

ВВЕДЕНИЕ В JAVASCRIPT ДЛЯ МАГА

© 1996, 1997 Стефан Кох (Stefan Koch)

Часть 6: Предопределенные объекты

Объект Date

В JavaScript Вам разрешено пользоваться некоторыми заранее заданными объектами. Примерами таких объектов могут служить Date, Array или Math. Есть еще несколько таких же объектов - полное описание см. в документации, предоставляемой фирмой Netscape.

Для начала давайте рассмотрим объект Date. Судя по названию, он позволяет Вам работать как со временем, так и с датой. Например, Вы можете легко определить, сколько дней еще остается до следующего рождества. Или можете внести в Ваш HTML-документ запись текущего времени.

Так что давайте начнем с примера, который высвечивает на экран текущее время. Сперва мы должны создать новый объект Date. Для этого мы пользуемся оператором new:

```
today= new Date()
```

Здесь создается новый объект Date, с именем today. Если при создании этого нового объекта Date Вы не указали какой-либо определенной даты и времени, то будут предоставлены текущие дата и время. То есть, после выполнения команды *today= new Date()* вновь созданный объект *today* будет указывать именно те дату и время, когда данная команда была выполнена.

Объект Date предоставляет нам кое-какие методы, которые теперь могут применяться к нашему объекту today. Например, это методы - *getHours()*, *setHours()*, *getMinutes()*, *setMinutes()*, *getMonth()*, *setMonth()* и так далее. Полное описание объекта Date и его методов Вы сможете найти в документации по JavaScript, предоставляемой фирмой Netscapes.

Обратите пожалуйста внимание, что объект Date лишь содержит определенную запись о дате и времени. Он не уподобляется часам, автоматически отслеживающим время каждую секунду, либо миллисекунду.

Чтобы зафиксировать какое-либо другие дату и время, мы можем воспользоваться видоизмененным конструктором (это будет метод *Date()*, который при создании нового объекта Date вызывается через оператор *new*):

```
today= new Date(1997, 0, 1, 17, 35, 23)
```

При этом будет создан объект `Date`, в котором будет зафиксировано первое января 1997 года 17:35 и 23 секунд. Таким образом, Вы выбираете дату и время по следующему шаблону:

`Date(year, month, day, hours, minutes, seconds)`

Заметьте, что для обозначения января Вы должны использовать число 0, а не 1, как Вы вероятно думали. Число 1 будет обозначать февраль, ну и так далее.

Теперь мы напишем скрипт, печатающий текущие дату и время. Результат будет выглядеть следующим образом:

`Time: 17:53`
`Date: 4/3/2010`

Сам же код выглядит следующим образом:

```
<script language="JavaScript">
<!-- hide

now= new Date();

document.write("Time: " + now.getHours() + ":" + now.getMinutes() + "<br>");
document.write("Date: " + (now.getMonth() + 1) + "/" + now.getDate() + "/" +
    (1900 + now.getYear()));

// -->
</script>
```

Здесь мы пользуемся такими методами, как `getHours()`, чтобы вывести на экран время и дату, указанные в объекте `Date` с именем `now`. Можно видеть, что мы добавляем к записи года еще число 1900. Дело в том, что метод `getYear()` указывает количество лет, прошедших после 1900 года. А стало быть, если сейчас 1997 год, то будет выдано значение 97, а если 2010 год - то 110, а не 10! Если мы так и будем всякий раз добавлять 1900, то у нас не будет проблемы 2000 года. Помните также, что мы обязаны увеличивать на единицу значение, получаемое от метода `getMonth()`.

В данном скрипте не выполняется проверки на тот случай, если количество минут окажется меньше, чем 10. Это значит, что Вы можете получить запись времени примерно в следующем виде: 14:3, что на самом деле должно было бы означать 14:03. Решение этой проблемы мы рассмотрим в следующем примере.

Рассмотрим теперь скрипт, создающий на экране изображение работающих часов:

```
<html>
<head>

<script Language="JavaScript">
<!-- hide

var timeStr, dateStr;
```

```

function clock() {
    now= new Date();

    // время
    hours= now.getHours();
    minutes= now.getMinutes();
    seconds= now.getSeconds();
    timeStr= "" + hours;
    timeStr+= ((minutes < 10) ? ":0" : ":") + minutes;
    timeStr+= ((seconds < 10) ? ":0" : ":") + seconds;
    document.clock.time.value = timeStr;

    // дата
    date= now.getDate();
    month= now.getMonth()+1;
    year= now.getYear();
    dateStr= "" + month;
    dateStr+= ((date < 10) ? "/0" : "/") + date;
    dateStr+= "/" + year;
    document.clock.date.value = dateStr;

    Timer= setTimeout("clock()",1000);
}

// -->
</script>
</head>

<body onLoad="clock()">

<form name="clock">
    Время:
    <input type="text" name="time" size="8" value=""><br>
    Дата:
    <input type="text" name="date" size="8" value="">
</form>

</body>
</html>

```

(online-версия руководства позволит Вам проверить этот скрипт немедленно)

Здесь для ежесекундной коррекции времени и даты мы пользуемся методом `setTimeout()`. Фактически это сводится к тому, что мы каждую секунду создаем новый объект `Date`, занося туда текущее время. Можно видеть, что функции `clock()` вызываются программой обработки события `onLoad`, помещенной в тэг `<body>`. В разделе `body` нашей HTML-страницы имеется два элемента формы для ввода текста. Функция `clock()` записывает в оба эти элемента в корректном формате текущие время и дату. Для этой цели используются две строки `timeStr` и `dateStr`.

Как мы уже упомянули ранее, существует проблема с индикацией, когда количество минут меньше 10 - в данном скрипте эта проблема решается с помощью следующей строки:

```
timeStr+= ((minutes < 10) ? ":0" : ":") + minutes;
```

Как видим, количество минут заносится в строку *timeStr*. Если у нас менее 10 минут, то мы еще должны приписать спереди 0. Для Вас эта строка в скрипте может показаться немного странной, и ее можно было бы переписать в более знакомом Вам виде:

```
if (minutes < 10) timeStr+= ":0" + minutes  
else timeStr+= ":" + minutes;
```

Объект Array

Массивы играют в программировании очень важную роль. Подумайте только, что бы Вы делали, если бы Вам понадобилось хранить 100 различных имен. Как бы Вы могли это сделать с помощью JavaScript? Хорошо, Вы могли бы явным образом задать 100 переменных и присвоить им различные имена. Но согласитесь, это будет весьма утомительно.

Массив может быть полезен там, где имеется много взаимосвязанных переменных. При этом к каждой из них Вы можете получить доступ, воспользовавшись общим названием и неким номером.

Допустим, есть массив в именем *names*. В этом случае мы можем получить доступ к первой переменной с именем *name*, написав *names[0]*. Вторая переменная носит *name[1]* и так далее.

Начиная с версии 1.1 языка JavaScript (Netscape Навигатор 3.0), Вы можете использовать объект *Array*. Вы можете создать новый массив, записав *myArray= new Array()*. После этого можно начать заносить в массив значения:

```
myArray[0]= 17;  
myArray[1]= "Stefan";  
myArray[2]= "Koch";
```

Массивы JavaScript обладают большой гибкостью. Например, Вам нет нужды беспокоиться о размере массива - он устанавливается динамически. Если Вы напишете *myArray[99]= "xyz"*, размер массива будет установлен 100 элементов. (В языке JavaScript размер массива может только увеличиваться - массив не может "сжиматься". Поэтому старайтесь делать Ваши массивы как можно компактнее.) Не имеет значения, заносите ли Вы в массив числа, строки, либо другие объекты. Я не останавливаюсь на каждой такой подробности структуры массивов, но надеюсь, Вы поймете, что массивы - очень важный элемент языка.

Конечно же многое станет понятнее, если рассматривать примеры. Следующий скрипт печатает следующий текст:

```
first element  
second element  
third element
```

Исходный код:

```
<script language="JavaScript">
<!-- hide

myArray= new Array();

myArray[0]= "first element";
myArray[1]= "second element";
myArray[2]= "third element";

for (var i= 0; i< 3; i++) {
    document.write(myArray[i] + "<br>");
}

// -->
</script>
```

Первым делом мы создаем здесь новый массив с именем *myArray*. Затем мы заносим в него три различных значения. После этого мы запускаем цикл, который трижды выполняет команду *document.write(myArray[i] + "
");*. В переменной *i* ведется отсчет циклов от 0 до 2. Заметим, что в цикле мы пользуемся конструкцией *myArray[i]*. И поскольку *i* меняет значения от 0 до 2, то в итоге мы получаем три различных вызова *document.write()*. Иными словами, мы могли бы расписать этот цикл как:

```
document.write(myArray[0] + "<br>");
document.write(myArray[1] + "<br>");
document.write(myArray[2] + "<br>");
```

Массивы в JavaScript 1.0

Поскольку в JavaScript 1.0 (Netscape Navigator 2.x, и Microsoft Internet Explorer 3.x) объекта *Array* еще не существовало, то мы должны думать и об его альтернативе. Следующий фрагмент кода можно найти в документации фирмы Netscape:

```
function initArray() {
    this.length = initArray.arguments.length
    for (var i = 0; i < this.length; i++)
        this[i+1] = initArray.arguments[i]
}
```

После этого Вы можете создавать массив одной строкой:

```
myArray= new initArray(17, 3, 5);
```

Числа в скобках - значения, которыми инициализируется массив (это можно также делать и с объектом *Array* из JavaScript 1.1). Обратите внимание, что данный тип массива не может включать все элементы, которые являются частью в объекта *Array* от

JavaScript 1.1 (например, там имеется метод `sort()`), который позволяет сортировать все элементы в определенном порядке).

Объект Math

Если Вам необходимо в скрипте выполнять математические расчеты, то некоторые полезные методы для этого Вы найдете в объекте Math. Например, имеется метод синуса *sin()*. Полную информацию об этом объекте Вы найдете в документации фирмы Netscape. Я бы хотел продемонстрировать работу метода *random()*. Если Вы в свое время читали первый выпуск этого материала, то знаете, что у нас были некоторые проблемы с методом *random()*. Тогда мы написали функцию, позволяющую генерировать случайные числа. Теперь, чтобы работать на всех без исключения платформах, нам не нужно ничего, кроме метода *random()*.

Если Вы вызовете функцию *Math.random()*, то получите случайное число, лежащее в диапазоне между 0 и 1. Один из возможных результатов вызова *document.write(Math.random())*:

.7184317731538611

©1996,1997 by Stefan Koch

e-mail: skoch@rumms.uni-mannheim.de

<http://rummelplatz.uni-mannheim.de/~skoch/>

Моя книга по JavaScript: <http://www.dpunkt.de/javascript>