







Frontender[1.0] JavaScript - Functions.

Функции, Scope, Declaration vs expression, naming

 YouTube	https://youtu.be/T_GNySJsksps
 Telegram	https://t.me/Dmitry_Kolotilshikov
 Github	https://github.com/DmitryKolotilshikov/
 Boosty	https://boosty.to/dmitry_ko
# Номер урока	17



Задачи к этому уроку тут https://boosty.to/dmitry_ko

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <script src="script.js"></script>
  <title>Functions scope</title>
</head>
<body>
</body>
</html>
```

```
// Functions - scope, default values, declaration vs expression

/*
Выбор имени функции
Функция – это действие. Поэтому имя функции обычно является глаголом. Оно должно быть кратки
м, точным и описывать действие функции, чтобы программист, который будет читать код, получил
верное представление о том, что делает функция.

Как правило, используются глагольные префиксы, обозначающие общий характер действия, после ко
торых следует уточнение. Обычно в командах разработчиков действуют соглашения, касающиеся зна
чений этих префиксов.

Например, функции, начинающиеся с "show" обычно что-то показывают.
```

Функции, начинающиеся с...

"get..." – возвращают значение,

"calc..." – что-то вычисляют,

"create..." – что-то создают,

"check..." – что-то проверяют и возвращают логическое значение, и т.д.

Примеры таких имён:

```
showMessage(..)    // показывает сообщение
getAge(..)          // возвращает возраст (получая его каким-то образом)
calcSum(..)         // вычисляет сумму и возвращает результат
createForm(..)      // создаёт форму (и обычно возвращает её)
checkPermission(..) // проверяет доступ, возвращая true/false
```

Одна функция – одно действие

Функция должна делать только то, что явно подразумевается её названием. И это должно быть одним действием.

*/

```
const log = console.log;
```

```
// ----- Function declaration vs Function expression -----
```

```
// Function declaration - можно вызывать функцию до ее объявления
```

```
/*
```

Function Declaration: функция объявляется отдельной конструкцией «function...» в основном потоке кода.

Function Declaration может быть вызвана раньше, чем она объявлена.

```
*/
```

```
greeting();
```

```
function greeting() {
  log("Hello friends [1]");
}
```

```
/*
```

Function Expression: функция, созданная внутри другого выражения или синтаксической конструкции.

Function Expression создаётся, когда выполнение доходит до него, и затем уже может использоваться.

```
*/
```

```
// greetingFn(); // error
```

```
const greetingFn = function greeting() {
  log("Hello friends [2]");
}
```

```
greetingFn();
```

```
// ----- Scope (Область видимости) -----
```

```
// глобальная область видимости (global scope)
```

```
const num1 = 10;
```

```

// {} - объявляют локальную (внутреннюю) область видимости
{
  // local scope
  const num2 = 20;
  log(num2);
  log(num1);
  var num3 = 100; // var - исключение
}

log(num1);
// log(num2); // error
log(num3);

function showScopeExample() {
  const num4 = 1000;
  log("showScopeExample: ", num1, num4);

  function nestedFn() {
    const num5 = 3000;
    log("nestedFn: ", num4, num5);
  }

  nestedFn();
}

showScopeExample();
// nestedFn(); // error
// log(num4); // error

// ----- Default values - Значения по умолчанию -----

// function formatGreeting(name, emoji = "😬") {
//   log(`Hello ${name} ${emoji}`);
// }
function formatGreeting(name, emoji = "😬") {
  return `Hello ${name} ${emoji}`;
}

log(formatGreeting("Dmitry", "👍"));
log(formatGreeting("Dmitry"));

// const formattedGreeting = formatGreeting("Dmitry", "😄💣😄"); // результат работы функции
const formattedGreeting = formatGreeting; // function expression

log(formattedGreeting("Dmitry", "😄💣😄"));

```