**Нововведения в Ecmascript 6**

**Let**

var в качестве области видимости дает в результате тело функции, в которой объявлена переменная, а область видимости переменной, объявленной с помощью let, – это только блок, в котором выполнено объявление.

for(let i = 0, l = list.length; i < l; i++) {

// do something with list[i]

}

console.log(i); // undefined

**CONST**

const MY\_CONSTANT = 1;

MY\_CONSTANT = 2 // Error

const SOME\_CONST; // Error

**Стрелочные функции**

let titles = books.map( item => item.title );

// без аргументов

books.map( () => 1 ); // [1, 1]

// несколько аргументов

[1,2].map( (n, index) => n \* index ); // [0, 2]

**Методы для обработка строк**

'my string'.startsWith('my'); //true

'my string'.endsWith('my'); // false

'my string'.includes('str'); // true

**Апргрейд Math**

Math.sign(5); // 1

Math.sign(-9); // -1

Math.trunc(5.9); // 5

Math.trunc(5.123); // 5

Math.cbrt(64); // 4

**Оператор распространения**

Оператор распространения (...) – это очень удобный синтаксис для разворачивания элементов массива в определенных местах, например, в качестве аргументов в вызовах функций.

let values = [1, 2, 4];

let some = [...values, 8]; // [1, 2, 4, 8]

let more = [...values, 8, ...values]; // [1, 2, 4, 8, 1, 2, 4]

**Деструктурирование**

Деструктурирование обеспечивает удобный способ для извлечения данных из объектов или массивов. С этим синтаксисом нескольким переменным могут быть присвоены значения за один раз.

let [x, y] = [1, 2]; // x = 1, y = 2

**Параметры**

Значения по умолчанию и остаточные параметры.

function doSomething(x, y = 2) {

return x \* y;

}

function doSomething(x, ...remaining) {

return x \* remaining.length;

}

doSomething(5, 0, 0, 0); // 15

**Модули**

// lib/math.js

export function sum(x, y) {

return x + y;

}

export var pi = 3.141593;

// app.js

import { sum, pi } from "lib/math";

console.log('2π = ' + sum(pi, pi));

Чтобы импортировать модуль целиком, должен быть использован символ \* в сочетании с ключевым словом, чтобы дать модулю локальное имя:

import \* as math from "lib/math";

console.log('2π = ' + math.sum(math.pi, math.pi));

**Классы**

**class** Vehicle {

**constructor**(name) {

this.name = name;

this.kind = 'vehicle';

}

getName() {

return this.name;

}

}

// Создадим экземпляр

let myVehicle = new Vehicle('rocky');

**Транспиляция**

Поддержка ES6 браузерами пока не очень обширна и сильно варьируется, поэтому код желательно подвергать транспиляции(преобразованию в предыдущие версии JavaScript).