Functions часть 2

- Стрелочные функции
- Функции обратного вызова
- Продвинутая работа с массивами
- Немедленно вызываемые функции
- Функции конструкторы

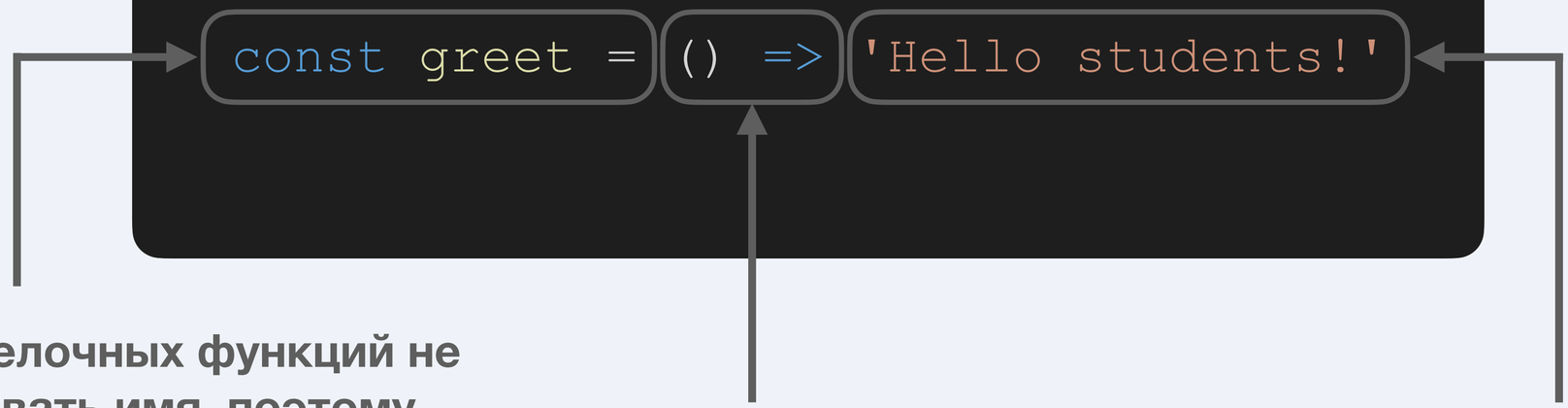
< Teach
Me
Skills / >

Урок

< Teach
Me
Skills / >

Стрелочная функция

```
const greet = () => 'Hello students!'
```



- **Выражение стрелочных функций не позволяют задавать имя, поэтому стрелочные функции анонимны, если их ни к чему не присвоить.**
- **Стрелочная функция**
- **Тело функции**

< Teach Me Skills / >

Стрелочная функция

Concise body

Block body

```
const greet = () => 'Hello students!'

const greet = () => {
    return 'Hello students!'
}
```

- **Тело стрелочной функции может иметь краткую (concise body) или блочную (block body) форму. Блочная форма не возвращает значение, необходимо явно вернуть значение.**

Особенности стрелочной функции

- **Стрелочные функции не содержат собственный контекст this, а используют значение this окружающего контекста.**

- **Стрелочные функции не могут быть использованы как конструктор и вызовут ошибку при использовании с new**

Callback функция


```
function foo(bar) {
    let name = 'Lizzy';
    bar(name);
}

foo(bar);

function bar(name) {
    console.log('Hello ' + name);
}
```

- **Функция обратного вызова - это функция, переданная в другую функцию в качестве аргумента, которая затем вызывается по завершению какого-либо действия.**

Метод массива forEach

Метод `forEach()` выполняет функцию `callback` один раз для каждого элемента, находящегося в массиве в порядке возрастания. Она не будет вызвана для удалённых или пропущенных элементов массива.

Функция, которая будет вызвана для каждого элемента массива. Она принимает от одного до трёх аргументов:

```
array.forEach(callback(value, index, array), this)
```

Текущий обрабатываемый элемент в массиве.

Индекс текущего обрабатываемого элемента в массиве.

Массив, по которому осуществляется проход.

Необязательный параметр. Значение, используемое в качестве `this` при вызове функции `callback`

Метод массива map

Функция, которая будет вызвана для каждого элемента массива. Она принимает от одного до трёх аргументов:

```
const newArray = array.map(callback(value, index, array), this)
```

Текущий обрабатываемый элемент в массиве.

Индекс текущего обрабатываемого элемента в массиве.

Массив, по которому осуществляется проход.

Необязательный параметр. Значение, используемое в качестве `this` при вызове функции `callback`

Метод `map` вызывает переданную функцию `callback` один раз для каждого элемента, в порядке их появления и конструирует новый массив из результатов её вызова.

Метод массива reduce

Метод `reduce()` выполняет функцию `callback` один раз для каждого элемента, присутствующего в массиве, за исключением пустот, принимая четыре аргумента: начальное значение (или значение от предыдущего вызова `callback`), значение текущего элемента, текущий индекс и массив, по которому происходит итерация.

Необязательный параметр. Объект, используемый в качестве первого аргумента при первом вызове функции `callback`.

```
const value = array.reduce(callback(accum, value, index, array), initAccum)
```

Аккумулятор, аккумулирующий значение, которое возвращает функция **callback** после посещения очередного элемента, либо значение `initialValue`, если оно предоставлено

Метод массива filter

Функция, которая будет вызвана для каждого элемента массива. Она принимает от одного до трёх аргументов:

```
const newArray = array.filter(callback(value, index, array), this)
```

Текущий обрабатываемый элемент в массиве.

Индекс текущего обрабатываемого элемента в массиве.

Массив, по которому осуществляется проход.

Необязательный параметр. Значение, используемое в качестве `this` при вызове функции `callback`

Метод `filter()` вызывает переданную функцию `callback` один раз для каждого элемента, присутствующего в массиве, и конструирует новый массив со всеми значениями, для которых функция `callback` вернула `true`

Метод массива find

Метод `find` вызывает переданную функцию `callback` один раз для каждого элемента, присутствующего в массиве, до тех пор, пока она не вернёт `true`. Если такой элемент найден, метод `find` немедленно вернёт значение этого элемента. В противном случае, метод `find` вернёт `undefined`.

```
const value = array.find(callback(value, index, array), this)
```

Текущий обрабатываемый элемент в массиве.

Индекс текущего обрабатываемого элемента в массиве.

Массив, по которому осуществляется проход.

Необязательный параметр. Значение, используемое в качестве `this` при вызове функции `callback`

Немедленно вызываемая функция

- **IIFE (Immediately Invoked Function Expression) это JavaScript функция, которая выполняется сразу же после того, как она была определена.**
- **Состоит из двух частей.**

```
(name => `Hello ${name}`)('Bob')
```

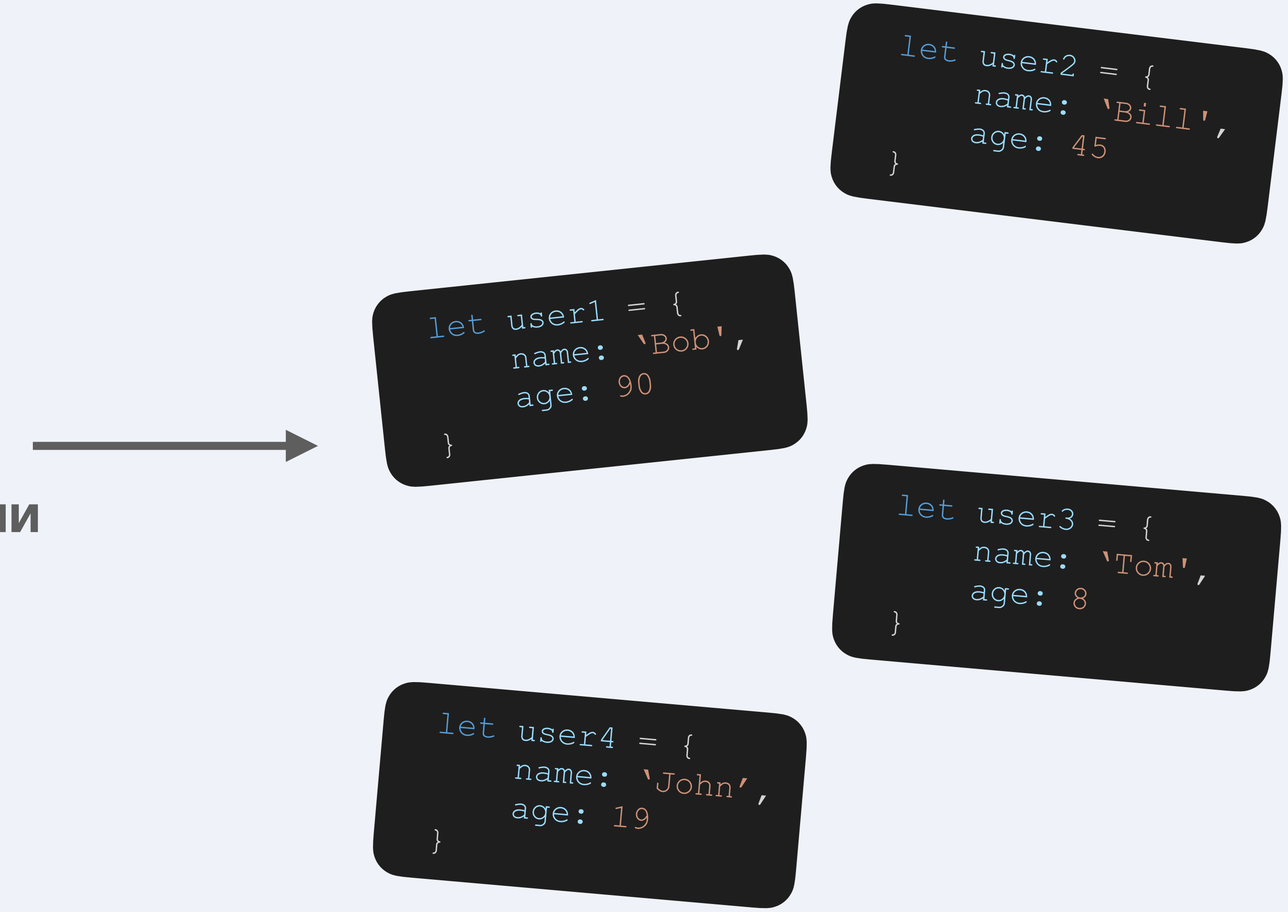
Оператор группировки ( )

Вызов выражения ( )

< Teach
Me
Skills / >

Функции конструкторы

Если нам нужно создать множество однотипных объектов мы можем воспользоваться функциями конструкторами



Функции конструкторы

Функции-конструкторы являются обычными функциями. Но есть два соглашения:

- **Имя функции-конструктора должно начинаться с большой буквы.**
- **Функция-конструктор должна вызываться при помощи оператора `"new"`**

Синтаксис

Присваиваем в объект переданные параметры

```
function Animal(name, voice){
    this.name = name;
    this.voice = voice;
}

let cat = new Animal('Cleo', 'Meaw');
let dog = new Animal('Sharif', 'Gav')
```

Оператор new создает экземпляр объекта

Под капотом движка

Создается пустой объект и присваиваем его в this

Возвращаем объект this

```
function Animal(name, voice){
    // this = {}

    this.name = name;
    this.voice = voice;

    // return this
}

let cat = new Animal('Cleo', 'Meaw');
let dog = new Animal('Sharik', 'Gav')
```

- Задача функции конструктора создать экземпляр объекта, а не возвращать явно какой-то результат

Методы

В this мы можем добавлять не только свойства, но и методы

```
function Animal(name, voice){
    this.name = name;
    this.voice = voice;

    this.getVoice = function(){
        console.log(this.voice);
    }
}

let cat = new Animal('Cleo', 'Meaw');
let dog = new Animal('Sharik', 'Gav')

cat.getVoice() // Meaw
```