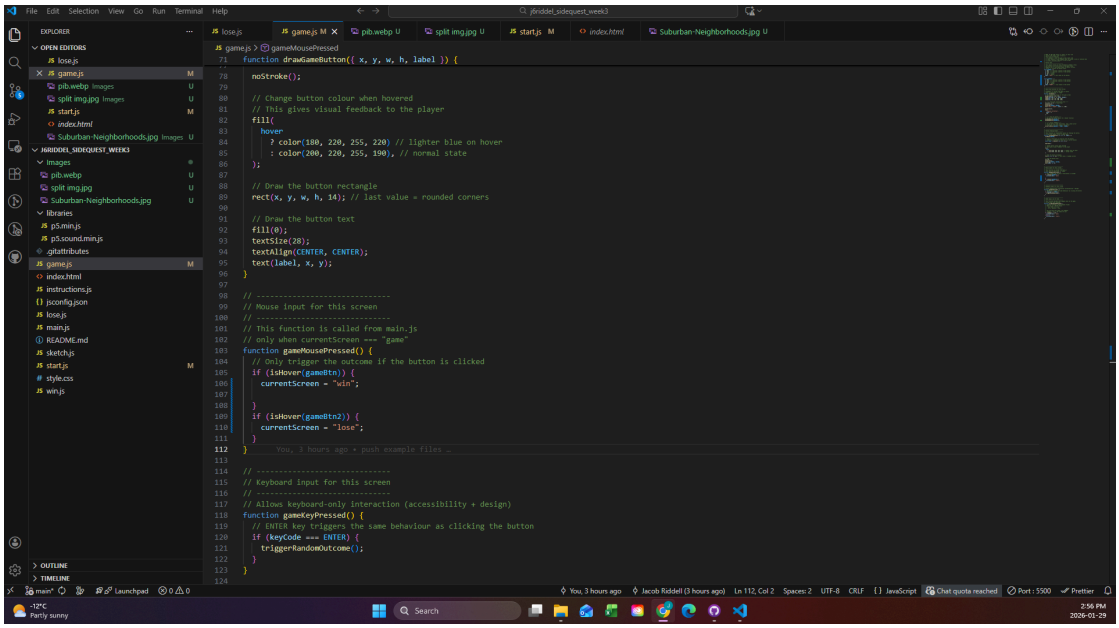
Process & Decision Documentation

Project/Assignment Decisions

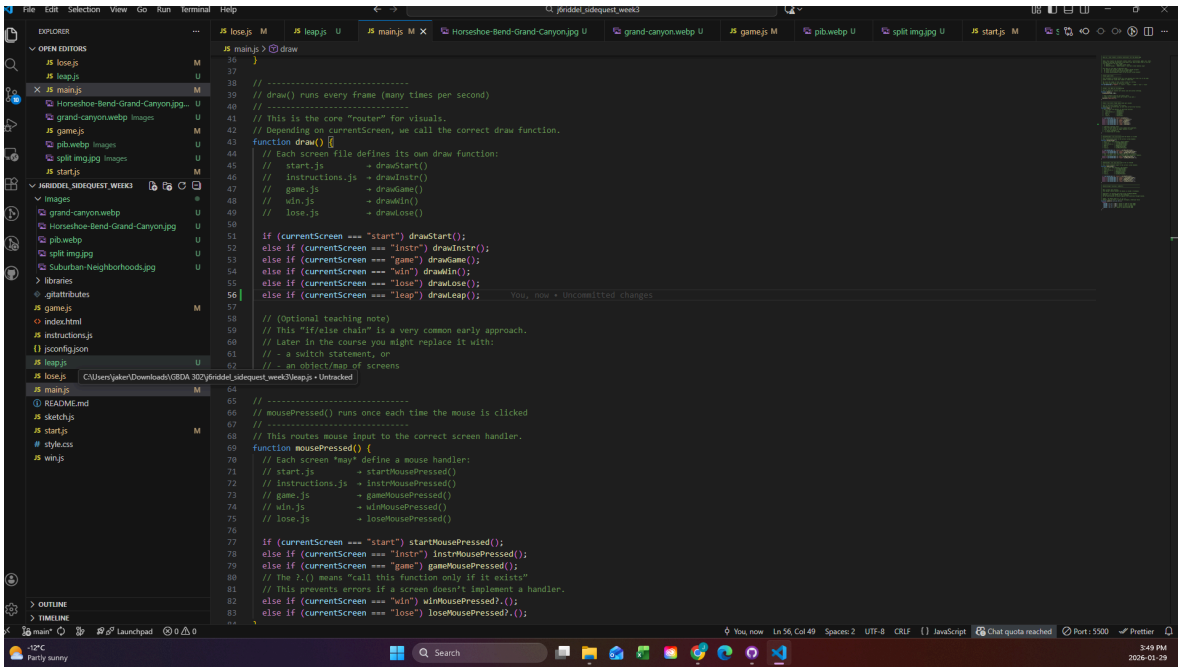
Side Quest Week 3

I added files and changed function naming and script calling to create a multi-layered story. The user has “right”, “wrong” and “retreat” selections. I added images and text to create comedic effects and moments.

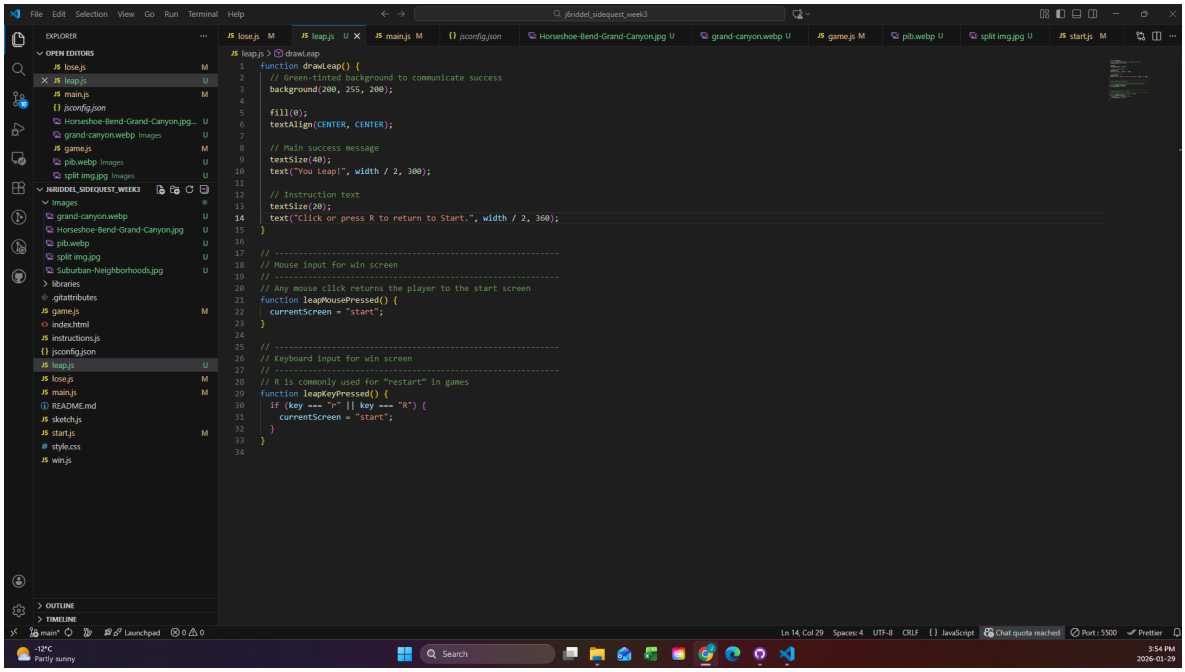
Role-Based Process Evidence



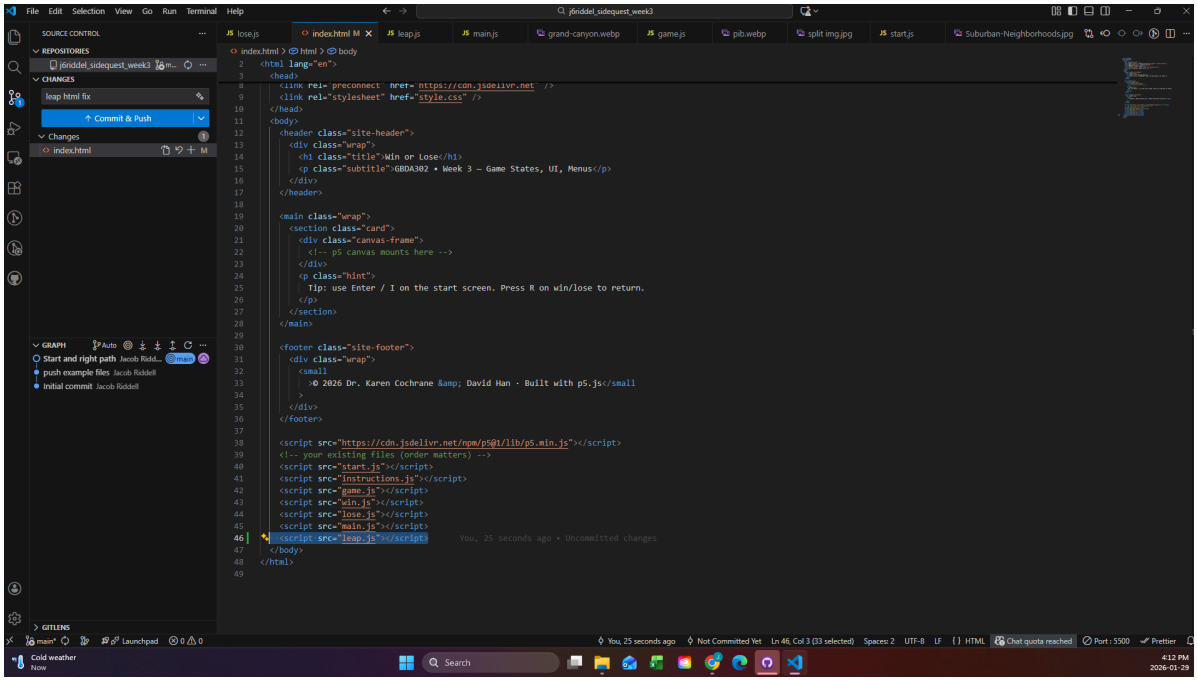
```
function drawGameButton({ x, y, w, h, label }) {

  noStroke();

  // Change button colour when hovered
  // This gives visual feedback to the player
  fill(
    hover
      ? color(180, 220, 255, 220) // lighter blue on hover
      : color(200, 220, 255, 190), // normal state
  );

  // Draw the button rectangle
  rect(x, y, w, h, 14); // last value = rounded corners

  // Draw the button text
  fill(0);
  textSize(28);
  textAlign(CENTER, CENTER);
  text(label, x, y);
}

// ----------------------------
// Mouse input for this screen
// ----------------------------
// This function is called from main.js
// only when currentScreen === "game"
function gameMousePressed() {
  // Only trigger the outcome if the button is clicked
  if (isHover(gameBtn)) {
    currentScreen = "win";

  }
  if (isHover(gameBtn2)) {
    currentScreen = "lose";
  }
}       You, 3 hours ago • push example files …

// ----------------------------
// Keyboard input for this screen
// ----------------------------
// Allows keyboard-only interaction (accessibility + design)
function gameKeyPressed() {
  // ENTER key triggers the same behaviour as clicking the button
  if (keyCode === ENTER) {
    triggerRandomOutcome();
  }
}
```



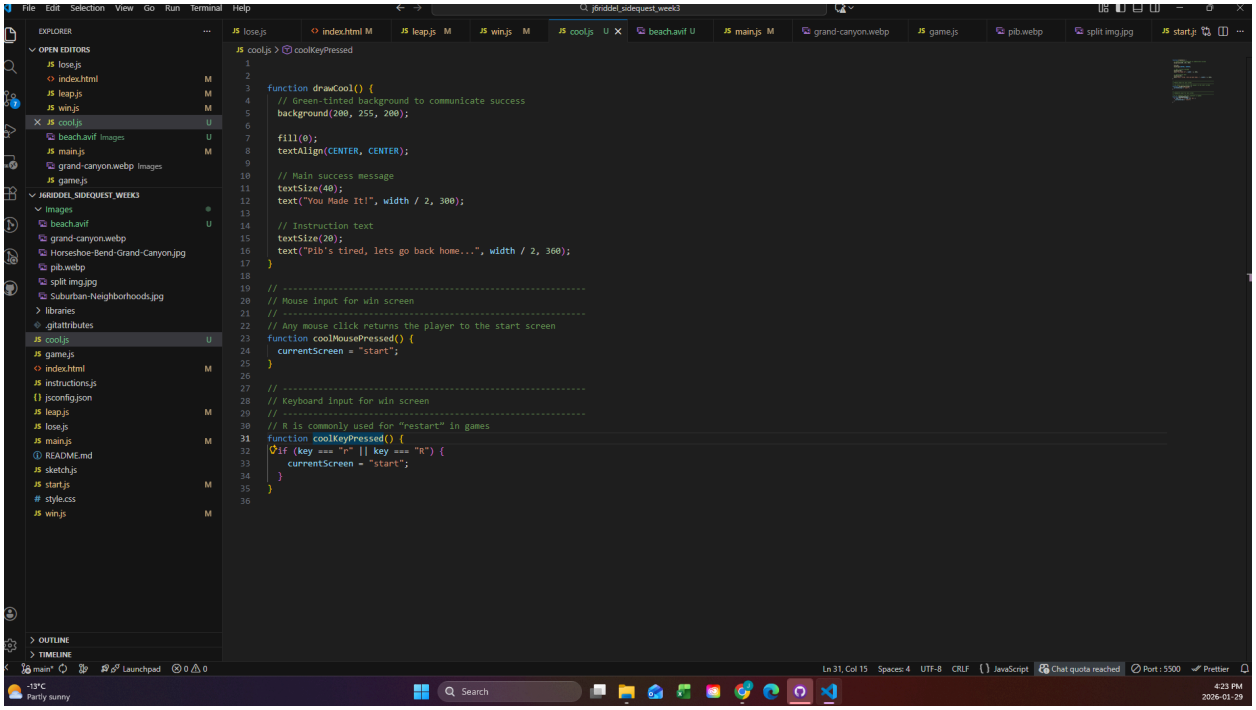
```
}

// ----------------------------
// draw() runs every frame (many times per second)
// ----------------------------
// This is the core "router" for visuals.
// Depending on currentScreen, we call the correct draw function.
function draw() {
  // Each screen file defines its own draw function:
  //   start.js         → drawStart()
  //   instructions.js  → drawInstr()
  //   game.js          → drawGame()
  //   win.js           → drawWin()
  //   lose.js          → drawLose()

  if (currentScreen === "start") drawStart();
  else if (currentScreen === "instr") drawInstr();
  else if (currentScreen === "game") drawGame();
  else if (currentScreen === "win") drawWin();
  else if (currentScreen === "lose") drawLose();
  else if (currentScreen === "leap") drawLeap();      You, now • Uncommitted changes

  // (Optional teaching note)
  // This "if/else chain" is a very common early approach.
  // Later in the course you might replace it with:
  // - a switch statement, or
  // - an object/map of screens
}

// ----------------------------
// mousePressed() runs once each time the mouse is clicked
// ----------------------------
// This routes mouse input to the correct screen handler.
function mousePressed() {
  // Each screen *may* define a mouse handler:
  // start.js         → startMousePressed()
  // instructions.js  → instrMousePressed()
  // game.js          → gameMousePressed()
  // win.js           → winMousePressed()
  // lose.js          → loseMousePressed()

  if (currentScreen === "start") startMousePressed();
  else if (currentScreen === "instr") instrMousePressed();
  else if (currentScreen === "game") gameMousePressed();
  // The ?.() means "call this function only if it exists"
  // This prevents errors if a screen doesn't implement a handler.
  else if (currentScreen === "win") winMousePressed?.();
  else if (currentScreen === "lose") loseMousePressed?.();
}
```



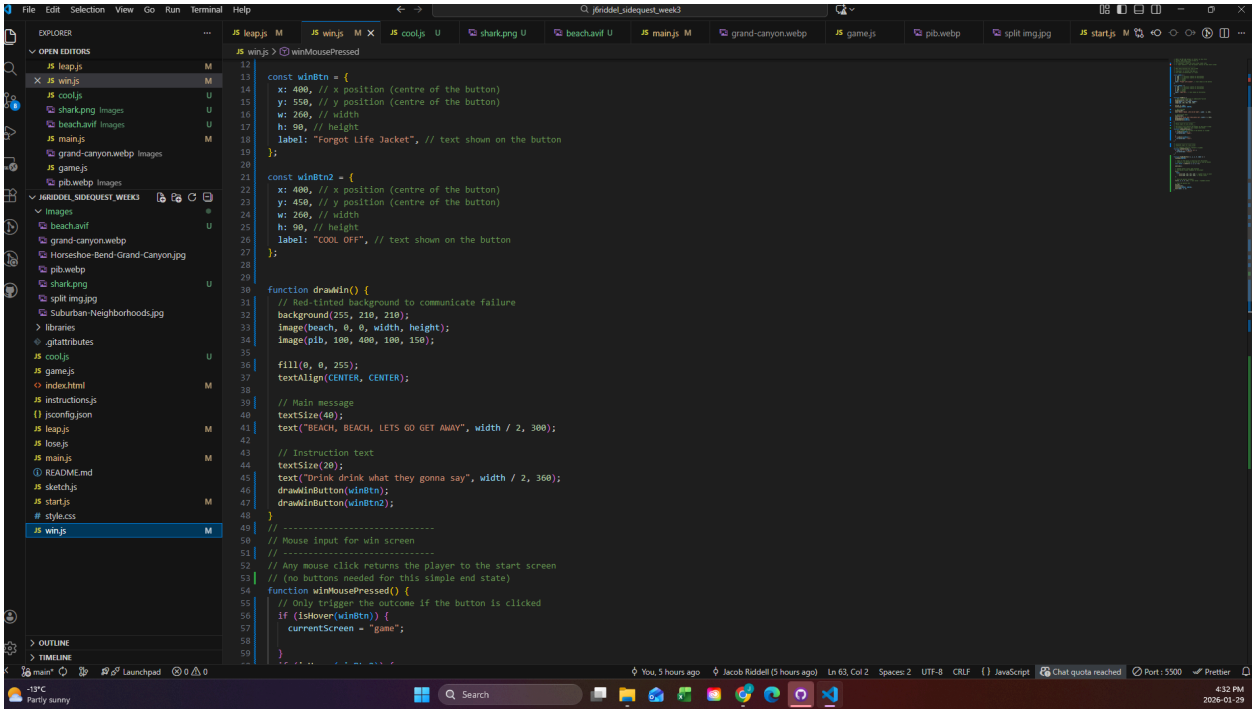
```
function drawLeap() {
  // Green-tinted background to communicate success
  background(200, 255, 200);

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

  // Main success message
  textSize(40);
  text("You Leap!", width / 2, 300);

  // Instruction text
  textSize(20);
  text("Click or press R to return to Start.", width / 2, 360);
}

// ---------------------------------------------------------
// Mouse input for win screen
// ---------------------------------------------------------
// Any mouse click returns the player to the start screen
function leapMousePressed() {
  currentScreen = "start";
}

// ---------------------------------------------------------
// Keyboard input for win screen
// ---------------------------------------------------------
// R is commonly used for "restart" in games
function leapKeyPressed() {
  if (key === "r" || key === "R") {
    currentScreen = "start";
  }
}
```



```
<!DOCTYPE html>
<html lang="en">
  <head>
    <link rel="preconnect" href="https://cdn.jsdelivr.net" />
    <link rel="stylesheet" href="style.css" />
  </head>
  <body>
    <header class="site-header">
      <div class="wrap">
        <h1 class="title">Win or Lose</h1>
        <p class="subtitle">6BDA302 • Week 3 — Game States, UI, Menus</p>
      </div>
    </header>

    <main class="wrap">
      <section class="card">
        <div class="canvas-frame">
          <!-- p5 canvas mounts here -->
        </div>
        <p class="hint">
          Tip: use Enter / I on the start screen. Press R on win/lose to return.
        </p>
      </section>
    </main>

    <footer class="site-footer">
      <div class="wrap">
        <small>
          >© 2026 Dr. Karen Cochrane &amp;amp; David Han · Built with p5.js</small
        >
      </div>
    </footer>

    <script src="https://cdn.jsdelivr.net/npm/p5@1/lib/p5.min.js"></script>
    <!-- your existing files (order matters) -->
    <script src="start.js"></script>
    <script src="instructions.js"></script>
    <script src="game.js"></script>
    <script src="win.js"></script>
    <script src="lose.js"></script>
    <script src="main.js"></script>
    <script src="leap.js"></script>
  </body>
</html>
```



```
function drawCool() {
  // Green-tinted background to communicate success
  background(200, 255, 200);

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

  // Main success message
  textSize(40);
  text("You Made It!", width / 2, 300);

  // Instruction text
  textSize(20);
  text("Pib's tired, lets go back home...", width / 2, 360);
}

// ---------------------------------------------------------
// Mouse input for win screen
// ---------------------------------------------------------
// Any mouse click returns the player to the start screen
function coolMousePressed() {
  currentScreen = "start";
}

// ---------------------------------------------------------
// Keyboard input for win screen
// ---------------------------------------------------------
// R is commonly used for "restart" in games
function coolKeyPressed() {
  if (key === "r" || key === "R") {
    currentScreen = "start";
  }
}
```



```
const winBtn = {
  x: 400, // x position (centre of the button)
  y: 550, // y position (centre of the button)
  w: 260, // width
  h: 90, // height
  label: "Forgot Life Jacket", // text shown on the button
};

const winBtn2 = {
  x: 400, // x position (centre of the button)
  y: 450, // y position (centre of the button)
  w: 260, // width
  h: 90, // height
  label: "COOL OFF", // text shown on the button
};


function drawWin() {
  // Red-tinted background to communicate failure
  background(255, 210, 210);
  image(beach, 0, 0, width, height);
  image(pib, 100, 400, 100, 150);

  fill(0, 0, 255);
  textAlign(CENTER, CENTER);

  // Main message
  textSize(40);
  text("BEACH, BEACH, LETS GO GET AWAY", width / 2, 300);

  // Instruction text
  textSize(20);
  text("Drink drink what they gonna say", width / 2, 360);
  drawWinButton(winBtn);
  drawWinButton(winBtn2);
}
// ----------------------------
// Mouse input for win screen
// ----------------------------
// Any mouse click returns the player to the start screen
// (no buttons needed for this simple end state)
function winMousePressed() {
  // Only trigger the outcome if the button is clicked
  if (isHover(winBtn)) {
    currentScreen = "game";
  }
```

Entry Header

Name: Jake Riddell

Role: Programmer

Goal of Work Session

I was working on the basic mechanics and game structure of multi file code. So how to build, integrate and call functions that exist in different places.

Tools, Resources, or Inputs Used

- https://p5js.org/reference/
- p5.js
- Vs code
- Github
- GBDA302 • Week 3 — Game States, UI, Menus Template

GenAI Documentation

No GenAI use