Internet-технологии

ЛЕКЦИЯ №7

ЯЗЫК НАПИСАНИЯ СЦЕНАРИЕВ JAVASCRIPT

Содержание

- Библиотека jQuery.
- Букмарклеты.
- Работа с файлами.
- Объекты. Добавление свойств, методов. Прототипы.
- ООП. Инкапсуляция, наследование и полиморфизм в языке JavaScript.

Из прошлой лекции… Приведение типов. Примеры

```
+" \n34 \n"       // 34
+" \n3\n 4 \n"    // NaN, \n или пробел посередине числа - ошибка
+" \n3 4 \n"      // NaN
+" \n3+4 \n"      // NaN

parseInt(" \n34 \n")      // 34
parseInt(" \n3\n4 \n")    // 3
parseInt(" \n3 4 \n")     // 3
parseFloat(" \n3 4 \n")   // 3
parseInt(" \n3+4 \n")     // 3
```

Из прошлой лекции… Примеры

```
+" 34.1"                // 34.1
parseInt(" 34.1")       // 34
parseFloat(" 34.1")     // 34.1
+"3e4"                  // 30000
parseInt("3e4")         // 3
parseFloat("3e4")       // 30000
+"oxFF"                 // 255
parseInt("oxFF")        // 255
parseFloat("oxFF")      // 0
```

Перевод из 16-ного представления в 8-ное:

```
var n = oxFF;
var n8 = parseInt(n.toString(8));
```

```
parseFloat("\n 40 years")   // 40
parseFloat("40 years")      // 40
parseInt("He was 40")       // NaN

parseInt("16",16)           // 22
parseInt("16",12)           // 18
```

16 и 12 в примерах – основание системы счисления для преобразуемой в число строки. Результат – число в десятичной системе.

jQuery

jQuery – библиотека JavaScript, позволяющая легко получать доступ к элементу DOM, обращаться к их атрибутам и содержимому, манипулировать ими, создавать анимацию, устанавливать обработчики событий. Также библиотека jQuery предоставляет и удобный API для работы с AJAX.

Библиотека представлена в виде обычного текстового файла, имеющего расширение .js. Сейчас существует две основные ветки версий jQuery 1.x и 2.x. Их отличия заключаются лишь в том, что в версиях 2.x перестали поддерживаться браузеры IE версий 8, 7 и 6.

jQuery доступна в **сжатом** и **несжатом** варианте.

Подключение jQuery:

```
<script type="text/javascript" src="js/jquery-1.6.1.min.js"></script>
```

jQuery. Подключение

Подключение jQuery с **CDN** (Content Delivery Network):

```
<script type="text/javascript" src="http://ajax.microsoft.com/ajax/jquery/jquery-1.4.2.min.js"></script>
```

Сначала попытайтесь загрузить jQuery со стороннего хранилища и в случае его отказа можете запросить jQuery со своего сервера:

```
<script type="text/javascript" src="http://ajax.microsoft.com/ajax/jquery/jquery-1.4.2.min.js"></script>
<script type="text/javascript">
  if (typeof jQuery == 'undefined') {
    document.write(unescape("%3Cscript src='/js/jquery-1.6.1.min.js' type='text/javascript'%3E%3C/script%3E"));
  }
</script>
```

jQuery. Синтаксис

Стандартный синтаксис jQuery команд:

```
$(селектор).метод();
```

Знак **$** сообщает, что символы идущие после него являются jQuery кодом. Можно переопределить.

Селектор позволяет выбрать элемент на странице;

Метод задает действие, которое необходимо совершить над выбранным элементом. Методы в jQuery разделяются на следующие группы:

- Методы для **манипулирования** DOM;
- Методы для **оформления** элементов;
- Методы для привязки обработчиков **событий**;
- Методы для создания **AJAX** запросов;
- Методы для создания **эффектов**.

jQuery() или $() . Поиск элементов

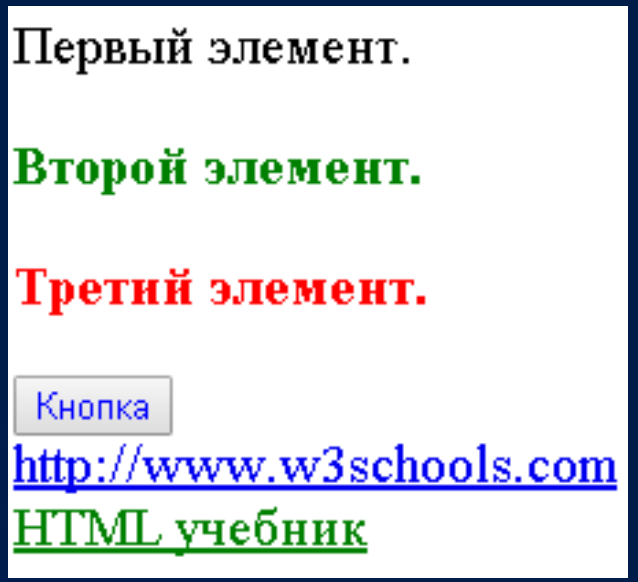
Функция **jQuery()** или **$()** возвращает объект jQuery, который содержит выбранные элементы и имеет методы для работы с этими элементами. С ее помощью можно как находить существующие элементы на странице, так и добавлять новые.

$("div")	вернет все div-элементы на странице.
$(".someBlock")	вернет все элементы с классом someBlock.
$("#content")	вернет элемент с идентификатором content.
$("#content2 div.someBlock")	вернет div-элементы с классом someBlock, которые находятся внутри элемента с идентификатором content2.
$("div:odd")	вернет div-элементы, находящиеся на странице под нечетными номерами.
$("[value = 5]")	вернет все элементы с атрибутом value, равным 5.

jQuery. Пример изменения свойств элементов

```
<p> Первый элемент.</p>
<p id="el2"> Второй элемент.</p>
<p class="el3">Третий элемент.</p>
<input type="button" value="Кнопка" />
<br><a href="http://www.w3schools.com">
http://www.w3schools.com</a>
<br><a href="http://www.htmlbook.ru">
HTML учебник</a>
```

```
$(document).ready(function() {
  $("p").css("fontSize","20px");
  $("#el2").css("color","green");
  $(".el3").css("color","red");
  $("#el2,.el3").css("fontWeight","bold");
  $("input").css("color","blue");
  $("[href]").css("fontSize","20px");
  $("[href='http://www.htmlbook.ru']").css("color","green");
});
```



jQuery. Преждевременное выполнение

Выполнение кода до полной загрузки документа часто приводит к ошибкам. Возможно четыре способа предотвращения этой пролемы:

```
<script type='text/javascript'>
 $(document).ready(function() {
  alert("1");
 });
 $().ready(function() {
  alert ("2");
 });
 $(function(){
  alert ("3");
 });
</script>
```

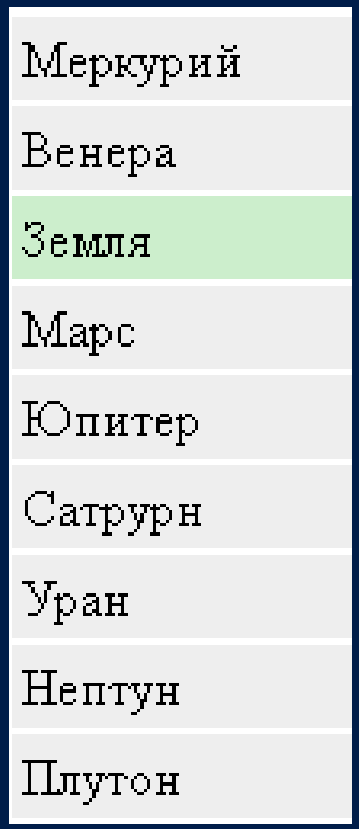
```
<body>
<div>Содержимое тела документа</div>

<script type='text/javascript'>
 alert ("4");
</script>
</body>
```

jQuery. Методы для изменения элементов

$("#bigIt").css("height")	возвратит значение высоты у элемента с идентификатором bigIt.
$("div").css("width", "20px")	установит новое значение ширины всем div-элемента на странице.
$("#bigIt").attr("class")	возвратит значение класса элемента с id = bigIt.
$("#bigIt").attr("class", "box")	установит новое значение атрибута class у элемента с id = bigIt.
$("#bigIt").html(New!)	изменит все html-содержимое элемента с id = bigIt, на заданное в методе html.
$("#bigIt").text()	возвратит текст, находящийся внутри элемента с id = bigIt.
$(".someBox").empty()	очистить от содержимого элементы с классом someBox.

jQuery. Пример

```
<body>
 <ul id="list">
   <li class="item">Меркурий</li >
   <li class="item">Венера</li >
   <li class="item">Земля</li >
   .....
   <li class="item">Плутон</li >
 </ul>
<script>
   $("#list .item").css("background-color", function(i,val){
            if($(this).text() == "Земля")
              return "#cceecc";
            else
              return val;
   });
</script></body>
```



jQuery. Работа с атрибутами и свойствами

.attr()	возвращает/изменяет значение атрибута у элементов на странице
.removeAttr()	удаляет атрибут у элементов на странице
.addClass()	добавляет класс элементам на странице
.removeClass()	удаляет класс(ы) у элементов на странице
.val()	возвращает/изменяет (в зависимости от числа параметров) значение атрибута value у элементов на странице

jQuery. Добавление содержимого

.html()	Возвращает/изменяет (в зависимости от числа параметров) html-содержимое элементов на странице
.text()	Возвращает/изменяет (в зависимости от числа параметров) текст, находящийся в элементах на странице
.append() **.appendTo()**	Добавляет заданное содержимое в конец элементов на странице
.after() **.insertAfter()**	Добавляет заданное содержимое после элементов на странице
.before() **.insertBefore()**	Добавляет заданное содержимое перед элементами на странице
.wrap() **.wrapAll()**	Окружает элементы на странице заданными html-элементами
.wrapInner()	Окружает содержимое элементов на странице заданными html-элементами

jQuery. Работа с html-содержимым элемента

Функция **.html()** в озвращает или изменяет html-содержимое выбранных элементов страницы. Функция имеет три варианта использования:

.html() – возвращает html-содержимое выбранного элемента. Если таких элементов несколько, то значение будет взято у первого.

.html(newHTML) – заменяет содержимое всех выбранных элементов на *newHTML*.

.html(function(index, value)) – заменяет содержимое выбранных элементов на возвращенное пользовательской функцией значение. Функция вызывается отдельно, для каждого из выбранных элементов. При вызове ей передаются следующие параметры: *index* – позиция элемента в наборе, *value* – текущее html-содержимое.

$(".topBlock").html()	вернет html-содержимое первого элемента с классом topBlock.
$(".topBlock").html(" New ")	изменит содержимое всех элементов с классом topBlock на параграф с текстом "New".

jQuery. Работа с текстовым содержимым элемента

Функция **.text()** возвращает или изменяет текстовое содержимое выбранных элементов страницы. Функция имеет три варианта использования:

.text() – возвращает текст содержащийся в выбранном элементе. Если таких элементов несколько, метод возвратит строку, в которой будет содержимое всех элементов, расположенное через пробел.

.text(newText) – заменяет все содержимое у выбранных элементов, на текст newText.

.text(function(index, value)) – заменяет все содержимое у выбранных элементов на возвращенный пользовательской функцией текст. Функция вызывается отдельно, для каждого из выбранных элементов.

$(".topBlock").text()	вернет текстовое содержимое всех элементов с классом topBlock (одной строкой).
$(".topBlock").text(" New ")	заменит содержимое всех элементов с классом topBlock на текст "New".

jQuery. Добавление содержимого после элементов

Функции **.after()** и **.insertAfter()** вставляют заданное содержимое сразу после определенных элементов страницы. Имеется два варианта использования:

elements.after(content), content.insertAfter(elements)

сразу после элементов *elements* будет добавлено содержимое *content*, который может быть задан html-текстом, объектом jQuery или DOM объектом. Различия функций заключается только в порядке следования содержимого и элементов, после которых это содержимое должно быть вставлено.

.after(function(index))

после выбранных элементов будет добавлен html-текст, который будет возвращен пользовательской функцией. Функция вызывается отдельно, для каждого из выбранных элементов. При вызове ей передается один параметр: index – позиция элемента в наборе.

jQuery. Добавление содержимого после элементов

```
<ul class="list l1">
<li class="item it1"> Высоко </li>
<li class="item it2"> Быстро </li>
<li class="item it3"> Сильно </li>
</ul>
<ul class="list l2">
<li class="item it1"> Выше </li>
<li class="item it2"> Быстрее </li>
<li class="item it3"> Сильнее </li>
</ul>
```

```
$(".it1").after("<li class='item'>Тест</li>");
```

или

```
$("<li class='item'>Тест</li>").insertAfter($(".it1"));
```

```
<ul class="list l1">
<li class="item it1"> Высоко </li>
<li class="item"> Тест </li>
<li class="item it2"> Быстро </li>
<li class="item it3"> Сильно </li>
</ul>
<ul class="list l2">
<li class="item it1"> Выше </li>
<li class="item"> Тест </li>
<li class="item it2"> Быстрее </li>
<li class="item it3"> Сильнее </li>
</ul>
```

jQuery. События

Общий вид определения обработчиков jQuery:
$(селектор).событие(function() {код_обработчика_события});

- mouseover – когда курсор мыши будет наведен на элемент.
- mouseout – когда курсор мыши будет выведен из границ элемента.
- click – после одинарного щелчка мыши на элементе.
- dblclick – после двойного щелчка мыши на элементе.
- focus – когда элемент станет активным.
- blur – когда элемент перестанет быть активным.
- change – при изменении содержимого элемента.

```
$(document).ready(function( event ){
  $("p").mouseover(function(){$("p").css("color","green")});
  $("p").mouseout(function(){$("p").css("color","black")});
});
```

Букмарклеты

Букмарклет (англ. bookmarklet; bookmark «закладка» и applet «апплет») – небольшая JavaScript-программа, оформленная как javascript: URL и сохраняемая как **браузерная закладка**. Альтернативное название букмарклетов – «favelets» (от слова «Favorites» — названия закладок в браузере «Internet Explorer»). Букмарклеты используются как инструменты, придающие браузеру дополнительную функциональность. Они могут, к примеру:

- поменять внешний вид страницы (цвета, размер букв, и т. д.),
- извлечь данные из страницы, например, все ссылки или все используемые изображения,
- помочь веб-разработчику — показать имена стилей, классов, свойства элементов, произвести операции с cookie.
- укорачивать ссылки
- переводить нужную вам страницу на какой-либо язык
- также букмарклеты могут блокировать определённые элементы на странице (картинки, Flash)

Букмарклеты

```
javascript:void(document.body.style.backgroundColor='gray');
javascript:if(confirm('Continue?'))location.href%20=%20'http://www.sc.com.ua';
```

Для запуска букмарклета из дополнительного файла:

```
javascript:(function(){
 var s=document.createElement('script');
 s.setAttribute('src', 'http://scripts.uadev.net/script.js');
 document.getElementsByTagName('body')[0].appendChild(s);void(s);
})()
```

Для запуска локально расположенного букмарклета:

```
javascript:(function(){
 var s=document.createElement('script');
 s.setAttribute('src', 'file:///D:/bookmarklet.js');
 document.getElementsByTagName('body')[0].appendChild(s);void(s);
})()
```

Работа с файлами

Доступ к пользовательским файлам на стороне клиента запрещен, но использование управлющего элемента <input type="file"> дает веб-странице разрешение на доступ.

<input type="file">

HTML5 определяет файловые ссылки для всех <input type="file"> в виде коллекции FileList, содержащей объекты типа File для каждого выбранного файла в поле <input type="file">.

Тип File определен в спецификации **File API** и является абстрактным представлением файла. Каждый экземпляр File имеет следующие свойства:

name – имя файла

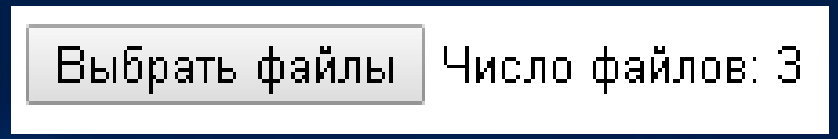
size – размер файла в байтах

type – MIME тип файла.

Объект типа File дает информацию о файле, не предоставляя прямой доступ к содержимому.

Работа с файлами. Получение свойств файлов

```
window.onload = function() {
  var control = document.getElementById("your-files");
  control.addEventListener("change", function(event) {
    // происходит изменение – значит, появились новые
файлы
    var i = 0, files = control.files, len = files.length;
    for (; i < len; i++) {
      console.log("Filename: " + files[i].name);
      console.log("Type: " + files[i].type);
      console.log("Size: " + files[i].size + " bytes");
    }
  });
}
```




```
Filename: test.xml
Type: text/xml
Size: 301 bytes
Filename: test.html
Type: text/html
Size: 1300 bytes
Filename: test.txt
Type: text/plain
Size: 4 bytes
```

Работа с файлами. FileReader

FileReader предназначен для чтения данных из файла и сохранения их в переменной JavaScript. Чтение осуществляется асинхронно, чтобы не блокировать браузер.

Чтение осуществляется с помощью вызова одного из следующих методов:

readAsText() – возвращает содержимое файла как plain text

readAsBinaryString() – возвращает содержимое файла в виде строки закодированных двоичных данных (устарел – вместо него readAsArrayBuffer())

readAsArrayBuffer() – возвращает содержимое файла как ArrayBuffer (рекомендуется для двоичных данных)

readAsDataURL() – возвращает содержимое файла как data URL.

Вы должны установить обработчик загрузки событие onload, прежде чем начать считывать содержимое файла. Результат чтения всегда представлены как event.target.result.

Работа с файлами. FileReader


```
window.onload = function() {
  if (window.File && window.FileList && window.FileReader) {
    console.log("File API OK");
    var control = document.getElementById("your-files");
    control.addEventListener("change", readSingleFile);
  }
  else {
    alert('The File APIs are not fully supported by your browser.');
  }
}
```

```
function readSingleFile(event) {
    var file = event.target.files[0];
    if (file) {
      var reader = new FileReader();
      reader.onload = function(e) {
        var contents = e.target.result;
        console.log("File content: " + contents);
      }
      reader.readAsText(file);
    } else {
      alert("Failed to load file");
    }
  }
```

```
File API OK
File content: test
Привет, мир!
```

Объекты

Объекты в JavaScript используются в качестве ассоциативных массивов и для реализаций возможностей ООП. Синтаксис создания пустого объекта: new Object(); либо {};

Объект может содержать в себе любые значения (в том числе и другие объекты) – свойства. Доступ к свойствам осуществляется по имени свойства («по ключу»): объект.свойство.

```
var menu = {
  width: 300,
  'height': 200,
  title: "Menu"
};
```

или

```
var menu = {};
menu.width = 300;
menu.height = 200;
menu['title'] = 'Menu';
```

```
for (var key in menu) {
  console.log( "Ключ: " + key + " значение: " + menu[key] );
}
```

```
Ключ: width значение: 300
Ключ: height значение: 200
Ключ: title значение: Menu
```

Объекты. Использование конструктора

Обычный синтаксис {...} не подходит, когда при создании свойств объекта нужны более сложные вычисления, требующие применения функции-конструктора. Конструктором становится любая функция, вызванная через new. Такая функция создает новый пустой объект {}, писваивает this ссылку на этот объект, добавляет объекту (возможно) свойства и методы, возвращает this.

return не нужен.

```
function Menu(width, height) {
  this.width = width;
  this.height = height;
  this['title'] = 'Menu';
}
var menu= new Menu(300, 200);
```

```
Ключ: width значение: 300
Ключ: height значение: 200
Ключ: title значение: Menu
```

Объекты. Управление свойствами

Основной метод для управления свойствами – Object.defineProperty. Этот метод позволяет объявить свойство объекта и настроить его параметры.

Object.defineProperty(obj, prop, descriptor)

obj – объект, в котором объявляется свойство.

prop – имя свойства, которое нужно объявить или модифицировать.

descriptor – дескриптор – объект, который описывает поведение свойства. В нём могут быть следующие поля:

value – значение свойства, по умолчанию undefined;

writable – значение свойства можно менять, если true. По умолчанию false;

configurable – если true, то свойство можно удалять, а также менять его в дальнейшем при помощи новых вызовов defineProperty. По умолчанию false.

enumerable – если true, то свойство просматривается в цикле for..in и методе Object.keys(). По умолчанию false.

get / set– функция, которая возвращает/устанавливает значение свойства. По умолчанию undefined.

Объекты. Управление свойствами. Пример

```
var menu = {
  width: 300,
  title: "Menu"
};

Object.defineProperty(menu, "height", { value: 200, configurable: true, writable:
false, enumerable: true });
menu.height=202;

for (var key in menu) {
  console.log( "Ключ: " + key + " значение: " + menu[key] );
}
alert( Object.keys(menu) ); // width, title, height (если enumerable: true)
alert( Object.getOwnPropertyNames(menu) ); // width, title, height
```

```
Ключ: width значение: 300
Ключ: title значение: Menu
Ключ: height значение: 200
```

Объекты. Создание методов

Методы определяют действия, которые могут быть совершены над объектами. В функции-конструкторе можно объявить локальные переменные и вложенные функции, которые будут видны только внутри. Публичные методы определяются при помощи ключевого слова this.

```
function User(firstName, lastName) {
 var phrase = "Привет";
 function getFullName() {
    return firstName + " " + lastName;
  }
 this.sayHi = function() {
  alert( phrase + ", " + getFullName() );
 };
}
var vasya = new User("Вася", "Петров");
vasya.sayHi(); // Привет, Вася Петров
```

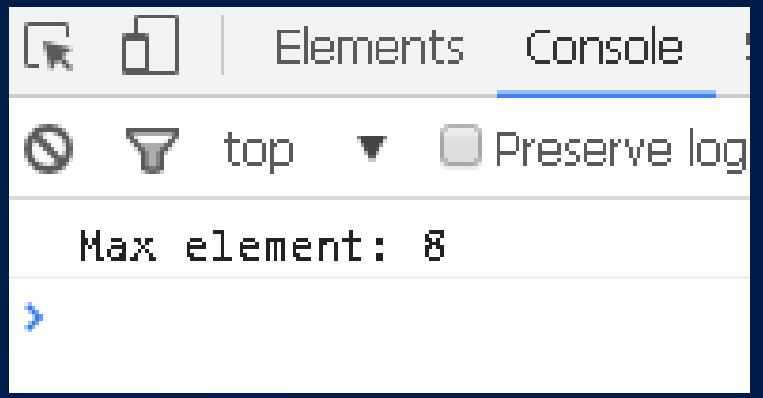
В любой **объект** в любое время можно **добавить новый метод** или удалить существующий:

```
vasya.sayHiNew = function() {
   alert( "Привет, Василий Петров");
 };
vasya.sayHiNew(); // Привет, Василий Петров
```

Объекты. Прототип

```
function array_max()
{
 var i, max = this[0];
 for (i = 1; i < this.length; i++)
 {
  if (max < this[i])
   max = this[i];
 }
 return max;
}
Array.prototype.max = array_max;
var x = [ 1, 2, 8, 4, 5, 6];
var y = x.max( );
console.log(y);
```

Свойство prototype используется для предоставления базового набора функциональных возможностей классу объектов.



Объекты. Инкапсуляция

Инкапсуляция является одним из принципов объектно-ориентированного программирования, заключается в изоляции данных в экземпляре класса от данных в другом экземпляре того же самого класса.

```
function MyClass() {
 this.MyData = "Some Text";
}
MyClass.prototype.MyFunction = function(newtext) {
 this.MyData = newtext;
 console.log("New text:\n"+this.MyData);
}
var c = new MyClass();
c.MyFunction("Some More Text");
var c2 = new MyClass();
c2.MyFunction("Some Different Text");
```

```
New text:
Some More Text
New text:
Some Different Text
```

Объекты. Наследование

Каждый объект имеет внутреннюю ссылку на другой объект, называемый его **прототипом**. Новые экземпляры объекта наследуют свойства и методы прототипа, присвоенного этому объекту.

```
function Animal(name) {
   this.name = name;
   this.canWalk = true;
}
function Rabbit(name) {
          this.name = name;
}
var animal = new Animal("животное");
Rabbit.prototype = animal;
```

```
big = new Rabbit('Chuk');
small = new Rabbit('Gek');

console.log(big.name);  // Chuk
console.log(small.name); // Gek
console.log(big.canWalk); // true

animal.canWalk = false;
console.log(big.canWalk);  // false
console.log(small.canWalk);  // false
```

Объекты. Наследование. Пример

```
function Person(name, age) {
   this.name = name;
   this.age = age;
}
var vasya = new Person("Vasya", 19);
function Student(univer) {
          this.univer = univer;
}
Student.prototype = vasya;

var studvasya = new Student("HNURE");
for (var key in studvasya) {
  console.log( "Ключ: " + key + " значение: " + studvasya[key] );
}
```

```
Ключ: univer значение: HNURE
Ключ: name значение: Vasya
Ключ: age значение: 19
```

Объекты. Полиморфизм

```
//Конструктор родительского класса
function Animal(name) {
 this.name = name;
}
Animal.prototype.speak = function() {
 console.log(this.name + " says:");
}
```

```
Шарик says:
Гав
Матроскин says:
Мяу
```

```
//Конструктор класса "Dog"
function Dog(name) {
 Animal.call(this, name);
}
Dog.prototype.speak = function() {
 Animal.prototype.speak.call(this);
 console.log("Гав");
}
```

```
//Конструктор класса "Cat"
function Cat(name) {
 Animal.call(this, name);
}
Cat.prototype.speak = function() {
 Animal.prototype.speak.call(this);
 console.log("Мяу");
}
```

```
var d = new Dog("Шарик"); d.speak();
var c = new Cat("Матроскин"); c.speak();
```

Вопросы

- Как избежать преждевременного выполнения кода JavaScript?
- Библиотека jQuery, способы подключеня и различия версий.
- Как получить элемент DOM при помощи jQuery?
- Какой метод jQuery позволяет добавить содержимое в конце заданного элемента?
- Проблемы использования букмарклетов.
- Возможно ли чтение из файла, расположенного локально на стороне клиента?
- Способы создания объектов в языке JavaScript.
- Для чего служит дескриптор?
- Как происходит добавление свойств и методов в объект?
- В чем заключается инкапсуляция объектов в языке JavaScript?
- Как реализуется наследование?
- Что такое полиморфизм? Как он реализуется в языке JavaScript?