER);
  image(dragonImg, 200, 320, 100, 100);
  image(moonwaterImg, 400, 320, 100, 100);
  image(goblinImg, 600, 320, 100, 100);
  // ---- Draw the button ----
  // We pass the button object to a helper function

  drawGameButtons(baseButtons);

  // ---- Cursor feedback ----
  // If the mouse is over the button, show a hand cursor
  // Otherwise, show the normal arrow cursor
  cursor(baseButtons.some(isHover) ? HAND : ARROW);
}

```

