Zagadnienia teoretyczne – Stałe, zmienne i operatory w JavaScript

Przedmiot: Programowanie aplikacji internetowych
Klasa: 1 Technik Informatyk
Forma: Lekcja odwrócona – materiały do samodzielnego opracowania

Cel lekcji

Po przerobieniu tych materiałów będziesz potrafił:

- Wyjaśnić, czym jest zmienna i stała w JavaScript
- Utworzyć zmienną używając let, const i var
- Rozróżnić różne typy danych (number, string, boolean, undefined, null)
- Stosować operatory arytmetyczne (+, -, *, /, %, **)
- Używać operatorów przypisania (=, +=, -=, *=, /=, %=)
- Zastosować operatory inkrementacji i dekrementacji (++, --)
- Łączyć teksty używając operatora konkatenacji (+)
- Rozumieć kolejność wykonywania operacji (precedencja operatorów)

KROK 1: Czym jest zmienna?

Wyjaśnienie:
Zmienna to jak pudełko, w którym przechowujesz dane. Możesz do niego coś włożyć, zmienić zawartość lub odczytać to, co jest w środku.
Przykład z życia codziennego:
Wyobraź sobie, że masz pudełko z etykietą 'mojaLiczba'. Możesz włożyć do niego liczbę 10, a później zmienić na 20. W JavaScript wygląda to tak:

```
let mojaLiczba = 10;  // tworzymy zmienną i wkładamy 10
mojaLiczba = 20;      // zmieniamy zawartość na 20
console.log(mojaLiczba);  // wyświetlamy zawartość: 20
```

Dlaczego zmienne są ważne?

- Pozwalają przechowywać dane, których będziemy potrzebować później
- Umożliwiają wielokrotne użycie tych samych wartości
- Ułatwiają zmianę wartości w jednym miejscu zamiast wielu
- Czytelność kodu – zamiast magicznych liczb używamy nazw

KROK 2: Tworzenie zmiennych – let, const, var

1. let – zmienna, którą można zmienić

```
let wiek = 18;
wiek = 19;  // można zmienić
console.log(wiek);  // 19
```

2. const – stała, której nie można zmienić

```
const PI = 3.14;
// PI = 3.15;  // BŁĄD! Nie można zmienić stałej
console.log(PI);  // 3.14
```

WAŻNE: const używamy dla wartości, które nie powinny się zmieniać (np. stałe matematyczne, nazwa firmy).
3. var – stary sposób (lepiej nie używać)
var jest starym sposobem tworzenia zmiennych. W nowoczesnym JavaScript używamy let i const.

```
var starySposob = 5;  // działa, ale lepiej używać let
```

Przykład porównawczy:

```
let zmienna = 10;     // można zmienić
const stala = 20;     // nie można zmienić

zmienna = 15;         // ✓ działa
// stala = 25;        // ✗ BŁĄD!
```

KROK 3: Typy danych w JavaScript

1. Number (liczba)

```
let liczba = 42;
let ujemna = -15;
let dziesietna = 3.14;
let zero = 0;
```

2. String (tekst)

```
let imie = 'Jan';          // pojedyncze cudzysłowy
let nazwisko = "Kowalski";  // podwójne cudzysłowy
let tekst = `Witaj!`;      // backticki (nowoczesne)
```
WAŻNE: Liczba w cudzysłowach to tekst, nie liczba!
```
let liczba = 42;       // typ: number
let tekst = '42';      // typ: string
```
3. Boolean (prawda/fałsz)
```
let jestPelnoletni = true;
let czyJestDeszcz = false;
```
4. undefined (niezdefiniowane)
```
let x;               // utworzenie bez wartości
console.log(x);        // undefined
```
5. null (puste)
```
let brakDanych = null;  // świadomie ustawione na 'puste'
```
Sprawdzanie typu danych:
```
let wiek = 18;
let imie = 'Anna';
let czyPrawda = true;

console.log(typeof wiek);      // 'number'
console.log(typeof imie);     // 'string'
console.log(typeof czyPrawda); // 'boolean'
```
KROK 4: Operatory arytmetyczne
Podstawowe operatory:
- + – dodawanie
- - – odejmowanie
- * – mnożenie
- / – dzielenie
- % – modulo (reszta z dzielenia)
- ** – potęgowanie

Przykłady:
```
let a = 10;
let b = 3;

console.log(a + b);   // 13 (dodawanie)
console.log(a - b);   // 7 (odejmowanie)
console.log(a * b);   // 30 (mnożenie)
console.log(a / b);   // 3.333... (dzielenie)
console.log(a % b);   // 1 (reszta z dzielenia 10 przez 3)
console.log(a ** b);  // 1000 (10 do potęgi 3)
```
Operator modulo (%) – szczególnie przydatny:
```
console.log(10 % 3);   // 1 (10 dzielone przez 3 = 3 reszty 1)
console.log(10 % 2);   // 0 (10 jest parzyste)
console.log(11 % 2);   // 1 (11 jest nieparzyste)
console.log(15 % 5);   // 0 (15 dzieli się przez 5)
```
KROK 5: Operatory przypisania
Podstawowy operator przypisania (=):
```
let x = 5;   // przypisujemy wartość 5 do zmiennej x
```
Operatory przypisania z operacją:
- += – dodaj i przypisz (x += 5 to to samo co x = x + 5)
- -= – odejmij i przypisz (x -= 3 to to samo co x = x - 3)
- *= – pomnóż i przypisz (x *= 2 to to samo co x = x * 2)
- /= – podziel i przypisz (x /= 4 to to samo co x = x / 4)
- %= – modulo i przypisz (x %= 3 to to samo co x = x % 3)

Przykłady:
```
let liczba = 10;
```

```
liczba += 5;   // liczba = 15 (10 + 5)
liczba -= 3;   // liczba = 12 (15 - 3)
liczba *= 2;   // liczba = 24 (12 * 2)
liczba /= 4;   // liczba = 6 (24 / 4)
liczba %= 5;   // liczba = 1 (6 % 5)

console.log(liczba);  // 1
```

KROK 6: Operatory inkrementacji i dekrementacji

Inkrementacja (++):

Zwiększa wartość o 1.

```
let x = 5;
x++;           // x = 6
console.log(x);   // 6
```

Dekrementacja (--):

Zmniejsza wartość o 1.

```
let y = 10;
y--;           // y = 9
console.log(y);   // 9
```

Prefix vs Postfix:

```
let a = 5;
let b = 5;

console.log(++a);  // 6 (zwiększa PRZED użyciem)
console.log(b++);  // 5 (zwiększa PO użyciu)
console.log(b);    // 6 (teraz b = 6)
```

KROK 7: Konkatenacja (łączenie tekstów)

Operator + dla tekstów:

Gdy używamy + z tekstami (stringami), JavaScript łączy je razem.

```
let imie = 'Jan';
let nazwisko = 'Kowalski';

let pelneImie = imie + ' ' + nazwisko;
console.log(pelneImie);  // 'Jan Kowalski'
```

Template literals (nowoczesny sposób):

```
let wiek = 18;
let wiadomosc = `Mam ${wiek} lat`;
console.log(wiadomosc);  // 'Mam 18 lat'
```

Template literals używają backticków (`) i pozwalają wstawiać zmienne używając ${}.

UWAGA: + dla liczb vs tekstów

```
let a = 5;
let b = 3;
let c = '5';
let d = '3';

console.log(a + b);   // 8 (liczby - dodawanie)
console.log(c + d);   // '53' (teksty - łączenie)
console.log(a + c);   // '53' (liczba + tekst = tekst)
```

KROK 8: Kolejność wykonywania operacji (precedencja)

Jak matematyka w szkole:

JavaScript wykonuje operacje w określonej kolejności, podobnie jak w matematyce:

- Najpierw: nawiasy ()
- Potem: potęgowanie **
- Następnie: mnożenie *, dzielenie /, modulo %
- Na końcu: dodawanie +, odejmowanie -

Przykłady:

```
console.log(2 + 3 * 4);     // 14 (najpierw 3*4=12, potem 2+12=14)
console.log((2 + 3) * 4);   // 20 (najpierw 2+3=5, potem 5*4=20)
console.log(10 / 2 + 3);    // 8 (najpierw 10/2=5, potem 5+3=8)
console.log(2 ** 3 + 1);    // 9 (najpierw 2**3=8, potem 8+1=9)
```

WAŻNE: Używaj nawiasów, gdy chcesz zmienić kolejność!

KROK 9: Nazewnictwo zmiennych – dobre praktyki

Zasady tworzenia dobrych nazw zmiennych:

- Używaj angielskich słów lub polskich, ale konsekwentnie
- Nazwa powinna opisywać, co przechowuje zmienna
- Używaj camelCase (pierwsza litera mała, kolejne słowa z dużą literą)
- Nie używaj polskich znaków diakrytycznych (ą, ć, ę, ł, ń, ó, ś, ź, ż)
- Nazwa może zawierać litery, cyfry, _ (podkreślenie), $ (dolar)
- Nazwa NIE MOŻE zaczynać się od cyfry
- Unikaj słów zastrzeżonych (let, const, if, else, function, itp.)

Dobre przykłady:

```
let userName = 'Jan';
let wiekUzytkownika = 18;
let czyJestPelnoletni = true;
const MAX_LICZBA = 100;
```

Złe przykłady:

```
let x = 5;                  // nie opisuje, co przechowuje
let user-name = 'Jan';      // myślnik nie jest dozwolony
let 123abc = 5;             // nie może zaczynać się od cyfry
let let = 5;                // let jest słowem zastrzeżonym
let imię = 'Jan';           // polskie znaki (lepiej unikać)
```

KROK 10: Podsumowanie

Zapamiętaj:

- Zmienna to pudełko do przechowywania danych
- let – zmienna, którą można zmienić
- const – stała, której nie można zmienić
- Główne typy danych: number, string, boolean, undefined, null
- Operatory arytmetyczne: +, -, *, /, %, **
- Operatory przypisania: =, +=, -=, *=, /=, %=
- Inkrementacja ++ zwiększa o 1, dekrementacja -- zmniejsza o 1
- Operator + łączy teksty (konkatenacja)
- Template literals (`) ułatwiają łączenie tekstów ze zmiennymi
- Kolejność wykonywania operacji: nawiasy → potęgowanie → mnożenie/dzielenie → dodawanie/odejmowanie

Następne kroki:

Teraz przejdź do 'Przykłady do wykonania' i spróbuj samodzielnie rozwiązać zadania!