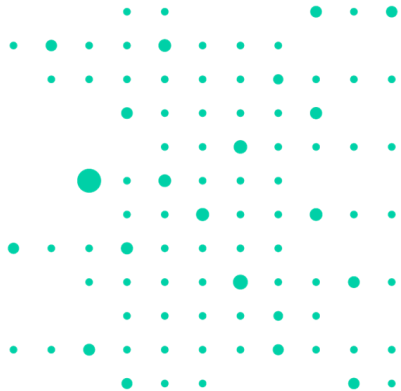




**Front-end overview.  
Functions, Arrow functions,  
default function parameters**



# Основы JavaScript

---



# Объявление функции

---

Конструкция

```
function showMessage() {  
    alert( 'Всем привет!' );  
}
```

```
showMessage();
```

```
function имя(параметры) { ...тело... }
```

# Функция и переменные

---

Конструкция

```
function showMessage() {  
    let message = "Привет, я JavaScript!"; // локальная переменная  
    alert( message );  
}  
  
showMessage(); // Привет, я JavaScript!  
alert( message ); // <-- будет ошибка
```

# Параметры функции

---

## Конструкция

```
function showMessage(from, text) { // аргументы: from, text  
    alert(from + ': ' + text);  
}
```

```
showMessage('Аня', 'Привет!'); // Аня: Привет! (*)
```

# Возврат значения

---

Конструкция

```
function sum(a, b) {  
    return a + b;  
}
```

```
let result = sum(1, 2);  
alert( result ); // 3
```

# Выбор имени функции

---

- Имя функции должно понятно и чётко отражать, что она делает. Увидев её вызов в коде, вы должны тут же понимать, что она делает, и что возвращает.
- Функция – это действие, поэтому её имя обычно является глаголом.
- Есть много общепринятых префиксов, таких как:  
create..., show..., get..., check... и т.д.

Пользуйтесь ими как подсказками, поясняющими, что делает функция.



# **Arrow functions**



# Синтаксис

---

Существует ещё более простой и краткий синтаксис для создания функций, который часто лучше, чем синтаксис Function Expression.

```
let func = (arg1, arg2, ...argN) => expression
```

# Многострочные стрелочные функции

---

`let sum = (a, b) => {` // фигурная скобка, открывающая тело  
многострочной функции

`let result = a + b;`

`return result;` // при фигурных скобках для возврата значения нужно

явно вызвать return

`};`

# Контекст выполнения

---

"Стрелки" не имеют своего собственного `this``. Она его **наследует** от родительской функции.

То есть `this`` таки существует, всегда на что-то ссылается. Вопрос только в какой ситуации на что он будет указывать.

- 1) Если стрелочная функция объявлена в глобальном контексте, т.е. просто написана посреди файла - ее `this`` ссылается на глобальный объект `Window``
- 2) Если стрелочная функция объявлена внутри обычной функции - `this`` стрелочной будет ссылаться туда, куда ссылается и `this`` обычной. Проще говоря, если `function`` объявлена как метод объекта, то и у стрелочной контекстом будет этот объект.