Функції короткий довідник

|  |  |
| --- | --- |
| Загальний формат опису | Приклад |
| **СПОСОБИ ОПИСУ** |  |
| **Function Declaration**  (через параметри передаємо у функцію величини, які необхідні для роботи функції)  **function** назва(параметр1, параметр2, … , параметрN ) {  тіло функції;  **return** результат;  }  Функція будується на першому етапі роботи інтерпретатора. Може бути використана і одразу | function sum(a, b) {  return a + b;  }  let s = sum(23, 9); |
| **Function Expression**  **var змінна = function** (параметр1, параметр2, … , параметрN ) {  тіло функції;  **return** результат;  }  Функція будується і присвоюється у змінну на другому етапі (виконання операторів) | var sum = function (a, b) {  return a + b;  }  let a = sum(23, 9); |
| **Лямбда вирази**  //------------ один параметр, один операнд в тілі функції (результат знаходиться одразу)  параметр => результат;  -------------------------------------------------------------------------------------------------------  //------------- багато параметрів, один операнд в тілі функції  (параметр1, параметр2, …, параметрN) => результат;  --------------------------------------------------------------------------------------------------------  //більше одного параметра у тілі функції  (параметр1, параметр2, …, параметрN) => {return результат;}  //Без параметрів  ( ) => операнд | var f = x => x \* x;  alert(f(5)); //25  ------------------------  var sum = (a, b) => a + b;  alert(sum(3,5)); //8  ------------------------  var sum = (a, b)=> {  return a + b;  }  ---------------  "use strict"  var area = (a, b, c) => {  var p = (a + b + c) / 2;  return Math.sqrt(p \* (p - a) \* (p - b) \* (p - c));  }  alert(area(3,5,4)); |
| Використання return без значення-результату дозволяє перервати роботу функції (по аналогії з break у циклах) | //До першого нуля елементи масиву множити на 2  function func(arr) {  for (var i = 0; i < arr.length; i++) {  if (arr[i] == 0) return; //Якщо 0 то припинити обробку  arr[i] \*= 2;  }  } |
| При використанні способу опису Function Declaration і режиму “use strict” функцію, яка описана в середині якогось блоку, можна використовувати тільки в середині цього ж блоку!!! | 'use strict';  if (true) {  sayHi(); // ок  function sayHi() {  alert("Привет!");  }  }  sayHi(); // Error |
| Під час виклику функції значення копіюються у локальні змінні - параметри | var a = 7;  function func(c) { // для с створено нову пам’ять  c++;  alert("c="+c); // c=8  }  func(a); //значення а скопіюється у с  alert("a="+a); // a=7 (значення не змінилося) |
| Якщо функція не повертає результат через return або ж використовується return без значення-результату, то її значення – undefined | function **func**(a) {  a++; // жодного сенсу  }  alert(func(2)); //undefined |
| Описані у функції змінні є локальними стосовно даної функції ; | function func() {  var a = 9;  }  alert(a); //a is not defined |
| Допустиме (але небажане, окрім випадків замикання) використання зовнішніх (глобальних, описаних поза функцією) змінних | var a = 7;  function func() {  a = 9; //**зверання до зовнішньої змінної**  alert(a);  }  func(); |
| ------------------------------------------- |  |
| **Задання значень формальних параметрів за замовчуванням** |  |
| Старий стиль (перевірка на undefined) | function func(a, b) {  if (a === undefined)  a = 10;  b = b || 7; //Обережно з значенням 0!!!  return a + b;  }  var s=func(); //значення a=10, b=7  alert("s="+s); // s=17  s = func(5); //значення a=5, b=7  alert("s=" + s); // s=12  s = func(5,4); //значення a=5, b=4  alert("s=" + s); // s=9 |
| Новий стиль (вказуємо при описі формальних параметрів)  **function** назва(параметр1=**поч.знач.1**, параметр2=**поч.знач.2**, … , параметрN=**поч.знач.N** ) {  тіло функції;  **return** результат;  } | "use strict"  function func(**a=10, b=7**) {  return a + b;  }  var s=func(); //значення a=10, b=7  alert("s="+s); // s=17  s = func(5); //значення a=5, b=7  alert("s=" + s); // s=12  s = func(5,4); //значення a=5, b=4  alert("s=" + s); // s=9 |
| Передача довільної кількості параметрів (усі значення, які передаємо у функцію заносяться у псевдо-масив **arguments**) | // Функцію знаходження довільної кількості чисел ------  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);  //--------------------------------------------------  // Знайти розмір заробітної плати (ставка + премія - податок)  function getFinalSalary() {  //arguments[0] - ставка, arguments[1] - премія, arguments[2] - податок  return (arguments[0] + arguments[1]) \* (1 - arguments[2])  }  // Знайти розмір заробітної плати (ставка + премія - податок)  // (значення упаковуємо в об’єкт)  function getFinalSalary(data) {  return (data.salary + data.extra) \* (1 - data.tax/100);  }  alert(getFinalSalary({ salary: 9000, extra: 1000, tax: 10 })); |
| Оператор spread «…»  Дозволяє деякі значення передавати в іменовані параметри, а усі інші у змінну-масив  **function** назва(параметр1, параметр2, … , параметрN,  . . . акумулючий масив ) {  тіло функції;  **return** результат;  }  Може бути тільки один і обов'язково повинен знаходитись у кінці | // Визначити скільки елементів належать заданому діапазону  function getCount(min,max,**...numbers**) {  var count = 0;  for (var i = 0; i < numbers.length; i++) {  if (numbers[i] >= min && numbers[i]<=max) {  count++;  }  }  return count;  }  alert(getCount ( 1, 10, 3, 21, 45, 3, 90, 7));  //min=1, max=10, numbers=[3, 21, 45, 3, 90, 7] |
| Деструктуризація масиву  Якщо при описі функції вказано параметри, а значення, які необхідно передати у функцію знаходяться у масиві, то ми можемо розбити масив на окремі складові з використанням синтаксису  **функція ( . . . масив)** | "use strict"  function f(a, b, c) {  return a + b + c;  }  var m = [2, 1, 5];  alert(**f(...m)**); //a=2, b=1, c=5 |
| Деструктуризація об’єкту | 'use strict';  let options = {  title: "Меню",  width: 100,  height: 200  };  function showMenu({title, width, height}) {  alert(title + ' ' + width + ' ' + height); // Меню 100 200  }  showMenu(options); |
| Повертаємо декілька значень з функції  За допомогою масиву  За допомогою об’єкта | function getData(a, b) {  return [a + b , a \* b]  }  let results = getData(2, 3);  alert("Sum=" + results[0] + " Prod=" + results[1]);  'use strict';  function getData(a, b) {  return { sum:a + b, prod: a \* b };  }  let results = getData(2, 3);  alert("Sum=" + results.sum + " Prod=" + results.prod); |