

Функции

План урока

1. Что такое функция? Объявление функции.
2. Разный синтаксис объявления функций
3. Параметры и аргументы функции
4. Параметры по умолчанию
5. Возвращаемое значение функцией
6. Область видимости функции
7. Стрелочные функции
8. Неизвестно количество аргументов
9. Функциональное программирование

Что такое функция?

Функция в программировании — фрагмент программного кода (подпрограмма), к которому можно обратиться из другого места программы.

Объявление функции в Javascript

```
let n = 10;
let sum = 0;
for(let i = 0; i < n; i++){
    sum += i;
}

let n2 = 100;
let sum2 = 0;
for(let i = 0; i < n2; i++){
    sum2 += i;
}
```

```
function getSum(n){
    let sum = 0;
    for(let i = 0; i < n; i++){
        sum += i;
    }
    return sum;
}

getSum(10);
getSum(100);
getSum(1000)
// и так далее
```

3 варианта объявления

1. Function declaration:

```
function getName() {  
  // тело функции  
}
```

Эта функция видна в любом месте программы

2. Function expression:

```
const getName = function() {  
  // тело функции  
}
```

Эта функция видна только после объявления

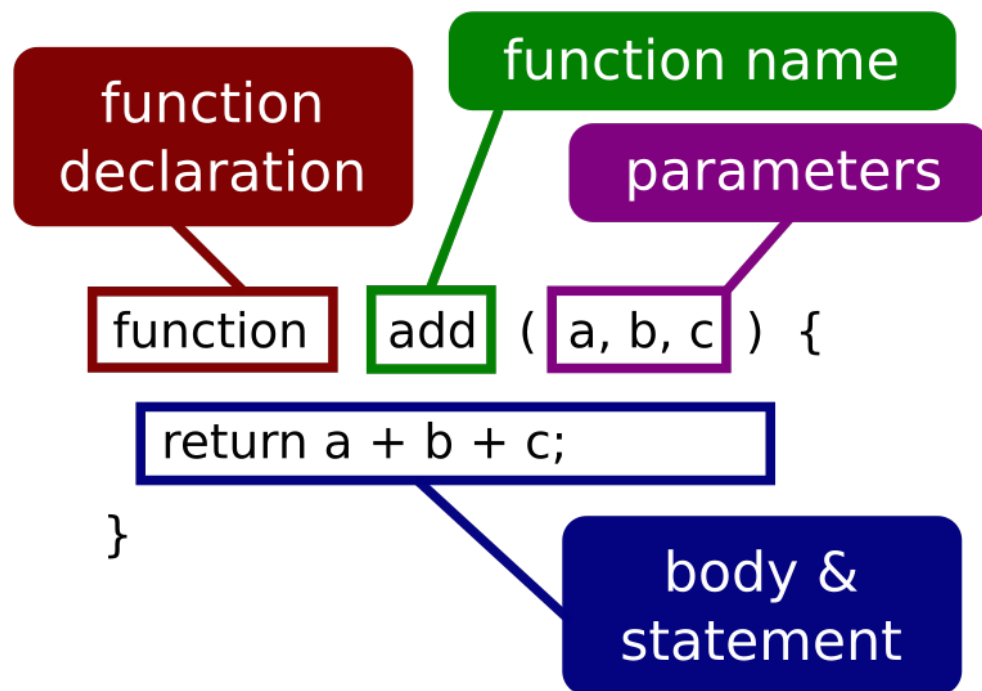
Стрелочные функции

```
const nothing = () => "nothing";  
const abs     = x   => Math.abs(x);  
const add     = (x, y) => x + y;
```

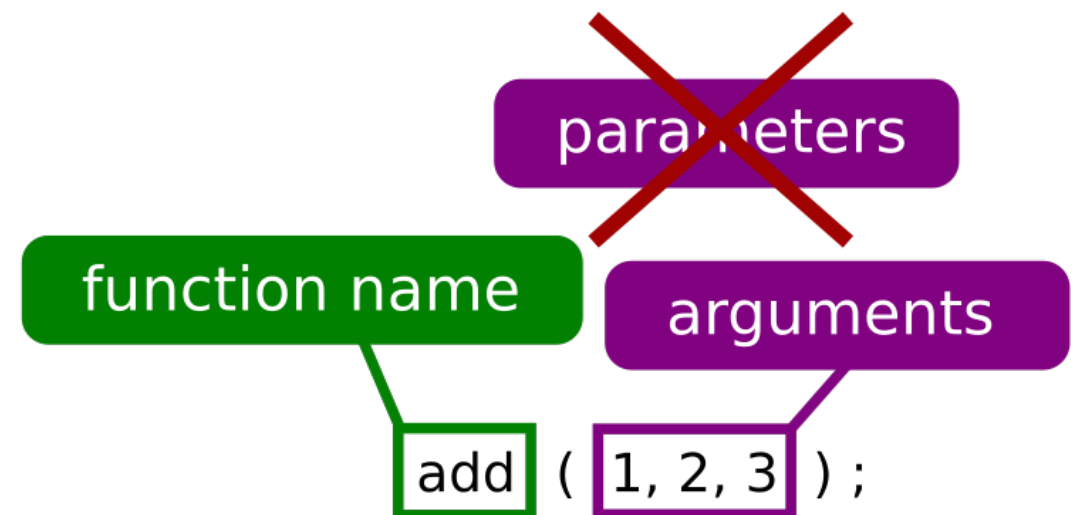
```
nothing(); // nothing  
abs(-5);   // 5  
add(-5, 5); // 0
```

Параметры и аргументы функции

Объявление функции

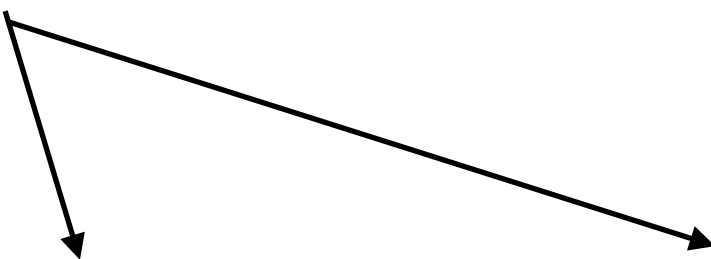


Вызов функции



Параметры по умолчанию

Если аргумент на этой позиции === undefined



```
function showTestHours (name = 'неизвестно', hours = 30) {  
  return `Имя студента(ки) ${name}, использовано ${hours} тестовых часа(ов)`;  
}
```

Очень полезно для типичных случаев

Return. Что вернет функция

```
let sum = 0;

function changeSum(n){
    for(let i = 0; i < n; i++){
        sum += i;
    }
    // Функция изменяет состояние программы.
    // Вернет undefined;
}
```

```
function changeSum(n){
    let sum = 0;
    for(let i = 0; i < n; i++){
        sum += i;
    }
    // Функция явно возвращает значение
    return sum;
}
```

Область видимости переменных

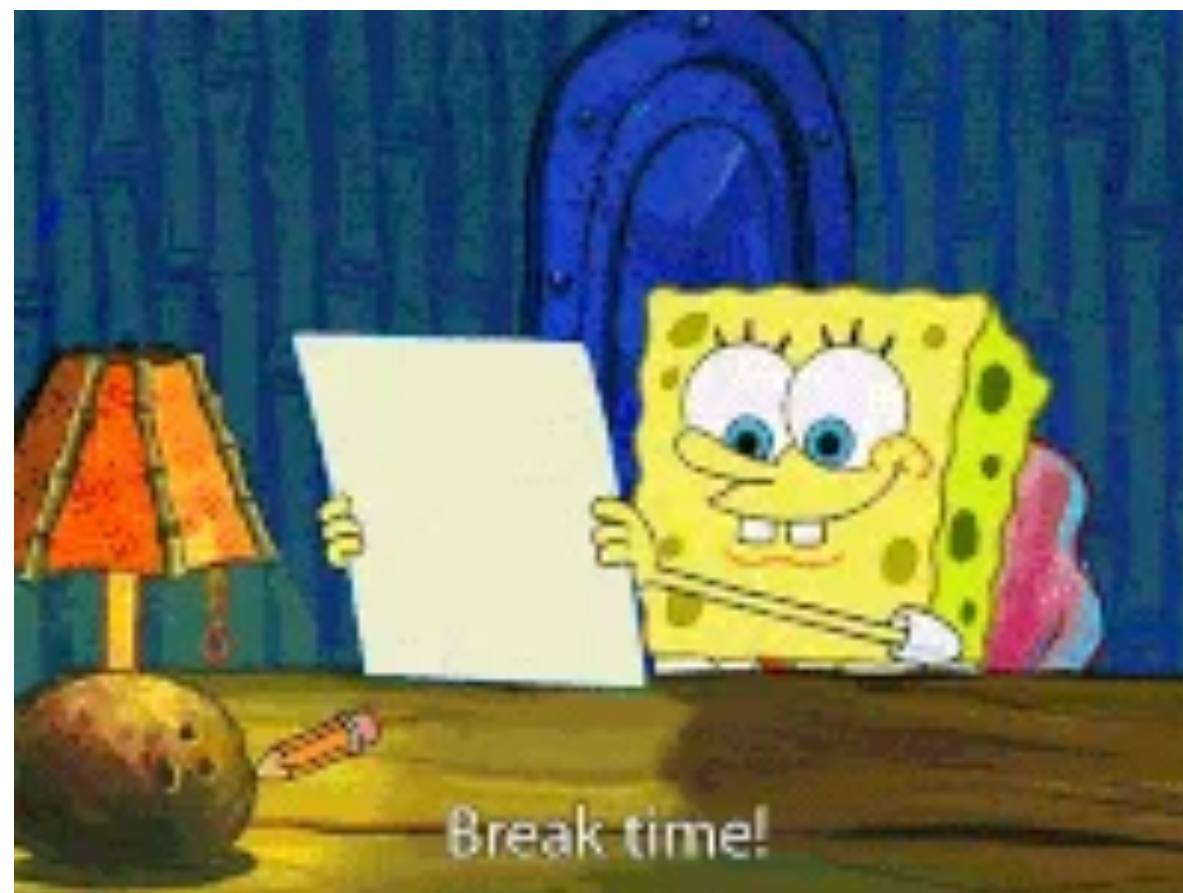
Функции создают собственную область видимости (фигурные скобки тела функции ограничивают эту область). Это означает, что переменная созданная с помощью `var`, `let` или `const` внутри функции не будет доступна коду вне этой функции.

```
let outer = 'Outer';  
function getInner(){  
    let inner = 'Inner!';  
    // Внешние переменные видны  
    // внутри функции  
    console.log(outer);  
    return inner;  
}  
  
console.log(inner); // Ошибка!  
// Переменная объявлена внутри функции  
// видна только в ней
```

Практика

1. Функция которая складывает переданные в неё два числа
2. Функция получает строку с именем, а возвращает строку: «Ваше имя: Vlad»
3. Функция получает число. Если число **<1024**, то переменная **device = 'mobile'**, в ином случае **'desktop'**
4. Функция складывает любое количество чисел переданных в неё (после перерыва)

Перерыв



REST и SPREAD операторы в функциях

Rest-оператор – Оператор, который позволяет собрать все переданные в функцию аргументы в один общий массив. (Массивы будут рассмотрены на следующем занятии)

Spread-оператор – Обратный оператор к rest. Позволяет разложить массив на аргументы в функции.

```
function getSum(...numbers){  
    document.writeln(numbers); // функция соберет все  
    аргументы переданные в нее и выведет массив`  
};  
  
const args = [1, 2, 3, 4];  
// при вызове функции мы можем разложить массив на  
    аргументы  
getSum(...args);
```

Функциональное программирование (ФП)

Это набор общепринятых стандартов написания функций – которые сделают ваш код читабельнее и надёжнее. В курсе мы будем рассматривать базовые концепции функционального программирования.

1. Мутабельность и иммутабельность
2. Чистые функции (Pure Functions)
3. IIFE
4. Single Responsibility

Мутабельность и иммутабельность

Не бро

Если выполняя свою работу функция меняет данные вне своей области видимости – она называется **мутабельной**

Твой бро

Иммутабельными – называются такие функции, которые не изменяют переменные вне своей области видимости. Мы должны стремиться всегда писать только такие функции.

Чистые функции

Иммутабельные функции результат которых **предсказуем**

Random и Date.now() – не использовать



IIFE

Это функция немедленного вызова – объявляется и тут же вызывается. (синтаксис ниже в примере.)
Такие функции нужны прежде всего для того, чтобы скрыть переменные в своей области видимости и использовались раньше для создания "модульности" кода.

```
(function(moduleName){  
  // переменные не будут видны за рамками текущей функции  
  document.writeln(moduleName);  
  let a = 100;  
  let b = 200;  
})("testModule");
```

Рекурсия

