JavaScript Уровень 1 Основы веб - программирования

Игорь Борисов
http://igor-borisov.ru

Темы курса

- Основы программирования
- Управляющие конструкции
- Функции
- Объекты и массивы
- Использование объектов JavaScript
- Объектно-ориентированное программирование

Основы программирования

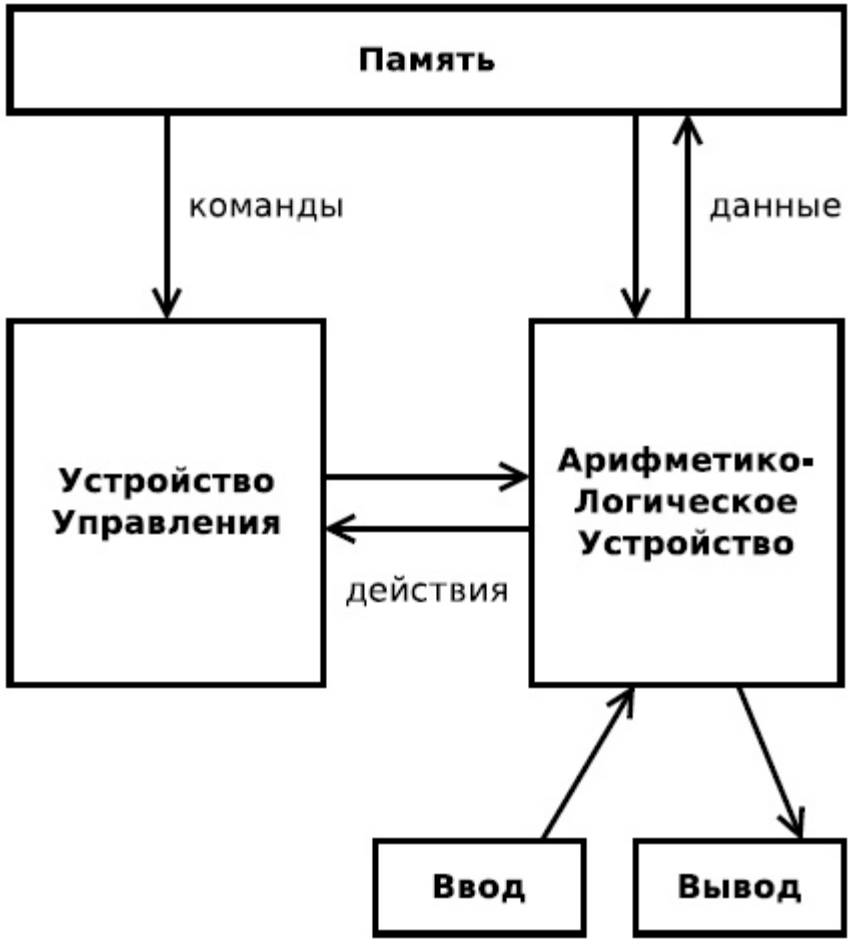
Игорь Борисов
http://igor-borisov.ru

Темы модуля

- Как устроен компьютер
- Как работают программы
- Введение в JavaScript
- Обзор базовых типов
- Операторы
- Выражения и инструкции
- Переменные
- Приведение типов
- Тривиальные типы

Компьютер!

Address	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
00000d20	00	00	00	00	00	00	00	00	00	00	00	00	00	53	4d	41
00000d30	52	54	20	43	41	4c	43	55	4c	41	54	4f	52	4e	4f	20
00000d40	54	49	4d	45	20	26	20	44	41	54	45	21	23	4e	4f	20



Программа

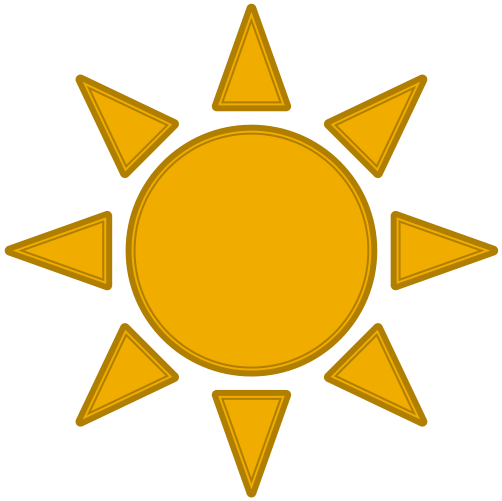
- 0 00110001 00000000 00000000
- 1 00110001 00000001 00000001
- 2 00110011 00000001 00000010
- 3 01010001 00001011 00000010
- 4 00100010 00000010 00001000
- 5 01000011 00000001 00000000
- 6 01000001 00000001 00000001
- 7 00010000 00000010 00000000
- 8 01100010 00000000 00000000

Assembly language

```
MOV dword[sum], 0
MOV dword[count], 1
_START:
MOV EAX, 11
CMP EAX, dword[count]
JZ _END
MOV EBX, dword[sum]
ADD EBX, dword[count]
MOV dword[sum], EBX
INC dword[count]
JMP _START
_END:
```

Компиляция

```
MOV dword[sum], 0
MOV dword[count], 1
_START:
MOV EAX, 11
CMP EAX, dword[count]
JZ _END
MOV EBX, dword[sum]
ADD EBX, dword[count]
MOV dword[sum], EBX
INC dword[count]
JMP _START
_END:
```



```
00110001 00000000 00000000
00110001 00000001 00000001
00110011 00000001 00000010
01010001 00001011 00000010
00100010 00000010 00001000
01000011 00000001 00000000
01000001 00000001 00000001
00010000 00000010 00000000
01100010 00000000 00000000
```

Разные реализации

```
Set Sum = 0;
Set Count = 1;
START:
If Count > 10 Then
   Goto END
Sum = Sum + Count
Count = Count + 1
Goto START
END:
Print Sum
```

```
Dim Sum = 0
Dim Count = 1
While Count <= 10
    Sum = Sum + Count
    Count = Count + 1
Wend
Print Sum
```

```
$sum = 0; $count = 1;
while(true){
   if($count > 10)
        break;
   $sum += $count; $count++;
}
echo $sum;
```

Что такое подпрограмма?

```
Procedure Sum_Nums N1, N2
    Sum = 0; Count = N1;
    While Count <= N2
          Sum = Sum + Count
          Count = Count + 1
    Wend
    Return Sum
EndProcedure
Print Sum_Nums 0, 10
```

```
function sum_nums($n1, $n2){
$sum = 0; $count = $n1;
   while ($count <= $n2){
         $sum += count;
         $count++;
   }
   return $sum;
}
echo sum_nums(0, 10);
```

JavaScript. Наконец-то!

- ECMAScript-262 (Wikipedia)
- Сценарии могут выполняться в различной среде
- Интерпретируемый язык
- Регистрозависимый язык

Инструменты

- Текстовый редактор
  - **Notepad++**, Sublime Text, LightTable
- Консоль JavaScript в браузерах
  - **Google Chrome (F12)**
  - Opera (Ctrl + Shift + I)
  - Mozilla FireFox (Ctrl + Shift + I) + addon FireBug
  - MS Internet Explorer (F12)
- Интерпретатор командной строки
  - **SpiderMonkey**, Rhino, V8

Базовые типы: числа (Number)

- `128`
- `10000000`
- `-57`

- `7.56`
- `3.754e7`
- `3.7540000`

Операторы

- 2 + 2
- Операторы бинарные и унарные
  - +
  - -
  - *
  - /
  - %
- Приоритет исполнения
  - ( )

Базовые типы: строки (String)

- Кавычки и апострофы
  - `"Hello, world!"`
  - `'Hello, world!'`
- Спецсимволы
  - `'Hello, \n\tworld!'`
  - `'Hello \'my\' world!'`
  - `'format disk c:\\ on computer'`
  - `'Это \u2014\x20 просто строка'`
- Конкатенация
  - `'Hello ' + 'my' + ' world' + '!'`

Базовые типы: булев (Boolean)

- Константы **true** и **false**
  - `3 < 4`
  - `3 > 4`
  - `5e2 == 500`
- Другие операторы сравнения
  - `===`
  - `<=` и `>=`
  - `!=` и `!==`

Логические операторы

- При проверке значения *операндов* преобразуются в *логические*
- *Внимание, приоритеты выполнения!*
  - `!` [логическое NOT (НЕ)]
  - `&&` [логическое AND (И)]
  - `||` [логическое OR (ИЛИ)]
- Число 0 и пустая строка преобразуются в **false**
  - `3 && 4`
  - `3 && 0`
  - `'Hello' && 2 || '' && 5`

Разминка

- Не запуская код вычислите, что вернёт следующая конструкция?

- ```
  ((5 >= 7) ||
  ("javascript" != "java")) &&
  !(((11 * 3) == 99) && true)
  ```
  ```

Выражения и инструкции

- Expression (выражение)
  - `5 + 10`
  - `'John'`
  - `!false`

- Statement (инструкция)
  - `5 + 10;`
  - `'John';`
  - `!false;`

Переменные

- Объявление переменной
  - `var num;`
- Инициализация переменной
  - `var user = 'John';`
  - `var x = 'a', y = 1, z;`
- Использование переменной
  - `num = 25;`
  - `num = num + 10;`
  - `num += 10;`

Манипуляции с типами

- Проблемы
  - `2 + '2'    // '22'`
  - `2 + true   // 3`
  - `'' + true  // 'true'`
  - `'x' * 10   // NaN! Что это???`
- Оператор определения типа
  - `typeof`
- Функции проверки
  - `isNaN()`
  - `isFinite()`

Приведение типов

- В число
  - `x = '5' * 1;`
  - `x = +'5';`
  - `x = parseInt('5.3');`
  - `x = parseFloat('5.3');`
  - `x = Number(true);`
- В строку
  - `x = 5 + '';`
  - `x = String(true);`
- В логический тип
  - `x = !!5;`
  - `x = Boolean(true);`

Тривиальные типы

- Пробуем:
  - `var result;`
  - `result = x;`
  - `ReferenceError: x is not defined`
  - `typeof x;`
- undefined
- null

Запускаем файл в консоли!

- ```
  // Это файл test.js
  var name = 'John'; // Имя пользователя
  print('Hello ' + name);
  ```

- ```
  js test.js
  ```

Запускаем файл в браузере!

- ```
  <script>
      /*
            Многострочный комментарий
      */
      var name = 'John';
      console.log('Hello ' + name);
  </script>
  ```

- ```
  <script src="file.js"></script>
  ```

Выводы

- Как устроен компьютер
- Как работают программы
- Введение в JavaScript
- Обзор базовых типов
- Операторы
- Выражения и инструкции
- Переменные
- Приведение типов
- Тривиальные типы

Управляющие конструкции

Игорь Борисов
http://igor-borisov.ru

Темы модуля

- Циклы
- Цикл while
- Операторы инкремента и декремента
- Цикл for
- Цикл do while
- Управляющие конструкции if – else if – else
- Метки
- Прерывание и продолжение цикла
- Управляющая конструкция switch

Цикл (loop)

- Инициализация "счетчика"
- Проверка условия
- Выполнение инструкций
- Изменение "счетчика"

while

```
while ( условие ) {
    инструкция 1;
    инструкция 2;
    ...;
}
```

Операторы ++ и --

- Инкремент / Декремент
  - `++ --`
- POST
  - `var i = 1;`
  - `var x = i++; // x==1, i==2`
  - `var x = i; i += 1;`
- PRE
  - `var i = 1;`
  - `var x = ++i; // x==2, i==2`
  - `i += 1; var x = i;`

Лабораторная работа – 2.1

- Напишите программу, которая возводит число **2** в степень **10**
- Использовать цикл **while**

for

- ```
  for (выражения; условие; выражения)
     инструкция;
  ```

- ```
  for (выражения; условие; выражения){
     инструкция 1;
     инструкция 2;
     ...;
  }
  ```

Лабораторная работа – 2.2

- Напишите программу, которая возводит число **2** в степень **10**
- Использовать цикл **for**

do - while

```
do {
   инструкция 1;
   инструкция 2;
   ...;
} while ( условие );
```

if – else if - else

```
if (условие 1){
   инструкция;
   ...;
}else if (условие 2){
    инструкция;
   ...;
}else{
    инструкция;
   ...;
}
```

Тернарный оператор ?:

- ```
  if (условие){
     инструкция;
     ...;
  }else{
      инструкция;
     ...;
  }
  ```

- ```
  (условие) ? если true : если false;
  ```

Лабораторная работа – 2.3

- Создайте переменную **age** и присвойте ей целочисленное значение
- Используя конструкцию **if - else if - else**, напишите программу, которая проверяет значение в переменной **age** и:
  - если значение попадает в диапазон 18 – 59 (вкл.)
    - выводит "Вам еще работать и работать"
  - если значение больше 59
    - выводит "Вам пора на пенсию"
  - если значение попадает в диапазон 1 – 17 (вкл.)
    - выводит "Вам работать еще рано - учитесь"

Прерывание и продолжение

```
for (выражения; условие; выражения){
    инструкция;
    if (условие 1)
        break;
    инструкция;
    if (условие 2)
        continue;
    инструкция;
}
```

Метки

```
outer: for (выражения; условие; выражения){
    inner:  while (условие){
                while-инструкция;
                if (условие 1)
                    break inner;
                while-инструкция;
            }
            for-инструкция;
        }
```

switch

```
switch(выражение){
   case условие 1:
        инструкции; break;
   case условие 2:
        инструкции; break;
   case условие n:
        инструкции; break;
   default: инструкции;
}
```

Выводы

- Циклы
- Цикл while
- Операторы инкремента и декремента
- Цикл for
- Цикл do while
- Управляющие конструкции if – else if – else
- Метки
- Прерывание и продолжение цикла
- Управляющая конструкция switch

Функции

Игорь Борисов
http://igor-borisov.ru

Темы модуля

- Понятие функций
- Возврат значений
- Области видимости
- Анонимная функция
- Замыкания
- Рекурсия

Функции

```
function sayHello() {
  console.log('Hello world!');
}
sayHello();

function sayHello(name) {
  console.log('Hello '+name+'!');
}
sayHello('John');
```

Возврат значений

```
function sum(x, y) {
   return x + y;
}

result = sum(10, 20);
console.log(result);

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

Лабораторная работа – 3.1

- Напишите функцию **power()**, которая принимает два целочисленных значения
- Функция должна **возвращать** результат возведения первого числа в степень второго

Области видимости

```
var x = "x-global"; var y = "y-global";
function outerFunction() {
    var x = "x-local";
    console.log(x); //x-local
    function innerFunction(z) {
        console.log(x); //x-local
        console.log(y); //y-global
        console.log(z); //z-local
        y = "y-local";
    }
    innerFunction('z-local');
}
outerFunction();
console.log(x); //x-global
console.log(y); //y-local
```

Использование переменных

- Функция как значение переменной
  - ```
    function foo(val){
     return val + val;
    }
    ```
  - ```
    var sum = foo;
    ```
  - ```
    sum(2);
    ```
- Анонимная функция
  - ```
    var sum = function (val){
                        return val + val;
                    }
    ```
  - ```
    sum(2);
    ```
    ```

Замыкания

```
var x = "global";
function outerFunction() {
    var x = "local";

    function innerFunction() {
        console.log(x);
    }
    return innerFunction;
}
var inner = outerFunction();
inner();
```

Лабораторная работа – 3.2

- Напишите функцию **compare()**, которая принимает целочисленное значение **x** и возвращает анонимную функцию
- Анонимная функция должна принимать целочисленное значение **y** и возвращать результат сравнения значений **y** и **x** как:
  - Если **y** больше, чем **x**, то возвращается **true**
  - Если **y** меньше, чем **x**, то возвращается **false**
  - Если значения равны, то возвращается **null**

Рекурсия

```
function countLine(count) {
    if (count > 10)
        return;
    else
        console.log("Line: " + count++);
        countLine(count);
}

countLine(1);
```

Выводы

- Понятие функций
- Возврат значений
- Области видимости
- Анонимная функция
- Замыкания
- Рекурсия

Объекты и массивы

Игорь Борисов
http://igor-borisov.ru

Темы модуля

- Объектный тип: Объект (Object)
- Свойства объекта
- Методы объекта
- Методы функций
- Объектный тип: Массив (Array)
- Свойство и методы массива
- Встроенный объект Math

Объектный тип: Объект (Object)

```
var user = {};

user.name = 'John';
user.age = 25;
user.admin = true;

console.log(user.name);
```

Свойства объекта

```
var user = {
    name: 'Guest',
    age: 0,
    admin: false
};
```

```
user.name = 'John';
user.age = 25;
```

Свойства объекта: варианты

```
var user = {
    'user name': 'Guest',
    age: 0,
    'var': false
};

user['user name'] = 'John';
console.log(user['var']);
```

Обращение к свойствам объекта

```
var user = {
    name: 'Guest',
    '2': 0
};

console.log(user['2']);
console.log(user[1+1]);

var x = 'name';
console.log(user[x]);
```

Проверка наличия свойств

- `var user = {x: 25};`

- `console.log('x' in user);`
- `console.log('y' in user);`

- `delete user.x`

Перебор свойств объекта

```
var o = {0:'Guest', 1:0, 2:false};
for (var cur = 0; cur in o; cur++){
  console.log(cur + ": " + o[cur]);
}
```

```
var o = {name:'Guest', age:0,
                       'var':false};
for (var cur in o){
  console.log(cur + ": " + o[cur]);
}
```

Лабораторная работа – 4.1

- Создайте пустой объект **book1**
- Добавьте объекту свойства: **title**, **pubYear** и **price** с произвольными значениями

- Создайте объект **book2** и установите свойства **title**, **pubYear** и **price** с произвольными значениями в момент создания объекта

- Выведите свойства объектов в цикле

Сравнение и передача значений

```
var obj1 = {x: 10};
var obj2 = obj1;
var obj3 = {x: 10};

obj1 == obj2; // true
obj1 == obj3; // false

obj1.x = 20;
console.log(obj2.x); // 20
```

Методы объекта

```
var o = {
    method_1: function(){
        console.log('Hello, world!');
    },
    method_2: function(x){
        console.log('Hello, '+x+'!');
    }
};
o.method_1();
o.method_2('John');
```

Что такое this?

```
var user = {
    name: 'Guest',
    getName: function(){
            console.log(this.name);
    }
};
```

```
user.name = 'John';
user.getName();
```

Ещё раз про this

```
function foo(){
   console.log(this.name);
}
var user = {
   name: 'John',
   getName: foo
};
user.getUserName = foo;
user.getName();
user.getUserName();
```

Лабораторная работа – 4.2

- Используйте код *Лабораторной работы 4-1*
- Добавьте объекту **book1** свойство **show**
- Свойство должно содержать анонимную функцию выводящую название и цену книги
- Опишите функцию **showBook()**, которая выводит название и цену книги
- Добавьте объекту **book2** свойство show
- Свойство должно содержать функцию **showBook()**
- Вызовите метод **show()** у обоих объектов

Методы функций

```
function sum(x, y){
   return this.num + x + y;
}
var obj = {num: 10};
```

```
obj.m = sum;
obj.m(20, 30);
delete obj.m;
```

```
sum.call(obj, 10, 20);
```

И ещё о функциях

```
function foo(x, y){
  console.log(foo.length); // 2
}
console.log(foo.length); // 2

foo(1, 2, 3);
function foo(x){
  console.log(arguments.length);//3
  console.log(arguments[0]); // 1
}
```

Объектный тип: Массив (Array)

```
var a = [];

var x = 'moon';
var y = function(){
                    console.log('Hello');
                };
var a = [10, "sun", x, y, true];
a[1];
a[3]();
a[9] = 100;
```

Длина массива

```
var a = [1, 5];
console.log(a.length); // 2

a[23] = 11;
console.log(a.length); // 24

var a = [1, 5];
a.length = 3; // [1, 5, undefined]
```

Перебор элементов массива

- `var a = [5, 'abc', 73]; a[99] = 1;`

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

- `var x = 0;`
- `for (var i in a) { x++; }`
  ```

Удаление элементов массива

- `var x = 0; var a = [4, 12, 7];`
- `a[1] = undefined;`

- `for (var i in a) { x++; }`

- `var x = 0; var a = [4, 12, 7];`
- `delete a[1];`

- `for (var i in a) { x++; }`

Получение строки из массива

- `var a = [1, 5, 7];`
- `var s = a.toString(); // 1,5,7`
- `var s = a.join(); // 1,5,7`
- `var s = a.join('---'); // 1---5---7`

Сложение массивов

```
var a = [1, 5];
var b = [11, 8];

var arr = a.concat(3, b);
  // [1, 5, 3, 11, 8]

var s = [1, 5] + [11, 8];
  // 1,511,8
```

Получение части массива

- `var a = [1, 5, 7, 12, 9];`

- `var arr = a.slice(2);`
  - `// [7, 12, 9]`
- `var arr = a.slice(1, 3);`
  - `// [5, 7]`
- `var arr = a.slice(-3, -1);`
  - `// [7, 12]`
- `var arr = a.slice(2, 1);`
  - `// []`

Сортировка массива

- `var a = [14, 51, 7, 2];`
- `a.reverse();`

- `var a = [14, 51, 7, 2];`
- `a.sort();`
  - `// [14, 2, 51, 7] !?`

- `function mySort(a, b) {return a-b;}`
- `a.sort(mySort);`
  - `// [2, 7, 14, 51]`

Работа с концом массива

- Исходный массив
  - `var a = [5, 'abc', 73];`
- Извлечение элемента
  - `var v = a.pop();`
    - `// [5, 'abc'], v = 73`
- Добавление элементов
  - `var v = a.push(12, 3);`
    - `//[5, 'abc', 12, 3], v = 4`

Работа с началом массива

- Исходный массив
  - `var a = [5, 'abc', 73];`
- Извлечение элемента
  - `var v = a.shift();`
    - `// ['abc', 73], v = 5`
- Добавление элементов
  - `var v = a.unshift(12, 3);`
    - `//[12, 3, 'abc', 73], v = 4`

Вставка и удаление в любом месте

- Исходный массив
  - `var a = [5, 'abc', 73, 12, 8];`

- `var arr = a.splice(1, 2);`
  - `// [5, 12, 8]`
  - `// arr = ['abc', 73]`
- `var arr = a.splice(1, 0, 3);`
  - `// [5, 3, 'abc', 73, 12, 8]`
  - `// arr = []`

Встроенный объект Math

- Свойства
  - E, LN10, LN2, LOG10E, LOG2E, PI, SQRT1_2, SQRT2
- Методы
  - sin(), asin(), cos(), acos(), tan(), atan(), atan2()
  - ceil(), floor(), round(), abs(), max(), min(), random()
  - exp(), log(), pow(), sqrt()

- `console.log(Math.PI);`
- `console.log(Math.pow(4, 2));`

Лабораторная работа – 4.3

- Исходный код:
  - ```
    var a = [5, 12];
    var b = [];
    a[99] = 7;
    ```
- Записать в массив **b** квадраты значений массива **a**:
  - с помощью цикла **for**
  - с помощью цикла **for/in**
    ```

ECMAScript 5 методы массива

- forEach:
  - `var nums = [1, 2, 3];`
  -
```
nums.forEach(function(v){
        console.log(v * 10);
});
```
- map:
  - `var nums = [1, 4, 9];`
  -
```
var result = nums.map(function(v) {
        return v * 10;
});
```

Выводы

- Объектный тип: Объект (Object)
- Свойства объекта
- Методы объекта
- Методы функций
- Объектный тип: Массив (Array)
- Свойство и методы массива
- Встроенный объект Math

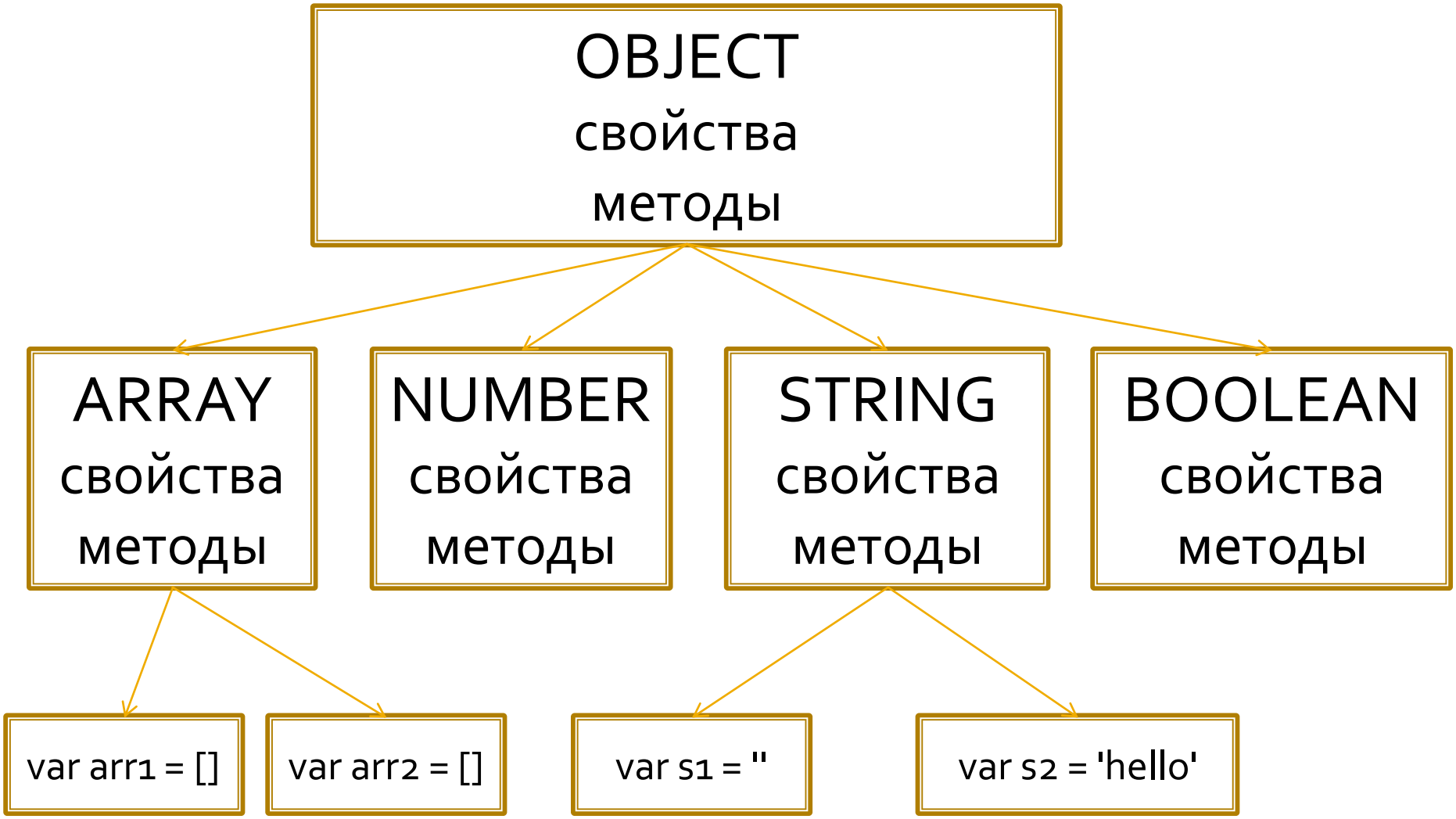
Использование объектов JavaScript

Игорь Борисов
http://igor-borisov.ru

Темы модуля

- Свойства и методы объекта Number
- Свойства и методы глобального объекта
- Свойство и методы объекта String
- Использование регулярных выражений

Что мы имеем?


OBJECT
свойства
методы
ARRAY
свойства
методы
NUMBER
свойства
методы
STRING
свойства
методы
BOOLEAN
свойства
методы
var arr1 = []
var arr2 = []
var s1 = ''
var s2 = 'hello'

Свойства и методы глобального объекта

- Infinity
- -Infinity
- NaN

- **NaN != NaN**

- isNaN(3 * 'a'); // true
- isFinite(3 * 'a'); // false
- isFinite(3 / 0); // false

Объект Number

- Статические свойства
  - Number.NEGATIVE_INFINITY
  - Number.POSITIVE_INFINITY
  - Number.NaN
  - Number.MIN_VALUE
  - Number.MAX_VALUE

Преобразование числа в строку

- toString()
- var n = 12345.6789;

- n.toFixed(2); // '12345.68'

- n.toExponential(2); // '1.23e+4'

- n.toPrecision(6); // '12345.7'
- n.toPrecision(4); // '1.235e+4'

Преобразование строки в число

```
var s = '37.5 км';

var n = parseFloat(s); // 37.5
var n = parseInt(s); // 37

var s = '$99.9';

var n = parseFloat(s); // NaN
var n = parseInt(s); // Nan
```

Использование систем счисления

```
var n = 255;
var s = n.toString(16); // ff

var s = 'ff';
var n = parseInt(s, 16); // 255
```

Длина строки

```
var str = 'Это \u2014\u0020просто\x20пример';
```

- `console.log(str.length);`

Сложение строк

```
var s1 = 'Это';
var s2 = ' \u2014\x20';
var s3 = 'просто пример';

var s = s1 + s2 + s3;

var s = s1.concat(s2, s3);
```

Регистр символов

- `var x = s.toLowerCase();`
- `var x = s.toUpperCase();`

Получение символа из строки

- `var s = 'просто пример';`

- `var x = s.charAt(4); // 'т'`
- `var x = s.charAt(40); // ''`

- `var x = s.charCodeAt(4);//1090 (U+0442)`
- `var x = s.charCodeAt(40); // NaN`

- Создание строки из кодов символов
- `var x = String.fromCharCode(1051, 1091, 1085, 1072); // Луна`

Получение части строки

```
var s = 'просто пример';
var x = s.slice(3, 6); // сто
var x = s.substring(3, 6); // сто

var x = s.slice(6); // пример
var x = s.substring(6); // пример

var x = s.slice(); // просто пример
var x = s.substring(); // просто пример

var x = s.slice(6, 3); // ""
var x = s.substring(6, 3); // сто

var x = s.slice(-6, -3); // при
```

Поиск по строке

- `var x = s.indexOf('пр'); // 6`
- `var x = s.indexOf('пр', 7); // 13`

- `var x = s.lastIndexOf('пр'); // 13`
- `var x = s.lastIndexOf('пр', 9); // 6`

Лабораторная работа – 5.1

- Исходные данные:
  - `var s = 'Мы знаем, что монохромный цвет – это градации серого';`
  - `var txt = 'хром';`
  - `var word;`
- В значении переменной **s** найдите слово, содержащее подстроку, которая является значением переменной **txt** и присвойте его переменной **word**

Замена в строке

```
var s = 'Это просто пример';

var x = s.replace('просто', 'сложный');

// 'Это сложный пример'
```

Разбиение строки

- `var s = 'Это просто пример';`
- `var x = s.split(' ');`
  - `// [Это, просто, пример]`
- `var x = s.split(' ', 2);`
  - `// [Это, просто]`

Регулярные выражения

- `var re = /regexp/флаги;`

- Методы объекта String
  - `replace(regexp, string);`
  - `split(regexp)`
  - `search(regexp)`
  - `match(regexp)`
- Методы объекта RegExp
  - `test(string);`
  - `exec(string);`

Базовое использование

```
var pattern = /@/;
var str = "vasya@gmail.com";
str.search(pattern); // 5 indexOf()

var re = /gmail|hotmail/;
str.search(pattern);

"folder/file".search(/\//); // 6
/\//.test("folder/file"); // true
```

Специальные последовательности

- \0 Символ NULL (\u0000)
- \t Горизонтальная табуляция (\u0009)
- \n Перевод строки (\u000A)
- \r Возврат каретки (\u000D)
- \\ Обратная косая черта (\u005C)
- \xXX Символ Latin-1, заданный двумя шестнадцатеричными цифрами
- \uXXXX Символ Unicode, заданный четырьмя шестнадцатеричными цифрами

- ^ $. * + ? = ! : | \ / () [] {}

Классы символов

- `var re = /[abc]/;`
- `var re = /[^abc]/;`
- `var re = /[a-f]/;`

.	Любой символ, кроме перевода строки	
\w	Любой символ ASCII	[a-zA-Z0-9_]
\W	Противоположное \w	[^a-zA-Z0-9_]
\d	Любая цифра ASCII	[0-9]
\D	Противоположное \d	[^0-9]
\s	Любой символ-разделитель Unicode	
\S	Противоположное \s	

Повторения

- {n,m}
  - Шаблон повторяется не менее n, но и не более m раз
- {n,}
  - Шаблон повторяется не менее n раз
- {n}
  - Шаблон повторяется точно n раз

- ? Эквивалентно {0, 1}
- + Эквивалентно {1,}
- * Эквивалентно {0,}

Лабораторная работа – 5.2

- Составить регулярное выражение, которое соответствует следующим форматам даты:

- 25-02-2012
- 25-2-2012
- 01-12-2011
- 2-12-1978

- Для проверки использовать метод **test()**

Позиции соответствия

- ^ Поиск с начала строки
- $ Поиск до конца строки
- \b Позиция между символом ASCII и не символом ASCII (граница слова)
- \B Позиция между двумя символами ASCII (не граница слова)

- `var re = /^abc/;`
- `var re = /abc$/;`
- `var re = /\bjava\b/;`
- `var re = /\bjava\B/;`

Флаги

- ignoreCase
  - `var r = /a/i; //ищутся символы 'a' и 'A'`
- global
  - `var r = /a/g; //ищутся все символы 'a'`
- multiline
  - `var r = /a/gim;`

Группировка и ссылки

- `var r = /ab(cd)?/;`
- Ищем содержимое в одинаковых кавычках
  - `var r = /['"][^'"]*['"]/; //проблема`
  - `var r = /(['"])[^'"]*\1/; //ссылка \1`
- Внешние ссылки
  - `'1A'.replace(/(\d+)([a-z]+)/i, '$2-$1');`

Методы match() и exec()

- `var re = /(ab)(cd)(xyz)/;`
- `var res = 'abcdxyz'.match(re);`
- Результат (переменная res)
  - ```
    [
    'abcdxyz', // индекс 0
    'ab',
    'cd',
    'xyz'
    ]
    ```
- `/(ab)(cd)(xyz)/.exec('abcdxyz');`
  - `['abcdxyz', 'ab', 'cd', 'xyz']`
    ```

Выводы

- Свойства и методы объекта Number
- Свойства и методы глобального объекта
- Свойство и методы объекта String
- Использование регулярных выражений

Объектно-ориентированное программирование

Игорь Борисов
http://igor-borisov.ru

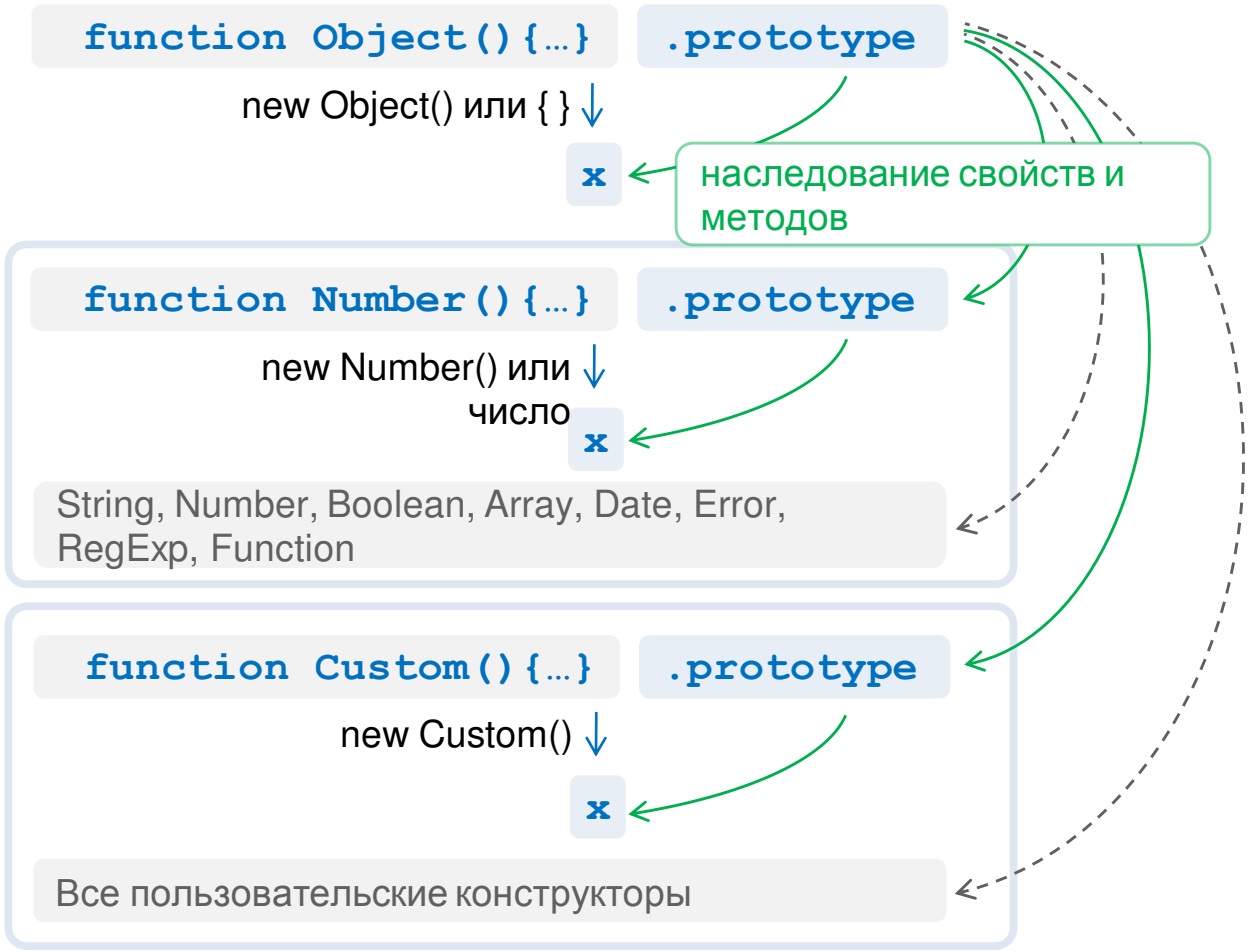
Темы модуля

- Конструкторы объектов
- Прототипы
- Объект Date
- Объект Error

Конструкторы

```
function Point(x, y) {
    this.x = x; this.y = y;
    this.getCoords = function(){
        return [this.x, this.y];
    };
}
var start = new Point(10, 20);
var coords = start.getCoords();
start.constructor;
```

Как всё происходит?



Новый взгляд на привычное

```
var n = new Number(5);
var s = new String('John');
var b = new Boolean(true);

var a = new Array(5);
var o = new Object();

var re = new RegExp('[a-z+]','i');
```

Проверка конструктора

- ```
  var myObject = [];
  ```
- ```
  myObject instanceof Array;
  ```
- ```
  var x = new Number(5);
  x instanceof Object; // true
  ```
- ```
  x++;
  x instanceof Object; // false
  ```

Прототипы

- ```
  Point.prototype.add = function(obj) {
      return new Point(
            this.x + obj.x, this.y + obj.y);
  };
  ```
- ```
  var start = new Point(10, 10);
  var finish = new Point(20, 20);
  var line = start.add(finish);
  ```
- ```
  var line = new Point(10, 10).add(new
  Point(20, 20);
  ```
- ```
  line.constructor.prototype.draw = ...
  ```

Методы объекта

- `var n = new Number(5);`
- `n.valueOf();`

- `var obj = {x: 10};`
- `obj.hasOwnProperty('x');`

- `Object.prototype.isPrototypeOf(obj);`

Лабораторная работа – 6.1

- Создайте конструктор **Book** со свойствами:
  - title
  - pubYear
  - price
- Создайте объект типа **Book** передав в конструктор произвольные значения
- Добавьте в **прототип** конструктора **Book** метод **show**, который выводит значения св-в **title** и **price**
- Вызовите метод **show** созданного объекта

Объект Date

- Время хранится в миллисекундах
- Точка отсчета – 1 января 1970 года (00:00:00)

- Greenwich Mean Time (GMT)
  - Дата и время указывается в соответствии с местным часовым поясом
  - Указывается смещение относительно Гринвичского меридиана
  - Смещение зависит от переходов на летнее и зимнее время
- Universal Time Coordinated (UTC)
  - Дата и время в любой точке планеты одинаково
  - Точка отсчета совпадает с точностью до долей секунды с точкой отсчета GMT
  - Никаких переходов на летнее и зимнее время в UTC нет

Создание объекта Date

- `var d = new Date();`
- `var d = new Date('Jan 01 2010 01:00:00');`
- `var d = new Date(1234567890000);`
- `var d = new Date(2011, 5);`
  - **год**, **номер месяца**, дата, часы, минуты, секунды, миллисекунды
- `var d = new Date(2011, 5, 21);`
- `var d = new Date(2011, 12, 40);`
- `var d = new Date(2011, -1);`

Методы объекта Date

- getFullYear()
  - getUTCFullYear()
- getMonth() 0-11
- getDate() 1-31
- getHours() 0-24
- getMinutes() 0-59
- getSeconds() 0-59
- getMilliseconds() 0-999
- getDay() 0-6
  - 0 - Воскресенье!

Дополнительные методы

- getTimezoneOffset()
  - смещение GMT в минутах

- Количество миллисекунд с 01.01.1970
  - getTime(), valueOf()
  - ```
    var n = Date.parse('Feb 09 2012
                                03:45:15');
    var d = new Date(n);
    ```

  - ```
    var n = Date.UTC(2012, 5);
    ```

Строчное представление даты

- toString() (GMT)
  - toLocaleString() (GMT)
- toUTCString()
- toTimeString()
  - toLocaleTimeString()
- toDateString()
  - toLocaleDateString()

Запись информации

- setFullYear()
  - setUTCFullYear()
- setMonth()
- setDate()
- setHours()
- setMinutes()
- setSeconds()
- setMilliseconds()
- setTime() не имеет UTC

Лабораторная работа – 6.2

- Описать функцию **getDate()**, которая
  - получает в качестве аргумента дату в виде строки в формате, описанном в *Лабораторной работе 5-2*
  - возвращает объект типа **Date**, соответствующий переданной дате

Объект Error

```
try{
   console.log(x);
}catch(e){
   console.log(e.name);
   console.log(e.message);
}
console.log('test');
```

Конструктор объекта Error

```
try{
    var a = 1, b = 0, x;
    if(b == 0)
        throw new Error('Ошибка!');
    x = a / b;
}catch(e){
    console.log(e.message);
}
```

Выводы

- Конструкторы объектов
- Прототипы
- Объект Date
- Объект Error