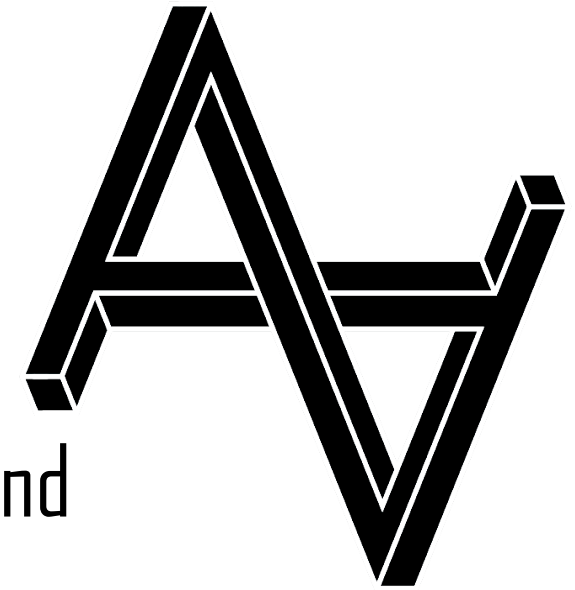
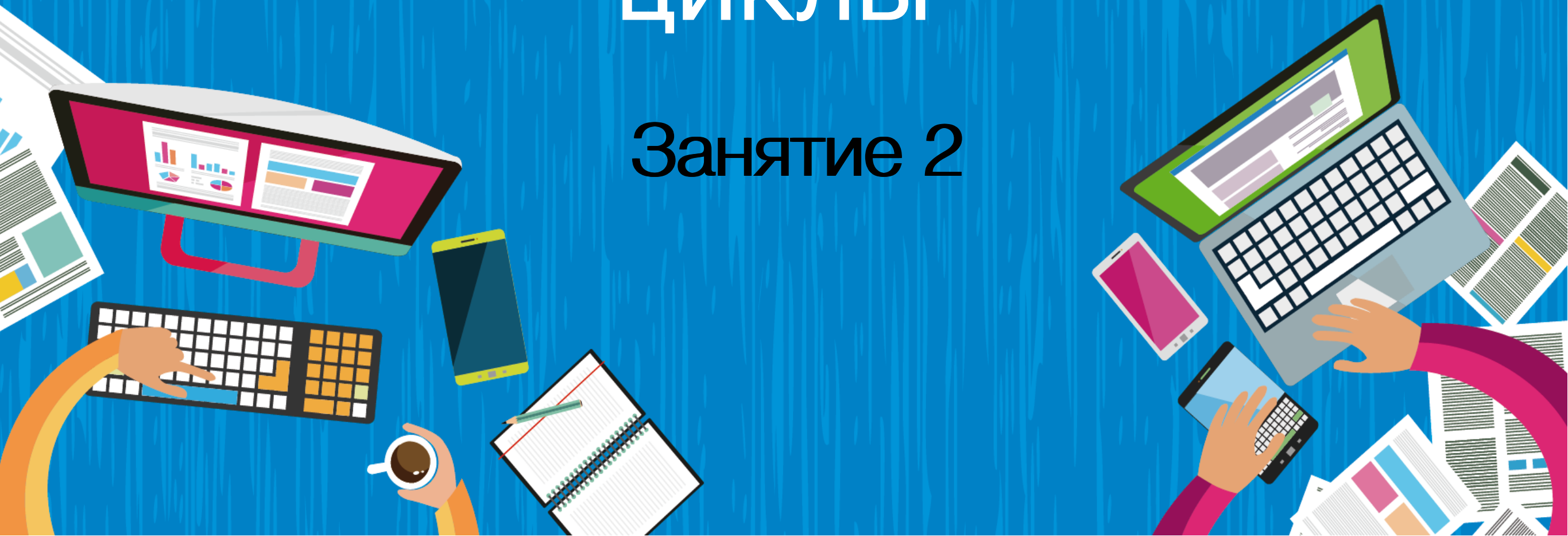


ArtFrontEnd



JS. Операторы, функции и ЦИКЛЫ

Занятие 2



План занятия

- Циклы
- Функции
- Функциональные выражения
- Рекурсия
- Стек
- Именованные функциональные выражения
- Структуры данных
- Методы и свойства

Циклы



do

```
do {  
    var name = prompt( 'Who are you?' );  
} while ( !name );  
  
console.log(name);
```

for

```
for (начало; условие; шаг) {  
    // ... тело цикла ...  
}
```

for

```
var i;
```

```
for (i = 0; i < 3; i++) {  
    console.log(i);  
}
```

for

```
for (var i = 0; i < 3; i++) {  
    console.log(i); // 0, 1, 2  
}
```

for

```
var result = 1;  
for (var counter = 0; counter < 10; counter++) {  
    result = result * 2;  
}  
console.log(result);  
// → 1024
```


Выход из цикла

```
for (var current = 20; ; current++) {  
    if (current % 7 == 0) {  
        break;  
    }  
}  
console.log(current);  
// → 21
```

Выход из цикла

```
var sum = 0;

while(true) {

    var value = +prompt( 'Введите число', '' );

    if( !value ) break; // (*)

    sum += value;

}

console.log( 'Сумма: ' + sum );
```

Следующая итерация: continue

```
for(var i = 0; i < 10; i++) {  
    if(i % 2 == 0) { continue; }  
    console.log(i);  
}
```

Wrong!

```
(i > 5) ? alert(i) : continue;
```

Следующая итерация: continue

```
outer: for(var i = 0; i < 3; i++) {  
    for(var j = 0; j < 3; j++) {  
        var input = prompt('Значение в координатах '+i+', '+j, '');  
  
        // если отмена ввода или пустая строка –  
        // завершить оба цикла  
        if (!input) break outer; // (*)  
    }  
}  
console.log('Готово!');
```

Задача 1

Что выведет эта программа?

```
var i = 3;  
  
while(i) {  
    console.log( i-- );  
}
```

Задача 2

При помощи цикла for выведите чётные числа от 2 до 10.

Задача 3

Напишите цикл, который за 7 вызовов console.log выводит такой треугольник:

```
'#'  
'##'  
'###'  
'####'  
'#####'  
'#####'  
'#####' '#####'
```


function
declaration

function name

argument

```
function makeRed(sender) {
```

```
  sender.style.color = 'red';
```

Функции

statement



function

```
function showMessage() {  
    alert( 'Привет всем присутствующим!' );  
}
```

function

```
function showMessage() {  
    alert( 'Привет всем присутствующим!' );  
}
```

```
showMessage();  
showMessage();
```

Локальные переменные

```
function showMessage() {  
    var message = 'Hello world!'; // local  
  
    console.log(message);  
}
```

```
showMessage(); // 'Hello world!'
```

```
console.log(message); // <-- Error
```

Локальные переменные

```
function count() {  
    // переменные i, j не будут уничтожены по окончании цикла  
    for(var i = 0; i < 3; i++) {  
        var j = i * 2;  
    }  
  
    console.log(i);  
    // i=3, последнее значение i, при нём цикл перестал работать  
    console.log(j);  
    // j=4, последнее значение j, которое вычислил цикл  
}
```

Локальные переменные

```
function count() {  
    var i, j;  
    for (i = 0; i < 3; i++) {  
        j = i * 2;  
    }  
  
    console.log(i); // i=3  
    console.log(j); // j=4  
}
```

Внешние переменные

```
var userName = 'Sergey';
```

```
function showMessage() {  
    var message = 'Hello, I am ' + userName;  
    console.log(message);  
}
```

```
showMessage(); // Hello, I am Sergey
```

Внешние переменные

```
var userName = 'Sergey';
```

```
function showMessage() {  
    userName = 'Dima';
```

```
    var message = 'Hello, I am ' + userName;  
    console.log(message);  
}
```

```
showMessage(); // Hello, I am Dima
```

```
console.log(userName); // Dima
```


Неявное объявление глобальных переменных!

```
function showMessage() {  
    message = 'Hello'; // без var!  
}
```

```
showMessage();
```

```
console.log(message); // Hello
```

Параметры

```
function showMessage(from, text) {  
    from = '** ' + from + ' **';  
    // здесь может быть сложный код оформления  
    console.log(from + ': ' + text);  
}  
  
showMessage( 'Маша', 'Привет!' );  
showMessage( 'Маша', 'Как дела?' );
```

Параметры

```
function showMessage(from, text) {  
    // меняем локальную переменную from  
    from = '**' + from + '**';  
    console.log(from + ': ' + text);  
}
```

```
var from = 'Маша';
```

```
showMessage(from, 'Привет');
```

```
console.log(from);  
// старое значение from без изменений,  
// в функции была изменена
```

Аргументы по умолчанию

```
function showMessage(from, text) {  
    if (text === undefined) {  
        text = 'текст не передан';  
    }  
  
    console.log(from + ': ' + text);  
}
```

```
showMessage('Маша', 'Привет!'); // Маша: Привет!  
showMessage('Маша'); // Маша: текст не передан
```

Аргументы по умолчанию

```
function showMessage(from, text) {  
    text = text || 'текст не передан';  
  
    console.log(from + ': ' + text);  
}
```

Возврат значения

```
function calcD(a, b, c) {  
    return b * b - 4 * a * c;  
}
```

```
var test = calcD(-4, 2, 1);  
console.log(test); // 20
```

Значение функции без return

```
function doNothing1() { /* empty */ }
```

```
function doNothing2() {  
    return;  
}
```

```
console.log(doNothing1()); //undefined  
console.log(doNothing2()); //undefined
```

Возврат значения

```
function showMovie(age) {  
    if (!checkAge(age)) {  
        return;  
    }  
  
    console.log( 'Фильм не для всех' );  
    // ...  
}
```


Выбираем имя

<code>showMessage(..)</code>	<i>// show, 'показывает'</i>
<code>getAge(..)</code>	<i>// get, 'получает' возраст</i>
<code>calcD(..)</code>	<i>// calc, 'вычисляет' дискриминант</i>
<code>createForm(..)</code>	<i>// create, 'создает' форму</i>
<code>checkPermission(..)</code>	<i>// check, 'проверяет' разрешение,</i>
<i>возвращает true/false</i>	

Задача 1

Написать функцию которая возвращает сумму 2х чисел.

```
console.log(summ(2, 5)); //7
```

Задача 2

Напишите функцию min(a,b), которая возвращает меньшее из чисел a,b.

```
console.log(min(2, 5)); //2  
console.log(min(3, -1)); //-1
```

```
x.html x app.js
<!DOCTYPE html>
<html
  type="html"
  lang="en">
</html>
<title></title>
<meta charset="utf-8">
<meta name="description" content="">
<meta http-equiv="X-UA-Compatible" content="IE=Edge">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="app.css">
</link>
</head>
<body>
  <main class="container-fluid ui-view">
    <div class="row">
      <div class="col-xs-12">
        <h1>Hello World!</h1>
        <a href="" ui-sref="app.example">Example state</a>
      </div>
    </div>
  </main>
</body>
<script src="app.min.js"></script>
<!-- <script src="registration.worker.js"></script> -->
</script>
</html>

1 import angular from 'angular';
2 import 'angular-ui-router';
3
4 import filtersModule from '../common/filters';
5
6 import exampleModule from '../example';
7
8 const MODULE_NAME = 'app';
9
10 angular.module(MODULE_NAME, [
11   'ui.router',
12   filtersModule.name,
13   exampleModule.name
14 ])
15
16 .config([
17   '$stateProvider',
18   '$urlRouterProvider',
19   '$httpProvider',
20   AppStateConfig
21 ])
22
23 .constant('version', require('../package.json').version)
24
25 .run(['$rootScope', '$state', '$stateParams',
26   function($rootScope, $state, $stateParams) {
27     $rootScope.$state = $state;
28     $rootScope.$stateParams = $stateParams;
29
30     $rootScope.$on('$routeChangeError', function() {
31       console.log('failed to change routes', arguments);
32     });
33   }
34 ]);
```

Функциональные выражения

```
    '$stateProvider', '$urlRouterProvider') {
      // ...
    }, {
      // ...
    }
  </div>

45
46 angular.bootstrap(document.querySelector('html'), [MODULE_NAME]);
47
```

Функциональные выражения

```
function sayHi() {  
    console.log( 'Привет' );  
}
```

```
console.log(sayHi); // выведет код функции
```

Функциональные выражения

```
function sayHi() { // (1)  
    alert( 'Привет' );  
}
```

```
var func = sayHi; // (2)  
func(); // Привет // (3)
```

```
sayHi = null;  
sayHi(); // ошибка (4)
```

Function Expression

```
var sayHi = function(person) {  
    console.log( 'Привет, ' + person );  
};  
  
sayHi( 'Вася' );
```

Сравнение с Function Declaration

```
// Function Declaration  
function sum1(a, b) {  
    return a + b;  
}
```

```
// Function Expression  
var sum2 = function(a, b) {  
    return a + b;  
}
```

Сравнение с Function Declaration

```
sayHi( 'Вася' ); // Привет, Вася
```

```
function sayHi1(name) {  
    alert( 'Привет, ' + name );  
}
```

```
sayHi( 'Вася' ); // ошибка!
```

```
var sayHi2 = function(name) {  
    alert( 'Привет, ' + name );  
}
```


Условное объявление функции

```
var age = +prompt( 'Сколько вам лет?', 20 );
```

```
if ( age >= 18 ) {  
    function sayHi() {  
        console.log( 'Прошу вас!' );  
    }  
} else {  
    function sayHi() {  
        console.log( 'До 18 нельзя' );  
    }  
}
```

```
sayHi();
```

Условное объявление функции

```
var age = prompt( 'Сколько вам лет?' );
```

```
var sayHi;
```

```
if (age >= 18) {  
    sayHi = function() {  
        alert( 'Прощу Вас!' );  
    }  
} else {  
    sayHi = function() {  
        alert( 'До 18 нельзя' );  
    }  
}
```

```
sayHi();
```

Условное объявление функции

```
var age = prompt( 'Сколько вам лет?' );
```

```
var sayHi = (age >= 18) ?  
    function() { alert( 'Прощу Вас!' ); } :  
    function() { alert( 'До 18 нельзя' ); };
```

```
sayHi();
```

Анонимные функции

```
function ask(question, yes, no) {  
    if (confirm(question)) {  
        yes();  
    } else {  
        no();  
    }  
}  
  
ask(  
    'Вы согласны?',  
    function() { alert('Вы согласились. '); },  
    function() { alert('Вы отменили выполнение. '); }  
);
```



Рекурсия

Рекурсия

$$\text{pow}(x, n) = x * \text{pow}(x, n - 1)$$

1. $\text{pow}(2, 4) = 2 * \text{pow}(2, 3)$
2. $\text{pow}(2, 3) = 2 * \text{pow}(2, 2)$
3. $\text{pow}(2, 2) = 2 * \text{pow}(2, 1)$
4. $\text{pow}(2, 1) = 2$

Рекурсия

```
function pow(x, n) {  
    if(n != 1) {  
        return x * pow(x, n - 1);  
    } else {  
        return x;  
    }  
}
```

```
console.log(pow(2, 3)); // 8
```

Контекст выполнения, стек

```
function pow(x, n) {  
    if(n != 1) {  
        return x * pow(x, n - 1);  
    } else {  
        return x;  
    }  
}
```

```
console.log(pow(2, 3));
```

Контекст: { x: 2, n: 3, line 3 }

Контекст: { x: 2, n: 2, line 3 }

Контекст: { x: 2, n: 1, line 1 }

Задача 1

Напишите функцию `sumTo(n)`, которая для данного `n` вычисляет сумму чисел от 1 до `n`

$$\text{sumTo}(3) = 3 + 2 + 1 = 6$$

$$\text{sumTo}(4) = 4 + 3 + 2 + 1 = 10$$

- Рекурсия
- Цикл
- Формула арифметической прогрессии

Задача 2

Написать функцию вычисляющую факториал числа n

$$n! = n * (n - 1) * (n - 2) * \dots * 1$$

Именованные функциональные выражения

Function

- Named Function Expression
 - `var add = function add() {...};`
 - Incompatible with IE
- Unnamed Function Expression
 - `var add = function () {...};`
 - Function Expression
 - Anonymous Function

Named Function Expression

// обычное функциональное выражение

```
var f1 = function(a) { /* тело функции */ };
```

// именованное функциональное выражение

```
var f2 = function sayHi(a) { /* тело функции */ };
```

Named Function Expression

```
var f = function sayHi(name) {  
    // изнутри функции – видно (выведет код функции)  
    console.log(sayHi);  
};  
  
// снаружи – не видно (ошибка: undefined variable 'sayHi')  
console.log(sayHi);  
  
var test = function sayHi(name) {  
    sayHi = 'test'; // попытка перезаписи  
    console.log(sayHi); // function...  
};  
  
test();
```

Named Function Expression

```
function f(n) {  
    return n ? n * f(n - 1) : 1;  
}
```

```
var g = f;  
f = null;
```

```
console.log( g(5) ); // error
```

Named Function Expression

```
var f = function factorial(n) {  
    return n ? n*factorial(n-1) : 1;  
};
```

```
var g = f;  
f = null;
```

```
console.log( g(5) ); // 120
```

Задача

Что выведет этот код?

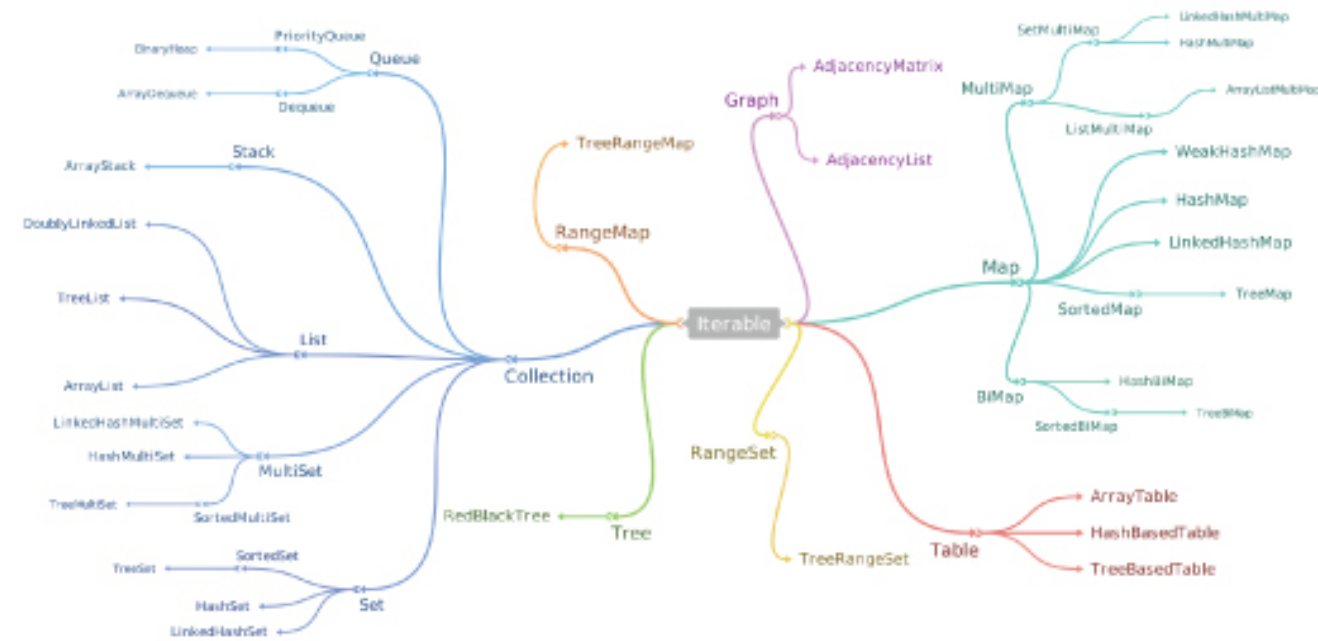
```
function g() { return 1; }  
console.log(g);
```

А этот?

```
(function g() { return 1; });  
console.log(g);
```


Структуры данных Методы и свойства

Data Structures for TypeScript and JavaScript



Problem

In JavaScript & TypeScript the only data structures provided are arrays, and string key hash maps. This limits the ability of developers to solve problems; where other data structures would work better with less code and more efficiency. Existing solutions include some of the following pitfalls:

- ⊖ Limited in scope
- ⊖ No static typings for TypeScript
- ⊖ Incompatible APIs
- ⊖ Lack modularity
- ⊖ Lack separation of interface and implementation
- ⊖ Doesn't take advantage of the upcoming features of JavaScript

Solution

Create a comprehensive library of well-researched data structures which will **increase the productivity** of JavaScript and TypeScript developers.

Release the library as **open source with permissive licensing**, and accept public contributions as well.

Results

Success was gauged by:

- ⊖ Published public modules containing the completed library
- ⊖ **Extensive test coverage**
- ⊖ Surveys conducted on various online development communities in order to determine the utility and quality of the library.
- ⊖ **Documentation for public APIs**

СВОЙСТВО str.length

```
console.log('Hello world'.length); //11
```

```
var text = 'Hello, IT World!';  
console.log(text.length); //15
```

Методы toUpperCase и toLowerCase

```
var text = 'Hello IT';
```

```
console.log(text.toUpperCase()); //HELLO IT  
console.log(text.toLowerCase()); //hello it
```

Метод num.toFixed(n)


```
var n = 65.432;
```

```
console.log( n.toFixed(2) ); // "65.43"  
console.log( n.toFixed(0) ); // "65"  
console.log( n.toFixed(5) ); // "65.43200"
```

Внимание!

```
console.log( 12.34.toFixed(1) ); // 12.3  
console.log(12.toFixed(1)); // ошибка!  
console.log( 12..toFixed(1) ); // 12.0
```



codewars 



Achieve code mastery through challenge

Codewars

Пробуем

www.codewars.com/kata/

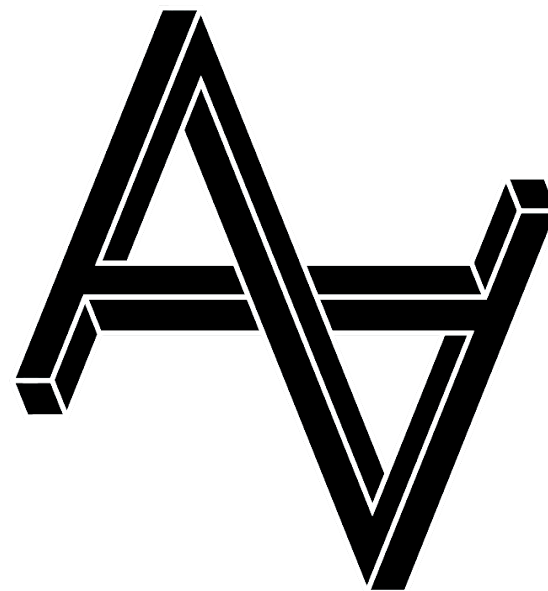
[even-or-odd](#)

[jennys-secret-message](#)

[return-negative](#)

[opposites-attract](#)

[convert-boolean-values-to-strings-yes-or-no](#)



2016