50+ JavaScript vežbi sa rešenjima
(vanilla JS umesto jQuery)

Priručnik za učenike

October 29, 2025

Cilj: isti zadaci kao u jQuery verziji, ali svako rešenje je čisti JavaScript (*vanilla JS*), bez bilo kakvih biblioteka.

Kako koristiti ovaj materijal

- Skripte su stavljene pri dnu `<body>` ili koristi `DOMContentLoaded`.
- Selektori: `document.querySelector` (`qs`) i `document.querySelectorAll` (`qsa`).
- Pomoćne funkcije (opciono korisne):

```
<script>
// Mini pomoćnici:
const qs = (sel, root=document)=>root.querySelector(sel);
const qsa = (sel, root=document)=>Array.from(root.querySelectorAll(sel));

// Event delegacija:
function on(root, evt, sel, handler){
  root.addEventListener(evt, e=>{
    if(e.target.closest(sel)) handler(e, e.target.closest(sel));
  });
}

// Jednostavan serialize(Form) -> query string:
function serializeForm(form){
  return new URLSearchParams(new FormData(form)).toString();
}

// Mali util za fade (bez jQuery):
function fadeTo(el, target=1, dur=150){
  el.style.transition = `opacity ${dur}ms`;
  // osiguraj početnu vrednost
  if(getComputedStyle(el).opacity==='') el.style.opacity = '1';
  requestAnimationFrame(()=>{ el.style.opacity = String(target); });
}
</script>
```

Blok A — Osnove selektora, manipulacije i događaja (1–15)

Zadatak 1. *Selektor po ID-ju i promena teksta*

Klik na dugme menja tekst naslova u „Zdravo, JS!“. **Rešenje.**

```
<h1 id="title">Naslov</h1>
<button id="btn">Promeni</button>
<script>
  qs('#btn').addEventListener('click', ()=>{
    qs('#title').textContent = 'Zdravo, JS!';
  });
</script>
```

Zadatak 2. *Selektor po klasi i CSS*

Dva paragrafa imaju klasu `note`. Klik dodaje žutu pozadinu svima. **Rešenje.**

```
<p class="note">Tekst 1</p><p class="note">Tekst 2</p>
<button id="mark">Označi</button>
<script>
  qs('#mark').addEventListener('click', ()=>{
    qsa('.note').forEach(p=>p.style.background='#fff7a8');
  });
</script>
```

Zadatak 3. *Složen selektor (neparne stavke)*

Oboji sve neparne stavke liste u sivo. **Rešenje.**

```
<ul id="items"><li>A</li><li>B</li><li>C</li><li>D</li></ul>
<script>
  qsa('#items li').forEach((li,i)=>{ if(i%2===1) li.style.background='#eee'; });
</script>
```

Zadatak 4. *`textContent` vs `innerHTML`*

Jednim klikom ubaci raw tekst „JS“, drugim klikom ubaci HTML. **Rešenje.**

```
<div id="box"></div>
<button id="asText">Kao tekst</button>
<button id="asHtml">Kao HTML</button>
<script>
  qs('#asText').addEventListener('click', ()=>qs('#box').textContent='<b>JS</b>');
  qs('#asHtml').addEventListener('click', ()=>qs('#box').innerHTML='<b>JS</b>');
</script>
```

Zadatak 5. *Dodavanje i uklanjanje klasa*

Toggle klasu `active` na kartici kad se klikne. **Rešenje.**

```
<div id="card">Kartica</div>
<style>.active{border:2px solid #4a90e2; padding:6px;}</style>
<script>
  qs('#card').addEventListener('click', e=>e.currentTarget.classList.toggle('active'))
      ;
</script>
```

Zadatak 6. *`value` unos*

Kopiraj vrednost inputa u `#out`. **Rešenje.**

```
<input id="name" placeholder="Ime"><button id="go">Prikaži</button>
<div id="out"></div>
<script>
  qs('#go').addEventListener('click', ()=>qs('#out').textContent = qs('#name').value);
</script>
```

Zadatak 7. *Atributi i svojstva*

Toggle `disabled` i promeni `title`. **Rešenje.**

```
<button id="action" title="Klikni me">Akcija</button>
<button id="toggle">Zaključaj/Otključaj</button>
<script>
  qs('#toggle').addEventListener('click', ()=>{
    const b = qs('#action');
    b.disabled = !b.disabled;
    b.title = b.disabled ? 'Zaključano' : 'Klikni me';
  });
</script>
```

Zadatak 8. *Umetanje elemenata: `append/prepend`*

Dodaj stavke na kraj i početak liste. **Rešenje.**

```
<ul id="lst"><li>Start</li></ul>
<button id="addEnd">Dodaj kraj</button>
<button id="addStart">Dodaj početak</button>
<script>
  qs('#addEnd').addEventListener('click', ()=>qs('#lst').insertAdjacentHTML('beforeend
      ','<li>Kraj</li>'));
  qs('#addStart').addEventListener('click', ()=>qs('#lst').insertAdjacentHTML('
      afterbegin','<li>Početak</li>'));
</script>
```

Zadatak 9. *`before` i `after`*

Ubaci objašnjenje pre i posle slike. **Rešenje.**

```
<img id="pic" src="x.png" alt="Slika" />
<script>
  const pic = qs('#pic');
  pic.insertAdjacentHTML('beforebegin','<p>Pre slike</p>');
  pic.insertAdjacentHTML('afterend','<p>Posle slike</p>');
</script>
```

Zadatak 10. *Uklanjanje vs pražnjenje*

„Empty“ prazni sadržaj, „Remove“ briše kontejner. **Rešenje.**

```
<div id="panel"><b>Tekst</b> i <i>deca</i></div>
<button id="btnEmpty">Empty</button>
<button id="btnRemove">Remove</button>
<script>
  qs('#btnEmpty').addEventListener('click', ()=>qs('#panel').innerHTML='');
  qs('#btnRemove').addEventListener('click', ()=>qs('#panel').remove());
</script>
```

Zadatak 11. *Chaining (sekvencijalne akcije)*

Dodaj klasu, promeni tekst, fadeOut pa fadeIn. **Rešenje.**

```
<div id="chip">Čip</div><button id="do">Anim</button>
<style>.hit{border:1px solid #aaa;}</style>
<script>
  qs('#do').addEventListener('click', ()=>{
    const el = qs('#chip');
    el.classList.add('hit');
    el.textContent = 'Obrađeno';
    fadeTo(el, 0, 200);
    setTimeout(()=>fadeTo(el, 1, 200), 220);
  });
</script>
```

Zadatak 12. *Delegacija događaja*

Klik na dinamički dodate `li` hvata se preko roditelja. **Rešenje.**

```
<ul id="menu"></ul>
<button id="add">Dodaj stavku</button>
<style>.sel{background:#d0ebff;}</style>
<script>
  let i=1;
  qs('#add').addEventListener('click', ()=>qs('#menu').insertAdjacentHTML('beforeend',
       `<li>Stavka ${i++}</li>`));
  on(qs('#menu'), 'click', 'li', (e, li)=>li.classList.toggle('sel'));
</script>
```

Zadatak 13. *Keyboard događaji*

Enter u inputu pokreće isti handler kao klik na „Traži“. **Rešenje.**

```
<input id="q" placeholder="Upit"><button id="search">Traži</button>
<div id="result"></div>
<script>
  function go(){ qs('#result').textContent = 'Tražim: ' + qs('#q').value; }
  qs('#search').addEventListener('click', go);
  qs('#q').addEventListener('keydown', e=>{ if(e.key==='Enter') go(); });
</script>
```

Zadatak 14. *Prevent default*

Link „Pošalji“ ne sme da otvara URL već da prikaže poruku. **Rešenje.**

```
<a id="send" href="https://example.com">Pošalji</a>
<script>
  qs('#send').addEventListener('click', e=>{
    e.preventDefault();
    alert('Slanje kroz JS!');
  });
</script>
```

Zadatak 15. *Stop propagation*

Klik na dugme u kartici ne sme da aktivira klik kartice. **Rešenje.**

```
<div id="card2" style="padding:10px;border:1px solid #aaa;">
  Kartica <button id="inner">Dugme</button>
</div>
<script>
  qs('#card2').addEventListener('click', ()=>alert('Kartica klik'));
  qs('#inner').addEventListener('click', e=>{ e.stopPropagation(); alert('Dugme klik')
      ; });
</script>
```

Blok B — DOM obilazak, forme, animacije, dimenzije (16–30)

Zadatak 16. *Traversing: `parentElement`, `children`, `querySelectorAll`*

Označi roditelja kliknutog elementa i svu njegovu decu. **Rešenje.**

```
<div class="box"><span class="hit">Klikni</span><i>još</i></div>
<script>
  on(document,'click','.hit',(e,el)=>{
    const p = el.parentElement;
    p.style.border='1px solid #4a90e2';
    Array.from(p.children).forEach(c=>c.style.background='#eef6ff');
  });
</script>
```

Zadatak 17. *`closest` i `siblings`*

Klik boji najbliži `.row` i susede elementa. **Rešenje.**

```
<ul class="row"><li>A</li><li class="x">B</li><li>C</li></ul>
<script>
  on(document,'click','.x',(e,el)=>{
    const r = el.closest('.row');
    r.style.outline='1px dashed #888';
    Array.from(el.parentElement.children).forEach(li=>{
      if(li!==el) li.style.opacity='0.6';
    });
  });
</script>
```

Zadatak 18. *Forme: `serialize()` ekvivalent*

Spreči submit i prikaži serijalizovane podatke. **Rešenje.**

```

<form id="f"
      action="https://httpbin.org/anything"
      method="GET"
      enctype="application/x-www-form-urlencoded"
      target="_blank">
<input name="ime"><input name="grad"><button>OK</button></form>
<pre id="out2"></pre>
<script>
  qs('#f').addEventListener('submit', e=>{
    e.preventDefault();
    qs('#out2').textContent = serializeForm(e.currentTarget);
```

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

Zadatak 19. *Validacija forme (minimalno)*

Ako je polje prazno, poruka + klasa greške. **Rešenje.**

```
<input id="email" placeholder="Email"><button id="save">Sačuvaj</button>
<div id="msg"></div>
<style>.err{border:1px solid #e03131;}</style>
<script>
  qs('#save').addEventListener('click', ()=>{
    const el = qs('#email'), v = el.value.trim();
    if(!v){ el.classList.add('err'); qs('#msg').textContent='Email je obavezan'; }
    else { el.classList.remove('err'); qs('#msg').textContent='OK'; }
  });
</script>
```

Zadatak 20. *`dataset` API*

Sačuvaj broj klikova u `data-n` i prikaži ga. **Rešenje.**

```
<button id="count" data-n="0">Klikni me</button>
<script>
  qs('#count').addEventListener('click', e=>{
    const b = e.currentTarget; b.dataset.n = String((+b.dataset.n)+1);
    b.textContent = 'Klikova: ' + b.dataset.n;
  });
</script>
```

Zadatak 21. *Show/Hide toggle*

Toggle panela na klik (bez biblioteka). **Rešenje.**

```
<button id="t">Prikaži/Sakrij</button>
<div id="p" style="padding:10px;background:#fafafa;">Panel</div>
<script>
  qs('#t').addEventListener('click', ()=>{
    const p = qs('#p');
    p.style.display = (p.style.display==='none' ? '' : 'none');
  });
</script>
```

Zadatak 22. *Fade efekat pri hover-u*

Koristi `fadeTo` pomoćnik iz uvoda. **Rešenje.**

```
<img id="photo" src="x.png" alt="x" style="opacity:.6;">
<script>
  const ph = qs('#photo');
  ph.addEventListener('mouseenter', ()=>fadeTo(ph,1,150));
  ph.addEventListener('mouseleave', ()=>fadeTo(ph,0.6,150));
</script>
```

Zadatak 23. *Akoridion (slideToggle stil)*

Jednostavno otvaranje/zatvaranje sledećeg elementa. **Rešenje.**

```
<h3 class="hdr">Naslov</h3>
<div class="cnt" style="display:none;">Sadržaj</div>
<script>
  on(document,'click','.hdr',(e,el)=>{
    const nx = el.nextElementSibling;
    nx.style.display = (nx.style.display==='none' || !nx.style.display) ? 'block' : '
        none';
  });
</script>
```

Zadatak 24. *`animate` širina (CSS transition)*

Povećaj širinu kutije sa 100px na 250px u 400ms. **Rešenje.**

```
<div id="bx" style="width:100px;height:20px;background:#ddd;transition:width .4s;"></
    div>
<button id="grow">Rasti</button>
<script>
  qs('#grow').addEventListener('click', ()=>{ qs('#bx').style.width='250px'; });
</script>
```

Zadatak 25. *Dimenzije: width/inner/outer ekvivalenti*

Prikaži dimenzije uključujući padding/border/margine. **Rešenje.**

```
<div id="box3" style="width:120px;padding:10px;border:5px solid #ccc;margin:8px;">.</
    div>
<pre id="dim"></pre>
<script>
  const b = qs('#box3');
  const rect = b.getBoundingClientRect();
  const width = b.clientWidth; // content + padding
  const innerWidth = b.clientWidth; // isto kao gore u CSS box-modelu
  const outerWidth = b.offsetWidth; // content + padding + border
  qs('#dim').textContent = `width=${width} innerWidth=${innerWidth} outerWidth=${
      outerWidth}`;
</script>
```

Zadatak 26. *Pozicija: `getBoundingClientRect` i `offsetLeft/Top`*

Prikaži offset stranice i relativnu poziciju u roditelju. **Rešenje.**

```
<div style="position:relative; margin-top:40px;">
  <div id="dot" style="position:absolute;left:30px;top:20px;">•</div>
</div>
<pre id="pos"></pre>
<script>
  const d = qs('#dot');
  const off = d.getBoundingClientRect();
  const pos = {left:d.offsetLeft, top:d.offsetTop};
  qs('#pos').textContent = 'offset=' + JSON.stringify(off) + ' position=' + JSON.
      stringify(pos);
</script>
```

Zadatak 27. *Skrol događaji*

Prikaži koliko je prozor skrolovan po Y osi. **Rešenje.**

```
<div style="height:1200px"></div>
<pre id="s"></pre>
<script>
  addEventListener('scroll', ()=>qs('#s').textContent = String(window.scrollY));
</script>
```

Zadatak 28. *Throttle sa `setTimeout`*

Na `keyup`, ažuriraj 300ms nakon poslednje promene. **Rešenje.**

```
<input id="deb"><div id="r"></div>
<script>
  let t=null;
  qs('#deb').addEventListener('keyup', e=>{
    clearTimeout(t);
    const v = e.currentTarget.value;
    t = setTimeout(()=>qs('#r').textContent=v, 300);
  });
</script>
```

Zadatak 29. *Rad sa checkbox-ovima*

„Označi sve“ selektuje/deselektuje sve stavke. **Rešenje.**

```
<label><input type="checkbox" id="all"> Označi sve</label>
<div>
  <label><input class="it" type="checkbox"> A</label>
  <label><input class="it" type="checkbox"> B</label>
</div>
<script>
  qs('#all').addEventListener('change', e=>{
    qsa('.it').forEach(c=>c.checked = e.currentTarget.checked);
  });
</script>
```

Zadatak 30. *Promena atributa funkcijom*

Povećavaj `data-score` za +10 pri svakom kliku. **Rešenje.**

```
<div id="sc" data-score="0">Score</div>
<script>
  qs('#sc').addEventListener('click', e=>{
    const el = e.currentTarget;
    const old = +el.getAttribute('data-score');
    el.setAttribute('data-score', String(old+10));
  });
</script>
```

Zadatak 31. *Filtriranje teksta*

Oboji stavke koje sadrže „a“, ostale dezaktiviraj (opacity). **Rešenje.**

```
<ul id="w"><li>Java</li><li>CSS</li><li>HTML</li><li>Scala</li></ul>
<script>
  qsa('#w li').forEach(li=>{
    if(li.textContent.toLowerCase().includes('a')) li.style.background='#e6fcf5';
```

```
    else li.style.opacity='0.5';
  });
</script>
```

Zadatak 32. *Kloniranje čvora*

Dupliraj element i ubaci ga posle originala. **Rešenje.**

```
<div id="badge">Tag</div><button id="dup">Dupliraj</button>
<script>
  qs('#dup').addEventListener('click', ()=>{
    const b = qs('#badge');
    const clone = b.cloneNode(true);
    b.insertAdjacentElement('afterend', clone);
  });
</script>
```

Blok C — AJAX, JSON, templating, skladištenje (31–40)

Zadatak 33. *GET statičkog JSON-a*

Učitaj `data.json` i prikaži `title`. **Rešenje.**

```
// data.json: { "title":"Pozdrav", "items":[1,2,3] }
<div id="ajax1"></div>
<script>
  fetch('data.json').then(r=>r.json()).then(d=>{
    qs('#ajax1').textContent = d.title;
  });
</script>
```

Zadatak 34. *POST forma*

Pošalji formu kao `application/x-www-form-urlencoded` i prikaži odgovor. **Rešenje.**

```
<form id="login"><input name="u"><input name="p" type="password"><button>Prijava</
    button></form>
<pre id="resp"></pre>
<script>
  qs('#login').addEventListener('submit', async e=>{
    e.preventDefault();
    const body = new URLSearchParams(new FormData(e.currentTarget));
    try{
      const r = await fetch('/api/login',{method:'POST',headers:{'Content-Type':'
          application/x-www-form-urlencoded'}, body});
      const data = await r.json();
      qs('#resp').textContent = JSON.stringify(data);
    }catch{
      qs('#resp').textContent = 'Greška';
    }
  });
</script>
```

Zadatak 35. *Fetch sa timeout-om i statusima*

`beforeSend` ekvivalent, `timeout`, `error` handleri. **Rešenje.**

```
<pre id="outA"></pre>
<script>
  const out = qs('#outA'); out.textContent = 'Učitavam...';
  const ctrl = new AbortController();
  const t = setTimeout(()=>ctrl.abort(), 2000);
  fetch('/api/data',{signal:ctrl.signal})
    .then(r=>{ if(!r.ok) throw new Error('HTTP '+r.status); return r.json(); })
    .then(d=> out.textContent = JSON.stringify(d))
    .catch(()=> out.textContent = 'Greška ili timeout')
    .finally(()=> clearTimeout(t));
</script>
```

Zadatak 36. *Dinamičko renderovanje liste iz JSON-a*

Mapiraj niz objekata u `li` elemente. **Rešenje.**

```
// Pretpostavimo: const data=[{name:'Ana'},{name:'Marko'}];
<ul id="listX"></ul>
<script>
  const data=[{name:'Ana'},{name:'Marko'}];
  qs('#listX').innerHTML = data.map(o=>`<li>${o.name}</li>`).join('');
</script>
```

Zadatak 37. *Loader/spinner tokom Fetch-a*

Pokaži „Učitavam...“ dok traje zahtev; sakrij kad stigne odgovor. **Rešenje.**

```
<div id="spin" style="display:none;">Učitavam...</div><div id="data"></div>
<script>
  const spin=qs('#spin'); spin.style.display='block';
  fetch('/api/items')
    .then(r=>r.json())
    .then(d=>qs('#data').textContent = d.length + ' stavki')
    .catch(()=>qs('#data').textContent='Greška')
    .finally(()=>spin.style.display='none');
</script>
```

Zadatak 38. *Keširanje u `localStorage`*

Ako postoji keš, koristi ga; inače učitaj pa sačuvaj. **Rešenje.**

```
<div id="cache"></div>
<script>
  const k='itemsCache';
  const cached = localStorage.getItem(k);
  if(cached){
    qs('#cache').textContent = 'Iz keša: ' + JSON.parse(cached).length;
  }else{
    fetch('/api/items').then(r=>r.json()).then(d=>{
      localStorage.setItem(k, JSON.stringify(d));
      qs('#cache').textContent = 'Sveže: ' + d.length;
    });
  }
</script>
```

Zadatak 39. *Form-data u JSON objekat*

Pretvori `FormData` u objekat pa pošalji kao JSON. **Rešenje.**

```
<form id="f2"><input name="a"><input name="b"><button>OK</button></form>
<pre id="j"></pre>
<script>
  qs('#f2').addEventListener('submit', async e=>{
    e.preventDefault();
    const fd = new FormData(e.currentTarget);
    const obj = Object.fromEntries(fd.entries());
    qs('#j').textContent = JSON.stringify(obj);
    await fetch('/api/save',{method:'POST',headers:{'Content-Type':'application/json'},
        body:JSON.stringify(obj)});
  });
</script>
```

Zadatak 40. *Greške (`catch`)*

Namerno zovi nepostojeći URL i prikaži poruku u crvenom. **Rešenje.**

```
<div id="err"></div>
<script>
  fetch('/nope').then(r=>{ if(!r.ok) throw new Error(r.status); })
    .catch(err=>{ const e=qs('#err'); e.style.color='#e03131'; e.textContent='Greška:
        '+(err.message||''); });
</script>
```

Zadatak 41. *Timeout i abort zahteva*

Prekini zahtev nakon 300ms ako nije završen. **Rešenje.**

```
<pre id="ab"></pre>
<script>
  const out=qs('#ab');
  const ctrl = new AbortController();
  const t = setTimeout(()=>{ ctrl.abort(); out.textContent='Prekinuto'; }, 300);
  fetch('/api/slow',{signal:ctrl.signal})
    .then(()=> out.textContent='OK')
    .catch(()=> out.textContent='Prekinuto')
    .finally(()=> clearTimeout(t));
</script>
```

Zadatak 42. *Mini templating funkcija*

Zameni {{name}} u stringu na osnovu objekta; prikaži rezultat. **Rešenje.**

```
<div id="tplOut"></div>
<script>
  function tpl(s, ctx){ return s.replace(/\{\{(\w+)\}\}/g, (_,k)=>ctx[k]??''); }
  const s = '<p>Zdravo, {{name}}!</p>';
  qs('#tplOut').innerHTML = tpl(s, {name:'Svet'});
</script>
```

Blok D — Obrasci, performanse, pristupačnost, integracije (41–50)

Zadatak 43. *„Plugin“ stil bez biblioteka*

Napiši funkciju `blink(el, times, speed)` koja treperi element N puta. **Rešenje.**

```
<div id="blinkMe">Treperim</div>
<script>
  function blink(el, times=3, speed=150){
    let i=0;
    (function step(){
      fadeTo(el,0,speed);
      setTimeout(()=>{ fadeTo(el,1,speed); if(++i<times) setTimeout(step, speed+10); },
           speed+10);
    })();
  }
  blink(qs('#blinkMe'), 4, 100);
</script>
```

Zadatak 44. *Opcije i podrazumevane vrednosti*

`highlight(el,color,duration)` privremeno boji pozadinu. **Rešenje.**

```
<p class="hl">Tekst</p>
<script>
  function highlight(el, opts={}){
    const o = Object.assign({color:'#fff3bf', duration:300}, opts);
    const old = getComputedStyle(el).backgroundColor;
    el.style.transition = `background-color ${o.duration}ms, opacity ${o.duration}ms`;
    el.style.backgroundColor = o.color;
    el.style.opacity='0.9';
    setTimeout(()=>{ el.style.opacity='1'; setTimeout(()=>{ el.style.backgroundColor =
         old; }, o.duration); }, o.duration);
  }
  qsa('.hl').forEach(el=>highlight(el,{color:'#c0ebff', duration:200}));
</script>
```

Zadatak 45. *„Event namespacing“ ekvivalent*

Čuvaj reference handlera pa ukloni samo jedan. **Rešenje.**

```
<button id="ns">Klik</button>
<script>
  const btn = qs('#ns');
  const handlerA = ()=>console.log('A');
  const handlerB = ()=>console.log('B');
  btn.addEventListener('click', handlerA);
  btn.addEventListener('click', handlerB);
  // Ukloni samo A:
  btn.removeEventListener('click', handlerA);
</script>
```

Zadatak 46. *Custom događaj*

Triggeruj i slušaj sopstveni događaj `done`. **Rešenje.**

```
<div id="job"></div>
<script>
  const job = qs('#job');
  job.addEventListener('done', e=> job.textContent = 'Gotovo: ' + e.detail.msg);
  setTimeout(()=> job.dispatchEvent(new CustomEvent('done',{detail:{msg:'uspeh'}})),
      200);
</script>
```

Zadatak 47. *Delegacija i performanse*

Jedan handler na kontejner umesto 1000 pojedinačnih bind-ova. **Rešenje.**

```
<div id="wrap"></div><pre id="log"></pre>
<script>
  const wrap = qs('#wrap');
  let html = '';
  for(let i=0;i<1000;i++) html += `<button class="b">#${i}</button>`;
  wrap.innerHTML = html;
  on(wrap,'click','.b',(e,btn)=>{ qs('#log').textContent = btn.textContent; });
</script>
```

Zadatak 48. *Pristupačnost: fokus i tastatura*

Enter/Space aktivira „link-karticu“ (`role="button"`). **Rešenje.**

```
<div id="card" tabindex="0" role="button" aria-pressed="false">Otvori</div>
<script>
  const card = qs('#card');
  card.addEventListener('keydown', e=>{
    if(e.key==='Enter' || e.key===' '){ e.preventDefault(); card.click(); }
  });
  card.addEventListener('click', ()=>{
    card.setAttribute('aria-pressed', card.getAttribute('aria-pressed')==='false'?'
        true':'false');
  });
</script>
```

Zadatak 49. *Promise*

Kreiraj veštački async zadatak (`Promise`) i koristi `then()`. **Rešenje.**

```
<pre id="pms"></pre>
<script>
  function asyncJob(){ return new Promise(res=>setTimeout(()=>res('OK'),200)); }
  asyncJob().then(v=>qs('#pms').textContent=v);
</script>
```

Zadatak 50. *Fetch API + render*

Učitaj JSON preko Fetch-a, prikaži vanilla JS-om. **Rešenje.**

```
<ul id="u"></ul>
<script>
  fetch('/api/users').then(r=>r.json()).then(d=>{
    qs('#u').innerHTML = d.map(x=>`<li>${x.name}</li>`).join('');
  });
</script>
```

Zadatak 51. *Debounce util za input događaje*

`debounce(fn, wait)` i `keyup`. **Rešenje.**

```
<input id="search2"><div id="outX"></div>
<script>
  function debounce(fn, wait){ let t; return function(...a){ clearTimeout(t); t=
      setTimeout(()=>fn.apply(this,a), wait); }; }
```

```
  qs('#search2').addEventListener('keyup', debounce(function(){
    qs('#outX').textContent = this.value;
  }, 300));
</script>
```

Zadatak 52. *Mala SPA navigacija (hashchange)*

Na promenu hash-a prikazuj odgovarajući „view“. **Rešenje.**

```
<nav>
  <a href="#home">Home</a> | <a href="#about">About</a>
</nav>
<div id="view"></div>
<script>
  function render(){
    const h=location.hash||'#home';
    qs('#view').textContent = (h==='#about') ? 'O aplikaciji' : 'Dobrodošli';
  }
  addEventListener('hashchange', render); render();
</script>
```

Zadatak 53. *Smoke test*

Proveri da je element dostupan pre bind-ovanja. **Rešenje.**

```
<div id="ok"></div>
<script>
  if(qs('#ok')){ qs('#ok').textContent='JS OK'; }
</script>
```

Bonus mini-vežbe (51–55)

Zadatak 54. *Dinamički `select` iz niza*

Popuni `select` i prikaži izbor. **Rešenje.**

```
<select id="sel"></select><div id="cho"></div>
<script>
  const arr=['Beograd','Novi Sad','Niš'];
  qs('#sel').innerHTML = arr.map(x=>`<option>${x}</option>`).join('');
  qs('#sel').addEventListener('change', e=>qs('#cho').textContent = e.target.value);
</script>
```

Zadatak 55. *Brojač karaktera*

Prikazuj preostale karaktere (maks 50) i oboji crveno kad je 0. **Rešenje.**

```
<textarea id="t" maxlength="50"></textarea>
<div id="c"></div>
<script>
  qs('#t').addEventListener('input', e=>{
    const left=50-e.target.value.length;
    const c=qs('#c'); c.textContent = left;
    c.style.color = (left===0)?'#e03131':'inherit';
  });
</script>
```

Zadatak 56. *Sortiranje (DOM reflow)*

Klik sortira `li` po alfabetu. **Rešenje.**

```
<ul id="l2"><li>banan</li><li>ana</li><li>cuk</li></ul>
<button id="sort">Sortiraj</button>
<script>
  qs('#sort').addEventListener('click', ()=>{
    const ul=qs('#l2');
    const li = Array.from(ul.children).sort((a,b)=>a.textContent.localeCompare(b.
        textContent));
    ul.innerHTML=''; li.forEach(x=>ul.appendChild(x));
  });
</script>
```

Zadatak 57. *Drag toggle (klasa pri držanju miša)*

Dok je dugme miša pritisnuto nad elementom, dodaj `dragging`. **Rešenje.**

```
<div id="drag" style="width:100px;height:40px;background:#ddd;">Drži</div>
<style>.dragging{outline:2px dashed #4a90e2;}</style>
<script>
  const d = qs('#drag');
  d.addEventListener('mousedown', ()=>d.classList.add('dragging'));
  d.addEventListener('mouseup', ()=>d.classList.remove('dragging'));
  d.addEventListener('mouseleave', ()=>d.classList.remove('dragging'));
</script>
```

Zadatak 58. *Clipboard (kopiranje teksta)*

Kopiraj tekst iz inputa u clipboard (Clipboard API). **Rešenje.**

```
<input id="clip" value="Kopiraj me"><button id="cpy">Kopiraj</button>
<script>
  qs('#cpy').addEventListener('click', async ()=>{
    try{ await navigator.clipboard.writeText(qs('#clip').value); alert('Kopirano'); }
    catch{ alert('Neuspeh'); }
  });
</script>
```

Tematska pokrivenost

- Selektori, traversing, manipulacija DOM-om i CSS-om
- Događaji: klik, tastatura, delegacija, custom događaji
- Forme: čitanje/validacija, serialize (`FormData`+`URLSearchParams`), `value`, `disabled`, `title`
- Animacije: fade (mali util), show/hide, CSS `transition`
- Dimenzije i pozicioniranje: `client/offsetWidth`, `getBoundingClientRect`, skrol
- AJAX/Fetch: `GET/POST`, JSON, loader, greške, abort/timeout
- Skladištenje: `localStorage`
- Utility: debounce, templating, Promise

- „Plugin“ obrasci implementirani funkcijama
- Mini SPA hash routing