

Встроенные объекты Java Script Array String Date

Math

JavaScript-объект Math используется для выполнения математических функций.

- Создать экземпляр этого объекта нельзя; вы просто используете объект Math как есть и вызываете его свойства и методы без создания экземпляра:

```
var pi = Math.PI;
```

Свойства Math:

Объект Math имеет много свойств и методов, реализующих математическую функциональность JavaScript. Все свойства Math являются константами с доступом только по чтению. Вот их список:

- E
- LN2
- LN10
- LOG2E
- LOG10E
- PI
- SQRT1_2
- SQRT2

- Свойство E возвращает основание натуральных логарифмов ("число Эйлера"), равное 2.718281828459045.
- Два других свойства, LN2 и LN10, используются для вычисления натуральных логарифмов.
- Свойство LN2 возвращает натуральный логарифм числа 2.
- Свойство LN10 возвращает натуральный логарифм числа 10.
- Свойства LOG2E и LOG10E используются для вычисления логарифма E по основанию 2 или 10. Результат LOG2E равен 1.4426950408889633, а результат LOG10E равен 0.4342944819032518. Большинство этих свойств используется не часто, если конечно вы не создаете калькулятор или другой математический проект.
- Однако число PI и квадратный корень используются чаще. Метод PI возвращает отношение длины окружности к ее диаметру.
- Значения квадратного корня возвращают два свойства: SQRT1_2 и SQRT2. Первое свойство возвращает квадратный корень из 0.5, а второе - квадратный корень из 2.

Кроме этих свойств, есть несколько методов, которые можно использовать для возврата различных числовых значений. Каждый из этих методов принимает числовое значение и возвращает значение в зависимости от названия метода. К сожалению, названия методов не всегда самоочевидны:

- ▶ `abs`. Абсолютное значение числа.
- ▶ `acos`. Арккосинус.
- ▶ `asin`. Арксинус.
- ▶ `atan`. Арктангенс.
- ▶ `atan2`. Арктангенс, зависящий от двух аргументов.
- ▶ `cos`. Косинус.
- ▶ `exp`. Экспонента.
- ▶ `log`. Натуральный логарифм числа.
- ▶ `pow`. Значение x в степени y .
- ▶ `sin`. Синус.
- ▶ `sqrt`. Квадратный корень.
- ▶ `tan`. Тангенс угла.

Для округления чисел в JavaScript используются три метода: *ceil*, *floor* и *round*.

- ▶ Метод *ceil* возвращает число, округленное до ближайшего большего значения. Этот метод полезен, когда необходимо округлить число в большую сторону до ближайшего целого значения.
- ▶ Метод *floor* имеет противоположную функциональность: он возвращает число, округленное до ближайшего меньшего значения. Этот метод полезен, когда необходимо округлить число в меньшую сторону до ближайшего целого значения.
- ▶ Метод *round* предоставляет обычную функциональность округления, увеличивая или уменьшая число в зависимости от имеющихся десятичных знаков

Последними тремя методами объекта `Math` являются: ***max***, ***min*** и ***random***.

- ▶ Метод ***max*** принимает несколько числовых аргументов и возвращает наибольшее значение.
- ▶ Метод ***min*** возвращает наименьшее значение. Эти методы могут быть полезны при сравнении переменных с числовыми значениями, особенно когда эти числовые значения неизвестны.
- ▶ Метод ***random*** возвращает случайное число в диапазоне от 0 до 1.

Объект Array

- ▶ В отличие от других языков программирования, JavaScript не имеет такого типа данных, как массив. Но это ограничение обходится благодаря тому, что можно использовать predetermined object of array - Array.

Для создания объекта-массива можно использовать один из следующих вариантов синтаксиса:

ИмяМассива = new Array(элемент1,
элемент2, ... элементN) ИмяМассива = new
Array(ДлинаМассива)

- ▶ В первом случае перечисляются все составляющие массива, во втором - просто указывается количество элементов. Допускается также использование литералов при объявлении массива:
- ▶ `computers = ["PC", "Mac", "Sun"];`
- ▶ Для заполнения элементов массива значениями, как и вообще для обращения к элементам массива, можно использовать индекс элемента. При этом следует учитывать, что индекс элементов массива начинается с нуля:
- ▶ `var colors = new Array(3); colors[0] = "Красный"; colors[1] = "Синий"; colors[2] = "Зеленый";`
- ▶ Довольно часто бывает удобно использовать предоставляемую языком JavaScript возможность заполнения массива непосредственно при его объявлении:
- ▶ `var colors = new Array("Красный", "Синий", "Зеленый");`
- ▶ Чтобы узнать длину массива (количество элементов, из которых состоит массив), следует использовать свойство `length`:
- ▶ `var NumColors = colors.length;`

- ▶ Помимо свойства ***length***, в JavaScript предусмотрен также целый ряд других свойств и методов для работы с массивами.

В частности, к числу свойств объекта Array, помимо `length`, относятся универсальные для всех объектов `constructor` и `prototype`, а так же предназначенные для использования массивов совместно с регулярными выражениями свойства `index` и `input`.

- ▶ Что касается методов, то помимо стандартных `toSource`, `toString` и `valueOf`, массивы наделены еще десятком собственных, перечисленных в таблице 4.11.

Таблица 4.11. Методы объекта Array

Метод	Описание
concat	Объединяет два массива, и возвращает новый
join	Объединяет все элементы массива в одну строку
pop	Удаляет последний элемент из массива, и возвращает его
<i>Push</i>	Добавляет один или более элементов в конец массива и возвращает его новую длину
reverse	Перемещает элементы массива таким образом, что первый становится последним, и наоборот
shift	Удаляет первый элемент массива и возвращает его
slice	Удаляет часть элементов массива, и возвращает новый массив
splice	Добавляет и (или) удаляет элемент из массива
sort	Сортирует элементы массива по алфавиту
unshift	Добавляет один или более элементов в начало массива, и возвращает новую длину массива (в MSIE 5.5 и 6 этот метод ничего не возвращает)

Таблица 4.11. Методы объекта Array

Метод	Описание
► concat	Объединяет два массива, и возвращает новый
► join	Объединяет все элементы массива в одну строку
► pop	Удаляет последний элемент из массива, и возвращает его
► push	Добавляет один или более элементов в конец массива и возвращает его новую длину
► reverse	Перемещает элементы массива таким образом, что первый становится последним, и наоборот
► Shift	Удаляет первый элемент массива и возвращает его
► slice	Удаляет часть элементов массива, и возвращает новый массив
► Splice	Добавляет и (или) удаляет элемент из массива
► sort	Сортирует элементы массива по алфавиту
► unshift	Добавляет один или более элементов в начало массива, и возвращает новую длину массива (в MSIE 5.5 и 6 этот метод ничего не возвращает)

Date

JavaScript-объект Date позволяет работать с датами и временем.

Этот объект можно создать разными путями в зависимости от требуемых результатов. Например, можно создать объект без аргументов: `var myDate = new Date();`

- ▶ Можно также в качестве аргумента передать milliseconds:

```
var myDate = new Date(milliseconds);
```

- ▶ Можно передать в качестве аргумента строку с датой:

```
var myDate = new Date(dateString);
```

- ▶ Можно передать несколько аргументов, представляющих составные части даты:

```
var myDate = new Date(year, month, day, hours, minutes, seconds, milliseconds);
```

Кроме того, для объекта `Data` доступны несколько методов, которые можно использовать сразу после создания объекта. Большинство доступных методов предназначено для получения определенных частей текущего времени.

Для объекта `Data` доступны следующие `getter`-методы:

- ▶ `getDate`
- ▶ `getDay`
- ▶ `getFullYear`
- ▶ `getHours`
- ▶ `getMilliseconds`
- ▶ `getMinutes`
- ▶ `getMonth`
- ▶ `getSeconds`
- ▶ `getTime`
- ▶ `getTimezoneOffset`

Как можно заметить, каждый метод довольно прост в плане возвращаемых им значений. Разница состоит в диапазоне возвращаемых значений.

Например:

- ▶ метод `getDate` возвращает номер дня месяца (от 1 до 31).
- ▶ метод `getDay` возвращает номер дня недели (от 0 до 6).
- ▶ метод `getHours` возвращает числовое значение часа (от 0 до 23).
- ▶ функция `getMilliseconds` возвращает числовое значение миллисекунд (от 0 до 999).
- ▶ Методы `getMinutes` и `getSeconds` возвращают значения в диапазоне от 0 до 59.
- ▶ метод `getMonth` возвращает номер месяца (от 0 до 11). Особняком в этом списке стоят только методы `getTime` и `getTimezoneOffset`.
- ▶ Метод `getTime` возвращает число миллисекунд, прошедших с полуночи 1 января 1970 года.
- ▶ метод `getTimezoneOffset` возвращает разницу в минутах между временем по Гринвичу и локальным временем.

Для большинства getter-методов существуют setter-методы, принимающие числовой аргумент соответствующего диапазона значений. Вот эти методы:

- ▶ setDate
- ▶ setFullYear
- ▶ setHours
- ▶ setMilliseconds
- ▶ setMinutes
- ▶ setMonth
- ▶ setSeconds
- ▶ setTime

Для всех перечисленных выше getter-методов существуют аналогичные методы, возвращающие те же диапазоны значений, но только по Гринвичу. Вот эти методы:

- ▶ `getUTCDate`
- ▶ `getUTCDay`
- ▶ `getUTCFullYear`
- ▶ `getUTCHours`
- ▶ `getUTCMilliseconds`
- ▶ `getUTCMinutes`
- ▶ `getUTCMonth`
- ▶ `getUTCSeconds`

Поскольку для всех оригинальных getter-методов существуют setter-методы, есть их аналоги, использующие время по Гринвичу.

Вот эти методы:

- ▶ `setUTCDate`
- ▶ `setUTCFullYear`
- ▶ `setUTCHours`
- ▶ `setUTCMilliseconds`
- ▶ `setUTCMinutes`
- ▶ `setUTCMonth`
- ▶ `setUTCSeconds`

Для объекта Date доступно несколько методов, немного по-разному преобразующих даты в строки, в том числе:

- ▶ `toString`
- ▶ `toLocaleDateString`
- ▶ `toLocaleTimeString`
- ▶ `toLocaleString`
- ▶ `getTimeString`
- ▶ `toUTCString`
- ▶ Метод `toString` преобразует дату в строку:
 - ▶ `var myDate = new Date();`
 - ▶ `document.write(myDate.toString());`
 - ▶ `toString` возвращает текущую дату в формате *Tue Jul 19 2011*.
- ▶ Метод `getTimeString` преобразует время из объекта Date в строку:
 - ▶ `var myDate = new Date();`
 - ▶ `document.write(myDate.getTimeString());`
 - ▶ `getTimeString` возвращает время в виде строки в формате *23:00:00 GMT-0700 (MST)*.

Последним методом, преобразующим дату в строку, является `toUTCString`, который работает с временем по Гринвичу.

JavaScript-объект String является оболочкой для текстовых значений.

- ▶ Объект String имеет всего одно свойство, length (длина), доступное только для чтения. Свойство length используется для возврата длины строки и не может быть изменено извне. Ниже приведен пример использования свойства length для определения количества символов в строке:

```
var myString = "My string";
```

```
document.write(myString.length);
```

```
// Результатом является числовое значение 9
```

- ▶ В результате этот код выдает значение 9, поскольку пробелы между двумя словами также учитываются при подсчете.

В объекте `String` есть несколько методов, которые используются для манипулирования текстовыми данными и сбора информации о них. Ниже приведен перечень доступных методов:

- `charAt`
- `charCodeAt`
- `concat`
- `fromCharCode`
- `indexOf`
- `lastIndexOf`
- `match`
- `replace`
- `search`
- `slice`
- `split`
- `substr`
- `substring`
- `toLowerCase`
- `toUpperCase`

- ▶ Метод `charAt` можно использовать для извлечения конкретного символа по индексу, переданному в качестве аргумента. Приведенный ниже пример кода возвращает первый символ строки:

```
var myString = "My string";  
document.write(myString.charAt(0));  
// Результаты в М
```

- Если необходим противоположный результат, существует пара методов, возвращающих индекс указанного символа или набора символов в строке. Это методы `indexOf` и `lastIndexOf`, которые принимают два параметра: `searchString` и `start`. Параметр `searchString` является начальным индексом, а параметр `start` указывает, где нужно начинать поиск. Разница между этими методами состоит в том, что `indexOf` возвращает первое появление в строке, а `lastIndexOf` - последнее.
- Метод `charCodeAt` похож на `charAt`. Единственное отличие состоит в том, что он возвращает символ в кодировке Unicode. Еще одним методом объекта `String`, работающим с Unicode, является `fromCharCode`, преобразующий Unicode в символы.
- Если нужно объединить строки, можно либо соединить их при помощи знака плюс (+), либо соответствующим образом использовать метод `concat`. Этот метод принимает неограниченное число строковых аргументов, соединяет их и возвращает результат в виде новой строки. В листинге 2 приведен пример объединения несколько строк в одну при помощи экземпляра `concat`.
- Листинг 2. Объединение нескольких строк при помощи метода `concat`

```
var myString1 = "My";  
var myString2 = " ";  
var myString3 = "string";  
document.write(myString.concat(myString1, myString2, myString3);  
// Результаты в "My String"
```

- ▶ Существует также группа методов String, принимающих в качестве аргументов регулярные выражения для поиска или изменения строк. К ним относятся методы match, replace и search. Метод match использует регулярное выражение для поиска определенной строки и возвращает все совпадения. Метод replace принимает подстроку либо регулярное выражение и замещающую строку (в качестве второго параметра) и возвращает обновленную строку. Последний из этих методов, search, ищет совпадение с регулярным выражением и возвращает его позицию.
- ▶ Для изменения строк существует несколько методов. Первым является метод slice, который извлекает и возвращает часть строки по заданному индексу или по комбинации начального и конечного индексов. Другим таким методом является метод split. Он разбивает строку на массив подстрок там, где обнаруживает символ-разделитель. Если, например, в качестве аргумента передан знак "запятая" (,), строка будет разделена на новые подстроки по каждой запятой. Менять строки могут также метод substr, принимающий в качестве аргументов начальную позицию и количество символов и извлекающий из строки подстроку соответствующей длины начиная с указанной позиции, и метод substring, извлекающий подстроку по начальному и конечному индексам, переданным в качестве аргументов. И, наконец, имеются методы toLowerCase и toUpperCase, преобразующие символы строки в нижний и верхний регистр соответственно. Эти методы полезны при сравнении значений строк, поскольку строки иногда могут записываться в разных регистрах. Эти методы позволяют сравнить значения без учета регистра символов.

- Выполнили: Ткенова
Демин
Гребцов