## ====================== ТЕОРІЯ ================

## Підпрограми

Часто необхідно повторювати одну і ту ж дію в багато раз. Наприклад, красиво вивести повідомлення привітання, при виході відвідувача на сайт. Щоб не повторювати один і той же код в багатьох місцях, існують підпрограми. В JS всі підпрограми є функціями. Існують вбудовані функції, з якими ви вже зустрічались - це alert, prompt, confirm та ін. А можна описати і свої.

Синтаксис оголошення функції:

function назва(параметр1, параметр2, … , параметрN ) {

тіло функції;

return результат;

}

Спочатку йде ключове слово function, після нього ім'я функції, потім список параметрів в дужках (він може бути порожній) і тіло функції – послідовність команд, що виконується при її виклику. Оголошена функція доступна по імені. Головна мета створення функцій - уникання дублювання коду. Ще одна перевага використання функцій полягає в тому, що якщо знадобиться внести зміни - достатньо змінити код в одному місці: у функції, яка його виконує.

*Приклад.* Створити підпрограму,що виводить привітання.

function showMessage () {

   alert ('Привіт всім присутнім!');

}

showMessage ();

showMessage ();

Цей скрипт виводить повідомлення два рази. Якщо знадобиться поміняти повідомлення або спосіб його виведення - достатньо змінити його в функції.

Функція може містити локальні змінні, оголошені через var. Такі змінні доступні тільки всередині функції. Блоки if, else, switch, for, while, do … while та інші не впливають на область видимості змінних. При оголошенні змінної в таких блоках, вона все одно буде видна у всій функції. Неважливо, де саме в функції і скільки разів оголошується змінна. Будь-яке оголошення спрацьовує один раз і поширюється на всю функцію. Функція може звернутися до зовнішньої змінної. При цьому, змінна буде доступна й зовні функції. Якщо всередині функції, є оголошена своя локальна змінна з тим же іменем, то всі звернення використовують її, і зовнішня змінна залишиться незмінною. Змінні, оголошені на рівні всього скрипта, називають глобальними змінними. Робити глобальними слід тільки ті змінні, які дійсно мають загальне значення для проекту.

var a,b,c;

a=b=c=1;

f(3); // Виклик функції з параметром рівним 3. Виклик можливий до опису.

function f(a){ // опис функції

var x=5; // локальні змінні

a++;b--;

alert(a); // 4 – при виклику функції a == 3, a++ == 4

alert(b); // NaN -- Локальній змінній не задано значення, тому вона undefined.

alert(c); // 1 -- глобальна змінна

alert(x); // 5

var b;

};

f(3);

alert(a); // 1 – зміна значення параметру функції ніяк не впливає на значення глобальної змінної

alert(b); // 1 -- зміна значення локальної змінної ніяк не впливає на значення глобальної змінної

alert(c); // 1

alert(x); // помилка, доступ до х є тільки в межах підпрограми.

f(c); // ніколи не виконається

Параметри копіюються в локальні змінні функції. Функцію можна викликати з будь-якою кількістю аргументів. Якщо параметр не переданий при виклику - він вважається рівним undefined. Для визначення значення за замовчуванням, тобто, такого, яке використовується, якщо аргумент не вказано, використовується два способи:

* 1. Можна перевірити, чи рівний аргумент undefined, і якщо так - то записати в нього значення за замовчуванням.
  2. Використовувати оператор | |:

function showMessage (from, text) {

   if (text === undefined) {

     text = '\*Текст відсутній\*';

  };

from = from || 'noname';

alert (from + ": " + text);

}

showMessage ("Вася", "Привіт!"); // Маша: Привіт!

showMessage ("Вася"); // Вася: \*Текст відсутній\*

showMessage (); // noname: \*Текст відсутній\*

У деяких мовах, програміст може створити дві функції з однаковим ім'ям, але різним набором аргументів, а при виклику інтерпретатор сам вибере потрібну. Це називається «поліморфізмом функцій» або «перевантаженням функцій». В JS нічого подібного немає.

Може бути тільки одна функція з таким іменем, яка викликається з будь-якими аргументами. А вже всередині вона може подивитися, з чим викликана і по-різному відпрацювати.

Як отримати значення аргументів, яких немає в списку параметрів?

Доступ до них здійснюється через псевдо-масив arguments. Він містить список аргументів за номерами: arguments [0], arguments [1] ..., а також властивість length.

function sayHi() {

for (var i = 0; i < arguments.length; i++) {

alert( "Привіт, " + arguments[i] );

}

}

sayHi("Вінні", "Пятачок"); // 'Привіт, Вінні', 'Привіт, Пятачок'

Всі параметри знаходяться в arguments, навіть якщо вони є в списку формальних параметрів. arguments - це не масив Array, хоч і схожий на нього. Це звичайний асоціативний масив, ключі якого є цілими числами і є властивість length. Ніяких особливих методів у нього немає, і методи масивів він теж не підтримує.

function sayHi () {

var a = arguments.shift (); // помилка! немає такого методу!

}

Функція може повернути результат, який буде передано в основну прграму її код. Для повернення значення використовується оператор return. Він може знаходитися в будь-якому місці функції. Як тільки виконання до нього доходить управління - функція завершується і значення передається в основну програму.

Наприклад Створити функцію, для обчислення дискримінанту квадратного рівняння за формулою

function calcD (a, b, c) {

    return b \* b - 4 \* a \* c;

};

var test = calcD (-4, 2, 1);

alert (test); // 20

У випадку, коли функція не повернула значення або return був без аргументів, вважається що вона повернула undefined.

function checkAge (age) {

   if (age> 18) {

     return true;

   }

   // ...

  return confirm ('Батьки дозволили?');

}

**Передача функції у функцію**

Задача. Створити функцію, яка могла би :

1)усі елементи множити на 2

2)усі елементи ділити на 7

3)усі елементи підносити до квадрату

4)усі елементи підносити до кубу

//------------- Описуємо узагальнену функцію

function Perebor(arr,action) {

for (var i = 0; i < arr.length; i++) {

arr[i] = action(arr[i]);

}

}

//------------- Описуємо окремі функції, які відображають дії над елементами

function f2(value) {

return value \* 2;

}

function f7(value) {

return value / 7;

}

function fsq(value) {

return value \* value;

}

function fcube(value) {

return Math.pow( value , 3);

}

function f5(value) {

return value \* 5;

}

//------------- Під час виклику передаємо одну із функцій

Perebor(b, f5);

**=================== ПРИКЛАДИ=====================**

//Знайти розмір заробітної плати (ставка + надбавка (25%) - податок (20%))

function getFinalSalary(salary, tax, extra) {

var s = salary \* (1 + extra / 100) \* (1 - tax/ 100);

return s;

}

var salary = 69000;

var tax = 20;

var extra = 25

var finalSalary = getFinalSalary(salary, tax, extra);

alert(“Final salary: ”+ finalSalary);

### ///=====================

//Згенерувати число в межах від min до max

function getRandomValue(min,max) {

return min+Math.floor(Math.random()\*(max-min+1));

}

//Функція повертає результат двох кидків гральних костей

function getScore() {

//var score1 = getRandomValue(1, 6);

//var score2 = getRandomValue(1, 6);

//return score1 + score2;

//--------- Варіант 2

return getRandomValue(1, 6) + getRandomValue(1, 6);

}

var player1Score = getScore();

var player2Score = getScore();

document.write("Score 1 ="+player1Score+", Score 2 ="+player2Score);

### Задача 1. Обчислити значення виразу

### 

function getMax3(a,b,c) {

var max = a;

if (b>max) {

max = b;

}

if (c>max) {

max = c;

}

return max;

}

//-----------------------------

var a = parseFloat(prompt("a=", "0"));

var b = parseFloat(prompt("b=", "0"));

var c = parseFloat(prompt("c=", "0"));

var res = getMax3(a, -b, 3) \* getMax3(a, 2 \* b, c) - getMax3(-a, b, 7);

document.write("Result=" + res);

Задача. Вивести товари, які може купити користувач

//money-кількість грошей користувача,prductsPrices - ціни товарів

function printPrices(money,productsPrices) {

for (var i = 0; i < productsPrices.length; i++) {

if (money>=productsPrices[i] ) {

document.write(productsPrices[i]+"<br>");

}

}

}

//---------------------------

var prices = [123, 34, 890, 234, 210, 78];

var money = 200;

printPrices(money, prices);

Задача 2.

//Функція, яка повертає масив, у якому рандомно розміщено зайців

function getRabbitsArray(cellCount,rabbitsCount) {

var table = new Array(cellCount);

//var table = [];

//Спочатку усі клітки порожні

for (var i = 0; i < cellCount; i++) {

table[i] = 0; //0 значить, що клітка пуста

}

//Випадковим чином розміщуємо зайців

for (var i = 0; i < rabbitsCount; i++) {

var position = Math.floor(Math.random() \* cellCount);

table[position]++;

}

return table;

}

//-----------------------------------------

function play(bulletCount,rabbitsCount,tableWithRabbits) {

while(bulletCount>0 && rabbitsCount>0){

var shootPosition = parseInt(prompt("Shoot position =", '0'));

if (tableWithRabbits[shootPosition]>0) {

alert("Ok. Rabbits: " + tableWithRabbits[shootPosition]);

rabbitsCount -= tableWithRabbits[shootPosition];

tableWithRabbits[shootPosition] = 0;

}

else {

alert("OOOOOOO");

}

}

return rabbitsCount == 0;

}

//-----------------------------------

var tableWithRabbits = getRabbitsArray(10, 5);

if (play(6,5,tableWithRabbits)) {

alert("You won!!!");

}

else {

alert("You loose!");

}

### //-----------------------------------------------------

<script>

//1.Функція, яка генерує випдкове число від мінмальног до макс.

function getRandomValue(min,max) {

return min + Math.floor(Math.random()\*(max-min+1));

}

//2.Функція, яка генерує масив з випадкових чисел від мін.до макс.

function getRandomArray(arrayLength, min,max) {

var array = new Array(arrayLength);

for (var i = 0; i < array.length; i++) {

array[i] = getRandomValue(min,max);

}

return array;

}

//3.Функція, яка перевіряє скільки є числе більше за задане

function getGreaterThanCount(array,valueForComparison) {

var count = 0;

for (var i = 0; i < array.length; i++) {

if (array[i]>valueForComparison) {

count++;

}

}

return count;

}

//4.Функція для виведення масиву

function printArray(array) {

document.write("<table border='2px'>");

document.write("<tr>");

for (var i = 0; i < array.length; i++) {

document.write("<td>"+array[i]+"</td>");

}

document.write("</tr>");

document.write("</table>");

}

//-------------------------------

var arr = getRandomArray(30, 0, 20);

var count = getGreaterThanCount(arr, 10);

printArray(arr);

document.write("Count="+count);

</script>

### Задача. Створити функцію, яка би могла знаходити суму для 2, 3, 4 і більше елементів.

function sum() {

var s = 0;

for (var i = 0; i < arguments.length; i++) {

s += arguments[i];

}

return s;

}

var r = sum(2, 34, 23, 45);

alert(r);

### Задача 1. Обчислити значення виразу

### 

Задача. Вивести товари, які може купити користувач

### Індивідуальні завдання

1. Задані координати вершин трикутника. Знайти його периметр та площу, за допомогою підпрограм.
2. Задані координати 3 точок. Визначити чи лежать вони на одній прямій, за допомогою підпрограми.
3. Задано координати вершин трикутника. Визначити чи є він прямокутним, за допомогою підпрограми. Якщо так – вивести довжину гіпотенузи.
4. Задані координати вершин чотирикутника в порядку обходу за годинниковою стрілкою. Визначити чи він є прямокутником, за допомогою підпрограми. Якщо так знайти його площу.
5. Задані координати вершин чотирикутника в порядку обходу за годинниковою стрілкою Знайти координати точки перетину його діагоналей, за допомогою підпрограми.
6. Задані координати 3 точок. Визначити площу трикутника, за допомогою підпрограми.
7. Задані координати 3 точок. Визначити радіус описаного кола, за допомогою підпрограми.
8. Задані координати точки на поверхні і центра кулі. Знайти її об’єм і площу поверхні, за допомогою підпрограм.
9. Створити підпрограми для знаходження n-го члена та суми перших n членів арифметичної прогресії, за відомими першим членом та різницею
10. Створити підпрограми для знаходження n-го члена та суми суми перших n членів геометричної прогресії, за відомими першим членом та знаменником
11. Створити підпрограму для знаходження суми нескінченної спадної геометричної прогресії, за відомими першим членом та знаменником. Перевірку чи є задана геометрична прогресія спадною оформити як окрему підпрограму.
12. Розв’язати квадратне рівняння, задане коефіцієнтами, за допомогою підпрограми. Знаходження дискримінанта оформити як окрему підпрограму.
13. Відома відсоткова ставка по депозиту. Скласти підпрограму для визначення суми, що отримає користувач через n років, в залежності від суми вкладу. Відсотки нараховуються щомісяця на всю суму на рахунку, в тому числі і на раніше нараховані відсотки.