



Блок 11. Введение в веб-разработку

11.2 — Основы JavaScript

Особенности архитектуры

- Динамическая типизация
- Слабая типизация
- Автоматическое управление памятью
- Прототипное программирование
- Функции являются объектами

Область применения JavaScript

- Настольные браузеры (Firefox, IE, Chrome, Opera...)
- Мобильные браузеры (Android Browser, Opera Mini...)
- Настольные приложения (Metro, Titanium, Adobe AIR...)
- Мобильные приложения (Phonegap, Tizen...)
- Серверные приложения (Node.js, Rhino...)
- NoSQL базы данных (MongoDB, CouchDB...)
- Операционные системы (Firefox OS, webOS...)

Преимущества JavaScript

- Полная интеграция с HTML/CSS
- Простые вещи делаются просто
- Поддерживается всеми распространёнными браузерами и включён по умолчанию

Что умеет JavaScript

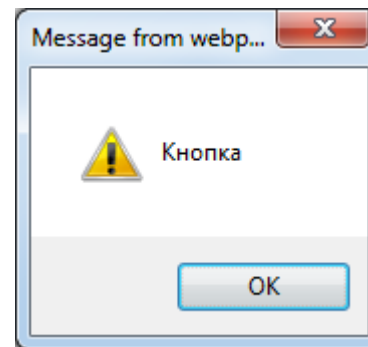
- Создавать новые HTML-теги, удалять существующие, менять стили элементов, прятать, показывать элементы и т.п.
- Реагировать на действия посетителя
- Посылать запросы на сервер и загружать данные без перезагрузки страницы
- Получать и устанавливать cookie, запрашивать данные, выводить сообщения...

Что не умеет JavaScript

- JavaScript не может читать/записывать произвольные файлы на жесткий диск, копировать их или вызывать программы.
- JavaScript не имеет прямого доступа к операционной системе.
- JavaScript, работающий в одной вкладке, почти не может общаться с другими вкладками и окнами.
- JavaScript не имеет универсальных механизмов сетевых взаимодействий (HTTP, WebSockets)
- JavaScript не может прочитать содержимое документов с других серверов

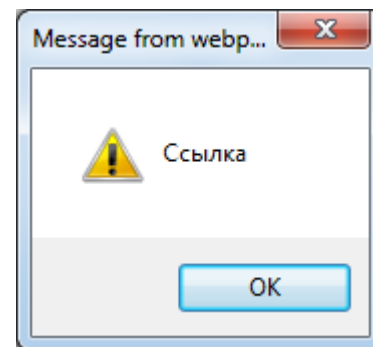
Примеры встраивания JavaScript. Обработчик события.

```
<html xmlns="http://www.w3.org/1999/xhtml">  
<head>  
  <title>Простая страница</title>  
</head>  
<body>  
  <button onclick="alert('Кнопка')">Кнопка</button>  
</body>  
</html>
```



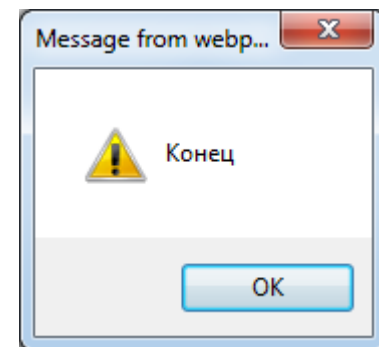
Примеры встраивания JavaScript. Ссылки.

```
<html xmlns="http://www.w3.org/1999/xhtml">  
<head>  
  <title>Простая страница</title>  
</head>  
<body>  
  <a href="javascript: alert('Ссылка')">Ссылка</a>  
</body>  
</html>
```



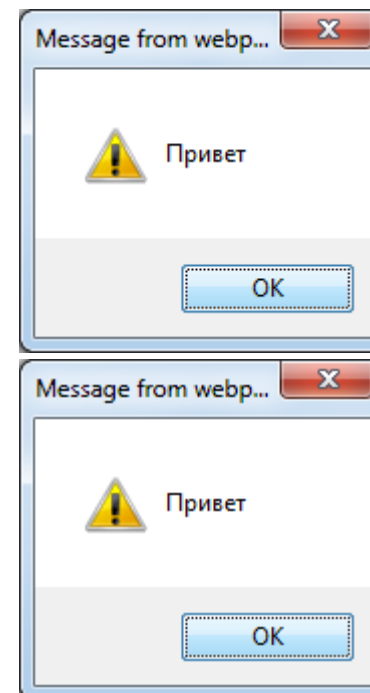
Примеры встраивания JavaScript. Тело документа.





```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>Простая страница</title>
  <script>
    alert("Начало");
  </script>
</head>
<body>
  <h1>Пример</h1>
  <script>
    alert("Конец");
  </script>
</body>
</html>
```

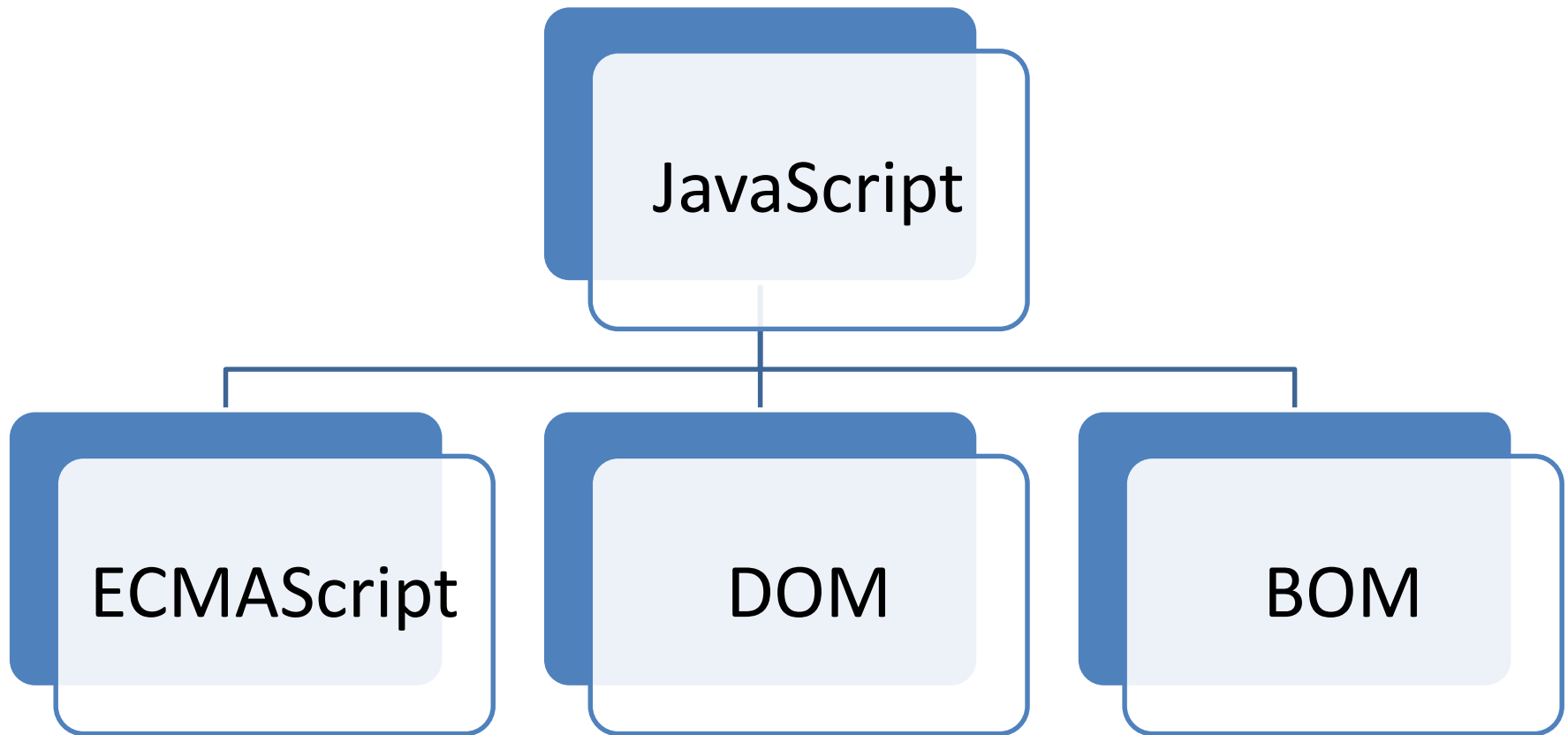


Примеры встраивания JavaScript. Файлы.

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>Простая страница</title>
  <script src="SimpleScript.js"></script>
</head>
<body>
  <h1>Пример</h1>
  <script src="SimpleScript.js"></script>
</body>
</html>
```



Name Path	Method	Status Text	Type	Initiator	Size Content	Time Latency	Timeline
 SimplePage.html /DOM	GET	200 OK	text/html	Other	802 B 323 B	26 ms 20 ms	
 SimpleScript.js /DOM	GET	304 Not Modified	applicat...	SimplePage.html:5 Parser	314 B 25 B	11 ms 4 ms	
2 requests 1.1 KB transferred 43 ms (onload: 1.58 s, DOMContentLoaded: 1.58 s)							



ECMAScript

Команды завершаются «;»

```
alert("Hello, world");      alert("Hello, world")  
alert("I say - \"HELLO\"!") alert("I can work without semicolon")
```

Комментарии

```
/*  
Как обычно, любое знакомство с языком программирования  
начинается с приветствия миру  
*/  
//Приветствие  
alert("Hello, world");
```

Идентификаторы

- могут содержать цифры, символы, \$ и _
- регистрозависимые
- не может начинаться с цифры
- может содержать символы Unicode

//Корректные имена

```
var aBcD1 = 5;  
var AbCd1 = 10;  
var $ = 15;  
var Не_делайте_так = 23;
```

//Некорректные имена

```
var 1aBcD = 5;  
var Ab-Cd1 = 10;
```

Зарезервированные слова

- ключевые слова

*break else new var case finally return void catch for switch while
continue function this with default if throw delete in try do
instanceof typeof*

- зарезервированные на будущее

*abstract enum int short boolean export interface static byte
extends long super char final native synchronized class float
package throws const goto private transient debugger
implements protected volatile double import public*

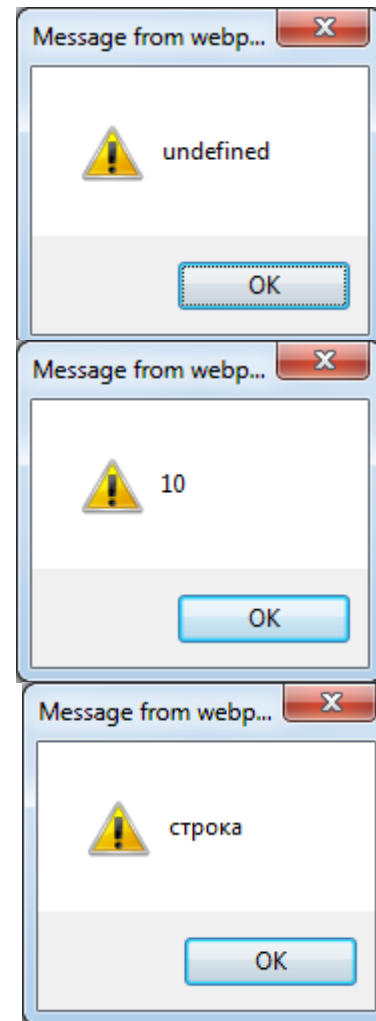
```
var class = 5;  
alert(class + 5);
```

ПЕРЕМЕННЫЕ

Переменные. Объявление

[var] *Переменная* [= Значение]

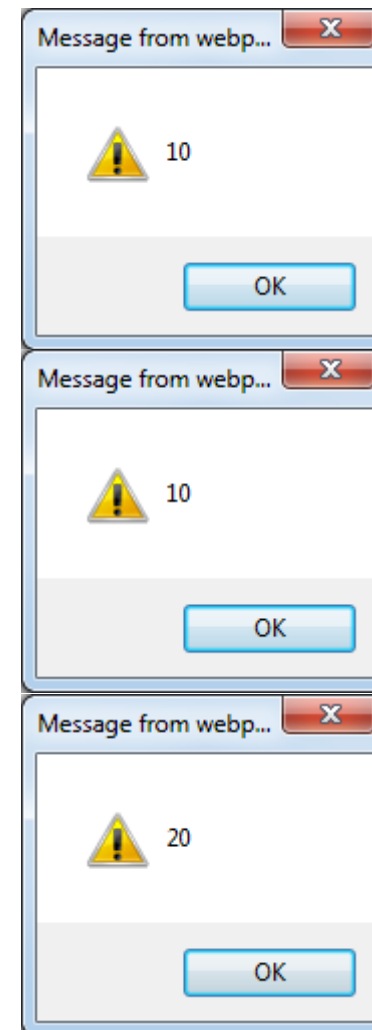
```
var temp;  
alert(temp);  
var temp = 10;  
alert(temp);  
temp = "строка";  
alert(temp);
```



Переменные. Область видимости.

- Область видимости – функция.

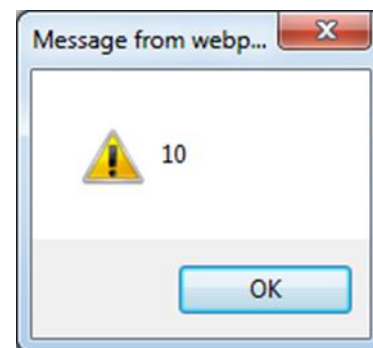
```
temp = 20;  
function Func() {  
    { var temp = 10; };  
    alert(temp);  
    function Func1() {  
        alert(temp);  
    }  
    Func1();  
}  
Func();  
alert(temp);
```



Переменные. Область видимости.

- Если внутри функции переменная объявляется без оператора `var`, то создается глобальное свойство.

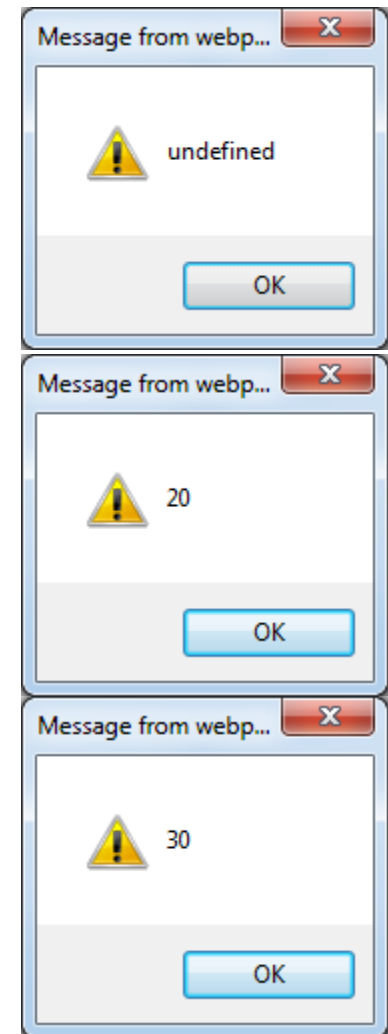
```
function Func() {  
    { temp = 10; };  
    alert(temp);  
}  
Func();  
alert(temp);
```



Переменные. Область видимости. Подъем.

- Внутри области видимости все объявленные в ней переменные.

```
temp = 30;  
function Func() {  
    alert(temp);  
    if (false) {  
        var temp = 10;  
    }  
    else {  
        temp = 20;  
    }  
    alert(temp);  
}  
Func();  
alert(temp);
```

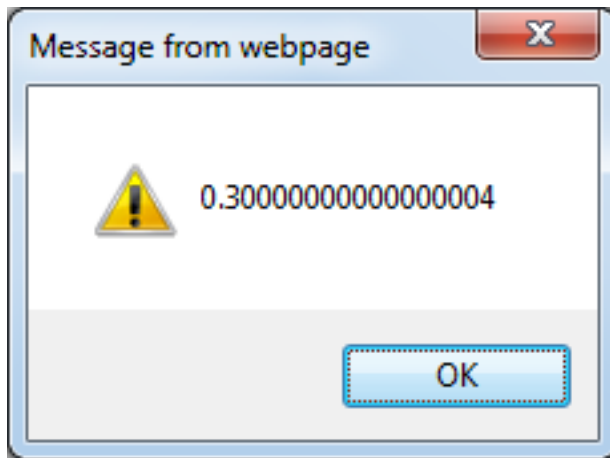


ОСНОВНЫЕ ТИПЫ ДАННЫХ

- Примитивные типы
 - Number
 - String
 - Null
 - Undefined
 - Boolean
- Object

- Number — вещественное число двойной точности формате IEEE-754

```
alert(0.1 + 0.2);
```



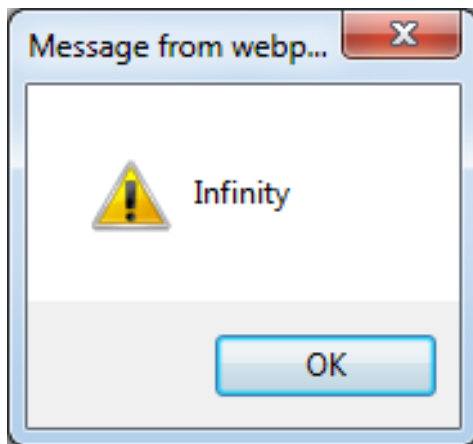
```
alert(0.1 + 0.2 == 0.3);
```



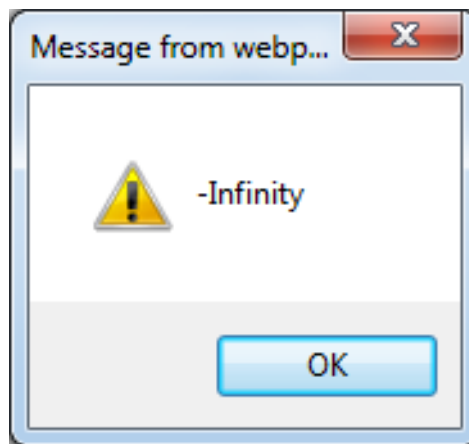
Специальные числовые значения

- Infinity
- -Infinity
- NaN

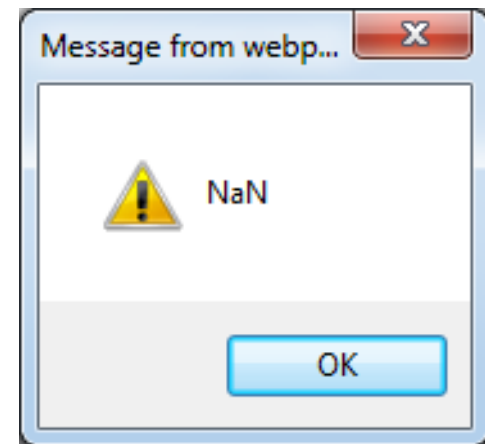
`alert(1/0);`



`alert(1 / -0);`

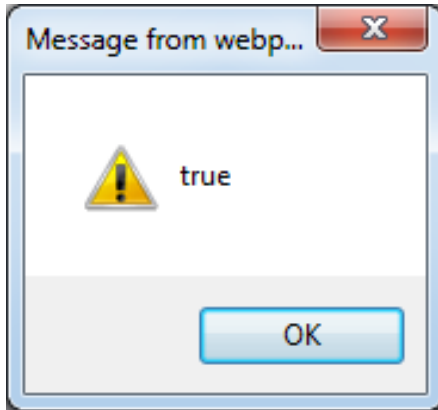


`alert(0/0);`

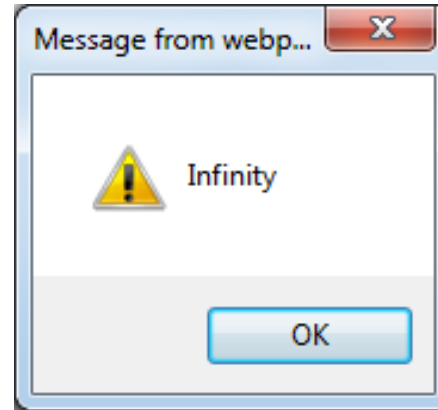


Работа с Infinity и -Infinity

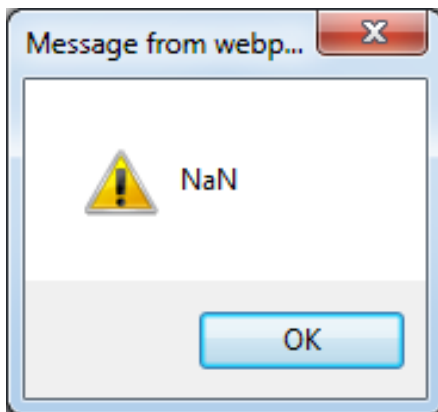
```
alert(Infinity == Infinity);
```



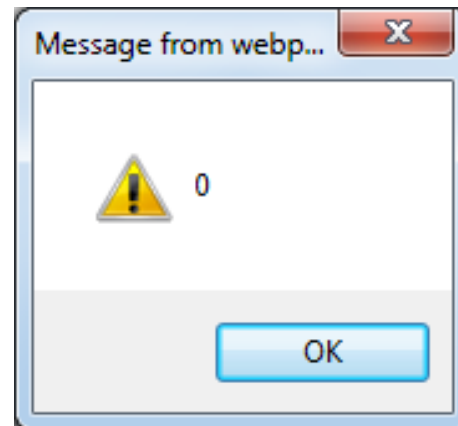
```
alert(Infinity + Infinity);
```



```
alert(Infinity-Infinity);
```



```
alert(1/Infinity);
```



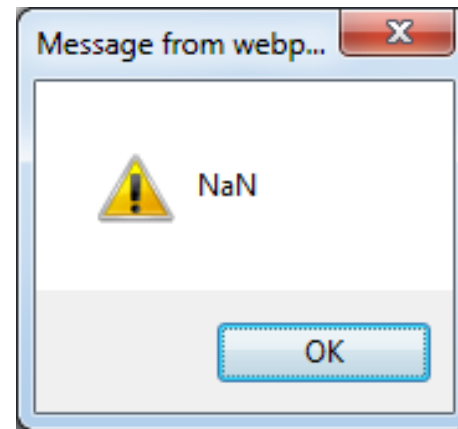
Типы данных. Number

- NaN — Not-a-Number, не число

```
alert(NaN == NaN);
```



```
alert(NaN + 1);
```



- Глобальные функции
 - `isFinite(n)` — проверяет значение на неравенство Infinity и NaN
 - `isNaN(n)` — проверяет значение на равенство NaN
- Методы
 - `toFixed(n)` — округляет число до точности n и возвращает результат в виде строки
 - `toPrecision(n)` — округляет до общего количества цифр вне зависимости: после запятой или нет.

- Математические методы

- Округление

- `Math.floor(x)` — возвращает наибольшее целое, меньшее или равное аргументу
 - `Math.ceil(x)` — возвращает наименьшее целое, большее или равное аргументу
 - `Math.round(x)` — округляет до ближайшего целого

- Тригонометрические функции

- `Math.sin(x)`, `Math.cos(x)`, `Math.acos(x)`, `Math.asin(x)` и т. д.

- Общие функции

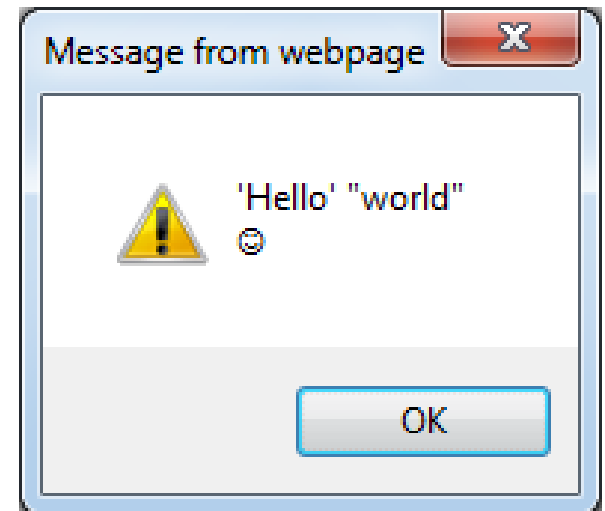
- `Math.sqrt(x)`, `Math.log(x)`, `Math.exp(x)` и т.д.
 - <http://es5.javascript.ru/x15.8.html#x15.8>

[http://msdn.microsoft.com/ru-ru/library/ie/b272f386\(v=vs.94\).aspx](http://msdn.microsoft.com/ru-ru/library/ie/b272f386(v=vs.94).aspx)

Типы данных. String.

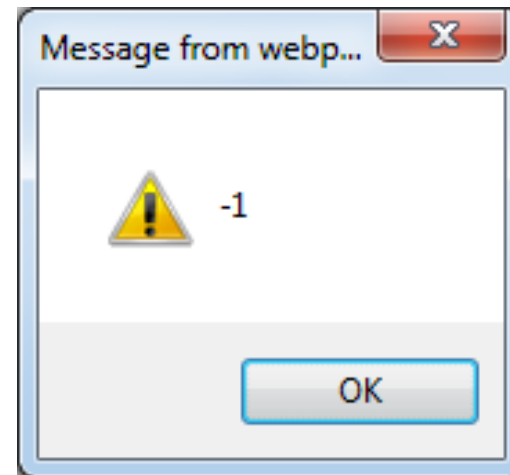
- String — представляет собой конечную упорядоченную последовательность нуля или более 16-битных беззнаковых целых значений (UTF-16).

```
var hello = "'Hello'";  
var world = '"world"';  
var smile = '\u263A';  
alert(hello + " " + world + '\n' + smile);
```



- Сравнение строк

```
alert("Пёс" > "Пять");    alert("Пёс".localeCompare("Пять"));
```

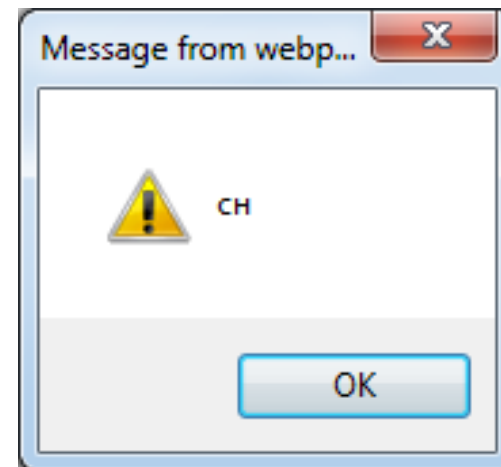


Типы данных. String

- Получение подстрок

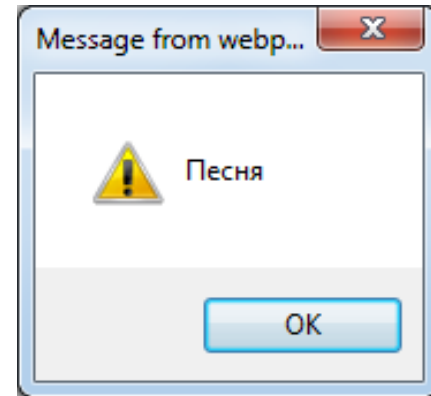
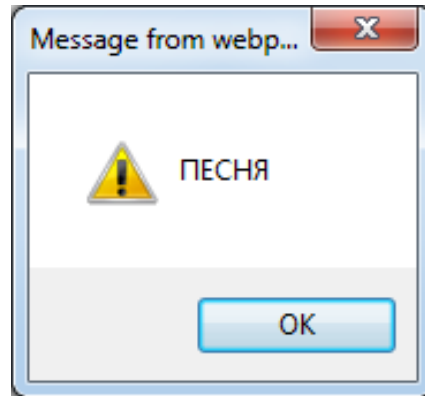
- `charOf(n)` — возвращает строку, содержащую символ, находящийся на позиции `n` в строке
- `substring(start [, end])` — возвращает подстроку с позиции `start` до, но не включая `end`.
- `substr(start [, length])` — возвращает подстроку с позиции `start`, длиной `length`
- `slice(start [, end])` — возвращает подстроку с позиции `start` до, но не включая `end`.

```
alert("Песня".substr(2, 2));  
alert("Песня".substring(2, 4));  
alert("Песня".slice(2, 4));  
alert("Песня".slice(2, -1));  
alert("Песня".substr(-3, 2));
```



- Изменение строк
 - `toLowerCase()` – меняет у всех символов строки регистр на нижний
 - `toUpperCase()` – меняет у всех символов строки регистр на верхний

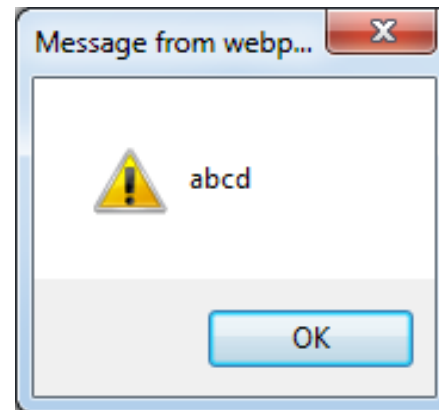
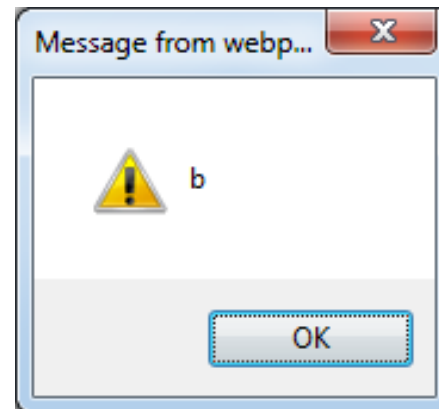
```
var song = "Песня";  
alert(song.toUpperCase());  
alert(song);
```



СТРОКИ НЕ ИЗМЕНЯЮТСЯ

Попытка модификации строки

```
var d = "abcd";  
alert(d[1]);  
d[1] = "s";  
alert(d);
```



Получение длины строки

```
var str = '\u263A' + "\n" + '☹';  
alert(str);  
alert(str.length);
```

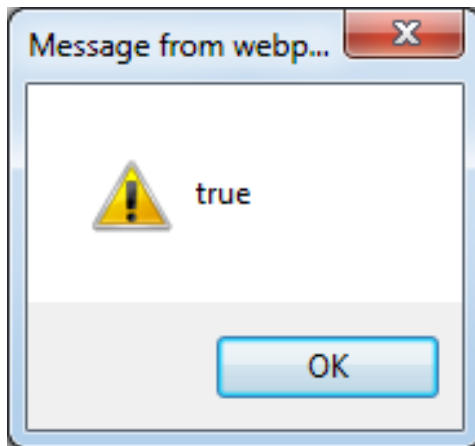


- Поиск подстроки
 - `indexOf(search [, Index])` — ищет первое вхождение `search` в строке начиная с `Index`
 - `lastIndexOf(search [, Index])` — ищет последнее вхождение `search` в строке начиная с `Index`
- Разбиение строки
 - `split(separator[, limit])` — возвращает `Array`, содержащий подстроки, разделённые `separator`, `limit` — максимальное число элементов

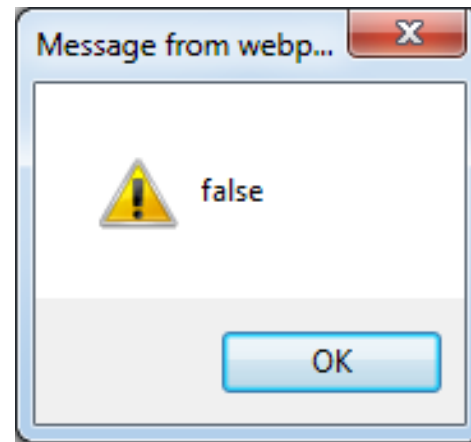
Специальные типы Undefined и Null

- Undefined — «не определено» — существует единственное значение — `undefined`.
Значением любой переменной, которой ещё не было присвоено значения, является `undefined`.
- Null — существует единственное значение — `null`.

```
alert(undefined == null);
```



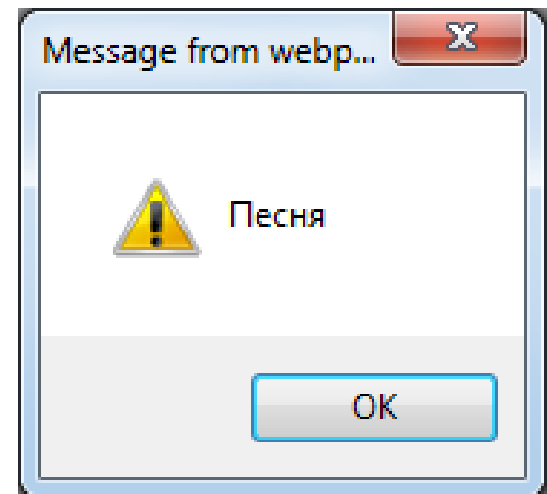
```
alert(undefined === null);
```



- Boolean — представляет собой логическую сущность, которая может принимать одно из двух значений: `true` или `false`.

- Object — («объект») представляет собой неупорядоченный набор свойств. Каждое свойство состоит из имени, значения и набора атрибутов. По сути является ассоциативным массивом.

```
var s = {};  
s.song = "Песня";  
alert(s.song);  
alert(s["song"]);
```



- Варианты объявления:

```
var object = { a : "a", "var" : "var", 'field for empty' : "" };
```

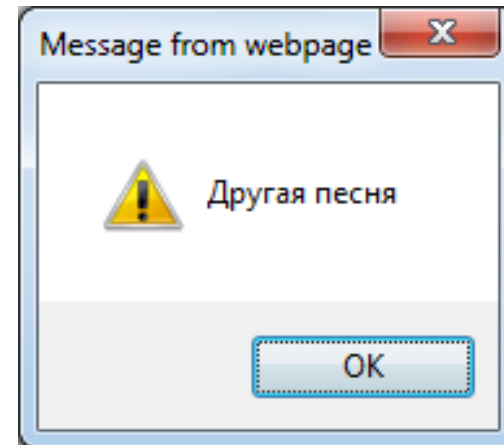
```
var object = new Object();
```

```
var object = new Object(10);
```

Типы данных. Object

- В отличие от примитивных типов объекты передаются по ссылке

```
var s = { a: "Песня" };  
var d = s;  
d.a = "Другая песня";  
alert(s.a);
```



ОПЕРАТОРЫ

Операторы.

Оператор	Назначение
-	Изменение знака на противоположный
+	Унарный +
!	Дополнение. Используется для реверсирования значения логических переменных
++	Увеличение значения переменной. Может применяться как префикс переменной или как ее суффикс
--	Уменьшение значения переменной. Может применяться как префикс переменной или как ее суффикс

Операторы. Арифметические операторы

Оператор	Назначение
-	Вычитание
+	Сложение
*	Умножение
/	Деление
%	Вычисление остатка от деления

Операторы. Арифметические операторы

- Если при сложении хоть один оператор строка, то результат тоже будет строка.

```
10 + 10 + "10" - 1 = 2009  
"10" + 10 + 10 - 1 = 101009
```

Операторы. Битовые операторы

Оператор	Логическая операция
&	И
	или
^	ИСКЛЮЧАЮЩЕЕ ИЛИ
~	НЕ
>>	Сдвиг в правую сторону
<<	Сдвиг в левую сторону
>>>	Сдвиг в правую сторону с заполнением освобождаемых разрядов нулями

Операторы. Операторы сравнения

Оператор	Условие
>	Больше
>=	Больше или равно
<	Меньше
<=	Меньше или равно
==	Равно
!=	Не равно
===	Идентично
!==	Не идентично

Операторы. Логические операторы.

Оператор	Описание
	Оператор ИЛИ.
&&	Оператор И.

- Логические операторы применяются не только к логическим переменным

10 && 15 = 15

15 && 10 = 10

10 && 0 = 0

10 || 15 = 10

15 || 10 = 15

10 || 0 = 10

"строка1" && "строка2" = "строка2"

"строка1" && 10 = 10

10 && null = null

10 || false = 10

null || "строка2" = "строка2"

"строка2" || 10 = "строка2"

Операторы. Операторы присваивания.

Оператор	Описание
=	Простое присваивание
+=	Увеличение численного значения или слияние строк
-=	Уменьшение численного значения
*=	Умножение
/=	Деление
%=	Вычисление остатка от деления
>>=	Сдвиг вправо
>>>=	Сдвиг вправо с заполнением освобождаемых разрядов нулями
<<=	Сдвиг влево
=	ИЛИ
&=	И
^=	ИСКЛЮЧАЮЩЕЕ ИЛИ

ПРЕОБРАЗОВАНИЕ ТИПОВ

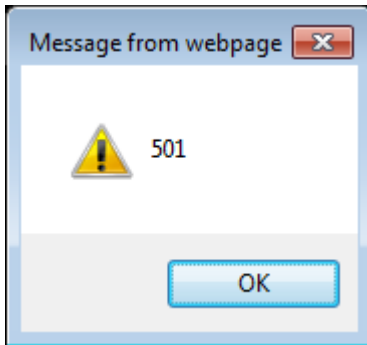
Преобразование типов

- Строковое преобразование
- Числовое преобразование
- Преобразование к логическому типу
- Преобразование к объекту

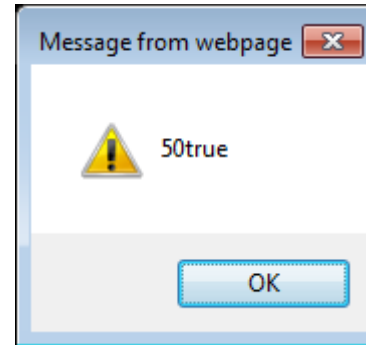
Строковое преобразование

Неявное преобразование

```
alert("50" + 1);
```

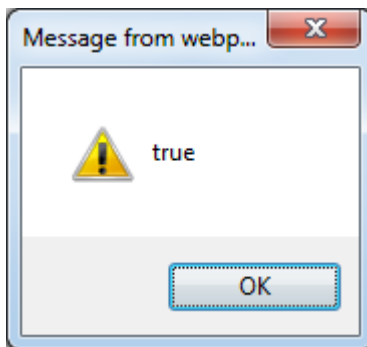


```
alert("50" + true);
```

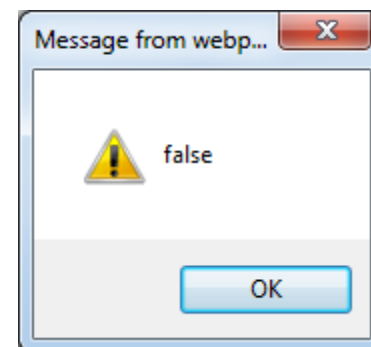


Явное преобразование

```
var value1 = new String(10);  
alert(value1 == "10");
```



```
alert(value1 === "10");
```



Строковое преобразование

Значение	Результат
undefined	"undefined"
null	"null"
true	"true"
false	"false"
1(конечное, ненулевое)	"1"
NaN	"NaN"
Infinity	"Infinity"
{}(любой объект)	toString() или valueOf() и преобразование к строке

Числовое преобразование

- Неявное преобразование

```
var value1 = "10";           var value1 = +"10";  
alert(value1 === 10);
```



- Явное преобразование

```
var value1 = new Number("10");
```

«Мягкое» преобразование к числу

- `parseInt(строка[,основание])` — преобразует строку в целое число по указанному основанию, а если это невозможно, возвращает NaN.
- `parseFloat(строка)` — преобразует строковый аргумент в число с плавающей точкой.

```
parseInt("10")      = 10
parseInt("010")     = 8
parseInt("0x10")    = 16
parseInt("10", 2)   = 2
parseInt("10px")    = 10
parseInt("10.10")   = 10
parseInt(" 10 ")    = 10
parseInt("px10")    = NaN
```

```
parseFloat("10.1")  = 10.1
parseFloat("10,1")  = 10
parseFloat("101E-1") = 10.1
parseFloat("10.10.10") = 10.1
parseFloat("10.1px") = 10.1
parseFloat("0.101e+2") = 10.1
parseFloat(" 10.1 ") = 10.1
parseFloat("px10.1") = NaN
```

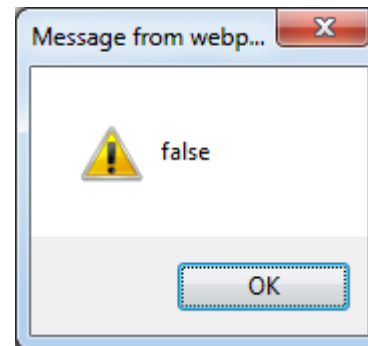
Числовое преобразование

Значение	Результат
undefined	NaN
null	0
true	1
false	0
"" (пустая строка)	0
"1.1"	1.1
"строка"	NaN
{}(любой объект)	valueOf() или toString() и преобразование к числу

Преобразование к логическому типу

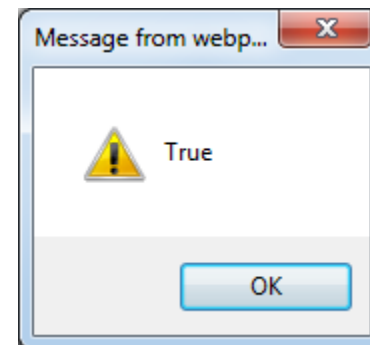
- Неявное преобразование

```
alert(!"false");
```



- Явное преобразование

```
var value = new Boolean(false);  
if (value)  
{ alert("True") }  
else  
{ alert("False") }
```



Преобразование к логическому типу

Значение	Результат
undefined	false
null	false
0	false
NaN	false
Infinity	true
1(конечное, ненулевое)	true
""(пустая строка)	false
"1.1"	true
"строка"	true
{}(любой объект)	true

Преобразование к объекту

Значение	Результат
undefined	Исключение
null	Исключение
0	new Number(0)
NaN	new Number(NaN)
Infinity	new Number(Infinity)
1(конечное, ненулевое)	new Number(1)
""(пустая строка)	new String("")
"1.1"	new String("1.1")
"строка"	new String("строка")
true	new Boolean(true)
false	new Boolean(false)

Стандартные шаблоны приведения типов

- Преобразование к строке

```
var str = "" + value;
```

- Преобразование к числу

```
var number = +value;
```

- Преобразование к булеву значению

```
var bool = !!value;
```

Курьёзы приведений типов

```
null == 0      – false  
null > 0       – false  
null >= 0      – true
```

```
"" == 0        – true  
"0" == 0       – true  
"0" == ""      – false
```

```
"10" == 10     – true  
"010" == 10    – true  
"10" == "010" – false
```

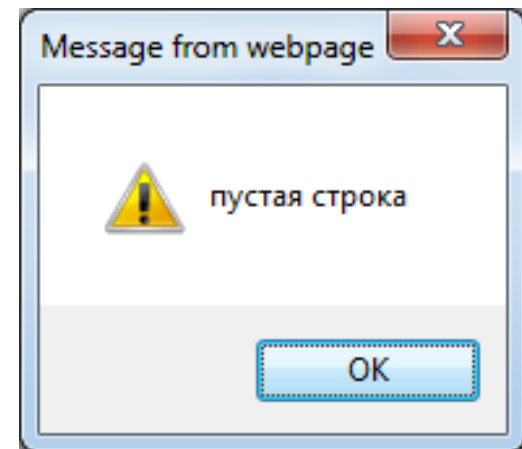
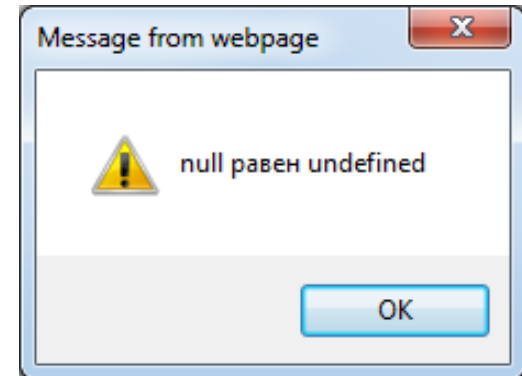
ИНСТРУКЦИИ

Инструкции. Условные инструкции.

if (*Выражение*) *Инструкция*
[else *Инструкция*]

```
if (null == undefined) {  
    alert("null равен undefined")  
}
```

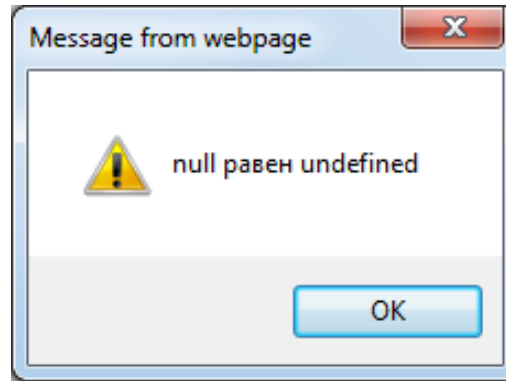
```
if ("") {  
    alert("не пустая строка")  
}  
else {  
    alert("пустая строка")  
}
```



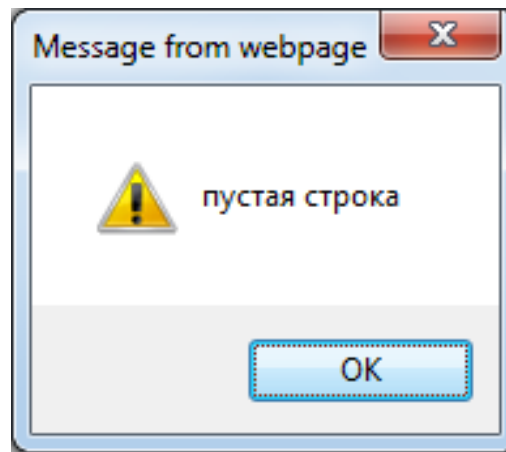
Инструкции. Альтернатива условной инструкции

- *(Условие) ? Результат : Альтернатива*

```
alert((null == undefined)? "null равен undefined" : "");
```



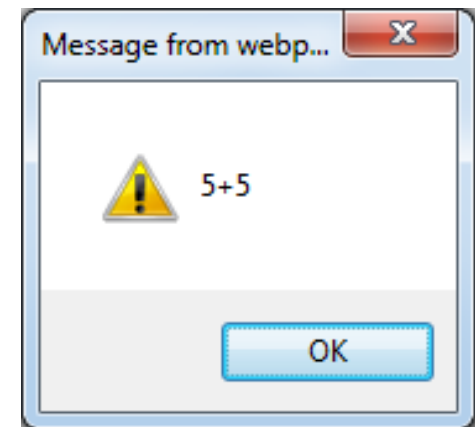
```
alert(("")? "не пустая строка" : "пустая строка");
```



Инструкции. Инструкция выбора.

```
switch (Выражение){  
    case Вариант1: Инструкции  
  
    ..... •  
    [case ВариантN: Инструкции]  
    [default : Инструкции]  
}
```

```
var a = 10;  
switch (a) {  
    case "10":alert("строка"); break;  
    case 5 + 5: alert("5 + 5");break;  
    case 10: alert("10"); break;  
    default: alert("Нет значения");  
}
```



Инструкции. Циклы.

do *Инструкция*

while (*Выражение*);

while (*Выражение*)

Инструкция

for ([*Инициализация*]; [*Условие*] ; [*Шаг*])

Инструкция

for (**var** *Ключ* **in** *Выражение*)

Инструкция

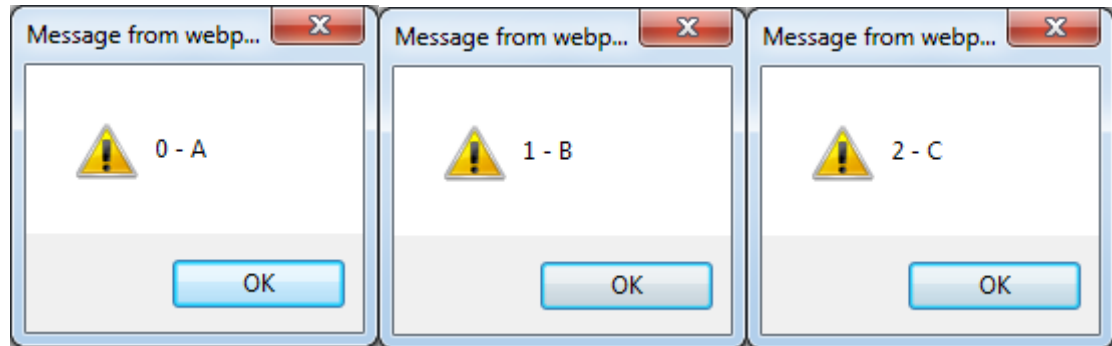
Инструкции. Циклы. for

```
for ( var i = 0, j = 3; i < 3; i++, j-- )  
{  
    alert(i + " " + j);  
}
```

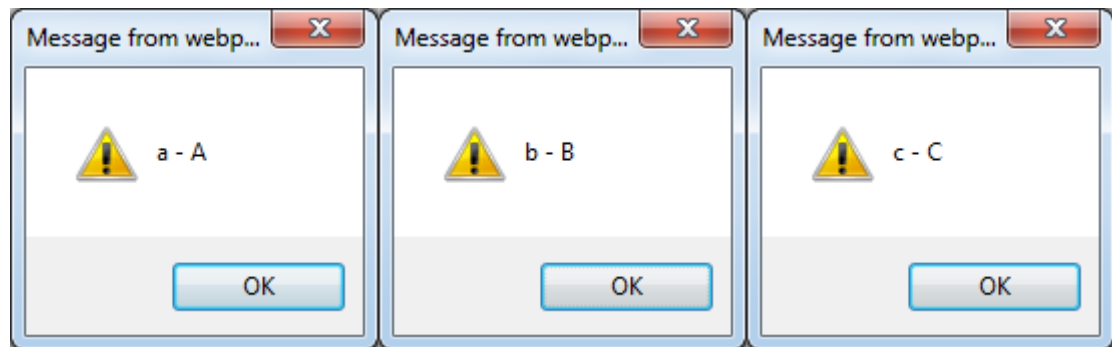


Инструкции. Циклы. for-in

```
var a = "ABC";  
for (var ch in a) {  
    alert(ch + " - " + a[ch]);  
}
```



```
var a={a:"A",b:"B",c:"C"};  
for (var ch in a) {  
    alert(ch + " - " + a[ch]);  
}
```

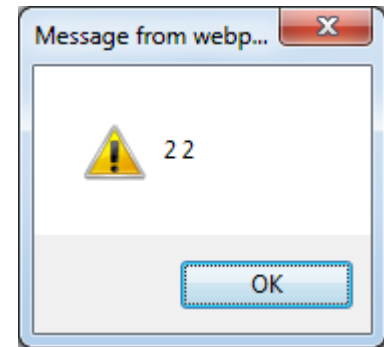


Инструкции. Операторы безусловного перехода

Инструкция	Описание
<code>break[метка]</code>	Завершает текущий цикл или конструкции <code>switch</code> и передает управление на следующий вызов
<code>continue[метка]</code>	Прекращает текущую итерацию цикла и продолжает выполнение со следующей итерации

Инструкции. Операторы безусловного перехода

```
var arr = [[1, 2, 3, 0], [0, 0, 0, 0], [1, 1, -1, 0]];
circle:for (var i = 0; i < arr.length; i++) {
    for (var j = 0; j < arr[i].length; j++) {
        if (arr[i][j] == 0) {
            continue circle;
        }
        if (arr[i][j] < 0) {
            break circle;
        }
    }
}
alert(i + " " + j);
```



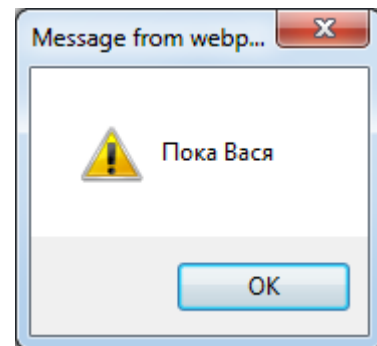
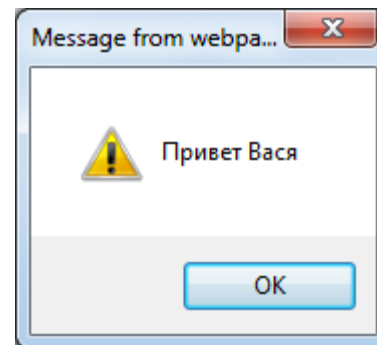
ФУНКЦИИ

Функция. Определение

- Функция — это объект. Наследник Object.
- Функция может содержать свойства и методы.
- Внутри функции могут объявляться другие функции.
- Функция может быть передана как аргумент, возвращена как результат действия функции, сохранена в переменной.

Функции. Объявление. Пример

```
Func("Вася");  
function Func(user) {  
    alert("Привет " + user);  
}  
var func = function (user) {  
    alert("Пока " + user);  
};  
func("Вася");
```



Спасибо за внимание!

Контактная информация:

Дмитрий Верескун

Инструктор

EPAM Systems, Inc.

Адрес: Саратов, Рахова, 181

Email: Dmitry_Vereskun@epam.com

<http://www.epam.com>