

# Типы данных. Продолжение

Чистяков Денис

В прошлой лекции мы обсудили:

- Строки, массивы, объекты и функции
- Отличия примитивных типов данных от сложных
- Основные методы для работы со строками и массивами

# Содержание

- Объекты и методы
- Нестрогое сравнение
- Даты
- Регулярные выражения
- Math

# Методы объекта

```
// Объект с predetermined набором свойств
var tweet = {
  ...
  likes: 16,
  getLikes: function() {
    return this.likes;
  },
  setLikes: function(value) {
    this.likes = parseInt(value) || 0;
    return this;
  },
  getAuthor: function() {
    return this.user.screenName;
  }
};
```

# Методы объекта

```
tweet.getLikes(); // 16
```

```
tweet.setLikes(17) // { ... }  
    .getLikes();   // 17
```

# Методы объекта

```
// Объект с predetermined набором свойств
var tweet = {
  ...
  _likes: 16,
  get likes() {
    return this._likes;
  },
  set likes(value) {
    this._likes = parseInt(value) || 0;
  },
  getAuthor: function() {
    return this.user.screenName;
  }
};
```

# Методы объекта

```
tweet.likes; // 16
```

```
tweet.likes = 17;  
tweet.likes; // 17
```

# Обработка исключений

```
// Объект с predefined набором свойств
var tweet = {
  ...
  _likes: 16,
  get likes() {
    return this._likes;
  },
  set likes(value) {
    var likes = parseInt(value);

    if (isNaN(lives) || likes < 0) {
      throw new TypeError('Передано неверное значение');
    }

    this._likes = likes;
  },
  getAuthor: function() {
    return this.user.screenName;
  }
};
```



# Обработка исключений

```
try {  
    tweet.likes = 'foo';  
} catch (e) {  
    if (e instanceof TypeError) {  
        tweet.likes = 0;  
    }  
    console.error(e);  
}  
  
tweet.likes; // 0
```

# TypeError

```
// Имя типа ошибки  
e.name; // 'TypeError'
```

```
// Сообщение ошибки  
e.message; // 'Передано неверное значение'
```

```
// Стек вызовов  
e.stack;  
// TypeError: Передано неверное значение  
//      at Object.set likes [as likes] (<anonymous>:10:15)  
//      at <anonymous>:18:15
```

🐼 === 128055?

```
var panda = {  
  valueOf: function() {  
    return 128060;  
  },  
  toString: function() {  
    return '&#x1F43C;';  
  }  
}
```

```
var pigCode = 128055;
```

```
panda == pigCode; // ???
```

🐼 === 128055?

```
isPrimitive(panda); // false
```

```
isPrimitive(pigCode); // true
```

```
typeof panda.valueOf() === 'function'; // true
```

```
panda.valueOf() === pigCode; // false
```

```
128060 === 128055; // false
```

```
panda == pigCode; // false
```

Операция сравнения двух сложных типов вернет истину только в том случае, если внутренние ссылки обоих объектов ссылаются на один и тот же объект в памяти

```
{ } == { }; // false
```

🐼 === 128055?

```
var panda = {  
  valueOf: function() {  
    return 128060;  
  },  
  toString: function() {  
    return '&#x1F43C;';  
  }  
}
```

```
var pandaCode = 128060
```

```
panda == pandaCode; // true
```

# Приведение объекта к числу

```
var panda = {  
  valueOf: function() { return 128060; },  
  toString: function() { return '#x1F43C'; }  
}
```

```
Number(panda); // 128060
```

```
+panda; // 128060
```

```
~~panda; // 128060
```

```
panda == 128060; // true
```

```
panda === 128060; // false
```

```
parseInt(panda); // NaN
```

```
parseFloat(panda); // NaN
```

```
panda.valueOf(); // 128060
```

# Приведение объекта к строке

```
var panda = {  
  valueOf: function() { return 128060; },  
  toString: function() { return '&#x1F43C;'; }  
}
```

```
String(panda); // '&#x1F43C;'
```

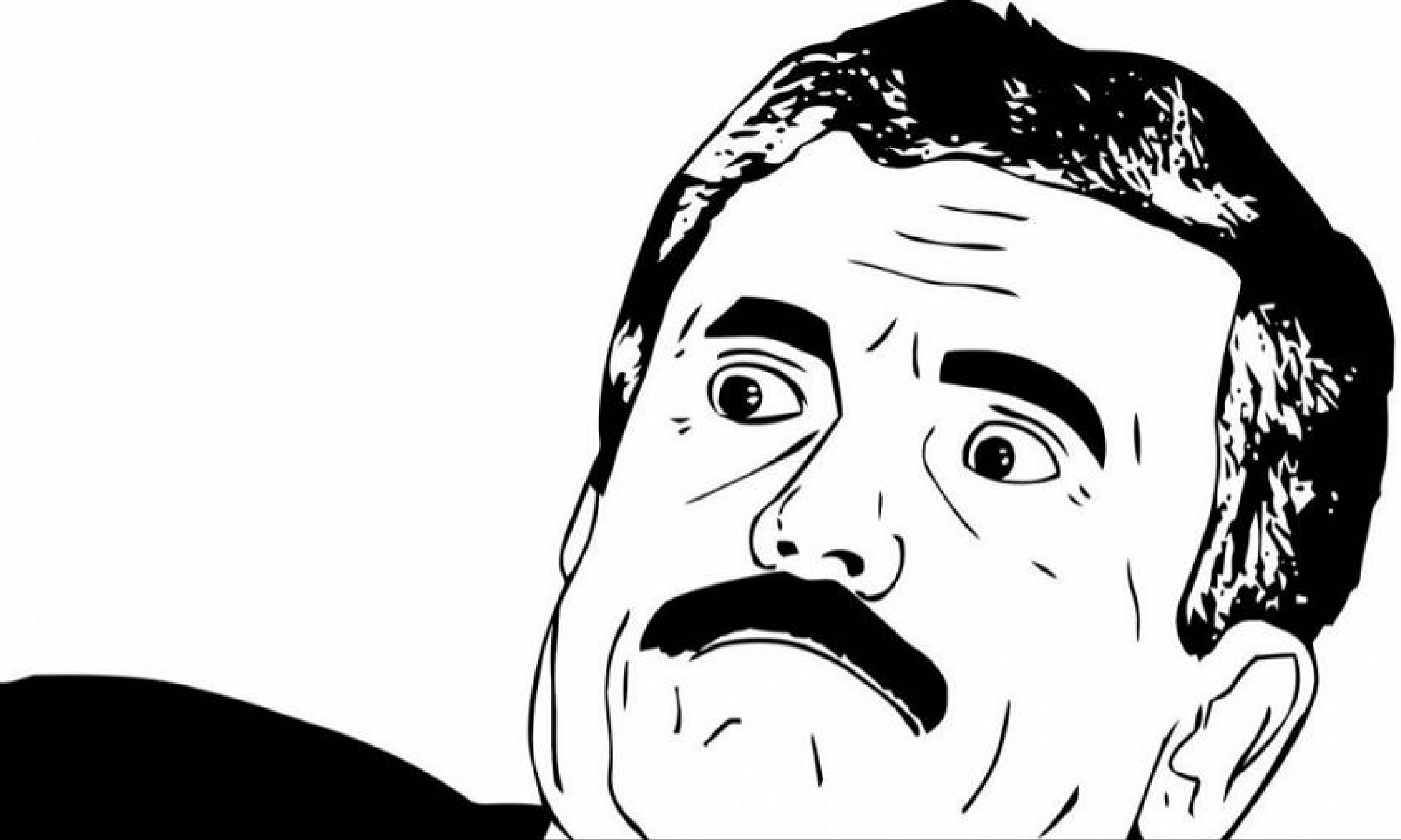
```
' ' + panda; // '128060'
```

```
panda == '128060'; // true
```

```
panda === '128060'; // false
```

```
panda.toString(); // '&#x1F43C;'
```





Не определяйте `valueOf` и `toString` одновременно. По возможности определяйте только `toString`. Его поведение более предсказуемо.

# Приведение объекта к строке

```
var panda = {  
  toString: function() { return '🐼'; }  
}
```

```
String(panda); // '🐼'
```

```
' ' + panda; // '🐼'
```

```
panda == '🐼'; // true
```

```
panda === '🐼'; // false
```

```
panda.toString(); // '🐼'
```

# Скрытые методы

```
var panda = {  
  valueOf: function() { return 128060; },  
  toString: function() { return '&#x1F43C;'; }  
}
```

```
Object.keys(panda); // ['valueOf', 'toString']
```

```
var emptyObject = {};
```

```
Object.keys(emptyObject); // []
```

```
typeof panda.valueOf === 'function'; // true
```

```
typeof emptyObject.valueOf === 'function'; // true
```

# Объявление методов объекта

```
var panda = {};
```

```
Object.defineProperty(panda, 'valueOf', {  
  value: function() {  
    return 128060;  
  },  
  writable: true,  
  enumerable: false,  
  configurable: true  
});
```

```
Object.defineProperty(panda, 'toString', {  
  value: function() {  
    return '&#x1F43C;';  
  },  
  writable: true,  
  enumerable: false,  
  configurable: true  
});
```

# Объявление методов объекта.writable

```
var tweet = {};  
  
Object.defineProperty(tweet, 'text', {  
  value: 'Я и IoT, пятый доклад на #wstdays в Питере',  
  writable: false  
})  
  
Object.getOwnPropertyDescriptor(tweet, 'text');  
// { value: 'Я и IoT, пятый доклад на #wstdays в Питере',  
//   writable: false,  
//   enumerable: false,  
//   configurable: false }  
  
tweet.text; // 'Я и IoT, пятый доклад на #wstdays в Питере'  
tweet.text = 'Вёрстка писем. Развенчиваем мифы. ... #wstdays';  
tweet.text; // 'Я и IoT, пятый доклад на #wstdays в Питере'
```

Значения параметров `writable`,  
`enumerable` и `configurable`  
по умолчанию — `false`.

# Объявление методов объекта. configurable

```
var tweet = {};  
  
Object.defineProperty(tweet, 'text', {  
    value: 'Я и IoT, пятый доклад на #wstdays в Питере',  
    configurable: false  
});  
  
Object.getOwnPropertyDescriptor(tweet, 'text');  
// { value: 'Я и IoT, пятый доклад на #wstdays в Питере',  
//   writable: false,  
//   enumerable: false,  
//   configurable: false }  
  
tweet.text; // 'Я и IoT, пятый доклад на #wstdays в Питере'  
tweet.hasOwnProperty('text'); // true  
delete tweet.text; // false  
tweet.text; // 'Я и IoT, пятый доклад на #wstdays в Питере'  
tweet.hasOwnProperty('text'); // true
```






THESE  
ARE THE  
COLD  
HARD  
FACTS...



FROSTBITE, THE  
FREEZING DEATH OF  
LIVING TISSUE, OCCURS  
AT TEMPERATURES  
BELOW 32 DEGREES  
FAHRENHEIT.

HOWEVER, THE  
HEART CAN ENDURE  
MUCH *COLDER*.

TO SAVE HIS BELOVED WIFE NORA FROM A  
TERMINAL MALADY, GOTHCORP SCIENTIST VICTOR  
FRIES EXPERIMENTED IN *CRYOGENICS* TO  
PRESERVE NORA UNTIL SHE MIGHT BE CURED.



BUT AN  
ACCIDENTAL  
PLUNGE IN HIS  
OWN *SUPER-  
COOLANT*  
CONCOCTIONS  
DID MORE THAN  
CHILL VICTOR  
TO THE BONE...

# Заморозка

```
var tweet = {  
  ...  
  likes: 16,  
  getLikes: function() {  
    return this.likes;  
  }  
};
```

```
Object.isFrozen(tweet); // false
```

```
Object.getOwnPropertyDescriptor(tweet, 'likes')  
// { value: 16,  
//   writable: true,  
//   enumerable: true,  
//   configurable: true }
```

# Заморозка

```
Object.freeze(tweet);
```

```
Object.isFrozen(tweet); // true
```

```
Object.getOwnPropertyDescriptor(tweet, 'likes')  
// { value: 16,  
//   writable: false,  
//   enumerable: true,  
//   configurable: false }
```

```
tweet.likes = 17;
```

```
tweet.likes; // 16
```

```
delete tweet.likes; // false
```

# Объект Даты

```
// Создает объект с текущей датой в системном часовом поясе
new Date(); // Mon Oct 17 2016 09:37:20 GMT+0500 (YEKT)

tweet.createdAt; // 'Sat Oct 01 12:01:08 +0000 2016'
// Пытаемся сконвертировать строку в дату
new Date(tweet.createdAt); // Sat Oct 01 2016 17:01:08 GMT+0500 (YEKT)

// Создаем дату из UNIX Timestamp
new Date(1475323268000); // Sat Oct 01 2016 17:01:08 GMT+0500 (YEKT)

// Создаем дату из набора параметров
new Date(2016, 9, 1, 17, 1, 8); // Sat Oct 01 2016 17:01:08 GMT+0500 (YEKT)

// Получаем UNIX Timestamp из даты
(new Date(2016, 9, 1, 17, 1, 8)).valueOf(); // 1475323268000

// Получаем текущее значение UNIX Timestamp
Date.now(); // 1476680054602
```

Date

# Регулярные выражения

## Имеют стандартный PCRE-синтаксис

PCRE (Perl Compatible Regular Expressions)

Руководство по регулярным выражениям



# Регулярные выражения

```
tweet.text; // 'Node.js, и модули, Джеймс о проблемах Node.js #nodejs #modules'  
  
// Проверяем содержится ли указанное регулярное выражение в строке  
/[a-z0-9]+/gi.test(tweet.text); // true
```

g — глобальное сопоставление

i — игнорирование регистра при сопоставлении

```
var tweetWithoutHashtag; // 'Я и IoT, пятый доклад на WSD в Питере'  
  
/[a-z0-9]+/gi.test(tweetWithoutHashtag); // false
```

# Регулярные выражения

```
var tweet = {  
  text: 'Node.js, и модули, Джеймс о проблемах Node.js #nodejs #modules #модули'  
};
```

```
Object.defineProperty(tweet, 'linkify', {  
  get: function() {  
    return this.text.replace(/#[a-z0-9]+/gi, '<a href="$1">$1</a>');  
  }  
});
```

```
Object.getOwnPropertyDescriptor(tweet, 'linkify');  
// { get: [Function: get],  
//   set: undefined,  
//   enumerable: false,  
//   configurable: false }
```

```
tweet.linkify;  
// 'Node.js, и модули, Джеймс о проблемах Node.js  
// <a href="$1">$1</a> <a href="$1">$1</a> #модули'
```



# Регулярные выражения

```
return this.text.replace(  
  /[a-z0-9a-я]+/gi,  
  '<a href="$1">$1</a>'  
);
```

```
tweet.linkify;  
// 'Node.js, и модули, Джеймс о проблемах Node.js'  
// <a href="$1">$1</a> <a href="$1">$1</a> <a href="$1">$1</a>
```

# Регулярные выражения

```
return this.text.replace(  
  /([a-z0-9a-я]+)/gi,  
  '<a href="$1">$1</a>'  
);
```

```
tweet.linkify;  
// 'Node.js, и модули, Джеймс о проблемах Node.js  
// <a href="#nodejs">#nodejs</a> <a href="#modules">#modules</a> <a href="#модули">#моду
```

# Регулярные выражения

```
return this.text.replace(  
  /([a-z0-9a-я]+)/gi,  
  '<a href="$2">$1</a>'  
);
```

```
tweet.linkify;  
// 'Node.js, и модули, Джеймс о проблемах Node.js'  
// <a href="nodejs">#nodejs</a> <a href="modules">#modules</a> <a href="модули">#модули</a>
```

# Регулярные выражения

```
return this.text.replace(  
  /(#([\w]+))/gi,  
  '<a href="$2">$1</a>'  
);
```

```
tweet.linkify;  
// 'Node.js, и модули, Джеймс о проблемах Node.js  
// <a href="nodejs">#nodejs</a> <a href="modules">#modules</a> #модули'
```

`\w` — соответствует любому цифробуквенному символу, включая нижнее подчеркивание.

Эквивалентен `[A-Za-z0-9_]`

Судя по всему, буквами они называют только латинский алфавит :)

# Math — библиотека математических функций и КОНСТАНТ

```
// Генерируем случайное число от 0 до 1  
Math.random(); // 0.4468546273336771
```

```
// Определяем меньшее из чисел  
Math.min(1, 5); // 1
```

```
// Определяем большее из чисел  
Math.max(1, 5, 10); // 10
```

# Math — библиотека математических функций и констант

```
// Округляем число до ближайшего целого  
Math.round(2.7); // 3  
Math.round(2.3); // 2
```

```
// Округляем число до целого в меньшую сторону  
Math.floor(2.7); // 2  
Math.floor(2.3); // 2
```

```
// Округляем число до целого в большую сторону  
Math.ceil(2.7); // 3  
Math.ceil(2.3); // 3
```

# Math — библиотека математических функций и КОНСТАНТ

```
// Возвращает натуральный (по основанию e) логарифм числа  
Math.log(10); // 2.302585092994046
```

```
// Возвращает основание, возведённое в степень  
Math.pow(2, 5); // 32
```

```
// Возвращает синус угла в радианах  
Math.sin(1); // 0.8414709848078965
```

```
// Возвращает тангенс угла в радианах  
Math.tan(1); // 1.5574077246549023
```