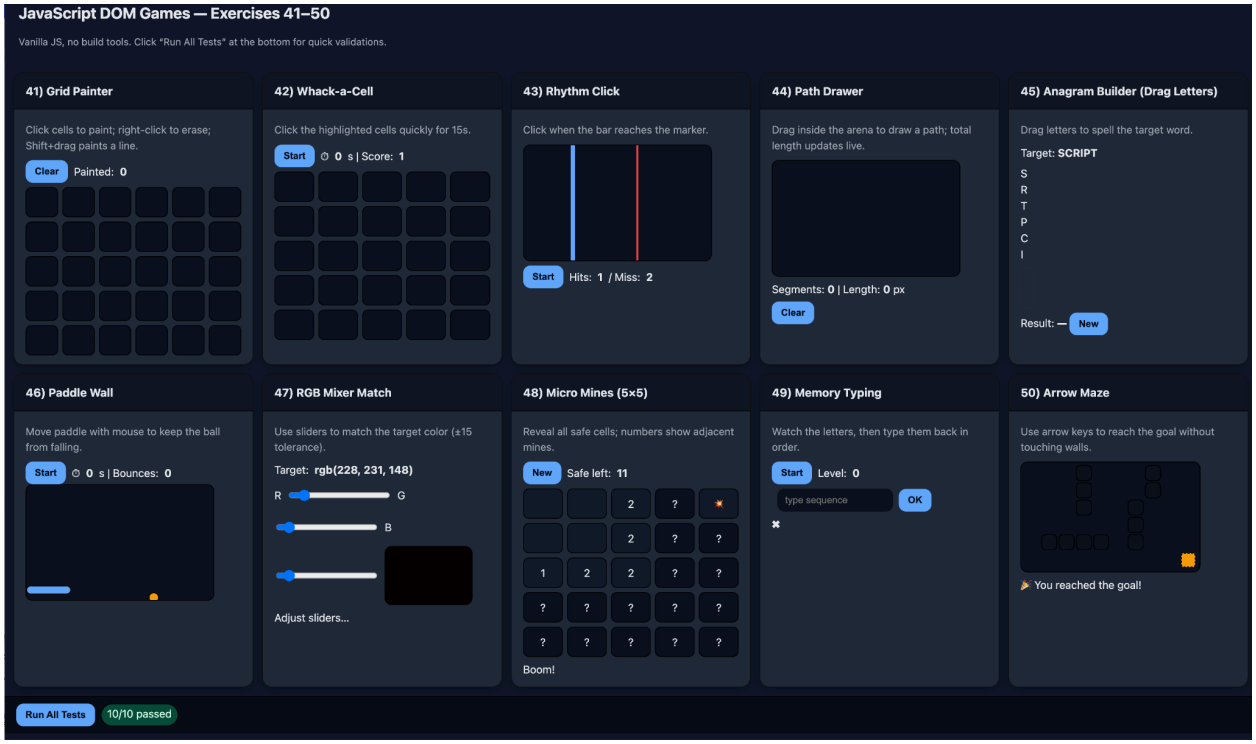
🎮 JavaScript DOM Games — Exercises 41–50



Learn by Building Interactive Browser Games with Vanilla JavaScript

JavaScript comes alive when you bring the DOM to life — and that's exactly what this next set of **10 hands-on exercises** does.
From painting grids to solving mazes, this collection (Exercises 41–50) dives into the kind of interactive logic that powers real-world interfaces, teaching you how to **think like a front-end engineer** while keeping things playful.

Every game is built using **pure JavaScript**, **HTML**, and **CSS** — no frameworks, no libraries — just creativity, logic, and the DOM.

🧩 Overview: What's Inside

Each mini-project is self-contained in the downloadable bundle:
👉 Download Exercises 41–50 (ZIP)

#	Game	Focus Area
41	**Grid Painter**	Mouse events, painting, and event delegation
42	**Whack-a-Cell**	Timers, random highlights, and click accuracy
43	**Rhythm Click**	Animation timing, precision input, and feedback
44	**Path Drawer**	Pointer capture, coordinates, and geometry
45	**Anagram Builder**	Drag & drop API and string manipulation
46	**Paddle Wall**	Motion, collision detection, and gameplay loops
47	**RGB Mixer Match**	Sliders, color theory, and dynamic styling
48	**Micro Mines (5×5)**	Recursive logic and neighborhood analysis
49	**Memory Typing**	Sequenced display and async gameplay
50	**Arrow Maze**	Keyboard controls and boundary collision

💡 Learning Objectives

These exercises don't just teach JavaScript syntax — they teach **how to build real interactions** in the browser. Here's what you'll gain:

1. DOM Creation and Event Handling

Games like **Grid Painter** and **Anagram Builder** show how to create, modify, and move DOM elements dynamically using the `createElement`, `appendChild`, and event APIs.

2. Timing and Animation

From **Whack-a-Cell** to **Rhythm Click**, you'll learn how to sync animations with user actions using `setInterval` and `requestAnimationFrame`.

3. State Management and Game Logic

Micro Mines and **Memory Typing** teach how to manage internal states, track progress, and control game flow — essential concepts for web apps and UI design.

4. Input and Interaction Patterns

You'll build skills across multiple input types:

- **Mouse events** (painting, dragging, clicking)
- **Keyboard controls** (Arrow Maze)
- **Pointer events** (Path Drawer)
- **Range sliders** (RGB Mixer)

5. Geometry, Collisions, and Physics

Paddle Wall introduces velocity, boundaries, and collision detection — the foundation of interactive motion and simple game physics.

6. Async and Sequenced Gameplay

Memory Typing and **Rhythm Click** demonstrate how to manage timed sequences with asynchronous functions and promises for smooth user experiences.

🧠 What You'll Learn by Doing

- How to structure code into small, reusable functions
- Techniques for event-driven programming and game loops
- DOM traversal and real-time style manipulation
- Handling user feedback with responsive UI design
- Building interactive components that feel alive

Every project is written in **pure JavaScript** — keeping you close to the fundamentals while providing solid patterns to build upon.

🧪 Testing and Validation

At the bottom of the file, you'll find a **"Run All Tests"** button.
It quickly validates that each mini-game is wired correctly and functioning — a simple way to verify that the events, timers, and key logic blocks respond as intended.

This makes the project great for **classrooms, workshops, or self-paced learning**, allowing

students to instantly confirm their progress.

🔧 How to Get Started

1. **Download the ZIP file:**
   JavaScript DOM Games (Exercises 41–50)

2. **Open `index.html`** in your browser.

3. **Play each exercise** directly in your browser window — no setup required.

4. **Inspect and edit** the inline JavaScript for each section to experiment and learn.

Each section is modular, so you can easily extract, remix, or expand individual games into standalone web apps.

🚀 Skills Strengthened

By completing this set, you'll have strengthened your abilities in:

- DOM manipulation & event handling
- Timing loops & animation frames
- Collision logic & coordinate math
- Keyboard & mouse input design
- Managing UI state transitions
- Writing clean, self-contained interactive logic

🏁 Why It Matters

The ability to manipulate the DOM interactively is at the **core of front-end development**. Whether you're animating dashboards, gamifying learning platforms, or designing UX prototypes, these exercises build the intuition needed to make interfaces feel responsive and

alive.

Every project encourages creative experimentation — so tweak the logic, change the visuals, or challenge yourself to expand them.

```
<!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 41–50</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}
  .grid-5{grid-template-columns:repeat(5,1fr)}
  .grid-6{grid-template-columns:repeat(6,1fr)}
  .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],input[type=range]{padding:.45rem
.6rem;border-radius:10px;border:1px solid #111;background:#0b1220;color:#e5e7eb}
  input[type=range]{width:120px}
  .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}

.dot{position:absolute;width:18px;height:18px;border-radius:50%;display:grid;place-items:center
;border:1px solid #000;background:#ef4444;color:#06121f;font-weight:700}

.cursor{position:absolute;width:16px;height:16px;border-radius:4px;background:#60a5fa;border:
1px solid #000;pointer-events:none}

.paddle{position:absolute;bottom:8px;left:110px;width:60px;height:10px;border-radius:8px;backg
round:#60a5fa;border:1px solid #000}

.ball{position:absolute;width:12px;height:12px;border-radius:50%;background:#f59e0b;border:1
px solid #000}
  .rgb-box{width:120px;height:80px;border-radius:10px;border:1px solid #000;background:#000}
  .tile{display:grid;place-items:center;border:1px solid
#000;border-radius:8px;background:#0b1220;height:64px;font-weight:700;font-size:1.1rem;curs
or:pointer}
  .wall{background:#0b0f1a}

.hero{position:absolute;width:18px;height:18px;border-radius:4px;background:#22c55e;border:1
px solid #000}

.goal{position:absolute;width:18px;height:18px;border-radius:4px;background:#f59e0b;border:1
px dashed #000}
  .letters{display:flex;gap:6px;flex-wrap:wrap}
  .letter{padding:.35rem .45rem;border:1px solid
#000;border-radius:8px;background:#0b1220;cursor:grab;user-select:none}
</style>
</head>
<body>
<header>
  <h1>JavaScript DOM Games — Exercises 41–50</h1>
  <p class="small">Vanilla JS, no build tools. Click “Run All Tests” at the bottom for quick
validations.</p>
</header>

<div class="wrap">

  <!-- 41) Grid Painter -->
  <section class="card" id="ex41">
    <header><h2>41) Grid Painter</h2></header>
    <div class="body">
      <div class="meta">Click cells to paint; right-click to erase; Shift+drag paints a line.</div>
      <div class="row"><button id="ex41-clear">Clear</button> Painted: <b
```

```
id="ex41-count">0</b></div>
      <div id="ex41-grid" class="grid grid-6" style="margin-top:6px"></div>
      <p class="note"><strong>Outcomes:</strong> event delegation, modifier keys, counting
state.</p>
    </div>
  </section>

  <!-- 42) Whack-a-Cell -->
  <section class="card" id="ex42">
    <header><h2>42) Whack-a-Cell</h2></header>
    <div class="body">
      <div class="meta">Click the highlighted cells quickly for 15s.</div>
      <div class="row"><button id="ex42-start">Start</button> ⏱ <b id="ex42-time">15</b>s |
Score: <b id="ex42-score">0</b></div>
      <div id="ex42-grid" class="grid grid-5" style="margin-top:6px"></div>
      <p class="note"><strong>Outcomes:</strong> timed rounds, selection state, scoring.</p>
    </div>
  </section>

  <!-- 43) Rhythm Click -->
  <section class="card" id="ex43">
    <header><h2>43) Rhythm Click</h2></header>
    <div class="body">
      <div class="meta">Click when the bar reaches the marker.</div>
      <div class="arena" id="ex43-arena">
        <div id="ex43-bar"
style="position:absolute;left:0;top:0;width:6px;height:100%;background:#60a5fa"></div>
        <div id="ex43-mark"
style="position:absolute;left:60%;top:0;width:3px;height:100%;background:#ef4444"></div>
      </div>
      <div class="row" style="margin-top:6px"><button id="ex43-start">Start</button> Hits: <b
id="ex43-hit">0</b> / Miss: <b id="ex43-miss">0</b></div>
      <p class="note"><strong>Outcomes:</strong> rAF loop, hit windows, timing tolerance.</p>
    </div>
  </section>

  <!-- 44) Path Drawer -->
  <section class="card" id="ex44">
    <header><h2>44) Path Drawer</h2></header>
    <div class="body">
      <div class="meta">Drag inside the arena to draw a path; total length updates live.</div>
      <div class="arena" id="ex44-arena"></div>
      <div style="margin-top:6px">Segments: <b id="ex44-seg">0</b> | Length: <b
id="ex44-len">0</b> px</div>
      <div class="row" style="margin-top:6px"><button id="ex44-clear">Clear</button></div>
      <p class="note"><strong>Outcomes:</strong> pointer capture, geometry, DOM
cleanup.</p>
    </div>
```

```
  </section>

  <!-- 45) Anagram Builder (Drag letters) -->
  <section class="card" id="ex45">
    <header><h2>45) Anagram Builder (Drag Letters)</h2></header>
    <div class="body">
      <div class="meta">Drag letters to spell the target word.</div>
      <div>Target: <b id="ex45-target">—</b></div>
      <div id="ex45-bank" class="letters" style="margin-top:6px"></div>
      <div class="row" style="margin-top:6px">
        <div id="ex45-drop" class="letters" style="min-height:44px;border:1px dashed
#333;border-radius:10px;padding:6px 8px"></div>
      </div>
      <div style="margin-top:6px">Result: <b id="ex45-res">—</b> <button
id="ex45-new">New</button></div>
      <p class="note"><strong>Outcomes:</strong> drag & drop, ordering, string compare.</p>
    </div>
  </section>

  <!-- 46) Paddle Wall -->
  <section class="card" id="ex46">
    <header><h2>46) Paddle Wall</h2></header>
    <div class="body">
      <div class="meta">Move paddle with mouse to keep the ball from falling.</div>
      <div class="row"><button id="ex46-start">Start</button> ⏱ <b id="ex46-time">20</b>s |
Bounces: <b id="ex46-score">0</b></div>
      <div class="arena" id="ex46-arena">
        <div class="paddle" id="ex46-pad"></div>
        <div class="ball" id="ex46-ball"></div>
      </div>
      <p class="note"><strong>Outcomes:</strong> collision (AABB), velocities, mouse
controls.</p>
    </div>
  </section>

  <!-- 47) RGB Mixer Match -->
  <section class="card" id="ex47">
    <header><h2>47) RGB Mixer Match</h2></header>
    <div class="body">
      <div class="meta">Use sliders to match the target color (±15 tolerance).</div>
      <div class="row">Target: <span id="ex47-txt" style="font-weight:700">rgb(?, ?,
?)</span></div>
      <div class="row" style="margin-top:6px">
        R <input id="r47" type="range" min="0" max="255" value="0">
        G <input id="g47" type="range" min="0" max="255" value="0">
        B <input id="b47" type="range" min="0" max="255" value="0">
        <div class="rgb-box" id="ex47-box"></div>
      </div>
```

```
      <div style="margin-top:6px" id="ex47-msg">—</div>
      <p class="note"><strong>Outcomes:</strong> live styles, tolerance checks, UI
feedback.</p>
    </div>
  </section>

  <!-- 48) Micro Mines (5×5) -->
  <section class="card" id="ex48">
    <header><h2>48) Micro Mines (5×5)</h2></header>
    <div class="body">
      <div class="meta">Reveal all safe cells; numbers show adjacent mines.</div>
      <div class="row"><button id="ex48-new">New</button> Safe left: <b
id="ex48-left">—</b></div>
      <div id="ex48-grid" class="grid grid-5" style="margin-top:6px"></div>
      <div style="margin-top:6px" id="ex48-msg">—</div>
      <p class="note"><strong>Outcomes:</strong> neighborhood counts, reveal logic, win/lose
states.</p>
    </div>
  </section>

  <!-- 49) Memory Typing -->
  <section class="card" id="ex49">
    <header><h2>49) Memory Typing</h2></header>
    <div class="body">
      <div class="meta">Watch the letters, then type them back in order.</div>
      <div class="row"><button id="ex49-start">Start</button> Level: <b
id="ex49-lvl">0</b></div>
      <div class="row" style="margin-top:6px">
        <span id="ex49-show" style="font-weight:700">—</span>
        <input id="ex49-in" type="text" placeholder="type sequence" disabled>
        <button id="ex49-ok" disabled>OK</button>
      </div>
      <div style="margin-top:6px" id="ex49-msg">—</div>
      <p class="note"><strong>Outcomes:</strong> async flashing, input gating, correctness
checks.</p>
    </div>
  </section>

  <!-- 50) Arrow Maze -->
  <section class="card" id="ex50">
    <header><h2>50) Arrow Maze</h2></header>
    <div class="body">
      <div class="meta">Use arrow keys to reach the goal without touching walls.</div>
      <div class="arena" id="ex50-arena"></div>
      <div style="margin-top:6px" id="ex50-status">Press any arrow key to begin</div>
      <p class="note"><strong>Outcomes:</strong> keyboard motion, collision with walls, win
detection.</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));

/* 41) Grid Painter */
(function(){
  const grid=$('#ex41-grid'), clear=$('#ex41-clear'), countEl=$('#ex41-count');
  const W=6,H=5; let painting=false, painted=0;
  function build(){
    grid.style.gridTemplateColumns='repeat(6,1fr)';
    grid.innerHTML=''; painted=0; countEl.textContent=painted;
    for(let i=0;i<W*H;i++){
      const d=document.createElement('div'); d.className='cell';
      d.addEventListener('mousedown', e=>{ if(e.button===0){ painting=true; paint(d); }});
      d.addEventListener('mouseenter', e=>{ if(painting && e.buttons===1 && e.shiftKey) paint(d);
});
      d.addEventListener('contextmenu', e=>{ e.preventDefault(); erase(d); });
      grid.appendChild(d);
    }
    document.addEventListener('mouseup', ()=>painting=false);
  }
  function paint(d){ if(d.dataset.on==='1') return; d.dataset.on='1'; d.style.background='#22c55e';
painted++; countEl.textContent=painted; }
  function erase(d){ if(d.dataset.on!=='1') return; d.dataset.on='0'; d.style.background='#0b1220';
painted--; countEl.textContent=painted; }
  clear.addEventListener('click', build);
  build();
  window._ex41={grid,clear};
})();

/* 42) Whack-a-Cell */
(function(){
  const grid=$('#ex42-grid'), start=$('#ex42-start'), tEl=$('#ex42-time'), scoreEl=$('#ex42-score');
```

```
  let cells=[], active=-1, t=15, timer=0, rounder=0, score=0, playing=false;
  for(let i=0;i<25;i++){ const d=document.createElement('div'); d.className='cell';
grid.appendChild(d); cells.push(d); }
  function setActive(i){ cells.forEach(c=>c.style.background='#0b1220'); if(i>=0)
cells[i].style.background='#ef4444'; active=i; }
  grid.addEventListener('click', e=>{
    if(!playing || !e.target.classList.contains('cell')) return;
    if(cells.indexOf(e.target)===active){ score++; scoreEl.textContent=score; setActive(-1); }
  });
  start.addEventListener('click', ()=>{
    if(playing) return; playing=true; score=0; scoreEl.textContent=score; t=15; tEl.textContent=t;
    clearInterval(timer); clearInterval(rounder);
    rounder=setInterval(()=>{ setActive(rand(25)); },700);
    timer=setInterval(()=>{ t--; tEl.textContent=t; if(t<=0){ clearInterval(timer);
clearInterval(rounder); playing=false; setActive(-1);} },1000);
  });
  window._ex42={start,grid};
})();

/* 43) Rhythm Click */
(function(){
  const arena=$('#ex43-arena'), bar=$('#ex43-bar'), mark=$('#ex43-mark'), start=$('#ex43-start'),
hitEl=$('#ex43-hit'), missEl=$('#ex43-miss');
  let running=false, last=0, x=0, vx=140; // px/s
  function loop(ts){
    if(!running) return;
    if(!last) last=ts; const dt=(ts-last)/1000; last=ts;
    const w=arena.clientWidth;
    x+=vx*dt; if(x>w-6 || x<0){ vx*=-1; x=clamp(x,0,w-6); }
    bar.style.left=x+'px';
    requestAnimationFrame(loop);
  }
  arena.addEventListener('click', ()=>{
    if(!running) return;
    const w=arena.clientWidth; const mx = 0.60*w; const cx=x+3;
    const hit = Math.abs(cx-mx) <= 14;
    if(hit){ hitEl.textContent=+hitEl.textContent+1; }
    else { missEl.textContent=+missEl.textContent+1; }
  });
  start.addEventListener('click', ()=>{
    if(running) return; running=true; last=0; x=0; hitEl.textContent='0'; missEl.textContent='0';
    requestAnimationFrame(loop);
    setTimeout(()=>running=false, 15000);
  });
  window._ex43={start,arena};
})();

/* 44) Path Drawer */
```

```
(function(){
  const arena=$('#ex44-arena'), segEl=$('#ex44-seg'), lenEl=$('#ex44-len'),
clear=$('#ex44-clear');
  let drawing=false, last=null, seg=0, len=0;
  function addDot(x,y){
    const d=document.createElement('div'); d.className='dot'; d.style.left=(x-4)+'px';
d.style.top=(y-4)+'px'; d.style.width='8px'; d.style.height='8px'; d.style.borderRadius='50%';
d.style.background='#60a5fa';
    arena.appendChild(d);
  }
  arena.addEventListener('pointerdown', e=>{
    drawing=true; arena.setPointerCapture(e.pointerId);
    const r=arena.getBoundingClientRect(); last={x:e.clientX-r.left,y:e.clientY-r.top};
addDot(last.x,last.y);
  });
  arena.addEventListener('pointermove', e=>{
    if(!drawing) return; const r=arena.getBoundingClientRect(); const x=e.clientX-r.left,
y=e.clientY-r.top;
    addDot(x,y); if(last){ seg++; const dx=x-last.x, dy=y-last.y; len+=Math.hypot(dx,dy); last={x,y};
segEl.textContent=seg; lenEl.textContent=Math.round(len); }
  });
  arena.addEventListener('pointerup', ()=>{ drawing=false; last=null; });
  clear.addEventListener('click', ()=>{ arena.innerHTML=''; seg=0; len=0; segEl.textContent='0';
lenEl.textContent='0'; });
  window._ex44={arena,clear};
})();

/* 45) Anagram Builder */
(function(){
  const words=['script','layout','render','object','binary','thread','driver','matrix','socket','bundle'];
  const targetEl=$('#ex45-target'), bank=$('#ex45-bank'), drop=$('#ex45-drop'),
res=$('#ex45-res'), btn=$('#ex45-new');
  let target='';
  function build(){
    target = words[rand(words.length)];
    targetEl.textContent = target.toUpperCase();
    res.textContent='—'; bank.innerHTML=''; drop.innerHTML='';
    const letters = target.split('').sort(()=>Math.random()-0.5);
    letters.forEach(ch=>{
      const d=document.createElement('div'); d.className='letter';
d.textContent=ch.toUpperCase(); d.draggable=true;
      d.addEventListener('dragstart', e=>{ e.dataTransfer.setData('text/plain', ch);
e.dataTransfer.setData('text/id', Date.now()+Math.random()); d.dataset.drag='1'; });
      d.addEventListener('dragend', ()=>{ delete d.dataset.drag; });
      d.addEventListener('dblclick', ()=>{ if(d.parentElement===drop) bank.appendChild(d); else
drop.appendChild(d); check(); });
      bank.appendChild(d);
    });
```

```
  }
  function onDragOver(e){ e.preventDefault(); }
  function onDrop(container,e){
    e.preventDefault();
    const el = bank.querySelector('.letter[data-drag="1"], .letters .letter[data-drag="1"]');
    if(el) container.appendChild(el);
    check();
  }
  function check(){
    const guess =
Array.from(drop.querySelectorAll('.letter')).map(n=>n.textContent.toLowerCase()).join('');
    if(!guess) { res.textContent='—'; return; }
    res.textContent = (guess===target) ? '✅ Correct!' : '…';
  }
  drop.addEventListener('dragover', onDragOver); drop.addEventListener('drop',
e=>onDrop(drop,e));
  bank.addEventListener('dragover', onDragOver); bank.addEventListener('drop',
e=>onDrop(bank,e));
  btn.addEventListener('click', build); build();
  window._ex45={bank,drop,btn};
})();

/* 46) Paddle Wall */
(function(){
  const arena=$('#ex46-arena'), pad=$('#ex46-pad'), ball=$('#ex46-ball'), start=$('#ex46-start'),
tEl=$('#ex46-time'), scoreEl=$('#ex46-score');
  let running=false, vx=120, vy=-140, x=130, y=100, t=20, timer=0, last=0, score=0;
  function place(){ ball.style.left=x+'px'; ball.style.top=y+'px'; }
  arena.addEventListener('mousemove', e=>{
    const r=arena.getBoundingClientRect(); const mx=e.clientX-r.left;
    pad.style.left = clamp(mx-30, 0, arena.clientWidth-60)+'px';
  });
  function loop(ts){
    if(!running) return;
    if(!last) last=ts; const dt=(ts-last)/1000; last=ts;
    x += vx*dt; y += vy*dt;
    if(x<0 || x>arena.clientWidth-12){ vx*=-1; x=clamp(x,0,arena.clientWidth-12); }
    if(y<0){ vy*=-1; y=0; }
    // paddle collision
    const pr=pad.getBoundingClientRect(), ar=arena.getBoundingClientRect();
    const bx=ar.left+x, by=ar.top+y;
    if(by+12 >= pr.top && by+12 <= pr.top+14 && bx+12 >= pr.left && bx <= pr.right){
      vy = -Math.abs(vy); y = pr.top - ar.top - 12; score++; scoreEl.textContent=score;
    }
    // floor lose
    if(y>arena.clientHeight-12){ running=false; }
    place(); requestAnimationFrame(loop);
  }
```

```
  start.addEventListener('click', ()=>{
    if(running) return; running=true; x=120; y=90; vx=120*(Math.random()>.5?1:-1); vy=-140;
place(); score=0; scoreEl.textContent=score; t=20; tEl.textContent=t; last=0;
    clearInterval(timer); timer=setInterval(()=>{ t--; tEl.textContent=t; if(t<=0){ clearInterval(timer);
running=false; } },1000);
    requestAnimationFrame(loop);
  });
  place();
  window._ex46={start,arena};
})();

/* 47) RGB Mixer Match */
(function(){
  const r=$('#r47'), g=$('#g47'), b=$('#b47'), box=$('#ex47-box'), txt=$('#ex47-txt'),
msg=$('#ex47-msg');
  let tr=0,tg=0,tb=0;
  function newTarget(){
    tr=40+rand(200); tg=40+rand(200); tb=40+rand(200);
    txt.textContent=`rgb(${tr}, ${tg}, ${tb})`; msg.textContent='—';
  }
  function update(){
    const cr=+r.value, cg=+g.value, cb=+b.value;
    box.style.background=`rgb(${cr},${cg},${cb})`;
    const ok = Math.abs(cr-tr)<=15 && Math.abs(cg-tg)<=15 && Math.abs(cb-tb)<=15;
    msg.textContent = ok ? '🎯 Close enough!' : 'Adjust sliders…';
  }
  [r,g,b].forEach(inp=>inp.addEventListener('input', update));
  newTarget(); update();
  window._ex47={r,g,b,box};
})();

/* 48) Micro Mines */
(function(){
  const grid=$('#ex48-grid'), btn=$('#ex48-new'), leftEl=$('#ex48-left'), msg=$('#ex48-msg');
  const W=5,H=5,M=5; let mines=new Set(), left=0, over=false;
  function idx(x,y){ return y*W+x; }
  function inb(x,y){ return x>=0&&x<W&&y>=0&&y<H; }
  function neighbors(x,y){ const res=[]; for(let dx=-1;dx<=1;dx++)for(let dy=-1;dy<=1;dy++){
if(dx||dy){ const nx=x+dx,ny=y+dy; if(inb(nx,ny)) res.push([nx,ny]); } } return res; }
  function build(){
    over=false; msg.textContent='—'; grid.innerHTML=''; mines=new Set(); left=W*H-M;
leftEl.textContent=left;
    while(mines.size<M){ mines.add(rand(W*H)); }
    for(let y=0;y<H;y++) for(let x=0;x<W;x++){
      const d=document.createElement('div'); d.className='cell'; d.dataset.x=x; d.dataset.y=y;
d.textContent='?';
      d.addEventListener('click', ()=>reveal(d));
      grid.appendChild(d);
```

```
  }
 }
 function reveal(d){
  if(over||d.dataset.done==='1') return;
  const x=+d.dataset.x,y=+d.dataset.y, id=idx(x,y);
  d.dataset.done='1'; d.style.background='#111a2a';
  if(mines.has(id)){ d.textContent='💥'; msg.textContent='Boom!'; over=true; return; }
  left--; leftEl.textContent=left;
  const n=neighbors(x,y).filter(([nx,ny])=>mines.has(idx(nx,ny))).length;
  d.textContent = n? String(n) : '';
  if(n===0){ neighbors(x,y).forEach(([nx,ny])=>{ const c=[...grid.children][idx(nx,ny)];
if(c.dataset.done!=='1') reveal(c); }); }
  if(left===0){ msg.textContent='✅ You cleared it!'; over=true; }
 }
 btn.addEventListener('click', build); build();
 window._ex48={btn,grid};
})();

/* 49) Memory Typing */
(function(){
 const start=$('#ex49-start'), lvlEl=$('#ex49-lvl'), show=$('#ex49-show'), inp=$('#ex49-in'),
ok=$('#ex49-ok'), msg=$('#ex49-msg');
 const ABC='ABCDEFGHJKLMNPQRSTUVWXYZ'; let seq='', idx=0, listening=false;
 function flash(s){ return new Promise(async res=>{ show.textContent=''; for(const ch of s){
show.textContent=ch; await wait(500); show.textContent=''; await wait(150); } res(); }); }
 function next(){ seq += ABC[rand(ABC.length)]; lvlEl.textContent=seq.length;
flash(seq).then(()=>{ listening=true; inp.disabled=false; ok.disabled=false; inp.value='';
inp.focus(); }); }
 ok.addEventListener('click', ()=>{ if(!listening) return; const v=inp.value.trim().toUpperCase();
if(v===seq){ msg.textContent='✅'; listening=false; inp.disabled=true; ok.disabled=true;
setTimeout(next,400); } else { msg.textContent='✖'; listening=false; seq=''; lvlEl.textContent='0';
inp.disabled=true; ok.disabled=true; } });
 start.addEventListener('click', ()=>{ seq=''; lvlEl.textContent='0'; msg.textContent='—'; next(); });
 window._ex49={start,ok,inp};
})();

/* 50) Arrow Maze */
(function(){
 const arena=$('#ex50-arena'), status=$('#ex50-status');
 const cols=10, rows=6, cell=24;
 arena.style.width=(cols*cell+8)+'px';
 arena.style.height=(rows*cell+8)+'px';
 // Build walls
 const walls=new Set(['3,0','3,1','3,2','6,2','6,3','6,4','1,4','2,4','3,4','4,4','7,0','7,1']);
 walls.forEach(key=>{
  const [x,y]=key.split(',').map(Number);
  const w=document.createElement('div'); w.className='wall'; w.style.position='absolute';
  w.style.left=(x*cell+4)+'px'; w.style.top=(y*cell+4)+'px';
```

```
    w.style.width=(cell-4)+'px'; w.style.height=(cell-4)+'px'; w.style.borderRadius='6px';
w.style.border='1px solid #000';
    arena.appendChild(w);
  });
  const hero=document.createElement('div'); hero.className='hero'; arena.appendChild(hero);
  const goal=document.createElement('div'); goal.className='goal'; arena.appendChild(goal);
  let hx=0, hy=0, gx=9, gy=5;
  function place(){ hero.style.left=(hx*cell+6)+'px'; hero.style.top=(hy*cell+6)+'px';
goal.style.left=(gx*cell+6)+'px'; goal.style.top=(gy*cell+6)+'px'; }
  function collide(nx,ny){ return nx<0||ny<0||nx>=cols||ny>=rows||walls.has(nx+','+ny); }
  window.addEventListener('keydown', e=>{
    const dir = {ArrowLeft:[-1,0],ArrowRight:[1,0],ArrowUp:[0,-1],ArrowDown:[0,1]}[e.key];
    if(!dir) return;
    const nx=hx+dir[0], ny=hy+dir[1];
    if(!collide(nx,ny)){ hx=nx; hy=ny; place(); status.textContent='Moving…'; }
    else { status.textContent='Blocked!'; }
    if(hx===gx && hy===gy){ status.textContent='🎉 You reached the goal!'; }
  });
  place();
  window._ex50={arena};
})();

/* ------------------------
   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 t41(){ const {grid,clear}=window._ex41;
grid.querySelector('.cell')?.dispatchEvent(new
MouseEvent('mousedown',{bubbles:true,button:0})); clear.click(); ok('ex41 wired'); }
  async function t42(){ const {start}=window._ex42; start.click(); ok('ex42 start'); }
  async function t43(){ const {start}=window._ex43; start.click(); ok('ex43 start'); }
  async function t44(){ const {arena,clear}=window._ex44; arena.dispatchEvent(new
PointerEvent('pointerdown',{bubbles:true})); arena.dispatchEvent(new
PointerEvent('pointerup',{bubbles:true})); clear.click(); ok('ex44 pointer'); }
  async function t45(){ const {btn}=window._ex45; btn.click(); ok('ex45 new'); }
  async function t46(){ const {start}=window._ex46; start.click(); ok('ex46 start'); }
  async function t47(){ const {r}=window._ex47; r.value='128'; r.dispatchEvent(new
Event('input',{bubbles:true})); ok('ex47 slider'); }
  async function t48(){ const {btn}=window._ex48; btn.click(); ok('ex48 new'); }
  async function t49(){ const {start}=window._ex49; start.click(); ok('ex49 start'); }
  async function t50(){ const {arena}=window._ex50; ok('ex50 arena exists: '+!!arena); }

  btn.addEventListener('click', async ()=>{
```

```
    logs=[];
    const tests=[t41,t42,t43,t44,t45,t46,t47,t48,t49,t50];
    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>
```