

DOM — modyfikowanie strony

Czas: 2 × 45 min

praca indywidualna

Cel: Zmienianie treści, klas, atrybutów i stylów elementów strony w reakcji na kliknięcia oraz bezpieczne wstawianie tekstu wpisanego przez użytkownika.

Po zajęciach potrafisz:

- wybiera elementy metodami `querySelector`, `querySelectorAll` i `getElementById`
- zmienia tekst (`textContent`), atrybuty (`src`, `alt`, `data-*`) i style elementów
- przełącza klasy CSS metodami `classList.toggle`, `add`, `remove`, `contains`
- tworzy nowe elementy (`createElement`, `append`) i aktualizuje licznik na żywo
- wyjaśnia, dlaczego dane od użytkownika wstawia się przez `textContent`, a nie `innerHTML`

Sprzęt i materiały

- komputer z Windows 11 i przeglądarką Chrome, Edge lub Firefox
- folder `r1_25_start` z Pliki_do_zadan (`index.html`, `styl.css`, pusty `skrypt.js`, folder `img`)

Oprogramowanie

- **Visual Studio Code** — rozszerzenie Live Server
code.visualstudio.com
- **Przeglądarka z narzędziami F12** — Elements — obserwuj zmiany klas i atrybutów na żywo; Application → Local Storage

Przygotowanie przed zajęciami

- Przypomnij sobie z prezentacji 1.17 metody wyboru elementów, `classList` i różnicę między `textContent` a `innerHTML`.
- Skopiuj folder `r1_25_start` jako `zad25` do swojego folderu `nazwisko_imie`.

Bezpieczeństwo

- Tekst wpisany przez użytkownika wstawiaj wyłącznie przez `textContent`. W D9 porównujesz obie metody tylko na zwykłym znaczniku formatowania `<b>` — nie testuj innych znaczników.
- Nie zmieniaj pliku CSS — wszystkie efekty uzyskujesz, dodając i usuwając gotowe klasy.

Przebieg zadania

1 Rozpoznanie strony

Cały kod piszesz w `skrypt.js` (dołączonym z `defer`). Obejrzyj w `styl.css` klasy, które będziesz przełączać: `ukryty`, `ciemny` (na body), `otwarte` (na `.pytanie`), `aktywna` (na miniaturze), `mało` (na liczniku).

```
const btn = document.querySelector("#btn-godziny");
btn.addEventListener("click", () => {
  // co ma się stać po kliknięciu
});
```

1. Uruchom stronę przez Live Server, otwórz F12 → Elements. Wpisz w komentarzu `skrypt.js` swoje imię i nazwisko.
2. W konsoli wypróbuj: `document.querySelector("h1").textContent = "Test"` oraz `document.body.classList.add("ciemny")`. Odśwież stronę — zmiany z konsoli znikają, a te zapisane w skrypcie zostaną.

2 Tekst, klasy i style

1. D1. Przycisk „Pokaż godziny otwarcia” pokazuje i ukrywa akapit `#godziny` (klasa `ukryty`). Tekst przycisku zmienia się na „Ukryj godziny otwarcia” i z powrotem.
2. D2. Przyciski A+ i A- zmieniają `document.body.style.fontSize` o 2 px, w zakresie od 12 do 24 px (zaczynij od 16). Sprawdź w Elements, że pojawia się atrybut `style` na body.
3. D3. Przycisk „Tryb ciemny” przełącza klasę `ciemny` na body, a jego napis zmienia się na „Tryb jasny”. Dodatkowo zapamiętaj wybór w `localStorage` (klucz `tryb`) i odtwórz go po odświeżeniu strony; sprawdź wpis w F12 → Application → Local Storage.

3 FAQ i galeria — wiele elementów naraz

`querySelectorAll` zwraca listę elementów — do każdego dodaj obsługę zdarzenia w pętli `forEach`.

1. D4. Kliknięcie pytania w FAQ przełącza klasę `otwarte` na jego rodzicu (`.pytanie`, właściwość `parentElement`) i ustawia atrybut `aria-expanded` na `true` lub `false`. Pozostałe pytania mają się nie zmieniać.
2. D5. Kliknięcie miniatury w galerii podmienia `src` dużego zdjęcia na wartość z atrybutu `data-duze` (`element.dataset.duze`), przepisuje `alt` i ustawia podpis `#podpis`. Klasa `aktywna` ma być tylko na klikniętej miniaturze.

4 Księga gości — licznik, podgląd i nowe elementy

1. D6. Licznik pod polem wpisu pokazuje na żywo (zdarzenie `input`) „Pozostało: N znaków” (limit 200). Gdy zostanie mniej niż 20 znaków, licznik dostaje klasę `mało`.
2. D7. Przycisk „Podgląd” wyświetla w `#podglad` tekst „Imię: treść wpisu” przez `textContent`. Gdy imię jest puste, użyj słowa „Gość”.
3. D8. Przycisk „Dodaj wpis” tworzy nowy element `li` (imię pogrubione w elemencie `strong`, potem treść) i wstawia go na początek listy `#lista-wpisow` (`prepend`). Zaktualizuj liczbę wpisów w `#liczba-wpisow`, wyczyść pole i licznik. Pusty wpis ma nie zostać dodany — wyświetl komunikat w `#podglad`.
4. D9. Eksperyment: wpisz w polu tekst `Piękne miejsce, <b>połączam</b>!` i kliknij Podgląd. Następnie tymczasowo zamień w D7 `textContent` na `innerHTML`, powtórz próbę i porównaj wynik oraz kod w Elements. Przywróć `textContent`.
5. Sprawdź konsolę (brak błędów) i zrób zrzuty: strona w trybie ciemnym z otwartym jednym pytaniem i inną miniaturą, księga gości z 2 wpisami, wyniki D9 dla obu metod.

W sprawozdaniu umieść

- ☐ tabela D1–D9 z fragmentami kodu i opisem działania
- ☐ zrzuty: tryb ciemny z otwartym pytaniem i zmienionym zdjęciem, księga gości z 2 wpisami, D9 dla `textContent` i `innerHTML`, Local Storage z kluczem `tryb`
- ☐ wniosek z eksperymentu D9
- ☐ folder `zad25` z plikami strony
- ☐ odpowiedzi na pytania kontrolne

Pytania kontrolne

1. Czym różni się `querySelector` od `querySelectorAll`? Co zwraca każda z nich, gdy nie znajdzie elementu?
2. Dlaczego lepiej przełączać klasę CSS (`classList.toggle`) niż ustawiać wiele właściwości `element.style` w skrypcie?
3. Co pokazał eksperyment D9? Dlaczego tekst od użytkownika wstawiamy przez `textContent`?
4. Do czego służą atrybuty `data-*` i jak odczytać je w JavaScript?

Kryteria oceny

Kryterium	Punkty
D1–D2 — pokazywanie godzin ze zmianą napisu, rozmiar czcionki z ograniczeniem 12–24 px	3
D3 — tryb ciemny przełączany klasą, zapamiętany po odświeżeniu	3
D4–D5 — FAQ niezależne dla każdego pytania z <code>aria-expanded</code> , galeria podmienia <code>src</code> , <code>alt</code> , podpis i klasę aktywna	5
D6–D8 — licznik z klasą mało, podgląd przez <code>textContent</code> , dodawanie wpisów z licznikiem i blokadą pustego wpisu	5
D9 i zrzuty — poprawny wniosek, kod bez błędów w konsoli	2
Odpowiedzi na pytania kontrolne	2
Razem	20

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