

Mini-aplikacja JS: quiz lub lista zakupów

Czas: 2 × 45 min

praca indywidualna (wariant do wyboru)

Cel: Zbudowanie małej aplikacji jednostronicowej, w której stan przechowywany jest w tablicy obiektów, a widok odtwarza jedna funkcja renderująca wywoływana po każdej zmianie danych.

Po zajęciach potrafisz:

- projektuje dane aplikacji jako tablicę obiektów i zmienne stanu
- pisze funkcję renderującą, która buduje widok z danych przez `createElement` i `textContent`
- obsługuje zdarzenia kliknięcia i formularza, zmieniając najpierw dane, a potem widok
- stosuje `filter`, `find` i `forEach` do operacji na danych aplikacji
- testuje aplikację według przygotowanych scenariuszy i dokumentuje wyniki

Sprzęt i materiały

- komputer z Windows 11 i przeglądarką Chrome, Edge lub Firefox
- folder `r1_27_start` z Pliki_do_zadan (podfoldery `quiz` i `lista` z gotowym HTML i CSS)

Oprogramowanie

- **Visual Studio Code** — rozszerzenie Live Server
code.visualstudio.com
- **Przeglądarka z narzędziami F12** — Console, Elements, Application → Local Storage (wariant B)

Przygotowanie przed zajęciami

- Powtórz zadania 24–26: tablice obiektów, `createElement`, `addEventListener`, `preventDefault`.
- Wybierz wariant: A — quiz z punktacją, B — lista zakupów. Skopiuj odpowiedni podfolder jako `zad27` do folderu `nazwisko_imie`.

Bezpieczeństwo

- Teksty z danych i od użytkownika wstawiaj przez `textContent` — w quizie odpowiedzi zawierają znaczniki (np. `<img>`), które mają się wyświetlić jako tekst.
- W `localStorage` zapisuj tylko dane aplikacji; nie przechowuj tam haseł ani danych osobowych.

Przebieg zadania

1 Zasada działania (oba warianty)

Zdarzenie → zmiana danych w tablicy → wywołanie funkcji renderującej. Nie zmieniaj ręcznie pojedynczych elementów listy — po każdej zmianie odtwórz widok z danych.

```
let dane = [ /* obiekty */ ];
function render() {
  kontener.textContent = "";           // 1. wyczyść widok
  dane.forEach(d => { /* 2. utwórz element z danych */ });
}
przycisk.addEventListener("click", () => {
  /* 3. zmień dane */
  render();                           // 4. odśwież widok
});
```

1. Otwórz wybrany podfolder w VS Code i uruchom Live Server. Obejrzyj HTML i CSS — wypisz w sprawozdaniu identyfikatory i klasy, których użyjesz (np. `#odpowiedzi`, `.dobra`, `.zla` albo `#lista`, `.kupione`, `.aktywny`).
2. Wpisz w komentarzu `skrypt.js` imię, nazwisko i wybrany wariant.

2 Wariant A — quiz „Jak działa WWW?”

Pytanie to obiekt `{ tresc, odpowiedzi: [4 teksty], poprawna: indeks }`.

1. Q1. Utwórz tablicę `pytania` z co najmniej 6 pytaniami o HTML, CSS, JavaScript i sieci (własne, z jedną poprawną odpowiedzią). Co najmniej jedna odpowiedź ma zawierać znacznik HTML, np. `<img>`.
2. Q2. Zadeklaruj zmienne stanu: bieżący numer pytania, liczba punktów, historia odpowiedzi.
3. Q3. Napisz `pokazPytanie()`: tekst „Pytanie 2 z 6”, szerokość paska `#pasek-wypelnienie` w %, treść pytania i 4 przyciski `.odp` utworzone w pętli.
4. Q4. Po kliknięciu odpowiedzi: zablokuj wszystkie przyciski (jedna próba), zaznacz poprawną klasą `dobra`, a błędnie wybraną klasą `zla`, wypisz komunikat w `#informacja`, dolicz punkt i odblokuj „Dalej”.
5. Q5–Q6. „Dalej” przechodzi do kolejnego pytania; po ostatnim pokaż `#ekran-wynik` (atrybut `hidden`) z wynikiem „4 / 6 (67%)”, oceną słowną (100% / co najmniej 50% / poniżej 50%) i listą pytań z oznaczeniem ✓/✗.
6. Q7–Q8. „Zagraj ponownie” zeruje stan i zaczyna od nowa. Pytania mają być za każdym razem w innej kolejności — napisz funkcję `potasuj(tab)` (algorytm Fishera–Yatesa z prezentacji lub własny), która zwraca przetasowaną kopię.

3 Wariant B — lista zakupów

Produkt to obiekt `{ id, nazwa, ilosc, kupione }`. Jako `id` możesz użyć `Date.now()`.

1. L1. Utwórz tablicę `produkty` z 3 przykładowymi pozycjami (jedna kupiona) i zmienną `filtr`.
2. L2. Napisz `render()`: dla każdego produktu element `li` z checkboxem, tekstem „Chleb × 1” i przyciskiem × (`.usun`); kupione mają klasę `kupione`. Pod listą licznik „Do kupienia: 2 z 3”.
3. L3. Formularz dodaje produkt (`submit` + `preventDefault`). Walidacja: nazwa niepusta, ilość całkowita 1–99 — komunikat w `#blad`. Po dodaniu wyczyść pole i ustaw w nim kursor.
4. L4–L5. Odhaczanie i usuwanie obsług **jednym** zdarzeniem `click` na `#lista` (delegowanie: `e.target.closest("li")`, `e.target.matches(...)`) — zmień dane i wywołaj `render()`.
5. L6–L7. Przycisk „Usuń kupione” (`filter`). Dodanie produktu o tej samej nazwie (bez względu na wielkość liter) zwiększa ilość zamiast tworzyć duplikat (`find`).
6. L8. Filtry Wszystkie / Do kupienia / Kupione (klasa `aktywny` na wybranym) oraz zapis listy w `localStorage` (`JSON.stringify` / `JSON.parse`) — po odświeżeniu lista ma zostać.

4 Testy i dokumentacja

1. T1. Wykonaj co najmniej 6 testów swojego wariantu i zapisz je w tabeli. Quiz: odpowiedź dobra, zła, próba drugiego kliknięcia, wynik końcowy, ponowna gra (inna kolejność), odpowiedź ze znacznikiem `<img>` wyświetlona jako tekst. Lista: pusta nazwa, ilość 150, dodanie „chleb” przy istniejącym „Chleb”, odhaczenie, filtr, odświeżenie strony.
2. T2. Sprawdź, czy konsola nie pokazuje błędów, zrób 3 zrzuty działającej aplikacji i — jeśli pracujesz z Git — zatwierdź gotową wersję (commit).

W sprawozdaniu umieść

- ☐ wybrany wariant i lista użytych identyfikatorów i klas
- ☐ tabela wykonanych poleceń (Q1–Q8 albo L1–L8) z fragmentami kodu
- ☐ tabela testów T1 (co najmniej 6 testów)
- ☐ 3 zrzuty ekranu działającej aplikacji
- ☐ folder `zad27` z plikami aplikacji
- ☐ odpowiedzi na pytania kontrolne

Pytania kontrolne

1. Dlaczego po każdej zmianie danych odtwarzamy cały widok funkcją `render()`, zamiast poprawiać pojedyncze elementy na stronie?
2. Quiz: dlaczego tasujemy kopię tablicy `pytania`, a nie oryginał? Lista: na czym polega delegowanie zdarzeń i jaką ma przewagę nad dodawaniem obsługi do każdego przycisku ×?
3. Dlaczego do `localStorage` zapisujemy `JSON.stringify(dane)`, a nie samą tablicę? Co zwróci `getItem`, gdy klucza nie ma?
4. Gdzie w Twojej aplikacji `textContent` chroni przed błędnym wyświetleniem danych?

Kryteria oceny

Kryterium	Punkty
Dane w tablicy obiektów (min. 6 pytań albo 3 produkty) i zmienne stanu	2
Funkcja renderująca budująca widok z danych (createElement, textContent), licznik/postęp	4
Podstawowa obsługa zdarzeń: Q3–Q5 albo L3–L5 działa bez błędów	4
Funkcje rozszerzone: Q6–Q8 albo L6–L8	4
Tabela testów T1 i zrzuty, konsola bez błędów	3
Czytelny kod z komentarzami i odpowiedzi na pytania kontrolne	3
Razem	20

Sprawozdanie przygotuj na wzorze dołączonym do zadania (plik .docx o tym samym numerze).