

Formularz z walidacją i localStorage

Czas: 2 × 45 min

praca indywidualna

Cel: Zbudowanie formularza kontaktowego z walidacją na żywo (atrybuty HTML, Constraint Validation API, wyrażenia regularne) oraz zapisywanie szkicu i listy notatek w localStorage tak, aby przetrwały odświeżenie strony.

Po zajęciach potrafisz:

- dobiera atrybuty walidacji HTML (`required`, `minlength`, `pattern`, typy pól)
- odczytuje stan pola przez `validity` i wyświetla własne komunikaty, także z `setCustomValidity`
- zapisuje i odczytuje dane w localStorage z użyciem `JSON.stringify` i `JSON.parse`
- tworzy listę zadań zachowywaną po odświeżeniu strony
- wyjaśnia różnicę między localStorage i sessionStorage oraz ograniczenia pamięci przeglądarki

Sprzęt i materiały

- komputer z Windows 11
- folder `r2_03_start` z Pliki_do_zadan (`index.html`, `styl.css`)

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)

Przygotowanie przed zajęciami

- Przejrzyj wykład 2.02 (Constraint Validation API, localStorage).

Bezpieczeństwo

- localStorage może odczytać każdy skrypt działający na stronie i każdy, kto usiądzie przy komputerze — nie zapisuj w nim haseł, numerów PESEL ani danych kart.
- Po zajęciach wyczyść dane strony: F12 → Aplikacja (Application) → Pamięć lokalna → Wyczyść, szczególnie na wspólnym komputerze w pracowni.

Przebieg zadania

1 Atrybuty walidacji w HTML

Pracownia ceramiki chce formularza kontaktowego i podręcznej listy notatek. Skopiuj `r2_03_start` do folderu `nazwisko_imie_z03` i uruchom Live Server.

1. Dodaj atrybuty: imię i nazwisko — wymagane, 3–60 znaków; e-mail — `type="email"`, wymagany; telefon — `type="tel"`, nieobowiązkowy, wzorzec 9 cyfr z opcjonalnymi spacjami po 3 cyfrach; kod pocztowy — wymagany, wzorzec `00-000`; temat i zgoda — wymagane; wiadomość — 20–500 znaków.
2. Usuń na chwilę `novalidate` z formularza, wyślij go pusty i zrób zrzut dymka przeglądarki. Przywróć `novalidate` — dalej komunikaty pokazuje Twój skrypt.

2 Walidacja na żywo (formularz.js)

Każde pole ma pod sobą akapit `#k-<id pola>`. Klasy `ok` i `zle` kolorują ramkę pola.

```
function tekstBledu(pole) {
  const v = pole.validity;
  if (v.valueMissing) return "To pole jest wymagane.";
  // dopisz: typeMismatch, patternMismatch (osobno kod i telefon),
  // tooShort z liczbą znaków, customError
  return "";
}
```

1. Dokończ funkcję `tekstBledu(pole)`. Dla zgody wyświetl „Zaznacz zgodę.”, dla zbyt krótkiej wiadomości np. „Za krótko: minimum 20 znaków (jest 6).”.
2. Napisz `sprawdzPole(pole)`, która wpisuje komunikat i przełącza klasy `ok` / `zle` (`classList.toggle`). Wywołuj ją przy zdarzeniach `input` i `blur` każdego pola.
3. Dodaj regułę własną przez `setCustomValidity`: imię i nazwisko musi składać się z co najmniej dwóch słów (podziel tekst wyrażeniem `/\s+/`). Pusty tekst w `setCustomValidity("")` usuwa błąd.
4. Pod wiadomością wyświetlaj licznik znaków „N / 500”. Przycisk „Wyślij” ma być nieaktywny, dopóki `form.checkValidity()` zwraca `false`.

3 Szkic w localStorage

Jeśli użytkownik zamknie kartę w połowie pisania, po powrocie formularz ma się wypełnić.

1. Przy każdym `input` zapisuj obiekt z wartościami wszystkich pól i czasem zapisu pod kluczem `glina-szkic` (`JSON.stringify`). Dla pola wyboru zapisuj `checked`.
2. Po załadowaniu strony odczytaj szkic (`JSON.parse`), wypełnij pola, sprawdź je i pokaż w `#info-szkic` napis „Przywrócono szkic z godz. GG:MM.”.
3. Po poprawnym wysłaniu usuń szkic (`removeItem`), wyczyść formularz i pokaż podziękowanie z liczbą wysłanych wiadomości w tej karcie, przechowywaną w **sessionStorage**.

4 Lista notatek (notatki.js)

Notatki to tablica obiektów `{ id, tekst, zrobione }` zapisywana w całości pod kluczem `glina-notatki`.

1. Przy starcie wczytaj tablicę: `JSON.parse(localStorage.getItem("glina-notatki") ?? "[]")`. Napisz funkcję `rysuj()`, która tworzy elementy `li` (pole wyboru, tekst przez `textContent`, przycisk „Usuń”) i pokazuje statystykę „Notatek: N, zrobionych: M”.
2. Obsłuż dodawanie (pusta notatka niedozwolona), oznaczanie jako zrobione (przekreślenie — klasa `zrobione`), usuwanie jednej notatki i przycisk „Usuń zrobione”. Po każdej zmianie zapisz tablicę i wywołaj `rysuj()`.
3. Test: dodaj 3 notatki, jedną oznacz, odśwież stronę (F5), zamknij i otwórz kartę. Obejrzyj klucze w F12 → Aplikacja → Pamięć lokalna i Pamięć sesji. Zrób zrzuty.

W sprawozdaniu umieść

- zrzut dymka walidacji przeglądarki (bez `novalidate`) i zrzut własnych komunikatów
- tabela testów pól z wpisanymi wartościami i komunikatami
- zrzut F12 → Aplikacja z kluczami `glina-szkic` i `glina-notatki` oraz zawartością `sessionStorage`
- kod funkcji `tekstBledu`, `zapiszSzkic` i `rysuj`
- spakowany folder `nazwisko_imie_z03`

Pytania kontrolne

1. Czym różni się localStorage od sessionStorage? Co stało się z licznikiem wysłanych wiadomości po zamknięciu karty?
2. Dlaczego obiekt trzeba zamienić na tekst `JSON.stringify` przed zapisem do localStorage? Co zapisze się bez tego?
3. Jakich danych nie należy przechowywać w localStorage i dlaczego?
4. Dlaczego walidacja w przeglądarce nie zwalnia serwera z ponownego sprawdzenia danych?

Kryteria oceny

Kryterium	Punkty
Atrybuty walidacji HTML zgodne z wymaganiami (7 pól, 2 wzorce)	3
Komunikaty z validity dla 5 rodzajów błędów, klasy ok/zle, reguła setCustomValidity	5
Licznik znaków i przycisk aktywny tylko dla poprawnego formularza	2
Szkic: zapis, przywrócenie z godziną, usunięcie po wysłaniu; licznik w sessionStorage	4
Notatki: dodawanie, oznaczanie, usuwanie, „Usuń zrobione”, zachowanie po odświeżeniu	4
Zrzuty z F12 i odpowiedzi na pytania	2
Razem	20

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