# ПРАКТИЧНА РОБОТА 5

**Тема:** Функції JavaScript

Мета: Вивчити набір правил за якими будуються та використовуються функції JavaScript

**Питання для повторення:**

1. Визначення функцій.
2. Вбудовані функції.
3. Створення функцій користувачем.
4. Конструктори та методи.

**Завдання:**

Використовуючи вбудовані та самостійно створені функції для розв'язання задач у JavaScript.

**Теоретичні відомості:**

При створенні програми розумно виділити в ній логічно незалежні частини (підпрограми). Один раз створену і відлагоджену підпрограму можна використовувати довільну кількість разів. Основним елементом для реалізації підпрограм в мові JavaScript є функція.

Функції JavaScript виконують дії, а також можуть повертати значення. Функції також називають "глобальними методами". Функції об'єднують кілька операцій під одним ім'ям. Це дозволяє оптимізувати код. Можна написати набір операторів, привласнити йому ім'я, а потім виконувати весь набір, викликаючи його і передаючи йому необхідні дані.

Якщо функцію створюють, задаючи їй ім'я - то це іменована функція.

Якщо функція ініціалізується без імені - це анонімна функція. Анонімні функції як правило присвоюються змінним, можуть ставати методами об'єктів, бути «коллбек»-функціями тощо.

Щоб передати дані в функцію, їх необхідно покласти в дужки, розташовані після імені функції. Фрагменти даних, що передаються в функцію, називаються аргументами або параметрами. Деякі функції не приймають ніяких аргументів, тоді як інші приймають один або кілька аргументів. Функції в JavaScript можна запускати з будь-яким числом параметрів. Якщо в функцію передано менше параметрів, ніж є у визначенні, то відсутні параметри отримують значення undefined.

Змінні визначені всередині функції невидимі поза цією функцією, так як змінні визначаються в області видимості всередині функції. Функція визначена в глобальній області видимості має доступ до всіх змінних визначених у глобальній області видимості.

В JavaScript підтримуються два типи функцій: вбудовані і створені користувачами.

Функції JavaScript визначаються за допомогою ключового слова function. Можна використовувати іменовані функції або функціональний вираз. 1, Іменовані функції не виконуються відразу. Вони "зберігаються для подальшого використання" і будуть виконані після виклику у наступному коді.

```
// створення функцій
function functionName(parameters) {
  //code to be executed
}

function myFunction(a, b) {
  return a * b;
}
```

2. Функція JavaScript також може бути визначена за допомогою виразу -*expression.* Функціональний вираз можна зберегти в змінній. Функціональний вираз являє собою функцію, яка не має ім'я - анонімну функцію. Після того, як функціональний вираз збережений в змінній, змінна може бути використана як функція:

```
var x = function (a, b) {return a * b};
var z = x(4, 3);
```

Функції також можна визначити за допомогою вбудованого конструктора функцій JavaScript, який називається Function (). Конструктор Function створює новий об'єкт myFunction. В JavaScript кожна функція є об'єктом Function.

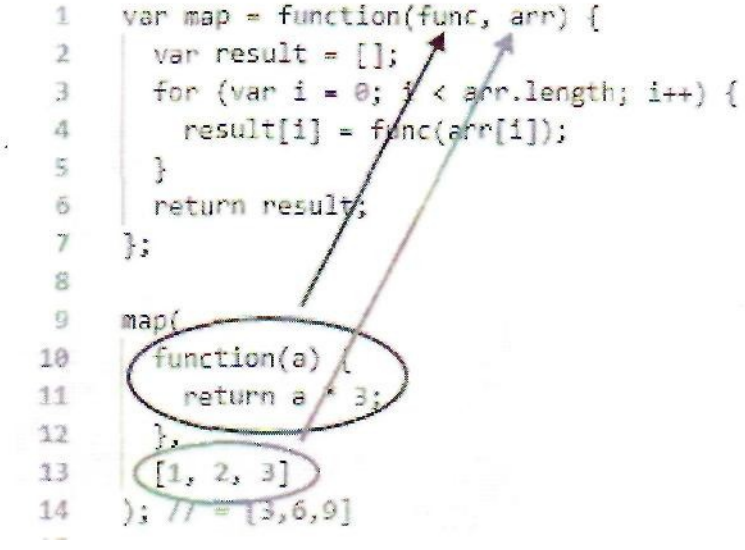
```
var myFunction = new Function("a", "b", "return a * b");
var x = myFunction(4, 3);
```

Виклик функції без () поверне сам код функції:

```
function test() {
  return "test";
}
alert(test); //виклик функції без ()
alert(test());
```

На відмінно від деяких мов програмування (напр. Pascal) JavaScript не має процедур, а лише функції. Різниця між процедурою і функцією полягає в тому що процедура не повертає результат, а функція повертає результат.

Функція у JavaScript завжди повертає результат, результатом може бути будь який тип даних. Повертається результат за допомогою оператора return. Якщо під час виконання функції результат не був повернутий тоді JavaScript автоматично



повертає undefined.

Можна створити анонімну функцію прямо під час виклику функції:

Іменовані функції можуть викликати самі себе - "self-invoking". Self-invoking вирази запускаються автоматично. Іменовані функції будуть виконуватись автоматично, якщо за її виразом слідує - ():

```
(function() {
  var x = "Hello!!"; // I will invoke myself
})();
```

Функція вище насправді є анонімною функцією "self-invoking" (функція без імені).

Функції JavaScript можуть бути використані як значення та вирази.

Також появився у стандарті ES6 спрощений синтаксис анонімної функції - стрілочні функції. Найпростіший синтаксис функції виглядає наступним чином:

```
var f = function() {return 15;};
//аналог
var f = () => 15;
```

Значення зліва від стрілки, які знаходяться в круглих дужках, визначають передані в функцію аргументи. Для одного аргументу дужки не потрібні. Якщо аргументи не передаються, необхідно використовувати круглі дужки. Визначення функції праворуч від стрілки може бути або виразом (наприклад, v + 1), або блоком операторів (інструкціями), укладеними в фігурні дужки ({}):

```
function(v) { return v + 1; }
//аналог
v => v + 1
```

```
var suma = function(x,y) {return x+y;};
//аналог
var suma = (x,y) => x+y;
```

або щоб повернути об'єкт чи тіло функції, яка не містить фігурних дужок, що виступають аргументами у стрілковій функції - необхідно обгорнути у круглі дужки:

```
var f = (s) => ({text:s});

var ob= f('obj');
console.log( ob.text ); // "obj"
```

**Хід роботи:**

1. Виконати індивідуальні завдання з використання інтерактивного середовища програмування в з інтерфейсом командного рядка Native Browser JavaScript з виведенням інформації в консоль інтерфейсу у хмарному сервісі https://repl.it/repls/HandsomeAmusedGuppy :

1.1. Записати функцію "computeAreaOfARectangle", яка повинна визначати визначити площу прямокутника з даних довжини та ширини сторін:

```
var output = computeAreaOfARectangle(4, 8);
console.log(output); // --> 32
```

1.2. Дано радіус кола Записати функцію "computeAreaOfACircle", яка розраховує площу круга.

```
var output = computeAreaOfACircle(4);
console.log(output); // --> 50.26548245743669
```

1.3. Задано число і експонента. Записати функцію "computePower", яка повертає задане число, підняте до заданої експоненти.

```
var output = computePower(2, 3);
console.log(output); // --> 8
```

1.4. Дано число. Напишіть функцію під назвою "computeSquareRoot", яка повертає квадратний корінь даного числа.

```
var output = computeSquareRoot(9);
console.log(output); // --> 3
```

1.5. Дано три слова. Записати функцію під назвою "getLengthOfThreeWords", яка повертає суму букв цих слів.

```
var output = getLengthOfThreeWords('some', 'other', 'words');
console.log(output); // --> 14
```

1.6. Дано два масиви. Записати функцію під назвою "joinArrays", яка повертає новий масив в якому спочатку йдуть елементи першого масиву, а потім - елементи другого.

```
var output = joinArrays([1, 2], [3, 4]);
console.log(output); // --> [1, 2, 3, 4]
```

1.7. Дано об'єкт та ключ. Записати функцію під назвою "getProductOfAllEIementsAtProperty", яка повертає добуток всіх елементів масиву з даним ключем заданого об'єкту.

- якщо масив порожній, функція повинна повертати 0;
- якщо властивість ключа не є масивом, то функція повинна повертати 0;
- якщо властивість ключа не має значення, то функція повинна повертати 0.

```
var obj = {
  key: [1, 2, 3, 4]
};
var output = getProductOfAllElementsAtProperty(obj, 'key');
console.log(output); // --> 24
```

1.8. Дано число. Записати функцію під назвою "sumDigits", яка повертає суму всіх цифр з яких складається число.

```
var output = sumDigits(1148);
console.log(output); // --> 14
```

Якщо число від'ємне, то перша цифра повинна також бути від'ємною:

```
var output = sumDigits(-316);
console.log(output); // --> 4
```

1.9. Дано маси чисел та слів. Записати функцію під назвою "findShortestWordAmongMixedElements", яка повертає найкоротше слово в даному масиві.

- якщо є слова, які повторюються, то функція повинна повертати перше;
- якщо даний масив порожній, він повинен повернути порожній рядок;
- якщо даний масив не містить рядкових значень, то функція повинена повернути порожній рядок.

```
var output = findShortestWordAfflongMixedElsments([4,
'two', 2, "three"]}; console.log(output); // –> "two*
```

1.10. Дано масив чисел та слів. Записати функцію під назвою "findSmallestNumberAmongMixedElements", яка повертає найменше число в даному масиві.

Зуваження:

- якщо даний масив порожній, він повинен повернути порожній рядок;
- якщо даний масив не містить чисел, то функція повинена повернути порожній рядок.

```
var output = findSmallestNumberAmongMixedElements([4, 'lincoln', 9, 'octopus']);
console.log(output); // --> 4
```

1.11. Дано два числа. Записати функцію під назвою "modulo", яка повертає залишок ділення одного числа на інше. Зауваження:

- не можна використовувати вбудований оператор %;
- 0%ANYNUMBER = 0;
- ANYNUMBER%0 = NaN;
- якщо операндом є NaN, то результат - NaN;
- modulo() завжди повертає знак першого номера.

```
var output = modulo(25, 4);
console.log(output); // --> 1
```

1.12. Переверніть (відзеркальте) кожен фрагмент з п символів у рядку, де п - це будь-яке натуральне число більше 1.

```
var input = 'a short example';
var output = flipEveryNChars(input, 5);
console.log(output); // --> ohs axe trelpma
/*
Розбиття цього прикладу по фрагментам:
'a sho' -> 'ohs a'
'rt ex' -> 'xe tr'
'ample' -> 'elpma'

-> 'ohs axe trelpma'
*/
```

1.13. Є рядок парних і непарних чисел. Знайдіть, яке число є єдиним парним чи єдиним непарним числом.

```
detectOutlierValue("2 4 7 8 10");
// => 3 - third number is odd, while the rest of the numbers are even

detectOutlierValue("1 10 1 1");
//=> 2 - second number is even, while the rest of the numbers are odd
```

1.14. Є масив невід'ємних цілих чисел і результуюча сума, знайдіть пару чисел, які при додаванні дають цю суму.

```
var pair = findPairForSum([3, 34, 4, 12, 5, 2], 9);

console.log(pair); // --> [4, 5] // 4 + 5 = 9
```

1.15. Є два рядки. Перевірте, чи є ці рядки дзеркальним відображенням?

```
String 1: 'hello world'
String 2: 'orldhello w'
--> true
```

1.16. Побудувати функцію бінарного пошуку.

При довільному масиві даних, пошук неможливо прискорити – час перебору завжди в лінійній залежності від розміру масиву. Що б ви не намагалися зробити, потрібно завжди перебрати усі елементи один за одним.

Ситуація змінюється, коли ми знаємо, що масив даних певним чином впорядкований. Для даного алгоритму елементи повинні бути сортованими в порядку зростання.

Головна ідея алгоритму полягає у тому, щоб розділити масив на половину і порівняти серединний елемент з шуканим елементом. Якщо даний елемент масиву рівний шуканому, тоді пошук переривається. Якщо він менший від шуканого, тоді усі елементи зліва від даного включно, виключаються з пошуку. А якщо він більший за шуканий елемент, з пошуку виключаються усі елементи з права від даного елементу масиву.

```
var arr = [1, 3, 16, 22, 31, 33, 34]
console.log(search(arr, 31)); // --> 4 - індекс в масиві
```

1.17. Ізограма - це слово, яке не має букв, що повторюються. Реалізуйте функцію, яка визначає, чи є рядок, що містить тільки літери, ізограмою.

1.18. Паліндром - слово, число, набір символів, словосполучення або віршований рядок, що однаково читається в обох напрямках (зліва направо та справа наліво). Реалізуйте функцію, яка визначає, чи є в реченні слова паліндроми.