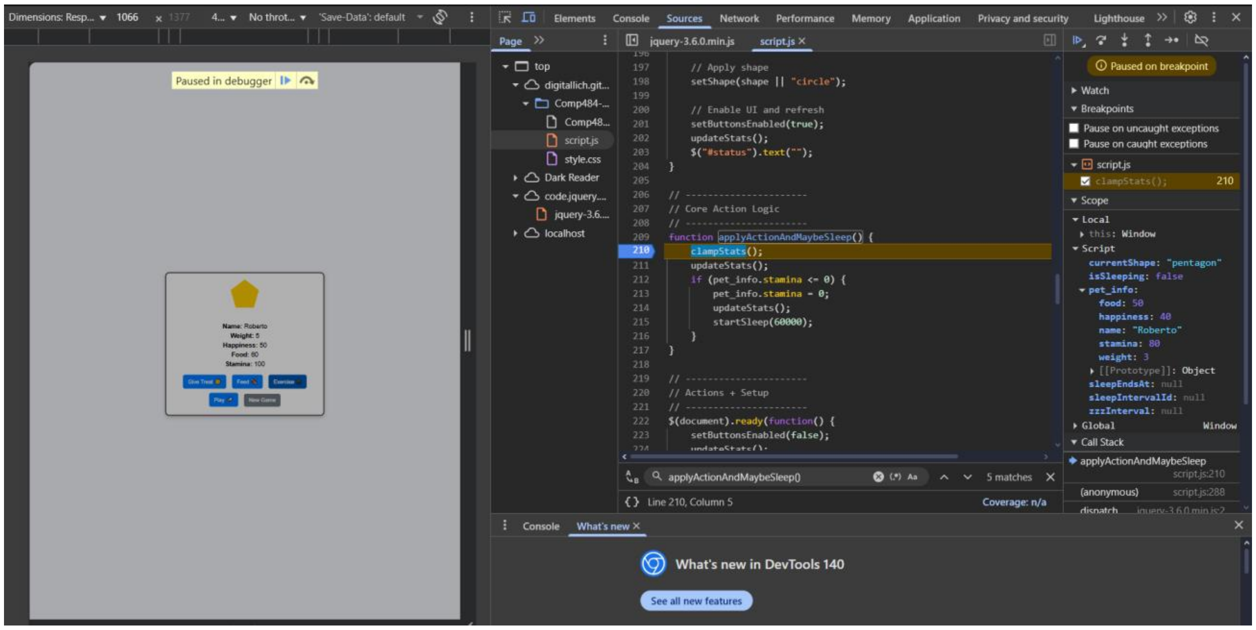
Homework 10

Devtools

I set a breakpoint in the applyActionAndMaybeSleep() function in script.js. When I clicked the Exercise button, the execution paused before clamping the stats. In the Scope panel I inspected pet_info and saw stamina and happiness values change after each click. This helped me understand how the game decides when to put the pet to sleep.



```
    // Apply shape
    setShape(shape || "circle");

    // Enable UI and refresh
    setButtonsEnabled(true);
    updateStats();
    $("#status").text("");
}

// ---------------------
// Core Action Logic
// ---------------------
function applyActionAndMaybeSleep() {
    clampStats();
    updateStats();
    if (pet_info.stamina <= 0) {
        pet_info.stamina = 0;
        updateStats();
        startSleep(60000);
    }
}

// ---------------------
// Actions + Setup
// ---------------------
$(document).ready(function() {
    setButtonsEnabled(false);
```

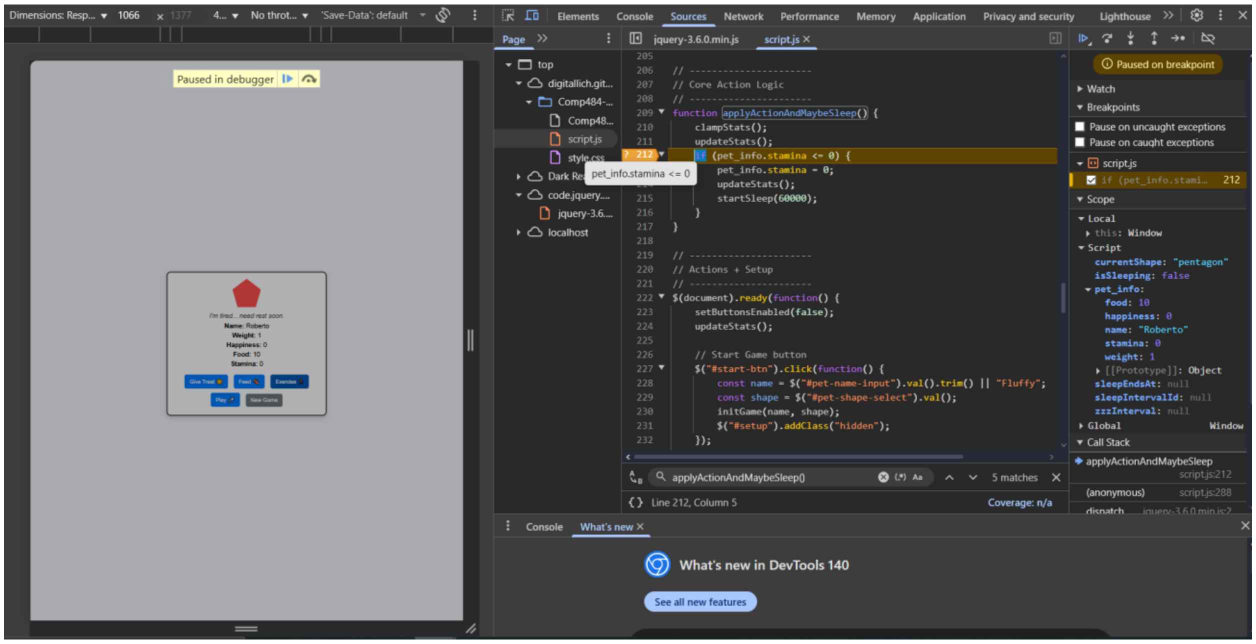
applyActionAndMaybeSleep() — 5 matches

Line 210, Column 5 — Coverage: n/a

Pause on uncaught exceptions
Pause on caught exceptions
script.js: clampStats(); 210

```
Scope
  Local
    this: Window
  Script
    currentShape: "pentagon"
    isSleeping: false
    pet_info:
      food: 50
      happiness: 40
      name: "Roberto"
      stamina: 80
      weight: 3
      [[Prototype]]: Object
    sleepEndsAt: null
    sleepIntervalId: null
    zzzInterval: null
  Global                Window
Call Stack
  applyActionAndMaybeSleep    script.js:210
  (anonymous)                 script.js:288
  dispatch                    jquery-3.6.0.min.js:2
```

I added a conditional breakpoint on the stamina check using pet_info.stamina <= 0. This made DevTools pause only when stamina dropped to zero. It let me debug the exact moment the pet falls asleep without stopping on every call, demonstrating how conditional breakpoints can focus debugging on specific states.



```
// ---------------------
// Core Action Logic
// ---------------------
function applyActionAndMaybeSleep() {
    clampStats();
    updateStats();
    if (pet_info.stamina <= 0) {
        pet_info.stamina = 0;
        updateStats();
        startSleep(60000);
    }
}

// ---------------------
// Actions + Setup
// ---------------------
$(document).ready(function() {
    setButtonsEnabled(false);
    updateStats();

    // Start Game button
    $("#start-btn").click(function() {
        const name = $("#pet-name-input").val().trim() || "Fluffy";
        const shape = $("#pet-shape-select").val();
        initGame(name, shape);
        $("#setup").addClass("hidden");
    });
```

pet_info.stamina <= 0

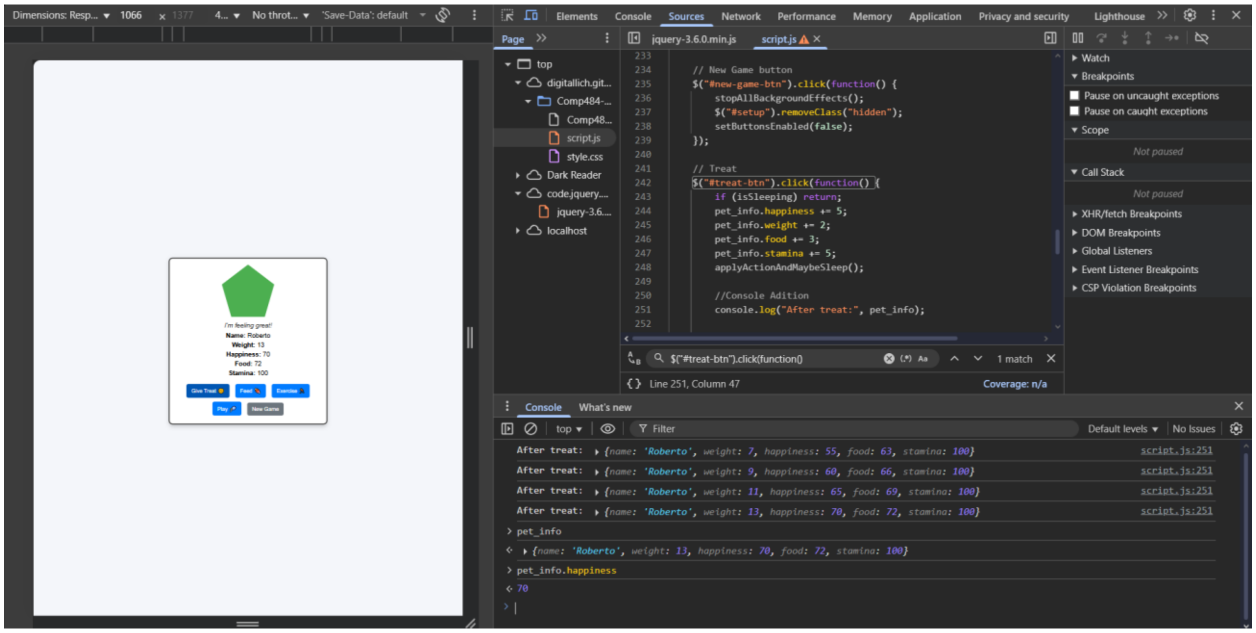
applyActionAndMaybeSleep() — 5 matches

Line 212, Column 5 — Coverage: n/a

Pause on uncaught exceptions
Pause on caught exceptions
script.js: if (pet_info.stami… 212

```
Scope
  Local
    this: Window
  Script
    currentShape: "pentagon"
    isSleeping: false
    pet_info:
      food: 10
      happiness: 0
      name: "Roberto"
      stamina: 0
      weight: 1
      [[Prototype]]: Object
    sleepEndsAt: null
    sleepIntervalId: null
    zzzInterval: null
  Global                Window
Call Stack
  applyActionAndMaybeSleep    script.js:212
  (anonymous)                 script.js:288
  dispatch                    jquery-3.6.0.min.js:2
```

I added a console.log("After treat:", pet_info) to the Treat button handler. Every time I clicked the Treat button, the Console showed the updated pet_info. I also evaluated pet_info and pet_info.happiness directly in the Console to inspect the state of my game without adding more UI elements.



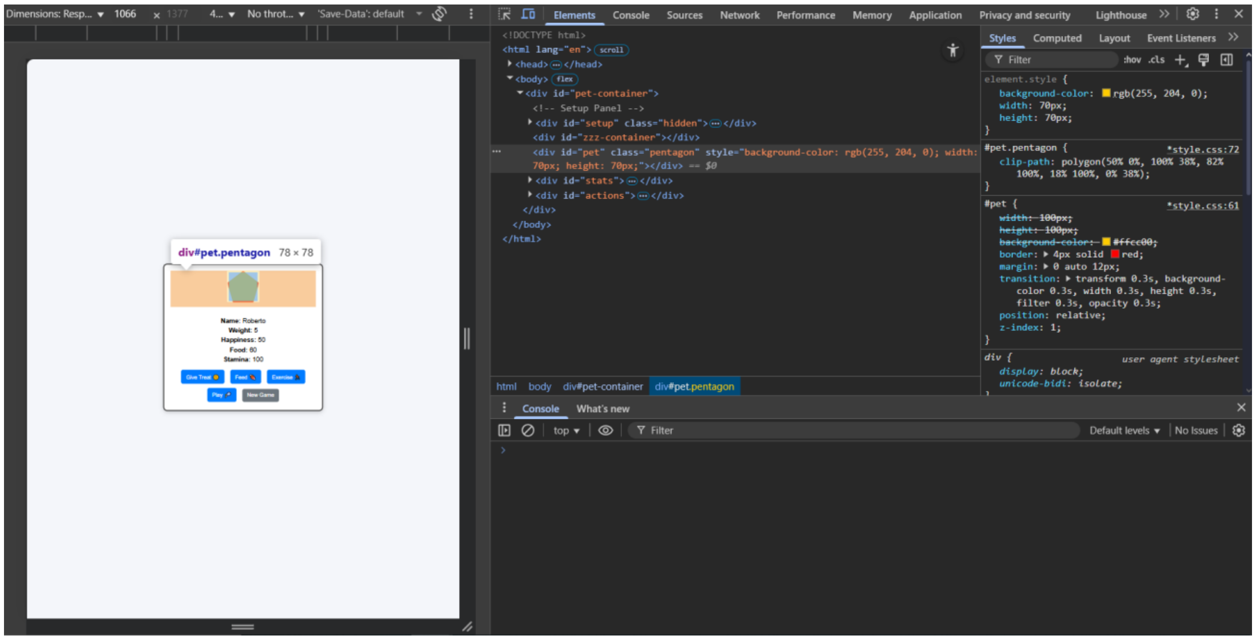
```
233
234     // New Game button
235     $("#new-game-btn").click(function() {
236         stopAllBackgroundEffects();
237         $("#setup").removeClass("hidden");
238         setButtonsEnabled(false);
239     });
240
241     // Treat
242     $("#treat-btn").click(function() {
243         if (isSleeping) return;
244         pet_info.happiness += 5;
245         pet_info.weight += 2;
246         pet_info.food += 3;
247         pet_info.stamina += 5;
248         applyActionAndMaybeSleep();
249
250         //Console Adition
251         console.log("After treat:", pet_info);
252
```

```
$("#treat-btn").click(function()    1 match
Line 251, Column 47    Coverage: n/a
```

```
After treat:  ▸ {name: 'Roberto', weight: 7, happiness: 55, food: 63, stamina: 100}     script.js:251
After treat:  ▸ {name: 'Roberto', weight: 9, happiness: 60, food: 66, stamina: 100}     script.js:251
After treat:  ▸ {name: 'Roberto', weight: 11, happiness: 65, food: 69, stamina: 100}    script.js:251
After treat:  ▸ {name: 'Roberto', weight: 13, happiness: 70, food: 72, stamina: 100}    script.js:251
> pet_info
<  ▸ {name: 'Roberto', weight: 13, happiness: 70, food: 72, stamina: 100}
> pet_info.happiness
< 70
>
```

DOM

Using the Elements tab, I inspected the <div id="pet"> element and live-edited its styles. I changed the background color, size, and added a border. This showed how I can visually tweak and debug CSS in the browser before changing my actual stylesheet.

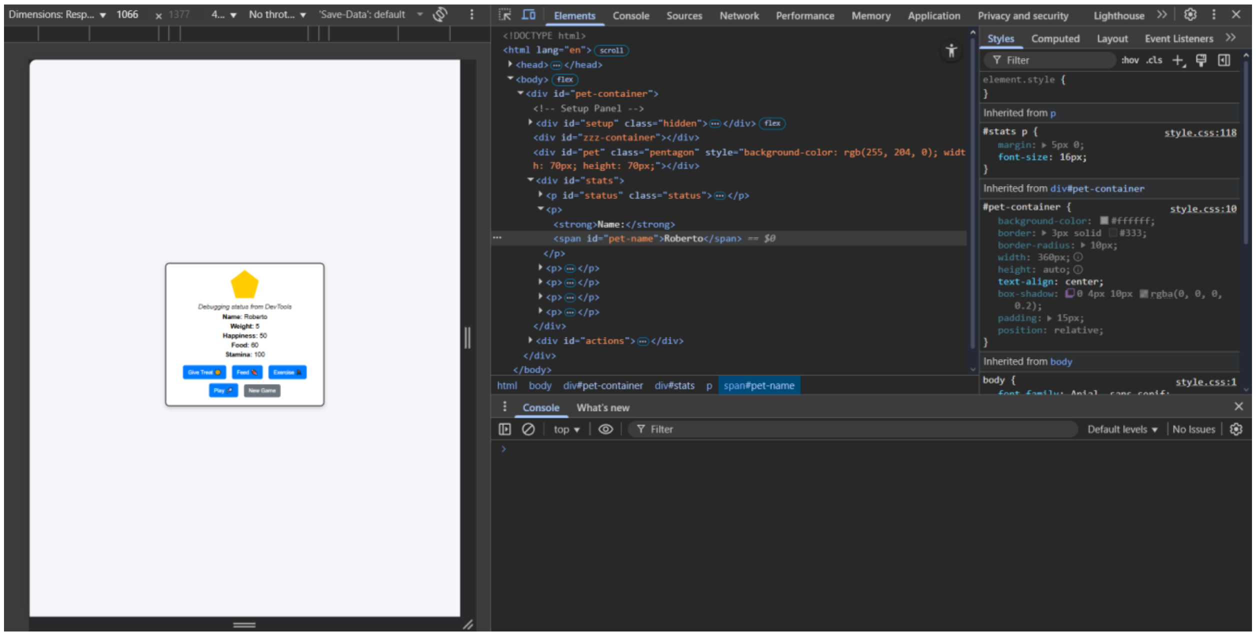


```
<!DOCTYPE html>
<html lang="en"> scroll
  ▸ <head> … </head>
  ▾ <body> flex
    ▾ <div id="pet-container">
        <!-- Setup Panel -->
      ▸ <div id="setup" class="hidden"> … </div>
        <div id="zzz-container"></div>
        <div id="pet" class="pentagon" style="background-color: rgb(255, 204, 0); width:
        70px; height: 70px;"></div> == $0
      ▸ <div id="stats"> … </div>
      ▸ <div id="actions"> … </div>
      </div>
    </body>
</html>
```

```
element.style {
    background-color: rgb(255, 204, 0);
    width: 70px;
    height: 70px;
}
#pet.pentagon {                                   *style.css:72
    clip-path: polygon(50% 0%, 100% 38%, 82%
        100%, 18% 100%, 0% 38%);
}
#pet {                                            *style.css:61
    ~~width: 100px;~~
    ~~height: 100px;~~
    ~~background-color: #ffcc00;~~
    border: ▸ 4px solid red;
    margin: ▸ 0 auto 12px;
    transition: ▸ transform 0.3s, background-
        color 0.3s, width 0.3s, height 0.3s,
        filter 0.3s, opacity 0.3s;
    position: relative;
    z-index: 1;
}
div {                             user agent stylesheet
    display: block;
    unicode-bidi: isolate;
```

```
html  body  div#pet-container  div#pet.pentagon
```

I directly edited the #status element using "Edit as HTML" in the Elements panel. The new text appeared immediately in the game UI, demonstrating how I can quickly test content and layout changes in the DOM.

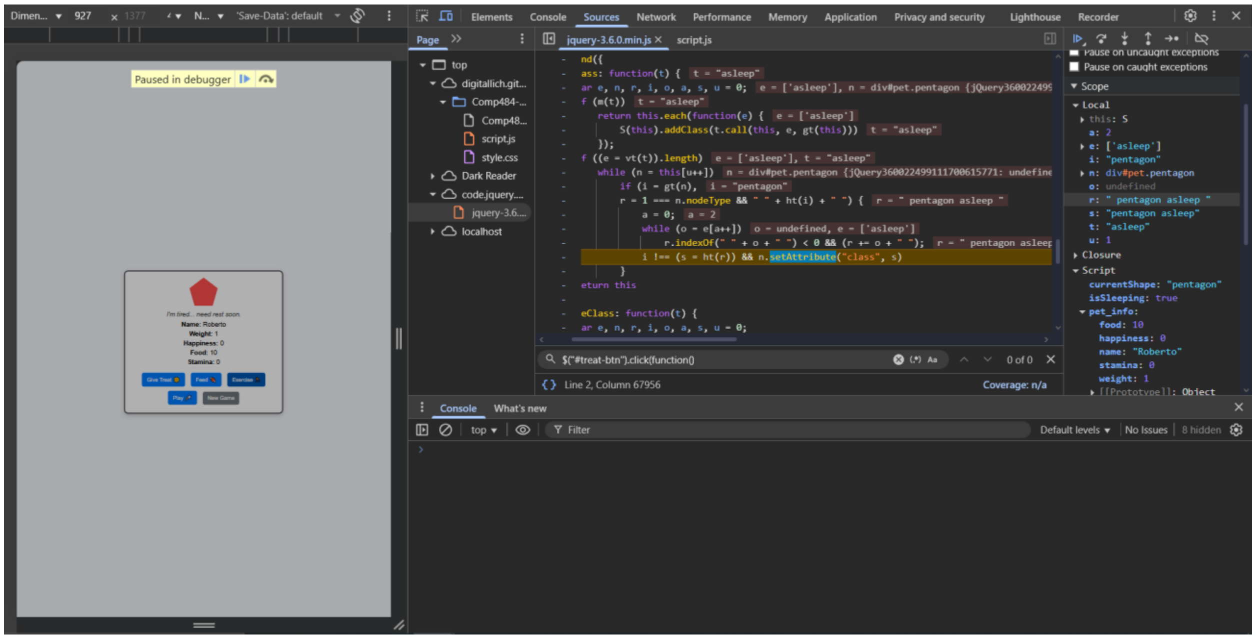


Debugging status from DevTools
Name: Roberto
Weight: 5
Happiness: 50
Food: 60
Stamina: 100

```
<!DOCTYPE html>
<html lang="en"> scroll
  <head> … </head>
  <body> flex
    <div id="pet-container">
      <!-- Setup Panel -->
      <div id="setup" class="hidden"> … </div> flex
      <div id="zzz-container"></div>
      <div id="pet" class="pentagon" style="background-color: rgb(255, 204, 0); widt
      h: 70px; height: 70px;"></div>
      <div id="stats">
        <p id="status" class="status"> … </p>
        <p>
          <strong>Name:</strong>
          <span id="pet-name">Roberto</span> == $0
        </p>
        <p> … </p>
        <p> … </p>
        <p> … </p>
        <p> … </p>
      </div>
      <div id="actions"> … </div>
    </div>
  </body>
```

```
element.style {
}
Inherited from p
#stats p {                                  style.css:118
  margin: ▸ 5px 0;
  font-size: 16px;
}
Inherited from div#pet-container
#pet-container {                            style.css:10
  background-color: #ffffff;
  border: ▸ 3px solid #333;
  border-radius: ▸ 10px;
  width: 360px;
  height: auto;
  text-align: center;
  box-shadow: 0 4px 10px rgba(0, 0, 0,
    0.2);
  padding: ▸ 15px;
  position: relative;
}
Inherited from body
body {                                      style.css:1
```

I added a DOM breakpoint on the #pet element to break when its attributes change. When stamina reached zero, DevTools paused as the asleep class was added. This helped me connect the visual state (greyed-out pet) with the underlying JavaScript (startSleep and wakeUp functions).



I'm tired... need rest soon.
Name: Roberto
Weight: 1
Happiness: 0
Food: 10
Stamina: 0

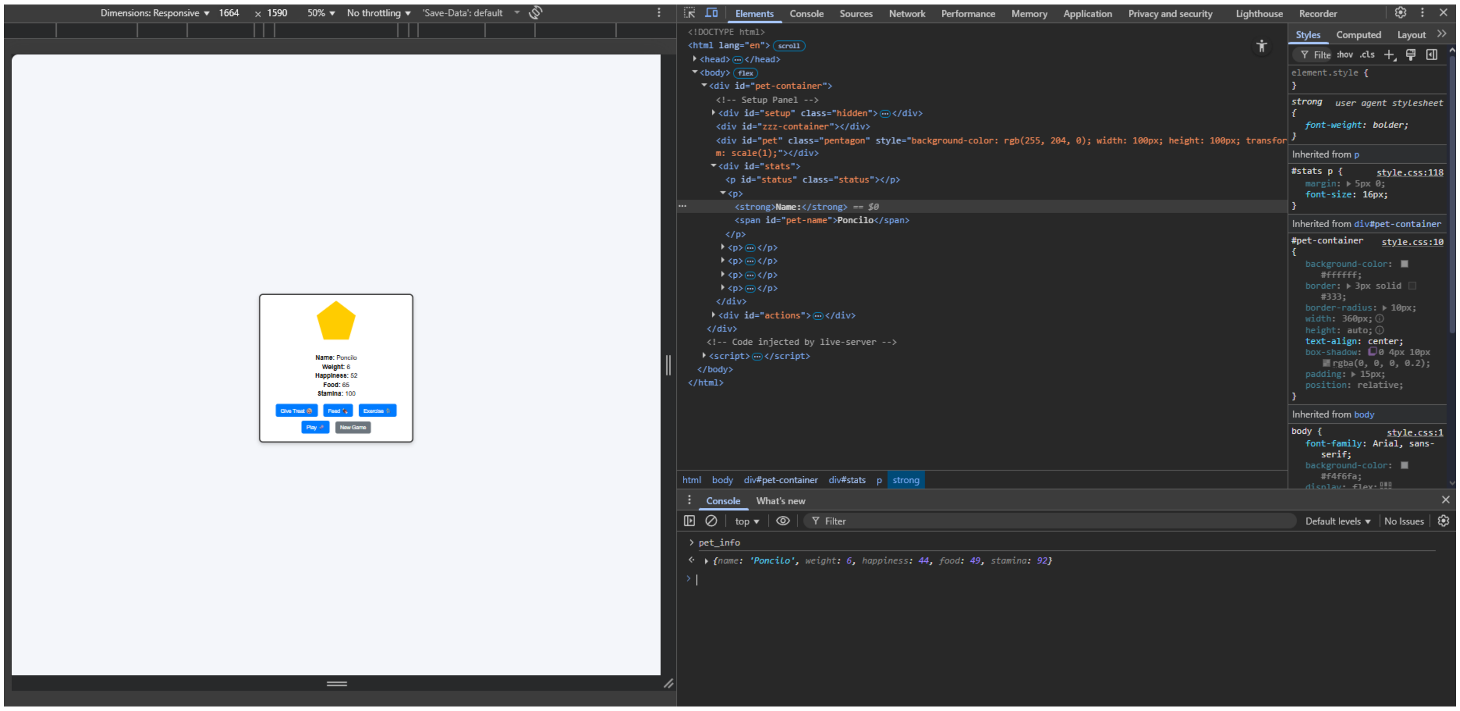
```
nd({
ass: function(t) {  t = "asleep"
ar e, n, r, i, o, a, s, u = 0;  e = ['asleep'], n = div#pet.pentagon {jQuery36002249
f (m(t))  t = "asleep"
   return this.each(function(e) {  e = ['asleep']
       S(this).addClass(t.call(this, e, gt(this)))  t = "asleep"
   });
f ((e = vt(t)).length)  e = ['asleep'], t = "asleep"
   while (n = this[u++])  n = div#pet.pentagon {jQuery3600224991117006615771: undefine
       if (i = gt(n),  i = "pentagon"
       r = 1 === n.nodeType && " " + ht(i) + " ") {  r = " pentagon asleep "
           a = 0;  a = 2
           while (o = e[a++])  o = undefined, e = ['asleep']
               r.indexOf(" " + o + " ") < 0 && (r += o + " ");  r = " pentagon asleep
           i !== (s = ht(r)) && n.setAttribute("class", s)
       }
eturn this

eClass: function(t) {
ar e, n, r, i, o, a, s, u = 0;
```

```
Scope
▾ Local
  ▸ this: S
    a: 2
  ▸ e: ['asleep']
    i: "pentagon"
  ▸ n: div#pet.pentagon
    o: undefined
    r: " pentagon asleep "
    s: "pentagon asleep"
    t: "asleep"
    u: 1
▸ Closure
▾ Script
    currentShape: "pentagon"
    isSleeping: true
  ▾ pet_info:
      food: 10
      happiness: 0
      name: "Roberto"
      stamina: 0
      weight: 1
    ▸ [[Prototype]]: Object
```

New Implementation: Stat Decay with time

After implementation, the pet’s on-screen stats remain unchanged even after several seconds, revealing that the time-based stat decay feature is not updating the UI as expected. DevTools Console showing that pet_info values decrease over time while the UI remains static, confirming that the internal game state is updating correctly but the DOM is not being refreshed.



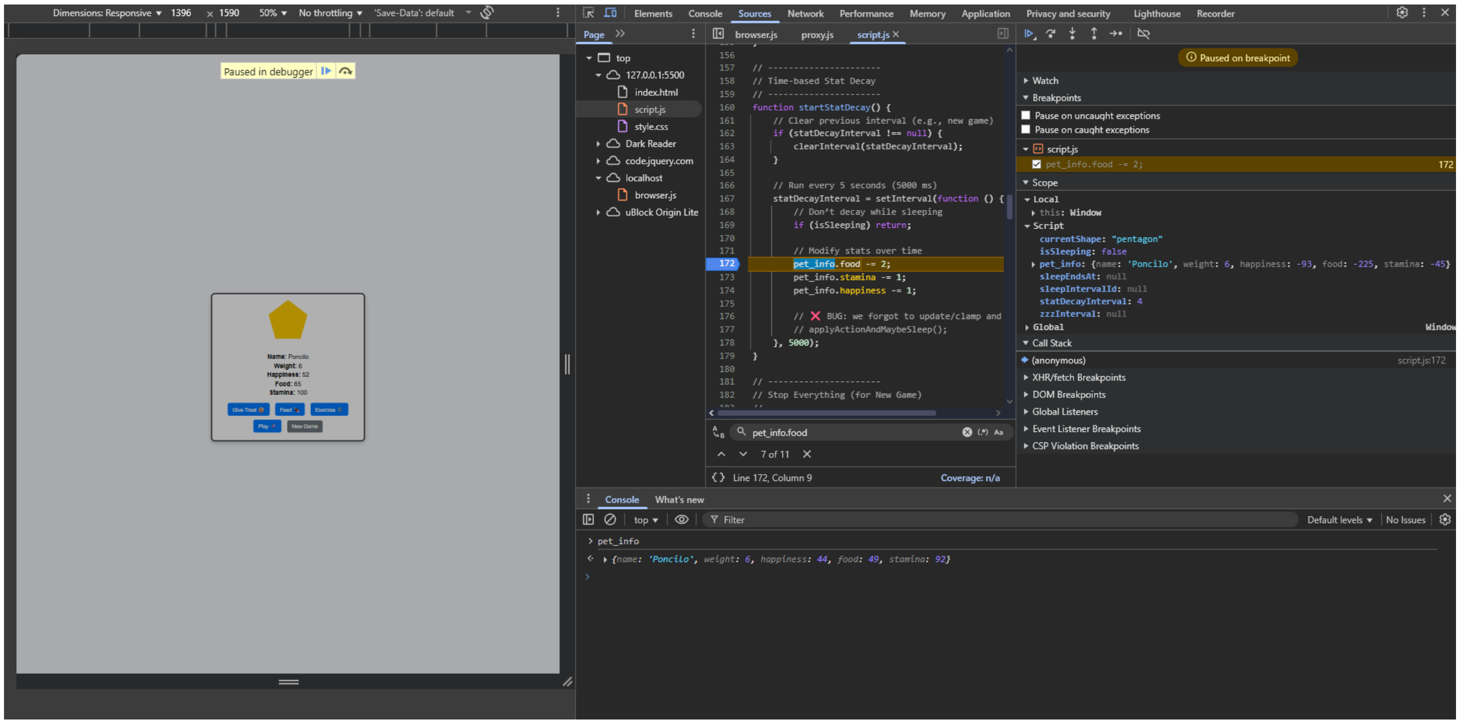
Name: Poncilo
Weight: 6
Happiness: 52
Food: 65
Stamina: 100

```
<!DOCTYPE html>
<html lang="en"> scroll
  <head> … </head>
  <body> flex
    <div id="pet-container">
      <!-- Setup Panel -->
      <div id="setup" class="hidden"> … </div>
      <div id="zzz-container"></div>
      <div id="pet" class="pentagon" style="background-color: rgb(255, 204, 0); width: 100px; height: 100px; transform: scale(1);"></div>
      <div id="stats">
        <p id="status" class="status"></p>
        <p>
          <strong>Name:</strong> == $0
          <span id="pet-name">Poncilo</span>
        </p>
        <p> … </p>
        <p> … </p>
        <p> … </p>
        <p> … </p>
      </div>
      <div id="actions"> … </div>
    </div>
    <!-- Code injected by live-server -->
    <script> … </script>
  </body>
</html>
```

```
element.style {
}
strong    user agent stylesheet
{
  font-weight: bolder;
}
Inherited from p
#stats p {          style.css:118
  margin: 5px 0;
  font-size: 16px;
}
Inherited from div#pet-container
#pet-container      style.css:10
{
  background-color: #ffffff;
  border: 3px solid #333;
  border-radius: 10px;
  width: 360px;
  height: auto;
  text-align: center;
  box-shadow: 0 4px 10px rgba(0, 0, 0, 0.2);
  padding: 15px;
  position: relative;
}
Inherited from body
body {              style.css:1
  font-family: Arial, sans-serif;
  background-color: #f4f6fa;
```

```
> pet_info
< ▸ {name: 'Poncilo', weight: 6, happiness: 44, food: 49, stamina: 92}
```

A breakpoint added inside the startStatDecay() interval function. This allows us to pause execution exactly when the decay logic runs and inspect variable changes. The Scope panel reveals that pet_info.food, pet_info.stamina, and pet_info.happiness have decayed values, confirming the interval logic is functioning even though the UI does not reflect these changes.



Paused in debugger

Name: Poncilo
Weight: 6
Happiness: 52
Food: 65
Stamina: 100

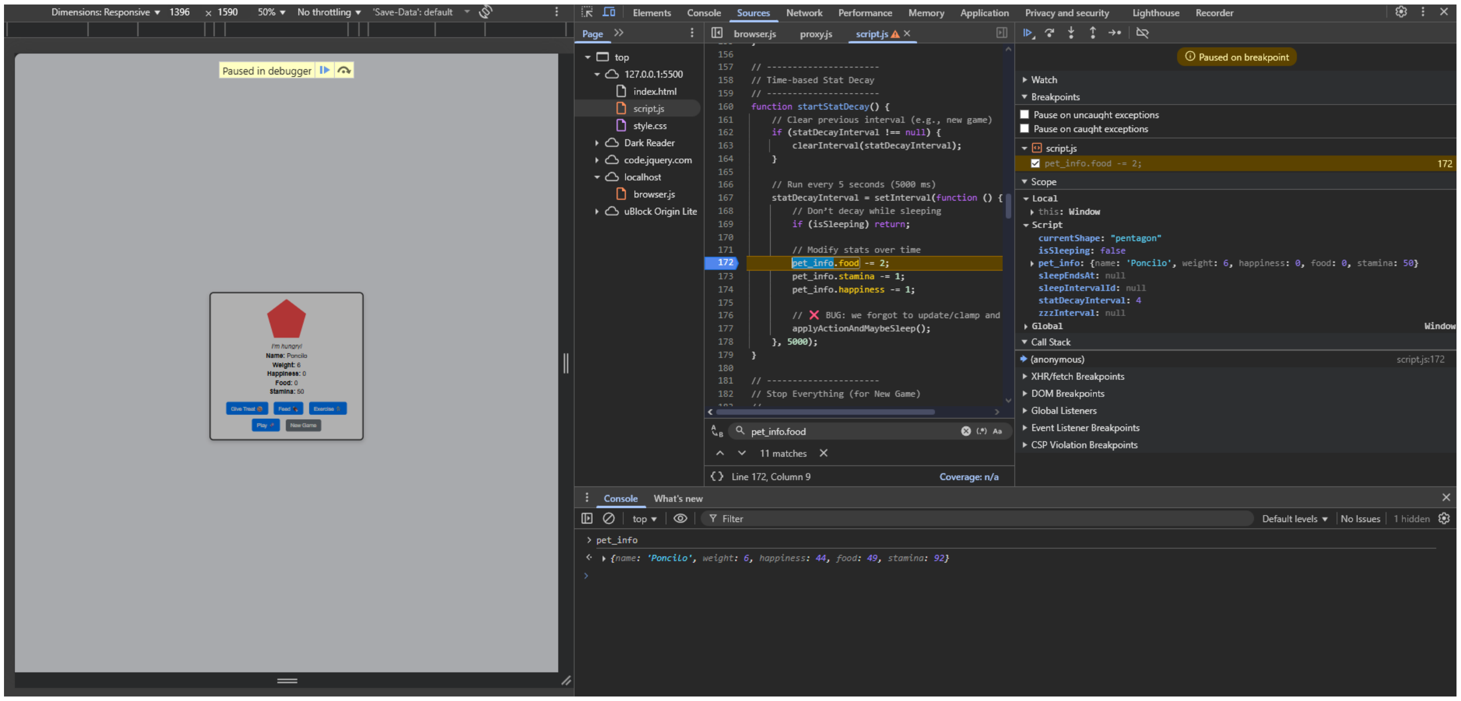
```
156
157 // ---------------------
158 // Time-based Stat Decay
159 // ---------------------
160 function startStatDecay() {
161     // Clear previous interval (e.g., new game)
162     if (statDecayInterval !== null) {
163         clearInterval(statDecayInterval);
164     }
165
166     // Run every 5 seconds (5000 ms)
167     statDecayInterval = setInterval(function () {
168         // Don’t decay while sleeping
169         if (isSleeping) return;
170
171         // Modify stats over time
172         pet_info.food -= 2;
173         pet_info.stamina -= 1;
174         pet_info.happiness -= 1;
175
176         // ❌ BUG: we forgot to update/clamp and
177         // applyActionAndMaybeSleep();
178     }, 5000);
179 }
180
181 // ---------------------
182 // Stop Everything (for New Game)
```

Line 172, Column 9

```
Paused on breakpoint
Breakpoints
  ☐ Pause on uncaught exceptions
  ☐ Pause on caught exceptions
  script.js
    ☑ pet_info.food -= 2;                     172
Scope
  Local
    ▸ this: Window
  Script
    currentShape: "pentagon"
    isSleeping: false
    ▸ pet_info: {name: 'Poncilo', weight: 6, happiness: -93, food: -225, stamina: -45}
    sleepEndsAt: null
    sleepIntervalId: null
    statDecayInterval: 4
    zzzInterval: null
  ▸ Global                                      Window
Call Stack
  (anonymous)                              script.js:172
```

```
> pet_info
< ▸ {name: 'Poncilo', weight: 6, happiness: 44, food: 49, stamina: 92}
```

The corrected version of startStatDecay(), where the applyActionAndMaybeSleep() function is restored, allowing stats to update visually and triggering sleep behavior when stamina becomes too low.



```
// ---------------------
// Time-based Stat Decay
// ---------------------
function startStatDecay() {
    // Clear previous interval (e.g., new game)
    if (statDecayInterval !== null) {
        clearInterval(statDecayInterval);
    }

    // Run every 5 seconds (5000 ms)
    statDecayInterval = setInterval(function () {
        // Don't decay while sleeping
        if (isSleeping) return;

        // Modify stats over time
        pet_info.food -= 2;
        pet_info.stamina -= 1;
        pet_info.happiness -= 1;

        // ❌ BUG: we forgot to update/clamp and
        applyActionAndMaybeSleep();
    }, 5000);
}

// ---------------------
// Stop Everything (for New Game)
```

```
Paused on breakpoint
Breakpoints
script.js
  pet_info.food -= 2;                                   172
Scope
Local
  this: Window
Script
  currentShape: "pentagon"
  isSleeping: false
  pet_info: {name: 'Poncilo', weight: 6, happiness: 0, food: 0, stamina: 50}
  sleepEndsAt: null
  sleepIntervalId: null
  statDecayInterval: 4
  zzzInterval: null
Global                                                  Window
Call Stack
(anonymous)                                             script.js:172
```

```
> pet_info
<  {name: 'Poncilo', weight: 6, happiness: 44, food: 49, stamina: 92}
```

After applying the fix, the pet's on-screen stats now decrease every few seconds, demonstrating that the decay logic and UI updates are properly synchronized.