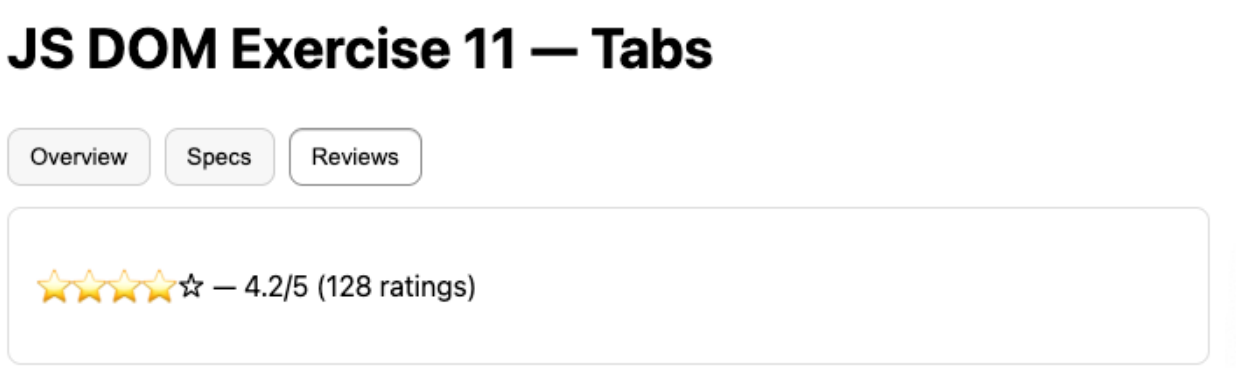
JavaScript DOM Coding Exercises

11) Tabs Component (ARIA)



Build: Keyboard-accessible tab interface with ARIA roles/states and panels.
Objectives:

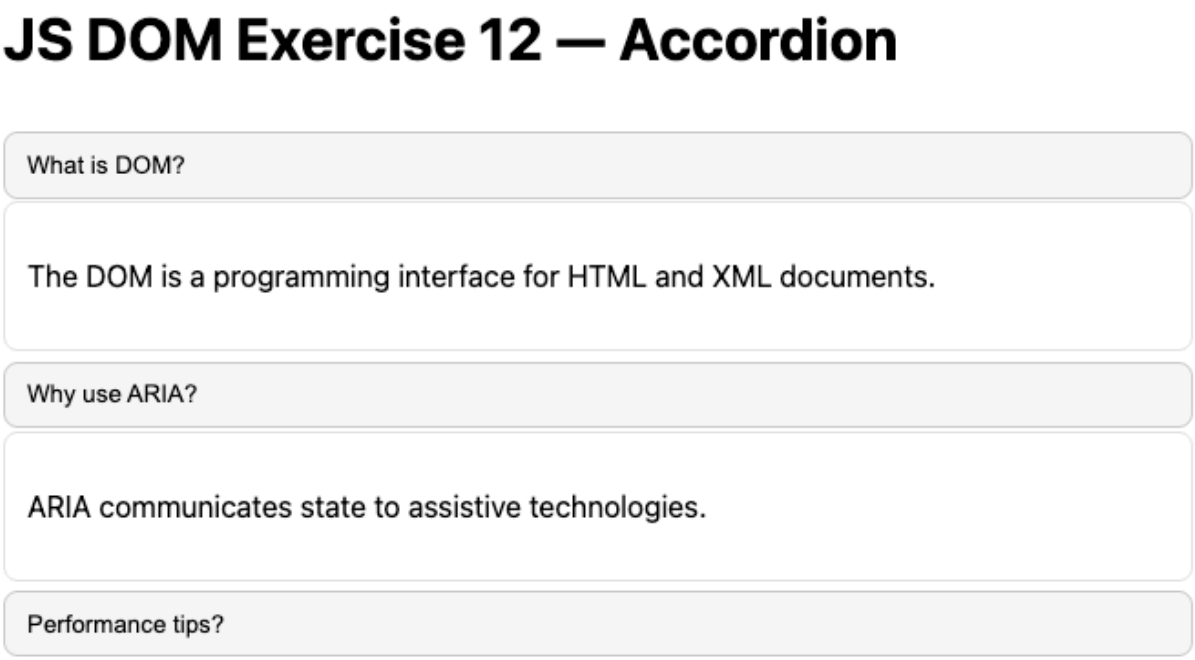
- Use role="tablist", role="tab", role="tabpanel" and aria-selected, aria-controls, aria-labelledby.
- Toggle panels and manage focus with arrow/Home/End keys.
- Keep only the active panel visible (hidden attribute).

Steps:

1. Click a tab or use Left/Right to switch; Home/End jump to first/last.
2. Confirm aria-selected updates on tabs and only one panel is visible.
3. Inspect code: activate(tab) updates ARIA, focus, and panel visibility.

---

12) Accordion (ARIA)



Build: Single-expand/collapse FAQ-style accordion with semantic buttons and ARIA linkage.

Objectives:

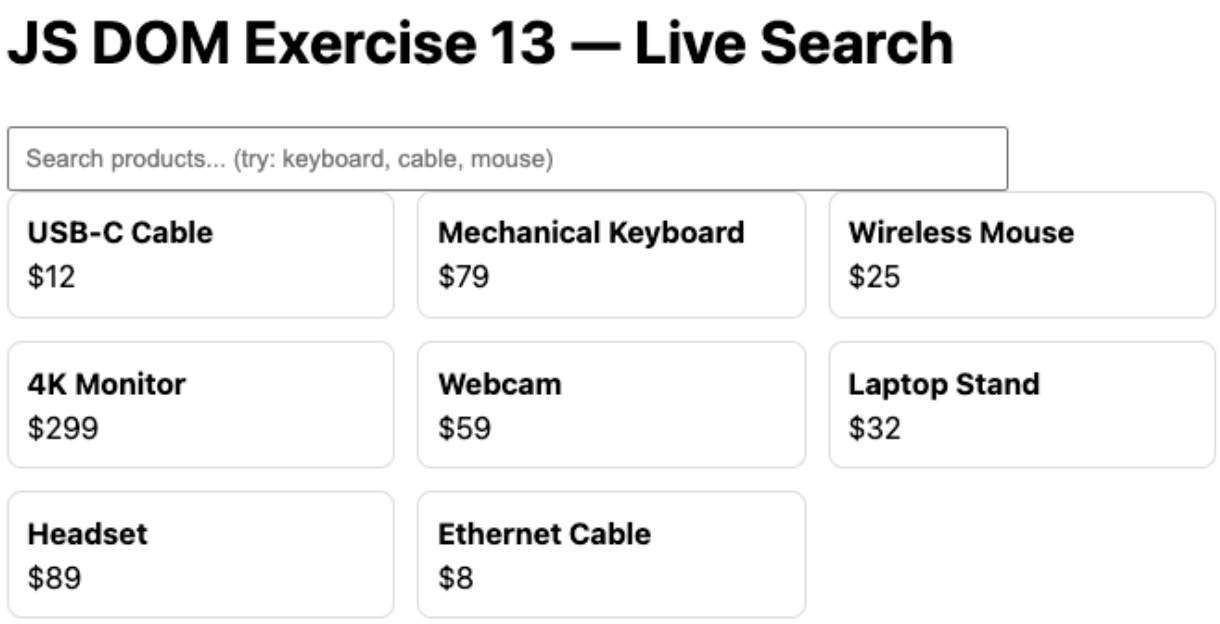
- Control aria-expanded on buttons and hidden on content panels.
- Connect headers and regions via aria-controls / aria-labelledby.
- Use event delegation for compact logic.

Steps:

1. Click a question to toggle its panel.
2. Watch aria-expanded flip and the panel’s hidden update.
3. Review the delegated click handler that reads aria-controls.

---

13) Live Search with Debounce



Build: Product cards filtered by a debounced input; matches highlighted.
Objectives:

- Implement a debounce utility for input events.
- Filter an in-memory array and render a result grid efficiently.
- Safely highlight matches with a RegExp built from user input.

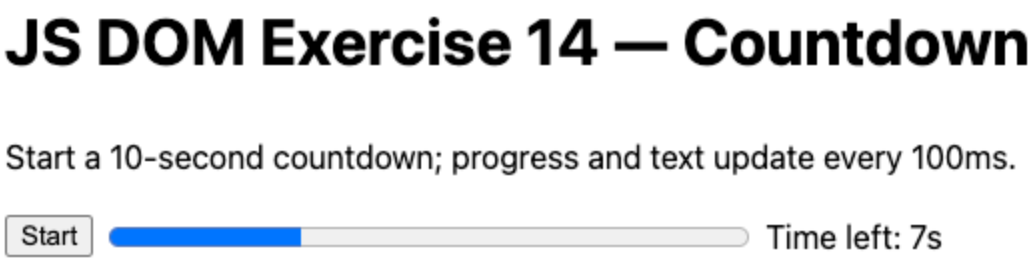
Steps:

1. Type in the search box; results update after ~200ms idle.
2. See <mark> highlighting on matching text.

3. Check the render pipeline and how the regex is escaped.

---

14) Countdown + Progress Bar



Build: 10-second timer that updates a <progress> bar and label every 100ms.
Objectives:

- Use setInterval and derive percentage from elapsed steps.
- Update <progress> and text status; stop cleanly at 100%.
- Reset safely if the user restarts mid-countdown.
  Steps:

1. Click **Start** to begin; watch the percent and seconds remaining.
2. Click **Start** again mid-run to restart.
3. Read code to see how steps → percent and when the interval clears.

---

15) File Input Preview + Validation

JS DOM Exercise 15 — Image Preview

Select an image under 1 MB; preview appears below.

Choose File js2.png

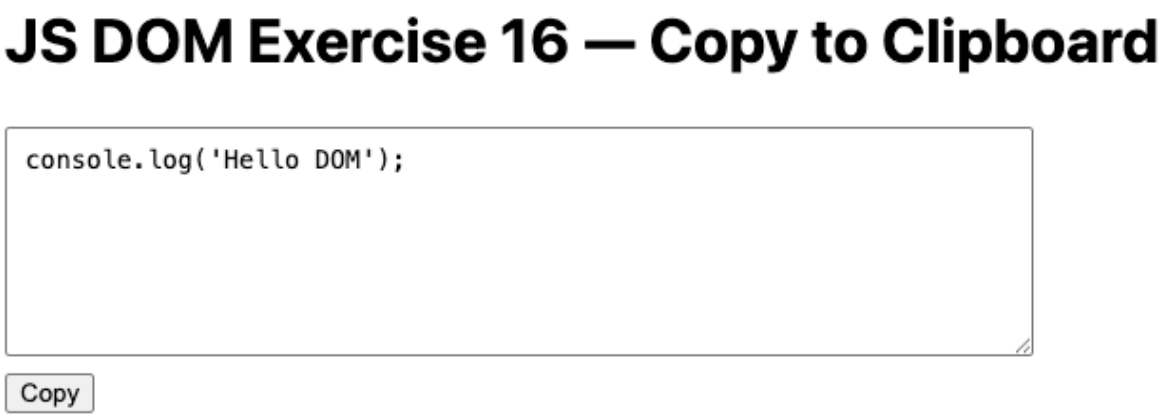
Build: Image file picker with type and size checks, live preview via FileReader.
Objectives:

- Validate type.startsWith('image/') and size (≤ 1 MB).
- Preview images using readAsDataURL.
- Provide clear error messaging and reset state on reselect.
  Steps:

1. Choose an image < 1 MB → preview appears.
2. Try a non-image or large file → see validation message.
3. Inspect the change handler for validation + preview logic.

---

16) Clipboard + Toast



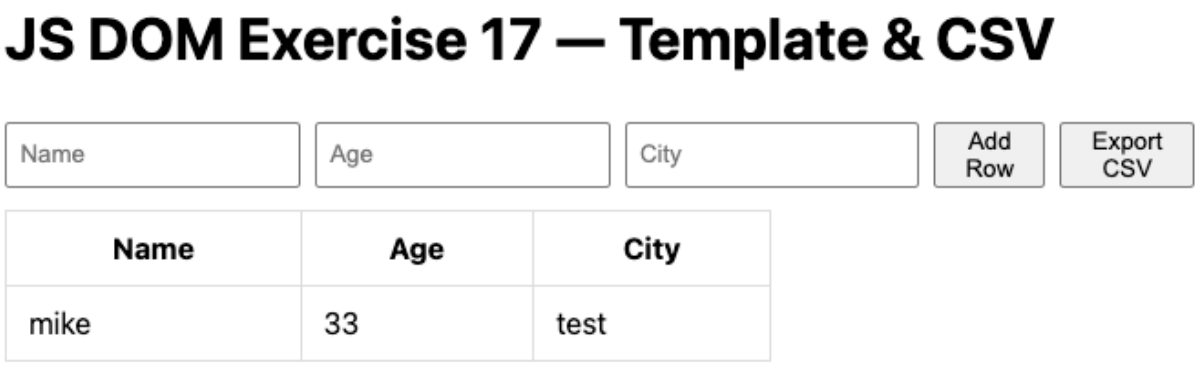
Build: Copy the textarea contents to clipboard with a confirmation toast.
Objectives:

- Use navigator.clipboard.writeText.
- Provide accessible status feedback (role="status" or aria-live).
- Handle failure exceptions gracefully.
  Steps:

1. Click **Copy** → toast shows "Copied to clipboard!".
2. (Optional) Reject permissions to see an error message.
3. Review the toast show/hide timing and try/catch.

---

17) <template> Rows + CSV Export



Build: Add table rows from inputs using a <template>, then export all rows as CSV.
Objectives:

- Clone and fill <template> fragments.
- Traverse the table to collect cell text.
- Generate CSV, escape quotes, and download via Blob + object URL.
  Steps:

1. Enter Name/Age/City → **Add Row**. Repeat a few times.
2. Click **Export CSV** to download people.csv.
3. Open the file to confirm values and proper CSV escaping.

---

18) Hash Router (SPA-like)

JS DOM Exercise 18 — Hash Router

Home About Contact

About

This is a tiny SPA using hashchange.

Build: Tiny single-page navigation using the URL hash to show/hide sections.
Objectives:

- Map routes like #/about → section IDs.
- Listen to hashchange and re-render the active section.
- Provide initial render when page loads (no hash defaults).
  Steps:

1. Click nav links or manually edit the hash.
2. Confirm only the matching section is visible.
3. Inspect the routes map and the render() function.

---

19) Web Component: Star Rating (Shadow DOM)

JS DOM Exercise 19 — Shadow DOM Component

★★★☆☆★★☆☆☆

Build: <star-rating> custom element with 1–5 clickable stars and internal Shadow DOM.

Objectives:

- Define a custom element class and register with customElements.define.
- Use Shadow DOM for encapsulated markup/styles.
- Reflect value via attributes and dispatch a change CustomEvent.

Steps:

1. Click stars to set the rating; stars fill accordingly.
2. Optionally listen for change events from the element.
3. Review observedAttributes, attribute → state sync, and rendering.

---

20) DOMParser + Simple Sanitizer

JS DOM Exercise 20 — Parse HTML Safely

Enter limited HTML. Only , , <code>, and <a> (with href) are allowed.

```
<strong>Hello</strong> <img src=x onerror=alert(1)> <a
href="https://example.com" onclick="evil()">link</a>
```

Render

Hello <img src="x" onerror="alert(1)"> link

Build: Parse user-provided HTML, allow only a safe tag subset, and strip dangerous attributes.

Objectives:

- Use DOMParser to parse strings into a document.
- Walk the DOM and replace disallowed nodes with text.
- For links, permit only href and add rel="noopener" + target="_blank".

Steps:

1. Enter mixed HTML (including disallowed tags/attrs).
2. Click **Render**; preview shows only allowed tags (strong, em, code, a[href]).
3. Inspect the sanitizer to see how attributes and tags are whitelisted.

Source Code

11) exercise11_tabs.html — Tabs Component (ARIA)

```
<!doctype html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Tabs Component (ARIA)</title>
<style>
  body { font:16px/1.5 system-ui, sans-serif; padding:1.5rem; }
  .tabs { max-width: 720px; }
  [role="tablist"] { display:flex; gap:.5rem; margin-bottom:.75rem; }
  [role="tab"] { padding:.5rem .75rem; border:1px solid #ccc;
border-radius:.5rem; background:#f7f7f7; cursor:pointer; }
  [role="tab"][aria-selected="true"] { background:white; border-color:#888;
box-shadow:0 1px 0 #ddd inset; }
  [role="tabpanel"] { border:1px solid #ddd; border-radius:.5rem; padding:1rem;
}
  [hidden] { display:none !important; }
</style>
</head>
<body>
<h1>JS DOM Exercise 11 — Tabs</h1>

<div class="tabs">
  <div role="tablist" aria-label="Sample Tabs">
    <button role="tab" id="tab-a" aria-selected="true" aria-controls="panel-a"
tabindex="0">Overview</button>
    <button role="tab" id="tab-b" aria-selected="false" aria-controls="panel-b"
tabindex="-1">Specs</button>
    <button role="tab" id="tab-c" aria-selected="false" aria-controls="panel-c"
tabindex="-1">Reviews</button>
  </div>
```

```
  <section role="tabpanel" id="panel-a" aria-labelledby="tab-a">
    <p>Overview content. Use left/right arrows to switch tabs; Home/End jump to ends.</p>
  </section>
  <section role="tabpanel" id="panel-b" aria-labelledby="tab-b" hidden>
    <ul><li>Weight: 1.2kg</li><li>Battery: 10h</li><li>Ports: USB-C x2</li></ul>
  </section>
  <section role="tabpanel" id="panel-c" aria-labelledby="tab-c" hidden>
    <p>⭐⭐⭐⭐☆ — 4.2/5 (128 ratings)</p>
  </section>
</div>

<script>
const tabs = document.querySelectorAll('[role="tab"]');
const panels = document.querySelectorAll('[role="tabpanel"]');

function activate(tab) {
  tabs.forEach(t => { t.setAttribute('aria-selected','false'); t.tabIndex = -1; });
  panels.forEach(p => p.hidden = true);
  tab.setAttribute('aria-selected','true'); tab.tabIndex = 0; tab.focus();
  document.getElementById(tab.getAttribute('aria-controls')).hidden = false;
}

tabs.forEach(t => {
  t.addEventListener('click', () => activate(t));
  t.addEventListener('keydown', e => {
    const i = [...tabs].indexOf(t);
    if (e.key === 'ArrowRight') activate(tabs[(i+1)%tabs.length]);
    else if (e.key === 'ArrowLeft') activate(tabs[(i-1+tabs.length)%tabs.length]);
    else if (e.key === 'Home') activate(tabs[0]);
```

```
    else if (e.key === 'End') activate(tabs[tabs.length-1]);
  });
});
</script>
</body>
</html>
```

---

12) exercise12_accordion.html — Accordion (ARIA)

```
<!doctype html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Accordion (ARIA)</title>
<style>
  body { font:16px/1.5 system-ui, sans-serif; padding:1.5rem; }
  .acc button { width:100%; text-align:left; padding:.6rem .75rem; border:1px solid #ccc; border-radius:.5rem; background:#f9f9f9; margin:.35rem 0; }
  .panel { padding:.75rem; border:1px solid #e5e5e5; border-radius:.5rem; margin-top:-.25rem; }
  [hidden]{ display:none!important; }
</style>
</head>
<body>
<h1>JS DOM Exercise 12 — Accordion</h1>

<div class="acc" id="acc">
  <button aria-expanded="false" aria-controls="p1" id="h1">What is DOM?</button>
  <div class="panel" id="p1" role="region" aria-labelledby="h1" hidden>
    <p>The DOM is a programming interface for HTML and XML documents.</p>
```

```
  </div>

  <button aria-expanded="false" aria-controls="p2" id="h2">Why use ARIA?</button>
  <div class="panel" id="p2" role="region" aria-labelledby="h2" hidden>
    <p>ARIA communicates state to assistive technologies.</p>
  </div>

  <button aria-expanded="false" aria-controls="p3" id="h3">Performance tips?</button>
  <div class="panel" id="p3" role="region" aria-labelledby="h3" hidden>
    <ul><li>Batch DOM changes.</li><li>Use delegation.</li><li>Avoid layout thrashing.</li></ul>
  </div>
</div>

<script>
document.getElementById('acc').addEventListener('click', e => {
  if(e.target.matches('button[aria-controls]')){
    const btn = e.target;
    const open = btn.getAttribute('aria-expanded') === 'true';
    btn.setAttribute('aria-expanded', String(!open));
    document.getElementById(btn.getAttribute('aria-controls')).hidden = open;
  }
});
</script>
</body>
</html>
```

---

13) exercise13_live_search.html — Live Search with Debounce

```
<!doctype html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Live Search with Debounce</title>
<style>
  body { font:16px/1.5 system-ui, sans-serif; padding:1.5rem; }
  .grid { display:grid; grid-template-columns:repeat(auto-fill, minmax(180px,1fr)); gap:.75rem; }
  .card { border:1px solid #ddd; border-radius:.5rem; padding:.6rem; }
  input { padding:.5rem; width:100%; max-width:520px; }
  mark { background:#fff2a8; }
</style>
</head>
<body>
<h1>JS DOM Exercise 13 — Live Search</h1>
<input id="q" placeholder="Search products... (try: keyboard, cable, mouse)" autocomplete="off">
<div class="grid" id="grid"></div>

<script>
const data = [
  {name:'USB-C Cable', price:12}, {name:'Mechanical Keyboard', price:79},
  {name:'Wireless Mouse', price:25}, {name:'4K Monitor', price:299},
  {name:'Webcam', price:59}, {name:'Laptop Stand', price:32},
  {name:'Headset', price:89}, {name:'Ethernet Cable', price:8}
];

const grid = document.getElementById('grid');
function render(list, q=''){
  grid.innerHTML = '';
```

```
  const rx = q ? new RegExp('('+q.replace(/[.*+?^${}()|[\\]\\\\]/g,'\\\\$&')+')','i')
: null;
  list.forEach(p => {
    const card = document.createElement('div');
    card.className = 'card';
    const n = rx ? p.name.replace(rx,'<mark>$1</mark>') : p.name;
    card.innerHTML = `<strong>${n}</strong><div>$${p.price}</div>`;
    grid.appendChild(card);
  });
}
render(data);

function debounce(fn, ms){ let t; return (...a)=>{ clearTimeout(t);
t=setTimeout(()=>fn(...a), ms); }; }
const onFilter = debounce(() => {
  const q = document.getElementById('q').value.trim();
  const filtered = q ? data.filter(p =>
p.name.toLowerCase().includes(q.toLowerCase())) : data;
  render(filtered, q);
}, 200);

document.getElementById('q').addEventListener('input', onFilter);
</script>
</body>
</html>
```

---

14) exercise14_countdown_progress.html — Countdown + Progress Bar

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

```
<head>
<meta charset="utf-8">
<title>Countdown + Progress Bar</title>
<style>
  body { font:16px/1.5 system-ui, sans-serif; padding:1.5rem; }
  .row{display:flex; gap:.5rem; align-items:center;}
  progress{width:320px; height:20px;}
</style>
</head>
<body>
<h1>JS DOM Exercise 14 — Countdown</h1>
<p>Start a 10-second countdown; progress and text update every 100ms.</p>
<div class="row">
  <button id="start">Start</button>
  <progress id="bar" value="0" max="100"></progress>
  <span id="label">Idle</span>
</div>

<script>
const start = document.getElementById('start');
const bar = document.getElementById('bar');
const label = document.getElementById('label');
let timer = null;

start.addEventListener('click', () => {
  clearInterval(timer);
  let ms = 10000, step = 100; // 10s
  const totalSteps = ms/step; let n = 0;
  timer = setInterval(() => {
    n++; const pct = Math.min(100, Math.round(n/totalSteps*100));
```

```
    bar.value = pct;
    label.textContent = pct===100 ? 'Done!' : `Time left: ${(ms - n*step)/1000}s`;
    if(pct===100) clearInterval(timer);
  }, step);
});
</script>
</body>
</html>
```

---

15) exercise15_file_preview.html — File Input Preview + Validation

```
<!doctype html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>File Input Preview + Validation</title>
<style>
  body{font:16px/1.5 system-ui, sans-serif; padding:1.5rem;}
  .thumb{max-width:280px; display:block; margin-top:.75rem; border:1px solid #ddd; border-radius:.5rem;}
  .error{color:#b00020;}
</style>
</head>
<body>
<h1>JS DOM Exercise 15 — Image Preview</h1>
<p>Select an image under 1 MB; preview appears below.</p>
<input type="file" id="file" accept="image/*">
<div id="msg" class="error"></div>
<img id="img" class="thumb" alt="">
```

```
<script>
const input = document.getElementById('file');
const img = document.getElementById('img');
const msg = document.getElementById('msg');

input.addEventListener('change', () => {
  msg.textContent = ''; img.removeAttribute('src');
  const f = input.files[0]; if(!f) return;
  if(!f.type.startsWith('image/')){ msg.textContent='Please select an image.'; return; }
  if(f.size > 1024*1024){ msg.textContent='File too large (>1MB).'; return; }
  const reader = new FileReader();
  reader.onload = e => { img.src = e.target.result; img.alt = f.name; };
  reader.readAsDataURL(f);
});
</script>
</body>
</html>
```

---

16) exercise16_clipboard_toast.html — Clipboard + Toast

```
<!doctype html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Clipboard + Toast</title>
<style>
  body{font:16px/1.5 system-ui, sans-serif; padding:1.5rem;}
  textarea{width:100%; max-width:520px; min-height:100px; padding:.6rem;}
```

```
  .toast{position:fixed; bottom:20px; left:50%; transform:translateX(-50%);
background:#111; color:#fff; padding:.5rem .75rem; border-radius:.5rem;
opacity:0; transition:opacity .2s;}
  .toast.show{opacity:1;}
</style>
</head>
<body>
<h1>JS DOM Exercise 16 — Copy to Clipboard</h1>
<textarea id="txt" placeholder="Type something to copy...">console.log('Hello
DOM');</textarea><br>
<button id="copy">Copy</button>
<div id="toast" class="toast" role="status" aria-live="polite"></div>

<script>
const copyBtn = document.getElementById('copy');
const txt = document.getElementById('txt');
const toast = document.getElementById('toast');

copyBtn.addEventListener('click', async () => {
  try{
    await navigator.clipboard.writeText(txt.value);
    showToast('Copied to clipboard!');
  }catch(err){
    showToast('Copy failed: ' + err.message);
  }
});

function showToast(message){
  toast.textContent = message;
  toast.classList.add('show');
  setTimeout(() => toast.classList.remove('show'), 1200);
```

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

---

17) exercise17_template_rows_csv.html — <template> Rows + CSV Export

```
<!doctype html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>&lt;template&gt; Rows + CSV Export</title>
<style>
  body{font:16px/1.5 system-ui, sans-serif; padding:1.5rem;}
  table{border-collapse:collapse; min-width:420px;}
  th,td{border:1px solid #ddd; padding:.5rem .75rem;}
  .row{display:flex; gap:.5rem; margin:.75rem 0;}
  input{padding:.4rem;}
</style>
</head>
<body>
<h1>JS DOM Exercise 17 — Template & CSV</h1>
<div class="row">
  <input id="name" placeholder="Name">
  <input id="age" type="number" placeholder="Age">
  <input id="city" placeholder="City">
  <button id="add">Add Row</button>
  <button id="export">Export CSV</button>
</div>
```

```
<table>
  <thead><tr><th>Name</th><th>Age</th><th>City</th></tr></thead>
  <tbody id="tbody"></tbody>
</table>

<template id="row">
  <tr><td class="n"></td><td class="a"></td><td class="c"></td></tr>
</template>

<script>
const tpl = document.getElementById('row');
const tb = document.getElementById('tbody');

document.getElementById('add').addEventListener('click', () => {
  const n = document.getElementById('name').value.trim();
  const a = document.getElementById('age').value.trim();
  const c = document.getElementById('city').value.trim();
  if(!n || !a || !c) return;
  const node = tpl.content.cloneNode(true);
  node.querySelector('.n').textContent = n;
  node.querySelector('.a').textContent = a;
  node.querySelector('.c').textContent = c;
  tb.appendChild(node);
  ['name','age','city'].forEach(id => document.getElementById(id).value='');
});

document.getElementById('export').addEventListener('click', () => {
  const rows = [...tb.querySelectorAll('tr')].map(tr => [...tr.children].map(td =>
td.textContent));
  const csv = ['Name,Age,City', ...rows.map(r => r.map(s =>
`"${s.replaceAll('"','""')}"`).join(','))].join('\n');
```

```
  const blob = new Blob([csv], {type:'text/csv'});
  const a = document.createElement('a'); a.href = URL.createObjectURL(blob); a.download = 'people.csv'; a.click();
  URL.revokeObjectURL(a.href);
});
</script>
</body>
</html>
```

---

18) exercise18_hash_router.html — Hash Router (SPA-like)

```
<!doctype html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Hash Router (SPA-like)</title>
<style>
  body{font:16px/1.5 system-ui,sans-serif;padding:1.5rem;}
  nav a{margin-right:.5rem;}
  section[hidden]{display:none!important;}
</style>
</head>
<body>
<h1>JS DOM Exercise 18 — Hash Router</h1>
<nav>
  <a href="#/home">Home</a>
  <a href="#/about">About</a>
  <a href="#/contact">Contact</a>
</nav>
```

```
<section id="home"><h2>Home</h2><p>Welcome! Change the hash to navigate.</p></section>
<section id="about" hidden><h2>About</h2><p>This is a tiny SPA using <code>hashchange</code>.</p></section>
<section id="contact" hidden><h2>Contact</h2><p>Email: hello@example.com</p></section>

<script>
const routes = { '/home':'home', '/about':'about', '/contact':'contact' };
function render(){
  const hash = location.hash.replace('#','') || '/home';
  document.querySelectorAll('section').forEach(s => s.hidden = true);
  const id = routes[hash] || 'home';
  document.getElementById(id).hidden = false;
}
window.addEventListener('hashchange', render); render();
</script>
</body>
</html>
```

---

19) exercise19_web_component_stars.html — Web Component: Star Rating

```
<!doctype html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Web Component: Star Rating</title>
</head>
<body>
<h1>JS DOM Exercise 19 — Shadow DOM Component</h1>
```

```
<star-rating value="3"></star-rating>
<star-rating value="5"></star-rating>

<script>
class StarRating extends HTMLElement{
  static get observedAttributes(){ return ['value']; }
  constructor(){
    super();
    this.attachShadow({mode:'open'});
    this.shadowRoot.innerHTML = `
      <style>
        .wrap{ display:inline-flex; gap:.25rem; cursor:pointer; }
        .star{ font-size:1.5rem; color:#bbb; }
        .star.filled{ color:#f5a623; }
      </style>
      <div class="wrap" role="slider" aria-valuemin="0" aria-valuemax="5"
aria-valuenow="0"></div>
    `;
    this.wrap = this.shadowRoot.querySelector('.wrap');
    this.value = Number(this.getAttribute('value')||0);
    for(let i=1;i<=5;i++){
      const span = document.createElement('span');
      span.className='star'; span.textContent='★'; span.dataset.v=i;
      this.wrap.appendChild(span);
    }
    this.wrap.addEventListener('click', e => {
      if(e.target.classList.contains('star')){
        this.value = Number(e.target.dataset.v);
        this.setAttribute('value', String(this.value));
        this.dispatchEvent(new CustomEvent('change',{detail:{value:this.value}}));
      }
```

```
    });
    this.render();
  }
  attributeChangedCallback(){ this.value = Number(this.getAttribute('value')||0); this.render(); }
  render(){
    [...this.wrap.children].forEach(star => star.classList.toggle('filled', Number(star.dataset.v) <= this.value));
    this.wrap.setAttribute('aria-valuenow', String(this.value));
  }
}
customElements.define('star-rating', StarRating);
</script>
</body>
</html>
```

---

20) exercise20_domparser_sanitize.html — DOMParser + Simple Sanitizer

```
<!doctype html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>DOMParser + Simple Sanitizer</title>
<style>
  body{font:16px/1.5 system-ui, sans-serif; padding:1.5rem;}
  textarea{width:100%; max-width:720px; min-height:120px; padding:.6rem;}
  .preview{border:1px solid #ddd; border-radius:.5rem; padding:1rem; margin-top:.5rem;}
</style>
</head>
```

```
<body>
<h1>JS DOM Exercise 20 — Parse HTML Safely</h1>
<p>Enter limited HTML. Only <code>&lt;strong&gt;</code>,
<code>&lt;em&gt;</code>, <code>&lt;code&gt;</code>, and
<code>&lt;a&gt;</code> (with <code>href</code>) are allowed.</p>
<textarea id="src">&lt;strong&gt;Hello&lt;/strong&gt; &lt;img src=x
onerror=alert(1)&gt; &lt;a href="https://example.com"
onclick="evil()"&gt;link&lt;/a&gt;</textarea><br>
<button id="render">Render</button>
<div id="out" class="preview"></div>

<script>
function sanitize(html){
  const doc = new DOMParser().parseFromString(html, 'text/html');
  const allowed = new Set(['STRONG','EM','CODE','A']);
  const walker = doc.createTreeWalker(doc.body, NodeFilter.SHOW_ELEMENT);
  let node;
  while(node = walker.nextNode()){
    if(!allowed.has(node.tagName)){
      const text =
doc.createTextNode(node.outerHTML.replace(/</g,'&lt;').replace(/>/g,'&gt;'));
      node.replaceWith(text);
    }else{
      [...node.attributes].forEach(attr => {
        if(!(node.tagName==='A' && attr.name==='href'))
node.removeAttribute(attr.name);
      });
      if(node.tagName==='A'){ node.setAttribute('target','_blank');
node.setAttribute('rel','noopener'); }
    }
  }
  return doc.body.innerHTML;
```

```
}

document.getElementById('render').addEventListener('click', () => {
  const src = document.getElementById('src').value;
  document.getElementById('out').innerHTML = sanitize(src);
});
</script>
</body>
</html>
```