

Functions часть 1

Функции - философия, синтаксис,
виды.

Локальные и внешние переменные

Урок

1	2	3	4	5	6
7	8	9	10	11	12
13	14	15	16	17	18

Функция позволяет вычислять значение или совершать процедуру множество раз

```
function calcBMI(weight, height) {  
  let bmi = weight / height / height;  
  
  if (bmi <= 18.5) {  
    return 'Underweight';  
  } else if (bmi > 18.5 && bmi <= 25.0) {  
    return 'Normal';  
  } else if (bmi > 25 && bmi <= 30.0) {  
    return 'Overweight';  
  } else return 'Obese';  
}
```

Объявление и вызов функции

Объявление функции

Для создания функции мы используем объявление функции

```
function greet() {  
    console.log('Hello students')  
}
```

Вызов функции

```
greet()
```

Вызов функции выполняет тело функции

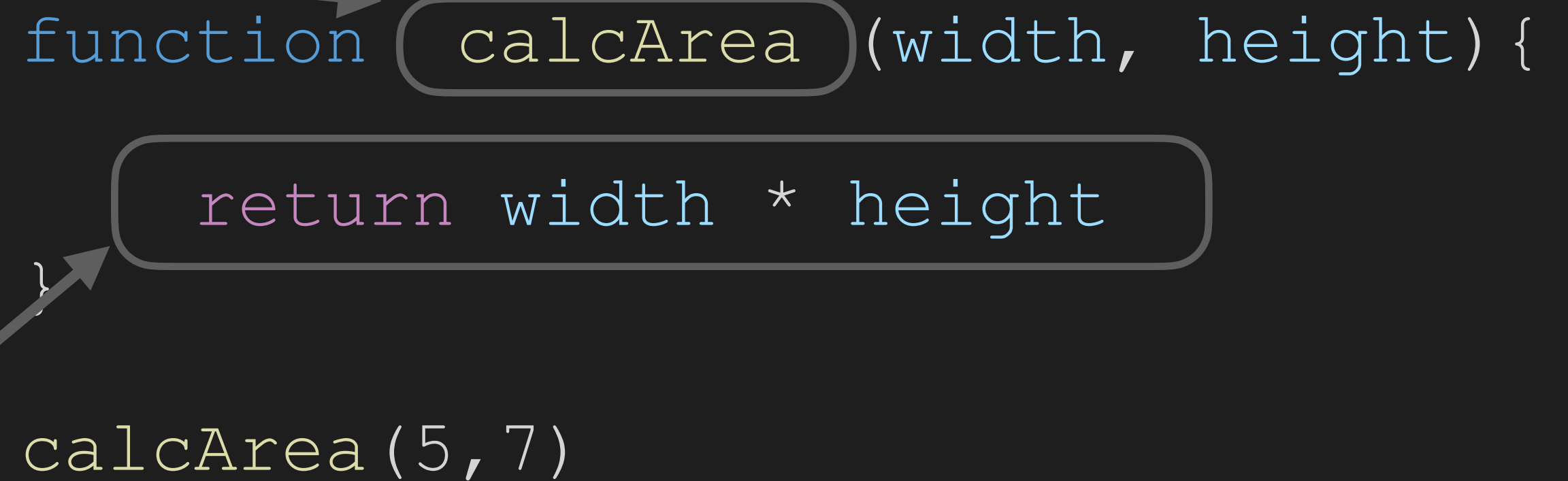
Название и тело функции

Название функции

Название функции служит
как ссылка для ее
применения

Тело функции

```
function calcArea(width, height) {  
    return width * height  
}  
calcArea(5, 7)
```



Функция состоит из последовательности
инструкций, называемой телом функции

Параметры и аргументы функции

Параметры функции

```
function calcSum(a, b) {  
    return a + b  
}
```

Аргументы функции

```
calcSum(41, 1)
```

Параметры служат как переменные с которыми работает функция внутри своей области видимости

Возвращение результата работы функции

Результат функции

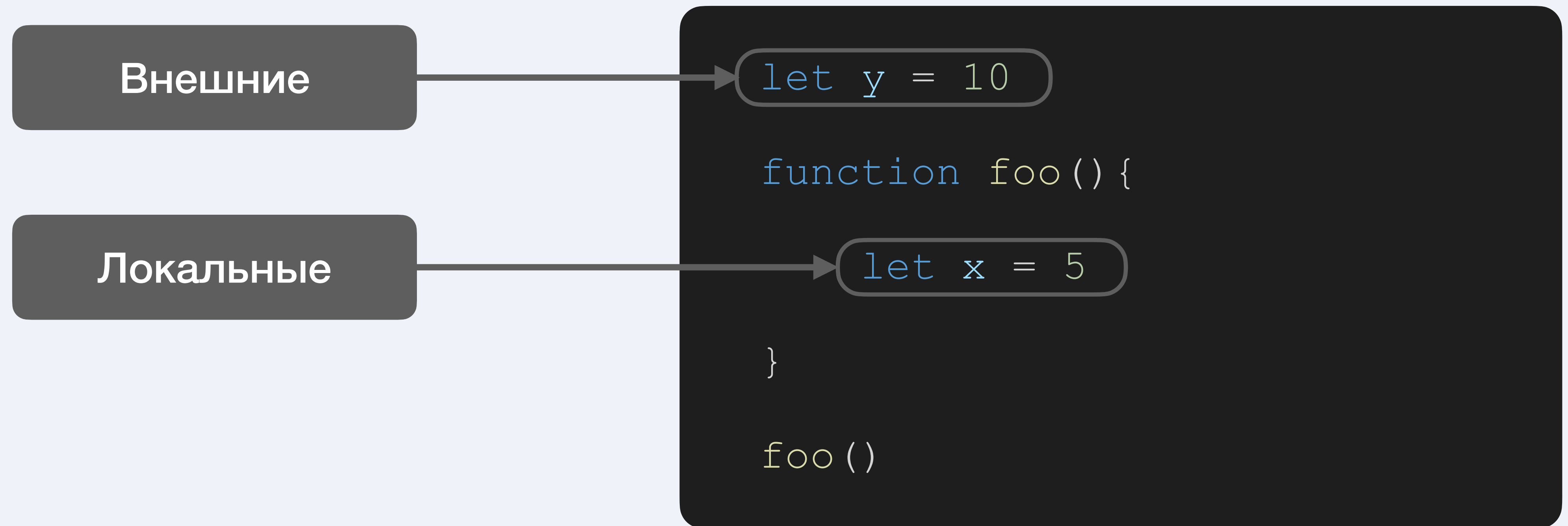
```
function calcSum(a, b) {  
  return a + b  
}
```

```
calcSum(41, 1)
```

Чтобы вернуть значение, отличное от значения по умолчанию, в функции должна быть инструкция **return**, которая указывает, что именно нужно вернуть.

Функция без него вернёт значение по умолчанию.

Локальные и внешние переменные



Функция может работать как с внешними так и с локальными переменными

Function Declaration vs Function Expression

```
function foo(a, b) {  
    return a + b  
}
```

```
foo(41, 1)
```

```
const foo = function(a, b) {  
    return a + b  
}
```

```
foo(41, 1)
```

Function expression

Function expression

```
let calcSum = function(a, b) {  
    return a + b;  
};  
  
calcSum(5, 8);
```

- Функциональные выражения в JavaScript не поднимаются (hoisting), в отличие от объявленных функций. Вы не можете использовать функциональные выражения прежде, чем вы их определили.
- Главным отличием между Function Declaration является *имя функции*, которое в случае функциональных выражений может быть опущено для создания *анонимных функций*.