SPRAWOZDANIE Z LABORATORIUM

Technologie Webowe (JavaScript)

INFORMACJE PODSTAWOWE

Imię i nazwisko studenta: Patryk Broński
Numer albumu: 27598
Grupa: inf26+ gr.3 spec. programowanie
Data wykonania laboratorium: 9.12.2025r.
Numer laboratorium: 3
Temat laboratorium: Javascript

1. CEL LABORATORIUM

Celem laboratorium było utrwalenie podstaw JavaScript oraz przejście przez kluczowe obszary języka: zmienne i typy danych, operatory, instrukcje sterujące (if/switch/pętle), funkcje i zakresy (w tym domknięcia), manipulację DOM i obsługę zdarzeń, walidację formularzy oraz asynchroniczność (Promises, async/await, Fetch API). Zadania rozszerzały wcześniej wykonane strony z HTML i CSS.

2. WYKONANE ZADANIA

Zadanie 1: Pierwsze kroki z JavaScript

Status: ☐ **Wykonane w pełni** ☐ Wykonane częściowo ☐ Nie wykonane

Opis realizacji:

W script1.js wykonałem część wymaganą w zadaniu:

- sprawdziłem podłączenie skryptu przez console.log('Skrypt załadowany!'),
- utworzyłem zmienne typów: string, number, boolean, array, object, null, undefined i wypisałem ich typy przez typeof,
- wykonałem operacje arytmetyczne: + - * / % ** na liczbach 10 i 5,
- porównałem == i === na przykładzie '5' i 5 (wypis w konsoli).

Napotkane problemy:

Brak.

Rozwiązanie:

Nie dotyczy.

Zadanie 2: Instrukcje warunkowe i pętle

Status: ☐ **Wykonane w pełni** ☐ Wykonane częściowo ☐ Nie wykonane

Opis realizacji:

W script2.js zrealizowałem wszystkie punkty z PDF:

- funkcja parzysta/nieparzysta (modulo),
- kalkulator ocen słownych 0–100 (if/else-if),
- switch dla dni tygodnia 1–7 (z default),
- operator trójargumentowy do pełnoletności,
- pętle: for 1–10, while 10–0, for...of po tablicy, for...in po obiekcie,
- użycie break i continue,
- tabliczka mnożenia 1–10 w dwóch pętlach.

Napotkane problemy:

Brak.

Rozwiązanie:

Nie dotyczy.

Zadanie 3: Funkcje i zakres zmiennych

Status: ☐ **Wykonane w pełni** ☐ Wykonane częściowo ☐ Nie wykonane

Opis realizacji:

W script3.js wykonałem:

- 3 sposoby definiowania tej samej funkcji (function declaration, function expression, arrow function) na przykładzie pola prostokąta,
- funkcję z parametrem domyślnym przywitanie(imie = 'Nieznajomy'),
- demonstrację różnicy var vs let w pętli z setTimeout,
- przykład closure (licznik zwracający funkcję z zapamiętanym stanem).

Napotkane problemy:

Brak.

Rozwiązanie:

Nie dotyczy.

Zadanie 4: Manipulacja DOM - Interaktywna galeria

Status: ☐ **Wykonane w pełni** ☐ Wykonane częściowo ☐ Nie wykonane

Opis realizacji:

W script4.js dodałem interaktywność do galerii:

- wybieranie elementów: getElementById, querySelectorAll, querySelector,
- nawigacja po DOM przez closest('img') (event delegation),
- dynamiczna zmiana tytułu galerii przez textContent,
- dodawanie/usuwanie klas na hover (classList.add/remove) + ustawianie stylu inline (cursor),
- praca na atrybutach: setAttribute, getAttribute, data-* (dataset.index, dataset.desc),
- zbudowanie lightboxa w JS: overlay, obraz, opis, przyciski Prev/Next, zamknięcie X lub kliknięciem na tło, oraz obsługa klawiatury (Escape/strzałki).

Napotkane problemy:

Zamykanie overlay tylko po kliknięciu tła, a nie elementów w środku.

Rozwiązanie:

Warunek if (e.target === overlay).

Zadanie 5: Formularze i walidacja

Status: ☐ **Wykonane w pełni** ☐ Wykonane częściowo ☐ Nie wykonane

Opis realizacji:

W script5.js wykonałem wszystkie części zadania:

- pobieranie danych z pól formularza oraz obsługa submit z e.preventDefault(),
- FormData API i wypis danych w konsoli jako obiekt (formDataToObject),

- walidacje zgodnie z PDF: imię (min 2 + polskie znaki), email (regex), telefon (opcjonalny, 9 cyfr), wiadomość (min 10), checkbox zgody,
- walidacja realtime (input/blur/change),
- UX: komunikaty błędów pod polem, czerwone/zielone oznaczenia, blokada przycisku i tekst „Wysyłanie…”, symulacja wysyłki 1500ms, komunikat sukcesu i reset formularza.

Zadanie 6: Asynchroniczność i API

Status: ☐ **Wykonane w pełni** ☐ Wykonane częściowo ☐ Nie wykonane

Opis realizacji:

W script6.js zrealizowałem wszystkie punkty:

- własna Promise z setTimeout (resolve/reject),
- .then(), .catch(), .finally(),
- Promise.all,
- Promise.race jako timeout (withTimeout),
- wersja async/await + try/catch, sekwencyjnie i równolegle (Promise.all),
- Fetch API na JSONPlaceholder: pobranie /users, dynamiczna tabela, loader, obsługa błędów, przycisk Refresh, kliknięcie użytkownika pobiera /users/{id}/posts i wyświetla posty.

Napotkane problemy:

fetch nie rzuca błędu automatycznie przy 404/500.

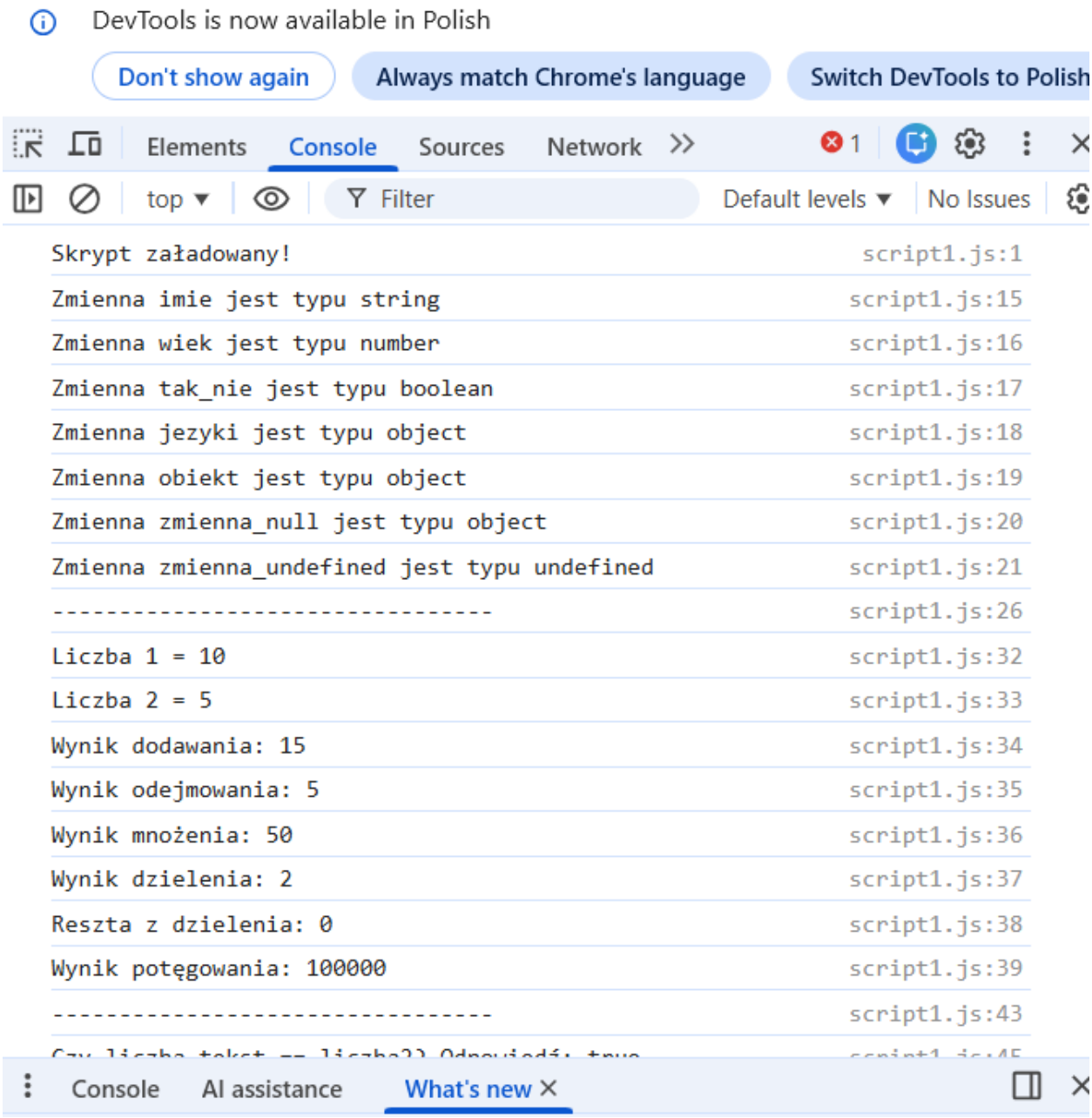
Rozwiązanie:

Sprawdzenie if (!res.ok) throw new Error('HTTP ' + res.status).

3. FRAGMENTY KODU oraz 4. ZRZUTY EKRANU

```
console.log('Skrypt załadowany!');
const imie = 'Patryk';
const wiek = 26;
const tak_nie = true;
const jezyki = ['js', 'java', 'dart'];
const obiekt = {
    imie: 'Patryk',
```

```
    wiek: 26,
    miasto: 'Krosno'
};
const zmienna_null = null;
let zmienna_undefined;

console.log("Zmienna imie jest typu " + typeof imie);
console.log("Zmienna wiek jest typu " + typeof wiek);
console.log("Zmienna tak_nie jest typu " + typeof tak_nie);
console.log("Zmienna jezyki jest typu " + typeof jezyki);
console.log("Zmienna obiekt jest typu " + typeof obiekt);
console.log("Zmienna zmienna_null jest typu " + typeof zmienna_null);
console.log("Zmienna zmienna_undefined jest typu " + typeof zmienna_undefined);



console.log("---------------------------------")

let liczba_tekst = '5';
let liczba1 = 10;
let liczba2 = 5;

console.log("Liczba 1 = 10");
console.log("Liczba 2 = 5");
console.log("Wynik dodawania: " + parseInt(liczba1 + liczba2));
console.log("Wynik odejmowania: " + parseInt(liczba1 - liczba2));
console.log("Wynik mnożenia: " + parseInt(liczba1 * liczba2));
console.log("Wynik dzielenia: " + parseInt(liczba1 / liczba2));
console.log("Reszta z dzielenia: " + parseInt(liczba1 % liczba2));
console.log("Wynik potęgowania: " + parseInt(liczba1 ** liczba2));


console.log("---------------------------------")

console.log("Czy liczba_tekst == liczba2? Odpowiedź: " + (liczba_tekst == liczba2));
console.log("Czy liczba_tekst === liczba2? Odpowiedź: " + (liczba_tekst === liczba2));
```




```
// Część A

// 4. Funkcja parzysta/nieparzysta
function parzystaCzyNie(liczba) {
    if (liczba % 2 === 0) {
        return 'parzysta'
    } else {
        return 'nieparzysta'
    }
```

```
}
console.log('4) 7:', parzystaCzyNie(7))
console.log('4) 12:', parzystaCzyNie(12))

// 5. Kalkulator ocen (0-100 -> ocena słowna)
function ocenaSlowna(punkty) {
    if (punkty < 0 || punkty > 100) return 'BŁĄD: zakres 0-100'

    if (punkty >= 90) return 'bardzo dobry'
    else if (punkty >= 75) return 'dobry'
    else if (punkty >= 60) return 'dostateczny'
    else if (punkty >= 50) return 'dopuszczający'
    else return 'niedostateczny'
}
console.log('5) 82:', ocenaSlowna(82))
console.log('5) 45:', ocenaSlowna(45))

// 6. switch - dzień tygodnia po numerze (1-7)
function dzienTygodnia(nr) {
    switch (nr) {
        case 1:
            return 'Poniedziałek'
        case 2:
            return 'Wtorek'
        case 3:
            return 'Środa'
        case 4:
            return 'Czwartek'
        case 5:
            return 'Piątek'
        case 6:
            return 'Sobota'
        case 7:
            return 'Niedziela'
        default:
            return 'BŁĄD: podaj 1-7'
    }
}
console.log('6) 6:', dzienTygodnia(6))

// 7. ternary - pełnoletność
function pelnoletni(wiek) {
    return wiek >= 18 ? 'pełnoletni' : 'niepełnoletni'
}
```

```
console.log('7) 17:', pelnoletni(17))
console.log('7) 22:', pelnoletni(22))

// Część B

// 8. for 1..10
console.log('8) for 1..10')
for (let i = 1; i <= 10; i++) {
    console.log(i)
}

// 9. while 10..0
console.log('9) while 10..0')
let x = 10
while (x >= 0) {
    console.log(x)
    x--
}

// 10. for...of po tablicy
console.log('10) for...of po tablicy')
const tab = ['JS', 'HTML', 'CSS']
for (const el of tab) {
    console.log(el)
}

// 11. for...in po obiekcie
console.log('11) for...in po obiekcie')
const osoba = { imie: 'Patryk', miasto: 'Krosno', rok: 3 }
for (const klucz in osoba) {
    console.log(klucz + ':', osoba[klucz])
}

// 12. break i continue
console.log('12) break/continue')
for (let i = 1; i <= 12; i++) {
    if (i === 3) continue // pomijamy 3
    if (i === 9) break // kończymy przy 9
    console.log(i)
}

// Część C
console.log('C) TABLICZKA MNOŻENIA')
for (let i = 1; i <= 10; i++) {
```

```
    let linia = ''
    for (let j = 1; j <= 10; j++) {
        linia += i * j + '\t'
    }
    console.log(linia)
}
```

```
imie: Patryk                                                    script2.js:84
miasto: Krosno                                                  script2.js:84
rok: 3                                                          script2.js:84
12) break/continue                                              script2.js:88
1                                                               script2.js:92
2                                                               script2.js:92
4                                                               script2.js:92
5                                                               script2.js:92
6                                                               script2.js:92
7                                                               script2.js:92
8                                                               script2.js:92
C) TABLICZKA MNOŻENIA                                           script2.js:96
1   2   3   4   5   6   7   8   9   10                          script2.js:102
2   4   6   8   10  12  14  16  18  20                          script2.js:102
3   6   9   12  15  18  21  24  27  30                          script2.js:102
4   8   12  16  20  24  28  32  36  40                          script2.js:102
5   10  15  20  25  30  35  40  45  50                          script2.js:102
6   12  18  24  30  36  42  48  54  60                          script2.js:102
7   14  21  28  35  42  49  56  63  70                          script2.js:102
8   16  24  32  40  48  56  64  72  80                          script2.js:102
9   18  27  36  45  54  63  72  81  90                          script2.js:102
10  20  30  40  50  60  70  80  90  100                         script2.js:102
Live reload enabled.                                            index2.html:118
```

```
// 1) funkcja declaration
function poleProstokata(a, b) {
    return a * b
}
console.log('poleProstokata:', poleProstokata(3, 4))
```

```
// 2) function expression
const poleProstokata2 = function (a, b) {
    return a * b
}
console.log('poleProstokata2:', poleProstokata2(5, 2))

// 3) arrow function
const poleProstokata3 = (a, b) => a * b
console.log('poleProstokata3:', poleProstokata3(10, 2))

// 4) parametry domyślne
function przywitanie(imie = 'Nieznajomy') {
    console.log('Cześć ' + imie)
}
przywitanie()
przywitanie('Patryk')

// 5) var vs let w pętli + setTimeout
for (var i = 1; i <= 3; i++) {
    setTimeout(function () {
        console.log('var i:', i)
    }, 0)
}

for (let j = 1; j <= 3; j++) {
    setTimeout(function () {
        console.log('let j:', j)
    }, 0)
}

// 6) closure: licznik
function zrobLicznik() {
    let licznik = 0

    return function () {
        licznik++
        return licznik
    }
}

const liczA = zrobLicznik()
console.log('liczA:', liczA()) // 1
console.log('liczA:', liczA()) // 2
console.log('liczA:', liczA()) // 3
```

```
const liczB = zrobLicznik()
console.log('liczB:', liczB()) // 1 (osobny licznik)
```

```
poleProstokata: 12                                    script3.js:5
poleProstokata2: 10                                   script3.js:11
poleProstokata3: 20                                   script3.js:15
Cześć Nieznajomy                                      script3.js:19
Cześć Patryk                                          script3.js:19
liczA: 1                                              script3.js:48
liczA: 2                                              script3.js:49
liczA: 3                                              script3.js:50
liczB: 1                                              script3.js:53
Live reload enabled.                                  index3.html:150
3 var i: 4                                            script3.js:27
let j: 1                                              script3.js:33
let j: 2                                              script3.js:33
let j: 3                                              script3.js:33
```

```
// Część A

//nadajemy ID pierwszej sekcji z obrazkami.
const pierwszaSekcja = document.querySelector('section')
if (pierwszaSekcja && !pierwszaSekcja.id) {
    pierwszaSekcja.id = 'gallery'
}

// 12. getElementById
const galeria = document.getElementById('gallery')
if (!galeria) {
    console.log('Brak kontenera #gallery')
} else {
    // 13. querySelectorAll
    const obrazki = galeria.querySelectorAll('img')

    // 14. querySelector z selektorami CSS
    const tytul = document.querySelector('h1')
```

```
// Część B

// 18. zmiana tytułu galerii
if (tytul) {
    tytul.textContent = 'Moja galeria multimedialna (JS + Lightbox)'
}

// 22. data-* atrybuty (indeks + opis)
obrazki.forEach((img, index) => {
    img.dataset.index = index
    img.dataset.desc = img.alt || 'Brak opisu'
    img.style.cursor = 'pointer' // 20. inline style
})

// 19. klasy CSS na hover
obrazki.forEach(img => {
    img.addEventListener('mouseenter', () => img.classList.add('hovered'))
    img.addEventListener('mouseleave', () => img.classList.remove('hovered'))
})

const style = document.createElement('style')
style.innerHTML = `
.hovered { outline: 3px solid #3b82f6; outline-offset: 2px; }
.lbOverlay {
  position: fixed; left: 0; top: 0; width: 100%; height: 100%;
  background: rgba(0,0,0,0.8);
  display: none;
  align-items: center;
  justify-content: center;
  z-index: 9999;
}
.lbBox {
  background: #111;
  padding: 10px;
  border-radius: 10px;
  max-width: 90%;
  max-height: 90%;
  color: white;
  font-family: Arial, sans-serif;
}
.lbBox img { max-width: 80vw; max-height: 70vh; display: block; margin: 0 auto; }
.lbControls { display: flex; gap: 8px; justify-content: center; margin-top: 8px; }
.lbControls button { padding: 6px 10px; cursor: pointer; }
.lbDesc { margin-top: 8px; font-size: 14px; text-align: center; }
```

```
.lbTop { display:flex; justify-content: flex-end; }
`
document.head.appendChild(style)

// 21. modyfikacja atrybutów
obrazki.forEach(img => {
    img.setAttribute('tabindex', '0')
})

//Część C - Lightbox

// overlay
const overlay = document.createElement('div')
overlay.className = 'lbOverlay'

const box = document.createElement('div')
box.className = 'lbBox'

const top = document.createElement('div')
top.className = 'lbTop'

// 11. przycisk X
const btnClose = document.createElement('button')
btnClose.textContent = 'X'

const bigImg = document.createElement('img')
const desc = document.createElement('div')
desc.className = 'lbDesc'

const controls = document.createElement('div')
controls.className = 'lbControls'

// 10. Poprzedni/Następny
const btnPrev = document.createElement('button')
btnPrev.textContent = 'Poprzedni'

const btnNext = document.createElement('button')
btnNext.textContent = 'Następny'

top.appendChild(btnClose)
controls.appendChild(btnPrev)
controls.appendChild(btnNext)

box.appendChild(top)
```

```
box.appendChild(bigImg)
box.appendChild(controls)
box.appendChild(desc)

overlay.appendChild(box)
document.body.appendChild(overlay)

let aktualnyIndex = 0

function pokazLightbox(index) {
    aktualnyIndex = index

    const img = obrazki[aktualnyIndex]
    bigImg.src = img.getAttribute('src') // 21 getAttribute
    desc.textContent = img.dataset.desc // 13. tytuł/opis pod obrazem

    overlay.style.display = 'flex'
}

function zamknij() {
    overlay.style.display = 'none'
}

function nastepny() {
    aktualnyIndex++
    if (aktualnyIndex >= obrazki.length) aktualnyIndex = 0
    pokazLightbox(aktualnyIndex)
}

function poprzedni() {
    aktualnyIndex--
    if (aktualnyIndex < 0) aktualnyIndex = obrazki.length - 1
    pokazLightbox(aktualnyIndex)
}

// 9. Klik w obrazek otwiera overlay (event delegation)
galeria.addEventListener('click', e => {
    const kliknietyImg = e.target.closest('img') // 15. closest
    if (!kliknietyImg) return

    const idx = Number(kliknietyImg.dataset.index)
    pokazLightbox(idx)
})
```

```
    // przyciski
    btnClose.addEventListener('click', () => zamknij())
    btnNext.addEventListener('click', () => nastepny())
    btnPrev.addEventListener('click', () => poprzedni())

    // 11. klik poza obrazem (czyli na overlay) zamyka
    overlay.addEventListener('click', e => {
        if (e.target === overlay) {
            zamknij()
        }
    })

    // 12. klawiatura: strzałki + Escape
    document.addEventListener('keydown', e => {
        if (overlay.style.display !== 'flex') return

        if (e.key === 'Escape') zamknij()
        if (e.key === 'ArrowRight') nastepny()
        if (e.key === 'ArrowLeft') poprzedni()
    })
}
```

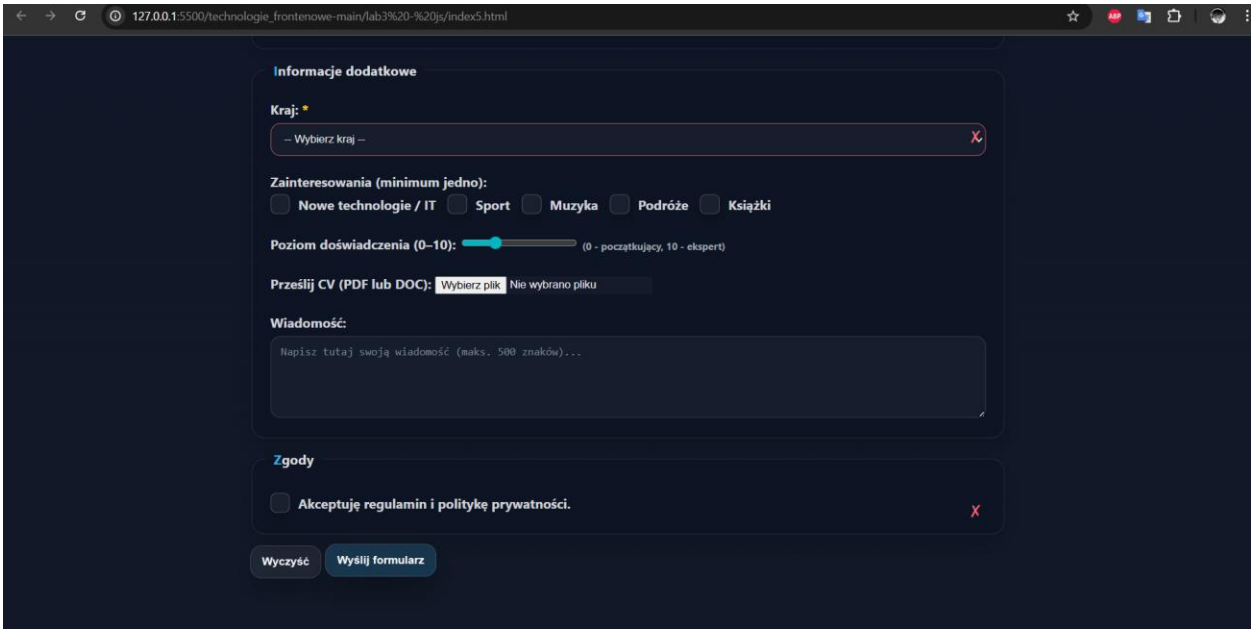
```
// Formularze i walidacja
const form = document.querySelector('form')
const submitBtn = form.querySelector('button[type="submit"]')

// Pola do walidacji
const fullname = document.getElementById('fullname')
const email = document.getElementById('email')
const phone = document.getElementById('phone')
const message = document.getElementById('message')
const terms = document.getElementById('terms')

// Style
const style = document.createElement('style')
style.innerHTML = `
  .errorText { color: red; font-size: 12px; margin-top: 4px; }
  .okField { outline: 2px solid green; }
  .badField { outline: 2px solid red; }
  .successBox { margin-top: 10px; padding: 10px; border: 1px solid green; color: green;
display:none; }
`
document.head.appendChild(style)

const successBox = document.createElement('div')
successBox.className = 'successBox'
successBox.textContent = 'Sukces! Formularz wysłany.'
form.appendChild(successBox)

// helper: pokaż błąd pod polem
function setError(field, text) {
    clearError(field)

    field.classList.remove('okField')
    field.classList.add('badField')

    const div = document.createElement('div')
    div.className = 'errorText'
    div.textContent = text

    // wstawiane po polu
    field.insertAdjacentElement('afterend', div)
}

function setOk(field) {
    clearError(field)
```

```
    field.classList.remove('badField')
    field.classList.add('okField')
}

function clearError(field) {
    const next = field.nextElementSibling
    if (next && next.classList.contains('errorText')) {
        next.remove()
    }
}

// 23. Imię: min 2 znaki, tylko litery (z polskimi znakami)
function validateName() {
    const val = fullname.value.trim()
    const re = /^[A-Za-zĄĆĘŁŃÓŚŹŻąćęłńóśźż\s-]{2,}$/

    if (val.length < 2) {
        setError(fullname, 'Imię i nazwisko: minimum 2 znaki.')
        return false
    }
    if (!re.test(val)) {
        setError(fullname, 'Imię i nazwisko: tylko litery (także polskie), spacja lub myślnik.')
        return false
    }
    setOk(fullname)
    return true
}

// 24. Email regex
function validateEmail() {
    const val = email.value.trim()
    const re = /^[^\s@]+@[^\s@]+\.[^\s@]+$/

    if (!re.test(val)) {
        setError(email, 'Email ma niepoprawny format.')
        return false
    }
    setOk(email)
    return true
}

// 25. Telefon: opcjonalny, ale jak jest to musi mieć 9 cyfr
function validatePhone() {
    const val = phone.value.trim()
```

```
    if (val === '') {
        // puste jest OK (opcjonalne). czyścimy oznaczenia
        phone.classList.remove('badField', 'okField')
        clearError(phone)
        return true
    }

    // wywalamy spacje, myślniki, +48 itd.
    const digits = val.replace(/\D/g, '')
    const last9 = digits.length > 9 ? digits.slice(-9) : digits

    if (!/^\d{9}$/.test(last9)) {
        setError(phone, 'Telefon: jeśli podany, musi mieć 9 cyfr.')
        return false
    }
    setOk(phone)
    return true
}

// 26. Wiadomość min 10 znaków
function validateMessage() {
    const val = message.value.trim()
    if (val.length < 10) {
        setError(message, 'Wiadomość: minimum 10 znaków.')
        return false
    }
    setOk(message)
    return true
}

// 27. Checkbox zgody
function validateTerms() {
    if (!terms.checked) {
        setError(terms, 'Musisz zaakceptować regulamin i politykę prywatności.')
        return false
    }
    // checkbox bez zielonej ramki, czyścimy błąd
    clearError(terms)
    terms.classList.remove('badField')
    return true
}

// 28. realtime
fullname.addEventListener('input', validateName)
```

```
fullname.addEventListener('blur', validateName)

email.addEventListener('input', validateEmail)
email.addEventListener('blur', validateEmail)

phone.addEventListener('input', validatePhone)
phone.addEventListener('blur', validatePhone)

message.addEventListener('input', validateMessage)
message.addEventListener('blur', validateMessage)

terms.addEventListener('change', validateTerms)

// 16-18: Pobieranie danych
function formDataToObject(fd) {
    const obj = {}
    for (const [k, v] of fd.entries()) {
        if (obj[k] === undefined) obj[k] = v
        else if (Array.isArray(obj[k])) obj[k].push(v)
        else obj[k] = [obj[k], v]
    }
    return obj
}

form.addEventListener('submit', e => {
    e.preventDefault()
    successBox.style.display = 'none'

    // Walidacja
    const ok = validateName() && validateEmail() && validatePhone() && validateMessage() &&
validateTerms()

    // 17. FormData API
    const fd = new FormData(form)

    // 18. pokaż w konsoli jako obiekt
    console.log('Dane formularza (wszystkie pola):', formDataToObject(fd))

    if (!ok) return

    // 17. UX: blokuj przycisk i pokaż "Wysyłanie..."
    const oldText = submitBtn.textContent
    submitBtn.disabled = true
    submitBtn.textContent = 'Wysyłanie...'
```

```
    // 18. Symulacja wysyłki 1500ms + sukces + reset
    setTimeout(() => {
        form.reset()

        // Czyścimy ramki po polach
        ;[fullname, email, phone, message].forEach(f => {
            f.classList.remove('okField', 'badField')
            clearError(f)
        })
        clearError(terms)

        successBox.style.display = 'block'

        submitBtn.disabled = false
        submitBtn.textContent = oldText
    }, 1500)
})
```




```
// 19. własna Promise z setTimeout (symulacja ładowania)
function fakeLoad(label, ms, fail) {
    return new Promise((resolve, reject) => {
        setTimeout(() => {
            if (fail) reject('Błąd: ' + label)
            else resolve('OK: ' + label)
        }, ms)
    })
```

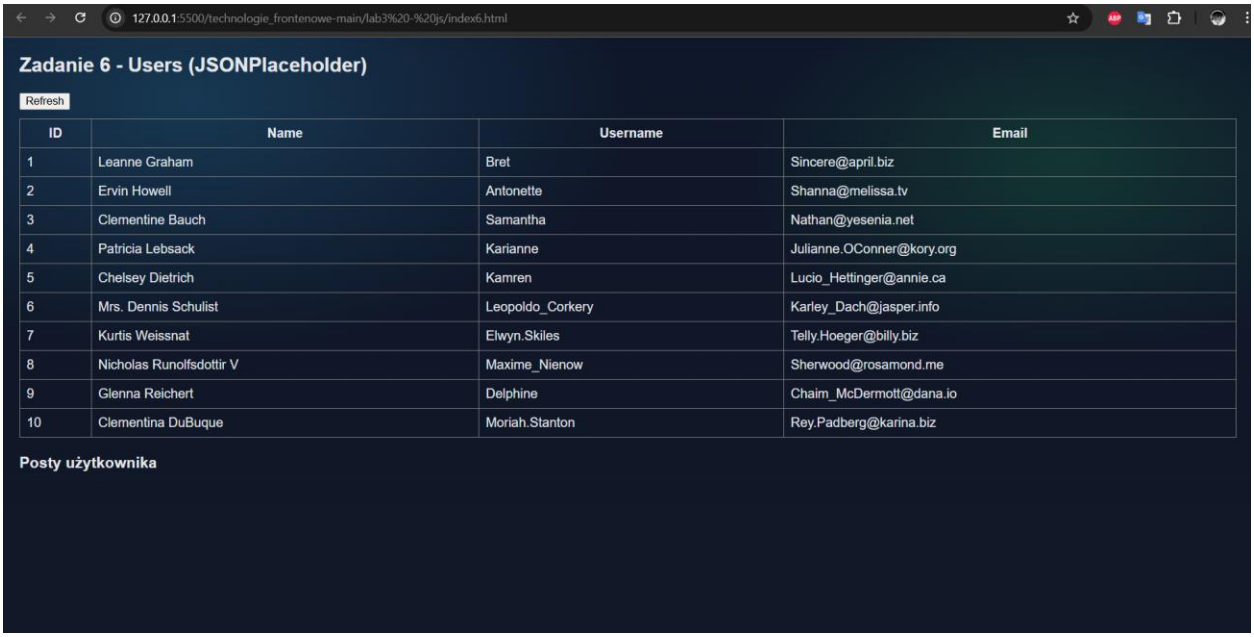
```
}

// 20. then/catch/finally
fakeLoad('A', 400, false)
    .then(res => console.log('20) then:', res))
    .catch(err => console.log('20) catch:', err))
    .finally(() => console.log('20) finally: koniec'))

// 21. Promise.all
Promise.all([fakeLoad('all-1', 300, false), fakeLoad('all-2', 500, false), fakeLoad('all-3', 200,
false)]).then(
    wyniki => {
        console.log('21) Promise.all:', wyniki)
    }
)

// 22. Promise.race jako timeout
function withTimeout(promise, ms) {
    const timeoutPromise = new Promise((_, reject) => {
        setTimeout(() => reject('Timeout po ' + ms + 'ms'), ms)
    })
    return Promise.race([promise, timeoutPromise])
}

withTimeout(fakeLoad('wolne', 1200, false), 600)
    .then(res => console.log('22) race OK:', res))
    .catch(err => console.log('22) race ERR:', err))

// 29-32: przepisanie na async/await + try/catch + sekwencyjnie + równolegle
async function demoAsync() {
    try {
        // 31. sekwencyjnie
        const a = await fakeLoad('seq-1', 200, false)
        const b = await fakeLoad('seq-2', 200, false)
        console.log('31) sekwencyjnie:', a, b)

        // 32. równolegle z Promise.all
        const wyniki = await Promise.all([fakeLoad('par-1', 300, false), fakeLoad('par-2', 350,
false)])
        console.log('32) równolegle:', wyniki)
    } catch (e) {
        // 30. try/catch błędów
        console.log('30) błąd:', e)
    }
```

```
}
demoAsync()

const API = 'https://jsonplaceholder.typicode.com'

const app = document.createElement('div')
app.style.margin = '20px'
app.style.fontFamily = 'Arial, sans-serif'

app.innerHTML = `
  <h2>Zadanie 6 - Users (JSONPlaceholder)</h2>
  <button id="refreshBtn">Refresh</button>
  <span id="loader" style="margin-left:10px; display:none;">Ładowanie...</span>
  <div id="error" style="color:red; margin-top:10px;"></div>

  <div id="tableWrap" style="margin-top:10px;"></div>

  <h3 style="margin-top:20px;">Posty użytkownika</h3>
  <div id="postsInfo"></div>
  <ul id="postsList"></ul>
`
document.body.appendChild(app)

const refreshBtn = document.getElementById('refreshBtn')
const loader = document.getElementById('loader')
const errorBox = document.getElementById('error')
const tableWrap = document.getElementById('tableWrap')
const postsInfo = document.getElementById('postsInfo')
const postsList = document.getElementById('postsList')

function showLoader(flag) {
    loader.style.display = flag ? 'inline' : 'none'
}

function setError(msg) {
    errorBox.textContent = msg || ''
}

// 19. pobierz listę users /users
async function loadUsers() {
    setError('')
    postsInfo.textContent = ''
    postsList.innerHTML = ''
    showLoader(true)
```

```
    try {
      const res = await fetch(API + '/users')

      // wskazówka: fetch nie rzuca błędu przy 404/500 -> response.ok
      if (!res.ok) throw new Error('HTTP ' + res.status)

      const users = await res.json() // response.json() też jest Promise

      // 20. tabela HTML tworzona dynamicznie
      renderTable(users)
    } catch (e) {
      // 22. obsługa błędów + komunikat
      setError('Błąd pobierania użytkowników: ' + e.message)
    } finally {
      showLoader(false)
    }
}

function renderTable(users) {
    tableWrap.innerHTML = ''

    const table = document.createElement('table')
    table.border = '1'
    table.cellPadding = '8'
    table.style.borderCollapse = 'collapse'
    table.style.width = '100%'

    const thead = document.createElement('thead')
    thead.innerHTML = `
    <tr>
      <th>ID</th>
      <th>Name</th>
      <th>Username</th>
      <th>Email</th>
    </tr>
  `
    table.appendChild(thead)

    const tbody = document.createElement('tbody')

    users.forEach(u => {
      const tr = document.createElement('tr')
      tr.style.cursor = 'pointer'
```

```
      tr.innerHTML = `
      <td>${u.id}</td>
      <td>${u.name}</td>
      <td>${u.username}</td>
      <td>${u.email}</td>
    `

      // 24. klik user -> /users/{id}/posts
      tr.addEventListener('click', () => loadPosts(u.id, u.name))

      tbody.appendChild(tr)
    })

    table.appendChild(tbody)
    tableWrap.appendChild(table)
}

// 24. pobierz posty użytkownika
async function loadPosts(userId, userName) {
    setError('')
    postsInfo.textContent = ''
    postsList.innerHTML = ''
    showLoader(true)

    try {
        const res = await fetch(API + '/users/' + userId + '/posts')
        if (!res.ok) throw new Error('HTTP ' + res.status)

        const posts = await res.json()

        postsInfo.textContent = 'Użytkownik: ' + userName + ' (ID ' + userId + '), postów: ' +
posts.length

        posts.forEach(p => {
            const li = document.createElement('li')
            li.textContent = p.title
            postsList.appendChild(li)
        })
    } catch (e) {
        setError('Błąd pobierania postów: ' + e.message)
    } finally {
        showLoader(false)
    }
```

```
}

// 23. przycisk Refresh
refreshBtn.addEventListener('click', loadUsers)

// start
loadUsers()
```



5. WNIOSKI I REFLEKSJE

Podczas laboratorium nauczyłem się praktycznie korzystać z języka JavaScript i zrozumiałem, jak działa warstwa logiki aplikacji webowej. Na początku przećwiczyłem podstawy, takie jak typy danych, operatory oraz różnice między == i ===, co pozwoliło mi lepiej zrozumieć sposób działania porównań w JS.

Kolejne zadania pomogły mi utrwalić instrukcje warunkowe i pętle. Dzięki nim potrafię tworzyć prostą logikę programu, np. kalkulator ocen czy tabliczkę mnożenia. W części dotyczącej funkcji zrozumiałem różne sposoby ich definiowania oraz różnicę między var i let. Najciekawszym elementem było domknięcie (closure), ponieważ pokazało mi, że funkcja może „pamiętać” swój stan między wywołaniami.

Najbardziej praktyczna była praca z DOM. Tworzenie galerii z lightboxem uświadomiło mi, jak dynamicznie dodawać elementy, zmieniać treść strony oraz obsługiwać zdarzenia użytkownika (kliknięcia, klawiatura). Dzięki temu zobaczyłem, że JavaScript realnie wpływa na interaktywność strony, a nie tylko na obliczenia w konsoli.

Przy formularzach nauczyłem się walidacji danych po stronie klienta, używania wyrażeń regularnych oraz API FormData. Pozwoliło to stworzyć formularz, który sprawdza dane w czasie rzeczywistym i daje użytkownikowi czytelne komunikaty błędów.

W ostatnim zadaniu poznałem asynchroniczność, Promise, async/await oraz fetch. Zrozumiałem, jak pobierać dane z zewnętrznego API i dynamicznie budować widok na podstawie odpowiedzi serwera. Było to najbardziej zbliżone do realnych zastosowań w nowoczesnych aplikacjach webowych.

Najtrudniejsze było dla mnie zrozumienie programowania asynchronicznego oraz poprawne zarządzanie zdarzeniami w DOM. W przyszłości chciałbym jeszcze przećwiczyć bardziej rozbudowane projekty z użyciem API oraz organizację większego kodu JavaScript w moduły.

6. SAMOOCENA

Ocena własnego zaangażowania: ☐ Bardzo wysokie ☐ **Wysokie** ☐ Średnie ☐ Niskie

Procent wykonanych zadań: 100 %

Dodatkowe uwagi: brak

Data wypełnienia sprawozdania: 10.01.2026r.
Podpis studenta: Patryk Broński