JavaScript
Функции

Въведение

Цитат от книгата: „Secrets of the JavaScript Ninja(2013,Manning) by John Resig & Bear Bibeaut“, „Chapter 3. Functions are fundamental“ :

You might have been somewhat surprised, upon turning to this part of the book dedicated to JavaScript fundamentals, to see that the first topic of discussion is to be functions rather than objects.

We'll certainly be paying plenty of attention to objects (particularly in chapter 6), but when it comes down to brass tacks, the main difference between writing JavaScript code like the average Joe (or Jill) and writing it like a JavaScript ninja is understanding JavaScript as a functional language. The level of the sophistication of all the code you'll ever write in JavaScript hinges upon this realization.

If you're reading this book, you're not a rank beginner and we're assuming that you know enough object fundamentals to get by for now (and we'll be taking a look at more advanced object concepts in chapter 6), but really understanding functions in JavaScript is the single most important weapon you can wield.

Дефиниция на функция (function definition)

```
function name( <списък параметри>) {
    <тяло на функцията>
};
```

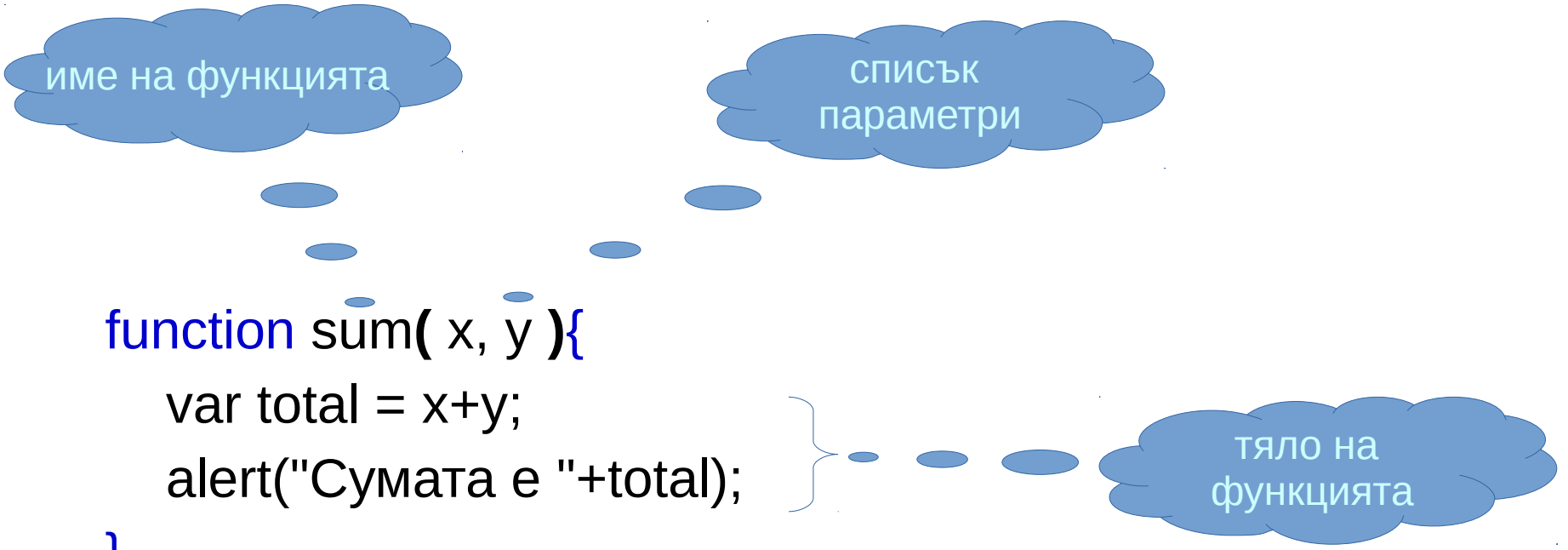
<списък параметри> - имена на параметри, разделени със ','

Параметрите на една функция по същество са локални променливи, които получават стойност при извикването на функцията.

<тяло на функцията> - изрази, завършващи с ';'

В тялото на функцията са действията, които извършва дадената функция.

Дефиниране на функция - пример



Обръщане към функция
Function Invocation

Синоними: Извикване, стартиране, изпълнение на функция

*name***(** <списък аргументи>**)** ;

NB! Кръглите скоби след името на функцията са задължителни, като между името и отварящата скоба не се слага спейс!

Аргументите са стойностите, които се предават на параметрите зададени в дефиницията на функцията.

JS не проверява за съответствие между типа и/или броя на аргументите и параметрите.

Аргументите се предават „по стойност“.

Извикването на функция може да бъде на всяко едно място, където може да имаме променлива.

Обръщане към функция
Function Invocation

```
// дефиниция на функцията sum()
function sum(x,y){
    return x+y;
};

// извикване на sum()
var total = sum(2,3);

// извикване на sum() като аргумент на друга функция/метод
console.log("total=", sum(4,3));
```

Връщане на стойност
Return statement

С **return** се прекратява изпълнението на функцията (изход от функцията). Ако в тялото на функцията има други изрази след него, то те не се изпълняват.

`return` <израз>;

<израз> се изчислява и резултата е стойността, която функцията връща.

```
var sum = function(x,y){
    return x+y;
    console.log("I want to say something!");
};

console.log("Function sum() returns: ", sum(2,3));
```

Анонимни функции

```
function( <списък параметри>) {
    <тяло на функцията>
};
```

```
// извикване на анонимна функция
var sum_result = function(x,y){
    return x+y;
}();
```

NB! Обърнете внимание на кръглите скоби '()' след тялото на фунцията, завършващо с '}'

Функцията като променлива

```
var name = function ( <списък параметри>) {
    <тяло на функцията>
};
```

```
// дефиниране
var sum_func = function(x,y){
    return x+y;
};
// извикване
console.log("total=", sum_func(4,3));
```

Обхват на променливите
Variables Scope

Една локална променлива (декларирана с var) е видима в рамките на блока в който е декларирана, както и във **всички негови под-блокове**. (за разлика от PHP, но точно както е и в Perl, Python, Rubi и др.).

```
var name = "iva";

function f(){
    console.log("name in f:", name);
}

f(); // [name in f: iva]
```

Локален обхват на променливите
Local Variables Scope

Ако една променлива е дефинирана в блок А, и бъде дефинирана (преддефинирана) и в блок Б, който е подблок на А, то в блок Б, променливата ще е свързана със новата си стойност, но в блок А, ще се запази старата стойност на name.

```
var name = "iva"; //дефинираме name

function change_name( ){
    var name = "Ada"; // предефинираме name
    console.log("name in function:", name); // [name in function: Ada]
}

change_name();

console.log("name in main:", name); // [name in main: Iva]
```

Заключение

JS функции: гъвкавост + голяма изразителна сила

// кодът илюстрира възможностите на функциите в JS – не се опитвайте да го разберете (освен, ако вече не сте прочели книга като: JavaScript: The Good Parts на Douglas Crockford)!

```
function mul( x,y ){
    return ( x*y)
};
function applyf( f ){
    return function( x ){
        return function( z ){
            return f(x, z)
        }
    }
};
applyf(mult)(7)(6); // 42
```