

## Лабораторна робота №1 (2 год. Максимальна оцінка 3 бали)

**Тема:** Операції над масивами даних.

**Мета:** Засвоєння основних операцій над масивами

### ТЕОРЕТИЧНІ ВІДОМОСТІ

Об'єкти дозволяють зберігати дані із строковими ключами. Але досить часто ми розуміємо, що нам необхідна упорядкована колекція даних, в якій присутні 1-й, 2-й, 3-й елементи і т.д. Наприклад, вона знадобиться нам для зберігання списку чогось: користувачів, товарів, елементів HTML і т.д.

В цьому випадку використовувати об'єкт незручно, так як він не надає методів управління порядком елементів. Ми не можемо вставити нову властивість «між» вже існуючими. Об'єкти просто не призначені для цих цілей.

Для зберігання впорядкованих колекцій існує особлива структура даних, яка називається масив, Array.

### ОГОЛОШЕННЯ

Існує два варіанти синтаксису для створення порожнього масиву:

```
let arr = new Array();  
let arr = [];
```

Практично завжди використовується другий варіант синтаксису. У дужках ми можемо вказати початкові значення елементів:

```
let fruits = ["Яблуко", "Апельсин", "Слива"];
```

Елементи масиву нумеруються, починаючи з нуля.

Ми можемо отримати елемент, вказавши його номер у квадратних дужках:

```
let fruits = ["Яблуко", "Апельсин", "Слива"];  
  
alert( fruits[0] ); // Яблуко  
alert( fruits[1] ); // Апельсин  
alert( fruits[2] ); // Слива
```

Ми можемо замінити елемент:

```
fruits[2] = 'Груша'; // тепер ["Яблуко", "Апельсин",  
"Груша"]
```

Додати новий до існуючого масиву:

```
fruits[3] = 'Лимон'; // тепер ["Яблуко", "Апельсин",  
"Груша", "Лимон"]
```

Загальна кількість елементів масиву міститься в його властивості length:

```
let fruits = ["Яблуко", "Апельсин", "Слива"];  
alert( fruits.length ); // 3
```

Вивести масив цілком можна за допомогою alert.

```
let fruits = ["Яблуко", "Апельсин", "Слива"];  
alert( fruits ); // Яблуко, Апельсин, Слива
```

### Методи pop / push, shift / unshift

Черга - один з найпоширеніших варіантів застосування масиву. В області комп'ютерних наук так називається впорядкована колекція елементів, що підтримує два види операцій:

push додає елемент в кінець.

shift видаляє елемент на початку, зрушуючи чергу, так що другий елемент стає першим.

Масиви підтримують обидві операції.

На практиці необхідність в цьому виникає дуже часто. Наприклад, черга повідомлень, які треба показати на екрані.

Існує й інший варіант застосування для масивів - структура даних, яка називається стек .

Вона підтримує два види операцій:

push додає елемент в кінець.

pop видаляє останній елемент.

Таким чином, нові елементи завжди додаються або видаляються з «кінця».

Прикладом стека зазвичай служить колода карт: нові карти кладуться наверх і беруться теж зверху:

Масиви в JavaScript можуть працювати і як чергу, і як стек. Ми можемо додавати / видаляти елементи як на початок, так і в кінець масиву.

У комп'ютерних науках структура даних, що робить це можливим, називається двостороння чергу .

Методи, що працюють з кінцем масиву:

#### **pop**

Видаляє останній елемент з масиву і повертає його:

```
let fruits = ["Яблуко", "Апельсин", "Груша"];  
alert( fruits.pop() ); // видаляємо "Груша" та виводимо його  
alert( fruits ); // Яблуко, Апельсин
```

#### **push**

Додає елемент в кінець масиву:

```
let fruits = ["Яблуко", "Апельсин"];  
fruits.push("Груша");  
alert( fruits ); // Яблуко, Апельсин, Груша
```

Виклик `fruits.push(...)` рівнозначний `fruits[fruits.length] = ...`

Методи, що працюють з початком масиву:

#### **shift**

Видаляє з масиву перший елемент і повертає його:

```
let fruits = ["Яблуко", "Апельсин", "Груша"];  
alert( fruits.shift() ); // видаляємо Яблуко  
alert( fruits ); // Апельсин, Груша
```

#### **unshift**

Додає елемент в початок масиву:

```
let fruits = ["Апельсин", "Груша"];  
fruits.unshift('Яблуко');  
alert( fruits ); // Яблуко, Апельсин, Груша
```

Методи `push` і `unshift` можуть відразу кілька елементів:

```
let fruits = ["Яблуко"];
fruits.push("Апельсин", "Груша");
fruits.unshift("Ананас", "Лимон");
// ["Ананас", "Лимон", "Яблуко", "Апельсин", "Груша"]
alert( fruits );
```

### Внутрішній устрій масиву

Масив - це особливий підвид об'єктів. Квадратні дужки, які використовуються для того, щоб отримати доступ до властивості `arr[0]`- це по суті звичайний синтаксис доступу по ключу, як `obj[key]`, де в ролі `obj` у нас `arr`, а в якості ключа - числовий індекс.

Масиви розширюють об'єкти, так як передбачають спеціальні методи для роботи з впорядкованими колекціями даних, а також властивість `length`. Але в основі все одно лежить об'єкт.

## ЗАВДАННЯ ДЛЯ САМОСТІЙНОГО ВИКОНАННЯ.

**Завдання 1.** Операції з масивом. Вивід здійснити в консоль.

1. Створіть масив `auto` із елементами «Volvo» та «Audi».
2. Додайте "Toyota" в кінець.
3. Замініть значення всередині на «Ford». Ваш код для пошуку значення всередині повинен працювати для масивів з будь-якою довжиною.
4. Видаліть перший елемент масиву та покажіть його.
5. Вставте «BMW» та «Nissan» початок масиву.

**Завдання 2.** Сума введених чисел.

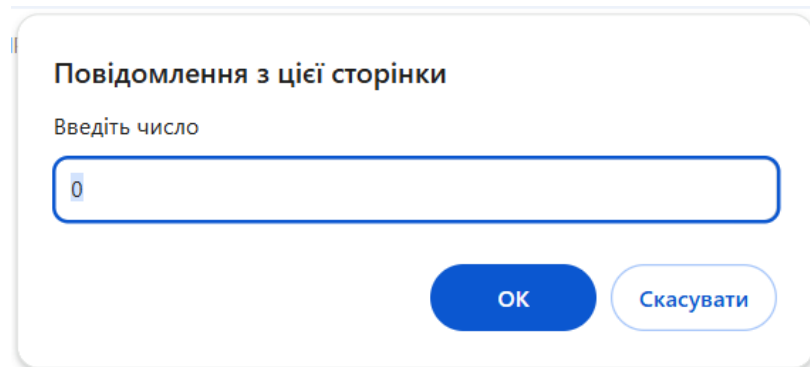
Напишіть функцію `sumInput()`, яка:

Просить користувача ввести значення, використовуючи `prompt` та зберігає їх у масив.

Закінчує запитувати значення, коли користувач введе не числове значення, порожній рядок або натисне «Скасувати».

Підраховує та повертає суму елементів масиву.

PS: Нуль 0 – вважається числом, не зупиняйте введення значень під час введення «0».



Повідомлення з цієї сторінки

Введіть число

0

OK Скасувати

**Завдання 3.** Масив паліндром.

Дано масив. Потрібно написати функцію, яка дозволяє повернути значення true, якщо масив є паліндромом, і false — якщо немає.

**Завдання 4.** Подвійний масив паліндром

Дано масив. Потрібно написати функцію, яка дозволяє повернути значення true, якщо масив є паліндромом та числа є паліндромами, і false — якщо немає.

**Завдання 5.** Кратність елементів масиву.

Потрібно написати функцію, що виводить в консоль числа з масиву за такими умовами:

- висновок fizz замість чисел, кратних 3;
- висновок buzz замість чисел, кратних 5;
- висновок fizzbuzz замість чисел, кратних як 3, так і 5.