Довідник по JS

## Основи синтаксису

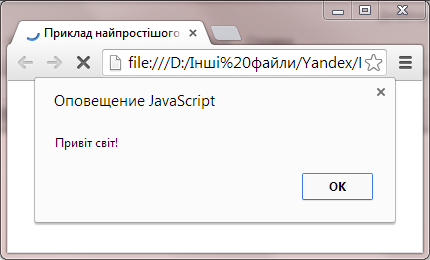
***Підключення (додавання) скриптів***

***1) безпосереднє включення в текст сторінки.***

Для цього оператори записують безпосередньо між тегами

|  |  |
| --- | --- |
| ***Загальна форма*** | ***Приклад*** |
| <scrip type="text/javascript">скрипт</script> | <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8">  <title>Приклад найпростішого скрипта</title>  **<script >**  **alert("Привіт світ!")**  **</script>**  </head>  <body>  Скрипт виконано.  </body>  </html> |

Для його запуску необхідно створити html сторінку, що містить наступний текст і відкрити за допомогою браузера.



***2) підключення окремих файлів з скриптами (мають розлишення \*.js)***

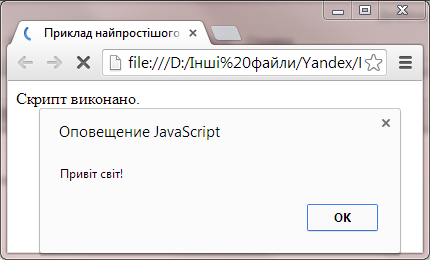
|  |  |
| --- | --- |
| Загальна форма | <script type="text/javascript" src="шлях/ім'я.js"> </script> |
| Приклад | ----------- Файл Hello.js---------  alert("Привіт світ!");  ----------- Сторінка HTML ---------  <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8">  <title>Приклад найпростішого скрипта</title>  **<script src="Hello world.js"> </script>**  </head>  <body>  Скрипт виконано.  </body>  </html> |

<body>

Скрипт виконано.

<script type="text/javascript" src="…../Hello world.js"></script>

</body>



***3)порядок виконання скриптів***

<script **async** src="шлях до скрипта">

<script src="1.js" ></script>

<script src="2.js" ></script>

**Типи даних**

У JS підтримуються :

* прості типи даних (відповідають скалярним значенням: числам, логічним значенням та ін.);
* складні типи даних посилання (об’єкти, функції).

|  |  |
| --- | --- |
| number | як цілі так і дробові, мають тип Number |
| string | текстові дані (як окремі символи, так і рядки) |
| boolean | логічні значення начення ( true (істина) і false (неправда)) |
| undefined | спеціальне значення, яке використовується як значення змінної, якій не присвоєно значення |
| object | тип даних посилання, використовується при описі об’єктів  (null - спеціальне значення, яке використовується для позначення «порожньої адреси») |

**Коментарі**

Приклад.

// Це однорядковий коментар \_

Приrлад.

**/\*** Це

багаторядковий

коментар **\*/**

*Літерали*

*Літерал* – це явно вказане значення декого типу. Розрізняють такі типи літералів:

|  |  |  |
| --- | --- | --- |
| *тип літерала* | *опис* | *приклади* |
| масив |  | *var a=[]*  *var b=[23,12,89]* |
| цілого типу | у десятковій системі числення – звичне нам ціле число | 125, -89, 108; |
| дійсного типу | у форматі з фіксованою крапкою – у записі числа є крапка, що розділяє цілу і дробову частити | 25.69, 2.0, 145.058 |
| у форматі з плаваючою крапкою – у записі використано символ «е» або «Е», що розділяє мантису від порядку | 3.5е5, 3е12, 678е2 |
| символьний | заданий у явному вигляді рядок у одинарних або подвійних лапках | var a=”Hello”  var b = “Peace”  var c=”” |
| прості ескейп-послідовності – службові символи починаються з символу «\» | \n -перехід на новий рядок,  \t -горизонтальна табуляція,  \’ - апостороф, і т. д. |
|  | ` … текст … ${ значення } … текст ` | // Інтерполяція рядків  var name = "Бобби", time ="сегодня";  `Привет ${name}, как ты ${time}?`  “Привет” +name+”, как ты”+time |
| логічний | може приймати два значення | true,  false |
| літерал об’єкта |  | var a={}  var b={name=”Ivan”} |
| порожньої адреси (нульовий літерал) | використовується у випадку, коли покажчик не містить жодної адреси | null |

| **Специальные символы в JavaScript** | |
| --- | --- |
| **Символ** | **Значение** |
| \b | Возврат (Backspace) |
| \f | Перевод или прогон страницы (Form feed) |
| \n | Перевод строки (New line) |
| \r | Возврат каретки (Carriage return) |
| \t | Табуляция (Tab) |
| \v | Вертикальная табуляция (Vertical tab) |
| \' | Апостроф или одинарная кавычка |
| \" | Двойная кавычка |
| \\ | Обратная косая черта (Backslash) |
| \*XXX* | Символ в кодировке Latin-1, представленный тремя восьмеричными числами *XXX* от 0 до 377. Например, \251 (символ ©). |
| \x*XX* | Символ в кодировке Latin-1, представленный двумя шестнадцатеричными числами *XX* от 00 до FF. Например, \xA9 (символ ©). |
| \u*XXXX* | Символ в Unicode, представленный четырьмя шестнадцатеричными числами *XXXX*. Например, \u00A9 (символ ©). |
| \u*{XXXXX}* | Символ в UTF-32BE. Например, \u{2F804} обозначает то же, что обычная запись \uD87E\uDC04. |

## Опис змінних

Опис може бути здійснено декількома способами:

* як глобальна змінна (як властивість window);
* за допомогою var;
* за допомогою let.

|  |  |
| --- | --- |
| Загальна форма | Приклад |
| змінна = значення; | a = 90; // рівносильно window.a=90 |

*Опис змінних за допомогою var*

|  |  |
| --- | --- |
| Загальна форма | Приклад |
| var змінна; | var a;  var b,c; |

При оголошенні також можна задавати початкове значення змінної.

|  |  |
| --- | --- |
| Загальна форма | Приклад |
| var змінна = початкове значення; | var a=7;  var b= “Hello”;  a=”Hi” |

Правильні ідентифікатори: MyID, identifier, x, $, abc123, my\_id, \_true.

Неправильні:

my id //містить пробіл

Ok! // містить недозволений знак оклику !

12th //починається з цифри

100 //починається з цифри

for //зарезервоване слово.

***Локальні змінні***

{

… блок …

}

Для цього необхідно описати ці змінні з використанням let

|  |  |
| --- | --- |
| Загальна форма | Приклад |
| {  let змінна;  } | {  let a;  a=23; 🡨 **Доступ можливий**  let b,c;  ...  a=a+5; 🡨 **Доступ можливий**  }  a=9; 🡨 **ПОМИЛКА! Змінна недоступна** |

## Опис констант

|  |  |
| --- | --- |
| Загальна форма | Приклад |
| const КОНСТАНТА = значення; | const RATE=15; |

## Введення та виведення даних

***Виведення даних***

***Виведення даних за допомогою alert***

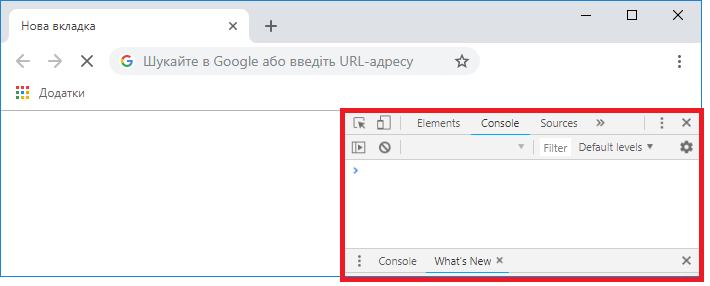
|  |  |  |
| --- | --- | --- |
| Загальна форма | Приклад | На екрані |
| alert (текст повідомлення); | alert (“Hello”); |  |

Приклад. Виведення значення змінної

|  |  |
| --- | --- |
| Загальна форма | alert (“ім’я змінної =”+змінна); |
| Приклад (вивести суму) | var a=9;  var b=7;  var sum=a+b;  **alert(“Sum=”+sum);** |
| На екрані |  |

***Виведення даних за допомогою console.log***

При використанні даного підходу інформація відображається у вікні консолі браузера (у Chrome для відображення вінка консолі необхідно скористатися F12).



|  |  |
| --- | --- |
| Загальна форма | ***console.log*** (“текстове повідомлення”); |
| Приклад (вивести суму) | var a = 9;  var b = 7;  var sum = a + b;  **console.log("Sum ="+sum);** |
| На екрані |  |

***Виведення даних за допомогою document.write***

|  |  |
| --- | --- |
| Загальна форма | ***document.write*** (“текстове повідомлення”); |
| Приклад (вивести суму) | **document.write**("Приклад використання <br>");  var a = 9;  var b = 7;  var sum = a + b;  **document.write("Sum ="+sum);** |
| На екрані |  |

*Введення цілого числа*

|  |  |
| --- | --- |
| *Загальна форма* | *Приклад* |
| var змінна =parseInt(prompt(“Змінна=”, “0”)) | var a= parseInt(prompt(“a=”, “0”)) |

*Введення дійсного числа*

|  |  |
| --- | --- |
| *Загальна форма* | *Приклад* |
| var змінна =parseFloat(prompt(“Змінна=”, “0”)) | var a= parseFloat(prompt(“a=”, “0”)) |

## Типи даних

### Числа

**Арифметичні вирази**

**Бінарні операції** (необхідно мати два операнди)

|  |  |
| --- | --- |
| x+y | Сума |
| x-y | Різниця |
| x\*y | Добуток |
| x/y | Частка |
| x%y | Остача від ділення |

**Унарні оператори (**необхідним є один операнд**)**

|  |  |
| --- | --- |
| -x | Зміна знаку на протилежний |
| +x | Збереження знаку. |
| x++,++x | Інкримент: збільшення на 1 |
| x--,--xx | Декримент: зменшення на 1 |

Основні методи:

|  |  |
| --- | --- |
| Math.floor (x) | Округлення вниз |
| Math.ceil (x) | Округлення вгору |
| Math.round (x) | Округлення до найближчого цілого |
| Math.acos (x) | Арккосинус x (в радіанах) |
| Math.asin (x) | Арксинус x (в радіанах) |
| Math.atan (x) | Арктангенс x (в радіанах) |
| Math.atan2 (y, x) | Арктангенс кута до точки (y, x). Виклик еквівалентний Math.atan (y / x). |
| Math.sin (x) | Синус x (в радіанах) |
| Math.cos (x) | Косинус x (в радіанах) |
| Math.tan (x) | Тангенс x (в радіанах) |
| Math.sqrt (x) | Квадратний корінь з x. |
| Math.log (x) | Натуральний логарифм x. |
| Math.pow (x, у) | Піднесення до степеня, повертає xу, наприклад Math.pow (2,3) = 8. |
| Math.abs (x) | Абсолютне значення числа |
| Math.exp (x) | Експонента ex. |
| Math.max (a, b, c ...) | Найбільший зі списку аргументів |
| Math.min (a, b, c ...) | Найменший зі списку аргументів |
| Math.random (x) | Псевдо-випадкове число в інтервалі [0,1) - тобто між 0 (включно) та 1 (не включаючи). Генератор випадкових чисел ініціалізуется поточним часом.   let min=10, max=78   let a =min + Math.floor(Math.random()\*(max-min+1)) |
| Math.PI | Число π |
| Math.Е | Константа Ейлера, е |

## **Перетворення типів**

***Перетворення у число (Number)***

|  |  |
| --- | --- |
| **Значення** | **Перетвориться в ...** |
| undefined | NaN |
| null | 0 |
| true | 1 |
| false | 0 |
| Рядок | Пробіли по краях обрізаються. Далі один з випадків:  1) якщо залишається порожній рядок, то 0;  2) з непорожього рядка зчитується число;  3) якщо у рядку не число, то результат NaN . |

***Перетворення до логічного типу даних (Boolean)***

|  |  |
| --- | --- |
| **Значення** | **Перетвориться в ...** |
| undefined , null | false |
| Числа | Всі true , крім 0 , NaN - false . |
| Рядки | Всі true , крім порожнього рядка "" - false |
| Об'єкти | Завжди true |

***Логічні вирази***

У JavScript використовуються такі оператори порівняння

|  |  |
| --- | --- |
| Більше | a > b |
| Менше | a < b . |
| Більше або дорівнює | a >= b |
| Менше або дорівнює | a <= b |
| Рівне  (a=b це присвоювання!) | a == b |
| Не рівно, ≠ | != |
| Тотожно рівно (співпадає тип і значення) | a === b |
| Не тотожно рівно (не співпадає тип, або не співпадають значення) | A !== b |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| x | y | x&&y  (логічне «і») | x || y  (логічне «або») | !x  (заперечення) |
| false | false | false | false | true |
| false | true | false | true | true |
| true | false | false | true | false |
| true | true | true | true | false |

## **Умовний оператор**

|  |  |
| --- | --- |
| Загальна форма | Приклад |
| **if** (умова)  оператор, якщо умова виконується  **else**  оператор, якщо умова не виконується | if(a>b)  max=a;  else  max=b; |

або скороченій формі.

|  |  |
| --- | --- |
| Загальна форма | Приклад |
| **if** (умова)  оператори, якщо умова виконується | max=a;  if(b>max)  max=b; |

Якщо при виконанні чи невиконанні деяких умов треба виконати більше ніж один оператор, то потрібно використовувати складений оператор (блок), який записується за допомогою фігурних дужок

{

оператор1;

оператор2;

. . . . .

}

|  |  |
| --- | --- |
| Загальна форма | Приклад |
| if (умова)  {  оператори, якщо умова виконується  }  else  {  оператори, якщо умова не виконується  } | if(a>b)  {  max=a;  min=b;  }  else  {  max=b;  min=a;  } |

або скороченій формі.

|  |  |
| --- | --- |
| Загальна форма | Приклад |
| if (умова)  {  оператори, якщо умова виконується  } | if(x!=0)  {  z=1/x;  alert(z);  } |

*Приклад.*  Скласти скрипт який за введеним віком і статтю визначає чи є користувач військовозобов’язаним.

if (!confirm('Ви чоловік?'))

{ alert('жінки не служать');

}

else

{

var age=prompt('Ваш вік','');

if (age==null)

{

alert('Ви не ввели вік')

}

else

{

age=parseInt(age);

if ((age>=18)&&(age<25))

{

alert('служити')} else{alert('не служити')};

}

}

***Тернарний оператор***

|  |  |
| --- | --- |
| Загальна форма | Змінна = умова ? значення1 (умова викон.) : значення2(умова не викон.);  ----- аналог з умовним оператором ------  if ( умова )  змінна = значення1;  else  змінна = значення2; |
| Приклад | **max= a>b ? a : b;**  ------ аналог з умовним оператором ------  if(a>b)  max=a;  else  max=b; |

*Приклад.*  Скласти скрипт для знаходження більшого з двох чисел.

var a=parseFloat(prompt('Введіть перше число',''));

var b=+prompt('Введіть друге число','');

var max=(a>b)?a:b;

alert('Більше число рівне '+max);

**ОПЕРАТОР ВИБОРУ SWITCH**

**switch**(селектор) {

**case** значення 1 : оператори   // if (селектор === значення1)

**break;**

**case** значення 2 : оператори // if (селектор === значення2)

**break;**

. . . . . . . . . . . . .

**case** значення n : оператори // if (селектор === значенняn)

**break;**

**default**: //інакше (всі інші значення)

оператори

}

|  |  |
| --- | --- |
| **Загальна форма** | **Приклад.** Вводиться оцінка – цифра, вивести оцінку прописом (селектор вибору цілого типу). |
| **switch (<селектор вибору>)**  **{**  **case** <знач. 1> : <оператор 1>;  **break;**  **case** < знач. 2> : <оператор 2>;  **break;**  **………………………………………….**  **case** < знач. N> : <оператор N>;  **break;**  **default :** <оператор N+1>;  **break;**  **}** | var score = parseInt(prompt("score", ""));  var result;  //-------------- Знаходимо результат за допомогою switch  switch (score)  {  case 2: result="Незадовільно";  break;  case 3: result="Задовільно.";  break;  case 4: result="Добре";  break;  case 5: result="Відмінно";  break;  default: result="Неправильна оцінка.";  break;  }  alert(result); |

## Цикли

**Оператор циклу з передумовою** while

|  |  |  |
| --- | --- | --- |
| **Програмна структура** | **Аналог на мові блок-схем** | **Приклад.**  Знаходити суму чисел, які вводить користувач, **поки сума є меншою за 100**. |
| while( умова )  оператор ; | **умова**  Оператор  +  – | var sum=0;  while(sum<100)  sum=sum+ +prompt(‘number=’,’0’); |

Якщо тіло циклу складається з більше ніж одного оператора, то необхідно використати складений оператор (записати ці оператори у фігурних дужках).

|  |  |  |
| --- | --- | --- |
| **Програмна структура** | **Аналог на мові блок-схем** | **Приклад.**  Знаходити суму і добуток чисел, які вводить користувач, **поки сума є меншою за 100** |
| while( умова )  **{**  оператор1 ;  . . . . . .  операторN ;  **}** | **умова**  Оператор1  +  –  ОператорN | var sum=0;  var mult=1;  while(sum<=100)  **{**  var number = +prompt(‘number=’,’0’);  sum = sum + number;  mult = mult \* number;  **}** |

Іноді цикл необхідно виконати певну кількість разів.

|  |  |
| --- | --- |
| Загальна форма | Приклад. 100 разів вивести на екран слово «Мир». |
| var лічильник = 0;  while( лічильник < кількість повторень )  {  оператор ;  лічильник ++ ;  } | var **i = 0**;  while (**i < 100**)  {  document.write(‘Мир’);    **i++;**  } |

Також цикл часто використовують у випадку, коли деяка величина (*параметр*) повинна змінюватися від деякого початково до кінцевого значення з певним кроком.

|  |  |
| --- | --- |
| Загальна форма | Приклад. Вивести на екран усі числа кратні трьом від 6 до 28. |
| var параметр = початкове значення ;  while( параметр <= кінцеве значення)  {  оператор ;  параметр = параметр + крок;  } | var **i = 6**;  while (**i <= 28**)  {  document.write( i );    **i=i+3;**  } |

**Цикл з післяумовою** do-while

|  |  |  |
| --- | --- | --- |
| **Програмна структура** | **Аналог на мові блок-схем** | **Приклад.**  Знаходити суму чисел, які вводить користувач поки не буде введено 0. Зауважимо, що принаймні одне число треба ввести (принаймні один раз тіло циклу потрібно виконати). |
| do  {  оператор ;  }  while ( умова ); | **умова**  Оператор  +  – | var sum=0;  var number;  do  {  number= +prompt(“number=”,“0”);  sum=sum+number;  }  while(number!=0); |

**Цикл з параметром for**

Загальний вигляд:

for (<вираз ініціалізації>;<умова продовження>;<ітераційна частина>)

<оператор>;

Аналог на мові блок-схем:

Початкова ініціалізація

Умова продовження циклу

Оператор

Ітераційна частина

+

-

Розв’яжемо раніше розглянуті приклад з використанням циклу for

**Приклад 1.**

|  |  |
| --- | --- |
| Загальна форма | for( лічильник = 0; лічильник < кількість повторень ; лічильник ++)  {  оператор ;  } |
| Приклад, розв’язаний з використанням  for | //----- **100 разів** вивести на екран слово «Мир» ----  for( var **i = 0** ; **i < 100** ; **i++ )**  {  document.write(‘Мир’);  } |
| Розв’язаний з використанням  циклу while | var **i = 0**;  while (**i < 100**)  {  document.write(‘Мир’);  **i++;**  } |

Приклад 2.

|  |  |
| --- | --- |
| Загальна форма | for(**параметр=поч.значення; параметр<=кін.значення; параметр= параметр+крок** )  {  оператор ;  } |
| Приклад, розв’язаний з використанням  for | //------ Вивести на екран усі числа кратні трьом **від 6 до 28**.  for(var **i = 6**; **i <= 28**; **i=i+3 )**  {  document.write( i );  } |
| Розв’язаний з використанням  циклу while | var **i = 6**;  while (**i <= 28**)  {  document.write( i );  **i=i+3;**  } |

**Опратори break і continue**

|  |
| --- |
| do {  var a=Math.floor(Math.random()\*100),  b=+(Math.random()\*100).toFixed(0);  var res=prompt(a+'+'+b+'=','');  if (res==null){  if (confirm('Ви справді бажаєте вийти?')){  break};  }  } while ((a+b!= +res));  if (res!=null){ alert('Вірно')}; |

Буває потрібно вийти одночасно з декількох рівнів циклу. Звичний виклик break не може перервати два цикли відразу. Для цього існують мітки.

|  |
| --- |
| exit:  for (var i = 0; i < 3; i++) {  for (var j = 0; j < 3; j++) {  var input = prompt(a[ '+i+','+j+’]’, ''); if (input == null) break exit; // (\*)  }  };  alert('Готово!'); |

Директива continue припиняє виконання поточної ітерації циклу та виконує перехід до наступної.. Вона також може застосовуватись з мітками.

*Приклад. 4.4.* Вивеси усі прості числа менші від n;

|  |
| --- |
| var n=+prompt('n=','');  document.writeln('<h1>Прості числа до '+n+'</h1><br>');  next: for (var i=1;i<=n;i++){  for (var j=2;j<=Math.sqrt(i);j++){  if (i%j==0){continue next}; //Число не просте, перевіримо наступне.  };  document.writeln(i);  }; |

## Підпрограми

Синтаксис оголошення функції:

|  |
| --- |
| function назва(параметр1, параметр2, … , параметрN ) {  тіло функції;  return результат;  }  Спочатку йде ключ |

*Приклад.* Створити підпрограму,що виводить привітання.

|  |
| --- |
| function showMessage () {     alert ('Привіт всім присутнім!');  }  showMessage ();  showMessage (); |

|  |
| --- |
| 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 т |

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 іодразу | let a = sum(23, 9);  function sum(a, b) {  return a + b;  } |
| **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); |