Vanilla React (UMD) — lokalno, bez build-a

58 primera (React reimplementacija)

Priručnik za učenike

October 29, 2025

Sve radi iz jednog `.html` *fajla, offline. Stavi sledeće fajlove u* `./libs`*:*
`react.development.js`, `react-dom.development.js`, `babel.min.js`.

```
<!-- React kod (JSX) -->
<script type="text/babel">
  // ========================================================================
  // React hooks & helpers
  // ========================================================================
  const {
    useState,
    useEffect,
    useRef,
    useMemo,
    useContext,
    createContext,
  } = React;

  // Small DOM helper (used in a couple of places)
  const qs = (sel, root = document) => root.querySelector(sel);

  // Simple fade wrapper for quick visual feedback
  const FadeWrap = ({ show = true, duration = 200, children }) => {
    const ref = useRef(null);

    useEffect(() => {
      const el = ref.current;
      if (!el) return;
      el.style.transition = `opacity ${duration}ms`;
      requestAnimationFrame(() => {
        el.style.opacity = show ? "1" : "0";
      });
    }, [show, duration]);

    return (
      <div ref={ref} style={{ opacity: show ? 1 : 0 }}>
        {children}
      </div>
    );
  };

  // Serialize a real <form> element to querystring (GET-like)
  const serializeForm = (form) =>
    new URLSearchParams(new FormData(form)).toString();

  // Tiny string templating
```

```
const tpl = (s, ctx) =>
  s.replace(/\{\{(\w+)\}\}/g, (_, k) => (k in ctx ? ctx[k] : ""));

// Offline mock API (so students don't need internet)
const mockApi = {
  data: () =>
    new Promise((res) =>
      setTimeout(() => res({ title: "Pozdrav", items: [1, 2, 3] }), 300)
    ),
  items: () =>
    new Promise((res) => setTimeout(() => res([1, 2, 3, 4, 5]), 300)),
  users: () =>
    new Promise((res) =>
      setTimeout(() => res([{ name: "Ana" }, { name: "Marko" }]), 300)
    ),
  login: (obj) =>
    new Promise((res) =>
      setTimeout(() => res({ ok: true, user: obj.u || "anon" }), 300)
    ),
  save: (obj) =>
    new Promise((res) => setTimeout(() => res({ ok: true, saved: obj }), 300)),
  slow: () => new Promise((res) => setTimeout(() => res({ ok: true }), 5000)),
  nope: () =>
    new Promise((_, rej) => setTimeout(() => rej(new Error("404")), 300)),
};

// ========================================================================
// 1-25  |  Osnove + događaji
// ========================================================================

// 1) Promena teksta
const Ex1 = () => {
  const [txt, setTxt] = useState("Naslov");
  return (
    <>
      <h2>{txt}</h2>
      <button onClick={() => setTxt("Zdravo, React!")}>Promeni</button>
    </>
  );
};

// 2) Obeležavanje (klasa/pozadina)
const Ex2 = () => {
  const [marked, setMarked] = useState(false);
  const bg = marked ? "#fff7a8" : "transparent";
  return (
    <>
      <div className="row">
        <p style={{ background: bg }}>Tekst 1</p>
        <p style={{ background: bg }}>Tekst 2</p>
      </div>
      <button onClick={() => setMarked(true)}>Označi</button>
    </>
  );
};

// 3) :odd stil (svaka druga stavka)
const Ex3 = () => {
```

```
  const items = ["A", "B", "C", "D"];
  return (
    <ul>
      {items.map((x, i) => (
        <li key={i} style={{ background: i % 2 ? "#eee" : "transparent" }}>
          {x}
        </li>
      ))}
    </ul>
  );
};

// 4) text vs html (čitljiv raspored)
const Ex4 = () => {
  const [mode, setMode] = useState("text");
  return (
    <>
      <div style={{ minHeight: 24 }}>
        {mode === "text" ? "<b>React</b>" : <b>React</b>}
      </div>

      <div className="row">
        <button onClick={() => setMode("text")}>Kao tekst</button>
        <button onClick={() => setMode("html")}>Kao HTML</button>
      </div>
    </>
  );
};

// 5) Toggle klasa (klik na karticu)
const Ex5 = () => {
  const [active, setActive] = useState(false);
  return (
    <div
      onClick={() => setActive((a) => !a)}
      className={active ? "active" : ""}
      role="button"
    >
      Kartica (klikni)
    </div>
  );
};

// 6) Čitanje inputa + ispis
const Ex6 = () => {
  const [v, setV] = useState("");
  return (
    <>
      <div className="row">
        <input
          placeholder="Ime"
          value={v}
          onChange={(e) => setV(e.target.value)}
        />
        <button onClick={() => alert(v || "(prazno)")}>Prikaži</button>
      </div>
      <div>Out: {v}</div>
    </>
```

```
  );
};

// 7) attr/prop (disabled + title)
const Ex7 = () => {
  const [disabled, setDisabled] = useState(false);
  return (
    <>
      <button title={disabled ? "Zaključano" : "Klikni me"} disabled={disabled}>
        Akcija
      </button>

      <button onClick={() => setDisabled((d) => !d)}>Zaključaj/Otključaj</button>
    </>
  );
};

// 8) append/prepend
const Ex8 = () => {
  const [list, setList] = useState(["Start"]);
  return (
    <>
      <ul>
        {list.map((x, i) => (
          <li key={i}>{x}</li>
        ))}
      </ul>

      <div className="row">
        <button onClick={() => setList((l) => [...l, "Kraj"])}>Dodaj kraj</button>
        <button onClick={() => setList((l) => ["Početak", ...l])}>
          Dodaj početak
        </button>
      </div>
    </>
  );
};

// 9) before/after (ovde kao fiksni blokovi)
const Ex9 = () => (
  <>
    <p>Pre slike</p>
    <img alt="Slika" src="x.png" style={{ maxWidth: 120 }} />
    <p>Posle slike</p>
  </>
);

// 10) empty vs remove
const Ex10 = () => {
  const [exists, setExists] = useState(true);
  const [content, setContent] = useState(true);

  return (
    <>
      {exists && (
        <div id="panel">
          {content ? (
            <>
```

```
              <b>Tekst</b> i <i>deca</i>
            </>
          ) : null}
        </div>
      )}

      <div className="row">
        <button onClick={() => setContent(false)}>Empty</button>
        <button onClick={() => setExists(false)}>Remove</button>
      </div>
    </>
  );
};

// 11) Chaining + fade feedback
const Ex11 = () => {
  const [txt, setTxt] = useState("Čip");
  const [show, setShow] = useState(true);
  const [hit, setHit] = useState(false);

  const go = () => {
    setHit(true);
    setTxt("Obrađeno");
    setShow(false);
    setTimeout(() => setShow(true), 250);
  };

  return (
    <>
      <div className={hit ? "hit" : ""}>
        <FadeWrap show={show}>{txt}</FadeWrap>
      </div>

      <button onClick={go}>Anim</button>
    </>
  );
};

// 12) Delegacija-inspiracija (klik na li menja klasu)
const Ex12 = () => {
  const [items, setItems] = useState([]);
  const [sel, setSel] = useState({});

  const add = () =>
    setItems((a) => [...a, `Stavka ${a.length + 1}`]);

  return (
    <>
      <ul>
        {items.map((x, i) => (
          <li
            key={i}
            className={sel[i] ? "sel" : ""}
            onClick={() => setSel((s) => ({ ...s, [i]: !s[i] }))}
          >
            {x}
          </li>
        ))}
```

```
        </ul>

        <button onClick={add}>Dodaj stavku</button>
      </>
    );
  };

  // 13) Enter pokreće isto kao klik
  const Ex13 = () => {
    const [q, setQ] = useState("");
    const [out, setOut] = useState("");

    const go = () => setOut(`Tražim: ${q}`);

    return (
      <>
        <div className="row">
          <input
            placeholder="Upit"
            value={q}
            onChange={(e) => setQ(e.target.value)}
            onKeyDown={(e) => {
              if (e.key === "Enter") go();
            }}
          />
          <button onClick={go}>Traži</button>
        </div>

        <div>{out}</div>
      </>
    );
  };

  // 14) preventDefault
  const Ex14 = () => (
    <a
      href="https://example.com"
      onClick={(e) => {
        e.preventDefault();
        alert("Slanje kroz React!");
      }}
    >
      Pošalji
    </a>
  );

  // 15) stopPropagation
  const Ex15 = () => (
    <div
      onClick={() => alert("Kartica klik")}
      style={{ padding: 10, border: "1px solid #aaa" }}
    >
      Kartica{" "}
      <button
        onClick={(e) => {
          e.stopPropagation();
          alert("Dugme klik");
        }}
```

```
      >
        Dugme
      </button>
    </div>
  );

  // 16) parent/children flavor
  const Ex16 = () => {
    const [hit, setHit] = useState(false);
    const bg = hit ? "#eef6ff" : "transparent";

    return (
      <div
        style={{
          border: hit ? "1px solid #4a90e2" : "1px solid transparent",
          padding: 6,
        }}
      >
        <span
          className="hit"
          style={{ background: bg, cursor: "pointer" }}
          onClick={() => setHit(true)}
        >
          Klikni
        </span>{" "}
        <i style={{ background: bg }}>još</i>
      </div>
    );
  };

  // 17) closest/siblings flavor
  const Ex17 = () => {
    const [dim, setDim] = useState([1, 1, 1]);
    const clickIdx = (i) =>
      setDim((d) => d.map((_, j) => (j === i ? 1 : 0.6)));

    return (
      <ul className="row" style={{ listStyle: "none", padding: 0, gap: 8 }}>
        {["A", "B", "C"].map((x, i) => (
          <li
            key={i}
            style={{ opacity: dim[i], cursor: "pointer" }}
            onClick={() => clickIdx(i)}
          >
            {x}
          </li>
        ))}
      </ul>
    );
  };

  // 18) serialize (bez mreže)
  const Ex18 = () => {
    const outRef = useRef();

    const onSubmit = (e) => {
      e.preventDefault();
      outRef.current.textContent = serializeForm(e.currentTarget);
```

```
  };

  return (
    <>
      <form onSubmit={onSubmit}>
        <input name="ime" placeholder="ime" />
        <input name="grad" placeholder="grad" />
        <button>OK</button>
      </form>

      <pre ref={outRef} />
    </>
  );
};

// 19) Minimalna validacija
const Ex19 = () => {
  const [email, setEmail] = useState("");
  const [msg, setMsg] = useState("");
  const [bad, setBad] = useState(false);

  const save = () => {
    if (!email.trim()) {
      setBad(true);
      setMsg("Email je obavezan");
    } else {
      setBad(false);
      setMsg("OK");
    }
  };

  return (
    <>
      <div className="row">
        <input
          className={bad ? "err" : ""}
          placeholder="Email"
          value={email}
          onChange={(e) => setEmail(e.target.value)}
        />
        <button onClick={save}>Sačuvaj</button>
      </div>

      <div>{msg}</div>
    </>
  );
};

// 20) show/hide
const Ex20 = () => {
  const [show, setShow] = useState(true);
  return (
    <>
      <button onClick={() => setShow((s) => !s)}>Prikaži/Sakrij</button>

      {show && (
        <div style={{ padding: 10, background: "#fafafa" }}>Panel</div>
      )}
```

```
    </>
  );
};

// 21) fade hover (CSS transition)
const Ex21 = () => {
  const [over, setOver] = useState(false);
  return (
    <img
      alt="x"
      src="x.png"
      style={{
        opacity: over ? 1 : 0.6,
        maxWidth: 120,
        transition: "opacity 150ms",
      }}
      onMouseEnter={() => setOver(true)}
      onMouseLeave={() => setOver(false)}
    />
  );
};

// 22) accordion
const Ex22 = () => {
  const [open, setOpen] = useState(false);
  return (
    <>
      <h3
        className="hdr"
        style={{ cursor: "pointer" }}
        onClick={() => setOpen((o) => !o)}
      >
        Naslov
      </h3>

      {open && <div className="cnt">Sadržaj</div>}
    </>
  );
};

// 23) animate (width)
const Ex23 = () => {
  const [w, setW] = useState(100);
  return (
    <>
      <div
        style={{
          width: w,
          height: 20,
          background: "#ddd",
          transition: "width .4s",
        }}
      />
      <button onClick={() => setW(250)}>Rasti</button>
    </>
  );
};
```

```
// 24) checkbox master
const Ex24 = () => {
  const [all, setAll] = useState(false);
  const [A, setA] = useState(false);
  const [B, setB] = useState(false);

  useEffect(() => {
    setA(all);
    setB(all);
  }, [all]);

  return (
    <>
      <label>
        <input
          type="checkbox"
          checked={all}
          onChange={(e) => setAll(e.target.checked)}
        />{" "}
        Označi sve
      </label>

      <div>
        <label>
          <input
            type="checkbox"
            checked={A}
            onChange={(e) => setA(e.target.checked)}
          />{" "}
          A
        </label>
        <label style={{ marginLeft: 8 }}>
          <input
            type="checkbox"
            checked={B}
            onChange={(e) => setB(e.target.checked)}
          />{" "}
          B
        </label>
      </div>
    </>
  );
};

// 25) debounce (setTimeout)
const Ex25 = () => {
  const [raw, setRaw] = useState("");
  const [deb, setDeb] = useState("");

  useEffect(() => {
    const t = setTimeout(() => setDeb(raw), 300);
    return () => clearTimeout(t);
  }, [raw]);

  return (
    <>
      <input
        value={raw}
```

```
        onChange={(e) => setRaw(e.target.value)}
        placeholder="kucaj..."
      />
      <div className="muted">debounced: {deb}</div>
    </>
  );
};

// ========================================================================
// 26-30  |  Dimenzije, pozicija, filtriranje, klon, attr-funkcija
// ========================================================================

// 26) width/innerWidth/outerWidth (approx in React)
const Ex26 = () => {
  const ref = useRef();
  const [txt, setTxt] = useState("");

  useEffect(() => {
    const el = ref.current;
    const width = el.clientWidth;
    const innerWidth = el.clientWidth;
    const outerWidth = el.offsetWidth;
    setTxt(
      `width=${width} innerWidth=${innerWidth} outerWidth=${outerWidth}`
    );
  }, []);

  return (
    <>
      <div
        ref={ref}
        style={{ width: 120, padding: 10, border: "5px solid #ccc" }}
      >
        .
      </div>
      <pre>{txt}</pre>
    </>
  );
};

// 27) offset/position flavor
const Ex27 = () => {
  const ref = useRef();
  const [info, setInfo] = useState("");

  useEffect(() => {
    const el = ref.current;
    const off = el.getBoundingClientRect();
    const pos = { left: el.offsetLeft, top: el.offsetTop };
    setInfo("offset=" + JSON.stringify(off) + " position=" + JSON.stringify(pos));
  }, []);

  return (
    <>
      <div style={{ position: "relative", marginTop: 40 }}>
        <div ref={ref} style={{ position: "absolute", left: 30, top: 20 }}>
          •
        </div>
```

```
      </div>
      <pre>{info}</pre>
    </>
  );
};

// 28) filter/not
const Ex28 = () => {
  const arr = ["Java", "CSS", "HTML", "Scala"];
  return (
    <ul>
      {arr.map((x, i) => {
        const hasA = x.toLowerCase().includes("a");
        return (
          <li
            key={i}
            style={{
              background: hasA ? "#e6fcf5" : "transparent",
              opacity: hasA ? 1 : 0.5,
            }}
          >
            {x}
          </li>
        );
      })}
    </ul>
  );
};

// 29) clone (ovde: dupliramo niz i renderujemo)
const Ex29 = () => {
  const [els, setEls] = useState(["Tag"]);
  return (
    <>
      <div>
        {els.map((x, i) => (
          <div key={i}>{x}</div>
        ))}
      </div>

      <button onClick={() => setEls((e) => [...e, "Tag"])}>Dupliraj</button>
    </>
  );
};

// 30) attr funkcija (data-score)
const Ex30 = () => {
  const [score, setScore] = useState(0);
  return (
    <div onClick={() => setScore((s) => s + 10)} data-score={score}>
      Score (klikni) -- data-score: {score}
    </div>
  );
};

// ========================================================================
// 31-40  |  AJAX/Fetch (offline: mockApi)
// ========================================================================
```

```
// 31) GET statičkog JSON (mock)
const Ex31 = () => {
  const [out, setOut] = useState("...");
  useEffect(() => {
    mockApi.data().then((d) => setOut(d.title));
  }, []);
  return <div>{out}</div>;
};

// 32) POST forma (mock)
const Ex32 = () => {
  const [resp, setResp] = useState("");
  const onSubmit = (e) => {
    e.preventDefault();
    const fd = new FormData(e.currentTarget);
    const obj = Object.fromEntries(fd.entries());
    mockApi.login(obj).then((r) => setResp(JSON.stringify(r)));
  };
  return (
    <>
      <form onSubmit={onSubmit}>
        <input name="u" placeholder="user" />
        <input name="p" type="password" placeholder="pass" />
        <button>Prijava</button>
      </form>
      <pre>{resp}</pre>
    </>
  );
};

// 33) timeout/error handler
const Ex33 = () => {
  const [out, setOut] = useState("Učitavam...");
  useEffect(() => {
    let aborted = false;
    const t = setTimeout(() => {
      aborted = true;
      setOut("Greška ili timeout");
    }, 2000);

    mockApi
      .data()
      .then((d) => {
        if (!aborted) {
          setOut(JSON.stringify(d));
          clearTimeout(t);
        }
      })
      .catch(() => {
        if (!aborted) setOut("Greška");
      });

    return () => clearTimeout(t);
  }, []);
  return <pre>{out}</pre>;
};
```

```
// 34) Render liste iz JSON-a
const Ex34 = () => {
  const [list, setList] = useState([]);
  useEffect(() => {
    mockApi.users().then(setList);
  }, []);
  return (
    <ul>
      {list.map((o, i) => (
        <li key={i}>{o.name}</li>
      ))}
    </ul>
  );
};

// 35) Loader/spinner
const Ex35 = () => {
  const [loading, setLoading] = useState(false);
  const [data, setData] = useState(null);

  useEffect(() => {
    setLoading(true);
    mockApi
      .items()
      .then((d) => setData(d.length))
      .finally(() => setLoading(false));
  }, []);

  return (
    <>
      <div id="spin" style={{ display: loading ? "block" : "none" }}>
        Učitavam...
      </div>
      <div id="data">{data ? `${data} stavki` : ""}</div>
    </>
  );
};

// 36) localStorage keš
const Ex36 = () => {
  const [out, setOut] = useState("");
  useEffect(() => {
    const k = "itemsCache";
    const cached = localStorage.getItem(k);
    if (cached) {
      setOut("Iz keša: " + JSON.parse(cached).length);
    } else {
      mockApi.items().then((d) => {
        localStorage.setItem(k, JSON.stringify(d));
        setOut("Sveže: " + d.length);
      });
    }
  }, []);
  return <div>{out}</div>;
};

// 37) Form-data => JSON, pa "save"
const Ex37 = () => {
```

```
  const [resp, setResp] = useState("");
  const onSubmit = (e) => {
    e.preventDefault();
    const obj = Object.fromEntries(new FormData(e.currentTarget).entries());
    setResp(JSON.stringify(obj));
    mockApi.save(obj);
  };
  return (
    <>
      <form onSubmit={onSubmit}>
        <input name="a" />
        <input name="b" />
        <button>OK</button>
      </form>
      <pre>{resp}</pre>
    </>
  );
};

// 38) Greška (fail)
const Ex38 = () => {
  const [out, setOut] = useState("");
  useEffect(() => {
    mockApi.nope().catch((err) => {
      setOut("Greška: " + (err.message || ""));
    });
  }, []);
  return <div style={{ color: "#e03131" }}>{out}</div>;
};

// 39) Abort/timeout (prekinuto)
const Ex39 = () => {
  const [out, setOut] = useState("...");
  useEffect(() => {
    let aborted = false;
    const t = setTimeout(() => {
      aborted = true;
      setOut("Prekinuto");
    }, 300);

    mockApi
      .slow()
      .then(() => {
        if (!aborted) {
          setOut("OK");
          clearTimeout(t);
        }
      })
      .catch(() => {
        setOut("Prekinuto");
      });

    return () => clearTimeout(t);
  }, []);
  return <pre>{out}</pre>;
};

// 40) Mini templating
```

```
const Ex40 = () => {
  return (
    <div
      dangerouslySetInnerHTML={{
        __html: tpl("<p>Zdravo, {{name}}!</p>", { name: "Svet" }),
      }}
    />
  );
};

// ============================================================================
// 41-50  |  "Plugin" stil, A11y, Promise, SPA
// ============================================================================

// 41) Blink util (imperativno)
const Ex41 = () => {
  const ref = useRef();

  const blink = (el, times = 3, speed = 150) => {
    let i = 0;
    (function step() {
      el.style.transition = `opacity ${speed}ms`;
      el.style.opacity = "0";
      setTimeout(() => {
        el.style.opacity = "1";
        if (++i < times) setTimeout(step, speed + 10);
      }, speed + 10);
    })();
  };

  useEffect(() => {
    blink(ref.current, 4, 100);
  }, []);

  return <div ref={ref}>Treperim</div>;
};

// 42) "Highlight" efekat sa opcijama (imperativno)
const Ex42 = () => {
  const ref = useRef();

  useEffect(() => {
    const el = ref.current;
    const old = getComputedStyle(el).backgroundColor;
    el.style.transition = "background-color 200ms, opacity 200ms";
    el.style.backgroundColor = "#c0ebff";
    el.style.opacity = "0.9";

    setTimeout(() => {
      el.style.opacity = "1";
      setTimeout(() => {
        el.style.backgroundColor = old;
      }, 200);
    }, 200);
  }, []);

  return <p ref={ref}>Tekst</p>;
};
```

```
// 43) "Event namespacing" analog (ručno dodavanje/uklanjanje)
const Ex43 = () => {
  const [log, setLog] = useState([]);

  const handlerA = () => setLog((l) => [...l, "A"]);
  const handlerB = () => setLog((l) => [...l, "B"]);

  useEffect(() => {
    const btn = qs("#nsBtn");
    btn.addEventListener("click", handlerA);
    btn.addEventListener("click", handlerB);
    return () => {
      btn.removeEventListener("click", handlerA);
      btn.removeEventListener("click", handlerB);
    };
  }, []);

  return (
    <>
      <button id="nsBtn">Klik</button>
      <button onClick={() => qs("#nsBtn").removeEventListener("click", handlerA)}>
        Ukloni A
      </button>
      <div>{log.join(", ")}</div>
    </>
  );
};

// 44) Custom događaj
const Ex44 = () => {
  const [out, setOut] = useState("");

  useEffect(() => {
    const el = qs("#jobX");
    const onDone = (e) => setOut("Gotovo: " + e.detail.msg);
    el.addEventListener("done", onDone);

    setTimeout(
      () => el.dispatchEvent(new CustomEvent("done", { detail: { msg: "uspeh" ...
↪ } })),
      200
    );

    return () => el.removeEventListener("done", onDone);
  }, []);

  return <div id="jobX">{out}</div>;
};

// 45) Delegacija i performance (1 handler)
const Ex45 = () => {
  const [txt, setTxt] = useState("");
  const wrapRef = useRef();

  useEffect(() => {
    const wrap = wrapRef.current;
    wrap.innerHTML = Array.from(
```

```
        { length: 1000 },
        (_, i) => `<button class="b">#${i}</button>`
      ).join("");

      const onClick = (e) => {
        const btn = e.target.closest(".b");
        if (btn) setTxt(btn.textContent);
      };

      wrap.addEventListener("click", onClick);
      return () => wrap.removeEventListener("click", onClick);
    }, []);

    return (
      <>
        <div ref={wrapRef} id="wrap" />
        <pre id="log">{txt}</pre>
      </>
    );
  };

  // 46) A11y: Enter/Space aktivira "karticu"
  const Ex46 = () => {
    const [pressed, setPressed] = useState(false);

    return (
      <div
        id="cardA11y"
        tabIndex="0"
        role="button"
        aria-pressed={pressed}
        onKeyDown={(e) => {
          if (e.key === "Enter" || e.key === " ") {
            e.preventDefault();
            e.currentTarget.click();
          }
        }}
        onClick={() => setPressed((p) => !p)}
      >
        Otvori
      </div>
    );
  };

  // 47) Promise/Deferred flavor
  const Ex47 = () => {
    const [out, setOut] = useState("");
    useEffect(() => {
      new Promise((res) => setTimeout(() => res("OK"), 200)).then((v) => setOut(v));
    }, []);
    return <pre>{out}</pre>;
  };

  // 48) Fetch analog (mockApi.users) + render
  const Ex48 = () => {
    const [list, setList] = useState([]);
    useEffect(() => {
      mockApi.users().then(setList);
```

```
  }, []);
  return (
    <ul id="u">
      {list.map((x, i) => (
        <li key={i}>{x.name}</li>
      ))}
    </ul>
  );
};

// 49) Debounce util + keyup
const Ex49 = () => {
  const [raw, setRaw] = useState("");
  const [out, setOut] = useState("");

  const debounce = (fn, wait) => {
    let t;
    return (...a) => {
      clearTimeout(t);
      t = setTimeout(() => fn(...a), wait);
    };
  };

  const onKey = useMemo(() => debounce((v) => setOut(v), 300), []);

  return (
    <>
      <input id="search2" onKeyUp={(e) => onKey(e.target.value)} />
      <div id="outX">{out}</div>
    </>
  );
};

// 50) Mini SPA (hashchange)
const Ex50 = () => {
  const [view, setView] = useState(location.hash || "#home");

  useEffect(() => {
    const f = () => setView(location.hash || "#home");
    addEventListener("hashchange", f);
    return () => removeEventListener("hashchange", f);
  }, []);

  return (
    <>
      <nav>
        <a href="#home">Home</a> | <a href="#about">About</a>
      </nav>

      <div id="view">{view === "#about" ? "O aplikaciji" : "Dobrodošli"}</div>
    </>
  );
};

// ========================================================================
// 51-55  |  Bonus
// ========================================================================
```

```
// 51) Dinamički <select>
const Ex51 = () => {
  const arr = ["Beograd", "Novi Sad", "Niš"];
  const [v, setV] = useState(arr[0]);
  return (
    <>
      <select id="sel" value={v} onChange={(e) => setV(e.target.value)}>
        {arr.map((x) => (
          <option key={x}>{x}</option>
        ))}
      </select>
      <div id="cho">{v}</div>
    </>
  );
};

// 52) Brojač karaktera (max 50)
const Ex52 = () => {
  const [v, setV] = useState("");
  const left = 50 - v.length;
  return (
    <>
      <textarea
        id="t"
        maxLength={50}
        value={v}
        onChange={(e) => setV(e.target.value)}
      />
      <div id="c" style={{ color: left === 0 ? "#e03131" : "inherit" }}>
        {left}
      </div>
    </>
  );
};

// 53) Sortiranje liste
const Ex53 = () => {
  const [list, setList] = useState(["banan", "ana", "cuk"]);
  const sort = () => setList((l) => [...l].sort((a, b) => a.localeCompare(b)));
  return (
    <>
      <ul id="l2">
        {list.map((x, i) => (
          <li key={i}>{x}</li>
        ))}
      </ul>
      <button id="sort" onClick={sort}>
        Sortiraj
      </button>
    </>
  );
};

// 54) Drag toggle (klasa pri držanju)
const Ex54 = () => {
  const [drag, setDrag] = useState(false);
  return (
    <div
```

```
        id="drag"
        className={drag ? "dragging" : ""}
        style={{ width: 100, height: 40, background: "#ddd" }}
        onMouseDown={() => setDrag(true)}
        onMouseUp={() => setDrag(false)}
        onMouseLeave={() => setDrag(false)}
      >
        Drži
      </div>
    );
  };

  // 55) Clipboard (kopiranje teksta)
  const Ex55 = () => {
    const [val, setVal] = useState("Kopiraj me");
    return (
      <>
        <input id="clip" value={val} onChange={(e) => setVal(e.target.value)} />
        <button
          id="cpy"
          onClick={async () => {
            try {
              await navigator.clipboard.writeText(val);
              alert("Kopirano");
            } catch {
              alert("Neuspeh");
            }
          }}
        >
          Kopiraj
        </button>
      </>
    );
  };

  // ============================================================================
  // 56-58  |  Dodatni praktični primeri
  // ============================================================================

  // 56) Master checkbox sa "indeterminate"
  const Ex56 = () => {
    const masterRef = useRef();
    const [checks, setChecks] = useState([false, false, false]);

    useEffect(() => {
      const all = checks.every(Boolean);
      const none = checks.every((x) => !x);
      const m = masterRef.current;
      m.checked = all;
      m.indeterminate = !all && !none;
    }, [checks]);

    const toggleMaster = (e) => {
      const v = e.target.checked;
      setChecks([v, v, v]);
    };

    const toggle = (i) =>
```

```
    setChecks((cs) => cs.map((c, j) => (j === i ? !c : c)));

  return (
    <>
      <label>
        <input type="checkbox" ref={masterRef} onChange={toggleMaster} /> Označi
        sve (sa indeterminate)
      </label>

      <div>
        {checks.map((c, i) => (
          <label key={i} style={{ marginRight: 8 }}>
            <input
              type="checkbox"
              checked={c}
              onChange={() => toggle(i)}
            />{" "}
            {String.fromCharCode(65 + i)}
          </label>
        ))}
      </div>
    </>
  );
};

// 57) Pregled slike iz <input type="file">
const Ex57 = () => {
  const [src, setSrc] = useState(null);

  const onFile = (e) => {
    const f = e.target.files?.[0];
    if (!f) return;
    const r = new FileReader();
    r.onload = () => setSrc(r.result);
    r.readAsDataURL(f);
  };

  return (
    <>
      <input type="file" accept="image/*" onChange={onFile} />
      {src && (
        <img
          alt="preview"
          src={src}
          style={{ maxWidth: 180, display: "block", marginTop: 8 }}
        />
      )}
    </>
  );
};

// 58) Tema (svetla/tamna) + localStorage persist
const ThemeCtx = createContext();

const ThemeProvider = ({ children }) => {
  const [s, setS] = useState(() => localStorage.getItem("theme") || "light");

  useEffect(() => {
```

```
    localStorage.setItem("theme", s);
    document.documentElement.style.background =
      s === "dark" ? "#0b0f17" : "white";
    document.body.style.color = s === "dark" ? "#e6eefb" : "black";
  }, [s]);

  return (
    <ThemeCtx.Provider value={{ theme: s, setTheme: setS }}>
      {children}
    </ThemeCtx.Provider>
  );
};

const Ex58Inner = () => {
  const { theme, setTheme } = useContext(ThemeCtx);
  return (
    <>
      <div>
        Aktivna tema: <b>{theme}</b>
      </div>
      <button onClick={() => setTheme(theme === "light" ? "dark" : "light")}>
        Prebaci temu
      </button>
    </>
  );
};

const Ex58 = () => (
  <ThemeProvider>
    <Ex58Inner />
  </ThemeProvider>
);

// ========================================================================
// Montiranje svih 58
// ========================================================================
const mount = (id, el) =>
  ReactDOM.createRoot(document.getElementById(id)).render(el);

mount("ex1", <Ex1 />);
mount("ex2", <Ex2 />);
mount("ex3", <Ex3 />);
mount("ex4", <Ex4 />);
mount("ex5", <Ex5 />);
mount("ex6", <Ex6 />);
mount("ex7", <Ex7 />);
mount("ex8", <Ex8 />);
mount("ex9", <Ex9 />);
mount("ex10", <Ex10 />);

mount("ex11", <Ex11 />);
mount("ex12", <Ex12 />);
mount("ex13", <Ex13 />);
mount("ex14", <Ex14 />);
mount("ex15", <Ex15 />);
mount("ex16", <Ex16 />);
mount("ex17", <Ex17 />);
mount("ex18", <Ex18 />);
```

```
    mount("ex19", <Ex19 />);
    mount("ex20", <Ex20 />);

    mount("ex21", <Ex21 />);
    mount("ex22", <Ex22 />);
    mount("ex23", <Ex23 />);
    mount("ex24", <Ex24 />);
    mount("ex25", <Ex25 />);
    mount("ex26", <Ex26 />);
    mount("ex27", <Ex27 />);
    mount("ex28", <Ex28 />);
    mount("ex29", <Ex29 />);
    mount("ex30", <Ex30 />);

    mount("ex31", <Ex31 />);
    mount("ex32", <Ex32 />);
    mount("ex33", <Ex33 />);
    mount("ex34", <Ex34 />);
    mount("ex35", <Ex35 />);
    mount("ex36", <Ex36 />);
    mount("ex37", <Ex37 />);
    mount("ex38", <Ex38 />);
    mount("ex39", <Ex39 />);
    mount("ex40", <Ex40 />);

    mount("ex41", <Ex41 />);
    mount("ex42", <Ex42 />);
    mount("ex43", <Ex43 />);
    mount("ex44", <Ex44 />);
    mount("ex45", <Ex45 />);
    mount("ex46", <Ex46 />);
    mount("ex47", <Ex47 />);
    mount("ex48", <Ex48 />);
    mount("ex49", <Ex49 />);
    mount("ex50", <Ex50 />);

    mount("ex51", <Ex51 />);
    mount("ex52", <Ex52 />);
    mount("ex53", <Ex53 />);
    mount("ex54", <Ex54 />);
    mount("ex55", <Ex55 />);
    mount("ex56", <Ex56 />);
    mount("ex57", <Ex57 />);
    mount("ex58", <Ex58 />);
  </script>
```

Offline napomena:

Preuzmi `react.development.js`, `react-dom.development.js` i `babel.min.js` i stavi ih u `./libs`. Za AJAX primere koristi se `mockApi` pa internet nije potreban. Slika `x.png` je placeholder — stavi lokalnu sliku u isti folder ili promeni putanju.