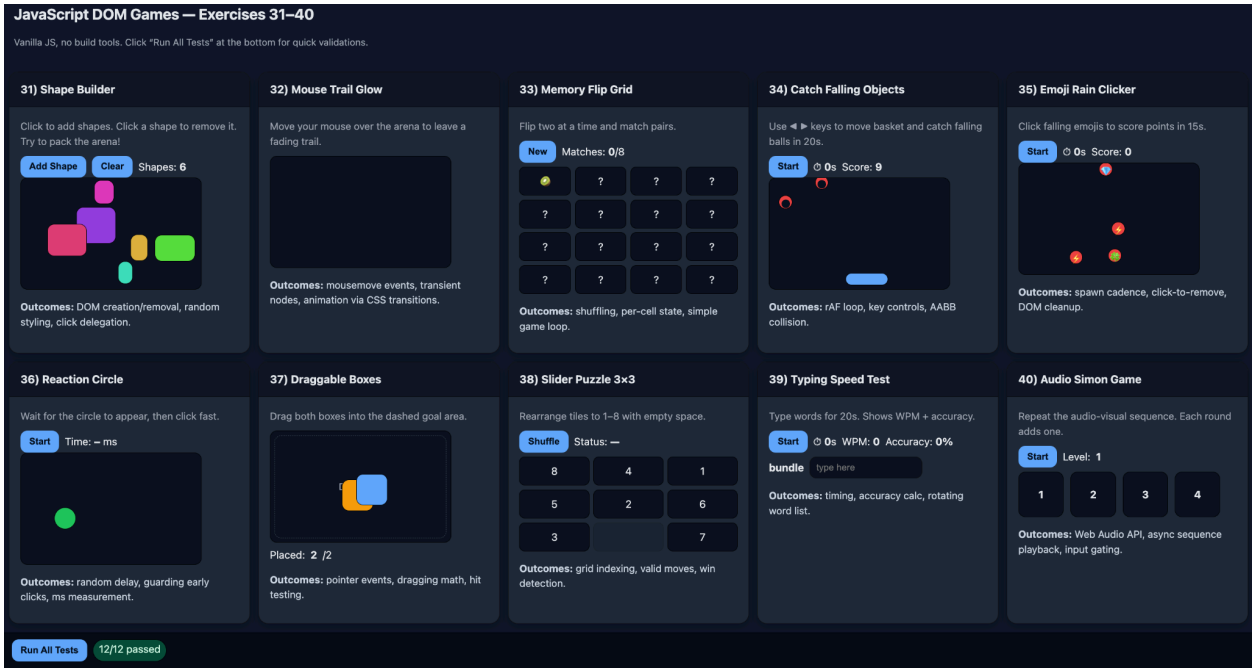
🎮 JavaScript DOM Games — Exercises 31–40



Hands-On DOM Coding for Creative Front-End Learners

Learning JavaScript is best done by doing — especially when it's fun.
This new set of **10 DOM-based mini-games** (Exercises 31–40) builds on previous collections, pushing your understanding of **interactivity, animation, and user input** even further.

Each mini-project explores a different concept through code you can see, touch, and experiment with directly in the browser. You'll not only sharpen your DOM manipulation skills, but also learn how to think like a front-end engineer — balancing logic, layout, and user feedback.

🧩 Overview: What's Inside

Each exercise is a **self-contained HTML card** — ready to copy, paste, or run from the provided ZIP.
Together, they cover a variety of essential front-end skills:

#	Game	Focus

31	**Shape Builder**	Creating, positioning, and removing DOM elements dynamically
32	**Mouse Trail Glow**	Tracking mouse movement and using transient animations
33	**Memory Flip Grid**	State logic, array shuffling, and matching logic
34	**Catch Falling Objects**	Animation loops, physics simulation, and keyboard control
35	**Emoji Rain Clicker**	Click events, timing control, and element cleanup
36	**Reaction Circle**	Measuring user reaction times and event timing
37	**Draggable Boxes**	Pointer events, hit detection, and dynamic feedback
38	**Slider Puzzle (3×3)**	Grid math, valid moves, and solving detection
39	**Typing Speed Test**	Keyboard events, timing, scoring, and accuracy tracking
40	**Audio Simon Game**	Web Audio API, async sequencing, and memory pattern logic

---

💡 Learning Outcomes

By completing these games, learners gain practical mastery over the DOM and browser APIs. Here's what you'll walk away with:

1. Dynamic DOM Manipulation

You'll create, style, and remove elements on the fly using `createElement`, event delegation, and inline CSS updates.

2. Event-Driven Programming

From `mousemove` to `keydown` and `pointermove`, you'll master how events drive user interactions in real time.

3. Animation and Timing

Games like **Catch Falling Objects** and **Emoji Rain Clicker** teach how to combine `requestAnimationFrame` with timers and smooth motion logic.

4. Logic and State Management

You'll learn how to store game states (like "playing," "paused," or "matched") using flags and

conditions — the foundation of larger web apps.

5. Layout and Geometry

By reading bounding boxes and coordinates (`getBoundingClientRect`), you'll understand how to manage collision, distance, and grid positioning.

6. Persistence and Feedback

Each exercise teaches UX thinking — showing scores, status, or feedback immediately to reinforce learning and maintain flow.

7. Advanced Browser APIs

Games like **Audio Simon** and **Resize Puzzle** go beyond the basics, introducing modern APIs like the **Web Audio API** and `ResizeObserver`.

🧠 How to Use This Project

1. **Download the ZIP:**
   👉 Download Exercises 31–40

2. **Open `index.html` in your browser.**
   Each exercise is live on the page — click buttons, drag, type, and explore.

3. **Use the "Run All Tests" button.**
   This verifies event wiring and logic integrity across all games.

4. **Inspect the code.**
   Each script is inline and heavily commented for learning clarity.

🪄 Skills You'll Strengthen

- Writing **clean, reusable functions** that handle DOM interaction
- Using **asynchronous logic** for game timing
- Practicing **event coordination** between elements
- Building **interactive interfaces** from scratch without frameworks

- Understanding how to design with **user feedback loops** in mind

---

🎓 Why It Matters

These aren't just fun browser toys — they're **real-world simulations** of how JavaScript drives interactivity everywhere: from dashboards to animations, from games to UX design.

Each exercise reinforces **problem-solving through code** — helping you break complex behaviors (like collisions or timing) into logical, testable parts.

Master these, and you'll have the foundation to build everything from **educational tools** to **interactive data visualizations** or **browser-based games** — all in pure vanilla JavaScript.

🏁 Final Thoughts

Exercises 31–40 are a continuation of a hands-on coding journey that grows in both creativity and complexity.
They show that **the DOM isn't just a structure** — it's a playground where logic and imagination meet.

Take the time to modify, remix, and expand these exercises. Add your own colors, sounds, or levels.
That's where learning truly happens — not just by reading code, but by bending it to your will.

```
<!doctype html>
<html lang="en">
<head>
<meta charset="utf-8"/>
<meta name="viewport" content="width=device-width,initial-scale=1"/>
<title>JavaScript DOM Games — Exercises 31–40</title>
<style>

:root{--bg:#0f172a;--panel:#111827;--card:#1f2937;--accent:#60a5fa;--ok:#10b981;--bad:#ef444
4;--muted:#9ca3af}
  body{margin:0;background:var(--bg);color:#e5e7eb;font:15px/1.5 system-ui,Segoe
UI,Roboto,Helvetica,Arial,sans-serif}
  header{padding:1.1rem 1.2rem .4rem}
  h1{margin:0;font-size:1.35rem}

.wrap{display:grid;grid-template-columns:repeat(auto-fill,minmax(320px,1fr));gap:12px;padding:
12px}
  .card{background:var(--card);border:1px solid
```

```
#111;border-radius:14px;overflow:hidden;box-shadow:0 4px 14px rgba(0,0,0,.25)}
  .card header{background:var(--panel);padding:.8rem 1rem;border-bottom:1px solid #000}
  .card h2{margin:0;font-size:1rem}
  .card .body{padding:.8rem 1rem}
  .meta{font-size:.9rem;color:var(--muted);margin:.3rem 0 .6rem}
  button{background:var(--accent);border:0;color:#06121f;padding:.5rem
.8rem;border-radius:10px;cursor:pointer;font-weight:600}
  .row{display:flex;gap:8px;align-items:center;flex-wrap:wrap}
  .small{font-size:.85rem;color:#9ca3af}
  .note{font-size:.9rem;color:#cbd5e1}
  .arena{position:relative;width:260px;height:160px;border:1px solid
#000;border-radius:10px;background:#0b1220;overflow:hidden}
  .grid{display:grid;gap:6px}
  .cell{height:40px;border-radius:8px;border:1px solid
#000;background:#0b1220;display:grid;place-items:center;cursor:pointer}
  input[type=text],input[type=number]{padding:.45rem .6rem;border-radius:10px;border:1px solid
#111;background:#0b1220;color:#e5e7eb}
  .testbar{position:sticky;bottom:0;background:#030712cc;border-top:1px solid
#000;padding:.6rem 1rem;display:flex;gap:8px;backdrop-filter: blur(6px)}
  .pill{border-radius:999px;padding:.1rem .5rem;border:1px solid #000}
  .ok{background:#064e3b;color:#d1fae5}
  .bad{background:#7f1d1d;color:#fee2e2}
  .hidden{display:none}
  .shape{position:absolute;border-radius:10px;border:1px solid #000}

.dot{position:absolute;width:8px;height:8px;border-radius:50%;background:#60a5fa;opacity:.9;p
ointer-events:none;filter:blur(0.5px);transition:transform .5s, opacity .5s}

.basket{position:absolute;bottom:6px;left:110px;width:60px;height:16px;border-radius:10px;back
ground:#60a5fa;border:1px solid #000}

.fall{position:absolute;top:-20px;width:18px;height:18px;border-radius:50%;display:grid;place-ite
ms:center;border:1px solid #000;background:#ef4444;color:#06121f;font-weight:700}
  .pad{width:64px;height:64px;border-radius:10px;border:1px solid
#000;background:#0b1220;display:grid;place-items:center;font-weight:700;cursor:pointer}
  .pad.on{filter:brightness(1.7)}
  .goal{position:absolute;inset:6px;border:1px dashed
#4b5563;border-radius:10px;display:grid;place-items:center;color:#9ca3af;font-size:.85rem}
</style>
</head>
<body>
<header>
  <h1>JavaScript DOM Games — Exercises 31–40</h1>
  <p class="small">Vanilla JS, no build tools. Click “Run All Tests” at the bottom for quick
validations.</p>
</header>

<div class="wrap">
```

```
  <!-- 31) Shape Builder -->
  <section class="card" id="ex31">
    <header><h2>31) Shape Builder</h2></header>
    <div class="body">
      <div class="meta">Click to add shapes. Click a shape to remove it. Try to pack the
arena!</div>
      <div class="row">
        <button id="ex31-add">Add Shape</button>
        <button id="ex31-clear">Clear</button>
        <span>Shapes: <b id="ex31-count">0</b></span>
      </div>
      <div class="arena" id="ex31-arena"></div>
      <p class="note"><strong>Outcomes:</strong> DOM creation/removal, random styling, click
delegation.</p>
    </div>
  </section>

  <!-- 32) Mouse Trail Glow -->
  <section class="card" id="ex32">
    <header><h2>32) Mouse Trail Glow</h2></header>
    <div class="body">
      <div class="meta">Move your mouse over the arena to leave a fading trail.</div>
      <div class="arena" id="ex32-arena"></div>
      <p class="note"><strong>Outcomes:</strong> mousemove events, transient nodes,
animation via CSS transitions.</p>
    </div>
  </section>

  <!-- 33) Memory Flip Grid -->
  <section class="card" id="ex33">
    <header><h2>33) Memory Flip Grid</h2></header>
    <div class="body">
      <div class="meta">Flip two at a time and match pairs.</div>
      <div class="row">
        <button id="ex33-new">New</button>
        <span>Matches: <b id="ex33-matches">0</b>/8</span>
      </div>
      <div class="grid" id="ex33-grid"
style="grid-template-columns:repeat(4,1fr);margin-top:6px"></div>
      <p class="note"><strong>Outcomes:</strong> shuffling, per-cell state, simple game
loop.</p>
    </div>
  </section>

  <!-- 34) Catch Falling Objects -->
  <section class="card" id="ex34">
    <header><h2>34) Catch Falling Objects</h2></header>
```

```
    <div class="body">
      <div class="meta">Use ◂ ▸ keys to move basket and catch falling balls in 20s.</div>
      <div class="row">
        <button id="ex34-start">Start</button>
        <span>⏱ <b id="ex34-time">20</b>s</span>
        <span>Score: <b id="ex34-score">0</b></span>
      </div>
      <div class="arena" id="ex34-arena"><div class="basket" id="ex34-basket"></div></div>
      <p class="note"><strong>Outcomes:</strong> rAF loop, key controls, AABB collision.</p>
    </div>
  </section>

  <!-- 35) Emoji Rain Clicker -->
  <section class="card" id="ex35">
    <header><h2>35) Emoji Rain Clicker</h2></header>
    <div class="body">
      <div class="meta">Click falling emojis to score points in 15s.</div>
      <div class="row">
        <button id="ex35-start">Start</button>
        <span>⏱ <b id="ex35-time">15</b>s</span>
        <span>Score: <b id="ex35-score">0</b></span>
      </div>
      <div class="arena" id="ex35-arena"></div>
      <p class="note"><strong>Outcomes:</strong> spawn cadence, click-to-remove, DOM
cleanup.</p>
    </div>
  </section>

  <!-- 36) Reaction Circle -->
  <section class="card" id="ex36">
    <header><h2>36) Reaction Circle</h2></header>
    <div class="body">
      <div class="meta">Wait for the circle to appear, then click fast.</div>
      <div class="row">
        <button id="ex36-start">Start</button>
        <span>Time: <b id="ex36-ms">–</b> ms</span>
      </div>
      <div class="arena" id="ex36-arena"></div>
      <p class="note"><strong>Outcomes:</strong> random delay, guarding early clicks, ms
measurement.</p>
    </div>
  </section>

  <!-- 37) Draggable Boxes -->
  <section class="card" id="ex37">
    <header><h2>37) Draggable Boxes</h2></header>
    <div class="body">
      <div class="meta">Drag both boxes into the dashed goal area.</div>
```

```
      <div class="arena" id="ex37-arena">
        <div class="goal">Drop here</div>
      </div>
      <div class="row" style="margin-top:6px">Placed: <b id="ex37-placed">0</b>/2</div>
      <p class="note"><strong>Outcomes:</strong> pointer events, dragging math, hit
testing.</p>
    </div>
  </section>

  <!-- 38) Slider Puzzle 3×3 -->
  <section class="card" id="ex38">
    <header><h2>38) Slider Puzzle 3×3</h2></header>
    <div class="body">
      <div class="meta">Rearrange tiles to 1–8 with empty space.</div>
      <div class="row"><button id="ex38-shuffle">Shuffle</button><span>Status: <b
id="ex38-status">—</b></span></div>
      <div class="grid" id="ex38-grid"
style="grid-template-columns:repeat(3,1fr);margin-top:6px"></div>
      <p class="note"><strong>Outcomes:</strong> grid indexing, valid moves, win
detection.</p>
    </div>
  </section>

  <!-- 39) Typing Speed Test -->
  <section class="card" id="ex39">
    <header><h2>39) Typing Speed Test</h2></header>
    <div class="body">
      <div class="meta">Type words for 20s. Shows WPM + accuracy.</div>
      <div class="row">
        <button id="ex39-start">Start</button>
        <span>⏱ <b id="ex39-time">20</b>s</span>
        <span>WPM: <b id="ex39-wpm">0</b></span>
        <span>Accuracy: <b id="ex39-acc">0%</b></span>
      </div>
      <div class="row" style="margin-top:6px"><b id="ex39-word">—</b><input id="ex39-in"
type="text" disabled placeholder="type here"></div>
      <p class="note"><strong>Outcomes:</strong> timing, accuracy calc, rotating word list.</p>
    </div>
  </section>

  <!-- 40) Audio Simon Game -->
  <section class="card" id="ex40">
    <header><h2>40) Audio Simon Game</h2></header>
    <div class="body">
      <div class="meta">Repeat the audio-visual sequence. Each round adds one.</div>
      <div class="row"><button id="ex40-start">Start</button> Level: <b
id="ex40-level">0</b></div>
      <div class="row" style="margin-top:6px;gap:10px">
```

```
        <div class="pad" id="p0">1</div>
        <div class="pad" id="p1">2</div>
        <div class="pad" id="p2">3</div>
        <div class="pad" id="p3">4</div>
      </div>
      <p class="note"><strong>Outcomes:</strong> Web Audio API, async sequence playback,
input gating.</p>
    </div>
  </section>

</div>

<!-- Test bar -->
<div class="testbar">
  <button id="run-tests">Run All Tests</button>
  <span id="test-summary" class="pill ok hidden"></span>
  <span id="test-errors" class="pill bad hidden"></span>
</div>

<script>
/* helpers */
const $ = s => document.querySelector(s);
const $$ = s => Array.from(document.querySelectorAll(s));
const rand = n => Math.floor(Math.random()*n);
const wait = ms => new Promise(r=>setTimeout(r,ms));
const clamp = (v,a,b)=>Math.max(a,Math.min(b,v));

/* 31) Shape Builder */
(function(){
  const arena=$('#ex31-arena'), add=$('#ex31-add'), clear=$('#ex31-clear'),
countEl=$('#ex31-count');
  function addShape(){
    const d=document.createElement('div'); d.className='shape';
    const w=20+rand(40), h=20+rand(40);
    d.style.width=w+'px'; d.style.height=h+'px';
    d.style.background=`hsl(${rand(360)},70%,55%)`;
    const r=arena.getBoundingClientRect();
    d.style.left=rand(Math.max(1, r.width-w))+'px';
    d.style.top=rand(Math.max(1, r.height-h))+'px';
    d.addEventListener('click', e=>{ e.stopPropagation(); d.remove(); update(); });
    arena.appendChild(d); update();
  }
  function update(){ countEl.textContent = arena.querySelectorAll('.shape').length; }
  add.addEventListener('click', addShape);
  clear.addEventListener('click', ()=>{ arena.innerHTML=''; update(); });
  arena.addEventListener('click', addShape);
  window._ex31={arena,add,clear,countEl};
})();
```

```
/* 32) Mouse Trail Glow */
(function(){
  const arena=$('#ex32-arena');
  arena.addEventListener('mousemove', e=>{
    const r=arena.getBoundingClientRect();
    const x=e.clientX-r.left, y=e.clientY-r.top;
    const d=document.createElement('div'); d.className='dot';
    d.style.left=(x-4)+'px'; d.style.top=(y-4)+'px';
    arena.appendChild(d);
    requestAnimationFrame(()=>{ d.style.opacity='0'; d.style.transform='translateY(-6px)'; });
    setTimeout(()=>d.remove(), 600);
  });
  window._ex32={arena};
})();

/* 33) Memory Flip Grid */
(function(){
  const grid=$('#ex33-grid'), btn=$('#ex33-new'), matchesEl=$('#ex33-matches');
  const icons=['🍎','🍌','🍇','🍒','🍊','🥝','🍑','🍉'];
  let first=null, lock=false, matches=0, cards=[];
  function build(){
    matches=0; matchesEl.textContent=matches;
    const pool=icons.concat(icons).sort(()=>Math.random()-0.5);
    grid.innerHTML=''; cards=[]; first=null; lock=false;
    pool.forEach((sym,i)=>{
      const d=document.createElement('div'); d.className='cell'; d.textContent='?';
d.dataset.sym=sym;
      d.addEventListener('click', ()=>flip(d));
      grid.appendChild(d); cards.push(d);
    });
  }
  function flip(c){
    if(lock||c.dataset.done==='1'||c===first) return;
    c.textContent=c.dataset.sym;
    if(!first){ first=c; return; }
    lock=true;
    if(first.dataset.sym===c.dataset.sym){
      first.dataset.done=c.dataset.done='1'; matches++; matchesEl.textContent=matches;
lock=false; first=null;
    }else{
      setTimeout(()=>{ first.textContent='?'; c.textContent='?'; first=null; lock=false; },600);
    }
  }
  btn.addEventListener('click', build);
  build();
  window._ex33={grid,btn};
})();
```

```
/* 34) Catch Falling Objects */
(function(){
  const arena=$('#ex34-arena'), basket=$('#ex34-basket');
  const start=$('#ex34-start'), tEl=$('#ex34-time'), scoreEl=$('#ex34-score');
  let running=false, last=0, spawnAcc=0, score=0, t=20, timer=0, raf=0, balls=[];
  function spawn(){
    const d=document.createElement('div'); d.className='fall'; d.textContent='●';
    d.style.left=rand(arena.clientWidth-18)+'px'; d.style.top='-20px';
    d.dataset.vy = 60+rand(80);
    arena.appendChild(d); balls.push(d);
  }
  function loop(ts){
    if(!running) return;
    if(!last) last=ts; const dt=(ts-last)/1000; last=ts;
    spawnAcc+=dt; while(spawnAcc>=0.6){ spawnAcc-=0.6; spawn(); }
    balls.forEach(b=>{
      const y=(parseFloat(b.style.top)||-20)+dt*parseFloat(b.dataset.vy);
      b.style.top=y+'px';
    });
    const br=basket.getBoundingClientRect(), ar=arena.getBoundingClientRect();
    balls = balls.filter(b=>{
      const r=b.getBoundingClientRect();
      const withinX = r.left < br.right && r.right > br.left;
      const withinY = r.bottom >= br.top && r.bottom <= br.bottom+18;
      if(withinX && withinY){ score++; scoreEl.textContent=score; b.remove(); return false; }
      if(parseFloat(b.style.top)>arena.clientHeight+30){ b.remove(); return false; }
      return true;
    });
    raf=requestAnimationFrame(loop);
  }
  window.addEventListener('keydown', e=>{
    if(!running) return;
    const x=parseFloat(basket.style.left)||110;
    if(e.key==='ArrowLeft') basket.style.left=clamp(x-20, 0, arena.clientWidth-60)+'px';
    if(e.key==='ArrowRight') basket.style.left=clamp(x+20, 0, arena.clientWidth-60)+'px';
  });
  start.addEventListener('click', ()=>{
    if(running) return; running=true; score=0; scoreEl.textContent=score; t=20; tEl.textContent=t;
    balls.forEach(b=>b.remove()); balls=[]; last=0; spawnAcc=0; basket.style.left='110px';
    clearInterval(timer); timer=setInterval(()=>{ t--; tEl.textContent=t; if(t<=0){ clearInterval(timer);
running=false; } },1000);
    raf=requestAnimationFrame(loop);
  });
  window._ex34={start,arena,basket};
})();

/* 35) Emoji Rain Clicker */
```

```
(function(){
  const arena=$('#ex35-arena'), start=$('#ex35-start'), tEl=$('#ex35-time'),
scoreEl=$('#ex35-score');
  const EMO=['⭐','🎈','🍀','💎','🔥','⚡','🌙','🍩'];
  let running=false, last=0, spawnAcc=0, t=15, score=0, timer=0, raf=0, nodes=[];
  function spawn(){
    const d=document.createElement('div'); d.className='fall';
d.textContent=EMO[rand(EMO.length)];
    d.style.left=rand(arena.clientWidth-18)+'px'; d.style.top='-22px';
    d.dataset.vy = 70+rand(120);
    d.addEventListener('click', ()=>{ if(!running) return; score++; scoreEl.textContent=score;
d.remove(); nodes=nodes.filter(n=>n!==d); });
    arena.appendChild(d); nodes.push(d);
  }
  function loop(ts){
    if(!running) return;
    if(!last) last=ts; const dt=(ts-last)/1000; last=ts;
    spawnAcc+=dt; while(spawnAcc>=0.45){ spawnAcc-=0.45; spawn(); }
    nodes.forEach(n=>{ n.style.top = (parseFloat(n.style.top)||-22) + dt*parseFloat(n.dataset.vy) +
'px'; });
    nodes = nodes.filter(n=>{ if(parseFloat(n.style.top)>arena.clientHeight+30){ n.remove();
return false; } return true; });
    raf=requestAnimationFrame(loop);
  }
  start.addEventListener('click', ()=>{
    if(running) return; running=true; t=15; score=0; scoreEl.textContent=score; tEl.textContent=t;
    nodes.forEach(n=>n.remove()); nodes=[]; last=0; spawnAcc=0;
    clearInterval(timer); timer=setInterval(()=>{ t--; tEl.textContent=t; if(t<=0){ clearInterval(timer);
running=false; } },1000);
    raf=requestAnimationFrame(loop);
  });
  window._ex35={start,arena};
})();

/* 36) Reaction Circle */
(function(){
  const arena=$('#ex36-arena'), start=$('#ex36-start'), msEl=$('#ex36-ms');
  let armed=false, shown=0, circle=null, to=0;
  function clearCircle(){ if(circle){ circle.remove(); circle=null; } }
  start.addEventListener('click', ()=>{
    armed=false; clearTimeout(to); msEl.textContent='–'; clearCircle();
    to=setTimeout(()=>{
      const d=document.createElement('div'); d.className='shape'; d.style.borderRadius='50%';
      d.style.width='30px'; d.style.height='30px'; d.style.background='#22c55e';
      d.style.left=rand(arena.clientWidth-30)+'px'; d.style.top=rand(arena.clientHeight-30)+'px';
      d.addEventListener('click', ()=>{ if(!armed) return;
msEl.textContent=String(Date.now()-shown); armed=false; clearCircle(); });
      arena.appendChild(d); shown=Date.now(); armed=true; circle=d;
```

```
    }, 300+rand(1200));
  });
  window._ex36={start,arena};
})();

/* 37) Draggable Boxes */
(function(){
  const arena=$('#ex37-arena'), placedEl=$('#ex37-placed');
  const goal = arena.querySelector('.goal');
  const boxes=[];
  function mkBox(x,y,c){
    const d=document.createElement('div'); d.className='shape'; d.style.width='44px';
d.style.height='44px'; d.style.left=x+'px'; d.style.top=y+'px'; d.style.background=c;
    d.style.cursor='grab';
    let ox=0, oy=0, dragging=false;
    d.addEventListener('pointerdown', e=>{ dragging=true; d.setPointerCapture(e.pointerId);
d.style.cursor='grabbing'; ox=e.clientX-parseFloat(d.style.left);
oy=e.clientY-parseFloat(d.style.top); });
    d.addEventListener('pointermove', e=>{ if(!dragging) return; const nx=e.clientX-ox,
ny=e.clientY-oy; d.style.left=clamp(nx,0,arena.clientWidth-44)+'px';
d.style.top=clamp(ny,0,arena.clientHeight-44)+'px'; check(); });
    d.addEventListener('pointerup', e=>{ dragging=false; d.style.cursor='grab'; });
    arena.appendChild(d); boxes.push(d);
  }
  function rect(el){ return el.getBoundingClientRect(); }
  function overlap(a,b){ const ar=rect(a), br=rect(b); return
!(ar.right<br.left||ar.left>br.right||ar.bottom<br.top||ar.top>br.bottom); }
  function check(){
    let p=0; boxes.forEach(b=>{ p += overlap(b,goal)?1:0; }); placedEl.textContent=p;
  }
  mkBox(10,10,'#f59e0b'); mkBox(70,50,'#60a5fa');
  window._ex37={arena};
})();

/* 38) Slider Puzzle 3×3 */
(function(){
  const grid=$('#ex38-grid'), shuffleBtn=$('#ex38-shuffle'), status=$('#ex38-status');
  let tiles=[];
  function build(arr){
    grid.innerHTML=''; tiles=arr||shuffle([0,1,2,3,4,5,6,7,8]);
    tiles.forEach((n,i)=>{
      const d=document.createElement('div'); d.className='cell'; d.textContent= n===8 ? '' :
(n+1);
      d.style.opacity=n===8?'.2':'1';
      d.addEventListener('click',()=>move(i));
      grid.appendChild(d);
    });
    status.textContent='—';
```

```
  }
  function idxToXY(i){ return [i%3, Math.floor(i/3)]; }
  function xyToIdx(x,y){ return y*3+x; }
  function move(i){
    const empty = tiles.indexOf(8);
    const [x,y]=idxToXY(i), [ex,ey]=idxToXY(empty);
    const can = (x===ex && Math.abs(y-ey)===1) || (y===ey && Math.abs(x-ex)===1);
    if(!can) return;
    [tiles[i], tiles[empty]]=[tiles[empty], tiles[i]];
    build(tiles);
    if(isSolved()) status.textContent='Solved! 🎉';
  }
  function isSolved(){ for(let i=0;i<8;i++) if(tiles[i]!==i) return false; return true; }
  function shuffle(a){ return a.slice().sort(()=>Math.random()-0.5); }
  shuffleBtn.addEventListener('click', ()=>build());
  build([0,1,2,3,4,5,6,7,8]);
  window._ex38={grid,shuffleBtn};
})();

/* 39) Typing Speed Test */
(function(){
  const words =
['array','object','event','promise','module','bundle','script','render','update','virtual','socket','driver','t
hread','kernel','binary','layout','reactive','pointer','matrix','canvas'];
  const start=$('#ex39-start'), tEl=$('#ex39-time'), wpmEl=$('#ex39-wpm'), accEl=$('#ex39-acc'),
wordEl=$('#ex39-word'), input=$('#ex39-in');
  let t=20, timer=0, typed=0, correct=0, chars=0;
  function newWord(){ const w=words[rand(words.length)]; wordEl.textContent=w; input.value='';
}
  start.addEventListener('click', ()=>{
    t=20; tEl.textContent=t; typed=correct=chars=0; input.disabled=false; input.focus();
newWord();
    clearInterval(timer); timer=setInterval(()=>{ t--; tEl.textContent=t; if(t<=0){ clearInterval(timer);
input.disabled=true; } },1000);
  });
  input.addEventListener('keydown', e=>{
    if(e.key==='Enter'){
      const w=wordEl.textContent.trim(); const v=input.value.trim(); typed++; chars+=v.length;
if(v===w) correct++; newWord();
      const wpm = Math.round((correct*5)/((20-t)/60 || 1));
      const acc = typed? Math.round(100*correct/typed):0;
      wpmEl.textContent=String(wpm); accEl.textContent=acc+'%';
    }
  });
  window._ex39={start,input};
})();

/* 40) Audio Simon Game */
```

```
(function(){
  const start=$('#ex40-start'), levelEl=$('#ex40-level');
  const pads=$$('#ex40 .pad');
  let seq=[], idx=0, listening=false, ctx=null;
  const freqs=[329.63,392.00,261.63,523.25];
  function beep(i,ms=350){
    if(!ctx) ctx=new (window.AudioContext||window.webkitAudioContext)();
    const o=ctx.createOscillator(), g=ctx.createGain();
    o.type='sine'; o.frequency.value=freqs[i]; o.connect(g); g.connect(ctx.destination);
    g.gain.setValueAtTime(0.001, ctx.currentTime);
    g.gain.exponentialRampToValueAtTime(0.2, ctx.currentTime+0.01);
    g.gain.exponentialRampToValueAtTime(0.001, ctx.currentTime+ms/1000);
    o.start(); o.stop(ctx.currentTime+ms/1000);
  }
  function flash(i,ms=350){ pads[i].classList.add('on'); beep(i,ms);
setTimeout(()=>pads[i].classList.remove('on'), ms); }
  async function play(){
    listening=false;
    for(const i of seq){ flash(i); await wait(450); }
    idx=0; listening=true;
  }
  function next(){ seq.push(rand(4)); levelEl.textContent=seq.length; play(); }
  pads.forEach((p,i)=>p.addEventListener('click', ()=>{
    if(!listening) return;
    flash(i,200);
    if(i===seq[idx]){ idx++; if(idx===seq.length){ listening=false; setTimeout(next,500); } }
    else{ listening=false; seq=[]; levelEl.textContent='0'; }
  }));
  start.addEventListener('click', ()=>{ seq=[]; levelEl.textContent='0'; next(); });
  window._ex40={start};
})();

/* ------------------------
   Minimal Test Harness
------------------------*/
(function(){
  const btn=$('#run-tests'), sum=$('#test-summary'), err=$('#test-errors');
  let logs=[];
  const ok = m=>logs.push({ok:true,msg:m});
  const bad = m=>logs.push({ok:false,msg:m});

  async function t31(){ const {arena,add}=window._ex31; add.click(); ok('ex31 add');
arena.click(); ok('ex31 arena click adds'); }
  async function t32(){ const {arena}=window._ex32; arena.dispatchEvent(new
MouseEvent('mousemove',{bubbles:true,clientX:10,clientY:10})); ok('ex32 trail'); }
  async function t33(){ const {grid,btn}=window._ex33; btn.click(); ok('ex33 new'); const
c=grid.querySelector('.cell'); c&&c.click(); ok('ex33 flip'); }
  async function t34(){ const {start}=window._ex34; start.click(); ok('ex34 start'); }
```

```
  async function t35(){ const {start}=window._ex35; start.click(); ok('ex35 start'); }
  async function t36(){ const {start}=window._ex36; start.click(); ok('ex36 start'); }
  async function t37(){ const {arena}=window._ex37; ok('ex37 arena'); }
  async function t38(){ const {grid}=window._ex38; grid.querySelector('.cell')?.click(); ok('ex38
click'); }
  async function t39(){ const {start,input}=window._ex39; start.click(); input.value='array';
input.dispatchEvent(new KeyboardEvent('keydown',{key:'Enter'})); ok('ex39 type'); }
  async function t40(){ const {start}=window._ex40; start.click(); ok('ex40 start'); }

  btn.addEventListener('click', async ()=>{
    logs=[];
    const tests=[t31,t32,t33,t34,t35,t36,t37,t38,t39,t40];
    for(const t of tests){ try{ await t(); }catch(e){ bad(t.name+': '+e.message); } }
    const pass = logs.filter(l=>l.ok).length, fail = logs.length-pass;
    sum.textContent = `${pass}/${logs.length} passed`; err.textContent = `${fail} failed`;
    sum.classList.remove('hidden'); err.classList.remove('hidden');
    sum.className='pill '+(fail?'bad':'ok'); err.className='pill '+(fail?'bad':'ok');
    if(!fail) err.classList.add('hidden');
  });
})();
</script>
</body>
</html>
```