

Функции



- Определение и вызов функции
- Аргументы функции и объект Arguments
- Функции как данные
- Функции как методы
- Методы и свойства объекта функции
- Конструктор Function()
- Область видимости и замыкания

Определение и вызов функции



Определение

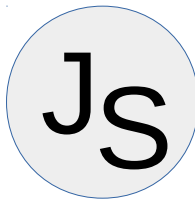
```
function print(msg) {  
    document.write(msg, "<br>");  
}
```

```
function distance(x1, y1, x2, y2) {  
    var dx = x2 - x1;  
    var dy = y2 - y1;  
    return Math.sqrt(dx*dx + dy*dy);  
}
```

ВЫЗОВ

```
print('Привет, ' + name);  
totalDistance =  
    distance(0,0,1,2) + distance(1,2,5,1);
```

Аргументы функции



```
function max( )
{
    var m = Number.NEGATIVE_INFINITY;
    for(var i = 0; i < arguments.length; i++) {
        if (arguments[i] > m) {
            m = arguments[i];
        }
    }
    return m;
}
```

```
var largest = max(1,10,100,2,3,1000,5,10000,6);
```

```
function factorial(x) {
    if (x <= 1) return 1;
    return x * arguments.callee(x-1);
}
```

Rest parameter

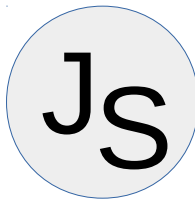


```
function f(a,b,...otherArgs)
{
    var m = Number.NEGATIVE_INFINITY;
    for(var i = 0; i < arguments.length; i++) {
        if (arguments[i] > m) {
            m = arguments[i];
        }
    }
    return m;
}
```

```
var largest = max(1,10,100,2,3,1000,5,10000,6);
```

```
function factorial(x) {
    if (x <= 1) return 1;
    return x * arguments.callee(x-1);
}
```

Функции как данные и как методы



```
function half(x) {return x/2;}  
var half = function(x) {return x/2;};  
var half = (x) => {return x/2};           // ES6  
var half = x => x/2;                       // ES6
```

```
var a = new Array(3);  
a[0] = function(x) {return x/2;}  
a[1] = 20;  
a[2] = a[0](a[1]);
```

```
var o = new Object();  
o.getHalf = new Function('x', 'return x/2;');  
y = o.getHalf(10);
```

Свойства и методы функции



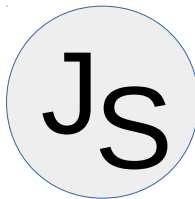
Свойства

- name
- arguments
- constructor
- length

Методы

- apply
- bind
- call
- toString
- valueOf

Параметр "остальные" (ES6)



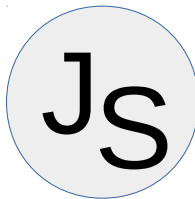
spread syntax

```
function f(a,b,...args) {  
    // ...  
}
```

ЭКВИВАЛЕНТНО

```
function f(a, b) {  
    var args = Array.prototype.slice.call(  
        arguments,  
        f.length  
    );  
  
    // ...  
}
```

Параметр "остальные" (ES6)



```
function f(...[a, b, c]) {  
  return a + b + c;  
}
```

```
f(1) // NaN
```

```
f(1, 2, 3) // 6
```

```
f(1, 2, 3, 4) // 6
```


Замыкание



Лексическое окружение (LexicalEnvironment)

- аргументы
- локальные переменные
- `[[Scope]]`

```
function sayHi(name) {  
  // { name: 'Вася', phrase: undefined }  
  var phrase = "Привет, " + name;  
  
  // { name: 'Вася', phrase: 'Привет, Вася'}  
  alert(phrase);  
}  
  
sayHi('Вася');
```

Замыкание



Текущее значение переменной

```
var greeting = 'Привет';

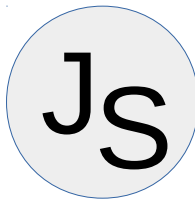
function sayHi(name) {
  alert(greeting + ', ' + name);
}

sayHi('Вася'); // Привет, Вася

greeting = 'Пока';

sayHi('Вася'); // Пока, Вася
```

Замыкание



```
function createInc(startValue) {  
    return function (step) {  
        startValue += step;  
        return startValue;  
    };  
}
```

```
> var inc5 = createInc(5);  
> inc(1)  
6  
> inc(2)  
8
```

```
> var inc0 = createInc(0);  
> inc(1)  
1  
> inc(2)  
3
```

Функции



- Определение и вызов функции
- Аргументы функции и объект Arguments
- Функции как данные
- Функции как методы
- Методы и свойства объекта функции
- Конструктор Function()
- Область видимости и замыкания

Определение и вызов функции



Определение

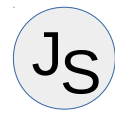
```
function print(msg){  
    document.write(msg, "<br>");  
}
```

```
function distance(x1, y1, x2, y2){  
    var dx = x2 - x1;  
    var dy = y2 - y1;  
    return Math.sqrt(dx*dx + dy*dy);  
}
```

Вызов

```
print('Привет, '+name);  
totalDistance =  
    distance(0,0,1,2) + distance(1,2,5,1);
```

Аргументы функции



```
function max( )
{
  var m = Number.NEGATIVE_INFINITY;
  for(var i = 0; i < arguments.length; i++){
    if (arguments[i] > m){
      m = arguments[i];
    }
  }
  return m;
}

var largest = max(1,10,100,2,3,1000,5,10000,6);
```

```
function factorial(x) {
  if (x <= 1) return 1;
  return x * arguments.callee(x-1);
}
```

Rest parameter



```
function f(a,b,...otherArgs)
{
  var m = Number.NEGATIVE_INFINITY;
  for(var i = 0; i < arguments.length; i++){
    if (arguments[i] > m){
      m = arguments[i];
    }
  }
  return m;
}

var largest = max(1,10,100,2,3,1000,5,10000,6);
```

```
function factorial(x) {
  if (x <= 1) return 1;
  return x * arguments.callee(x-1);
}
```

Функции как данные и как методы



```
function half(x) {return x/2;}  
var half = function(x) {return x/2;};  
var half = (x) => {return x/2};           // ES6  
var half = x => x/2;                       // ES6
```

```
var a = new Array(3);  
a[0] = function(x) {return x/2;}  
a[1] = 20;  
a[2] = a[0](a[1]);
```

```
var o = new Object();  
o.getHalf = new Function('x', 'return x/2;');  
y = o.getHalf(10);
```


Свойства и методы функции



Свойства

- name
- arguments
- constructor
- length

Методы

- apply
- bind
- call
- toString
- valueOf

Параметр "остальные" (ES6)



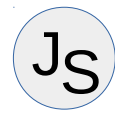
spread syntax

```
function f(a,b,...args){  
  // ...  
}
```

ЭКВИВАЛЕНТНО

```
function f(a, b) {  
  var args = Array.prototype.slice.call(  
    arguments,  
    f.length  
  );  
  
  // ...  
}
```

Параметр "остальные" (ES6)



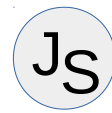
```
function f(...[a, b, c]) {  
  return a + b + c;  
}
```

```
f(1)           // NaN
```

```
f(1, 2, 3)     // 6
```

```
f(1, 2, 3, 4)  // 6
```

Замыкание

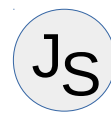


Лексическое окружение (LexicalEnvironment)

- аргументы
- локальные переменные
- [[Scope]]

```
function sayHi(name) {  
  // { name: 'Вася', phrase: undefined }  
  var phrase = "Привет, " + name;  
  
  // { name: 'Вася', phrase: 'Привет, Вася'}  
  alert(phrase);  
}  
  
sayHi('Вася');
```

Замыкание



Текущее значение переменной

```
var greeting = 'Привет';

function sayHi(name) {
  alert(greeting + ', ' + name);
}

sayHi('Вася'); // Привет, Вася

greeting = 'Пока';

sayHi('Вася'); // Пока, Вася
```

Замыкание



```
function createInc(startValue) {  
  return function (step) {  
    startValue += step;  
    return startValue;  
  };  
}
```

```
> var inc5 = createInc(5);  
> inc(1)  
6  
> inc(2)  
8
```

```
> var inc0 = createInc(0);  
> inc(1)  
1  
> inc(2)  
3
```