Javascript

Синтаксис: переменные, операторы, ветвления и циклы

Динамическая типизация: var k = 3; k = “blabla”;

Операторы, ветвления и циклы такие же, как в C++ или Java.

Функции

- Вызов функций

```
myFunc(arg1, arg2, ...);
```

- Объявление функций с помощью ключевого слова ‘function’

– Нет ни типа возвращаемого значания, ни типов аргументов

```
function add(numOne, numTwo) {
        return numOne + numTwo;
}
```

Ввод/вывод

Возможен вывод прямо в html

- Без перевода строки
document.write(“I am <B>BOLD</B>”);
– С переводом строки
document.writeln(“I am <U>underlined</U>”);

Логирование:

Console.log(“blabla”);

Объекты

- Не объектно-ориентированный
- Есть объекты, которые записываются, как функции.

```
function Person(myName, myAge) {
    this.name = myName;
    this.age = myAge;
}

var someGuy = new Person(“Shawn”, 28);
```

Объекты

• Доступ к свойствам объекта

```
var someGuy = new Person(“Shawn”, 28);
document.writeln(‘Name: ‘ + someGuy.name);
```

• Объекты и ассоциативные массивы – одно и то же, что значит, доступ к свойствам объекта возможен, как к элементам массива

```
someGuy[“age”] or someGuy[“name”]
document.writeln(‘Name: ‘ + someGuy[“name”]);
```

Объекты

• Методы классов
– Функции класса (объекта) записываются, как поля.

```
function displayName() {
        document.writeln(“I am “ + this.name);
}
```

Конструктор будет выглядеть так:

```
function Person(myName, myAge) {
    this.name = myName;
    this.age = myAge;
    this.displayMe = displayName;
}
```

Прототипы

- Свойства и функции можно задавать элементу извне с помощью ключевого слова prototype

```
Person.prototype.myfunc = function() {
        …
}

Person.prototype.height = 180;
```

С помощью этого механизма обеспечивается полиморфизм.

Наследование

- Нет встроенного наследования

- Runtime наследование: клонируем объекты и добавляем дополнительные свойства (с помощью prototype)

this имеет такое же значение, как и в C++

Встроенные объекты

Как и в C++, в javascript есть встроенные объекты, которые вам не надо определять самим:

- Number, Boolean, String
- var str = “abc”;
- var is a String object
- Date
- Math

Массивы

• Создание массивов

```
var a = new Array();

var b = new Array(“dog”, 3, 8.4);
var c = new Array(10); // array of size 10
var d = [2, 5, ‘a’, ‘b’];
```

• Присвоение значение

```
c[15] = “hello”;
c.push(“hello”);
```

• Ассоциативные массивы

```
var e = new Array();
e[“key”] = “value”;
e[“numKey”] = 123;
```

Подключение к HTML

– Напрямую (в HTML файле)

```
<script type="text/javascript">
...code here...
</script>
```

– В отдельном файле

```
<script type="text/javascript" src="test.js">
</script>
```

• Положение: <HEAD> vs. <BODY>

– Код, расположенный в head можно вызвать из любого места

– Код, который находится в body будет исполнен при парсинге документа

• Лучше использовать первый вариант

Объект Window

Мы уже видели объект ‘document’, когда рассматривали ввод/вывод

• объект ‘window’ – JavaScript представление окна браузера

– closed – свойство, которое говорит о том, открыто ли окно.

– defaultStatus – дефолтное сообщение, которое выводится в статусбаре при загрузке;

Пример Window

Методы:
alert("message")
window.close()
focus()
open("URLname","Windowname",["options"])
open(“http://google.com”, “My Google”,
“toolbar=no,alwaysRaised=yes”);

Свойства:
– height, weight, alwaysRaised, location, menubar, etc..

Свойства браузера

• window.location – адрес, который открыт на текущей странице
– свойство href представляет URL.
– hostname представляет host:port.

window.location.href=“http://google.com”;

• window.history – история посещений
– length представляет число посещений
– history.go(-1)
– history.back()

Объект Form

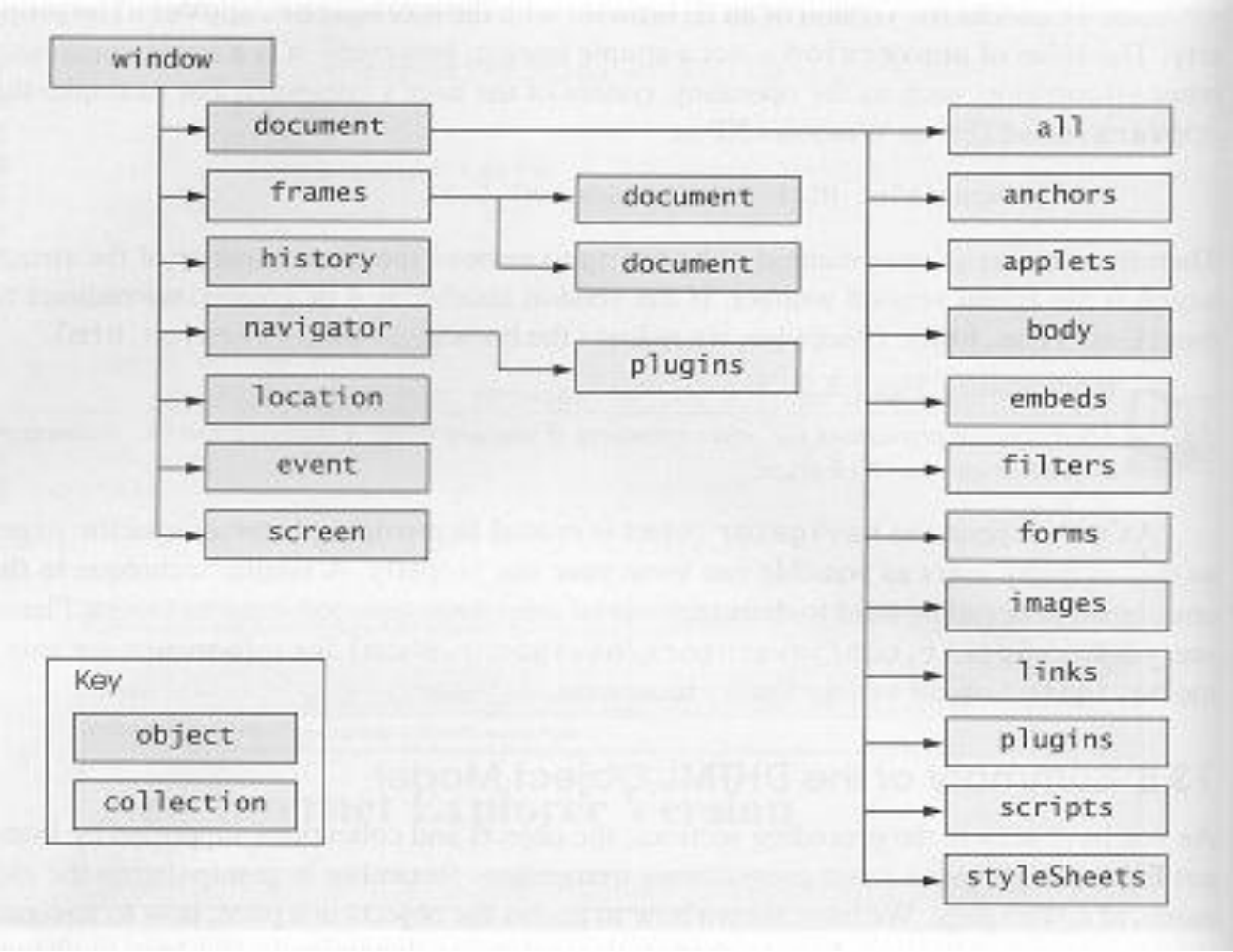
• Объекты формы могут быть доступны через:
window.document.myForm или
window.document.forms[0]

• Свойства:
– action, target, length, method, etc...

• Функции
– window.document.myForm.submit();
– window.document.myForm.reset();

• Доступ к значениям полей
– window.document.myForm.firstname.value

(D)HTML иерархия объектов



Обработчики событий

• События появляются, как результат того, что пользователь что-то сделал. Например, кликнул мышкой или изменил текстовое поле.

• Примеры событий: Click, change, focus, load, mouseover, mouseout, reset, submit, select

Обработчики событий

• Вы можете использовать обработчики событий, такие, как onChange и onClick, чтобы заставить ваши скрипты реагировать на события.

<input type=“button” onClick=“javascript:doButton()>
<select onChange=“javascript:doChange()”>
<a onClick=“javascript:doSomething()”>
<form onSubmit=“javascript:validate()”>
<body onLoad=“javascript:init()”>

DOM – Объектная модель документа

• Это метод доступа/изменения XML информации на странице, то есть, древовидной структуры всех элементов, включая атрибуты и их значения.

Их содержание может быть изменено или удалено. Можно создать новые DOM элементы, и вставить их в страницу.

• Эта концепция используется и в других языках программирования (google: Java DOM)

Объект Document

• Объект document – это JavaScript интерфейс к DOM-модели любой HTML страницы.

• Доступ к элементам:
– По имени
document.getElementsByTagName(‘td')[indexOfColumn]

– По ИД
document.getElementById('id')

– Произвольный элемент дерева достать можно так:
document.childNodes[0].childNodes[1].childNodes[4]

Регулярные выражения

• Объект RegExp

var beerSearch = /beer/

var beerSearch = new RegExp(“beer”)

• Методы

– Искать паттерн и вернуть результаты в массиве

exec(str)

– Вернет true, если сматчится

test(str)

```
if (beerSearch.test(“beer tastes bad”)) {
alert(“I found a beer”);
}
```

Литература

- http://www.w3schools.com/js/default.asp
- http://wp.netscape.com/eng/mozilla/3.0/handbook/javascript/index.html