

Nowoczesny JavaScript w praktyce

Czas: 2 × 45 min

praca indywidualna

Cel: Przepisanie działającego, ale przestarzałego skryptu sklepu na współczesny JavaScript: moduły ES, szablony tekstowe, funkcje strzałkowe, destrukuryzację, spread, operatory `?.` i `??` oraz klasę.

Po zajęciach potrafisz:

- zastępuje `var` deklaracjami `let` / `const` i konkatenację szablony tekstowymi
- dzieli kod na moduły ES z `export` / `import` i uruchamia je przez `type="module"`
- stosuje destrukuryzację, spread, parametry domyślne oraz operatory `?.` i `??`
- pisze klasę z polem prywatnym, metodami i getterami
- porównuje działanie starej i nowej wersji tymi samymi testami

Sprzęt i materiały

- komputer z Windows 11
- folder `r2_02_start` z Pliki_do_zadan (`index.html`, `styl.css`, `stary.js`)

Oprogramowanie

- **Visual Studio Code** — rozszerzenie Live Server (podgląd przez `http://127.0.0.1:5500`)
code.visualstudio.com
- **Przeglądarka Chrome, Edge lub Firefox** — narzędzia deweloperskie F12 (Elementy, Konsola, Sieć, Aplikacja)
- **Git** — opcjonalnie: commit przed i po refaktoryzacji
git-scm.com/downloads

Przygotowanie przed zajęciami

- Przeglądaj wykład 2.01 (ES6+, moduły, klasy).

Bezpieczeństwo

- Nazwy produktów i opisy wstawiaj przez `textContent`; w szablonie HTML umieszczaj tylko liczby i stałe teksty.
- Moduły ES nie działają z adresu `file://` — otwieraj stronę wyłącznie przez Live Server (`http://127.0.0.1:5500`).

Przebieg zadania

1 Stara wersja i testy odniesienia

Najpierw sprawdź, jak działa stary kod — po przepisaniu wyniki muszą być takie same.

1. Skopiuj `r2_02_start` do folderu `nazwisko_imie_z02`, otwórz go w VS Code i uruchom Live Server. Jeśli używasz Gita, zrób commit „stara wersja”.
2. Wykonaj test odniesienia: dodaj do koszyka 2 × „Plecak Fjord 30 l” i 1 × „Namiot Polaris 2-os.”. Zapisz sumę. Wybierz kategorię „akcesoria” i zapisz liczbę kart.
3. Przeczytaj `stary.js` i wypisz w sprawozdaniu co najmniej 6 miejsc do unowocześnienia (np. `var`, pętla szukająca produktu, `rabat === undefined`).

2 Moduły: dane i narzędzia

Docelowa struktura: `dane.js`, `narzedzia.js`, `koszyk.js`, `main.js`. W `index.html` zamień `<script src="stary.js">` na `<script type="module" src="main.js"></script>`. W ścieżkach importu pisz `./` i rozszerzenie `.js`.

```
// narzedzia.js – przykład eksportu
export const zl = (kwota) => `${kwota.toFixed(2).replace(".", ",")} zł`;

// main.js – przykład importu
import { zl } from "./narzedzia.js";
```

- W `dane.js` wyeksportuj tablicę `produkty` (`export const`) oraz obiekt `nowosc`: Mata Puch, akcesoria, 119 zł, stan 5, id 9.
- W `narzedzia.js` wyeksportuj funkcje strzałkowe `zl(kwota)` i `cenaPoRabacie({ cena, rabat })` — z destrukuryzacją parametru i operatorem `??` zamiast warunku `=== undefined`.
- Dodaj eksport domyślny `kategorie(lista)`, który zwraca kategorie bez powtórzeń przy użyciu `Set` i spreadu (`[...new Set(...)]`) zamiast pętli z `indexOf`.

3 Klasa Koszyk

Koszyk przechowuje pozycje w prywatnym polu `#pozycje` typu `Map` (klucz — id produktu, wartość — `{ produkt, ilosc }`).

- W `koszyk.js` napisz i wyeksportuj klasę `Koszyk` z metodami `dodaj(produkt, ilosc = 1)`, `usun(id)`, `wyczysc()`. W `dodaj` odczytaj obecną ilość wyrażeniem `this.#pozycje.get(id)?.ilosc ?? 0`.
- Dodaj gettery: `pozycje` (tablica wartości z `Map`), `liczba` (suma sztuk) i `suma` (kwota po rabatach) — dwa ostatnie przez `reduce`.
- Gdy strona już działa (krok 4), dopisz tymczasowo na końcu `main.js` linię `console.log(koszyk.#pozycje);`, odśwież stronę, zanotuj błąd z konsoli F12 i usuń tę linię.

4 main.js — interfejs

Główny moduł importuje pozostałe i obsługuje stronę. Nie zmieniaj tablicy z `dane.js` — twórz kopie.

- Utwórz tablicę `produkty` jako `[...bazowe, { ...nowosc, rabat: 5 }]` (spread tablicy i obiektu). Wyświetlaj karty funkcją `karta({ id, nazwa, opis, stan, rabat, ...reszta })` z szablonem tekstowym; opis wstaw przez `textContent` z `opis ?? ""`.
- Obsłuż filtr kategorii i nową listę „Sortuj” (cena rosnąco, nazwa A–Z): sortuj kopię `[...lista].sort(...)`, nazwy porównuj `localeCompare(..., "pl")`.
- Kliknięcia „Do koszyka” obsłuż jednym nasłuchem na `#produkty` (delegowanie zdarzeń, `e.target.dataset.id`). W koszyku pokaż przycisk `×` usuwający pozycję oraz działający przycisk „Wyczyść koszyk”; pod sumą wyświetl liczbę sztuk.
- Powtórz test odniesienia z kroku 1 — suma musi być identyczna. Sprawdź, że „Mata Puch” ma cenę po rabacie 5%. Zrób zrzut ekranu i (opcjonalnie) commit „wersja ES6+”.

W sprawozdaniu umieść

- ☐ wynik testu odniesienia dla starej i nowej wersji (suma koszyka, liczba akcesoriów)
- ☐ tabela „stary kod → nowy kod” z co najmniej 6 zmianami
- ☐ kod klasy `Koszyk` i komunikat przy próbie odczytu pola prywatnego
- ☐ zrzut ekranu nowej wersji z koszykiem i sortowaniem
- ☐ spakowany folder `nazwisko_imie_z02` (lub link do repozytorium)

Pytania kontrolne

1. Czym różni się `??` od `||`? Podaj przykład z produktem, który ma `rabat: 0`.
2. Dlaczego strona z modułami nie działa po otwarciu pliku dwuklikiem (adres `file://`)?
3. Czym różni się eksport nazwany od domyślnego i jak wygląda import każdego z nich?
4. Dlaczego przed `sort()` robimy kopię tablicy `[...lista]`?

Kryteria oceny

Kryterium	Punkty
Analiza starego kodu i test odniesienia (min. 6 miejsc do zmiany)	3
Moduły dane.js i narzedzia.js: eksporty nazwane i domyślny, Set, ??, destrukuryzacja	4
Klasa Koszyk: pole prywatne, Map, 3 metody, 3 gettery, poprawna suma	5
main.js: szablony, spread, filtr i sortowanie kopii, delegowanie zdarzeń, usuwanie z koszyka	5
Zgodność wyników starej i nowej wersji, zrzuty, odpowiedzi na pytania	3
Razem	20

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