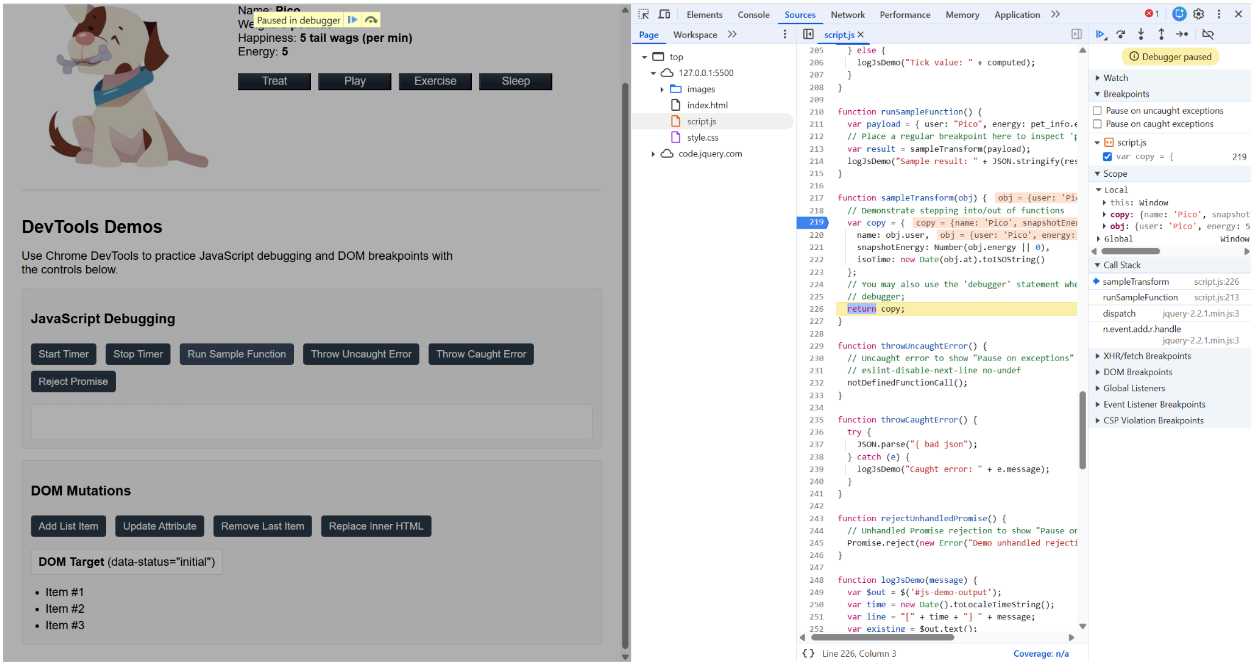
JavaScript Debugging Demos

1) Breakpoints, Step Over / Into / Out

What to do:

1. In DevTools, open the **Sources** panel and locate `script.js`.
2. Find the function `sampleTransform` and **set a breakpoint** inside the function body.
3. In the page UI, click **“Run Sample Function”**.
4. When execution pauses:
   - Use **Step Into** to follow function calls inside `sampleTransform`.
   - Use **Step Over** to run individual lines without going into inner calls.
   - Use **Step Out** to finish the current function and return to its caller.
5. Inspect:
   - The values of `payload` and `copy`.
   - Local and closure variables in the **Scope** panel.
   - The **Call Stack** to see how `sampleTransform` was reached.



Name: **Pico**
Happiness: **5 tail wags (per min)**
Energy: **5**

DevTools Demos

Use Chrome DevTools to practice JavaScript debugging and DOM breakpoints with the controls below.

JavaScript Debugging

DOM Mutations

DOM Target (data-status="initial")

- Item #1
- Item #2
- Item #3

```
  } else {
    logJsDemo("Tick value: " + computed);
  }
}

function runSampleFunction() {
  var payload = { user: "Pico", energy: pet_info.e
  // Place a regular breakpoint here to inspect 'p
  var result = sampleTransform(payload);
  logJsDemo("Sample result: " + JSON.stringify(res
}

function sampleTransform(obj) {  obj = {user: 'Pi
  // Demonstrate stepping into/out of functions
  var copy = {  copy = {name: 'Pico', snapshotEner
    name: obj.user,  obj = {user: 'Pico', energy:
    snapshotEnergy: Number(obj.energy || 0),
    isoTime: new Date(obj.at).toISOString()
  };
  // You may also use the 'debugger' statement whe
  // debugger;
  return copy;
}

function throwUncaughtError() {
  // Uncaught error to show "Pause on exceptions"
  // eslint-disable-next-line no-undef
  notDefinedFunctionCall();
}

function throwCaughtError() {
  try {
    JSON.parse("{ bad json");
  } catch (e) {
    logJsDemo("Caught error: " + e.message);
  }
}

function rejectUnhandledPromise() {
  // Unhandled Promise rejection to show "Pause or
  Promise.reject(new Error("Demo unhandled rejecti
}

function logJsDemo(message) {
  var $out = $('#js-demo-output');
  var time = new Date().toLocaleTimeString();
  var line = "[" + time + "] " + message;
  var existing = $out.text();
```

Line 226, Column 3

□ Pause on uncaught exceptions
□ Pause on caught exceptions

script.js
☑ var copy = {   219

Scope
Local
this: Window
copy: {name: 'Pico', snapshot
obj: {user: 'Pico', energy: 5
Global   Window

Call Stack
sampleTransform   script.js:226
runSampleFunction   script.js:213
dispatch   jquery-2.2.1.min.js:3
n.event.add.r.handle   jquery-2.2.1.min.js:3

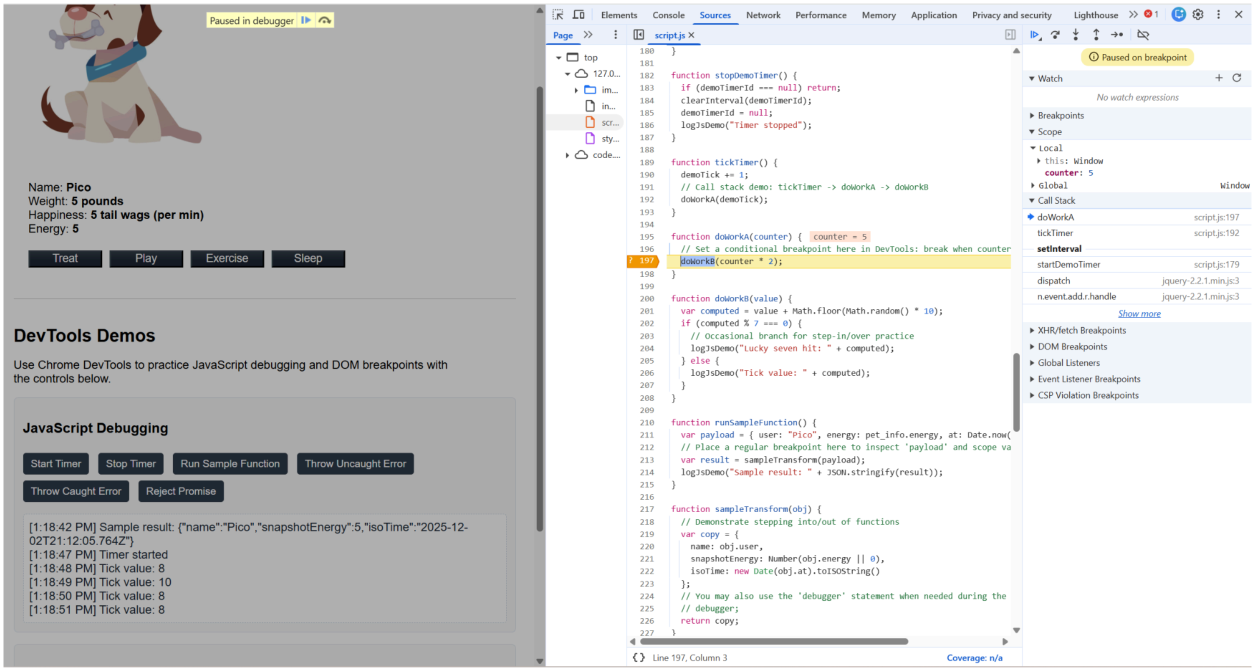
2) Conditional Breakpoint / Logpoint

What to do:

1. In `script.js`, find the function `doWorkA(counter)`.
2. In the **Sources** panel, set a breakpoint on the line inside `doWorkA`.
3. Right-click the breakpoint and choose **“Edit breakpoint…”**, then enter the condition:

   ```
   counter % 5 === 0
   ```

4. In the page, click **“Start Timer”**.
5. Observe how DevTools **only pauses** when `counter` is a multiple of 5.
6. Alternatively, convert the breakpoint to a **logpoint**:
   - Right-click the same line and choose **“Edit logpoint…”** (or similar option, depending on Chrome version).
   - Log something like: `"doWorkA counter:", counter`
7. Run the timer again and watch the messages in the **Console** without pausing execution.



Name: **Pico**
Weight: **5 pounds**
Happiness: **5 tail wags (per min)**
Energy: **5**

DevTools Demos

Use Chrome DevTools to practice JavaScript debugging and DOM breakpoints with the controls below.

JavaScript Debugging

```
[1:18:42 PM] Sample result: {"name":"Pico","snapshotEnergy":5,"isoTime":"2025-12-
02T21:12:05.764Z"}
[1:18:47 PM] Timer started
[1:18:48 PM] Tick value: 8
[1:18:49 PM] Tick value: 10
[1:18:50 PM] Tick value: 8
[1:18:51 PM] Tick value: 8
```

```
}

function stopDemoTimer() {
  if (demoTimerId === null) return;
  clearInterval(demoTimerId);
  demoTimerId = null;
  logJsDemo("Timer stopped");
}

function tickTimer() {
  demoTick += 1;
  // Call stack demo: tickTimer -> doWorkA -> doWorkB
  doWorkA(demoTick);
}

function doWorkA(counter) {  counter = 5
  // Set a conditional breakpoint here in DevTools: break when counter
  doWorkB(counter * 2);
}

function doWorkB(value) {
  var computed = value + Math.floor(Math.random() * 10);
  if (computed % 7 === 0) {
    // Occasional branch for step-in/over practice
    logJsDemo("Lucky seven hit: " + computed);
  } else {
    logJsDemo("Tick value: " + computed);
  }
}

function runSampleFunction() {
  var payload = { user: "Pico", energy: pet_info.energy, at: Date.now(
  // Place a regular breakpoint here to inspect 'payload' and scope va
  var result = sampleTransform(payload);
  logJsDemo("Sample result: " + JSON.stringify(result));
}

function sampleTransform(obj) {
  // Demonstrate stepping into/out of functions
  var copy = {
    name: obj.user,
    snapshotEnergy: Number(obj.energy || 0),
    isoTime: new Date(obj.at).toISOString()
  };
  // You may also use the 'debugger' statement when needed during the
  // debugger;
  return copy;
}
```

```
Local
  this: Window
  counter: 5
Global                                Window
```

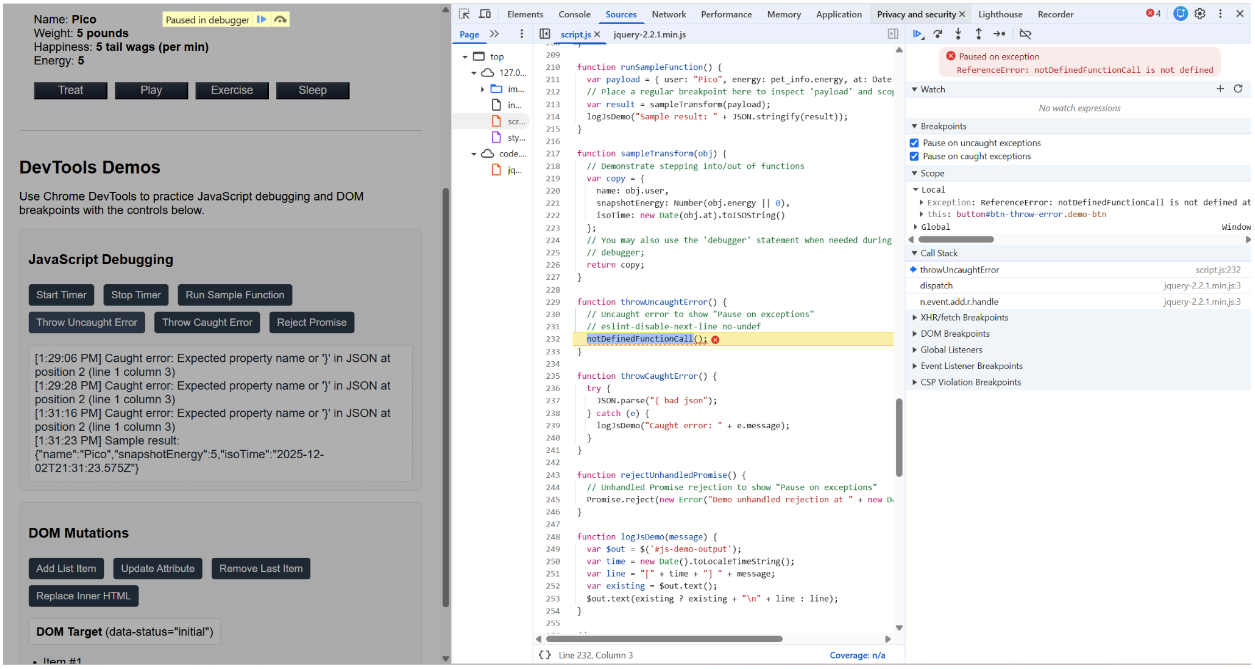
Call Stack

```
doWorkA                    script.js:197
tickTimer                  script.js:192
setInterval
startDemoTimer             script.js:179
dispatch                   jquery-2.2.1.min.js:3
n.event.add.r.handle       jquery-2.2.1.min.js:3
```

3) Pause on Exceptions (Uncaught and Caught)

What to do:

1. In DevTools, go to the **Sources** panel and enable **“Pause on exceptions”** using the pause icon with a stop sign.
2. Optionally also enable **“Pause on caught exceptions”**.
3. Back in the page, click **“Throw Uncaught Error”**:
    - DevTools should pause at the line where the uncaught error is thrown.
4. Then click **“Throw Caught Error”**:
    - With only “Pause on exceptions” enabled, DevTools may not pause because the error is handled in a `try/catch`.
    - With **“Pause on caught exceptions”** also enabled, DevTools will pause even for the handled error, allowing you to inspect the error object and stack.



Name: **Pico**
Weight: **5 pounds**
Happiness: **5 tail wags (per min)**
Energy: **5**

DevTools Demos

Use Chrome DevTools to practice JavaScript debugging and DOM breakpoints with the controls below.

JavaScript Debugging

```
[1:29:06 PM] Caught error: Expected property name or '}' in JSON at
position 2 (line 1 column 3)
[1:29:28 PM] Caught error: Expected property name or '}' in JSON at
position 2 (line 1 column 3)
[1:31:16 PM] Caught error: Expected property name or '}' in JSON at
position 2 (line 1 column 3)
[1:31:23 PM] Sample result:
{"name":"Pico","snapshotEnergy":5,"isoTime":"2025-12-
02T21:31:23.575Z"}
```

DOM Mutations

DOM Target (data-status="initial")

- Item #1

```
function runSampleFunction() {
  var payload = { user: "Pico", energy: pet_info.energy, at: Date
  // Place a regular breakpoint here to inspect 'payload' and scop
  var result = sampleTransform(payload);
  logJsDemo("Sample result: " + JSON.stringify(result));
}

function sampleTransform(obj) {
  // Demonstrate stepping into/out of functions
  var copy = {
    name: obj.user,
    snapshotEnergy: Number(obj.energy || 0),
    isoTime: new Date(obj.at).toISOString()
  };
  // You may also use the 'debugger' statement when needed during
  // debugger;
  return copy;
}

function throwUncaughtError() {
  // Uncaught error to show "Pause on exceptions"
  // eslint-disable-next-line no-undef
  notDefinedFunctionCall();
}

function throwCaughtError() {
  try {
    JSON.parse("{ bad json");
  } catch (e) {
    logJsDemo("Caught error: " + e.message);
  }
}

function rejectUnhandledPromise() {
  // Unhandled Promise rejection to show "Pause on exceptions"
  Promise.reject(new Error("Demo unhandled rejection at " + new Da
}

function logJsDemo(message) {
  var $out = $('#js-demo-output');
  var time = new Date().toLocaleTimeString();
  var line = "[" + time + "] " + message;
  var existing = $out.text();
  $out.text(existing ? existing + "\n" + line : line);
}
```

Line 232, Column 3   Coverage: n/a

Paused on exception
`ReferenceError: notDefinedFunctionCall is not defined`

Watch
No watch expressions

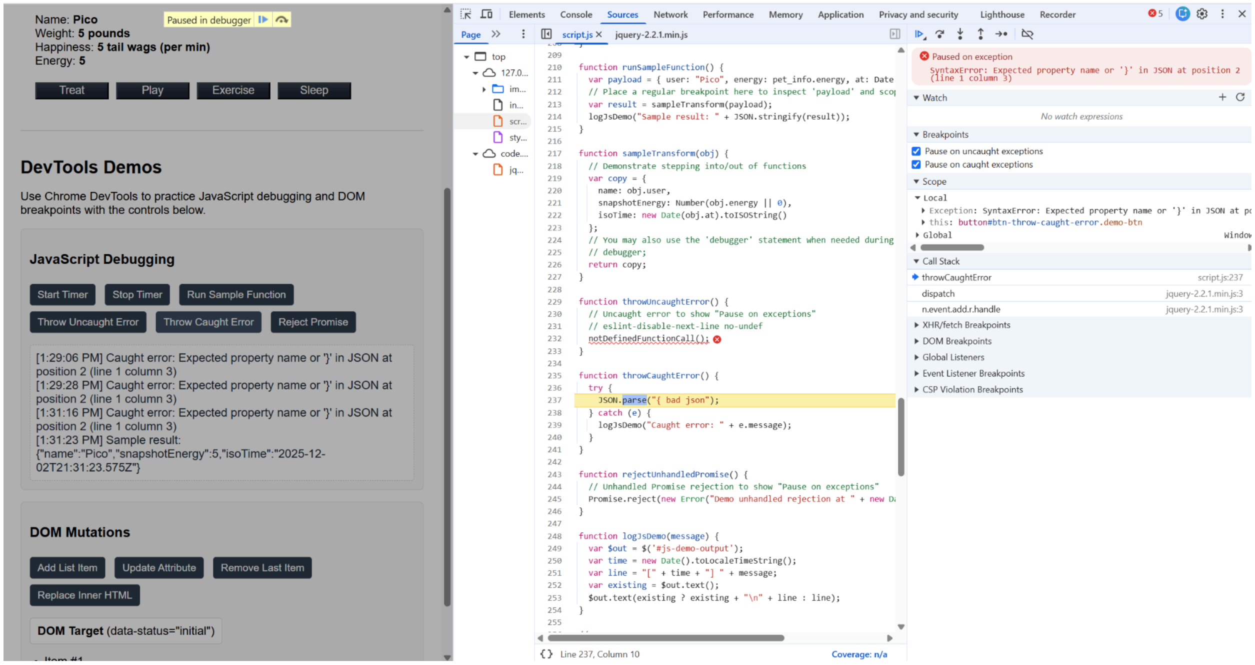
Breakpoints
- ☑ Pause on uncaught exceptions
- ☑ Pause on caught exceptions

Scope
```
Local
  Exception: ReferenceError: notDefinedFunctionCall is not defined at
  this: button#btn-throw-error.demo-btn
Global                                                        Window
```

Call Stack

throwUncaughtError	script.js:232
dispatch	jquery-2.2.1.min.js:3
n.event.add.r.handle	jquery-2.2.1.min.js:3

XHR/fetch Breakpoints
DOM Breakpoints
Global Listeners
Event Listener Breakpoints
CSP Violation Breakpoints



Name: **Pico**
Weight: **5 pounds**
Happiness: **5 tail wags (per min)**
Energy: **5**

DevTools Demos

Use Chrome DevTools to practice JavaScript debugging and DOM breakpoints with the controls below.

JavaScript Debugging

```
[1:29:06 PM] Caught error: Expected property name or '}' in JSON at
position 2 (line 1 column 3)
[1:29:28 PM] Caught error: Expected property name or '}' in JSON at
position 2 (line 1 column 3)
[1:31:16 PM] Caught error: Expected property name or '}' in JSON at
position 2 (line 1 column 3)
[1:31:23 PM] Sample result:
{"name":"Pico","snapshotEnergy":5,"isoTime":"2025-12-
02T21:31:23.575Z"}
```

DOM Mutations

DOM Target (data-status="initial")

- Item #1

```
function runSampleFunction() {
  var payload = { user: "Pico", energy: pet_info.energy, at: Date
  // Place a regular breakpoint here to inspect 'payload' and scop
  var result = sampleTransform(payload);
  logJsDemo("Sample result: " + JSON.stringify(result));
}

function sampleTransform(obj) {
  // Demonstrate stepping into/out of functions
  var copy = {
    name: obj.user,
    snapshotEnergy: Number(obj.energy || 0),
    isoTime: new Date(obj.at).toISOString()
  };
  // You may also use the 'debugger' statement when needed during
  // debugger;
  return copy;
}

function throwUncaughtError() {
  // Uncaught error to show "Pause on exceptions"
  // eslint-disable-next-line no-undef
  notDefinedFunctionCall();
}

function throwCaughtError() {
  try {
    JSON.parse("{ bad json");
  } catch (e) {
    logJsDemo("Caught error: " + e.message);
  }
}

function rejectUnhandledPromise() {
  // Unhandled Promise rejection to show "Pause on exceptions"
  Promise.reject(new Error("Demo unhandled rejection at " + new Da
}

function logJsDemo(message) {
  var $out = $('#js-demo-output');
  var time = new Date().toLocaleTimeString();
  var line = "[" + time + "] " + message;
  var existing = $out.text();
  $out.text(existing ? existing + "\n" + line : line);
}
```

Line 237, Column 10   Coverage: n/a

Paused on exception
```
SyntaxError: Expected property name or '}' in JSON at position 2
(line 1 column 3)
```

Watch
No watch expressions

Breakpoints
- ☑ Pause on uncaught exceptions
- ☑ Pause on caught exceptions

Scope
```
Local
  Exception: SyntaxError: Expected property name or '}' in JSON at po
  this: button#btn-throw-caught-error.demo-btn
Global                                                        Window
```

Call Stack

throwCaughtError	script.js:237
dispatch	jquery-2.2.1.min.js:3
n.event.add.r.handle	jquery-2.2.1.min.js:3

XHR/fetch Breakpoints
DOM Breakpoints
Global Listeners
Event Listener Breakpoints
CSP Violation Breakpoints

4) Unhandled Promise Rejection

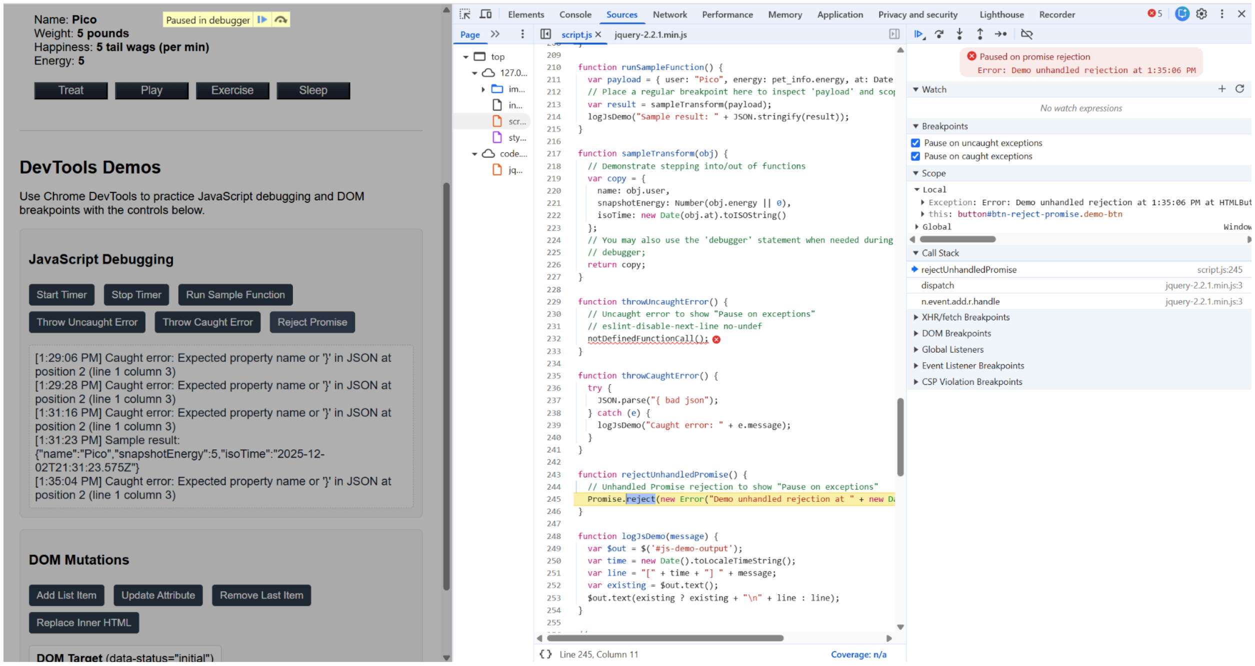
What to do:

1. In the page, click **“Reject Promise”**.

2. This triggers an **unhandled promise rejection** in `script.js`.

3. With **Pause on exceptions** enabled, DevTools should pause where the rejection occurs.

4. Open the **Console** to see:
    - The unhandled rejection message.
    - Any associated stack trace or context.



Name: **Pico**
Weight: **5 pounds**
Happiness: **5 tail wags (per min)**
Energy: **5**

DevTools Demos

Use Chrome DevTools to practice JavaScript debugging and DOM breakpoints with the controls below.

JavaScript Debugging

```
[1:29:06 PM] Caught error: Expected property name or '}' in JSON at position 2 (line 1 column 3)
[1:29:28 PM] Caught error: Expected property name or '}' in JSON at position 2 (line 1 column 3)
[1:31:16 PM] Caught error: Expected property name or '}' in JSON at position 2 (line 1 column 3)
[1:31:23 PM] Sample result: {"name":"Pico","snapshotEnergy":5,"isoTime":"2025-12-02T21:31:23.575Z"}
[1:35:04 PM] Caught error: Expected property name or '}' in JSON at position 2 (line 1 column 3)
```

DOM Mutations

DOM Target (data-status="initial")

```

function runSampleFunction() {
  var payload = { user: "Pico", energy: pet_info.energy, at: Date
  // Place a regular breakpoint here to inspect 'payload' and scop
  var result = sampleTransform(payload);
  logJsDemo("Sample result: " + JSON.stringify(result));
}

function sampleTransform(obj) {
  // Demonstrate stepping into/out of functions
  var copy = {
    name: obj.user,
    snapshotEnergy: Number(obj.energy || 0),
    isoTime: new Date(obj.at).toISOString()
  };
  // You may also use the 'debugger' statement when needed during
  // debugger;
  return copy;
}

function throwUncaughtError() {
  // Uncaught error to show "Pause on exceptions"
  // eslint-disable-next-line no-undef
  notDefinedFunctionCall();
}

function throwCaughtError() {
  try {
    JSON.parse("{ bad json");
  } catch (e) {
    logJsDemo("Caught error: " + e.message);
  }
}

function rejectUnhandledPromise() {
  // Unhandled Promise rejection to show "Pause on exceptions"
  Promise.reject(new Error("Demo unhandled rejection at " + new Da
}

function logJsDemo(message) {
  var $out = $('#js-demo-output');
  var time = new Date().toLocaleTimeString();
  var line = "[" + time + "] " + message;
  var existing = $out.text();
  $out.text(existing ? existing + "\n" + line : line);
}

```

Paused on promise rejection
`Error: Demo unhandled rejection at 1:35:06 PM`

No watch expressions

☑ Pause on uncaught exceptions
☑ Pause on caught exceptions

```
▾ Local
  ▸ Exception: Error: Demo unhandled rejection at 1:35:06 PM at HTMLBut
  ▸ this: button#btn-reject-promise.demo-btn
▸ Global                                                        Window
```

rejectUnhandledPromise	script.js:245
dispatch	jquery-2.2.1.min.js:3
n.event.add.r.handle	jquery-2.2.1.min.js:3

DOM Demos

5) DOM Subtree Modification

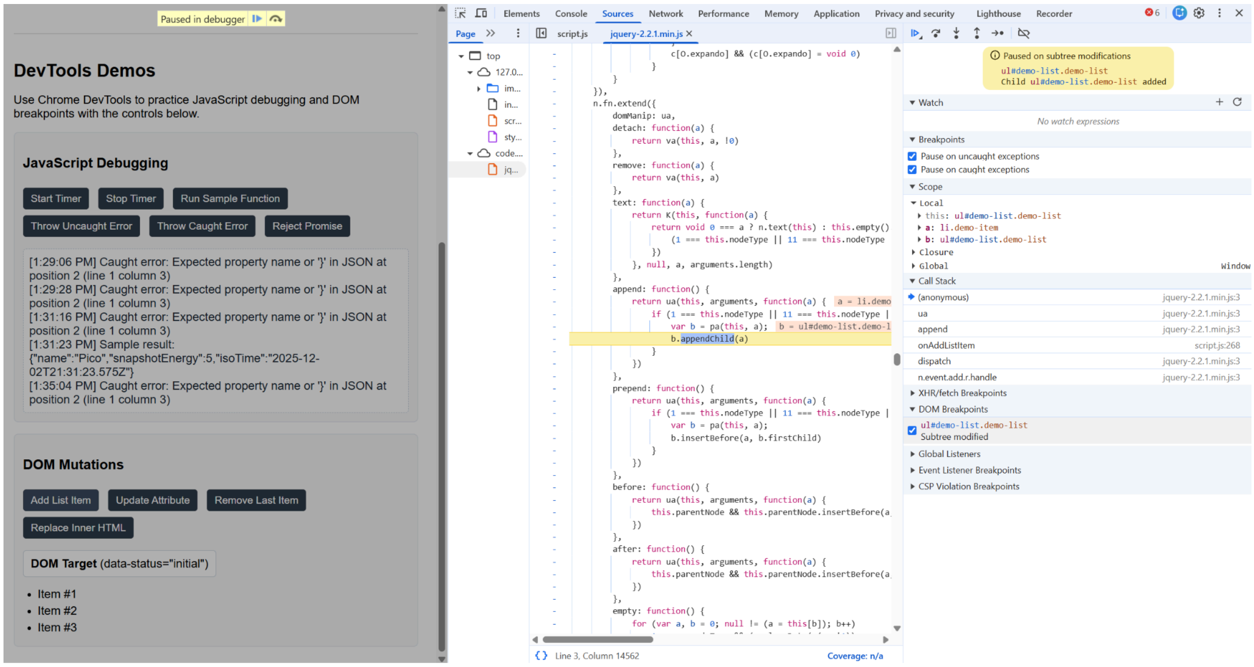
What to do:

1. In DevTools, open the **Elements** panel.

2. Locate the element with id `demo-list` (e.g., `<ul id="demo-list">`).

3. Right-click `#demo-list` and choose:
   Break on → Subtree modifications.

4. Back in the page, click **“Add List Item”**:
    - DevTools should pause at the JavaScript line that appends a new `<li>` to `#demo-list`.
5. Click **“Replace Inner HTML”**:
    - DevTools should pause again when the inner HTML of `#demo-list` is replaced.



DevTools Demos

Use Chrome DevTools to practice JavaScript debugging and DOM breakpoints with the controls below.

JavaScript Debugging

```
[1:29:06 PM] Caught error: Expected property name or '}' in JSON at position 2 (line 1 column 3)
[1:29:28 PM] Caught error: Expected property name or '}' in JSON at position 2 (line 1 column 3)
[1:31:16 PM] Caught error: Expected property name or '}' in JSON at position 2 (line 1 column 3)
[1:31:23 PM] Sample result: {"name":"Pico","snapshotEnergy":5,"isoTime":"2025-12-02T21:31:23.575Z"}
[1:35:04 PM] Caught error: Expected property name or '}' in JSON at position 2 (line 1 column 3)
```

DOM Mutations

DOM Target (data-status="initial")

- Item #1
- Item #2
- Item #3

```
                c[O.expando] && (c[O.expando] = void 0)
            }
        }
    }),
    n.fn.extend({
        domManip: ua,
        detach: function(a) {
            return va(this, a, !0)
        },
        remove: function(a) {
            return va(this, a)
        },
        text: function(a) {
            return K(this, function(a) {
                return void 0 === a ? n.text(this) : this.empty()
                    (1 === this.nodeType || 11 === this.nodeType
                })
            }, null, a, arguments.length)
        },
        append: function() {
            return ua(this, arguments, function(a) {  a = li.demo
                if (1 === this.nodeType || 11 === this.nodeType |
                    var b = pa(this, a);  b = ul#demo-list.demo-l
                    b.appendChild(a)
                }
            })
        },
        prepend: function() {
            return ua(this, arguments, function(a) {
                if (1 === this.nodeType || 11 === this.nodeType |
                    var b = pa(this, a);
                    b.insertBefore(a, b.firstChild)
                }
            })
        },
        before: function() {
            return ua(this, arguments, function(a) {
                this.parentNode && this.parentNode.insertBefore(a
            })
        },
        after: function() {
            return ua(this, arguments, function(a) {
                this.parentNode && this.parentNode.insertBefore(a
            })
        },
        empty: function() {
            for (var a, b = 0; null != (a = this[b]); b++)
```

Line 3, Column 14562

Paused on subtree modifications
ul#demo-list.demo-list
Child ul#demo-list.demo-list added

☑ Pause on uncaught exceptions
☑ Pause on caught exceptions

```
Local
  this: ul#demo-list.demo-list
  a: li.demo-item
  b: ul#demo-list.demo-list
Closure
Global                                   Window
```

(anonymous)	jquery-2.2.1.min.js:3
ua	jquery-2.2.1.min.js:3
append	jquery-2.2.1.min.js:3
onAddListItem	script.js:268
dispatch	jquery-2.2.1.min.js:3
n.event.add.r.handle	jquery-2.2.1.min.js:3

☑ ul#demo-list.demo-list
Subtree modified

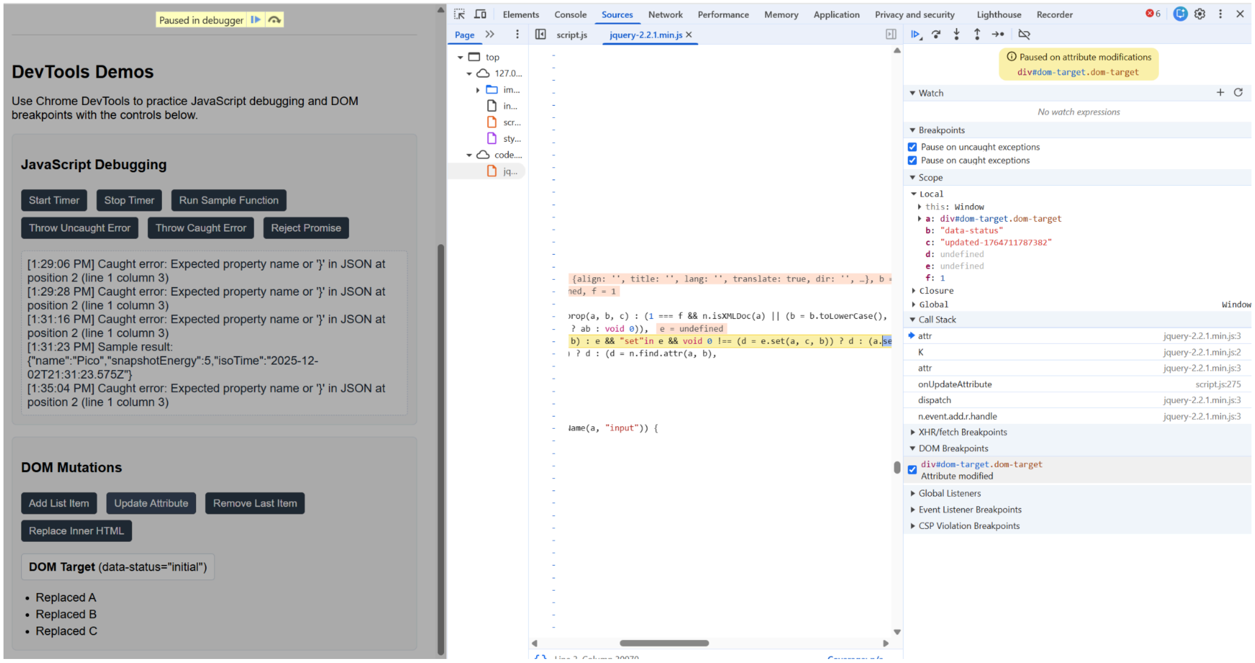
6) Attribute Modification

What to do:

1. In the **Elements** panel, find the element with id `dom-target`.
2. Right-click `#dom-target` and choose:
   Break on → Attribute modifications.
3. In the page, click **“Update Attribute”**:

- This should change the `data-status` attribute (for example, from `"idle"` to `"active"`).

4. DevTools will pause at the line in `script.js` that updates the attribute.
5. Inspect:
    - The **Call Stack** to see how this code was reached.
    - The **Elements** panel to confirm the updated attribute.



DevTools Demos

Use Chrome DevTools to practice JavaScript debugging and DOM breakpoints with the controls below.

JavaScript Debugging

Start Timer | Stop Timer | Run Sample Function | Throw Uncaught Error | Throw Caught Error | Reject Promise

```
[1:29:06 PM] Caught error: Expected property name or '}' in JSON at position 2 (line 1 column 3)
[1:29:28 PM] Caught error: Expected property name or '}' in JSON at position 2 (line 1 column 3)
[1:31:16 PM] Caught error: Expected property name or '}' in JSON at position 2 (line 1 column 3)
[1:31:23 PM] Sample result: {"name":"Pico","snapshotEnergy":5,"isoTime":"2025-12-02T21:31:23.575Z"}
[1:35:04 PM] Caught error: Expected property name or '}' in JSON at position 2 (line 1 column 3)
```

DOM Mutations

Add List Item | Update Attribute | Remove Last Item | Replace Inner HTML

DOM Target (data-status="initial")

- Replaced A
- Replaced B
- Replaced C

```
 {align: '', title: '', lang: '', translate: true, dir: '', …}, b =
ned, f = 1

rop(a, b, c) : (1 === f && n.isXMLDoc(a) || (b = b.toLowerCase(),
 ? ab : void 0)),  e = undefined
b) : e && "set"in e && void 0 !== (d = e.set(a, c, b)) ? d : (a.se
 ? d : (d = n.find.attr(a, b),

Jame(a, "input")) {
```

Paused on attribute modifications
div#dom-target.dom-target

Watch

No watch expressions

Breakpoints

☑ Pause on uncaught exceptions
☑ Pause on caught exceptions

Scope

```
Local
  this: Window
  a: div#dom-target.dom-target
  b: "data-status"
  c: "updated-1764711787382"
  d: undefined
  e: undefined
  f: 1
Closure
Global                                Window
```

Call Stack

```
attr                     jquery-2.2.1.min.js:3
K                        jquery-2.2.1.min.js:2
attr                     jquery-2.2.1.min.js:3
onUpdateAttribute        script.js:275
dispatch                 jquery-2.2.1.min.js:3
n.event.add.r.handle     jquery-2.2.1.min.js:3
```

XHR/fetch Breakpoints

DOM Breakpoints

☑ div#dom-target.dom-target
Attribute modified

Global Listeners

Event Listener Breakpoints

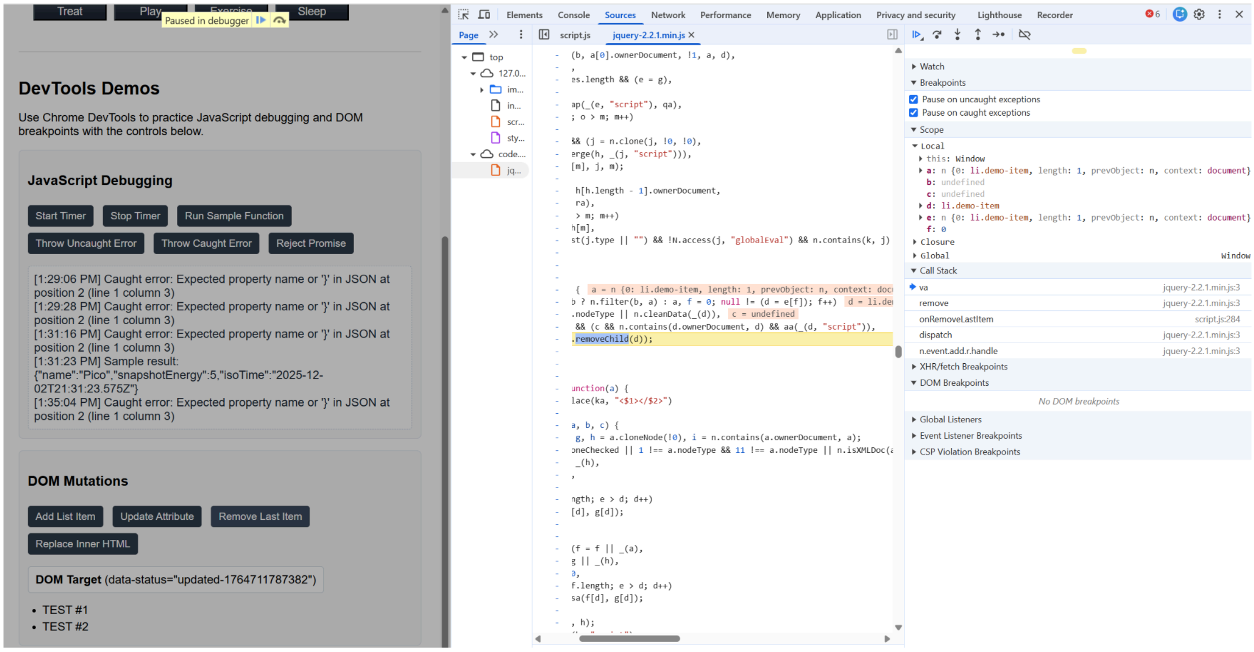
CSP Violation Breakpoints

7) Node Removal

What to do:

1. In the **Elements** panel, again select `#demo-list`.
2. Right-click and choose:
   Break on → Node removal.
3. In the page, click **"Remove Last Item"**:
    - The script should remove the last `<li>` from `#demo-list`.

4. DevTools will pause at the line where the node is removed.



DevTools Demos

Use Chrome DevTools to practice JavaScript debugging and DOM breakpoints with the controls below.

JavaScript Debugging

```
[1:29:06 PM] Caught error: Expected property name or '}' in JSON at position 2 (line 1 column 3)
[1:29:28 PM] Caught error: Expected property name or '}' in JSON at position 2 (line 1 column 3)
[1:31:16 PM] Caught error: Expected property name or '}' in JSON at position 2 (line 1 column 3)
[1:31:23 PM] Sample result: {"name":"Pico","snapshotEnergy":5,"isoTime":"2025-12-02T21:31:23.575Z"}
[1:35:04 PM] Caught error: Expected property name or '}' in JSON at position 2 (line 1 column 3)
```

DOM Mutations

DOM Target (data-status="updated-1764711787382")

- TEST #1
- TEST #2

```
(b, a[0].ownerDocument, !1, a, d),
,
es.length && (e = g),

ap(_(e, "script"), qa),
; o > m; m++)

&& (j = n.clone(j, !0, !0),
erge(h, _(j, "script"))),
[m], j, m);

 h[h.length - 1].ownerDocument,
 ra),
 > m; m++)
h[m],
st(j.type || "") && !N.access(j, "globalEval") && n.contains(k, j)



 {  a = n {0: li.demo-item, length: 1, prevObject: n, context: doc
b ? n.filter(b, a) : a, f = 0; null != (d = e[f]); f++)  d = li.de
.nodeType || n.cleanData(_(d)),  c = undefined
 && (c && n.contains(d.ownerDocument, d) && aa(_(d, "script")),
.removeChild(d));



unction(a) {
lace(ka, "<$1></$2>")

a, b, c) {
 g, h = a.cloneNode(!0), i = n.contains(a.ownerDocument, a);
oneChecked || 1 !== a.nodeType && 11 !== a.nodeType || n.isXMLDoc(a
 _(h),
,

ngth; e > d; d++)
[d], g[d]);


(f = f || _(a),
g || _(h),
0,
f.length; e > d; d++)
sa(f[d], g[d]);

, h);
```

☑ Pause on uncaught exceptions
☑ Pause on caught exceptions

```
Local
  this: Window
  a: n {0: li.demo-item, length: 1, prevObject: n, context: document}
  b: undefined
  c: undefined
  d: li.demo-item
  e: n {0: li.demo-item, length: 1, prevObject: n, context: document}
  f: 0
Closure
Global                                                     Window
```

```
va                        jquery-2.2.1.min.js:3
remove                    jquery-2.2.1.min.js:3
onRemoveLastItem                   script.js:284
dispatch                  jquery-2.2.1.min.js:3
n.event.add.r.handle      jquery-2.2.1.min.js:3
```

No DOM breakpoints

8) Editing in Elements Panel

What to do:

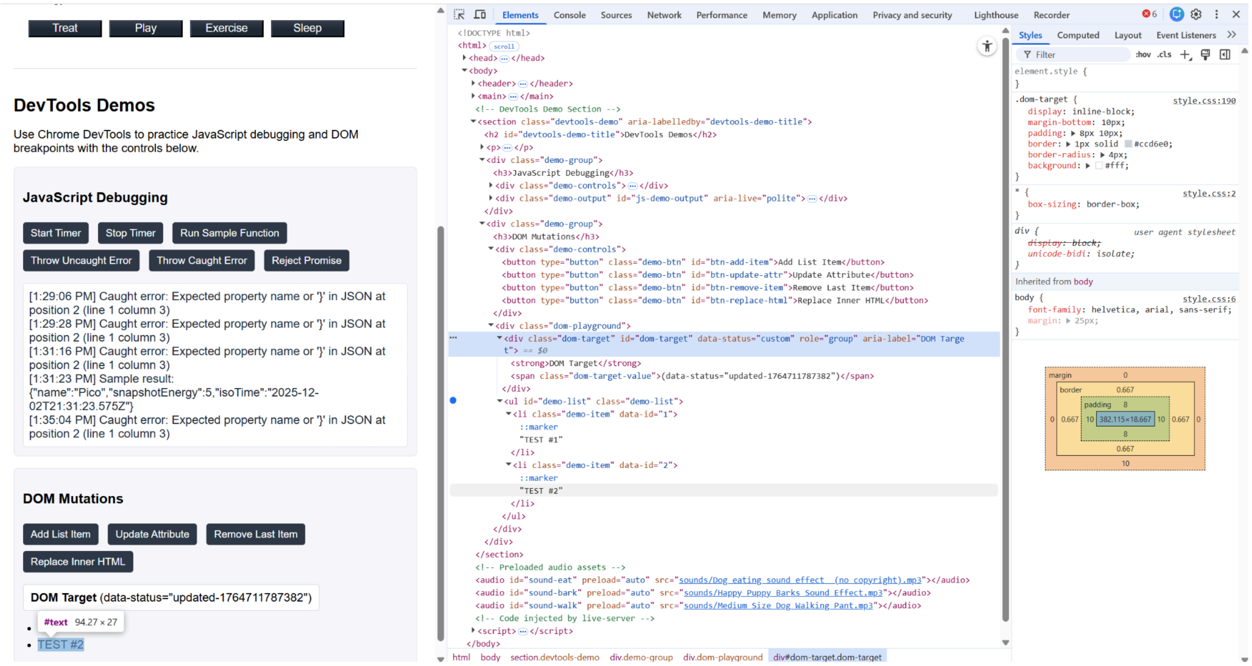
1. In the **Elements** panel, locate an element with the class `.demo-item` inside the DevTools demo section.

2. Double-click the text content of a `.demo-item` and edit it (for example, change “Original Item” to “Edited via DevTools”).

3. Press Enter and observe the change reflected immediately on the page.

4. Select `#dom-target` and modify one of its attributes (for example, add `title="Edited in DevTools"`).

5. Confirm that the new attribute value is visible in the page UI (e.g., hover to see the title tooltip).



DevTools Demos

Use Chrome DevTools to practice JavaScript debugging and DOM breakpoints with the controls below.

JavaScript Debugging

```
[1:29:06 PM] Caught error: Expected property name or '}' in JSON at position 2 (line 1 column 3)
[1:29:28 PM] Caught error: Expected property name or '}' in JSON at position 2 (line 1 column 3)
[1:31:16 PM] Caught error: Expected property name or '}' in JSON at position 2 (line 1 column 3)
[1:31:23 PM] Sample result: {"name":"Pico","snapshotEnergy":5,"isoTime":"2025-12-02T21:31:23.575Z"}
[1:35:04 PM] Caught error: Expected property name or '}' in JSON at position 2 (line 1 column 3)
```

DOM Mutations

DOM Target (data-status="updated-1764711787382")

-
- TEST #2

```
<!DOCTYPE html>
<html> scroll
  <head> … </head>
  <body>
    <header> … </header>
    <main> … </main>
    <!-- DevTools Demo Section -->
    <section class="devtools-demo" aria-labelledby="devtools-demo-title">
      <h2 id="devtools-demo-title">DevTools Demos</h2>
      <p> … </p>
      <div class="demo-group">
        <h3>JavaScript Debugging</h3>
        <div class="demo-controls"> … </div>
        <div class="demo-output" id="js-demo-output" aria-live="polite"> … </div>
      </div>
      <div class="demo-group">
        <h3>DOM Mutations</h3>
        <div class="demo-controls">
          <button type="button" class="demo-btn" id="btn-add-item">Add List Item</button>
          <button type="button" class="demo-btn" id="btn-update-attr">Update Attribute</button>
          <button type="button" class="demo-btn" id="btn-remove-item">Remove Last Item</button>
          <button type="button" class="demo-btn" id="btn-replace-html">Replace Inner HTML</button>
        </div>
        <div class="dom-playground">
          <div class="dom-target" id="dom-target" data-status="custom" role="group" aria-label="DOM Targe
          t"> == $0
            <strong>DOM Target</strong>
            <span class="dom-target-value">(data-status="updated-1764711787382")</span>
          </div>
          <ul id="demo-list" class="demo-list">
            <li class="demo-item" data-id="1">
              ::marker
              "TEST #1"
            </li>
            <li class="demo-item" data-id="2">
              ::marker
              "TEST #2"
            </li>
          </ul>
        </div>
      </div>
    </section>
    <!-- Preloaded audio assets -->
    <audio id="sound-eat" preload="auto" src="sounds/Dog eating sound effect  (no copyright).mp3"></audio>
    <audio id="sound-bark" preload="auto" src="sounds/Happy Puppy Barks Sound Effect.mp3"></audio>
    <audio id="sound-walk" preload="auto" src="sounds/Medium Size Dog Walking Pant.mp3"></audio>
    <!-- Code injected by live-server -->
    <script> … </script>
  </body>
```

html body section.devtools-demo div.demo-group div.dom-playground div#dom-target.dom-target

```
element.style {
}
.dom-target {                              style.css:190
    display: inline-block;
    margin-bottom: 10px;
    padding: ▶ 8px 10px;
    border: ▶ 1px solid #ccd6e0;
    border-radius: ▶ 4px;
    background: ▶ #fff;
}
* {                                        style.css:2
    box-sizing: border-box;
}
div {                          user agent stylesheet
    ~~display: block;~~
    unicode-bidi: isolate;
}
Inherited from body
body {                                     style.css:6
    font-family: helvetica, arial, sans-serif;
    margin: ▶ 25px;
}
```

```
margin              0
  border          0.667
    padding         8
0  0.667  10  382.115×18.667  10  0.667  0
                    8
                  0.667
                   10
```