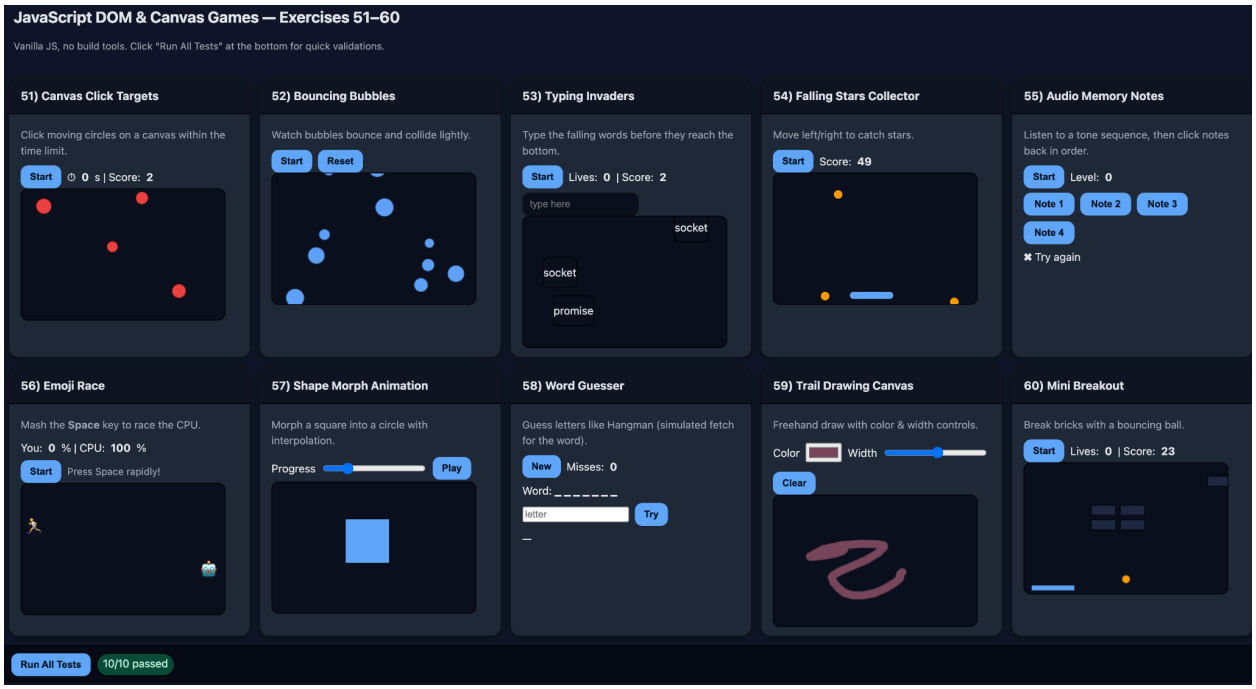
🎮 JavaScript DOM & Canvas Games — Exercises 51–60



Where Code Meets Creativity

The DOM is your playground, and JavaScript is the tool that brings it to life.
In this latest collection of **interactive exercises (51–60)**, you'll take your front-end skills to the next level — blending DOM manipulation, Canvas drawing, motion physics, and even sound.

Each mini-project is designed to teach a distinct concept while keeping things fun, visual, and practical. No frameworks, no boilerplate — just pure JavaScript powering creativity in the browser.

✨ Overview

This set focuses on **DOM and Canvas integration**. You'll move beyond static interactions and start working with **animation loops, physics, and audio events**, exploring how code transforms into dynamic visual behavior.

Every project can be opened directly in a browser — just download the ZIP, open `index.html`, and start experimenting.

🕹️ Exercise Highlights

51 — Canvas Click Targets

You'll build a fast-paced clicking challenge where red circles bounce around a canvas. The player must click them before time runs out. This project introduces the Canvas 2D API, animation frames, and collision reflection logic — all cornerstones of modern browser-based games.

52 — Bouncing Bubbles

A visual dance of motion and collisions. Here you'll simulate bubbles that move and lightly bounce off each other and the canvas edges. It's an elegant introduction to velocity, direction, and elastic collisions — the building blocks of interactive motion.

53 — Typing Invaders

Words fall from the top of the screen, and your task is to type them before they reach the bottom. It's a pure test of typing speed and focus. This exercise sharpens your understanding of DOM element creation, timed updates, and input-driven gameplay.

54 — Falling Stars Collector

Move a paddle left and right to catch falling stars. You'll use `requestAnimationFrame` for smooth animation and `getBoundingClientRect()` for collision detection. This game demonstrates the core principles behind 2D platformers and arcade mechanics.

55 — Audio Memory Notes

Here, sound becomes your challenge. The browser plays a sequence of tones that you must reproduce by clicking colored buttons. It uses the Web Audio API to generate tones dynamically and tests both your listening skills and async JavaScript handling.

56 — Emoji Race

It's human versus machine! Mash the spacebar to move your runner emoji faster than the CPU. This simple race game highlights event-driven design, keyboard controls, and loop-based AI behavior.

57 — Shape Morph Animation

Witness smooth transitions between shapes as a square morphs into a circle. Using interpolation and CSS transforms, you'll see how incremental changes over time create fluid

motion — a technique that underpins many UI animations.

58 — Word Guesser

A hangman-inspired guessing game built entirely in the browser. You'll simulate fetching a random word and then process each guessed letter. It's a hands-on way to learn conditional logic, state management, and asynchronous operations.

59 — Trail Drawing Canvas

This is your digital sketchpad. As you move your mouse or stylus, colorful lines follow your path. You'll explore pointer events, drawing APIs, and input customization — great practice for building paint or annotation tools.

60 — Mini Breakout

A tribute to the classic arcade game. Break bricks with a bouncing ball and a movable paddle, all coded with physics-like logic. You'll combine geometry, animation, collision detection, and scorekeeping into a complete mini-game that's surprisingly powerful for a few lines of JavaScript.

🧠 What You'll Learn

Each of these exercises builds practical front-end development instincts.
You'll understand not just *what* the code does, but *why* it works — a mindset that separates a casual coder from a confident developer.

You'll learn to:

- Use the **Canvas 2D context** for drawing, movement, and collisions.
- Manage **game loops** with `requestAnimationFrame` for smooth visuals.
- Handle **keyboard, mouse, and pointer events** for interactive input.
- Use **AudioContext** for generating tones and creating responsive feedback.
- Apply **state machines and modular logic** to control game flow.
- Balance **DOM and Canvas layers** effectively for hybrid interfaces.

🧩 The Experience

What makes this set special is its variety. You'll move fluidly between visual, auditory, and interactive challenges. One exercise has you drawing with color and precision; the next tests your reflexes or rhythm.

And with the built-in **“Run All Tests”** button, you can verify event bindings and logic instantly — making this collection ideal for classrooms, bootcamps, or self-paced learning.

🚀 Why It Matters

Animation, interaction, and sound are no longer extras — they’re part of what makes modern interfaces feel *alive*. These exercises prepare you to build more than games; they equip you to design responsive, intuitive experiences.

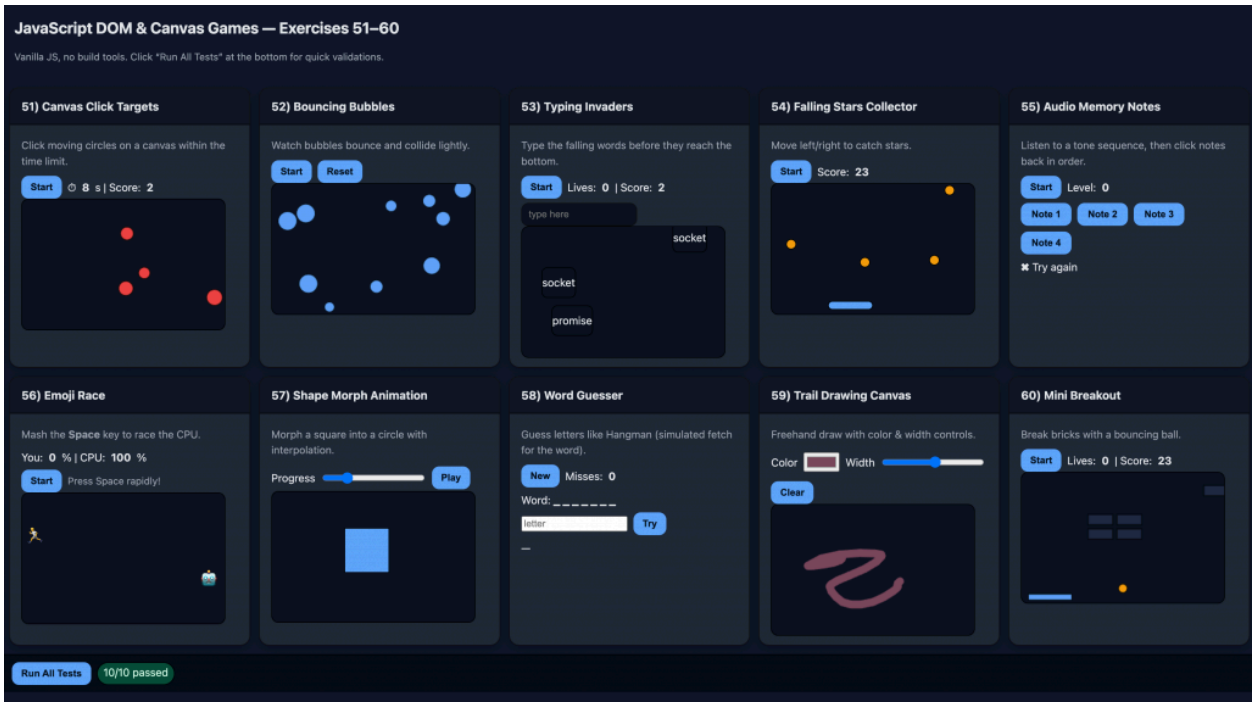
Whether you’re working on dashboards, web apps, or learning platforms, mastering these patterns means mastering how humans interact with your code.

🏁 Next Steps

Try tweaking each game.
Change colors, add more elements, or make the logic harder. Turn the Typing Invaders into a full vocabulary trainer, or add new levels to Breakout.

That’s where the real growth happens — when you make the code your own.

Stay tuned — and keep coding with creativity.



🧩 JavaScript DOM Games — Exercises 51–60

#	Game	Core Concept
51	**Canvas Click Targets**	`CanvasRenderingContext2D`, hit detection, and drawing APIs
52	**Bouncing Bubbles**	`requestAnimationFrame`, collision physics, and color dynamics
53	**Typing Invaders**	Keyboard input, timing, and object removal
54	**Falling Stars Collector**	Animation with score tracking and collision bounding boxes
55	**Audio Memory Notes**	Web Audio API tones, sequencing, and async memory challenge
56	**Emoji Race**	Keypress vs. CPU speed simulation
57	**Shape Morph Animation**	DOM → Canvas hybrid animation using interpolation
58	**AI Word Guesser**	Random API-style word guessing using `fetch` simulation
59	**Trail Drawing Canvas**	Freehand drawing with mouse tracking and color controls

60 **Mini Breakout** Ball-paddle-brick collision with scoring and lives system

---

💡 Learning Outcomes

- Use **Canvas 2D API** for dynamic graphics.
- Combine **DOM and canvas layers** effectively.
- Implement **basic physics** and motion loops.
- Practice **keyboard, pointer, and audio events**.
- Structure small games with **state and modular functions**.

```
<!doctype html>
<html lang="en">
<head>
<meta charset="utf-8"/>
<meta name="viewport" content="width=device-width,initial-scale=1"/>
<title>JavaScript DOM & Canvas Games — Exercises 51–60</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:280px;height:180px;border:1px solid
#000;border-radius:10px;background:#0b1220;overflow:hidden}
  .grid{display:grid;gap:6px}
  .grid-5{grid-template-columns:repeat(5,1fr)}
```

```
  .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}
  canvas{background:#0b1220;border-radius:10px;border:1px solid #000;display:block}
  .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}

.paddle{position:absolute;bottom:6px;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}
</style>
</head>
<body>
<header>
  <h1>JavaScript DOM & Canvas Games — Exercises 51–60</h1>
  <p class="small">Vanilla JS, no build tools. Click “Run All Tests” at the bottom for quick
validations.</p>
</header>

<div class="wrap">
  <section class="card" id="ex51"><header><h2>51) Canvas Click Targets</h2></header><div
class="body"><div class="meta">Click moving circles on a canvas within the time
limit.</div><div class="row"><button id="ex51-start">Start</button> ⏱ <b
id="ex51-time">15</b>s | Score: <b id="ex51-score">0</b></div><canvas id="ex51-c"
width="280" height="180"></canvas></div></section>

  <section class="card" id="ex52"><header><h2>52) Bouncing Bubbles</h2></header><div
class="body"><div class="meta">Watch bubbles bounce and collide lightly.</div><div
class="row"><button id="ex52-start">Start</button> <button
id="ex52-reset">Reset</button></div><canvas id="ex52-c" width="280"
height="180"></canvas></div></section>

  <section class="card" id="ex53"><header><h2>53) Typing Invaders</h2></header><div
class="body"><div class="meta">Type the falling words before they reach the bottom.</div><div
class="row"><button id="ex53-start">Start</button> Lives: <b id="ex53-lives">3</b> | Score: <b
id="ex53-score">0</b></div><div class="row" style="margin-top:6px"><input id="ex53-in"
type="text" placeholder="type here" disabled></div><div class="arena"
id="ex53-a"></div></div></section>

  <section class="card" id="ex54"><header><h2>54) Falling Stars Collector</h2></header><div
```

```
class="body"><div class="meta">Move left/right to catch stars.</div><div class="row"><button id="ex54-start">Start</button> Score: <b id="ex54-score">0</b></div><div class="arena" id="ex54-a"></div></div></section>

  <section class="card" id="ex55"><header><h2>55) Audio Memory Notes</h2></header><div class="body"><div class="meta">Listen to a tone sequence, then click notes back in order.</div><div class="row"><button id="ex55-start">Start</button> Level: <b id="ex55-level">0</b></div><div class="row" id="ex55-bank" style="margin-top:6px"></div><div id="ex55-msg" style="margin-top:6px">—</div></div></section>

  <section class="card" id="ex56"><header><h2>56) Emoji Race</h2></header><div class="body"><div class="meta">Mash the <b>Space</b> key to race the CPU.</div><div class="row">You: <b id="ex56-you">0</b>% | CPU: <b id="ex56-cpu">0</b>%</div><div class="row" style="margin-top:6px"><button id="ex56-start">Start</button> <span class="small">Press Space rapidly!</span></div><div class="arena" id="ex56-a"></div></div></section>

  <section class="card" id="ex57"><header><h2>57) Shape Morph Animation</h2></header><div class="body"><div class="meta">Morph a square into a circle with interpolation.</div><div class="row">Progress <input id="ex57-r" type="range" min="0" max="100" value="0"> <button id="ex57-play">Play</button></div><div class="arena" id="ex57-a"></div></div></section>

  <section class="card" id="ex58"><header><h2>58) Word Guesser</h2></header><div class="body"><div class="meta">Guess letters like Hangman (simulated fetch for the word).</div><div class="row"><button id="ex58-new">New</button> Misses: <b id="ex58-miss">0</b></div><div style="margin-top:6px">Word: <b id="ex58-word">—</b></div><div class="row" style="margin-top:6px"><input id="ex58-in" maxlength="1" placeholder="letter"><button id="ex58-try">Try</button></div><div id="ex58-msg" style="margin-top:6px">—</div></div></section>

  <section class="card" id="ex59"><header><h2>59) Trail Drawing Canvas</h2></header><div class="body"><div class="meta">Freehand draw with color & width controls.</div><div class="row">Color <input id="ex59-color" type="color" value="#60a5fa"> Width <input id="ex59-w" type="range" min="1" max="20" value="4"> <button id="ex59-clear">Clear</button></div><canvas id="ex59-c" width="280" height="180"></canvas></div></section>

  <section class="card" id="ex60"><header><h2>60) Mini Breakout</h2></header><div class="body"><div class="meta">Break bricks with a bouncing ball.</div><div class="row"><button id="ex60-start">Start</button> Lives: <b id="ex60-lives">3</b> | Score: <b id="ex60-score">0</b></div><canvas id="ex60-c" width="280" height="180"></canvas></div></section>
</div>

<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>
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));

/* 51 Canvas Click Targets */
(function(){
  const c=$('#ex51-c'), ctx=c.getContext('2d'), start=$('#ex51-start'), tEl=$('#ex51-time'),
scoreEl=$('#ex51-score');
  let t=15, targets=[], running=false, timer=0;
  function spawn(){
    const r=8+rand(10), x=r+rand(c.width-2*r), y=r+rand(c.height-2*r);
    const vx=(rand(2)?1:-1)*(40+rand(80))/60, vy=(rand(2)?1:-1)*(40+rand(80))/60;
    targets.push({x,y,r,vx,vy});
  }
  function step(){
    ctx.clearRect(0,0,c.width,c.height);
    if(Math.random()<0.05 && targets.length<6) spawn();
    for(const t of targets){
      t.x+=t.vx; t.y+=t.vy;
      if(t.x<t.r||t.x>c.width-t.r) t.vx*=-1;
      if(t.y<t.r||t.y>c.height-t.r) t.vy*=-1;
      ctx.beginPath(); ctx.fillStyle='#ef4444'; ctx.arc(t.x,t.y,t.r,0,Math.PI*2); ctx.fill();
ctx.strokeStyle='#000'; ctx.stroke();
    }
    if(running) requestAnimationFrame(step);
  }
  c.addEventListener('click', e=>{
    if(!running) return;
    const r=c.getBoundingClientRect(), mx=e.clientX-r.left, my=e.clientY-r.top;
    for(const t of targets){
      const dx=mx-t.x, dy=my-t.y;
      if(dx*dx+dy*dy<=t.r*t.r){ scoreEl.textContent=+scoreEl.textContent+1; t.x=-999; break; }
    }
  });
  start.addEventListener('click', ()=>{
    if(running) return; running=true; t=15; tEl.textContent=t; scoreEl.textContent='0';
targets.length=0; spawn(); spawn();
    clearInterval(timer); timer=setInterval(()=>{ t--; tEl.textContent=t; if(t<=0){ clearInterval(timer);
running=false; } },1000);
    requestAnimationFrame(step);
  });
  window._ex51={start,c};
```

```
})();

/* 52 Bouncing Bubbles */
(function(){
  const c=$('#ex52-c'), ctx=c.getContext('2d'), start=$('#ex52-start'), reset=$('#ex52-reset');
  let balls=[], running=false;
  function build(){
    balls=[]; for(let i=0;i<10;i++){ const r=6+rand(10), x=r+rand(c.width-2*r),
y=r+rand(c.height-2*r); balls.push({x,y,r,vx:(-1+Math.random()*2)*2,vy:(-1+Math.random()*2)*2});
}
  }
  function loop(){
    ctx.clearRect(0,0,c.width,c.height);
    for(let i=0;i<balls.length;i++){
      const b=balls[i]; b.x+=b.vx; b.y+=b.vy;
      if(b.x<b.r||b.x>c.width-b.r) b.vx*=-1;
      if(b.y<b.r||b.y>c.height-b.r) b.vy*=-1;
      for(let j=i+1;j<balls.length;j++){
        const a=balls[j], dx=a.x-b.x, dy=a.y-b.y, d=Math.hypot(dx,dy);
        if(d<b.r+a.r){ // simple elastic swap
          const t=b.vx; b.vx=a.vx; a.vx=t; const u=b.vy; b.vy=a.vy; a.vy=u;
        }
      }
      ctx.beginPath(); ctx.fillStyle='#60a5fa'; ctx.arc(b.x,b.y,b.r,0,Math.PI*2); ctx.fill();
ctx.strokeStyle='#000'; ctx.stroke();
    }
    if(running) requestAnimationFrame(loop);
  }
  start.addEventListener('click', ()=>{ if(running) return; running=true; loop(); });
  reset.addEventListener('click', ()=>{ build(); if(!running) { ctx.clearRect(0,0,c.width,c.height);
for(const b of balls){ ctx.beginPath(); ctx.arc(b.x,b.y,b.r,0,Math.PI*2); ctx.fillStyle='#60a5fa';
ctx.fill(); ctx.stroke(); } } });
  build();
  window._ex52={start,reset,c};
})();

/* 53 Typing Invaders */
(function(){
  const arena=$('#ex53-a'), start=$('#ex53-start'), inp=$('#ex53-in'), livesEl=$('#ex53-lives'),
scoreEl=$('#ex53-score');
  const
WORDS=['array','event','object','promise','module','script','render','update','binary','driver','thread'
,'socket'];
  let items=[], running=false, timer=0, spawnId=0, lives=3, score=0;
  function spawn(){
    const w=WORDS[rand(WORDS.length)], el=document.createElement('div');
    el.className='cell'; el.textContent=w; el.style.position='absolute';
el.style.left=(10+rand(arena.clientWidth-70))+'px'; el.style.top='-30px';
```

```
    arena.appendChild(el);
    items.push({word:w,y:-30,el,speed:0.5+Math.random()*0.6});
  }
  function loop(){
    for(const it of items){ it.y+=it.speed; it.el.style.top=it.y+'px'; }
    items=items.filter(it=>{
      if(it.y>arena.clientHeight-10){ it.el.remove(); lives--; livesEl.textContent=lives; return false; }
      return true;
    });
    if(running){ requestAnimationFrame(loop); if(lives<=0){ running=false; clearInterval(spawnId);
inp.disabled=true; } }
  }
  start.addEventListener('click', ()=>{
    if(running) return; running=true; lives=3; score=0; livesEl.textContent=lives;
scoreEl.textContent=score; arena.innerHTML=''; items=[];
    clearInterval(spawnId); spawnId=setInterval(spawn, 900); inp.disabled=false; inp.value='';
inp.focus(); loop();
  });
  inp.addEventListener('input', ()=>{
    const v=inp.value.trim().toLowerCase(); if(!v) return;
    const hit = items.find(it=>it.word===v);
    if(hit){ score++; scoreEl.textContent=score; hit.el.remove(); items=items.filter(x=>x!==hit);
inp.value=''; }
  });
  window._ex53={start,inp,arena};
})();

/* 54 Falling Stars Collector */
(function(){
  const arena=$('#ex54-a'), start=$('#ex54-start'), scoreEl=$('#ex54-score');
  const pad=document.createElement('div'); pad.className='paddle'; arena.appendChild(pad);
  let running=false, last=0, x=110, stars=[], score=0;
  arena.addEventListener('mousemove', e=>{ const r=arena.getBoundingClientRect();
x=clamp(e.clientX-r.left-30,0,arena.clientWidth-60); pad.style.left=x+'px'; });
  function spawn(){ const d=document.createElement('div'); d.className='ball';
d.style.left=(rand(arena.clientWidth-12))+'px'; d.style.top='-10px'; arena.appendChild(d);
stars.push({el:d,y:-10,vy:60+rand(80)}); }
  function loop(ts){
    if(!running) return;
    if(!last) last=ts; const dt=(ts-last)/1000; last=ts;
    if(Math.random()<0.03) spawn();
    for(const s of stars){ s.y+=s.vy*dt; s.el.style.top=s.y+'px'; }
    stars=stars.filter(s=>{
      const r=s.el.getBoundingClientRect(), pr=pad.getBoundingClientRect();
      if(r.top>=pr.top-8 && r.left<pr.right && r.right>pr.left && r.bottom<=pr.bottom+18){ score++;
scoreEl.textContent=score; s.el.remove(); return false; }
      if(s.y>arena.clientHeight){ s.el.remove(); return false; }
      return true;
```

```
  });
  requestAnimationFrame(loop);
 }
 start.addEventListener('click', ()=>{ if(running) return; running=true; last=0; score=0;
scoreEl.textContent=score; stars.forEach(s=>s.el.remove()); stars=[];
requestAnimationFrame(loop); });
 window._ex54={start,arena};
})();

/* 55 Audio Memory Notes */
(function(){
 const start=$('#ex55-start'), bank=$('#ex55-bank'), levelEl=$('#ex55-level'),
msg=$('#ex55-msg');
 const ctx = (window.AudioContext? new AudioContext(): null);
 const freqs=[261.63,293.66,329.63,349.23]; // C D E F
 const buttons=[]; let seq=[], idx=0, listening=false;
 function tone(f,ms=400){ if(!ctx) return wait(ms); const o=ctx.createOscillator(); const
g=ctx.createGain(); o.type='sine'; o.frequency.value=f; o.connect(g); g.connect(ctx.destination);
g.gain.value=0.001; o.start(); g.gain.exponentialRampToValueAtTime(0.2,
ctx.currentTime+0.02); g.gain.exponentialRampToValueAtTime(0.001,
ctx.currentTime+ms/1000); setTimeout(()=>{ o.stop(); }, ms+10); return wait(ms+80); }
 function flash(btn){ btn.style.transform='scale(1.08)'; setTimeout(()=>btn.style.transform='',
180); }
 function playSeq(){ listening=false; (async()=>{ for(const i of seq){ flash(buttons[i]); await
tone(freqs[i], 380); await wait(80);} idx=0; listening=true; })(); }
 function next(){ seq.push(rand(4)); levelEl.textContent=seq.length; playSeq(); }
 freqs.forEach((f,i)=>{ const b=document.createElement('button'); b.textContent='Note '+(i+1);
b.style.minWidth='70px'; b.addEventListener('click', async ()=>{ flash(b); await tone(f,220);
if(!listening) return; if(i===seq[idx]){ idx++; if(idx===seq.length){ listening=false;
setTimeout(next,400); } } else { msg.textContent='✖ Try again'; listening=false; seq=[];
levelEl.textContent='0'; } }); bank.appendChild(b); buttons.push(b); });
 start.addEventListener('click', ()=>{ seq=[]; levelEl.textContent='0'; msg.textContent='—';
next(); });
 window._ex55={start,bank};
})();

/* 56 Emoji Race */
(function(){
 const a=$('#ex56-a'), start=$('#ex56-start'), youEl=$('#ex56-you'), cpuEl=$('#ex56-cpu');
 const you=document.createElement('div'), cpu=document.createElement('div');
you.textContent='🏃'; cpu.textContent='🤖';
 [you,cpu].forEach(el=>{ el.style.position='absolute'; el.style.top= (el===you? '40px':'100px');
el.style.left='6px'; el.style.fontSize='24px'; a.appendChild(el); });
 let youP=0, cpuP=0, running=false, cpuTimer=0;
 start.addEventListener('click', ()=>{ if(running) return; running=true; youP=0; cpuP=0;
youEl.textContent='0'; cpuEl.textContent='0'; you.style.left='6px'; cpu.style.left='6px';
clearInterval(cpuTimer); cpuTimer=setInterval(()=>{ cpuP = clamp(cpuP + (Math.random()*6), 0,
100); cpuEl.textContent=cpuP.toFixed(0); cpu.style.left=(6 + (a.clientWidth-40)*(cpuP/100))+'px';
```

```
if(cpuP>=100){ running=false; clearInterval(cpuTimer);} }, 150); });
  window.addEventListener('keydown', e=>{ if(!running) return; if(e.code==='Space'){ youP =
clamp(youP+4, 0, 100); youEl.textContent=youP.toFixed(0); you.style.left=(6 +
(a.clientWidth-40)*(youP/100))+'px'; if(youP>=100){ running=false; clearInterval(cpuTimer);} } });
  window._ex56={start,a};
})();

/* 57 Shape Morph Animation */
(function(){
  const a=$('#ex57-a'), r=$('#ex57-r'), play=$('#ex57-play');
  const box=document.createElement('div'); box.style.position='absolute'; box.style.left='100px';
box.style.top='50px'; box.style.width='60px'; box.style.height='60px';
box.style.background='#60a5fa'; box.style.border='1px solid #000'; a.appendChild(box);
  function render(t){ const size=60+(40*t); box.style.width=size+'px'; box.style.height=size+'px';
box.style.borderRadius=(t*50)+'%'; box.style.transform='translateX('+ (t*60)+'px)'; }
  r.addEventListener('input', ()=>render(+r.value/100));
  play.addEventListener('click', async ()=>{ for(let i=0;i<=100;i+=2){ render(i/100); await
wait(16);} for(let i=100;i>=0;i-=2){ render(i/100); await wait(16);} });
  render(0);
  window._ex57={a,play,r};
})();

/* 58 Word Guesser (Hangman-lite) */
(function(){
  const newBtn=$('#ex58-new'), missEl=$('#ex58-miss'), wordEl=$('#ex58-word'),
inp=$('#ex58-in'), tryBtn=$('#ex58-try'), msg=$('#ex58-msg');
  const
bank=['vanilla','browser','closure','context','promise','element','render','adapter','package','networ
k'];
  let target='', view=[], misses=0;
  function fetchWord(){ // simulate fetch
    return new Promise(res=>setTimeout(()=>res(bank[rand(bank.length)]), 200));
  }
  async function newRound(){ target = await fetchWord(); view = Array.from(target, _=>'_');
misses=0; missEl.textContent=misses; wordEl.textContent=view.join(' '); msg.textContent='—';
inp.value=''; inp.focus(); }
  tryBtn.addEventListener('click', ()=>{ const ch=(inp.value||'').toLowerCase();
if(!ch||ch<'a'||ch>'z') return; inp.value=''; let hit=false; for(let i=0;i<target.length;i++){
if(target[i]===ch){ view[i]=ch; hit=true; } } if(!hit){ misses++; missEl.textContent=misses; }
wordEl.textContent=view.join(' '); if(view.join('')===target){ msg.textContent='🎉 Correct!'; } });
  newBtn.addEventListener('click', newRound); newRound();
  window._ex58={newBtn,tryBtn,inp};
})();

/* 59 Trail Drawing Canvas */
(function(){
  const c=$('#ex59-c'), ctx=c.getContext('2d'), color=$('#ex59-color'), w=$('#ex59-w'),
clear=$('#ex59-clear');
```

```
  let drawing=false, last=null;
  c.addEventListener('pointerdown', e=>{ drawing=true; c.setPointerCapture(e.pointerId); const
r=c.getBoundingClientRect(); last={x:e.clientX-r.left,y:e.clientY-r.top}; });
  c.addEventListener('pointermove', e=>{ if(!drawing) return; const r=c.getBoundingClientRect();
const x=e.clientX-r.left, y=e.clientY-r.top; ctx.strokeStyle=color.value; ctx.lineWidth=+w.value;
ctx.lineCap='round'; ctx.beginPath(); ctx.moveTo(last.x,last.y); ctx.lineTo(x,y); ctx.stroke();
last={x,y}; });
  c.addEventListener('pointerup', ()=>{ drawing=false; last=null; });
  clear.addEventListener('click', ()=>ctx.clearRect(0,0,c.width,c.height));
  window._ex59={c,clear};
})();

/* 60 Mini Breakout */
(function(){
  const c=$('#ex60-c'), ctx=c.getContext('2d'), start=$('#ex60-start'), livesEl=$('#ex60-lives'),
scoreEl=$('#ex60-score');
  const W=c.width, H=c.height; let bx,by,bvx,bvy, px=110, bricks=[], lives=3, score=0,
running=false, last=0;
  function resetBall(){ bx=W/2; by=H-20; bvx=(Math.random()>.5?1:-1)*120; bvy=-140; }
  function buildBricks(){ bricks=[]; const cols=7, rows=4, bw=34, bh=14, gap=6, offX=12,
offY=18; for(let y=0;y<rows;y++)for(let x=0;x<cols;x++){
bricks.push({x:offX+x*(bw+gap),y:offY+y*(bh+gap),w:bw,h:bh,alive:true}); } }
  c.addEventListener('mousemove', e=>{ const r=c.getBoundingClientRect(); const
mx=e.clientX-r.left; px=clamp(mx-30,0,W-60); });
  function draw(){
    ctx.clearRect(0,0,W,H);
    // paddle
    ctx.fillStyle='#60a5fa'; ctx.fillRect(px,H-12,60,8); ctx.strokeStyle='#000';
ctx.strokeRect(px,H-12,60,8);
    // ball
    ctx.beginPath(); ctx.fillStyle='#f59e0b'; ctx.arc(bx,by,6,0,Math.PI*2); ctx.fill();
ctx.strokeStyle='#000'; ctx.stroke();
    // bricks
    for(const b of bricks){ if(!b.alive) continue; ctx.fillStyle='#1f2a44'; ctx.fillRect(b.x,b.y,b.w,b.h);
ctx.strokeStyle='#000'; ctx.strokeRect(b.x,b.y,b.w,b.h); }
  }
  function loop(ts){
    if(!running) return;
    if(!last) last=ts; const dt=(ts-last)/1000; last=ts;
    bx+=bvx*dt/1; by+=bvy*dt/1;
    if(bx<6||bx>W-6){ bvx*=-1; bx=clamp(bx,6,W-6); }
    if(by<6){ bvy*=-1; by=6; }
    // paddle hit
    if(by+6>=H-12 && by+6<=H-4 && bx>=px && bx<=px+60){ bvy=-Math.abs(bvy); by=H-12-6; }
    // brick hits
    for(const b of bricks){ if(!b.alive) continue; if(bx>b.x && bx<b.x+b.w && by>b.y &&
by<b.y+b.h){ b.alive=false; bvy*=-1; score++; scoreEl.textContent=score; break; } }
    if(by>H+8){ lives--; livesEl.textContent=lives; resetBall(); if(lives<=0){ running=false; } }
```

```
    draw(); requestAnimationFrame(loop);
  }
  start.addEventListener('click', ()=>{ if(running) return; running=true; lives=3; score=0;
livesEl.textContent=lives; scoreEl.textContent=score; buildBricks(); resetBall(); last=0;
requestAnimationFrame(loop); });
  window._ex60={start,c};
})();

/* Minimal Tests */
(function(){
  const btn=$('#run-tests'), sum=$('#test-summary'), err=$('#test-errors');
  let logs=[]; const ok=m=>logs.push({ok:true,msg:m}); const
bad=m=>logs.push({ok:false,msg:m});
  async function t51(){ const {start}=window._ex51; start.click(); ok('ex51 start'); }
  async function t52(){ const {start}=window._ex52; start.click(); ok('ex52 start'); }
  async function t53(){ const {start}=window._ex53; start.click(); ok('ex53 start'); }
  async function t54(){ const {start}=window._ex54; start.click(); ok('ex54 start'); }
  async function t55(){ const {start}=window._ex55; start.click(); ok('ex55 start'); }
  async function t56(){ const {start}=window._ex56; start.click(); ok('ex56 start'); }
  async function t57(){ const {play}=window._ex57; play.click(); ok('ex57 play'); }
  async function t58(){ const {newBtn}=window._ex58; newBtn.click(); ok('ex58 new'); }
  async function t59(){ const {c}=window._ex59; c.dispatchEvent(new
PointerEvent('pointerdown',{bubbles:true})); c.dispatchEvent(new
PointerEvent('pointerup',{bubbles:true})); ok('ex59 pointer'); }
  async function t60(){ const {start}=window._ex60; start.click(); ok('ex60 start'); }
  btn.addEventListener('click', async ()=>{
    logs=[]; const tests=[t51,t52,t53,t54,t55,t56,t57,t58,t59,t60];
    for(const t of tests){ try{ await t(); }catch(e){ bad(t.name+': '+e.message); } }
    const pass=logs.filter(l=>l.ok).length, fail=logs.length-pass;
    sum.textContent=`${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>
```