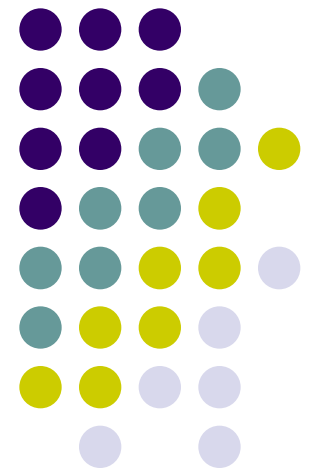
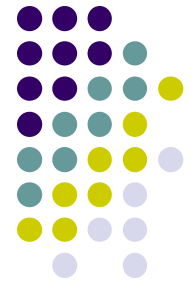


# Особенности Javascript

---





# "Javascript создан за 10 дней"

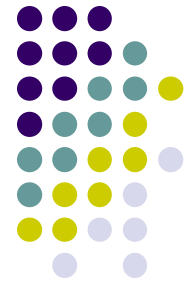
## (с) Brendan Eich

- ⑩ наследование через прототип
- ⑩ ошибки в языке
  - сохраняются для совместимости
- ⑩ и вообще, много странного...

*smiroshnick@gmail.com*

*0677222222*





*Sergey Miroshnikov*  
*smiroshnick@*  
*0677223233*

# ВСТРОЕННЫЕ ОБЪЕКТЫ



# Number

➤ все числа - double word ( 8 байт)

➤ НЕТОЧНЫЕ ВЫЧИСЛЕНИЯ

- `alert(0.1 + 0.2 == 0.3)`
  - `0.1.toFixed(20)`
  - Также в PHP, C, Ruby...

```
var i = 0
while(i!=10) {
  i += 0.2
}
```

➤ маскировка ошибок

- `1/0 = Infinity`
- `-1/0 = -Infinity`

⑩ битовые операции

- округление: `~~2.5` `2.5^0`
- проверка на -1: `if (~str.indexOf(..)) { match() }`



# Number

## ⑩ конвертация в число

- игнорируются начальные и конечные пробелы

```
+ "\n0,1\t" = 0.1
```

```
+ "\n\n" = 0
```

## ⑩ n.toString

- .toString(36) для random
- .toString(16) для цвета



# String

- кавычки работают одинаково

- сравнение unicode

`'a' > 'z'`

`'ë' > 'я'`

- unicode-последовательности `\u****`:

- `var a = '\u002a test \u002a' ( *test *)`



# Стиль кода

- ⑩ [Google JavaScript Style Guide](#)
- ⑩ [Closure Linter](#)

*Sergey Miroshnyk*  
*smiroshnick@gmail.com*  
*0677223233*



# Приведение типов

- ⑩ Арифметические операции приводят к числу

- `+ - * / > == ...`  $\longrightarrow$  `valueOf`

- ⑩ Строковые операции приводят к строке

- `alert, innerHTML`  $\longrightarrow$  `toString`

```
var foo = {  
  toString: function () {  
    return "foo"  
  },  
  valueOf: function () {  
    return 2  
  }  
}
```

```
alert(foo.toString() + "b")  
alert(foo + 1)  
alert(foo + "3")
```





# Приведение типов

- 10 Логические операции приводят к Boolean
- `&& || if`  $\longrightarrow$  `[[toBoolean]]`

Тип аргумента	Результат
<code>true/false</code>	Без преобразования
<code>undefined, null</code>	<code>false</code>
<code>Number</code>	<code>0, NaN == false *</code>
<code>String</code>	<code>"" == false *</code>
<code>Object</code>	<code>true</code>

\* - остальное `true`



# Приведение типов

## 10 Пример

```
[0] == 0  
"\n\n" == 0
```

```
if (0) => false
```

```
// HO
```

```
if ([0]) => true
```

```
if ("\n0\n") => true
```



# Приведение типов

## [[toBoolean]]

### 10 Явное преобразование к Boolean

- Неверное:

```
value = new Boolean(false)

if (value) {
  alert(1)
}
```

Все объекты - **true**

- Правильное:

`Boolean(value)` или `!!value`



# Без приведения

- ⑩ `obj1 == obj2`
  - только если это один и тот же объект
- ⑩ `x == undefined`
  - Сравнение с `undefined`, `null` прописано отдельно
- ⑩ `"str1" == "str2"`
  - Одинаковый тип - без приведения



# RegExp

## 10 multiline не влияет на точку

- `.*` не матчит `\n`

- Рабочий вариант:

```
text.replace(/\[u\](\[s\S]*?)\[\/u\]/gim, '<u>$1</u>')
```

*Sergey Miroshnyk*

*smiroshnick@gmail.com*

*0677223233*



# Циклы

## ➤ МЕТКИ

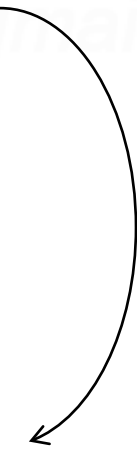
```
outer: for(...) {  
    ...  
    for (...) {  
        ...  
        break outer;  
        ...  
        continue outer;  
        ...  
    }  
}
```



# Циклы

- Метки на блоке

```
outer: {  
  
    for(var i=0; i<10; i++) {  
        break outer  
    }  
  
    alert("SKIP ME")  
}  
  
alert("HERE")
```



*Removed in ES 5 /Strict/*

# With



- создает область видимости из объекта

```
obj = {  
  size: {  
    width: 15,  
    height: 10  
  },  
  weight: 100  
}
```

...ОДИНАКОВО...



```
obj = {}  
obj.size = {}  
obj.size.width = 15  
obj.size.height = 10  
obj.weight = 100
```

```
with (obj) {  
  return weight / (size.width + size.height)  
}
```

```
with (obj.size) {  
  return width + height  
}
```

```
with (obj) {  
  with (size) {  
    return weight / (width + height)  
  }  
}
```

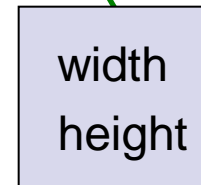




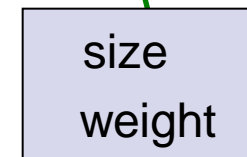
# With и область видимости

```
...  
with (obj) {  
  ...  
  with (size) {  
    ...  
    return weight / (width+height)  
  }  
  ...  
}  
...
```

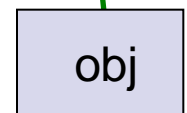
область size



область obj



window





# With != работе с объектом

```
obj = {  
  weight: 100  
}
```

weight	100
--------	-----

```
with (obj) {  
  weight = 777  
}
```

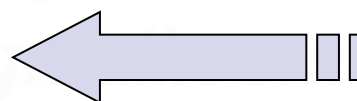
weight	777
--------	-----

```
with (obj) {  
  size = 10  
}
```

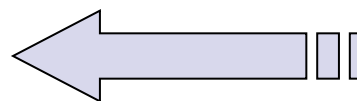
добавить переменную нельзя

weight	777
--------	-----

alert(window.size) => 10 — но можно нечаянно замусорить window



*weight* найден  
и изменен  
в области *obj*



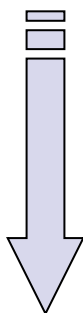
*size* найден  
и изменен  
в области *window*



## With: альтернатива

- оператор with редко используется
  - малоизвестен
  - не четко задано, откуда будет взята переменная
- есть альтернатива

```
with (obj.size) {  
    return width + height  
}
```



```
var s = obj.size  
return s.width + s.height
```

не так элегантно, зато

- меньше скобок
- гарантированно внутри объекта



# window.onerror

## 10 HTML5

- Firefox, IE, Chrome 10, Safari 5.1
- стек вручную
- Библиотека javascript-stacktrace



```
window.onerror = function(msg[, file, line]) {  
    $.post(msg, location, ...)  
}
```

Отлов JS-ошибок на prod

Недостатки: мусор в логах



# Переменные

- ⑩ На уровне всей функции
- ⑩ **var** обрабатывается при входе в функцию

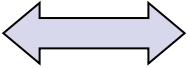
```
function getChar(event) {  
    if (event.which == null) {  
        chr = String.fromCharCode(event.keyCode);  
    } if (event.which != 0 && event.charCode != 0) {  
        var chr = String.fromCharCode(event.which);  
    }  
    return chr;  
}
```



# Блоки

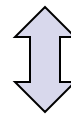
- блоки не создают отдельной области видимости

```
...  
{  
  var i = 0  
  ...  
}  
...
```



```
...  
var i = 0  
{  
  ...  
}  
...
```

```
for(var i=0; i<a.length; i++) {  
  ...  
}
```



```
var i  
for(i=0; i<a.length; i++) {  
  ...  
}
```



# Функции

## объявление

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

- доступны везде  
в области видимости

```
func() OK!
...
function func() {
  // ...
}
```

## выражение

```
var имя = function(параметры) {
  ...
}
```

```
var имя = new Function(параметры, '...')
```

- доступны только после  
объявления

```
func() Ошибка!
...
var func = function() {
  // ...
}
```

- *NFE*



# Функции

## 10 Аргументы

```
function func() {  
    for(var i=0; i<arguments.length;i++) {  
        ... arguments[i] ...  
    }  
}
```

```
function func() {  
    var args = arguments.slice(1)  
    var args = [].slice.call(arguments,1)  
}
```

```
(function(arg1, arg2) {  
    alert(arg1+arg2)  
    [].shift.apply(arguments);  
    alert(arg1+arg2)  
})(1, 2, 3)
```

*Fixed in ES 5*







# Функции

- именованные аргументы

```
function run(setup) {  
  var speed = setup.speed || 10  
  var distance = setup.distance || 50  
  if (!setup.smile) {  
    throw { name: "SmileRequired" }  
  }  
  return distance / speed  
}
```

```
var setup = {  
  speed : 5,  
  smile: {  
    size: 'large'  
    teeth: 'white'  
  },  
  field: makeField()  
}
```

Удобно, когда возможных  
комбинаций параметров много:

~~run(5, null, 10)  
run(null, null, 10)~~

run(setup) => 10  
setup.distance = 15  
run(setup) => 3

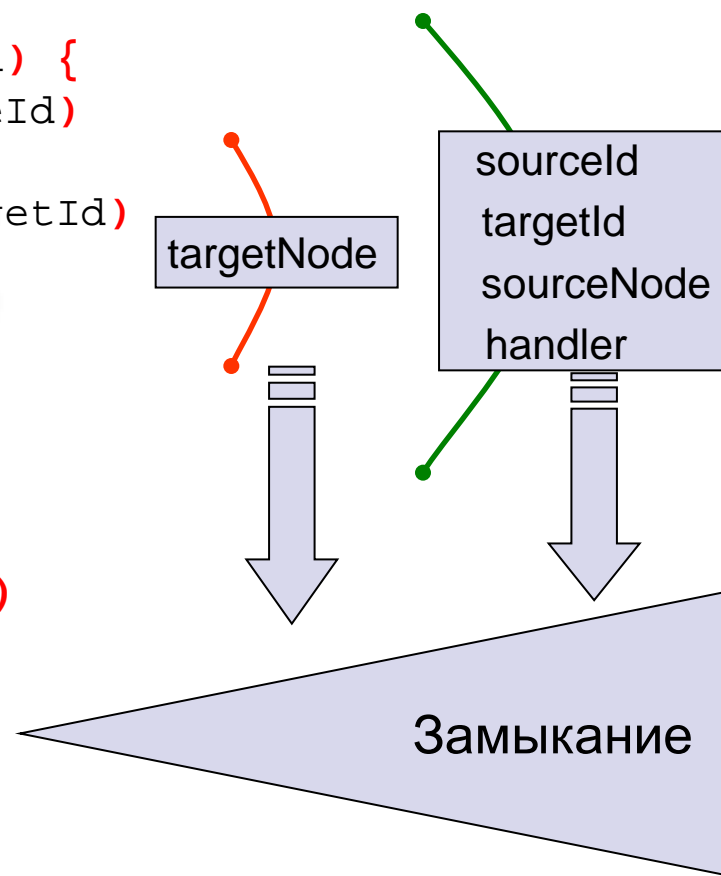


# Функции: замыкание

- каждая функция формирует свой контекст выполнения

```
function addHideHandler(sourceId, targetId) {  
  var sourceNode = d.getElementById(sourceId)  
  var handler = function() { // [[Scope]]  
    var targetNode = d.getElementById(targetId)  
    targetNode.style.display = "none"  
  }  
  sourceNode.onclick = handler  
  // sourceNode = null  
}
```

- addHideHandler("myButton", "myDiv")
- onclick запустится когда-нибудь потом
- onclick увидит targetId
- контекст выполнения существует,  
пока жива хоть одна внутренняя функция





# Функции: замыкание

## ➤ Ошибка в замыкании

```
function addEvents(divs) {  
    for(var i=0; i<10; i++) {  
        divs[i].innerHTML = i  
        divs[i].onclick = function(){ alert(i) }  
    }  
}
```

## ➤ Промежуточная функция

```
divs[i].onclick = (function(x) {  
    return function() { alert(x) }  
})(i)
```

- + Function Expression можно без скобок
- + можно также for (function

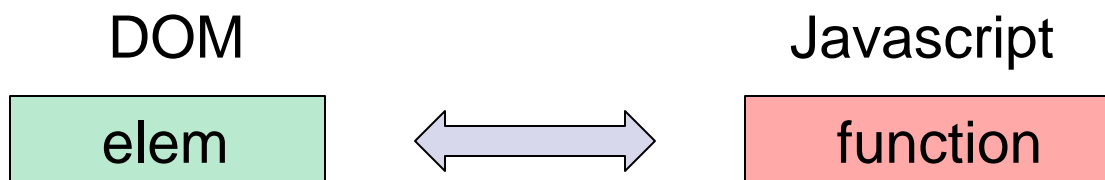


# Круговая утечка

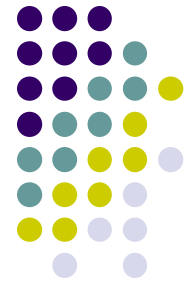
## Не очищается при уходе со страницы

- IE6 < 07.2007, IE7 (внутри страницы)

```
function() {  
  
    var elem = document.getElementById('id')  
    elem.onclick = function(e) {  
        // ...  
    }  
}
```

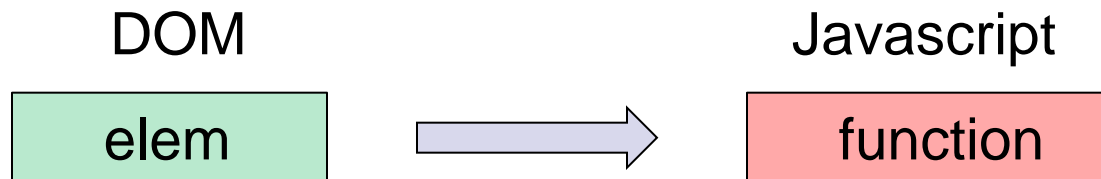


Утечки не в замыкании



# Круговая утечка

```
function() {  
    var elem = document.getElementById('id')  
    elem.onclick = function(e) {  
        // ...  
    }  
    elem = null  
}
```





# Утечка через лишние ссылки

```
function f() {  
  
    var xhr = $.ajax({ ... })  
    ...  
    elem.onclick = function(e) {  
        alert(1);  
    }  
}
```

весь ответ сервера  
останется в памяти...

... несмотря на то,  
что не нужен



# Контекст `this`

## ➤ 4 варианта вызова

- как функция

```
func(параметры)
```



*undefined in ES 5 /Strict/*

**this** = window

- как метод

```
obj.func(параметры)
```

```
obj["func"](параметры)
```

**this** = obj

- call/apply

```
func.apply(obj, [параметры])
```

```
func.call(obj, параметры)
```

⇔ **func(параметры)**

**this** = obj

- конструктор

```
new func(параметры)
```

**this** = НОВЫЙ ОБЪЕКТ



# Apply для функций над массивом

- ⑩ Наименьшее / наибольшее значение в массиве

```
var array = [1,2,3]
```

```
Math.max.apply(Math, array)
```

- ⑩ Так же - с любой функцией с переменным числом аргументов





# Ecma-262 5th

## 10 Новые фичи ES 5

<http://kangax.github.com/es5-compat-table/>

Современные браузеры => можно использовать.

<https://github.com/krisKowal/es5-shim>

Sergey Miroshnikov  
smiroshnick@gmail.com  
0677223233