ЛАБОЛАТОРНА РОБОТА №8

Тема: ‘Ознайомлення з мовою програмування JavaSript. Типи даних. Аргументи функції. Оператори. Масиви’

ХІД РОБОТИ

Завдання 1 (згідно lec_1):

Вивести всі види змінних з їх типами (рядок, об’єкт, масив, рядок, число , ф-ція).

Виконнання (JS Code):

```
// lec_1

// 1
let letString = 'Hellow world!';
console.log( "1: ",letString," - ", typeof letString);

// 2
let letObject = {
    name:"Ernest"
}
console.log( "\n2: ", letObject," - ", typeof letObject);

//3
let letArray = [1,2,3,4,5];
console.log( "\n3: ",letArray, " - ", typeof letArray);

//4
let letNumber = 10;
console.log( "\n4: ",letNumber," - ", typeof letNumber);

//5
function letFunction () {
    return "call!";
}
console.log( "\n5: ",letFunction()," - ", typeof letFunction);
```

Консоль:

```
1:  Hellow world!  -  string

2:  { name: 'Ernest' }  -  object

3:  [ 1, 2, 3, 4, 5 ]  -  object

4:  10  -  number

5:  call!  -  function
```

2.Для рядка і числа змінити значення так, щоб їх типи змінилися :

Виконання (JS Code):

```
//TASK 2
console.log("\n\nTASK 2:\n")
let string = "538";
console.log("String:", string, " - ", typeof string);
let number = 543;
console.log("Number:", number, " - ", typeof number, "\n");


let numberOfString = String(number);
console.log("String (but number in past):", numberOfString," - ",typeof numberOfString);

let stringOfNumber = Number(string);
console.log("Number (but string in past):", stringOfNumber," - ", typeof stringOfNumber);
```

Консоль:

```
TASK 2:

String: 538  -  string
Number: 543  -  number

String (but number in past): 543  -  string
Number (but string in past): 538  -  number
```

3. Порівняти між собою рядок і число з ‘однаковими’ значеннями:

Виконання (JS Code):

```
// TASK 3
console.log("\n\nTASK 3:\n");

let valueOne = "92";

let valueTwo = 92;

if (valueOne == valueTwo) {
    console.log("They`re equals!!");

}

if (valueOne === valueTwo) {
    console.log("They`re not equals!!");
}
```

Консоль:

```
TASK 3:

They`re equals!!
```

4. Простим способом зашифрувати / дешифрувати числову інформацію:

```
// TASK 4 (LAST)
console.log("\n\nTASK 4:\n");
let numberio = 678395;
console.log("Number: ", numberio, "\n");
let encrypted = numberio
  .toString()
  .split("")
  .reverse()
  .join("");
  console.log("Encypted value:", encrypted, "");

  let decrypted = encrypted
  .toString()
  .split("")
  .reverse()
  .join("");
  console.log("Decrypted value:", decrypted);
```

Консоль:

```
TASK 4:

Number:  678395

Encypted value: 593876
Decrypted value: 678395
```

Завдання 2 (Згідно lec_2):

Написати код в консолі браузера/Node.js. Який буде результат виконання?

КОД:

```
var foo = 1;
function bar() {
    if (!foo) {
        var foo = 10;
    }
    alert(foo);
}
bar();
```

Виведе: `1`, foo = 1;

Пояснення: Умова для змінної foo в викликаній ф-ції bar поверне false, так як присвоєне значення змінній foo - це 1, а значить `не true` (!foo) це fasle! Щоб умова спрацювала - потрібно змінній foo присвоїти 0, АЛЕ! Виведеться... 0. Проблема в тому, що ініціалізована змінна в умові (if) існує ЛИШЕ в тілі умови. Простими словами,навіть при звертанні до цієї змінної в ТІЛІ ф-ції (не в умові) всерівно виведе 0 (початкове let foo = 0).

3.Обгрунтувати такий код:

```
var a = 1;
function b() {
    a = 10;
    return;
    function a() {}
}
b();
console.log(a);
```

Вивід: a = 1;

Пояснення:

ініціалізація a, присвоєння одиниці, виклик ф-ції. В тілі ф-ції змінній присвоюємо 10, створюємо ще одну вкладену ф-цію. Так чому ж не виводить 10? Чому змінній a не присвоюється інше значення? Проблема в тонкостях JS, де ф-ція `a` піднімається на верх, тим самим число 10 присвоюється ф-ції (так, це можливо), а не змінній. Змінна не змінюється і залишається зі значенням 1.

Як виправити?

Видалити або перейменувати ф-цію `a`;

Завдання 3 (згідно з lec_3):

1. Навести приклад унарних бінарних, тернарних операторів.

Код (JS):

```
// lec_3
console.log("\n\nLECTURE 3:\n")

// TASK 1
console.log("\n\nTASK 1:\n")

let numb = 10;
numb++;
console.log('\nunary operator: ', numb);

let one = 10;
let two = 15;
let sum = one + two;
console.log('\nbinary operator: ', sum);


let age = 18;

let isGroseAssMan = (age >= 18) ? "realy grose!" : "he is balabol";
console.log('\nternarny operator: ', isGroseAssMan);
```

Консоль:

```
TASK 1:


unary operator:  11

binary operator:  25

ternarny operator:  realy grose!
```

2. Написати програму, використовуючи метод concat() для побудови рядка такого вигляду:

Нехай завжди буде сонце,
Нехай завжди буде небо,
Нехай завжди буде мама,
Нехай завжди буду я.

Код (JS):

```
// TASK 2
console.log("\n\nTASK 2\n");
let str1 = "Прийшла апатія кінця..\n";
let str2 = "влетіли мрії в бездну дна,\n";
let str3 = "я сам забув той сенс буття - \n";
let str4  = "якби...\n";
let str5 = "якби вернутись до життя!\n";

let  resultString = str1.concat( str2, str3,str4,str5);

console.log(resultString);
```

Консоль:

```
TASK 2

Прийшла апатія кінця..
влетіли мрії в бездну дна,
я сам забув той сенс буття -
якби...
якби вернутись до життя!
```

Завдання 4 (Згідно lec_arrays):

Завдання 1: Знайти непреривний підмасив масиву, сума якого є максимальною.

Щоб виконати це завдання, я використав алгоритм Кадане:

Код (JS):

```
// Lec_arrays
console.log("\n\nLECTURE ARRAYS:\n")


//TASK 1
console.log("\n\nTASK 1\n");

function Kokade(arr) {
    let max = arr[0];
    let current = arr[0];

    for (let i = 1; i < arr.length; i++) {
        current = Math.max(arr[i], current + arr[i]);
        max = Math.max(max, current);
    }

    return max;
}
console.log("Result: ", Kokade([-2, 1, -3, 4, -2, 2, 1, -5, 4]));
```

Консоль

```
LECTURE ARRAYS:



TASK 1

Result:  5
```

2.Створити ф-цію сумування двох дуже довгих чисел, представлених строкою.

Код (JS):

```
//TASK 2
console.log("\n\nTASK 2\n");

function addLongNumbers(num1, num2) {
    let len1 = num1.length;
    let len2 = num2.length;
    let carry = 0;
    let result = "";


    for (let i = 0; i < Math.max(len1, len2); i++) {

        let digit1 = i < len1 ? parseInt(num1[len1 - 1 - i]) : 0;
        let digit2 = i < len2 ? parseInt(num2[len2 - 1 - i]) : 0;

        let sum = digit1 + digit2 + carry;
        result = (sum % 10) + result;
        carry = Math.floor(sum / 10);
    }
    if (carry > 0) {
        result = carry + result;
    }

    return result;
}
let num1 = "123456789123456789";
let num2 = "987654321987654321";
console.log(addLongNumbers(num1, num2));
```

Консоль:

```
TASK 2

1111111111111111110
```

Написати метод, який знахлдить ріницю 2 масивів. Причому різниця має враховувати кількість однакових елементів.

Код (JS):

```
console.log("\n\nTASK 3\n");

function arrayDifference(arr1, arr2) {
    let result = [...arr1];
    for (let elem of arr2) {
        const index = result.indexOf(elem);
        if (index !== -1) {
            result.splice(index, 1);
        }
    }

    return result;
}

const array1 = [1, 2, 2, 3, 4];
const array2 = [2, 3, 5];
console.log(arrayDifference(array1, array2));
```

Консоль:

```
TASK 3

[ 1, 2, 4 ]
```

ВИСНОВОК:

Робота з масивами і числами в JavaScript часто вимагає застосування нестандартних підходів, особливо коли йдеться про обробку великих даних або складні операції з елементами масивів. Зокрема, для завдання з різницею двох масивів важливо враховувати кількість однакових елементів, що дозволяє правильно виконати операцію, не втрачаючи важливі дані. Використання методів, як-от indexOf і splice, дозволяє точково видаляти елементи, зберігаючи потрібні входження. Для обчислення суми або різниці великих чисел, представлених у вигляді рядків, необхідно реалізовувати власні функції, оскільки звичайні операції JavaScript не підтримують арифметику з такими значеннями через обмеження в розмірі чисел. Подібні задачі демонструють важливість алгоритмічного підходу, який оптимізує використання

пам'яті та обчислень. Робота з масивами також показує, як важливо знати структуру даних, щоб мінімізувати час обробки великих обсягів інформації. У підсумку, вміння маніпулювати масивами та великими числами є важливою навичкою для ефективного програмування.