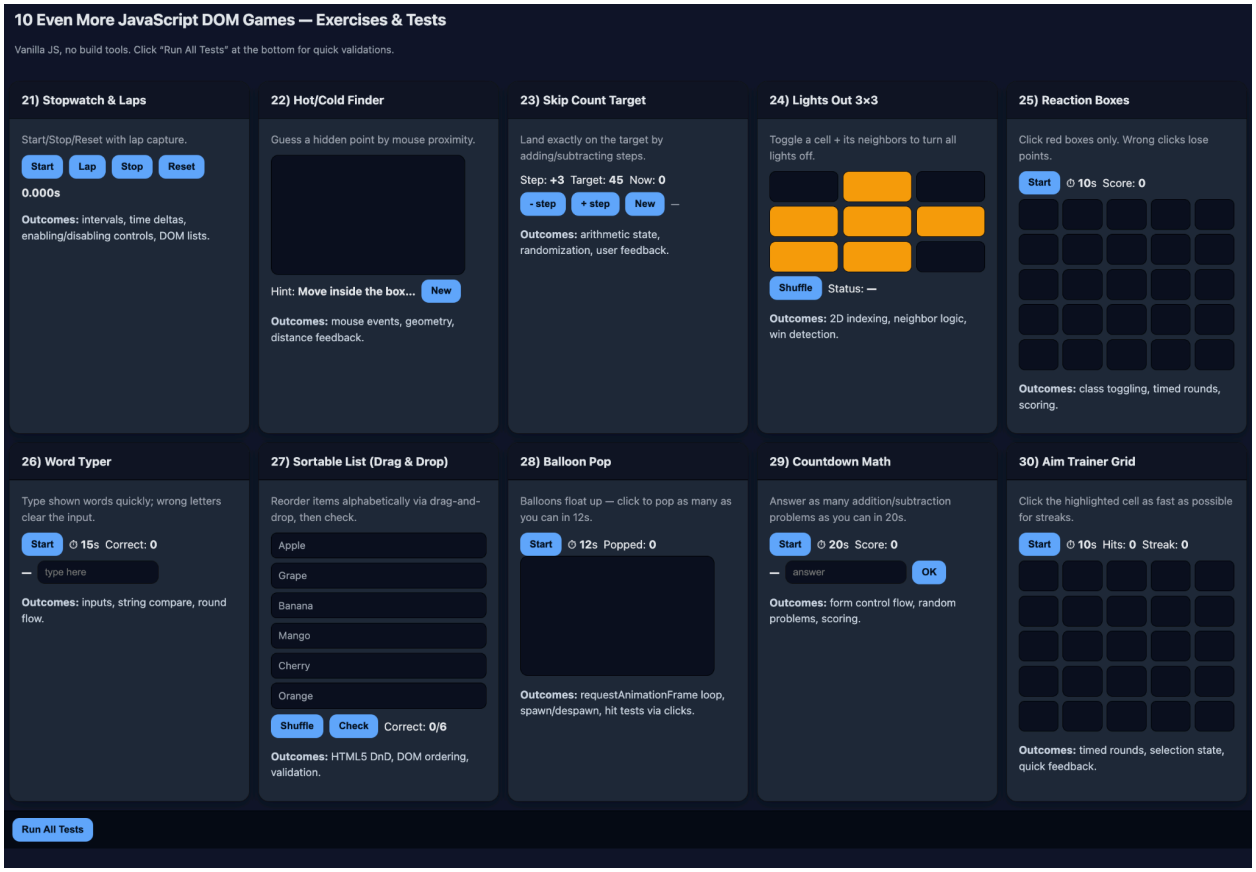

🎮 10 Even More JavaScript DOM Games — Learn by Building (Vanilla JS)

Miss the joy of learning JavaScript by making tiny interactive things? This sequel brings **10 new DOM mini-games** (Exercises **21–30**) you can run by simply opening an `index.html`. They're built with **vanilla JavaScript**, **HTML**, and a dash of **CSS**—no build step, no frameworks. Each card includes a short description, minimal UI, and a **Run All Tests** bar for quick sanity checks.

This volume focuses on: precise **event handling**, **animation loops**, **time-based logic**, **drag & drop**, DOM **state transitions**, and **lightweight architecture** that scales.

What's Inside (Exercises 21–30)

21) Stopwatch & Laps

Goal: Start/stop a millisecond timer, capture laps, reset cleanly.

Core APIs: `setInterval`, `performance.now()`, enabling/disabling controls, DOM list updates.
How it works:

- Buttons are wired with `addEventListener` and a simple `enable()` utility toggles `disabled`.
- `tick()` computes elapsed time as `accumulated + (now - base)` and renders `toFixed(3)`.
- Laps are appended as `<li>` nodes to an ordered list.

Learning outcomes: debouncing controls, measuring time deltas reliably, rendering at ~30 FPS without jank.

---

22) Hot/Cold Finder

Goal: Find a hidden point in a box using proximity hints ("Cold/Warm/Hot").
Core APIs: mouse events, rectangle math (`getBoundingClientRect()`), Euclidean distance.
How it works:

- A random target `[x, y]` is chosen inside the arena.
- On `mousemove`, we compute the cursor's local point and display a hint based on `Math.hypot(dx, dy)`.
- On `click`, if close enough, mark the round as "Found!".

Learning outcomes: mapping page coordinates → local container coordinates; simple geometry for UX.

---

23) Skip Count Target

Goal: Reach an exact target by adding/subtracting a random **step size**.
Core APIs: text updates, button events, simple arithmetic state machine.
How it works:

- A new round picks a step from `[2,3,4,5,7]` and a target between 15–55.
- `+` and `−` mutate `now`, and the UI reports **Exact**, **Overshoot**, or **Below**.

Learning outcomes: communicating state transitions clearly via minimal DOM text updates.

---

24) Lights Out (3×3)

Goal: Toggle cells to switch off all lights; each click toggles the cell + its neighbors.
Core APIs: grid generation, 2D indexing, neighbor toggling.
How it works:

- Cells are created once; each node stores `data-on` (`"1"` or `"0"`).
- On click, we compute `(x, y)` and toggle `[self, up, down, left, right]`.
- A “You win!” check uses `every(c => c.dataset.on === '0')`.

Learning outcomes: compact modeling (data attributes), side-effects on adjacent nodes, win detection.

---

25) Reaction Boxes

Goal: In 10 seconds, click only the **red** boxes; wrong clicks lose points.
Core APIs: periodic rounds (`setInterval`), per-cell styling, scoring logic.
How it works:

- A 5×5 grid is built once; each second, three random cells are colored red.
- Clicks compare the clicked cell’s color; correct → green, wrong → muted slate; score updates instantly.

Learning outcomes: separating **round scheduling** from **click handling**, keeping UI honest with state.

---

26) Word Typer

Goal: Type the displayed word; incorrect letter resets your input.
Core APIs: input events, string prefix checks, timers.
How it works:

- On start, a 15s timer counts down; `input` compares `value` vs `cur`.

- If `!cur.startsWith(value)`, we clear the input. If equal, increment score and rotate the word.

Learning outcomes: resilient input validation and quick feedback loops.

---

27) Sortable List (Drag & Drop)

Goal: Drag items to sort them alphabetically, then click **Check**.
Core APIs: HTML5 Drag & Drop (`dragstart`, `dragover`, `drop`), DOM reordering.
How it works:

- We store the dragged text via `dataTransfer`.
- On `drop`, we compute indices and reinsert the node using `insertBefore()` accordingly.
- **Check** compares current DOM order to the alphabetized array.

Learning outcomes: safe DOM reordering patterns, minimal DnD without dependencies.

---

28) Balloon Pop (Fixed & Smooth)

Goal: Balloons float upward—pop as many as you can in 12 seconds.
Core APIs: `requestAnimationFrame`, **time delta** physics, steady spawning, pointer events, DOM recycling.
What changed (and why it's solid now):

- Uses a **persistent DOM** (no `innerHTML = ''` each frame), so clicks don't get lost.
- Movement uses **time-based** motion (`dt = (ts - lastTs)/1000`) for stable speed across devices.
- Spawning is **cadenced** (every ~350ms) instead of random per frame, ensuring a consistent flow.
- Balloons are cleaned up **in place** (when popped or offscreen), preventing memory leaks.

Learning outcomes: real-time loops with `rAF`, predictable spawning, safe element lifecycle.

---

29) Countdown Math

Goal: Solve fast addition/subtraction problems for 20 seconds.
Core APIs: gated inputs, `setInterval` countdown, small problem generator.
How it works:

- A round produces `a op b`, enables input, and awaits **OK**.
- Correct answers increment score and produce a new problem; out of time disables controls.

Learning outcomes: micro-game loop with form controls and scoring.

---

30) Aim Trainer Grid

Goal: Hit the highlighted cell; maintain streaks before time runs out.
Core APIs: grid creation, "active" index, timed randomization.
How it works:

- A 5×5 grid is built once; every second, a random active index is highlighted.
- Clicks compare `i === active` to update **Hits** and **Streak**.

Learning outcomes: tight feedback, quick state flips, pacing via timers.

Architecture Notes

- **Helpers**: tiny `$()` and `$$()` selectors, `rand(n)`, and `wait(ms)` keep code clear.
- **One-time DOM builds**: grids/lists are created once; rounds only update styles/data, not the structure.
- **State machines**: each game keeps local state (`running`, `timeLeft`, `score`, etc.) for predictable flows.
- **Test bar**: a minimal runner simulates clicks/inputs to verify that events are wired and core logic responds.

Key Techniques You'll Practice

- Timekeeping with `performance.now()` and `setInterval`
- Animation with `requestAnimationFrame` + **time deltas**
- Mouse & keyboard events, pointer events on touch devices
- Drag & Drop reordering without libraries
- Grid layout generation and neighbor logic
- Clean element lifecycle (create → update → remove)

How to Use These Games

1. Save or clone the project and open `index.html` in your browser.
2. Play any card—each one is self-contained.
3. Click **Run All Tests** to sanity-check your wiring.
4. Tweak rules, styles, and difficulty to make each mechanic your own.

Extend the Challenges

- **Stopwatch**: add split averages and best lap highlighting.
- **Hot/Cold**: show a mini radar dot when you're close.
- **Lights Out**: change board size (5×5) and auto-generate solvable states.
- **Reaction Boxes**: scale difficulty by time, count of red boxes, or penalties.
- **Balloon Pop**: spawn color tiers with different speeds/scores; add gentle sway (`x` sinusoid).
- **Aim Trainer**: record reaction time per hit and chart streaks.

Final Thoughts

These compact projects turn DOM APIs into **tangible skills**—you'll read coordinates, reason about time, move pixels, and ship tiny features with confidence. Because everything's **vanilla JS**, you can lift ideas straight into any codebase or classroom.

```
<!doctype html>
<html lang="en">
<head>
<meta charset="utf-8"/>
<meta name="viewport" content="width=device-width,initial-scale=1"/>
<title>10 Even More JavaScript DOM Games</title>
<style>

:root{--bg:#0f172a;--panel:#111827;--card:#1f2937;--accent:#60a5fa;--ok:#10b981;--bad:#ef4444;--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}
 input[type=text],input[type=number]{padding:.45rem
.6rem;border-radius:10px;border:1px solid
#111;background:#0b1220;color:#e5e7eb}
 progress{width:100%}
 .grid{display:grid;gap:6px}
 .cell{height:40px;border-radius:8px;border:1px solid
#000;background:#0b1220;display:grid;place-items:center;cursor:pointer}
 .grid-3{grid-template-columns:repeat(3,1fr)}
 .grid-5{grid-template-columns:repeat(5,1fr)}
 .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}
 /* Balloon visuals and clickability */
.balloon{
```

```
 position:absolute;
 border-radius:50%;
 background:#ef4444;
 border:1px solid #000;
 touch-action:none;          /* better pointerdown on touch */
 cursor:pointer;
 will-change: transform, top; /* smoother movement */
}
.balloon.popped{
 background:#22c55e;
 transform: scale(0.85);
 opacity: .8;
}

</style>
</head>
<body>
<header>
 <h1>10 Even More JavaScript DOM Games — Exercises & Tests</h1>
 <p class="small">Vanilla JS, no build tools. Click “Run All Tests” at the
bottom for quick validations.</p>
</header>

<div class="wrap">

 <!-- 21) Stopwatch & Laps -->
 <section class="card" id="ex21">
   <header><h2>21) Stopwatch & Laps</h2></header>
   <div class="body">
```

```
    <div class="meta">Start/Stop/Reset with lap capture.</div>
    <div class="row">
      <button id="ex21-start">Start</button>
      <button id="ex21-lap" disabled>Lap</button>
      <button id="ex21-stop" disabled>Stop</button>
      <button id="ex21-reset" disabled>Reset</button>
      <strong><span id="ex21-time">0.000</span>s</strong>
    </div>
    <ol id="ex21-laps" class="small" style="margin:.5rem 0 0"></ol>
    <p class="note"><strong>Outcomes:</strong> intervals, time deltas,
enabling/disabling controls, DOM lists.</p>
  </div>
 </section>

 <!-- 22) Hot/Cold Finder -->
 <section class="card" id="ex22">
  <header><h2>22) Hot/Cold Finder</h2></header>
  <div class="body">
    <div class="meta">Guess a hidden point by mouse proximity.</div>
    <div class="arena" id="ex22-arena" title="move mouse"></div>
    <div class="row" style="margin-top:6px">
      <span>Hint: <b id="ex22-hint">Move inside the box…</b></span>
      <button id="ex22-new">New</button>
    </div>
    <p class="note"><strong>Outcomes:</strong> mouse events,
geometry, distance feedback.</p>
  </div>
 </section>
```

```
 <!-- 23) Skip Count Target -->
 <section class="card" id="ex23">
   <header><h2>23) Skip Count Target</h2></header>
   <div class="body">
     <div class="meta">Land exactly on the target by adding/subtracting
steps.</div>
     <div class="row">
       <span>Step: <b id="ex23-step">+3</b></span>
       <span>Target: <b id="ex23-target">—</b></span>
       <span>Now: <b id="ex23-now">0</b></span>
     </div>
     <div class="row" style="margin-top:6px">
       <button id="ex23-minus">- step</button>
       <button id="ex23-plus">+ step</button>
       <button id="ex23-new">New</button>
       <span class="small" id="ex23-msg">—</span>
     </div>
     <p class="note"><strong>Outcomes:</strong> arithmetic state,
randomization, user feedback.</p>
   </div>
 </section>

 <!-- 24) Lights Out 3×3 -->
 <section class="card" id="ex24">
   <header><h2>24) Lights Out 3×3</h2></header>
   <div class="body">
     <div class="meta">Toggle a cell + its neighbors to turn all lights
off.</div>
     <div id="ex24-grid" class="grid grid-3" style="margin-top:6px"></div>
```

```
    <div class="row" style="margin-top:6px">
      <button id="ex24-new">Shuffle</button>
      <span>Status: <b id="ex24-status">—</b></span>
    </div>
    <p class="note"><strong>Outcomes:</strong> 2D indexing, neighbor logic, win detection.</p>
  </div>
</section>

<!-- 25) Reaction Boxes -->
<section class="card" id="ex25">
  <header><h2>25) Reaction Boxes</h2></header>
  <div class="body">
    <div class="meta">Click red boxes only. Wrong clicks lose points.</div>
    <div class="row">
      <button id="ex25-start">Start</button>
      <span>⏱ <b id="ex25-time">10</b>s</span>
      <span>Score: <b id="ex25-score">0</b></span>
    </div>
    <div id="ex25-grid" class="grid grid-5" style="margin-top:6px"></div>
    <p class="note"><strong>Outcomes:</strong> class toggling, timed rounds, scoring.</p>
  </div>
</section>

<!-- 26) Word Typer -->
<section class="card" id="ex26">
  <header><h2>26) Word Typer</h2></header>
  <div class="body">
```

```
    <div class="meta">Type shown words quickly; wrong letters clear the input.</div>
    <div class="row">
      <button id="ex26-start">Start</button>
      <span>⏱ <b id="ex26-time">15</b>s</span>
      <span>Correct: <b id="ex26-cnt">0</b></span>
    </div>
    <div class="row" style="margin-top:6px"><b id="ex26-word">—</b><input id="ex26-in" type="text" disabled placeholder="type here"></div>
    <p class="note"><strong>Outcomes:</strong> inputs, string compare, round flow.</p>
  </div>
 </section>

 <!-- 27) Sortable List (DnD) -->
 <section class="card" id="ex27">
  <header><h2>27) Sortable List (Drag & Drop)</h2></header>
  <div class="body">
    <div class="meta">Reorder items alphabetically via drag-and-drop, then check.</div>
    <ul id="ex27-list" class="small" style="list-style:none;padding-left:0;margin:.5rem 0;display:grid;gap:6px"></ul>
    <div class="row">
      <button id="ex27-shuffle">Shuffle</button>
      <button id="ex27-check">Check</button>
      <span>Correct: <b id="ex27-ok">0</b>/<b id="ex27-total">0</b></span>
```

```
    </div>
    <p class="note"><strong>Outcomes:</strong> HTML5 DnD, DOM ordering, validation.</p>
  </div>
 </section>

 <!-- 28) Balloon Pop (Upward drift) -->
 <section class="card" id="ex28">
  <header><h2>28) Balloon Pop</h2></header>
  <div class="body">
    <div class="meta">Balloons float up — click to pop as many as you can in 12s.</div>
    <div class="row">
      <button id="ex28-start">Start</button>
      <span>⏱ <b id="ex28-time">12</b>s</span>
      <span>Popped: <b id="ex28-score">0</b></span>
    </div>
    <div class="arena" id="ex28-arena"></div>
    <p class="note"><strong>Outcomes:</strong> requestAnimationFrame loop, spawn/despawn, hit tests via clicks.</p>
  </div>
 </section>

 <!-- 29) Countdown Math -->
 <section class="card" id="ex29">
  <header><h2>29) Countdown Math</h2></header>
  <div class="body">
    <div class="meta">Answer as many addition/subtraction problems as you can in 20s.</div>
```

```
    <div class="row">
      <button id="ex29-start">Start</button>
      <span>⏱ <b id="ex29-time">20</b>s</span>
      <span>Score: <b id="ex29-score">0</b></span>
    </div>
    <div class="row" style="margin-top:6px">
      <b id="ex29-q">—</b>
      <input id="ex29-in" type="number" disabled placeholder="answer"/>
      <button id="ex29-ok" disabled>OK</button>
    </div>
    <p class="note"><strong>Outcomes:</strong> form control flow,
random problems, scoring.</p>
  </div>
 </section>

 <!-- 30) Aim Trainer Grid -->
 <section class="card" id="ex30">
  <header><h2>30) Aim Trainer Grid</h2></header>
  <div class="body">
    <div class="meta">Click the highlighted cell as fast as possible for
streaks.</div>
    <div class="row">
      <button id="ex30-start">Start</button>
      <span>⏱ <b id="ex30-time">10</b>s</span>
      <span>Hits: <b id="ex30-hits">0</b></span>
      <span>Streak: <b id="ex30-streak">0</b></span>
    </div>
    <div id="ex30-grid" class="grid grid-5" style="margin-top:6px"></div>
    <p class="note"><strong>Outcomes:</strong> timed rounds, selection
```

```
state, quick feedback.</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));

/* 21) Stopwatch & Laps */
(function(){
 const start=$('#ex21-start'), lap=$('#ex21-lap'), stop=$('#ex21-stop'),
reset=$('#ex21-reset'), tEl=$('#ex21-time'), list=$('#ex21-laps');
 let running=false, base=0, acc=0, id=null;
 function tick(){ const ms = acc + (running ? (performance.now()-base) :
0); tEl.textContent = (ms/1000).toFixed(3); }
 function enable(a,b,c,d){ start.disabled=!a; lap.disabled=!b;
stop.disabled=!c; reset.disabled=!d; }
```

```
 start.addEventListener('click', ()=>{ if(running) return; running=true;
base=performance.now(); id=setInterval(tick,33);
enable(false,true,true,true); });
 stop.addEventListener('click', ()=>{ if(!running) return; running=false; acc
+= performance.now()-base; clearInterval(id); tick();
enable(true,false,false,true); });
 reset.addEventListener('click', ()=>{ running=false; base=0; acc=0;
clearInterval(id); tEl.textContent='0.000'; list.innerHTML='';
enable(true,false,true,false); });
 lap.addEventListener('click', ()=>{ if(!running) return; const
li=document.createElement('li'); li.textContent=tEl.textContent+'s';
list.appendChild(li); });
 enable(true,false,true,false);
 window._ex21 = {start,stop,reset,lap,tEl};
})();

/* 22) Hot/Cold Finder */
(function(){
 const box=$('#ex22-arena'), hint=$('#ex22-hint'),
newBtn=$('#ex22-new');
 let target=[0,0];
 function newRound(){ const r=box.getBoundingClientRect();
target=[rand(r.width), rand(r.height)]; hint.textContent='Move inside the
box…'; }
 function dist([x,y],[a,b]){ const dx=x-a, dy=y-b; return Math.hypot(dx,dy);
}
 box.addEventListener('mousemove', e=>{
   const r=box.getBoundingClientRect(); const p=[e.clientX - r.left, e.clientY
- r.top];
```

```
  const d=dist(p,target);
  hint.textContent = d<20 ? '🔥 Hot! Click!' : d<60 ? 'Warm' : 'Cold';
 });
 box.addEventListener('click', e=>{
  const r=box.getBoundingClientRect(); const p=[e.clientX - r.left, e.clientY
- r.top];
  if(dist(p,target)<20){ hint.textContent='🎯 Found!'; }
 });
 newBtn.addEventListener('click', newRound); newRound();
 window._ex22 = {box,newBtn,hint};
})();

/* 23) Skip Count Target */
(function(){
 const stepEl=$('#ex23-step'), targetEl=$('#ex23-target'),
nowEl=$('#ex23-now'), msg=$('#ex23-msg');
 const plus=$('#ex23-plus'), minus=$('#ex23-minus'),
btnNew=$('#ex23-new');
 let step=3, now=0, target=0;
 function setRound(){ step = [2,3,4,5,7][rand(5)]; now=0; target = 15 +
rand(41); stepEl.textContent='+'+step; nowEl.textContent=now;
targetEl.textContent=target; msg.textContent='—'; }
 plus.addEventListener('click', ()=>{ now+=step; nowEl.textContent=now;
msg.textContent = now===target ? '🎯 Exactly!' : (now>target?
'Overshoot!':'...'); });
 minus.addEventListener('click', ()=>{ now-=step; nowEl.textContent=now;
msg.textContent = now===target ? '🎯 Exactly!' : (now<target?
'Below...':'...'); });
 btnNew.addEventListener('click', setRound); setRound();
```

```
 window._ex23 = {plus,minus,btnNew,nowEl,targetEl};
})();

/* 24) Lights Out 3×3 */
(function(){
 const g=$('#ex24-grid'), status=$('#ex24-status'), btn=$('#ex24-new');
 let cells=[];
 function idx(x,y){ return y*3+x; }
 function render(seed=true){
   g.innerHTML=''; cells=[];
   for(let i=0;i<9;i++){ const d=document.createElement('div');
d.className='cell'; d.dataset.on = seed ? (rand(2)?'1':'0') : '0';
d.style.background = d.dataset.on==='1'?'#f59e0b':'#0b1220';
g.appendChild(d); cells.push(d); }
 }
 function toggle(x,y){ if(x<0||x>=3||y<0||y>=3) return; const
c=cells[idx(x,y)]; c.dataset.on = c.dataset.on==='1'?'0':'1';
c.style.background = c.dataset.on==='1'?'#f59e0b':'#0b1220'; }
 g.addEventListener('click', e=>{
   if(!e.target.classList.contains('cell')) return;
   const i = cells.indexOf(e.target), x=i%3, y=Math.floor(i/3);
   [[0,0],[1,0],[-1,0],[0,1],[0,-1]].forEach(([dx,dy])=>toggle(x+dx,y+dy));
   if(cells.every(c=>c.dataset.on==='0')) status.textContent='You win!';
 });
 btn.addEventListener('click', ()=>{ render(true); status.textContent='—';
});
 render(true);
 window._ex24 = {g,btn,status};
})();
```

```
/* 25) Reaction Boxes */
(function(){
 const grid=$('#ex25-grid'), start=$('#ex25-start'), tEl=$('#ex25-time'),
scoreEl=$('#ex25-score');
 let t=10, timer=null, active=[], score=0, playing=false;
 for(let i=0;i<25;i++){ const d=document.createElement('div');
d.className='cell'; grid.appendChild(d); }
 const cells=$$('#ex25-grid .cell');
 function clearColors(){ cells.forEach(c=>c.style.background='#0b1220'); }
 function roundTick(){
   clearColors(); active = [];
   for(let i=0;i<3;i++){ const c = cells[rand(25)];
c.style.background='#ef4444'; active.push(c); }
 }
 start.addEventListener('click', ()=>{
   if(playing) return; playing=true; score=0; scoreEl.textContent=score;
t=10; tEl.textContent=t; roundTick();
   timer=setInterval(()=>{ t--; tEl.textContent=t; roundTick(); if(t<=0){
clearInterval(timer); playing=false; clearColors(); } },1000);
 });
 grid.addEventListener('click', e=>{
   if(!playing || !e.target.classList.contains('cell')) return;
   if(e.target.style.background.includes('ef4444')){ score++;
e.target.style.background='#22c55e'; }
   else { score=Math.max(0, score-1);
e.target.style.background='#334155'; }
   scoreEl.textContent=score;
 });
```

```
 window._ex25 = {start,grid};
})();

/* 26) Word Typer */
(function(){
 const words =
['array','object','event','promise','module','bundle','script','render','update','vi
rtual','kernel','binary','socket','driver','thread'];
 const start=$('#ex26-start'), tEl=$('#ex26-time'), cntEl=$('#ex26-cnt'),
wordEl=$('#ex26-word'), input=$('#ex26-in');
 let t=15, score=0, timer=null, cur='';
 function newWord(){ cur = words[rand(words.length)];
wordEl.textContent=cur; input.value=''; }
 start.addEventListener('click', ()=>{
   score=0; cntEl.textContent=score; t=15; tEl.textContent=t;
input.disabled=false; input.focus(); newWord();
   clearInterval(timer);
   timer=setInterval(()=>{ t--; tEl.textContent=t; if(t<=0){
clearInterval(timer); input.disabled=true; wordEl.textContent='—'; }
},1000);
 });
 input.addEventListener('input', ()=>{
   const v=input.value;
   if(!cur.startsWith(v)){ input.value=''; return; }
   if(v===cur){ score++; cntEl.textContent=score; newWord(); }
 });
 window._ex26 = {start,input,wordEl};
})();
```

```
/* 27) Sortable List (DnD) */
(function(){
 const items=['Banana','Apple','Grape','Orange','Mango','Cherry'];
 const list=$('#ex27-list'), shuffle=$('#ex27-shuffle'),
check=$('#ex27-check'), okEl=$('#ex27-ok'), totalEl=$('#ex27-total');
 totalEl.textContent = items.length;
 function render(arr){
   list.innerHTML='';
   arr.forEach((txt,i)=>{
     const li=document.createElement('li'); li.textContent=txt;
li.draggable=true; li.style.padding='.4rem .6rem';
li.style.background='#0b1220'; li.style.border='1px solid #000';
li.style.borderRadius='8px';
     li.addEventListener('dragstart',e=>{ e.dataTransfer.setData('text/plain',
txt); });
     li.addEventListener('dragover',e=>e.preventDefault());
     li.addEventListener('drop',e=>{
       e.preventDefault();
       const draggedText=e.dataTransfer.getData('text/plain');
       const src =
Array.from(list.children).findIndex(n=>n.textContent===draggedText);
       const dst = Array.from(list.children).indexOf(li);
       if(src<0||dst<0||src===dst) return;
       const nodes = Array.from(list.children);
       list.insertBefore(nodes[src], dst>src? nodes[dst].nextSibling :
nodes[dst]);
     });
     list.appendChild(li);
   });
```

```
 }
 shuffle.addEventListener('click', ()=>{
render(items.slice().sort(()=>Math.random()-0.5)); okEl.textContent='0';
});
 check.addEventListener('click', ()=>{
   const got = Array.from(list.children).map(n=>n.textContent);
   let ok=0; const sorted = items.slice().sort();
   for(let i=0;i<sorted.length;i++){ if(got[i]===sorted[i]) ok++; }
   okEl.textContent = ok;
 });
 render(items.slice().sort(()=>Math.random()-0.5));
 window._ex27 = {list,shuffle,check};
})();

/* 28) Balloon Pop */

(function(){
 const start=$('#ex28-start'), tEl=$('#ex28-time'),
scoreEl=$('#ex28-score'), arena=$('#ex28-arena');

 let running=false, score=0, timeLeft=12;
 let rafId=0, lastTs=0, spawnAccum=0, timerId=0;

 /** Balloon model: { x, y, r, vy, dead, el } */
 const balloons=[];

 function makeBalloon(){
   const w = arena.clientWidth || 260;
   const h = arena.clientHeight || 160;
```

```
  const r = 12 + rand(12);          // radius
  const x = r + rand(Math.max(1, w - 2*r)); // keep inside arena
  const y = h + r + 4;              // just below bottom
  const vy = 90 + rand(90);         // px/sec
  const el = document.createElement('div');
  el.className = 'balloon';
  // size + position
  el.style.width = (2*r) + 'px';
  el.style.height = (2*r) + 'px';
  el.style.left = (x - r) + 'px';
  el.style.top = (y - r) + 'px';
  el.addEventListener('pointerdown', ()=>{
    if(!running) return;
    const b = balloons.find(k => k.el === el);
    if(!b || b.dead) return;
    b.dead = true;
    el.classList.add('popped');
    score++; scoreEl.textContent = score;
  });
  arena.appendChild(el);
  balloons.push({x,y,r,vy,dead:false,el});
}

function reset(){
  running=false;
  cancelAnimationFrame(rafId);
  clearInterval(timerId);
  lastTs=0; spawnAccum=0;
  balloons.splice(0, balloons.length);
```

```
  arena.innerHTML = ''; // clear once
  score=0; scoreEl.textContent=score;
  timeLeft=12; tEl.textContent=timeLeft;
}

function loop(ts){
  if(!running) return;
  if(!lastTs) lastTs = ts;
  const dt = (ts - lastTs) / 1000; // seconds
  lastTs = ts;

  // steady spawn: one balloon every ~350ms
  spawnAccum += dt;
  while(spawnAccum >= 0.35){
    spawnAccum -= 0.35;
    makeBalloon();
  }

  // move + cull
  const h = arena.clientHeight || 160;
  for(const b of balloons){
    if(b.dead) continue;
    b.y -= b.vy * dt;
    if(b.y < -b.r - 10){
      b.dead = true; // drifted off top
    }
  }

  // update DOM without nuking it
```

```
  for(const b of balloons){
    if(b.el && !b.dead){
      b.el.style.top = (b.y - b.r) + 'px';
    }
  }

  // remove popped/offscreen nodes in-place
  for(let i=balloons.length-1;i>=0;i--){
    const b = balloons[i];
    if(b.dead){
      b.el?.remove();
      balloons.splice(i,1);
    }
  }

  rafId = requestAnimationFrame(loop);
}

start.addEventListener('click', ()=>{
  if(running) return;
  reset();
  running = true;

  // countdown
  timerId = setInterval(()=>{
    timeLeft--;
    tEl.textContent = timeLeft;
    if(timeLeft <= 0){
      clearInterval(timerId);
```

```
      running = false;
    }
  }, 1000);

  rafId = requestAnimationFrame(loop);
 });

 // expose for tests if needed
 window._ex28 = { start, arena };
})();



/* 29) Countdown Math */
(function(){
 const start=$('#ex29-start'), tEl=$('#ex29-time'),
scoreEl=$('#ex29-score'), qEl=$('#ex29-q'), input=$('#ex29-in'),
ok=$('#ex29-ok');
 let t=20, score=0, a=0,b=0,op='+',ans=0, timer=null;
 function next(){ a=1+rand(20); b=1+rand(20); op = rand(2)?'+':'-'; ans =
op==='+' ? a+b : a-b; qEl.textContent=`${a} ${op} ${b} =`;
input.value=''; input.focus(); }
 start.addEventListener('click', ()=>{ score=0; scoreEl.textContent=score;
t=20; tEl.textContent=t; input.disabled=false; ok.disabled=false; next();
clearInterval(timer); timer=setInterval(()=>{ t--; tEl.textContent=t;
if(t<=0){ clearInterval(timer); input.disabled=true; ok.disabled=true;
qEl.textContent='—'; } },1000); });
 ok.addEventListener('click', ()=>{ if(input.disabled) return;
if(+input.value===ans){ score++; scoreEl.textContent=score; next(); }
```

```
else { input.value=''; } });
 window._ex29 = {start,ok,input};
})();

/* 30) Aim Trainer Grid */
(function(){
 const grid=$('#ex30-grid'), start=$('#ex30-start'), tEl=$('#ex30-time'),
hitsEl=$('#ex30-hits'), streakEl=$('#ex30-streak');
 let t=10, hits=0, streak=0, active=-1, timer=null, playing=false;
 for(let i=0;i<25;i++){ const d=document.createElement('div');
d.className='cell'; grid.appendChild(d); }
 const cells=$$('#ex30-grid .cell');
 function setActive(i){ cells.forEach(c=>c.style.background='#0b1220');
if(i>=0){ cells[i].style.background='#60a5fa'; } active=i; }
 start.addEventListener('click', ()=>{
   if(playing) return; playing=true; t=10; hits=0; streak=0;
hitsEl.textContent=hits; streakEl.textContent=streak;
   setActive(rand(25));
   clearInterval(timer);
   timer=setInterval(()=>{ t--; tEl.textContent=t; setActive(rand(25));
if(t<=0){ clearInterval(timer); playing=false; setActive(-1); } },1000);
 });
 grid.addEventListener('click', e=>{
   if(!playing || !e.target.classList.contains('cell')) return;
   const i = cells.indexOf(e.target);
   if(i===active){ hits++; streak++; } else { streak=0; }
   hitsEl.textContent=hits; streakEl.textContent=streak; setActive(rand(25));
 });
 window._ex30 = {start,grid};
```

```
})();

/* ----------------------
  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 t21(){ const {start,stop,reset,lap,tEl}=window._ex21;
start.click(); await wait(120); lap.click(); stop.click(); reset.click(); ok('ex21
controls wired'); }
 async function t22(){ const {box}=window._ex22; box.dispatchEvent(new
MouseEvent('mousemove',{bubbles:true,clientX:box.getBoundingClientRect(
).left+5,clientY:box.getBoundingClientRect().top+5})); ok('ex22 proximity
responds'); }
 async function t23(){ const {plus,minus}=window._ex23; plus.click();
minus.click(); ok('ex23 buttons wired'); }
 async function t24(){ const {g}=window._ex24; const
c=g.querySelector('.cell'); c && c.click(); ok('ex24 grid clickable'); }
 async function t25(){ const {start,grid}=window._ex25; start.click(); await
wait(300); grid.click(); ok('ex25 round runs'); }
 async function t26(){ const {start,input,wordEl}=window._ex26;
start.click(); await wait(50); input.value=wordEl.textContent;
input.dispatchEvent(new Event('input',{bubbles:true})); ok('ex26 typing
works'); }
 async function t27(){ const {list}=window._ex27; ok('ex27 list exists:
```

```
'+!!list); }
 async function t28(){ const {start}=window._ex28; start.click(); ok('ex28 starts'); }
 async function t29(){ const {start,ok:okBtn,input}=window._ex29; start.click(); await wait(50); input.value='0'; okBtn.click(); ok('ex29 submit hooked'); }
 async function t30(){ const {start,grid}=window._ex30; start.click(); await wait(200); grid.click(); ok('ex30 playable'); }

 btn.addEventListener('click', async ()=>{
   logs=[];
   const tests=[t21,t22,t23,t24,t25,t26,t27,t28,t29,t30];
   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>
```