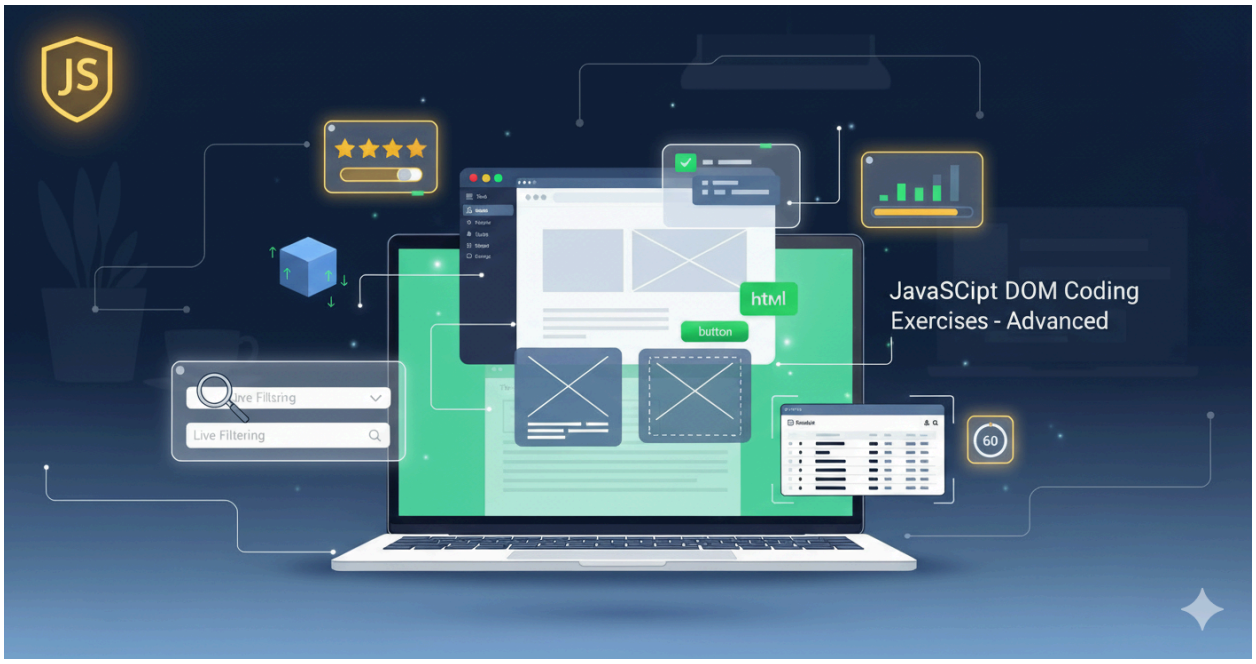
JavaScript DOM Coding Exercises



JavaScript DOM Coding Exercises **1**

31) MutationObserver: Live DOM Change Log 2

32) IntersectionObserver: Lazy Images + Sentinels 3

33) Drag & Drop Sortable List 4

34) Multi-Step Form Wizard (with Validation) 5

35) Accessible Tooltip (hover + focus + positioning) 6

36) Modal Dialog with Focus Trap 7

37) contenteditable Notes with Undo/Redo 8

38) Notification Center (Delegation + localStorage) 9

39) Palette Builder (Color Picker + Copy) 10

40) Scroll-Spy Navigation (IntersectionObserver) 11

Source code **12**

31) MutationObserver: Live DOM Change Log — exercise31_mutation_observer.html 12

32) IntersectionObserver: Lazy Images + Sentinels — exercise32_intersection_observer_lazy.html 15

33) Drag & Drop Sortable List — exercise33_sortable_list.html 17

34) Multi-Step Form Wizard (with Validation) — exercise34_form_wizard.html 20

35) Accessible Tooltip (hover + focus + positioning) — exercise35_tooltip.html 23

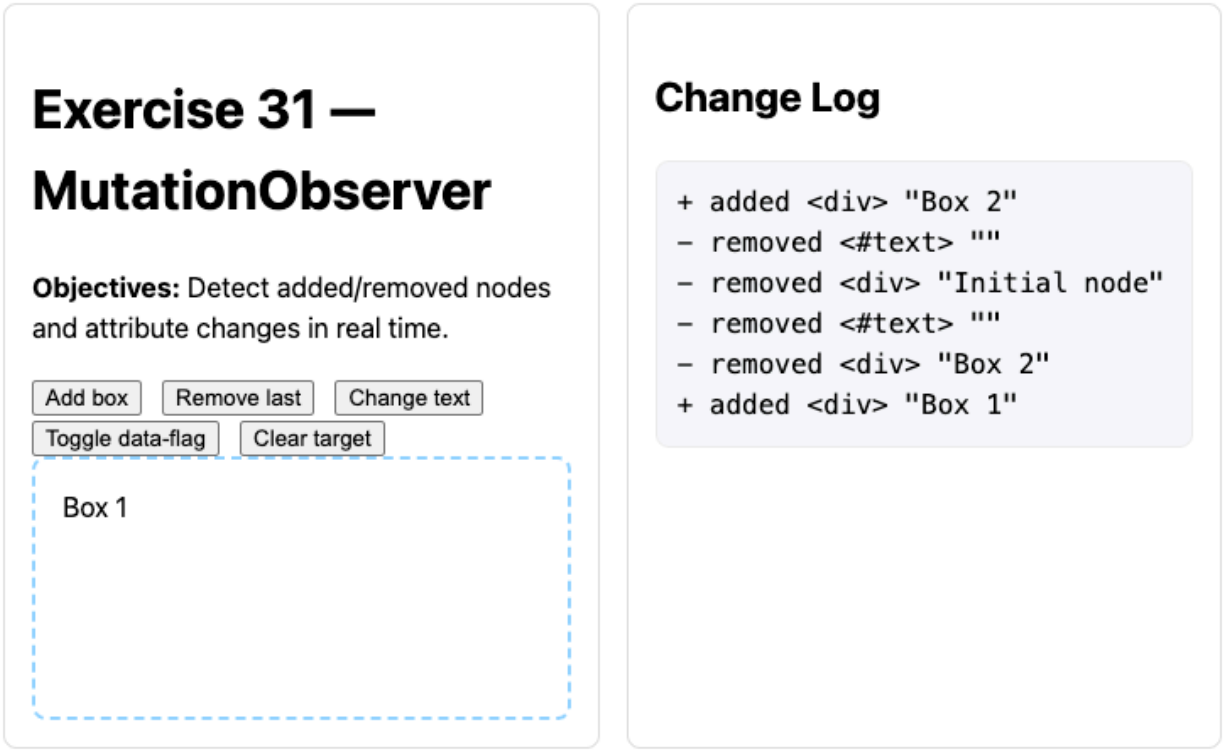
36) Modal Dialog with Focus Trap — exercise36_modal_focus_trap.html 25

37) contenteditable Notes with Undo/Redo — exercise37_contenteditable_undo.html 27

38) Notification Center (Delegation + localStorage) —

exercise38_notifications_localstorage.html 30
39) Palette Builder (Color Picker + Copy) — exercise39_palette_builder.html 33
40) Scroll-Spy Navigation (IntersectionObserver) — exercise40_scroll_spy.html 36

31) MutationObserver: Live DOM Change Log



Build: A playground that logs DOM changes (add/remove nodes, attribute/text changes) from a target container in real time.

Objectives:

- Configure MutationObserver with childList, attributes, subtree, and characterData.
- Inspect MutationRecords to distinguish node additions/removals and attribute changes.
- Append log lines efficiently and auto-scroll a log view.

Steps:

1. Open the file → see the **target** box and **log** panel.

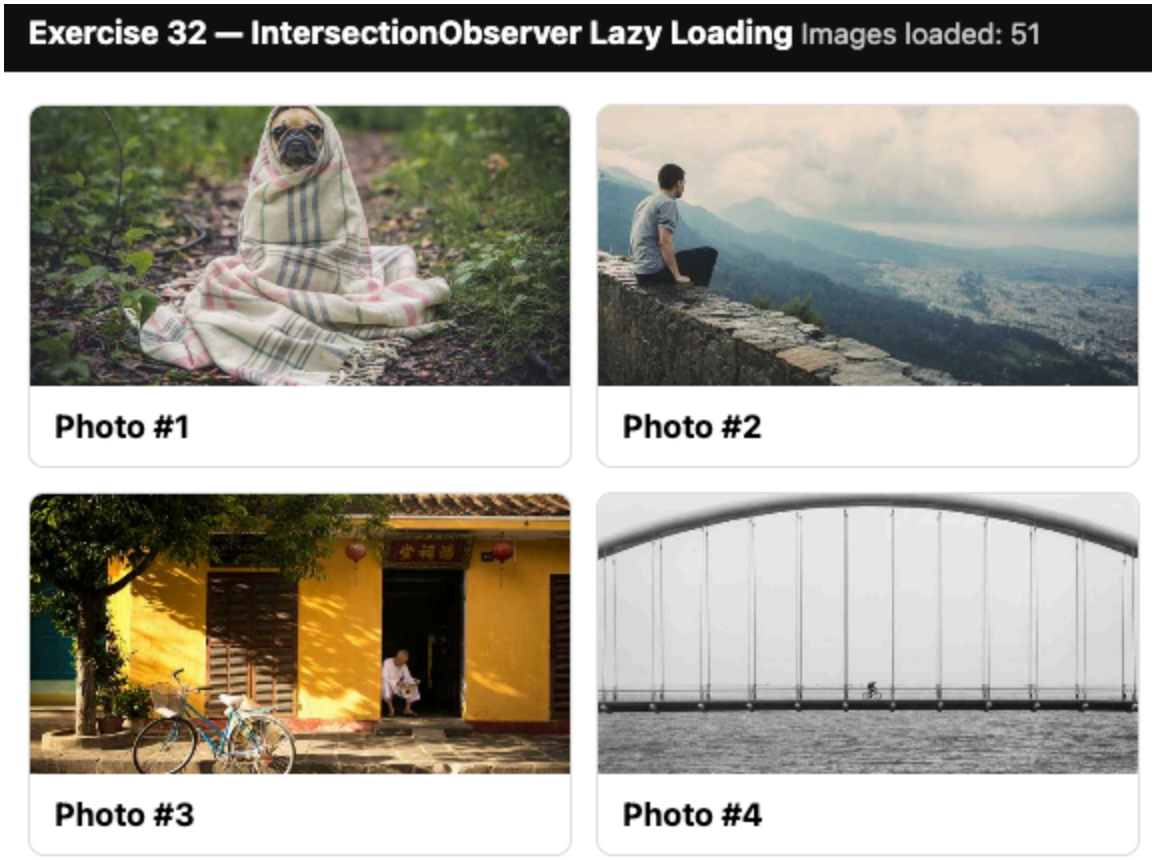
2. Click **Add box**, **Remove last**, **Change text**, **Toggle data-flag**, or **Clear target**.

3. Watch the **Change Log** update immediately with the type of mutation and relevant details.

4. Review how the observer is started and options are chosen; stop or modify as needed.

---

32) IntersectionObserver: Lazy Images + Sentinels



Build: A grid of 60 images that load only when they approach the viewport; a sentinel at the end indicates “end reached.”
Objectives:

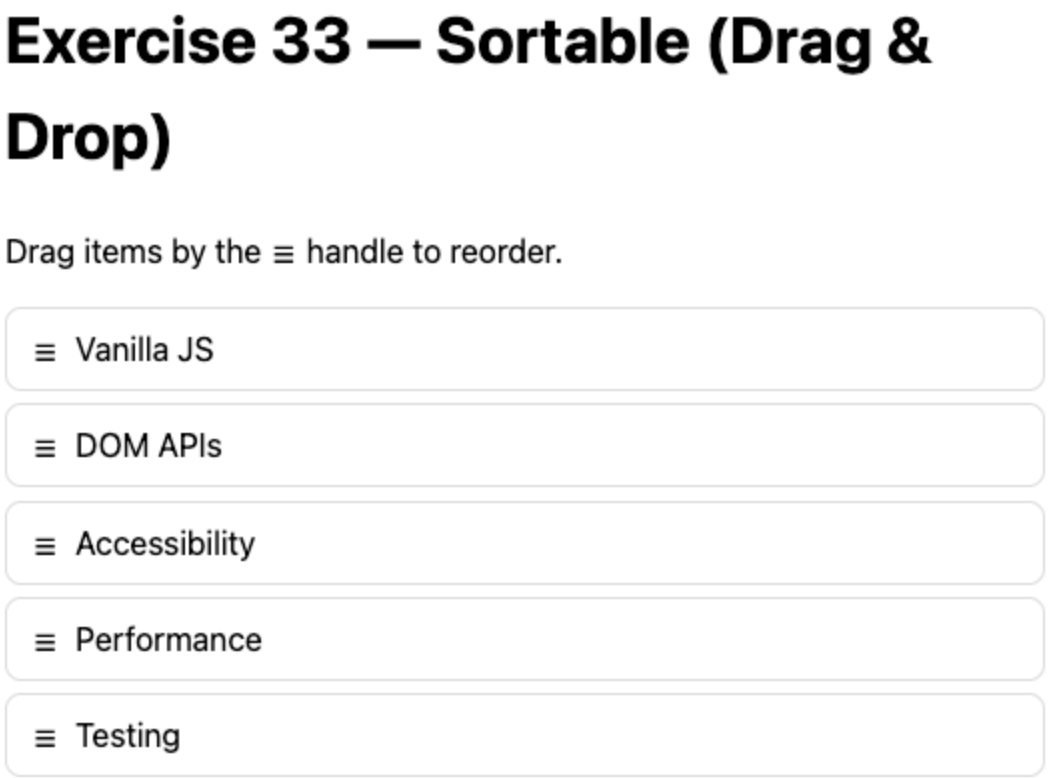
- Use IntersectionObserver for lazy-loading images via data-src.

- Tune rootMargin for earlier/later preloading.
- Optionally observe a sentinel element (pattern used for infinite scroll).
  Steps:

1. Open the page and **scroll the grid**—images replace placeholders as they near the viewport.
2. Watch **"Images loaded: N"** increment in the sticky header.
3. Scroll to the bottom; the sentinel updates status to show the end.
4. Inspect how each img is unobserved after it loads.

---

33) Drag & Drop Sortable List



Build: A list that can be reordered by dragging items using the HTML5 Drag & Drop

API.

Objectives:

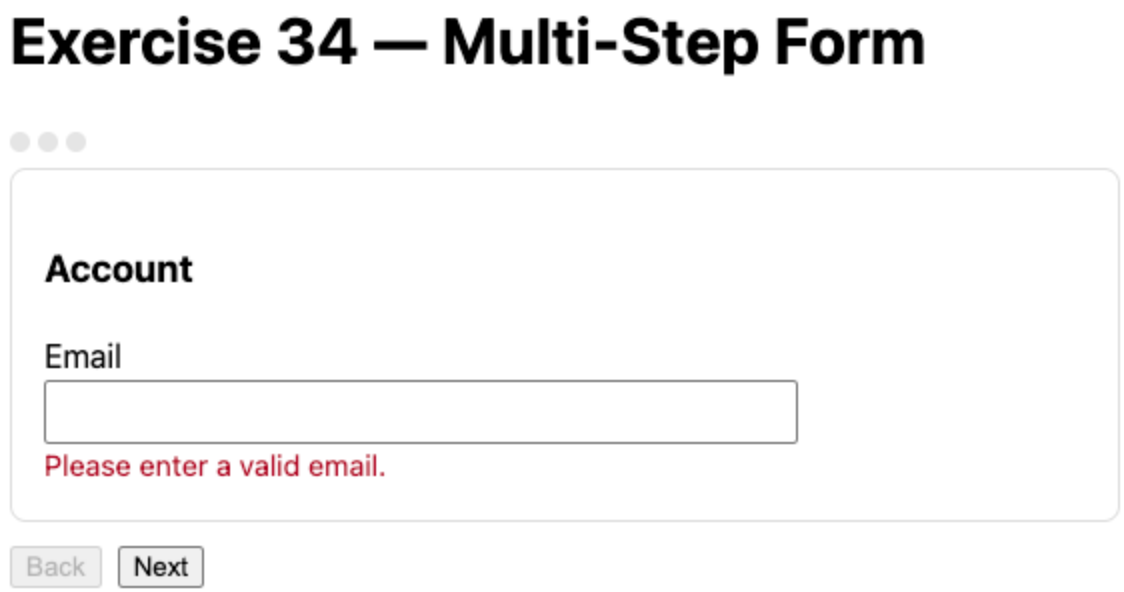
- Wire up dragstart, dragover, drop, dragend.
- Compute the **insertion point** using cursor Y vs element midline.
- Use a **placeholder** element to indicate drop position.

Steps:

1. Grab an item by the ☰ **handle** and drag.
2. The dashed placeholder shows the drop target.
3. Release to drop; item reorders in the DOM.
4. Read the helper getAfterElement to understand the midline math.

---

34) Multi-Step Form Wizard (with Validation)



Build: 3-step form (Account → Profile → Confirm) with client-side validation and progress bullets.

Objectives:

- Manage step state and UI (previous/next navigation).
- Validate per-step inputs and show inline errors.
- Aggregate a **summary** before final submission; handle submit.

Steps:

1. Fill **Email** (must include @) → click **Next**.
2. Fill **Display name** (min 2 chars) → **Next**.
3. Review **summary**, click **Submit** → see a success alert.
4. Navigate back/forth to see state updates and bullet indicators.

---

35) Accessible Tooltip (hover + focus + positioning)

Build: A single, reusable tooltip that appears near hovered/focused controls (hover and keyboard focus supported).

Objectives:

- Use aria-describedby + a single role="tooltip" element.
- Compute **inline positioning** with getBoundingClientRect.
- Handle **hover** and **focus** states; hide on mouseout/blur.

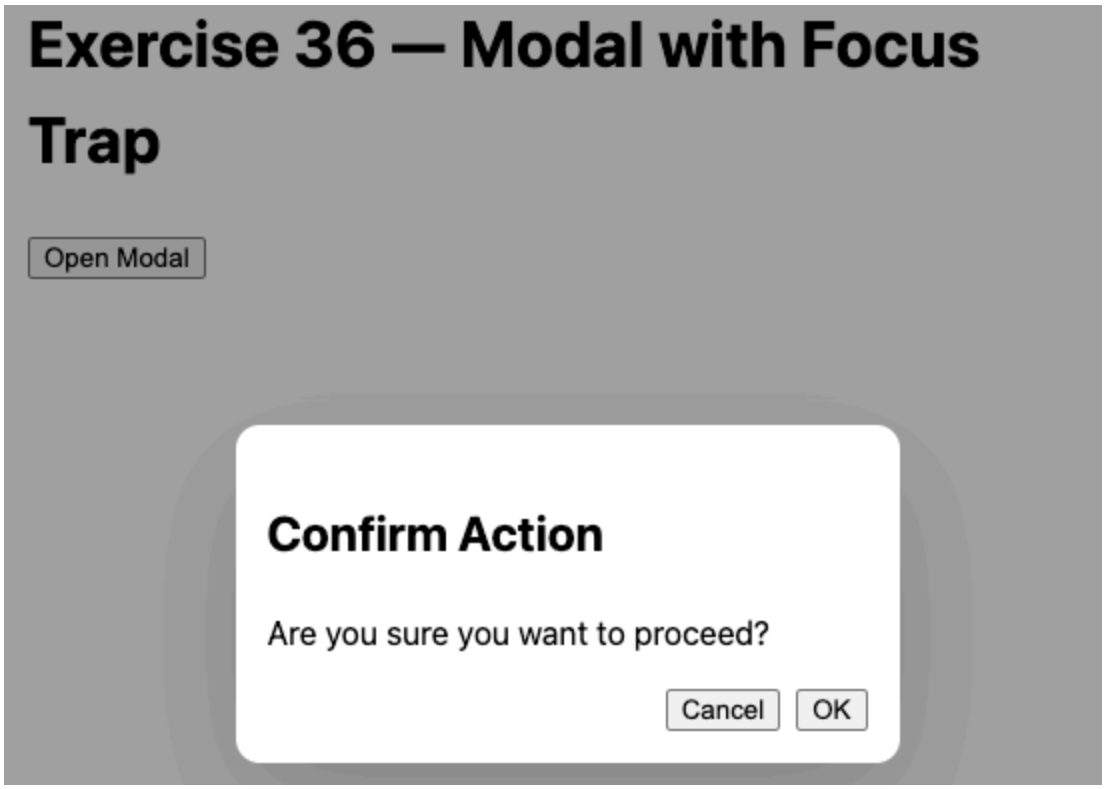
Steps:

1. Hover or **Tab** to a button to show the tooltip.

2. Observe tooltip follows that control’s position.
3. Shift focus or move mouse away to hide it.
4. Inspect JS for how the same tooltip instance is reused.

---

36) Modal Dialog with Focus Trap



Build: An accessible modal with overlay, focus trapping, Esc to close, click-outside to dismiss, and return focus to opener.

Objectives:

- Toggle aria-modal, trap focus with a keydown handler for Tab cycling.
- Support **Escape** and overlay click for dismissal.
- Restore focus to the **trigger** after close.

Steps:

1. Click **Open Modal**; focus moves inside the dialog.
2. Tab forward/backward – focus stays within modal controls.
3. Click **Cancel** / **OK**, press **Esc**, or click the overlay to close.
4. Focus returns to the **Open Modal** button.

---

37) contenteditable Notes with Undo/Redo

Build: A simple rich-text note area that records a debounced history of changes, with Undo/Redo controls.

Objectives:

- Manage a **history stack** of HTML snapshots.
- Debounce input events to avoid excessive snapshots.
- Restore selection/caret at the end after undo/redo.

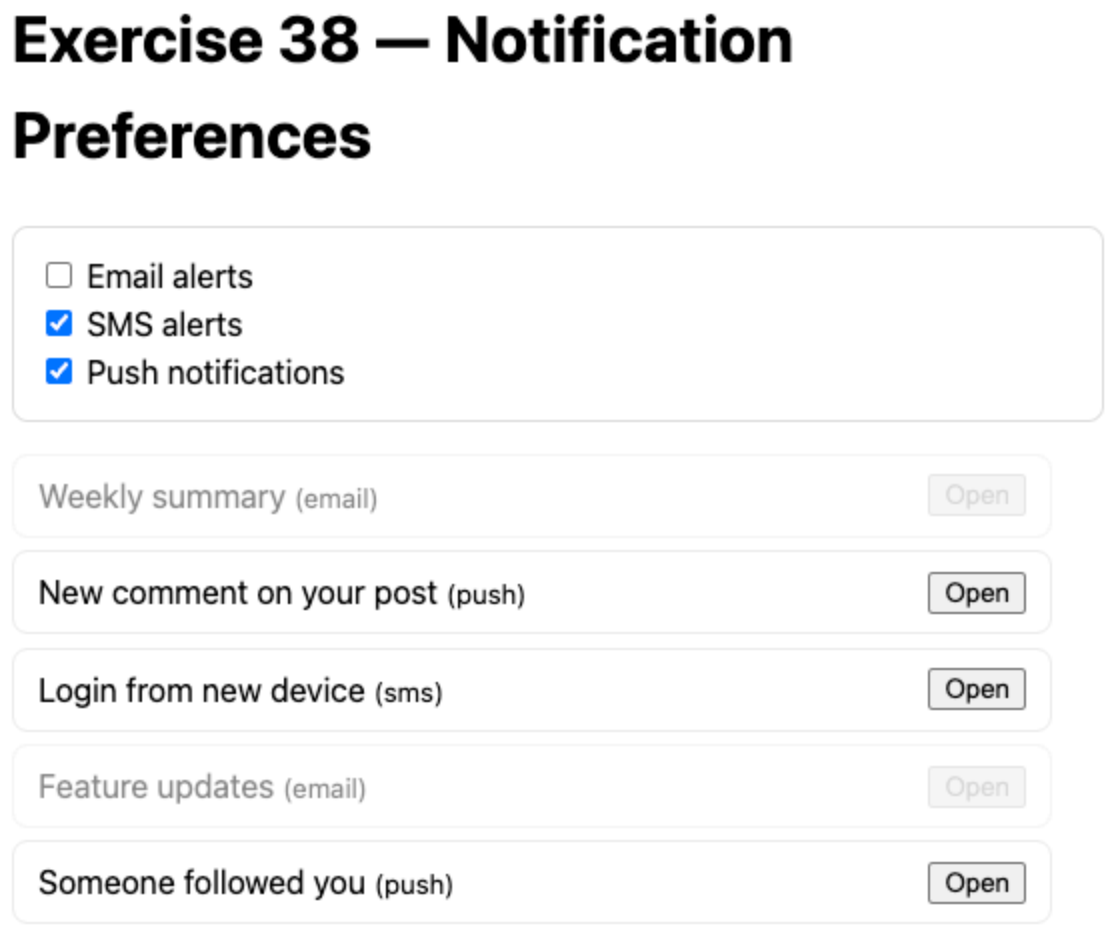
Steps:

1. Type in the **note** area; pause to create snapshots.

2. Click **Undo** to step back through edits; **Redo** to go forward.
3. Notice buttons enable/disable based on stack position.
4. Inspect how selection/caret is restored after applying a snapshot.

---

38) Notification Center (Delegation + localStorage)



Build: Preference checkboxes control which notification items are "enabled"; state persists via localStorage.

Objectives:

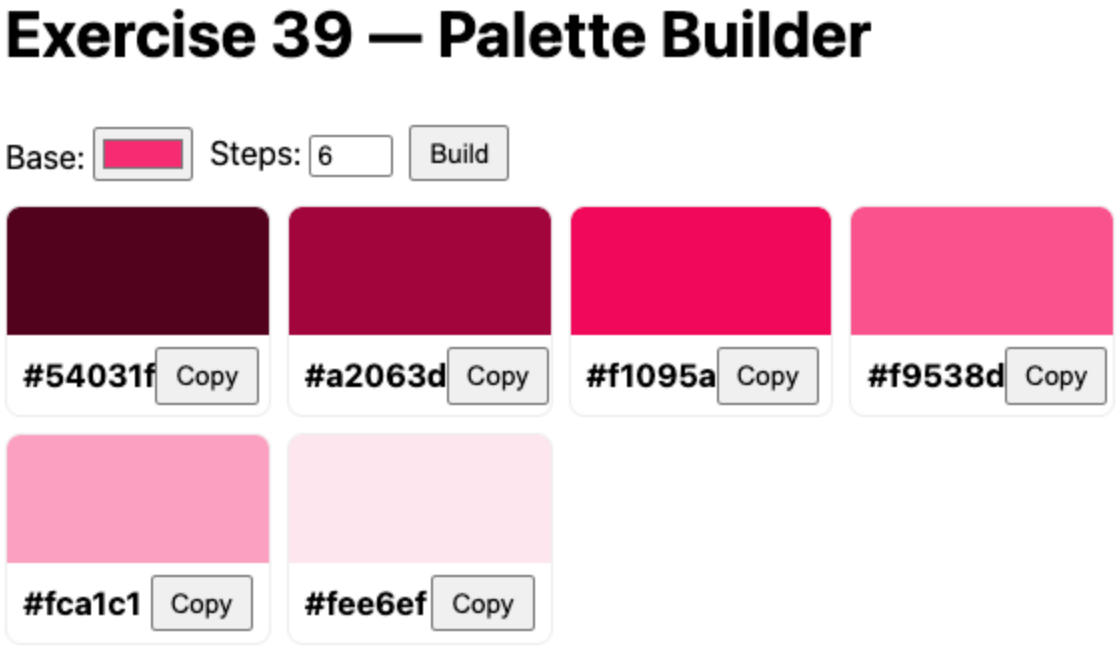
- Use **event delegation** for checkbox and action button handling.

- Read/write persistent preferences with localStorage.
- Render UI based on state (disable buttons and dim items when muted).
  Steps:

1. Toggle **Email/SMS/Push** checkboxes.
2. See the list re-render: some items dim or buttons disable.
3. Refresh the page—your preferences are remembered.
4. Click **Open** on an enabled item to simulate handling it.

---

39) Palette Builder (Color Picker + Copy)



Build: Generate a tonal palette from a base color; copy swatch HEX values to clipboard.
Objectives:

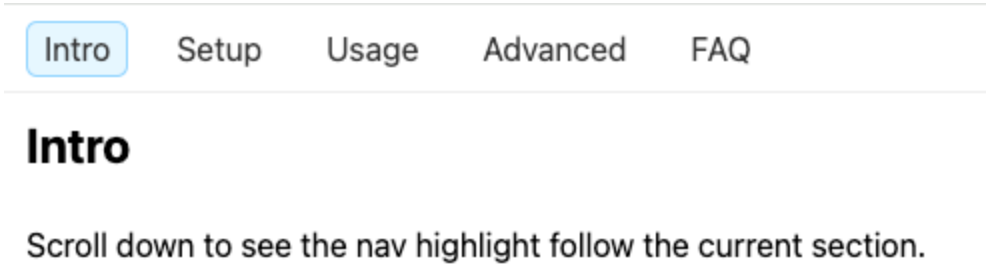
- Convert between **HEX** ↔ **HSL** and compute stepped lightness values.

- Create swatch cards dynamically.
- Use the **Clipboard API** to copy text with user feedback.
  Steps:

1. Choose a **Base** color and **Steps**; click **Build**.
2. Click **Copy** on any swatch—button briefly shows “Copied!”.
3. Adjust steps (3–10) to see different tonal ranges.
4. Review color math functions for HEX/HSL conversions.

---

40) Scroll-Spy Navigation (IntersectionObserver)



Build: Sticky top navigation that highlights the link for the section most visible in the viewport.
Objectives:

- Observe sections and compute **visibility ratio** to pick the active one.
- Apply/remove an active class on the corresponding nav link.
- Use rootMargin + multiple thresholds for smooth behavior.
  Steps:

1. Scroll the page; the **active** link updates as sections change.
2. Click a nav link to jump to that section; spy keeps in sync.

3. Experiment with thresholds/rootMargin to adjust sensitivity.
4. Inspect mapping from section.id → <a href="#id">.

Source code

31) MutationObserver: Live DOM Change Log — exercise31_mutation_observer.html

```
<!doctype html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>MutationObserver — Live DOM Change Log</title>
<style>
  body{font:16px/1.5 system-ui,sans-serif;padding:1rem;display:grid;gap:1rem;grid-template-columns:1fr 1fr}
  .col{border:1px solid #ddd;border-radius:.5rem;padding:1rem}
  .controls button{margin-right:.5rem}
  .target{border:2px dashed #91d5ff;border-radius:.5rem;padding:1rem;min-height:120px}
  pre{background:#f6f8fa;border:1px solid #eee;border-radius:.5rem;padding:.75rem;max-height:360px;overflow:auto}
  .log-entry{margin:0 0 .25rem}
</style>
</head>
<body>
  <div class="col">
    <h1>Exercise 31 — MutationObserver</h1>
```

```
    <p><strong>Objectives:</strong> Detect added/removed nodes and attribute changes in real time.</p>
    <div class="controls">
      <button id="add">Add box</button>
      <button id="remove">Remove last</button>
      <button id="text">Change text</button>
      <button id="attr">Toggle data-flag</button>
      <button id="clear">Clear target</button>
    </div>
    <div id="target" class="target" aria-live="polite">
      <div class="box">Initial node</div>
    </div>
  </div>
  <div class="col">
    <h2>Change Log</h2>
    <pre id="log"></pre>
  </div>

<script>
const target = document.getElementById('target');
const log = document.getElementById('log');

function write(message){
  log.textContent += message + "\n";
  log.scrollTop = log.scrollHeight;
}

const observer = new MutationObserver(mutations => {
  for (const m of mutations){
    if (m.type === 'childList'){
      m.addedNodes.forEach(n => write(`+ added
```

```
<${n.nodeName.toLowerCase()}> "${n.textContent.trim()}"`));
      m.removedNodes.forEach(n => write(`- removed
<${n.nodeName.toLowerCase()}> "${n.textContent.trim()}"`));
    } else if (m.type === 'attributes'){
      write(`* attribute changed: ${m.attributeName} ->
${target.getAttribute(m.attributeName)}`);
    } else if (m.type === 'subtree' || m.type === 'characterData'){
      write(`~ character data changed`);
    }
  }
});

observer.observe(target, { childList: true, attributes: true, subtree: true,
characterData: true });

document.getElementById('add').onclick = () => {
  const d = document.createElement('div');
  d.className = 'box';
  d.textContent = 'Box ' + (target.children.length + 1);
  target.appendChild(d);
};
document.getElementById('remove').onclick = () =>
target.lastElementChild?.remove();
document.getElementById('text').onclick = () => target.firstElementChild &&
(target.firstElementChild.textContent += ' •');
document.getElementById('attr').onclick = () => {
  const cur = target.getAttribute('data-flag');
  target.setAttribute('data-flag', cur === 'on' ? 'off' : 'on');
};
document.getElementById('clear').onclick = () => target.innerHTML = '';
</script>
</body>
```

```
</html>
```

---

32) IntersectionObserver: Lazy Images + Sentinels — exercise32_intersection_observer_lazy.html

```
<!doctype html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>IntersectionObserver — Lazy Images</title>
<style>
  body{font:16px/1.5 system-ui,sans-serif;margin:0}
  header{position:sticky;top:0;background:#111;color:#fff;padding:.5rem 1rem;z-index:2}

.grid{display:grid;grid-template-columns:repeat(auto-fill,minmax(220px,1fr));gap:.75rem;padding:1rem}
  .card{border:1px solid #ddd;border-radius:.5rem;overflow:hidden;background:#fff}
  .card img{display:block;width:100%;height:140px;object-fit:cover;background:#f1f1f1}
  .status{font-size:.9rem;opacity:.75}
  .sentinel{height:40px}
</style>
</head>
<body>
<header>
  <strong>Exercise 32 — IntersectionObserver Lazy Loading</strong>
  <span class="status" id="status">Images loaded: 0</span>
</header>
```

```
<div class="grid" id="grid"></div>
<div class="sentinel" id="sentinel"></div>

<script>
const grid = document.getElementById('grid');
const status = document.getElementById('status');
const sentinel = document.getElementById('sentinel');

let loaded = 0;
const TOTAL = 60;
function makeCard(i){
  const card = document.createElement('div');
  card.className = 'card';
  const img = document.createElement('img');
  img.alt = 'Placeholder ' + i;
  img.dataset.src = `https://picsum.photos/seed/dom${i}/600/400`;
  img.src = 'data:image/svg+xml,<svg xmlns="http://www.w3.org/2000/svg"
width="600" height="400"/>'; // tiny placeholder
  const cap = document.createElement('div');
  cap.style.padding = '.5rem .75rem';
  cap.innerHTML = `<strong>Photo #${i}</strong>`;
  card.append(img, cap);
  return card;
}

// initial batch
for (let i=1;i<=TOTAL;i++) grid.appendChild(makeCard(i));

const io = new IntersectionObserver(entries => {
  for (const e of entries){
    if (e.isIntersecting){
```

```
      const img = e.target;
      img.src = img.dataset.src;
      img.removeAttribute('data-src');
      loaded++;
      status.textContent = `Images loaded: ${loaded}`;
      io.unobserve(img);
    }
  }
}, { rootMargin: '200px 0px' });

document.querySelectorAll('img[data-src]').forEach(img => io.observe(img));

// Optional sentinel to simulate infinite list (no new data here, just status)
const so = new IntersectionObserver((entries) => {
  if (entries.some(e => e.isIntersecting)){
    status.textContent = `Images loaded: ${loaded} (end reached)`;
  }
}, {rootMargin:'200px'});
so.observe(sentinel);
</script>
</body>
</html>
```

---

33) Drag & Drop Sortable List — exercise33_sortable_list.html

```
<!doctype html>
<html lang="en">
<head>
```

```
<meta charset="utf-8">
<title>Sortable List — HTML5 Drag & Drop</title>
<style>
  body{font:16px/1.5 system-ui,sans-serif;padding:1.5rem}
  ul{list-style:none;padding:0;max-width:520px}
  li{border:1px solid #ddd;border-radius:.5rem;padding:.5rem .75rem;margin:.4rem 0;background:#fff;display:flex;align-items:center;gap:.5rem}
  .handle{cursor:grab;user-select:none}
  .placeholder{border:2px dashed #91d5ff;height:2.2rem;border-radius:.5rem;margin:.4rem 0}
</style>
</head>
<body>
<h1>Exercise 33 — Sortable (Drag & Drop)</h1>
<p>Drag items by the ☰ handle to reorder.</p>
<ul id="list">
  <li draggable="true"><span class="handle">☰</span> Vanilla JS</li>
  <li draggable="true"><span class="handle">☰</span> DOM APIs</li>
  <li draggable="true"><span class="handle">☰</span> Accessibility</li>
  <li draggable="true"><span class="handle">☰</span> Performance</li>
  <li draggable="true"><span class="handle">☰</span> Testing</li>
</ul>

<script>
const list = document.getElementById('list');
let dragging = null;
let placeholder = document.createElement('li'); placeholder.className = 'placeholder';

list.addEventListener('dragstart', e => {
  if (!e.target.matches('li')) return;
```

```
  dragging = e.target;
  e.dataTransfer.effectAllowed = 'move';
  e.dataTransfer.setData('text/plain', dragging.textContent.trim());
  setTimeout(()=> dragging.style.opacity = '.3'); // visual
});
list.addEventListener('dragend', () => {
  dragging && (dragging.style.opacity = '');
  placeholder.remove();
  dragging = null;
});
list.addEventListener('dragover', e => {
  e.preventDefault();
  const after = getAfterElement(list, e.clientY);
  if (after == null) list.appendChild(placeholder);
  else list.insertBefore(placeholder, after);
});
list.addEventListener('drop', e => {
  e.preventDefault();
  if (!dragging) return;
  list.insertBefore(dragging, placeholder);
});

function getAfterElement(container, y) {
  const els = [...container.querySelectorAll('li:not(.placeholder)')];
  let nearest = null, offset = Number.NEGATIVE_INFINITY;
  for (const el of els){
    const rect = el.getBoundingClientRect();
    const dy = y - rect.top - rect.height / 2;
    if (dy < 0 && dy > offset){ offset = dy; nearest = el; }
  }
```

```
  return nearest;
}
</script>
</body>
</html>
```

---

34) Multi-Step Form Wizard (with Validation) — exercise34_form_wizard.html

```
<!doctype html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Form Wizard — Steps & Validation</title>
<style>
  body{font:16px/1.5 system-ui,sans-serif;padding:1rem}
  .step{display:none;border:1px solid
#ddd;border-radius:.5rem;padding:1rem;margin-bottom:.75rem}
  .step.active{display:block}
  .nav{display:flex;gap:.5rem}
  input{padding:.4rem;width:100%;max-width:360px}
  .error{color:#b00020;font-size:.9rem}
  .bullets{display:flex;gap:.25rem;margin-bottom:.5rem}
  .bullets span{width:10px;height:10px;border-radius:50%;background:#e5e5e5}
  .bullets span.on{background:#5ac8fa}
</style>
</head>
<body>
<h1>Exercise 34 — Multi-Step Form</h1>
<div class="bullets"
```

```
id="bullets"><span></span><span></span><span></span></div>

<form id="form" novalidate>
  <div class="step active" data-step="0">
    <h3>Account</h3>
    <label>Email<br><input id="email" type="email" required></label>
    <div class="error" id="e1"></div>
  </div>
  <div class="step" data-step="1">
    <h3>Profile</h3>
    <label>Display name<br><input id="name" required></label>
    <div class="error" id="e2"></div>
  </div>
  <div class="step" data-step="2">
    <h3>Confirm</h3>
    <p id="summary"></p>
    <button type="submit">Submit</button>
  </div>
</form>

<div class="nav">
  <button id="prev" disabled>Back</button>
  <button id="next">Next</button>
</div>

<script>
let step = 0;
const steps = [...document.querySelectorAll('.step')];
const bullets = document.getElementById('bullets').children;
```

```
function show(i){
  steps.forEach(s=>s.classList.remove('active'));
  steps[i].classList.add('active');
  [...bullets].forEach((b,idx)=> b.classList.toggle('on', idx<=i));
  document.getElementById('prev').disabled = i===0;
  document.getElementById('next').textContent = i===steps.length-1 ? 'Finish' :
'Next';
}
function validate(i){
  if (i===0){
    const email = document.getElementById('email');
    const ok = email.value.includes('@');
    document.getElementById('e1').textContent = ok ? '' : 'Please enter a valid
email.';
    return ok;
  } else if (i===1){
    const name = document.getElementById('name');
    const ok = name.value.trim().length >= 2;
    document.getElementById('e2').textContent = ok ? '' : 'Name must be at least 2
chars.';
    return ok;
  }
  return true;
}
document.getElementById('next').onclick = () => {
  if (!validate(step)) return;
  if (step < steps.length-1){
    step++;
    if (step===2){
      document.getElementById('summary').textContent = `Email: ${email.value}
— Name: ${name.value}`;
    }
```

```
    show(step);
  } else {
    document.getElementById('form').requestSubmit();
  }
};
document.getElementById('prev').onclick = () => { step = Math.max(0, step-1); show(step); };
document.getElementById('form').addEventListener('submit', e => {
  e.preventDefault();
  alert('Submitted!\n' + JSON.stringify({email: email.value, name: name.value}, null, 2));
});
</script>
</body>
</html>
```

---

35) Accessible Tooltip (hover + focus + positioning) — exercise35_tooltip.html

```
<!doctype html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Accessible Tooltip</title>
<style>
  body{font:16px/1.5 system-ui,sans-serif;padding:1.5rem}
  button{padding:.5rem .75rem}

.tip{position:fixed;transform:translate(-50%,-100%);background:#111;color:#fff;padding:.35rem
.5rem;border-radius:.35rem;white-space:nowrap;pointer-events:none;opacity:0;tr
```

```
ansition:opacity .12s}
  .tip.show{opacity:1}
</style>
</head>
<body>
<h1>Exercise 35 — Tooltip</h1>
<p>Focus or hover the buttons to show tooltips.</p>
<button aria-describedby="t1" data-tip="Create a new document">New</button>
<button aria-describedby="t1" data-tip="Upload files here">Upload</button>
<button aria-describedby="t1" data-tip="Open settings">Settings</button>
<div id="t1" role="tooltip" class="tip" aria-hidden="true"></div>

<script>
const tip = document.getElementById('t1');
function show(el){
  tip.textContent = el.dataset.tip;
  const r = el.getBoundingClientRect();
  tip.style.left = (r.left + r.width/2) + 'px';
  tip.style.top = (r.top - 6) + 'px';
  tip.classList.add('show'); tip.setAttribute('aria-hidden','false');
}
function hide(){ tip.classList.remove('show'); tip.setAttribute('aria-hidden','true'); }

document.addEventListener('mouseover', e => {
  const t = e.target.closest('[data-tip]'); if(!t) return hide();
  show(t);
});
document.addEventListener('focusin', e => { const t =
e.target.closest('[data-tip]'); if(t) show(t); });
document.addEventListener('mouseout', e => { if (!e.relatedTarget ||
!e.relatedTarget.closest('[data-tip]')) hide(); });
```

```
document.addEventListener('focusout', hide);
</script>
</body>
</html>
```

---

36) Modal Dialog with Focus Trap — exercise36_modal_focus_trap.html

```
<!doctype html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Modal Dialog — Focus Trap</title>
<style>
  body{font:16px/1.5 system-ui,sans-serif;padding:1.5rem}

.overlay{position:fixed;inset:0;background:rgba(0,0,0,.4);display:none;align-items:center;justify-content:center}
  .overlay.show{display:flex}

.dialog{background:#fff;border-radius:.75rem;padding:1rem;min-width:300px;box-shadow:0 20px 60px rgba(0,0,0,.2)}
  .row{display:flex;gap:.5rem;justify-content:flex-end;margin-top:.75rem}
</style>
</head>
<body>
<h1>Exercise 36 — Modal with Focus Trap</h1>
<button id="open">Open Modal</button>

<div id="overlay" class="overlay" role="dialog" aria-modal="true" aria-labelledby="title">
```

```
  <div class="dialog">
    <h2 id="title">Confirm Action</h2>
    <p>Are you sure you want to proceed?</p>
    <div class="row">
      <button id="cancel">Cancel</button>
      <button id="ok">OK</button>
    </div>
  </div>
</div>

<script>
const overlay = document.getElementById('overlay');
const openBtn = document.getElementById('open');
const cancelBtn = document.getElementById('cancel');
const okBtn = document.getElementById('ok');
let lastFocus = null;

function focusTrap(e){
  if (!overlay.classList.contains('show')) return;
  const focusables = overlay.querySelectorAll('button, [href], input, select, textarea,
[tabindex]:not([tabindex="-1"])');
  const first = focusables[0], last = focusables[focusables.length-1];
  if (e.key === 'Tab'){
    if (e.shiftKey && document.activeElement === first){ last.focus();
e.preventDefault(); }
    else if (!e.shiftKey && document.activeElement === last){ first.focus();
e.preventDefault(); }
  } else if (e.key === 'Escape'){ close(); }
}

function open(){
```

```
  lastFocus = document.activeElement;
  overlay.classList.add('show');
  cancelBtn.focus();
  document.addEventListener('keydown', focusTrap);
}
function close(){
  overlay.classList.remove('show');
  document.removeEventListener('keydown', focusTrap);
  lastFocus?.focus();
}

openBtn.onclick = open;
cancelBtn.onclick = close;
okBtn.onclick = () => { alert('Confirmed!'); close(); };
overlay.addEventListener('click', (e)=>{ if (e.target === overlay) close(); });
</script>
</body>
</html>
```

---

37) contenteditable Notes with Undo/Redo — exercise37_contenteditable_undo.html

```
<!doctype html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>contenteditable — Undo/Redo Stack</title>
<style>
  body{font:16px/1.5 system-ui,sans-serif;padding:1rem}
```

```
  .toolbar{display:flex;gap:.5rem;margin-bottom:.5rem}
  .note{border:1px solid
#ddd;border-radius:.5rem;padding:.75rem;min-height:120px}
  .hint{color:#666}
</style>
</head>
<body>
<h1>Exercise 37 — Notes with Undo/Redo</h1>
<div class="toolbar">
  <button id="undo" disabled>Undo</button>
  <button id="redo" disabled>Redo</button>
</div>
<div id="note" class="note" contenteditable="true">Type here…</div>
<p class="hint">This implements a tiny history: snapshots on input with
debounce.</p>

<script>
const note = document.getElementById('note');
const undoBtn = document.getElementById('undo');
const redoBtn = document.getElementById('redo');

let history = [note.innerHTML];
let idx = 0;
let t = null;

function updateButtons(){
  undoBtn.disabled = idx === 0;
  redoBtn.disabled = idx >= history.length - 1;
}
function snapshot(){
  const val = note.innerHTML;
```

```
  if (history[idx] !== val){
    history = history.slice(0, idx+1);
    history.push(val);
    idx++;
    updateButtons();
  }
}

note.addEventListener('input', () => {
  clearTimeout(t);
  t = setTimeout(snapshot, 250); // debounce
});

undoBtn.onclick = () => {
  if (idx > 0){ idx--; note.innerHTML = history[idx]; placeCaretEnd(note);
updateButtons(); }
};
redoBtn.onclick = () => {
  if (idx < history.length-1){ idx++; note.innerHTML = history[idx];
placeCaretEnd(note); updateButtons(); }
};

function placeCaretEnd(el){
  el.focus();
  const range = document.createRange();
  range.selectNodeContents(el);
  range.collapse(false);
  const sel = window.getSelection();
  sel.removeAllRanges();
  sel.addRange(range);
}
```

```
updateButtons();
</script>
</body>
</html>
```

---

38) Notification Center (Delegation + localStorage) — exercise38_notifications_localstorage.html

```
<!doctype html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Notifications — Delegation & localStorage</title>
<style>
  body{font:16px/1.5 system-ui,sans-serif;padding:1rem}
  .prefs{border:1px solid #ddd;border-radius:.5rem;padding:.75rem;max-width:520px;margin-bottom:1rem}
  .list{max-width:520px}
  .item{border:1px solid #eee;border-radius:.5rem;padding:.5rem .75rem;margin:.4rem 0;display:flex;justify-content:space-between;align-items:center}
  .muted{opacity:.5}
</style>
</head>
<body>
<h1>Exercise 38 — Notification Preferences</h1>
<div class="prefs" id="prefs">
  <label><input type="checkbox" data-key="email" checked> Email alerts</label><br>
```

```
  <label><input type="checkbox" data-key="sms"> SMS alerts</label><br>
  <label><input type="checkbox" data-key="push" checked> Push notifications</label>
</div>

<div class="list" id="list"></div>

<script>
const prefs = document.getElementById('prefs');
const list = document.getElementById('list');
const items = [
  {id:1, type:'email', text:'Weekly summary'},
  {id:2, type:'push', text:'New comment on your post'},
  {id:3, type:'sms', text:'Login from new device'},
  {id:4, type:'email', text:'Feature updates'},
  {id:5, type:'push', text:'Someone followed you'}
];

function load(){
  try{ return JSON.parse(localStorage.getItem('notif-prefs')) ?? {email:true,sms:false,push:true}; }
  catch{ return {email:true,sms:false,push:true}; }
}
function save(state){ localStorage.setItem('notif-prefs', JSON.stringify(state)); }
let state = load();

// init checkboxes
[...prefs.querySelectorAll('input[type=checkbox]')].forEach(cb => {
  cb.checked = !!state[cb.dataset.key];
});
```

```
function render(){
  list.innerHTML = '';
  items.forEach(it => {
    const div = document.createElement('div');
    const enabled = !!state[it.type];
    div.className = 'item' + (enabled ? '' : ' muted');
    div.innerHTML = `<span>${it.text} <small>(${it.type})</small></span>
                  <button data-id="${it.id}" data-type="${it.type}"
${enabled?'':'disabled'}>Open</button>`;
    list.appendChild(div);
  });
}
render();

prefs.addEventListener('change', (e) => {
  if (!e.target.matches('input[type=checkbox]')) return;
  state[e.target.dataset.key] = e.target.checked;
  save(state);
  render();
});

list.addEventListener('click', (e) => {
  const btn = e.target.closest('button[data-id]'); if(!btn) return;
  alert('Opened: ' + items.find(i=>i.id==btn.dataset.id).text);
});
</script>
</body>
</html>
```

39) Palette Builder (Color Picker + Copy) — exercise39_palette_builder.html

```
<!doctype html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Palette Builder — Copy Hex</title>
<style>
  body{font:16px/1.5 system-ui,sans-serif;padding:1rem}

.controls{display:flex;gap:.5rem;align-items:center;margin-bottom:.75rem;flex-wrap:wrap}

.swatches{display:grid;grid-template-columns:repeat(auto-fill,minmax(120px,1fr));gap:.5rem}
  .swatch{border-radius:.5rem;overflow:hidden;border:1px solid #eee}
  .swatch .box{height:64px}
  .swatch footer{display:flex;justify-content:space-between;align-items:center;padding:.35rem .5rem}
  button{padding:.3rem .5rem}
</style>
</head>
<body>
<h1>Exercise 39 — Palette Builder</h1>
<div class="controls">
  <label>Base: <input type="color" id="base" value="#2d7ef7"></label>
  <label>Steps: <input type="number" id="steps" min="3" max="10" value="6"></label>
  <button id="build">Build</button>
</div>
```

```
<div class="swatches" id="swatches"></div>


<script>
function hexToHsl(hex){
  const m = hex.match(/^#?([a-f\d]{2})([a-f\d]{2})([a-f\d]{2})$/i);
  if(!m) return [0,0,0];
  let r = parseInt(m[1],16)/255, g = parseInt(m[2],16)/255, b =
parseInt(m[3],16)/255;
  const max = Math.max(r,g,b), min = Math.min(r,g,b);
  let h, s, l = (max + min) / 2;
  if(max === min){ h = s = 0; }
  else {
    const d = max - min;
    s = l > .5 ? d / (2 - max - min) : d / (max + min);
    switch(max){
      case r: h = (g - b) / d + (g < b ? 6 : 0); break;
      case g: h = (b - r) / d + 2; break;
      case b: h = (r - g) / d + 4; break;
    }
    h /= 6;
  }
  return [Math.round(h*360), Math.round(s*100), Math.round(l*100)];
}
function hslToHex(h,s,l){
  s/=100; l/=100;
  const C = (1-Math.abs(2*l-1))*s;
  const X = C*(1-Math.abs(((h/60)%2)-1));
  const m = l - C/2;
  let r=0,g=0,b=0;
  if (0<=h&&h<60){r=C;g=X;} else if(60<=h&&h<120){r=X;g=C;}
  else if(120<=h&&h<180){g=C;b=X;} else if(180<=h&&h<240){g=X;b=C;}
```

```
  else if(240<=h&&h<300){r=X;b=C;} else {r=C;b=X;}
  const toHex = v => Math.round((v+m)*255).toString(16).padStart(2,'0');
  return '#'+toHex(r)+toHex(g)+toHex(b);
}

const base = document.getElementById('base');
const steps = document.getElementById('steps');
const swatches = document.getElementById('swatches');

function build(){
  swatches.innerHTML = '';
  const [h,s,l] = hexToHsl(base.value);
  for (let i=0;i<Number(steps.value);i++){
    const ll = Math.max(5, Math.min(95, l - 40 + (i*(80/(steps.value-1)))));
    const hex = hslToHex(h, s, ll);
    const div = document.createElement('div');
    div.className = 'swatch';
    div.innerHTML = `<div class="box" style="background:${hex}"></div>
                <footer><strong>${hex}</strong> <button
data-hex="${hex}">Copy</button></footer>`;
    swatches.appendChild(div);
  }
}
build();

document.getElementById('build').onclick = build;
swatches.addEventListener('click', async (e) => {
  const btn = e.target.closest('button[data-hex]'); if(!btn) return;
  await navigator.clipboard.writeText(btn.dataset.hex);
  btn.textContent = 'Copied!';
  setTimeout(()=>btn.textContent='Copy', 900);
```

```
});
</script>
</body>
</html>
```

---

40) Scroll-Spy Navigation (IntersectionObserver) — exercise40_scroll_spy.html

```
<!doctype html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Scroll Spy — Active Section Highlight</title>
<style>
  body{margin:0;font:16px/1.5 system-ui,sans-serif}
  nav{position:sticky;top:0;background:#fff;border-bottom:1px solid
#eee;z-index:2}
  nav ul{display:flex;gap:1rem;list-style:none;margin:0;padding:.6rem 1rem}
  nav a{padding:.25rem
.5rem;border-radius:.35rem;text-decoration:none;color:#333}
  nav a.active{background:#e6f7ff;border:1px solid #91d5ff}
  section{min-height:75vh;padding:2rem 1rem;border-bottom:1px solid #f3f3f3}
</style>
</head>
<body>
<nav>
  <ul>
    <li><a href="#intro" class="spy">Intro</a></li>
    <li><a href="#setup" class="spy">Setup</a></li>
```

```
    <li><a href="#usage" class="spy">Usage</a></li>
    <li><a href="#advanced" class="spy">Advanced</a></li>
    <li><a href="#faq" class="spy">FAQ</a></li>
  </ul>
</nav>

<section id="intro"><h2>Intro</h2><p>Scroll down to see the nav highlight follow the current section.</p></section>
<section id="setup"><h2>Setup</h2><p>Some text here…</p></section>
<section id="usage"><h2>Usage</h2><p>More text here…</p></section>
<section id="advanced"><h2>Advanced</h2><p>Even more text…</p></section>
<section id="faq"><h2>FAQ</h2><p>Last section.</p></section>

<script>
const links = [...document.querySelectorAll('a.spy')];
const map = Object.fromEntries(links.map(a => [a.getAttribute('href').slice(1), a]));

const io = new IntersectionObserver(entries => {
  // choose the most visible entry
  const visible = entries.filter(e => e.isIntersecting)
                         .sort((a,b)=> b.intersectionRatio - a.intersectionRatio)[0];
  if (!visible) return;
  const id = visible.target.id;
  links.forEach(a => a.classList.toggle('active', a === map[id]));
}, { rootMargin: '-30% 0px -60% 0px', threshold: [0, .25, .5, .75, 1] });

document.querySelectorAll('section').forEach(sec => io.observe(sec));
</script>
</body>
```

```
</html>
```