Internet-технологии

ЛЕКЦИЯ №8

ЯЗЫК НАПИСАНИЯ СЦЕНАРИЕВ JAVASCRIPT

Содержание

- Букмарклеты.
- Работа с файлами.
- ООП. Объекты, область видимости, прототипы, наследование.

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

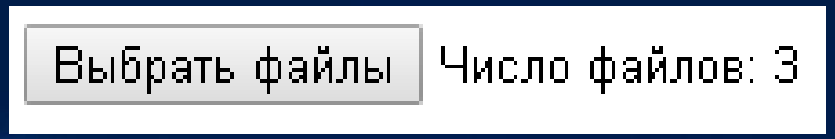
size – размер файла в байтах

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
```

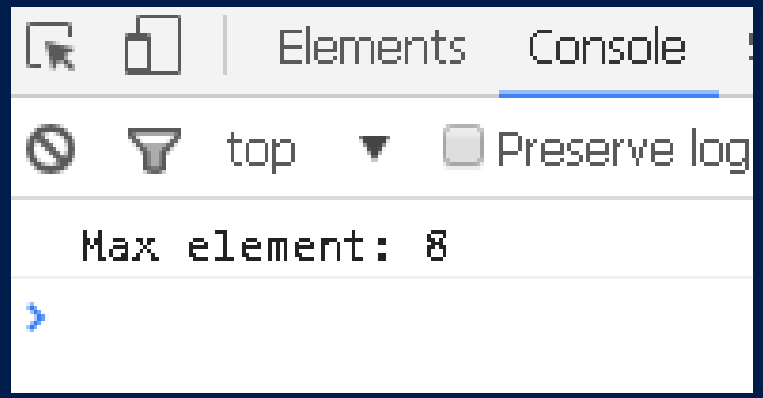
В любой **объект** в любое время можно **добавить новый метод** или удалить существующий:

```
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
```

Вопросы

- 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?