Programowanie 2022 / 2023

Laboratorium 5

Przechowywanie danych w zmiennych i tablicach, wykorzystanie operatorów

Wstęp

Przechowywanie danych w zmiennych

http://kursjs.pl/kurs/super-podstawy/zmienne

let name = "John"; // zmienna przechowująca imię "John"
let age = 30; // zmienna przechowująca wiek 30
let isStudent = false; // zmienna przechowująca wartość logiczną false

W języku JavaScript istnieją trzy różne sposoby deklarowania zmiennych: let, const i var.

let: Jest to zmienna, której wartość może być zmieniana po jej zadeklarowaniu. Możemy ją zadeklarować używając słowa kluczowego let, na przykład:

let x = 10; // zadeklarowanie zmiennej x i przypisanie jej wartości 10

x = 20; // zmiana wartości zmiennej x na 20

const: Jest to zmienna, której wartość jest stała i nie może być zmieniana po jej zadeklarowaniu. Możemy ją zadeklarować używając słowa kluczowego const, na przykład:

const y = 30; // zadeklarowanie zmiennej y i przypisanie jej wartości 30

// y = 40; // błąd! nie można zmieniać wartości zmiennej y

var: Jest to starszy sposób deklarowania zmiennych w JavaScript, który nie jest już zalecany do stosowania. Zmienne zadeklarowane za pomocą var mają zakres funkcyjny, co może prowadzić do nieprzewidywalnego zachowania w pewnych przypadkach. Zamiast tego, zaleca się używanie let lub const w zależności od potrzeb.

var z = 50; // zadeklarowanie zmiennej z i przypisanie jej wartości 50

z = 60; // zmiana wartości zmiennej z na 60

Let i const są bardziej nowoczesnymi i zalecanymi sposobami deklarowania zmiennych w JavaScript, gdzie let pozwala na zmianę wartości, a const tworzy zmienną tylko do odczytu. Z kolei var jest starszym sposobem, który nie jest już zalecany do stosowania w nowych projektach.

Przechowywanie danych w tablicach

http://kursjs.pl/kurs/super-podstawy/tablice

let numbers = [1, 2, 3, 4, 5]; // tablica przechowująca liczby 1, 2, 3, 4, 5
let names = ["John", "Alice", "Bob"]; // tablica przechowująca imiona "John", "Alice" i "Bob"
let fruits = ["apple", "banana", "orange"]; // tablica przechowująca owoce "apple", "banana" i "orange"

Zbiory danych można przechowywać w tablicach, które są jednym z podstawowych typów danych używanych do przechowywania kolekcji wartości.

```
// Deklarowanie i inicjalizowanie tablicy
let liczby = [1, 2, 3, 4, 5]; // Deklaracja tablicy liczby i przypisanie jej wartości

// Wyświetlanie zawartości tablicy w konsoli
console.log(liczby); // Wypisanie zawartości tablicy liczby w konsoli

// Dostęp do elementów tablicy
console.log(liczby[0]); // Wyświetlenie pierwszego elementu tablicy
console.log(liczby[2]); // Wyświetlenie trzeciego elementu tablicy

// Modyfikowanie wartości elementów tablicy
liczby[1] = 10; // Zmiana wartości drugiego elementu tablicy na 10

// Dodawanie nowych elementów do tablicy
liczby.push(6); // Dodanie wartości 6 na koniec tablicy

// Usuwanie elementów z tablicy
liczby.splice(3, 1); // Usunięcie jednego elementu z tablicy o indeksie 3

// Pobieranie długości tablicy
console.log(liczby.length); // Wyświetlenie długości tablicy

// Iterowanie po elementach tablicy
for (let i = 0; i < liczby.length; i++) {
  console.log(liczby[i]); // Wyświetlanie kolejnych elementów tablicy w konsoli
}
```

Tablice w JavaScript pozwalają nam na przechowywanie wielu wartości w jednym miejscu i dostęp do nich za pomocą indeksów. Możemy modyfikować wartości elementów tablicy, dodawać nowe elementy na koniec tablicy, usuwać elementy oraz iterować po elementach tablicy przy użyciu pętli. Tablice są bardzo użytecznym narzędziem do przechowywania i manipulowania kolekcjami danych w JavaScript.

Tablice posiadają wiele wbudowanych metod, które można wykorzystać do manipulowania danymi w tablicach. Oto kilka przykładowych metod tablic:

1. push(): Dodaje jeden lub więcej elementów na koniec tablicy i zwraca nową długość tablicy.

```
const tablica = [1, 2, 3];
tablica.push(4, 5);
console.log(tablica); // [1, 2, 3, 4, 5]
```

2. pop(): Usuwa ostatni element z tablicy i zwraca ten element.

```
const tablica = [1, 2, 3];
const usunietyElement = tablica.pop();
console.log(usunietyElement); // 3 console.log(tablica); // [1, 2]
```

3. shift(): Usuwa pierwszy element z tablicy i zwraca ten element.

```
const tablica = [1, 2, 3];
const usunietyElement = tablica.shift();
console.log(usunietyElement); // 1
console.log(tablica); // [2, 3]
```

4. unshift(): Dodaje jeden lub więcej elementów na początek tablicy i zwraca nową długość tablicy.

```
const tablica = [1, 2, 3];
tablica.unshift(-1, 0);
console.log(tablica); // [-1, 0, 1, 2, 3]
```

5. slice(): Zwraca nową tablicę, która jest kopią części tablicy, określonej przez indeks początkowy i indeks końcowy (bez inkluzji).

```
const tablica = [1, 2, 3, 4, 5];
const podtablica = tablica.slice(1, 4);
console.log(podtablica); // [2, 3, 4]
```

6. splice(): Zmienia zawartość tablicy, usuwając, zamieniając lub dodając elementy na określonych indeksach.

```
const tablica = [1, 2, 3, 4, 5];
tablica.splice(2, 1); // Usunięcie jednego elementu na indeksie 2
console.log(tablica); // [1, 2, 4, 5]
tablica.splice(1, 0, 6, 7); // Dodanie elementów 6 i 7 na indeksie 1
console.log(tablica); // [1, 6, 7, 2, 4, 5]
```

7. concat(): Zwraca nową tablicę, która jest połączeniem dwóch lub więcej tablic.

```
const tablica1 = [1, 2, 3];
const tablica2 = [4, 5, 6];
const polaczonaTablica = tablica1.concat(tablica2);
console.log(polaczonaTablica); // [1, 2, 3, 4, 5, 6]
```

8. join(): Zwraca ciąg znaków, który jest wynikiem połączenia elementów tablicy w ciąg znaków, oddzielonych podanym separatorem.

```
const tablica = [1, 2, 3, 4, 5];
const lancuchZnakow = tablica.join('-');
console.log(lancuchZnakow); // "1-2-3-4-5"
```

9. find(): Zwraca pierwszy element w tablicy, który spełnia określony warunek w postaci funkcji zwrotnej.

```
const tablica = [1, 2, 3, 4, 5];
const znalezionyElement = tablica.find(element => element > 3);
console.log(znalezionyElement); // 4
```

10. filter(): Zwraca nową tablicę z elementami, które spełniają określony warunek w postaci funkcji zwrotnej.

```
const tablica = [1, 2, 3, 4, 5];
const przefiltrowanaTablica = tablica.filter(element => element % 2 === 0);
console.log(przefiltrowanaTablica); // [2, 4]
```

11. map(): Zwraca nową tablicę z wynikami wywołania funkcji na każdym elemencie tablicy.

```
const tablica = [1, 2, 3, 4, 5];
const przetworzonaTablica = tablica.map(element => element * 2);
console.log(przetworzonaTablica); // [2, 4, 6, 8, 10]
```

12. reduce(): Zwraca pojedynczą wartość, która jest wynikiem redukcji elementów tablicy do jednej wartości na podstawie określonej funkcji zwrotnej.

```
const tablica = [1, 2, 3, 4, 5];
const suma = tablica.reduce((a, b) => a + b, 0);
console.log(suma); // 15
```

13. forEach(): jest używana do iteracji (przechodzenia) przez elementy w tablicy i wykonywania określonej funkcji zwrotnej (callback) dla każdego z nich. Jest to popularna i prostsza alternatywa dla pętli for lub for...of do iteracji przez elementy w tablicy.

```
tablica.forEach(function(element, indeks, tablica) {
  // kod wykonywany dla każdego elementu w tablicy
});
```

Operator rozprzestrzeniania, znany również jako "spread operator", to operator który pozwala na rozbijanie wartości z jednej tablicy lub obiektu na oddzielne elementy. Może być używany do tworzenia kopii tablicy, łączenia kilku tablic w jedną, przekazywania argumentów do funkcji w bardziej czytelny sposób oraz kopiowania obiektów.

Przykładowe użycie operatora rozprzestrzeniania na tablicach:

Kopia tablicy:

```
let tablica1 = [1, 2, 3];
let tablica2 = [...tablica1]; // kopia tablica1
```

Łączenie tablic:

```
let tablica1 = [1, 2, 3];
let tablica2 = [4, 5, 6];
let tablica3 = [...tablica1, ...tablica2]; // [1, 2, 3, 4, 5, 6]
```

Kopia obiektu:

```
let obiekt1 = { a: 1, b: 2, c: 3 };
let obiekt2 = { ...obiekt1 }; // kopia obiekt1
```

Operator rozprzestrzeniania jest bardzo przydatnym narzędziem w języku JavaScript, które pozwala na bardziej zwięzłe i efektywne operacje na tablicach i obiektach. Warto zapoznać się z jego zastosowaniami i możliwościami w celu ułatwienia i zoptymalizowania kodu.

Wykorzystanie operatorów

http://kursjs.pl/kurs/super-podstawy/operatory
Wykorzystanie operatorów arytmetycznych:

```
let a = 10;
let b = 5;
let sum = a + b; // suma a i b
let difference = a - b; // różnica a i b
let product = a * b; // iloczyn a i b
let quotient = a / b; // iloraz a i b
let remainder = a % b; // reszta z dzielenia a przez b
```

Wykorzystanie operatorów porównania:

```
let x = 10;
let y = 5;
console.log(x > y); // true - x jest większe od y
console.log(x < y); // false - x jest mniejsze od y
console.log(x >= y); // true - x jest większe lub równe y
console.log(x <= y); // false - x jest mniejsze lub równe y
console.log(x === y); // false - x jest równe y (porównanie wartości i typów)
console.log(x !== y); // true - x jest różne od y (porównanie wartości i typów)
```

Wykorzystanie operatorów logicznych:

```
let isSunny = true;
let isWarm = false;
console.log(isSunny && isWarm); // false - oba warunki muszą być spełnione
console.log(isSunny || isWarm); // true - jeden z warunków musi być spełniony
console.log(!isSunny); // false - negacja wartości logicznej
```

Wykorzystanie operatorów przypisania:

```
let num = 10;
num += 5; // num = num + 5
console.log(num); // 15
num -= 3; // num = num - 3
console.log(num); // 12
num *= 2; // num = num * 2
console.log(num); // 24
num /= 4; // num = num / 4
console.log(num); // 6
num %= 3; // num = num % 3
console.log(num); // 0
```

Zadania

Zadanie 1. Napisz program, który poprosi użytkownika o podanie swojego wieku. Zapisz to do zmiennej i sprawdź, czy jest pełnoletni. Wypisz odpowiedni komunikatu na ekranie.

Zadanie 2. Dokonaj konwersji stopni Celsjusza na stopnie Fahrenheita na podstawie podanej temperatury. Wypisz wartość konsoli.

Zadanie 3. Stwórz tablice liczb i wykonaj na niej poniższe operacje. Do operacji wykorzystaj metody tablic. Wynik operacji wypisz w konsoli.

- zsumuj wartości
- znajdź liczby parzyste
- pomnóż wartości razy 3
- dodaj do tablicy swój numer albumu. Znajdź jego index w tablicy
- oblicz średnią arytmetyczna
- znajdź największą liczbę
- zlicz ilość wystąpień wybranej wartości

Zadanie 4. Stwórz tablice z 100 elementami zawierający kolejnymi liczbami ciągu Fibonacciego.