Встроенные объекты языка JavaScript

String	Объект предназначен для работы со строками. Содержит свойства и методы, позволяющие изменять строку в целом или отдельные её символы, менять регистр, искать соответствие по маске или точному совпадению.
Array	Объект предназначен для хранения и управления наборами формализованных данных. Под формализованными, в данном случае понимаются данные, которые могут быть приведены к одному из типов данных JavaScript (например: строки, числа и объекты).
Math	Объект содержит все основные математические константы, а так же предоставляет методы для выполнения типовых математических операций (округление, тригонометрические функции и т.д)
Date	Объект предоставляет свойства и методы для получения и манипулирования датами и временем.
Number	Содержит базовые константы и методы работы с числовыми данными. Так же содержит ряд специальных значений, определяющих критические или нечисловые значения (например, Infinity (бесконечность) и NaN (не-число))
Boolean	Объект-оболочка для простых логических типов данных и операций над ними. В работе используется очень редко, т.к. не содержит практически никаких полезных свойств или методов.
Function	Объект представляет собой строку, которая при выполнении компилируется в функцию. Объект позволяет работать с параметрами и телом такой функции "на лету", во время выполенния скрипта.
Object	Объект-родитель всех объектов JavaScript. Содержит все базовые методы, имеющиеся в любом объекте (например, toString()). Обычно этот объект в явном виде не используется.

Объект Date

Объект содержит информацию о дате и времени.

dateObj = new Date(parameters)

где dateObj - переменная, в которую будет записан новый объект Date.

Значения аргумента:

- ✔пустой параметр, например date() в данном случае дата и время - системные.
- ✔строку, представляющую дату и время в виде: "месяц, день, год, время", например "March 1, 2013, 17:00:00" Время представлено в 24-часовом формате;
- ✔значения года, месяца, дня, часа, минут, секунд. Например, строка "2013,4,1,12,30,0" означает 1 апреля 2013 года, 12:30.
- ✔целочисленные значения только для года, месяца и дня, например "2013,5,1" означает 1 мая 2013 года, сразу после полночи, так, как значения времени равны нулю.

Методы объекта Date

✔**getDate()** - Возвращает день месяца из объекта в пределах от 1 до 31

✔**getDay()** - Возвращает день недели из объекта: 0 - вс, 1 - пн, 2 - вт, 3 - ср, 4 - чт, 5 - пт, 6 - сб.

✔**getHours()** - Возвращает время из объекта в пределах от 0 до 23

✔**getMinutes()** - Возвращает значение минут из объекта в пределах от 0 до 59

✔**getMonth()** - Возвращает значение месяца из объекта в пределах от 0 до 11

✔**getSeconds()** - Возвращает значение секунд из объекта в пределах от 0 до 59

✔**getYear()** - Возвращает значение года из объекта

- **setDate(day)** - устанавливается день месяца в объекте от 1 до 31
- **setHours(hours)** - устанавливается часы в объекте от 0 до 23
- **setMinutes(minutes)** - метода устанавливаются минуты в объекте от 0 до 59
- **setMonth(month)** - устанавливается месяц в объекте от 1 до 12
- **setSeconds(seconds) -** устанавливаются секунды в объекте от 0 до 59
- **setYear(year)** - устанавливается год в объекте year должно быть больше 1900.

Пример

```
<script language "JavaScript">
function showh() {
  var theDate = new Date();
  document.writeln("<table cellpadding=5 width=100%
border=0>" +
      "<tr><td width=95% bgcolor=gray align=left>" +
      "<font color=white>Date: " + theDate +
      "</font></td></tr></table><p>");
}
showh();
</script>
```

Объект Math

Содержит свойства и методы, используемые для выполнения математических операций. Объект Math включает также некоторые широко применяемые математические константы.

Свойства

Свойствами объекта Math являются константы:

E	Константа Эйлера. Приближенное значение 2.718 . . .
LN2	Значение натурального логарифма числа два. Приближенное значение 0.693 . . .
LN10	Значение натурального логарифма числа десять. Приближенное значение 2.302 . . .
LOG2_E	Логарифм *e* по основанию 2 (не вижу смысла в этой константе - это же корень из двух.) Приближенное значение 1.442 . . .)
LOG10_E	Десятичный логарифм *e*. Приближенное значение 0.434 . . .
PI	Число ПИ. Приближенное значение 3.1415 . . .
SQRT2	Корень из двух, (ыгы, это все равно, как еще и натуральный логарифм 2**e* в степени 1/2)

Методы

Методы объекта Math представляют собой математические функции.

- **abs**() Возвращает абсолютное значение аргумента.
- **acos**()Возвращает арккосинус аргумента
- **asin**() Возвращает арксинус аргумента
- **atan**()Возвращает арктангенс аргумента
- **ceil**() Возвращает большее целое число аргумента, округление в большую сторону. Math.ceil(3.14) вернет 4
- **cos**() Возвращает косинус аргумента
- **exp**() Возвращает экспоненту аргумента
- **floor**() Возвращает наибольшее целое число аргумента, отбрасывает десятичную часть

- **log**() Возвращает натуральный логарифм аргумента
- **max**()Возвращает больший из 2-х числовых аргументов. Math.max(3,5) вернет число 5
- **min**() Возвращает меньший из 2-х числовых аргументов.
- **pow**()Возвращает результат возведения в степень первого аргумента вторым. Math.pow(5,3) вернет 125
- **round**() Округление аргумента до ближайшего целого числа.

- **sin**() Возвращает синус аргумента
- **sqrt**() Возвращает квадратный корень аргумента
- **tan**() Возвращает тангенс аргумента

Строковые объекты. Объект String

Строки можно создавать тремя способами:

1. Создать строковую переменную при помощи оператора var и присвоить ей строковое значение;
2. Присвоить значение строковой переменной только посредством оператора присваивания (=);
3. Использовать конструктор String().

```
var str=“GJRC”;
st=“ПОКС”;
str1=new String(“Привет”);
```

Свойства

Значением свойства length является длина строки. Например, выражение "Script".length вернет значение 6, поскольку строка "Script" содержит 6 символов.

Методы

✔**ig()** - Аналогично тегам HTML <big> . . .</big>. позволяет отобразить более крупным шрифтом.

✔**bold()** - делает символы жирными.

✔**charAt(arg)** - Возвращает символ, находящийся в заданной позиции строки.

Пример:

vpos="Sc<FONT COLOR="#FF0000">ript".charAt(3); переменной vpos будет присвоено значение "r".

✔**fontcolor(arg)** - Аналогично . . ..

✔**fontsize(arg)** - Позволяет изменять размер шрифта.

- **indexOf(arg1[,arg2])** - Возвращает позицию в строке, где впервые встречается символ - arg1, неябзательный числовой аргумент arg2 указывает начальную позицию для поиска.
- **italics() -** Аналогично тегам HTML . . .. позволяет отобразить италиком.
- **lastIndexOf(arg1[,arg2]) -** Возвращает либо номер позиции в строке, где в последний раз встретился символ - arg1, либо -1, если символ не найден. Arg2 задает начальную позицию для поиска.
- **link() -** Аналогично тегам HTML <a href> . . ..

- **small() -** Аналогично тегам HTML <small> . . .</small>. позволяет отображать строку мелким шрифтом.
- **sub() -** Аналогично тегам HTML _{. . .}. позволяет отображать строку нижним индексом.
- **substring(arg1,arg2)** - Позволяет извлечь подстроку длинной arg2, начиная с позиции arg1
- **sup()** - Аналогично тегам HTML ^{. . .}. позволяет отображать строку верхним индексом.
- **toLowerCase()** - Преобразует символы строкового объекта в строчные
- **toUpperCase()** - Преобразует символы строкового объекта в прописные

Объект Array

Массив – это тип данных, хранящий пронумерованные значения. Каждое пронумерованное значение называется элементом массива, а число, с которым связывается элемент, называется его индексом. Массивы JavaScript нетипизированы, это значит, что элемент массива может иметь любой тип, причем разные элементы одного массива могут иметь разные типы. Помимо этого массивы JavaScript являются динамическими, это значит, что объявлять фиксированный размер не нужно и можно добавить новые элементы в любое время.

Создание массива

Массив можно создать двумя способами:

1. создать массив с помощью литерала массива - квадратные скобки, внутри которых расположен список элементов, разделенных запятыми.

Пример

```
var empty = [];  //пустой массив
var numbers = [4, 1, 2, 5]; //массив с 4 числовыми элементами
var diff = [1.5, false, "текст"];  //массив с 3 элементами различного типа
```

Значения не обязательно должны быть простыми (числа или строки) - это также могут быть и любые другие выражения, например: литералы объектов, другие массивы или функции.

Пример

```
var num = 700;
var tab = [function(a) { alert(a) }, { name: 'Петя' }, [1, 2, 3], num + 1];
```

2. Вызов конструктора Array().

Вызвать конструктор Array() можно тремя способами:

✔**Вызов конструктора без аргументов:**

```
var b = new Array();
```

В этом случае создается пустой массив, эквивалентный пустому литералу [].

- **В конструкторе явно указываются значения n элементов массива:**

```
var b = new Array(1, 3, 5, 8, "строка", true);
```

В этом случае конструктор получает список аргументов, которые становятся элементами нового массива. Аргументы записываются в массив в том порядке, в котором указаны.

- **Выделение места для последующего присваивания значений.**

Это делается путем указания при определении массива одного числа в круглых скобках:

var b = new Array(5);

Этот способ определения массива предполагает выделение массиву определенного количества элементов (каждый из которых имеет значение undefined) с возможностью последующего присваивания значений по ходу сценария.

Чтение, запись и добавление элементов массива

Доступ к элементам массива осуществляется с помощью оператора []. Элементы массива в JavaScript нумеруются, начиная с нуля. Чтобы получить нужный элемент массива, надо указать его номер в квадратных скобках.

Пример

```
var numbers = [4, 1, 2, 5];
 document.write(numbers[0] + ", "); //первый элемент массива
document.write(numbers[1] + ", "); //второй элемент массива
document.write(numbers[2] + ", "); //третий элемент массива
document.write(numbers[3]);        //четвертый элемент массива
```

Элементы массива можно изменять

```
var numbers = [4, 1, 2, 5];
  numbers[0] = 10; //изменили первый элемент массива - [10, 1, 2, 5]
```

Чтобы добавить новый элемент массива, достаточно присвоить новое значение:

```
var numbers = [4, 1];
 numbers[2] = 7; //стало [4, 1, 7]
```

Примечание: в массивах JavaScript может храниться любое число элементов любого типа.

Длина массива

Все массивы, как созданные с помощью конструктора Array(), так и определенные с помощью литерала массива, имеют специальное свойство **length**, которое возвращает общее число элементов, хранимых в массиве.

```
var v = new Array();  // v.length == 0 (ни один элемент не определен)
v = new Array(1,2,3); // v.length == 3 (определены элементы 0–2)
v = [4, 5];           // v.length == 2 (определены элементы 0 и 1)
 document.write(v.length);
```

Методы Array

1. **concat(массив)** - конкатенация массивов, объединяет два массива в третий массив.

Синтаксис: имя_массива1.concat(массив2)

2. **join(разделитель)** - создает строку из элементов массива с указанным разделителем между ними; является строкой символов (возможно, пустой).
Синтаксис: имя_массива.join(строка)

3. **pop()** - удаляет последний элемент массива и возвращает его значение.

Синтаксис: имя_массива.pop()

Возвращает значение удаленного элемента массива. Данный метод изменяет исходный массив.

4. **push(значение|объект)** - добавляет к массиву указанное значение в качестве последнего элемента и возвращает новую длину массива.

Синтаксис: имя_массива1.push(значение|объект)

Данный метод изменяет исходный массив.

5. **shift()** - удаляет первый элемент массива и возвращает его значение.

Синтаксис: имя_массива.shift ()

Возвращает значение удаленного элемента массива. Данный метод изменяет исходный массив.

6. **unshift (значение|объект)** - добавляет к массиву указанное значение в качестве первого элемента.

Синтаксис: имя_массива. unshift(значение|объект)

Возвращает: ничего. Данный метод изменяет исходный массив.

7. **reverse ()** - изменяет порядок следования элементов массива на противоположный.

Синтаксис: имя_массива.reverse()

Возвращает массив. Данный метод изменяет исходный массив.