ПРАКТИЧНА РОБОТА 4

Тема: Управління виконанням програми JavaScript.

Мета: Навчитися управляти послідовністю дій, під час виконання сценарію у JavaScript

Питання для повторення:

1. Оператори JavaScript.
2. Пріоритет операторів JavaScript.
3. Умовний оператор **if** ... **else.**
4. Оператор вибору **switch.**
5. Оператори циклу **for, while, do...while, break** та **continue.**
6. Оператор ітерації for... in.
7. Оператор вказування об'єкту with;
8. Структурування програмного потоку за допомогою обробки виключень,

Завдання:

Застосувати логічну організацію управління в програмі з послідовним виконанням скрипта для передачі програмного потоку в іншу частину програми.

Вивести інформацію основних різновидів структур JavaScript у HTML-документі або консолі браузера HTML.

Теоретичні відомості:

1. Передача управління в програмі на основі структури вибору

Структура вибору використовується для вказування альтернативних варіантів виконання програми шляхом створення точки її розгалуження. В JavaScript доступні чотири структури вибору:

1) структура єдиного вибору (if):

```
if (3 < 4) {
    console.log("yes");
};
```

2) структура подвійного вибору (if ... else): Послідовність Фібоначчі $(F_n = F_{n-1}, + F_{n-2}, F_0 = 0, F_1 = l)$

```
function fib(n) {
  if (n < 2) {
    return n;
  } else {
    return fib(n - 1) + fib(n - 2);
  }
}
console.log("Fib(9) = " + fib(9)); // Fib(9) = 34
```

3) вбудований тернарний оператор (test ? exprl : expr2) "if then else" - якщо test еквівалентно true, то виконується вираз expr1, у протилежному випадку exprг2:

```
console.log((4 > 5) ? "yes":"no"); // no
```

або більш практичний приклад

```
var stop = false, age = 23;
age > 18 ? (
    alert("Вітаємо Вас на нашій сторінці!"),
    document.location.assign("http://edition.cnn.com/").reload(true)
) : (
    stop = true,
    alert("Вибачте, доступ обмежено!")
);
```

4) структура множинного вибору (switch).

Розглянемо код:

```
if (variable == "value1") action1();
else if (variable == "value2") action2();
else if (variable == "value3") action3();
else defaultAction();
```

Цей код можна переписати у вигляді структури множинного вибору:

```
switch (variable) {
case "value1":
    action1();
    break;
// без "break" буде продовжувати виконуватися код нижче
// без перевірки логічних умов
case "value2":
    action2();
    break;
case "value3":
    action3();
    break;
default:
    defaultAction();
}
```

switch використовує умову === 2.

2. Передача управління в програмі на основі структури повторення

Структура повторення використовується для вказування необхідності повторювати дію, поки певна умова залишається дійсною. Є декілька способів повторного виконання оператора або блоку операторів. Повторюване виконання називається циклом або ітерацією. Інакше кажучи, ітерація - це одноразове виконання циклу.

В JavaScript доступні чотири структури повторення:

1) перевірка виразу на початку циклу (while);
2) перевірка умови в кінці циклу (do / while);
3) цикл під керуванням лічильника (for);

4) виконання для кожної властивості об'єкта (for / in).

Перевірка виразу на початку циклу:

```
let x = 0;
do {
  console.log(x);
  x++;
} while (x !== 5);
// prints 0 to 4
```

Перевірка умови в кінці циклу:

```
let x = 0;
do {
  console.log(x);
  x++;
} while (x !== 5);
// prints 0 to 4
```

Цикл під керуванням лічильника: оператор for вказує змінну лічильника, умову перевірки і дію, яка оновлюючe лічильник. Умова перевіряється перед кожною ітерацією циклу. У разі успішної перевірки виконується код всерединіциклу. Якщо перевірка не пройдена успішно, код всередині циклу не виконується, а програма продовжує роботу з першого рядка, що йде наступною безпосередньо після циклу. Після виконання циклу змінна лічильника оновлюється перед початком наступної ітерації. Якщо умова циклу не виконується, цикл не запускається. Якщо умова циклу виконується завжди, утворюється нескінченний цикл.

```
for (statement_1; statement_2; statement_3) {
    //блок коду, який буде виконаний
}
```

- statement_l виконується ініціалізація та оголошується змінні;
- statement_2 визначає умову для запуску циклу;
- statement_3 виконується кожного разу після ітерації циклу (блоку коду).

```
for (let i = 0; i < 9; i++) {
  console.log(i);
}
// 0, 1, 2, ... , 8
```

continue - вихід із поточної ітерації в циклі:

```
for (let i = 0; i <= 5; i++) {
```

```
for (let i = 0; i < 5; i++) {
  console.log(i);
  if (i === 3) {
    break;
  }
} // prints 0 to 3
```

Виконання для кожної властивості об'єкта.

Цикл for...in - призначений для проходження по всіх властивостях об'єкту або елементах масиву. Лічильник циклу for…in являє собою не число, а рядок. Він містить ім'я поточної властивості або індекс поточного елемента масиву.

Цикл for... in проходить тільки по перелічуваних властивостях, за довільною послідовністю. Цикл можна зупинити за допомогою оператора break або перейти до наступного проходу циклу за допомогою оператора continue.

```
var person = {fname:"John", lname:"Doe", age:25};

var text = "";
var x;
for (x in person) {
    text += person[x] + " ";
}
console.log(text); //"John Doe 25 "
```

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. Маємо три слова. Записати функцію "findMinLengthOfThreeWords", яка повинна визначати мінімальну довжину з цих слів:

```
var output = findMinLengthOfThreeWords('a', 'be', 'see');
console.log(output); // --> 1
```

1.2. Дано масив з числами. Записати функцію "filterOddElements", яка повертає масив де кожний елемент є непарним числом заданого масиву.

```
function filterOddElements(arr) {
  // your code here
}
var output = filterOddElements([1, 2, 3, 4, 5]);
console.log(output); // --> [1, 3, 5]
```

1.3. Дано масив. Записати функцію "getLengthOfShortestElement", яка повертає довжину самого короткого слова, що міститься у даному масиві.

```
function getLengthOfShortestElement(arr) {
  // your code here
}
var output = getLengthOfShortestElement(["one", "two", "three"]);
console.log(output); // --> 3
```

1.4. Дано масив з масивів. Записати функцію "joinArrayOfArrays", яка повертає новий масив, що містить всі елементи вкладених масивів даного масиву. Для розв'язання задачі необхідно застосувати метод "concat".

```
function joinArrayOfArrays(arr) {
  // your code here
}
var output = joinArrayOfArrays([[1, 4], [true, false], ["x", "y"]]);
console.log(output); // --> [1, 4, true, false, 'x', 'y']
```

1.5. Дано масив з різних елементів. Записати функцію "findSmallestNumberAmongMixedElements", яка повертає саме маленьке число в даному масиві. Зауваження:

1. Якщо масив не містить елементів, функція повинна повернути 0.
2. Якщо масив не містить чисел, функція повинна повернути 0.

```
function findSmallestNumberAmongMixedElements(arr) {
  // your code here
}
var output = findSmallestNumberAmongMixedElements([4, "lincoln", 9, "octopus"]
console.log(output); // --> 4
```

1.6. Дано число. Записати функцію "computeSummationToN", яка повертає суму послідовності чисел від нуля до даного числа. Зауваження: Якщо n = 4, то слід обчислити суму 1 + 2 + 3 + 4, а повернути 10.

```
function computeSummationToN(n) {
  // your code here
}

var output = computeSummationToN(6);
console.log(output); // --> 21
```

1.7. Дано рейтинг студента записаний числом. Записати функцію "convertScoreToGrade", яка повертає рейтинг у форматі шкали ECTS. Зауваження:

1. (100 - 90) --> 'A'
2. (89 - 82) --> 'B'
3. (81 - 75) --> 'C'
4. (74 - 69) --> 'D'
5. (68 - 60) --> 'E'
6. (59 - 35) --> 'Fx'
7. (34 - 0) --> 'F'

Якщо маємо значення більше 100 та менше 0, то функція повинна повернути рядкове значення 'INVALID SCORE'.

```
function convertScoreToGrade(score) {
  // your code here
}

var output = convertScoreToGrade(91);
console.log(output); // --> 'A'
```

1.8. Дано три слова. Записати функцію "getLongestOfThreeWords", яка повертає найдовше слово з даних слів. Зауваження:
Якщо маємо послідовність однакових слів, то функція повинна повернути перше слово в списку аргументів.

```
function getLongestOfThreeWords(word1, word2, word3) {
  // your code here
}

var output = getLongestOfThreeWords('these', 'three', 'words');
console.log(output); // --> 'these'
```

1.9. Записати функцію "multiply", яка повертає добуток цих чисел. Зауваження: ви не можете використовувати оператор *.

```
function multiply(num1, num2){
  // your code here
}

var output = multiply(4, 7);
console.log(output); // --> 28
```

1.10. Дано два цілих числа. Записати функцію "computeSumBetween", яка повертає суму чисел, що знаходяться між цими даними цілими числами. Зауваження:

- сума між числами 1 і 4 є 1+2+3=6;
- якщо число 2 менше числа 1, то функція повинна повернути 0.

```
function multiplyBetween(num1, num2) {
  // your code here
}

var output = computeSumBetween(2, 5);
console.log(output); // --> 9
```