SCENARIUSZ LEKCJI

z informatyki dla klasy 3 (technik informatyk)

Autor scenariusza - Sylwester Płachta

TEMAT: „Dlaczego to nie działa?” – analiza błędów logicznych i weryfikacja danych w JavaScript jako trening myślenia krytycznego.

CELE KSZTAŁCENIA – wymagania ogólne (podstawa programowa)

- Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych: projektowanie, tworzenie i testowanie programów komputerowych.
- Posługiwanie się komputerem, urządzeniami cyfrowymi i sieciami komputerowymi: przestrzeganie zasad bezpieczeństwa i higieny pracy oraz praw autorskich.
- Rozwijanie umiejętności myślenia analitycznego i algorytmicznego.

WYMAGANIA SZCZEGÓŁOWE

1. Tworzenie i uruchamianie skryptów po stronie klienta (JavaScript).
2. Stosowanie instrukcji warunkowych, pętli oraz funkcji do sterowania przepływem programu.
3. Lokalizowanie i poprawianie błędów składniowych i logicznych w kodzie (debugowanie).

Uczeń potrafi:

- Rozróżnić błąd składniowy (syntax error) od błędu logicznego.
- Zastosować narzędzia deweloperskie przeglądarki (konsola, debugger) do analizy problemu.
- Zastosować rutynę myślenia krytycznego „Widzę – Myślę – Zastanawiam się” do analizy cudzego kodu.
- Przewidzieć skutki braku walidacji danych wejściowych (np. w formularzach).
- Naprawić błędny skrypt, uzasadniając swoją decyzję (argumentacja).

KSZTAŁTOWANE KOMPETENCJE z „profilu absolwenta”

- Krytyczne myślenie: umiejętność weryfikacji informacji (danych wejściowych) i kwestionowania poprawności kodu, który na pierwszy rzut oka „działa”.
- Rozwiązywanie problemów: identyfikacja przyczyn błędów (root cause analysis).
- Współpraca: umiejętność pracy w parze (pair programming) i komunikowania technicznego problemu.

Metody pracy:

- Metoda problemowa (analiza „zepsutego” kodu).
- Rutyny Myślenia Krytycznego: „Widzę – Myślę – Zastanawiam się”
- Praca w parach
- Burza mózgów

Środki dydaktyczne:

- Komputery z dostępem do edytora kodu (np. VS Code) i przeglądarki internetowej.
- Przygotowany plik bledny_kod.js (zawierający pułapki, np. koercję typów, pętlę nieskończoną, błąd zakresu zmiennych).
- Projektor/tablica multimedialna do wyświetlenia kodu.

PRZEBIEG ZAJĘĆ: czas trwania zajęć 45 minut

Część wstępna - rozpoczynanie lekcji (10 minut)

1. **Czynności organizacyjne:** Sprawdzenie obecności.
2. Nauczyciel wyświetla na tablicy krótki fragment kodu JS:

```
JavaScript

let wiek = prompt("Ile masz lat?");

let wynik = wiek + 10;

console.log("Za 10 lat będziesz mieć: " + wynik);
```

Pytanie do klasy: „Użytkownik wpisuje 18. Co wyświetli konsola? 28 czy 1810?”.

3. **Wprowadzenie:** Uczniowie zauważają, że wynikiem jest „1810” (konkatenacja stringów). Nauczyciel wprowadza cel lekcji: Dziś nie piszemy nowego kodu, ale uczymy się **myśleć krytycznie** o tym, co już zostało napisane. Będziemy szukać błędów logicznych, których kompilator nam nie podkreśli na czerwono.

Część zasadnicza (25 minut)

1. **Analiza przypadku (Praca w parach):**
   - Uczniowie otrzymują plik z bardziej złożonym kodem (np. prosty kalkulator walut lub formularz logowania), który zawiera ukryte błędy logiczne (np. użycie operatora przypisania = zamiast porównania == w instrukcji if, problem z zasięgiem zmiennych var vs let).
2. **Rutyna Krytycznego Myślenia „Widzę – Myślę – Zastanawiam się”:**
   - Uczniowie nie mogą od razu naprawiać kodu. Mają 5 minut na wypisanie w zeszycie/pliku tekstowym:
     - **WIDZĘ:** Co dokładnie robi kod? (Fakty, np. „Pobiera wartość z inputa”).
     - **MYŚLĘ:** Gdzie może być problem? (Hipotezy, np. „Myślę, że typ zmiennej jest tekstowy, a nie liczbowy”).
     - **ZASTANAWIAM SIĘ:** Co się stanie, jeśli użytkownik wpisze litery zamiast cyfr? (Pytania testowe).

3. **Weryfikacja i naprawa:**
    - Uczniowie uruchamiają kod, otwierają Konsolę Deweloperską (F12) i testują swoje hipotezy.
    - Naprawiają błędy, stosując funkcje walidujące (np. isNaN(), parseInt()).

Prezentacja rozwiązań: Wybrane pary przedstawiają, jaki błąd znalazły i dlaczego on wystąpił. Nauczyciel prosi o uzasadnienie: „Dlaczego to rozwiązanie jest bezpieczniejsze?".

Podsumowanie lekcji –ewaluacja

1. Nauczyciel zadaje pytanie: „Dlaczego błędy logiczne są trudniejsze do wykrycia niż błędy składniowe?". (Oczekiwana odpowiedź: Bo program działa, ale daje złe wyniki, co może prowadzić do poważnych konsekwencji w firmach/bankach).
2. Uczniowie wchodzą na Menti/Forms lub piszą na kartce jedną rzecz: „Podaj przykład sytuacji z życia codziennego, w której brak krytycznego sprawdzenia danych (walidacji) doprowadził do błędu".

Komentarz dydaktyczny -uwzględnienie SPE uczniów , sposób/ forma oceniania, sposób refleksji uczniowskiej- samoocena.

- **Dostosowanie do SPE (Specjalne Potrzeby Edukacyjne):**
    - Uczniowie z dysleksją: korzystają z kolorowania składni w edytorze kodu (ułatwia percepcję) oraz pracują w parze z uczniem wspierającym (peer tutoring). Kod jest dostępny w formie cyfrowej (możliwość powiększenia czcionki).
    - Uczniowie zdolni: otrzymują zadanie dodatkowe – zabezpieczenie kodu przed atakiem typu XSS (wpisanie skryptu w pole formularza).
- **Sposób oceniania:**
    - Ocena kształtująca (formatywna): Nauczyciel obserwuje pracę w parach i sposób argumentacji podczas rutyny „Widzę-Myślę". Plusy za trafne zidentyfikowanie „root cause" (przyczyny źródłowej).
- **Refleksja uczniowska:**
    - Metoda świateł: Uczniowie oznaczają przy wyjściu (np. na marginesie kartki): Zielony (rozumiem debugowanie), Żółty (mam wątpliwości co do typów zmiennych), Czerwony (nie rozumiem, jak znaleźć błąd logiczny).

Bibliografia,

1. Dokumentacja MDN Web Docs (Mozilla Developer Network) – sekcja JavaScript.
2. Ritchhart, R., Church, M., & Morrison, K. (2011). *Making Thinking Visible*. (Rutyny myślenia krytycznego).
3. Podstawa programowa kształcenia w zawodzie technik informatyk (351203).

ZAŁĄCZNIK 1: Kod źródłowy (Wersja z błędami)

Skopiuj poniższy kod do pliku o nazwie sklep_logika.js lub udostępnij go na platformie typu JSFiddle/CodePen.

```
/*
 * SCENARIUSZ: Prosty kalkulator ceny w sklepie internetowym.
 * Skrypt ma pobrać cenę produktu, doliczyć podatek VAT (23%)
 * oraz przyznać rabat 20 zł, jeśli zakupy robi osoba powyżej 60 roku życia (Senior).
 */

function obliczKoszyk() {
    console.log("--- Rozpoczynam obliczenia ---");

    // 1. Pobieranie danych (Symulacja danych z formularza)
    let cenaProduktu = prompt("Podaj cenę netto produktu (w PLN):");
    let wiekKlienta = prompt("Podaj swój wiek:");
    const stawkaVAT = 0.23;

    // 2. Obliczanie podatku
    // Oczekujemy: Cena 100 -> VAT 23 -> Brutto 123
    let kwotaVAT = cenaProduktu * stawkaVAT;
    let cenaBrutto = cenaProduktu + kwotaVAT;

    console.log("Wyliczony VAT: " + kwotaVAT);
    console.log("Cena Brutto przed rabatem: " + cenaBrutto);

    // 3. Weryfikacja rabatu dla Seniora
    // Jeśli klient ma 60 lat lub więcej, odejmujemy 20 zł
    if (wiekKlienta > 60); {
        console.log("Przyznano rabat dla Seniora!");
        cenaBrutto = cenaBrutto - 20;
    }

    // 4. Wynik końcowy
    // Błąd logiczny: jeśli cena wyjdzie ujemna (produkt tańszy niż rabat), sklep traci!
    document.write("Do zapłaty: " + cenaBrutto + " PLN");
}

// Uruchomienie funkcji
obliczKoszyk();
```

ZAŁĄCZNIK 2: Karta Pracy Ucznia (Rutyna Krytycznego Myślenia)

Polecenie: Uruchom powyższy kod w przeglądarce. Wpisz cenę 100 i wiek 25.

Obserwuj wynik, a następnie wypełnij tabelę, nie patrząc od razu w kod rozwiązania.

Rutyna	**Pytania pomocnicze**	**Twoje notatki / Obserwacje**
WIDZĘ (Fakty)	Co dokładnie wypisała konsola/ekran? Czy wynik jest matematycznie poprawny? *(Np. Czy 100 + 23 to na pewno to, co widzę?)*	
MYŚLĘ (Hipotezy)	Dlaczego program zachował się w ten sposób? Jaki typ danych ma zmienna `cenaProduktu`? Dlaczego rabat naliczył się (lub nie) wbrew logice?	
ZASTANAWIAM SIĘ (Testy)	Co się stanie, jeśli wpiszę cenę `10` zł i wiek `70`? Czy program pozwoli mi "kupić" towar i jeszcze dopłaci (wynik ujemny)?	

ZAŁĄCZNIK 3: Klucz odpowiedzi dla nauczyciela

Poniżej znajduje się analiza błędów zawartych w kodzie, którą możesz wykorzystać podczas podsumowania lekcji.

1. Błąd Typów (Type Coercion) – "Pułapka Stringów"
Gdzie: Linia let cenaBrutto = cenaProduktu + kwotaVAT;

Objaw: Po wpisaniu ceny 100, kwotaVAT wynosi 23 (JS niejawnie rzutuje string na liczbę przy mnożeniu), ALE przy dodawaniu (+) następuje konkatenacja.

Wynik: Zamiast 123, otrzymujemy 10023 (ciąg znaków "100" połączony z liczbą 23).

Naprawa: Należy użyć parseFloat(cenaProduktu) lub Number(cenaProduktu) zaraz po pobraniu danych.

2. Błąd Składniowy/Logiczny – "Średnik Widmo"
Gdzie: Linia if (wiekKlienta > 60); {

Objaw: Średnik postawiony zaraz po warunku if kończy instrukcję. Blok kodu w klamrach { ... } traktowany jest jako zwykły blok kodu, niezależny od warunku.

Skutek: Rabat jest przyznawany KAŻDEMU, niezależnie od wieku. Kod w klamrach wykonuje się zawsze.

Naprawa: Usunięcie średnika po nawiasie zamykającym warunek.

3. Błąd Logiki Biznesowej – "Ujemna Cena"
Gdzie: Końcowe odejmowanie rabatu.

Objaw: Jeśli produkt kosztuje 10 zł, a rabat wynosi 20 zł, cena końcowa wyniesie -10 zł.

Skutek: Sklep "dopłaca" klientowi za zakupy. Jest to krytyczny błąd w systemach e-commerce.

Naprawa: Dodanie warunku zabezpieczającego (tzw. Guard Clause):

JavaScript

```
cenaBrutto = cenaBrutto - 20;
if (cenaBrutto < 0) cenaBrutto = 0; // Cena nie może być ujemna
```

Poprawiony kod (Wzorzec):
JavaScript

```
function obliczKoszykPoprawny() {
    let inputCena = prompt("Podaj cenę netto:");
    let inputWiek = prompt("Podaj wiek:");
```

```
    // Walidacja danych wejściowych (Krytyczne myślenie: "A co jeśli wpiszą tekst?")
    if (isNaN(inputCena) || inputCena === "") {
        alert("Błąd: Cena musi być liczbą!");
        return;
    }

    let cenaProduktu = parseFloat(inputCena); // Konwersja typu
    let wiekKlienta = parseInt(inputWiek);
    const stawkaVAT = 0.23;

    let kwotaVAT = cenaProduktu * stawkaVAT;
    let cenaBrutto = cenaProduktu + kwotaVAT; // Teraz sumuje liczby: 100 + 23 = 123

    // Poprawny warunek bez średnika
    if (wiekKlienta >= 60) {
        console.log("Przyznano rabat.");
        cenaBrutto -= 20;
    }

    // Zabezpieczenie przed ceną ujemną
    if (cenaBrutto < 0) cenaBrutto = 0;

    // Zaokrąglenie do 2 miejsc po przecinku (problem zmiennoprzecinkowy)
    console.log("Do zapłaty: " + cenaBrutto.toFixed(2) + " PLN");
}
```