Школа

JavaScript

Добро пожаловать в школу JavaScript

Школа JavaScript[1] (open link)

JavaScript - это язык программирования в для веба. На школе JavaScript вы узнаете как писать скрипты на JavaScript и встраивать их в свои HTML-документы, как сделать страницы более динамичными и интерактивными. Изучайте JavaScript!

Примеры JavaScript[2] (open link)

Более ста примеров! С помощью нашего редактора вы можете вносить изменения в код и, нажимая кнопку, сразу видеть результат, попробуйте!

Тест по JavaScript[3] (open link)

Проверьте свое знание JavaScript, пройдите наш тест!

Справочник по JavaScript[4] (open link)

Здесь вы найдете полный справочник по JavaScript - описание объектов, их свойств и методов.

Книги по JavaScript[5] (open link)

Ищете хорошую книгу по JavaScript? Посмотрите наше обозрение!

Школа JavaScript: содержание

1: http://xml.nsu.ru/js/js_intro.xml
2: http://xml.nsu.ru/js/js_examples.xml
3: http://www.w3schools.com/js/js_quiz.asp
4: http://xml.nsu.ru/js/js_reference.xml
5: http://www.w3schools.com/js/js_books.asp
6: http://xml.nsu.ru/js/js_intro.xml
7: http://xml.nsu.ru/js/js_howto.xml
8: http://xml.nsu.ru/js/js_whereto.xml

Переменные[9] (open link)

В этой главе рассказывается о том, как создавать и применять переменные в JavaScript.

Операторы[10] (open link)

В этой главе рассказывается о том, как применять операторы в JavaScript

Функции[11] (open link)

В этой главе рассказывается о том, как создавать и вызывать функции в JavaScript.

Условные выражения[12] (open link)

В этой главе рассказывается о том, как применять условия вида if..else (если... иначе...) и выражения-переключатели.

Выражения организации цикла[13] (open link)

В этой главе рассказывается о том, как в JavaScript создавать циклы.

Рекомендации[14] (open link)

Прежде, чем начинать писать скрипты, познакомьтесь с этими рекомендациями.

Объект типа String (строка)[15] (open link)

В этой главе рассказывается об объекте String и приведен действующий пример.

Объект Array (массив)[16] (open link)

В этой главе рассказывается, как связать с переменной более одного значения.

Объект типа Date (дата)[17] (open link)

В этой главе рассказывается об объекте Date и приведен действующий пример.

Объект типа Math (математический объект)[18] (open link)

В этой главе рассказывается об объекте Math и приведен действующий пример.

Примеры работы с окнами[19] (open link)

В этой главе приводится примеры работы с окнами. В каждом примере вы можете изучить исходный код и разобраться, как он работает.

Примеры работы с фреймами[20] (open link)

В этой главе приводится примеры работы с фреймами. В каждом примере вы можете изучить исходный код и разобраться, как он работает.

Примеры работы с формами[21] (open link)

В этой главе приводится примеры работы с формами. В каждом примере вы можете изучить исходный код и разобраться, как он работает.

Примеры определения типа браузера[22] (open link)

В этой главе приводится примеры определения типа браузера. В каждом примере вы можете изучить исходный код и разобраться, как он работает.

9: http://xml.nsu.ru/js/js_variables.xml
10: http://xml.nsu.ru/js/js_operators.xml
11: http://xml.nsu.ru/js/js_functions.xml
12: http://xml.nsu.ru/js/js_conditionals.xml
13: http://xml.nsu.ru/js/js_looping.xml
14: http://xml.nsu.ru/js/js_guidelines.xml
15: http://xml.nsu.ru/js/js_string.xml
16: http://xml.nsu.ru/js/js_arrays.xml
17: http://xml.nsu.ru/js/js_datetime.xml
18: http://xml.nsu.ru/js/js_math.xml
19: http://xml.nsu.ru/js/js_window.xml
20: http://xml.nsu.ru/js/js_frames.xml
21: http://xml.nsu.ru/js/js_form.xml
22: http://xml.nsu.ru/js/js_browser.xml

Примеры JavaScript и тест

Примеры по JavaScript[23] (open link)

Множество примеров по JavaScript!

Тест по JavaScript[24] (open link)

Проверьте свое знание JavaScript!

Описание элементов JavaScript

Элементы JavaScript[25] (open link)

Наш справочник по JavaScript содержит описание всех встроенных объектов JavaScript, описание их методов, а также информацию о поддержке объектов и методов различными браузерами.

Ресурсы по JavaScript

Книги по JavaScript[26] (open link)

Ищете хорошую книгу по JavaScript? Посмотрите наше обозрение!

Введение в JavaScript

JavaScript - скриптовый язык, разработанный компанией Netscape.
JavaScript работает на всех основных браузерах третей версии или выше.

Что вы должны уже знать

Прежде, чем вы продолжите, вы должны иметь общее представление о WWW, HTML и об основах создания Web-страниц.

Что такое JavaScript?

- JavaScript - это скриптовый язык
- Скриптовый язык - это облегченный вариант языка программирования
- JavaScript - это строки исполняемого компьютерного кода
- JavaScript можно вставлять в HTML-страницы
- JavaScript - открытый для использования скриптовый язык, им может пользоваться каждый без получения лицензии или покупки
- JavaScript поддерживается всеми основными браузерами, такими, как Netscape и Internet Explorer

23: http://xml.nsu.ru/js/js_examples.xml
24: http://www.w3schools.com/js/js_quiz.asp
25: http://xml.nsu.ru/js/js_reference.xml
26: http://www.w3schools.com/js/js_books.asp

Как он работает?

Когда JavaScript вставлен в HTML-документ, браузер читает HTML и интерпретирует JavaScript. JavaScript может исполняться либо немедленно либо по наступлении определенного события.

Что может делать JavaScript?

JavaScript предоставляет HTML-дизайнерам инструмент программирования

Авторы HTML-документов обычно не являются программистами, но поскольку JavaScript - очень легкий язык программирования с очень простым синтаксисом, почти каждый может научиться встраивать небольшие кусочки кода в свои HTML-документы.

JavaScript может помещать на HTML-страницу динамически изменяющийся текст

Выражение на языке JavaScript: document.write("<h1>" + name + "</h1>") может выводить при показе HTML-страницы переменный текст точно также, как это статично делается в HTML: <h1>Bill Gates</h1>.

JavaScript может реагировать на события

Можно сделать так, что JavaScript будет исполняться после наступления определенного события, например, после того, как страница заканчивает загружаться или после того, как пользователь кликнул какой-либо HTML-элемент.

JavaScript может считывать и вписывать HTML-элементы

JavaScript может считывать HTML-элементы и изменять содержимое HTML-элементов.

JavaScript можно применять для оценки данных

JavaScript можно применять для оценки введенных в форму данных еще до того как они отправляются на сервер. Эта функция особенно хорошо приспособлена для того, чтобы экономить вычислительные ресурсы сервера.

JavaScript: как?

Для вставки JavaScript в HTML- документ применяйте тэг <script>

Примеры

Вписывание текста:
Посмотрите сами, как действует вписывание в страницу текста с помощью JavaScript:[27] (open editor)

Вписывание форматированного текста:
А так действует вписывание в страницу текста, форматированного с помощью HTML:[28] (open editor)

Как поместить JavaScript в HTML-документ

27: http://www.w3schools.com/js/tryit.asp?filename=tryjs_text
28: http://www.w3schools.com/js/tryit.asp?filename=tryjs_formattext

```
<html>
<head>
</head>
<body>
<script type="text/javascript">
document.write("Hello World!")
</script>
</body>
</html>
```

Эта процедура приведет к следующему результату:

```
Hello World!
```

Для вставки скритпа в HTML-документ применяется тэг <script>. Для определения скриптового языка используйтся атрибут type.

```
<script type="text/javascript">
```

Далее идет собственно JavaScript: в нем команда для вписывания в страницу какого-либо текста - это document.write:

```
document.write("Hello World!")
```

Скрипт заканчивается так:

```
</script>
```

Нужно ли заканчивать выражения точкой с запятой?

В традиционных языках программирования, таких как C++ и Java каждое выражение должно заканчиваться точкой с запятой.

Многие программисты по привычке делают также и программируя на JavaScript, но вообще-то в нем применение точки с запятой не обязательно и требуется только тогда, когда вы помещаете на строку несколько выражений.

Что делать со старыми браузерами?

Браузеры, которые не поддерживают скрипты будут показывать их как содержание страницы. Чтобы они этого не делали, используйте HTML-тэг комментария:

```
<script type="text/javascript">
<!--
    Выражения JavaScript
//-->
```

```
</script>
```

Два прямых слэша, которыми начинается завершающая страка комментария (//) - это символ комментария в JavaScript, они не дают JavaScript пытаться компилировать эту строку.

Обратите внимание, что нельзя помещать // перед первой строкой комментария (например, так: //<!--), потому что более старые браузеры ее покажут. Смешно? Да! Но именно так обстоят дела.

JavaScript: где?

Скрипты на странице исполняются немедленно после окончания загрузки страницы. Это не всегда нужно. Иногда нам нужно, чтобы скрипт исполнялся после загрузки страницы, а иногда нужно, чтобы он запускался каким-то событием

Примеры

Скрипт внутри элемента HEAD:
Скрипты, которые содержат функции располагаются внутри элемента страницы HEAD. При этом скрипт наверняка оказывается полностью загруженным, когда вызывается содержащаяся в нем функция:[29] (open editor)

Скрипт внутри элемента BODY:
Выполнение скрипта, размещенного в элементе BODY:[30] (open editor)

Внешний скрипт:
Получение доступа к внешнему скритпу:[31] (open editor)

Как размещать JavaScript

Скрипты на странице исполняются немедленно после окончания загрузки страницы. Это не всегда нужно. Иногда нам нужно, чтобы скрипт исполнялся после загрузки страницы, а иногда нужно, чтобы он запускался каким-то событием.

Скрипт внутри элемента HEAD: Скрипты, которые должны исполняться после их вызова или по наступлению какого-либо события размещаются внутри элемента HEAD. Размещая там скрипт вы может быть уверены: когда скрипт будет вызван, он уже будет полностью загружен.

```
<html>
<head>
<script type="text/javascript">
     Выражения JavaScript
</script>
</head>
```

Скрипт внутри элемента BODY: Скрипты, которые могут исполняться еще во время продолжающейся загрузки страницы размещаются внутри элемента BODY. Размещенный там скрипт

29: http://www.w3schools.com/js/tryit.asp?filename=tryjs_headsection
30: http://www.w3schools.com/js/tryit.asp?filename=tryjs_bodysection
31: http://www.w3schools.com/js/tryit.asp?filename=tryjs_externalexample

участвует в генерации содержания страницы.

```
<html>
<head>
</head>
<body>
<script type="text/javascript">
     Выражения JavaScript
</script>
</body>
```

Скрипты и внутри элемента HEAD и внутри элемента BODY: Вы можете размещать на странице неограниченное количество скриптов, так что скрипты могут быть одновременно и внутри элемента HEAD и внутри элемента BODY.

```
<html>
<head>
<script type="text/javascript">
     Выражения JavaScript
</script>
</head>
<body>
<script type="text/javascript">
     Выражения JavaScript
</script>
</body>
```

Как запустить внешний скрипт

Иногда нужно применять один и тот же скрипт на нескольких страницах, для этого приходится пмсать скрипт на каждой странице.

Чтобы упростить дело, вы можете содержать скрипт во внешнем файле с расширением .js. Допустим, мы имеем следующий скрипт:

```
document.write("This script is external")
```

Сохраните этот скрипт как файл xxx.js. При этом, во-первых, название файла должно быть не длиннее 8 знаков, а во-вторых, в тексте внешнего скрипта не должно быть тэга <script>.

Теперь вы можете вызвать этот скрипт из любой части страницы, используя атрибут "src":

```
<html>
<head>
</head>
<body>
<script src="xxx.js">
</script>
</body>
</html>
```

Обратите внимание: ссылка на скрипт помещается именно там, где обычно располагается сам скрипт.

Переменные в JavaScript

Переменные - это своего рода контейнеры, в которых можно хранить информацию

Примеры

Работа с переменной:
Переменные используются для хранения данных. В этом примере показано как именно это делается:[32] (open editor)

Переменные

Переменные - это своего рода контейнеры, в которых можно хранить информацию. Значение переменной может изменяться. Для того, чтобы получить или изменить значение переменной, вы можете вызывать ее по имени.

Правила именования переменных:

- Имена переменных чувствительны к регистру (заглавная - маленькая буква)
- Имена переменных должны начинаться с буквы или с символа подчеркивания

Объявление переменной

Переменная создается с помощью выражения var:

```
var strname = some value
```

Можно создать переменную и без использования выражения var:

```
strname = some value
```

Установление значения переменной

Значение переменной устанавливается следующим образом:

```
var strname = "Hege"
```

Или вот так:

```
strname = "Hege"
```

Имя переменной записывается слева, ее значение - справа. Теперь значение переменной "strname" равно "Hege".

Время жизни переменной

Если вы объявили переменную внутри функции, ей можно пользоваться только внутри этой функции. После выхода из функции переменная разрушается. Такие переменные называ-

32: http://www.w3schools.com/js/tryit.asp?filename=tryjs_variable

ются локальными. Вы можете создавать различные локальные переменные в разных функциях с одним и тем же именем, поскольку каждая будет действовать только внутри своей функции.

Если вы объявили переменную вне функции, все функции на странице смогут ее использовать. Время жизни такой переменной начинается с момента ее объявления и заканчивается после закрытия страницы.

Операторы в JavaScript

Операторы применяются для работы с переменными

Арифметические операторы

Оператор	Описание	Пример	Результат
+	Сумма	2+2	4
-	Разность	5-2	3
*	Произведение	4*5	20
/	Частное	5/2	2.5
%	Остаток. Возвращает остаток от деления.	5%2	1
++	Инкремент	x=5; x++	6
--	Декремент	x=5; x--	4

Операторы сравнения

Оператор	Описание	Пример	Результат
==	Равно	5==8	false
!=	Не равно	5!=8	true
>	Больше, чем	5>8	false
<	Меньше, чем	5<8	true
>=	Больше или равно	5>=8	false
<=	Меньше или равно	5<=8	true

Сокращенная запись операторов

Оператор	Пример	То же самое, что и..
+=	x+=y	x=x+y

-=	x-=y	x=x-y
***=**	x*=y	x=x*y
/=	x/=y	x=x/y
%=	x%=y	x=x%y

Логические операторы

Оператор	Описание	Пример	Результат
&&	and	x=6; y=3; (x < 10 && y > 1)	true
||	or	x=6; y=3; (x==5 || y==5)	false
!	not	x=6; y=3; x != y	true

Строковые операторы

Строки чаще всего являются текстом, например, "Hello World!". Чтобы склеить две или более строчных переменных вместе, используйте оператор +:

```
txt1="What a very"
txt2="nice day!"
txt3=txt1+txt2
```

Теперь переменная txt3 имеет значение "What a verynice day!"

Чтобы добавить пробел между двумя строковыми переменными, вставьте пробел в выражение или в одну из строковых переменных:

```
txt1="What a very"
txt2="nice day!"
txt3=txt1+" "+txt2
или
txt1="What a very "
txt2="nice day!"
txt3=txt1+txt2
```

Теперь переменная txt3 имеет значение "What a very nice day!"

Функции в JavaScript

Функция - это многократно исполняемый кусок кода, он выполняется по наступлению какого-либо события или при его вызове

Примеры

Функция:
Вызов функции:[33] (open editor)

33: http://www.w3schools.com/js/tryit.asp?filename=tryjs_function1

Функция с аргументами:
Как передать функции переменную и как затем использовать значение этой переменной в функции:[34] (open editor)

Функция с аргументами (2):
Как передать функции переменные и как затем использовать значения этих переменных в функции:[35] (open editor)

Функция, возвращающая значение:
Как заставить функцию возвратить значение:[36] (open editor)

Функция с аргументами, возвращающая значение:
Как заставить функцию найти сумму двух аргументов и возвратить получившееся значение:[37] (open editor)

Функции

Функция содержит некоторый код, который выполняется по наступлению какого-либо события или при вызове этой функции. Функция состоит из набора выражений. Вы можете многократно применять функции в одном и том же скрипте или в других документах. Вы определяете функции в начале файла (в разделе HEAD) и вызываете их из далее следующих частей кода. Теперь самое время познакомиться с тем, как вызвать выскакивающее сообщение (alert-box).

Этот метод JavaScript применяется для выдачи сообщений пользователю.

```
alert("здесь находится текст сообщения")
```

Как определить функцию

Для того, чтобы создать функцию, вы определяете ее имя, некоторые значения ("аргументы") и набор выражений:

```
function myfunction(argument1,argument2,etc)
{
набор выражений
}
```

Даже если у функции нет аргументов, она должна быть написана с парой круглых скобок:

```
function myfunction()
{
набор выражений
}
```

Аргументы - это переменные, которые будут использоваться в функции. Значения этих переменных при вызове функции передаются ей.

Размещая функции в разделе HEAD вашего документа, вы позволяете коду функций загрузиться раньше, чем они будут вызываться.

Некоторые функции возвращают определенные значения вызывающим функцию выражениям:

34: http://www.w3schools.com/js/tryit.asp?filename=tryjs_function2
35: http://www.w3schools.com/js/tryit.asp?filename=tryjs_functionarg2
36: http://www.w3schools.com/js/tryit.asp?filename=tryjs_function_return2
37: http://www.w3schools.com/js/tryit.asp?filename=tryjs_function_return

```
function result(a,b)
{
c=a+b
return c
}
```

Как вызвать функцию

Функция не выполняется до тех пор, пока не будет вызвана.

Вы можете вызвать функцию, содержащую аргументы:

```
myfunction(argument1,argument2,etc)
```

или без аргументов:

```
myfunction()
```

Выражение return

В функции, которая возвращает некоторое значение должно применяться выражение "return". Это выражение определяет значение, которое будет передано туда, откуда была вызвана данная функция. Например, если ваша функция возвращает сумму двух чисел:

```
function total(a,b)
{
result=a+b
return result
}
```

когда вы вызываете эту функцию, вы должны передать ей два аргумента:

```
sum=total(2,3)
```

Возвращаемое функцией значение (5) будет сохранено в переменной sum.

Условные выражения

Условные выражения в JavaScript применяются для организации выполнения нескольких различных действий в зависимости от выполнения различных условий

Примеры

Выражение if (если):
Как записать выражение if. Применяйте это выражение, когда вы хотите, чтобы определенный кусок кода исполнялся только при выполнении какого-либо условия:[38] (open editor)

Выражение if...else (если...иначе):
Как записать выражение if...else. Применяйте это выражение, когда вы хотите, чтобы один кусок кода исполнялся при выполнении какого-либо условия, а другой - при его невыполнении:[39] (open editor)

Случайная ссылка:

38: http://www.w3schools.com/js/tryit.asp?filename=tryjs_ifthen
39: http://www.w3schools.com/js/tryit.asp?filename=tryjs_ifthenelse

В этом примере создается ссылка, которая с равной вероятностью может вас привести на W3Schools.com или на W3AppML.com:[40] (open editor)

Выражение-переключатель switch:
Как написать выражение-переключатель. Используйте это выражение, когда вам нужно выбрать для исполнения один из многих кусков кода:[41] (open editor)

Условные выражения

Очень часто при написании кода приходится учитывать ситуации, когда в зависимости от определенных условий нужно выполнять различные куски кода. Для этого следует применять условные выражения.

В JavaScript есть три вида условных выражений:

- if (если) - применяйте это выражние, когда определенный кусок кода должен исполняться, если какое-то условие выполняется
- if...else (если...иначе) - применяйте это выражение, когда нужно выбрать один из двух кусков кода
- switch (переключатель) - применяйте это выражение, когда нужно выбрать один из многих кусков кода

Выражения if и if...else

Выражение if используется, когда кусок кода должен выполняться при выполнении определенного условия.

Синтаксис

```
if (условие)
{
кусок кода, который исполняется,
если выполняется условие
}
```

Пример

```
<script type="text/javascript">
//Если в браузере установлено время меньше 10 часов,
//вы получите приветствие "Good morning".

var d=new Date()
var time=d.getHours()

if (time<10)
{
document.write("<b>Good morning</b>")
}
</script>
```

Заметьте, что в коде не используется конструкция ..else.. В коде просто дается указание выполнять определенный кусок кода при выполнении заданного условия.

Если вы хотите, чтобы при выполнении заданного условия выполнялся один кусок кода, а при его не-выполнении - другой, используйте выражение if....else.

40: http://www.w3schools.com/js/tryit.asp?filename=tryjs_randomlink
41: http://www.w3schools.com/js/tryit.asp?filename=tryjs_switch

Синтаксис

```
if (условие)
{
кусок кода, который исполняется,
если выполняется условие
}
else
{
кусок кода, который исполняется,
если условие не выполняется
}
```

Пример

```
<script type="text/javascript">
//Если в браузере установлено время меньше 10 часов,
//вы получите приветствие "Good morning".
//Иначе вы получите приветствие "Good day".

var d = new Date()
var time = d.getHours()

if (time < 10)
{
document.write("Good morning!")
}
else
{
document.write("Good day!")
}
</script>
```

Выражение switch

Выражение switch используется в тех случаях, когда нужно выбрать один из нескольких кусков кода на основе выполнения условия.

Синтаксис

```
switch (выражение)
{
case label1:
  кусок кода, который выполняется,
  если выражение = label1
  break
case label2:
  кусок кода, который выполняется,
  если выражение = label2
  break
default:
  кусок кода, который выполняется, если значение
  выражения отличается и от label1 и от label2
}
```

Это работает так: Мы имеем определенное выражение (чаще всего это - переменная), его значение определяется в самом начале. Затем значение этого выражения сравнивается со значениями, указанными после слов case. Если имеется совпадение с одним из указанных случаев, выполняется кусок кода, связанный с этим случаем. Использование слова break не позволяет автоматически выполняться куску кода, соответствующему следующему случаю.

Пример

```
<script type="text/javascript">
//Мы будем получать различные приветствия в
//зависимости от дня недели. При этом, Sunday=0,
//Monday=1, Tuesday=2, и т.д.

var d=new Date()
theDay=d.getDay()
switch (theDay)
{
case 5:
  document.write("Finally Friday")
  break
case 6:
  document.write("Super Saturday")
  break
case 0:
  document.write("Sleepy Sunday")
  break
default:
  document.write("I'm looking forward to this weekend!")
}
</script>
```

Условный оператор

Кроме того, в JavaScript имеется условный оператор, который устанавливает значение переменной на основе выполнения некоторого условия.

Синтаксис

```
variablename=(условие)?value1:value2
```

Пример

```
greeting=(visitor=="PRES")?"Dear President ":"Dear "
```

Если значение переменной visitor равно "PRES", значение переменной greeting устанавливается равным "Dear President ". Если значение переменной visitor не равно "PRES", тогда значение переменной greeting приравнивается строке "Dear ".

Выражения организации цикла

Выражения организации цикла используются для выполнения определенного куска кода заданное число раз

Примеры

Выражение for:
Организация цикла на основе выражения for. Используйте это выражение для выполнения заданного куска кода определенное число раз:[42] (open editor)

Циклическое выведение HTML-заголовков:
Применение цикла на основе выражения for для выведения HTML-заголовков:[43] (open editor)

Выражение while:
Организация цикла на основе выражения while. Применяйте это выражение для исполнения куска кода до тех пор пока не будет выполняться некоторое условие:[44] (open editor)

Выражение do while:
Организация цикла на основе выражения do while. Применяйте это выражение для организации циклического выполнения куска кода до тех пор, пока соблюдается определенное условие. Заданный в выражении кусок кода будет выполняться хотя бы один раз, даже если условие не соблюдается, поскольку он выполняется до того, как проверяется условие:[45] (open editor)

Организация цикла

Очень часто при написании кода приходится организовывать выполнение определенного куска кода заданное число раз. Для этого применяются выражения организации цикла.

В JavaScript имеются следующие выражения организации цикла:

- while - циклическое выполнение куска кода до тех пор, пока выполняется условие
- do...while - разовое выполнение куска кода и циклическое его выполнение до тех пор, пока соблюдается условие
- for - выполнение куска кода заданное число раз

Выражение while

Выражение while выполняет кусок кода, пока выполняется условие.

```
while (условие)
{
    исполняемый кусок кода
}
```

Выражение do...while

Выражение do...while один раз выполняет кусок кода и продолжает его циклически выполнять, пока соблюдается условие.

```
do
{
    исполняемый кусок кода
}
while (условие)
```

42: http://www.w3schools.com/js/tryit.asp?filename=tryjs_fornext
43: http://www.w3schools.com/js/tryit.asp?filename=tryjs_fornext_header
44: http://www.w3schools.com/js/tryit.asp?filename=tryjs_while
45: http://www.w3schools.com/js/tryit.asp?filename=tryjs_dowhile

Выражение for

Выражение for выполняет заданный кусок кода определенное число раз.

```
for (инициализация; условие; приращение)
{
    исполняемый кусок кода
}
```

Рекомендации

Несколько вещей, которые нужно знать о JavaScript

JavaScript чувствителен к регистру

Функция, имеющая имя "myfunction" не тождественна функции, имеющей имя "myFunction". Поэтому следите за регистром букв, когда даете названия функциям, объектам или переменным.

Символы

Открывающие парные символы, такие, как ({ [" ', должны сопровождаться закрывающими символами ' "] }).

Пробелы

JavaScript игнорирует лишние пробелы. Вы можете добавлять в строчки кода пробелы, чтобы сделать их более читабельными. Эти две строчки имеют совершенно одинаковое значение:

```
name="Hege"
name = "Hege"
```

Разбиение строчек кода

Вы можете разбивать строчки кода внутри текста обратными слэшами. В этом примере текст будет отображаться правильно:

```
document.write("Hello \
World!")
```

Обратите внимание: нельзя разбивать строчки кода вот так:

```
document.write \
("Hello World!")
```

Этот пример вызовет ошибку.

Вставка специальных символов

Вы можете вставлять в код специальные символы (например, " ' ; &) используя обратный

слэш:

```
document.write ("You \& I sing \"Happy Birthday\".")
```

Этот пример выдаст следующий результат:

```
You & I sing "Happy Birthday".
```

Комментарии

Вы можете добавлять комментарии в свой код JavaScript, размещая их после "//":

```
sum=a + b  //вычисление суммы
```

Можно также располагать комментарии в коде, начиная их с "/*" и заканчивая "*/":

```
sum=a + b  /*вычисление суммы*/
```

Использование "/*" и "*/" - единственный способ сделать многострочный комментарий:

```
/* Это блок комментариев.
Он состоит из нескольких строк*/
```

Объект типа String (строка)

Объект String применяется для работы с текстом

Примеры

Метод length:
Метод length возвращает число символов в строке:[46] (open editor)

Метод indexOf():
Проверка - содержит ли данная строка определенный символ или под-строку. Возвращает в виде целого числа положение символа (строки), если он найден, или число -1, если не найден. Обратите внимание, что положение первого символа в строке равно 0:[47] (open editor)

Метод match():
Работает подобно методу indexOf, но этот метод возвращает заданные вами символы или "null", если строка не содержит заданные вами символы:[48] (open editor)

Метод substr():
Возвращает заданную часть строки. Если вы укажете в качестве парметра метода (3,6), вы получите строку, начинающуюся с третьего символа и длиной в 6 символов. (Обратите внимание, что поскольку первым символом в строках является нулевой, строка-результат будет фактически начинаться с четвертого символа):[49] (open editor)

Метод substring():
Возвращает заданную часть строки. Парметры метода (3,6) выдадут символы начиная с третьего до символа с номером 6-1. (Обратите внимание, что поскольку первым символом в строках является нулевой, строка-результат будет фактически начинаться с четвертого символа и заканчиваться шестым):[50] (open editor)

46: http://www.w3schools.com/js/tryit.asp?filename=tryjs_length
47: http://www.w3schools.com/js/tryit.asp?filename=tryjs_indexof
48: http://www.w3schools.com/js/tryit.asp?filename=tryjs_match
49: http://www.w3schools.com/js/tryit.asp?filename=tryjs_substr
50: http://www.w3schools.com/js/tryit.asp?filename=tryjs_substring

Методы toLowerCase() и toUpperCase():
Преобразуют строку в нижний и верхний регистр соответственно:[51] (open editor)

Наиболее распространенные методы

NN - для браузера Netscape, IE - для браузера Internet Explorer:

Метод	Описание	NN	IE
`length`	Возвращает число символов в строке	2.0	3.0
`indexOf()`	Возвращает положение первого вхождения заданного символа или под-строки в данной строке или число -1 если вхождения не обнаружены	2.0	3.0
`lastIndexOf()`	Аналогично indexOf, но начинает работать справа и двигается налево	2.0	4.0
`match()`	Аналогично indexOf и lastIndexOf, но этот метод возвращает заданные символы или "null", а не числовое значение	4.0	4.0
`substr()`	Возвращает заданные символы: (14,7) возвращает 7 символов начиная с 14-го	4.0	4.0
`substring()`	Возвращает заданные символы: (7,14) возвращает все символы между 7-ым и 14-1ым	2.0	3.0
`toLowerCase()`	Преобразует строку в нижний регистр	2.0	3.0
`toUpperCase()`	Преобразует строку в верхний регистр	2.0	3.0

Объект Array (массив)

Объект Array используется для хранения набора значений под одним именем переменной

Примеры

Массив:
Массивы используются для множества однородных данных. В этом примере показано как можно использовать массив для хранения множества имен:[52] (open editor)

Массив (2):

51: http://www.w3schools.com/js/tryit.asp?filename=tryjs_lowerupper
52: http://www.w3schools.com/js/tryit.asp?filename=tryjs_array

Другой способ создания массива приводит к тем же результатам. Обратите внимание как метод length используется для определения количества элементов в массиве:[53] (open editor)

Объект Array

Объект Array используется для хранения набора значений под одним именем переменной. Каждое значение является элементом массива и идентифицируется по своему индексному номеру.

Вы можете вызывать определенный элемент массива используя имя массива и индексный номер элемента. Первый элемент имеет индексный номер 0.

Чтобы создать экземпляр объекта Array, используется ключевое слово "new":

```
var family_names=new Array(5)
```

В круглые скобки вписывается ожидаемое число элементов массива, в данном случае 5.

Данные вписываются в элементы массива следующим образом:

```
family_names[0]="Tove"
family_names[1]="Jani"
family_names[2]="Stele"
family_names[3]="Hege"
family_names[4]="Kai"
```

Данные извлекаются из массива указанием конкретного индексного номера элемента массива:

```
mother=family_names[0]
father=family_names[1]
```

Наиболее распространенные методы

NN - для браузера Netscape, IE - для браузера Internet Explorer:

Метод	Описание	NN	IE
length	Возвращает число элементов в массиве. Это свойство получает значение при создании массива	3.0	4.0
reverse()	Возвращает массив в обратном порядке	3.0	4.0
slice()	Возвращает заданную часть массива	4.0	4.0
sort()	Возвращает отсортированный массив	3.0	4.0

Объект типа Date (дата)

Объект Date используется для работы с датой и временем

53: http://www.w3schools.com/js/tryit.asp?filename=tryjs_array_length

Примеры

Дата:
Возвращает текущую дату включая день, месяц и год. Обратите внимание, что метод getMonth возвращает число 0 для января, 1 - для февраля и т.д. Так что для отображения правильной даты следует к результату, выдаваемому этим методом прибавлять единицу:[54] (open editor)

Время:
Возвращает текущее местное время включая час, минуты и секунды. Для получения универсального времени используйте методы getUTCHours, getUTCMinutes и т.д.:[55] (open editor)

Установка даты:
Кроме того, вы можете установить дату и время в объекте date с помощью методов setDate, setHour и т.д. Обратите внимание, что в этом примере устанавливается только FullYear:[56] (open editor)

Универсальное время:
Метод getUTCDate возвращает универсальное время (Universal Coordinated Time) - это время по установленному мировому стандарту:[57] (open editor)

Отображение недели:
Пример простого скрипта, который позволяет получать имя текущего дня недели вместо его номера. Обратите внимание, что для хранения имен дней недели используется массив, в котором нулевой элемент соответствует воскресенью (Sunday), первый - понедельнику (Monday) и т.д.:[58] (open editor)

Отображение полной даты:
Как получить полную дату с именем дня недели и именем месяца:[59] (open editor)

Отображение времени:
Как отобразить время на ваших страницах. Обратите внимание, что этот скрипт похож на предыдущий, но время выводится в текстовом поле. Кроме того, время обновляется раз в секунду:[60] (open editor)

Объект Date

Объект Date используется для работы с текущим временем и датой.

Вы можете создать экземпляр объекта Date с помощью ключевого слова "new".

Вот так можно сохранить текущую дату в переменной "my_date":

```
var my_date=new Date()
```

После создания экземпляра объекта Date, вы можете получить доступ ко всем методам этого объекта через переменную "my_date". Если, например, вы хотите получить у объекта Date день месяца (от 1 до 31), вам следует действовать так:

```
my_date.getDate()
```

54: http://www.w3schools.com/js/tryit.asp?filename=tryjs_datedate
55: http://www.w3schools.com/js/tryit.asp?filename=tryjs_datetime
56: http://www.w3schools.com/js/tryit.asp?filename=tryjs_datesetfullyear
57: http://www.w3schools.com/js/tryit.asp?filename=tryjs_dateutcdate
58: http://www.w3schools.com/js/tryit.asp?filename=tryjs_date_weekday
59: http://www.w3schools.com/js/tryit.asp?filename=tryjs_date_fulldate
60: http://www.w3schools.com/js/tryit.asp?filename=tryjs_time

Вы также можете вписывать данные внутрь круглых скобок объекта Date(), вот так:

```
new Date("Month dd, yyyy hh:mm:ss")
new Date("Month dd, yyyy")
new Date(yy,mm,dd,hh,mm,ss)
new Date(yy,mm,dd)
new Date(milliseconds)
```

Приводим примеры того, как можно создавать объекты Date каждым из выше перечисленных способов:

```
var my_date=new Date("October 12, 1988 13:14:00")
var my_date=new Date("October 12, 1988")
var my_date=new Date(88,09,12,13,14,00)
var my_date=new Date(88,09,12)
var my_date=new Date(500)
```

Наиболее распространенные методы

NN - для браузера Netscape, IE - для браузера Internet Explorer:

Метод	Описание	NN	IE
`Date()`	Возвращает объект Date	2.0	3.0
`getDate()`	Возвращает день месяца объекта Date (от 1 до 31)	2.0	3.0
`getDay()`	Возвращает день недели объекта Date (от 0 до 6. 0=Воскресенье, 1=Понедельник, и т.д.)	2.0	3.0
`getMonth()`	Возвращает месяц объекта Date (от 0 до 11. 0=Январь, 1=Февраль, и т.д.)	2.0	3.0
`getFullYear()`	Возвращает год объекта Date (четыре цифры)	4.0	4.0
`getHours()`	Возвращает час объекта Date (от 0 до 23)	2.0	3.0
`getMinutes()`	Возвращает минуты объекта Date (от 0 до 59)	2.0	3.0
`getSeconds()`	Возвращает секунды объ-екта Date (от 0 до 59)	2.0	3.0

Объект типа Math

Встроенный объект Math содержит в себе математические константы и функции

Примеры

Округление:
Округление заданного числа до ближайшего целого:[61] (open editor)
61: http://www.w3schools.com/js/tryit.asp?filename=tryjs_math_round

Случайное число:
Метод random возвращает случайное число из диапазона от 1 до 0:[62] (open editor)

Случайное число из диапазона от 0 до10:
Как получить случайное число из диапазона от 0 до 10 используя округление и метод random:[63] (open editor)

Максимальное число:
Как проверить, какое из двух чисел больше:[64] (open editor)

Минимальное число:
Как проверить, какое из двух чисел меньше:[65] (open editor)

Объект Math

Встроенный объект Math содержит в себе математические константы и функции. Вам не нужно сначала создавать этот объект перед использованием:

Сохранение случайного числа из диапазона от 0 до 1 в переменной "r_number":

```
r_number=Math.random()
```

Сохранение округленноно числа 8.6 в переменной "r_number":

```
r_number=Math.round(8.6)
```

Наиболее распространенные методы

NN - для браузера Netscape, IE - для браузера Internet Explorer:

Метод	Описание	NN	IE
`max(x,y)`	Возвращает максимальное число из пары x и y	2.0	3.0
`min(x,y)`	Возвращает минимальное число из пары x и y	2.0	3.0
`random()`	Возвращает случайное число из диапазона от 0 до 1	2.0	3.0
`round(x)`	Округляет число x до ближайшего целого числа	2.0	3.0

Примеры работы с окнами

Примеры

62: http://www.w3schools.com/js/tryit.asp?filename=tryjs_math_random
63: http://www.w3schools.com/js/tryit.asp?filename=tryjs_math_random09
64: http://www.w3schools.com/js/tryit.asp?filename=tryjs_math_max
65: http://www.w3schools.com/js/tryit.asp?filename=tryjs_math_min

Выскакивающее сообщение (alert box):
Как создать выскакивающее сообщение:[66] (open editor)

Выскакивающий запрос на подтверждение (confirm box):
Как создать запрос на подтверждение:[67] (open editor)

Запрос пользователю (prompt box):
Как создать запрос пользователю:[68] (open editor)

Новое окно:
Как открыть новое окно:[69] (open editor)

Новое окно (2):
Как открыть новое окно и указать его положение:[70] (open editor)

Множественные окна:
Как с одного клика открыть сразу много окон:[71] (open editor)

Location:
Как отправить клиента на другой адрес (URL/страницу):[72] (open editor)

Обновление:
Как обновить страницу:[73] (open editor)

Статусная строка:
Как вписывать текст в статусную строку окна:[74] (open editor)

Печать страницы:
Как отправить страницу на печать:[75] (open editor)

Примеры работы с фреймами

Примеры

Удаление фреймов:
Как выйти из фреймов:[76] (open editor)

Обновление содержимого фреймов:

66: http://www.w3schools.com/js/tryit.asp?filename=tryjs_alert
67: http://www.w3schools.com/js/tryit.asp?filename=tryjs_confirm
68: http://www.w3schools.com/js/tryit.asp?filename=tryjs_prompt
69: http://www.w3schools.com/js/tryit.asp?filename=tryjs_openwindow
70: http://www.w3schools.com/js/tryit.asp?filename=tryjs_openallwindow
71: http://www.w3schools.com/js/tryit.asp?filename=tryjs_multiwindows
72: http://www.w3schools.com/js/tryit.asp?filename=tryjs_location
73: http://www.w3schools.com/js/tryit.asp?filename=tryjs_reload
74: http://www.w3schools.com/js/tryit.asp?filename=tryjs_statusbar
75: http://www.w3schools.com/js/tryit.asp?filename=tryjs_print
76: http://www.w3schools.com/js/tryit.asp?filename=tryjs_breakout

Как с одного клика обновить содержимое двух фреймов:[77] (open editor)

Обновление содержимого фреймов (2):
Как из одного фрейма обновить содержимое двух других:[78] (open editor)

Примеры работы с формами

Примеры

Проверка адреса e-mail
Как проверить адрес e-mail, введенный в поле ввода:[79] (open editor)

Проверка значения:
Как проверить значение, введенное в поле ввода на вхождение в допустимый диапазон:[80] (open editor)

Проверка длины:
Как проверить количество символов, введенных в поле ввода:[81] (open editor)

Проверка формы:
Форма, содержащая все приведенные выше проверки:[82] (open editor)

Установка фокуса:
Как установить фокус на поле ввода:[83] (open editor)

Выделение:
Как сделать содержимое поле ввода выделенным:[84] (open editor)

Радиокнопка:
Как позволить клиенту делать выбор с помощью радиокнопок:[85] (open editor)

Checkbox:
Как позволить клиенту делать выбор с помощью чек-боксов:[86] (open editor)

Выпадающее меню:
Как позволить клиенту делать выбор с помощью выпадающего меню:[87] (open editor)

Выбор более, чем одной опции:
Как позволить клиенту выбрать несколько пунктов в выпадающем меню:[88] (open editor)

77: http://www.w3schools.com/js/tryit.asp?filename=tryjs_twoframes
78: http://www.w3schools.com/js/tryit.asp?filename=tryjs_threeframes
79: http://www.w3schools.com/js/tryit.asp?filename=tryjs_email
80: http://www.w3schools.com/js/tryit.asp?filename=tryjs_value
81: http://www.w3schools.com/js/tryit.asp?filename=tryjs_lengthvalidate
82: http://www.w3schools.com/js/tryit.asp?filename=tryjs_formvalidate
83: http://www.w3schools.com/js/tryit.asp?filename=tryjs_focus
84: http://www.w3schools.com/js/tryit.asp?filename=tryjs_select2
85: http://www.w3schools.com/js/tryit.asp?filename=tryjs_form_radio
86: http://www.w3schools.com/js/tryit.asp?filename=tryjs_form_checkbox
87: http://www.w3schools.com/js/tryit.asp?filename=tryjs_putdropdown
88: http://www.w3schools.com/js/tryit.asp?filename=tryjs_putmore

Примеры определения типа браузера

Примеры

Определение типа браузера:
Как определить тип браузера клиента:[89] (open editor)

Дополнительная информация:
Как получить более детальную информацию о клиенте:[90] (open editor)

Монитор:
Как получить информацию о мониторе клиента:[91] (open editor)

Редирект:
Как перенаправлять клиента на различные страницы в зависимовти от типа браузера:[92] (open editor)

Сообщения:
Как выдавать клиенту различные сообщения в зависимости от типа браузера:[93] (open editor)

Примеры по JavaScript

Более ста простых примеров действующих скриптов. Вы можете разобраться в их работе с помощью специального web-редактора

Основы

Выведение текста с помощью JavaScript:[94] (open editor)

Форматирование текста HTML-тэгами:[95] (open editor)

Размещение JavaScript

JavaScript, размещенный в разделе head:[96] (open editor)

JavaScript, размещенный в разделе body:[97] (open editor)

Внешний JavaScript:[98] (open editor)

89: http://www.w3schools.com/js/tryit.asp?filename=tryjs_browser
90: http://www.w3schools.com/js/tryit.asp?filename=tryjs_browserdetails
91: http://www.w3schools.com/js/tryit.asp?filename=tryjs_browsermonitor
92: http://www.w3schools.com/js/tryit.asp?filename=tryjs_crossbrowser
93: http://www.w3schools.com/js/tryit.asp?filename=tryjs_crossbrowser2
94: http://www.w3schools.com/js/tryit.asp?filename=tryjs_text
95: http://www.w3schools.com/js/tryit.asp?filename=tryjs_formattext
96: http://www.w3schools.com/js/tryit.asp?filename=tryjs_headsection
97: http://www.w3schools.com/js/tryit.asp?filename=tryjs_bodysection
98: http://www.w3schools.com/js/tryit.asp?filename=tryjs_externalexample

Переменные

Объявление переменной, установка ее значения и ее отображение:[99] (open editor)

Функции

Функция:[100] (open editor)

Функция с аргументом:[101] (open editor)

Функция с аргументом (2):[102] (open editor)

Функция, возвращающая значение:[103] (open editor)

Функция с аргументами, возвращающая значение:[104] (open editor)

Условные выражения

Выражение If:[105] (open editor)

Выражение If...else:[106] (open editor)

Случайная ссылка:[107] (open editor)

Выражение-переключатель Switch:[108] (open editor)

Циклы

Цикл for:[109] (open editor)

Циклическая обработка HTML-заголовков:[110] (open editor)

Цикл while:[111] (open editor)

Цикл do..while:[112] (open editor)

Объект String (строка)

Подсчет количества букв в строке (метод length()):[113] (open editor)

Проверка, содержит ли строка заданные символы (метод indexOf()):[114] (open editor)

99: http://www.w3schools.com/js/tryit.asp?filename=tryjs_variable
100: http://www.w3schools.com/js/tryit.asp?filename=tryjs_function1
101: http://www.w3schools.com/js/tryit.asp?filename=tryjs_function2
102: http://www.w3schools.com/js/tryit.asp?filename=tryjs_functionarg2
103: http://www.w3schools.com/js/tryit.asp?filename=tryjs_function_return2
104: http://www.w3schools.com/js/tryit.asp?filename=tryjs_function_return
105: http://www.w3schools.com/js/tryit.asp?filename=tryjs_ifthen
106: http://www.w3schools.com/js/tryit.asp?filename=tryjs_ifthenelse
107: http://www.w3schools.com/js/tryit.asp?filename=tryjs_randomlink
108: http://www.w3schools.com/js/tryit.asp?filename=tryjs_switch
109: http://www.w3schools.com/js/tryit.asp?filename=tryjs_fornext
110: http://www.w3schools.com/js/tryit.asp?filename=tryjs_fornext_header
111: http://www.w3schools.com/js/tryit.asp?filename=tryjs_while
112: http://www.w3schools.com/js/tryit.asp?filename=tryjs_dowhile
113: http://www.w3schools.com/js/tryit.asp?filename=tryjs_length

Проверка, содержит ли строка заданные символы (метод match()):[115] (open editor)

Получение заданной части строки (метод substr()):[116] (open editor)

Изменение регистра символов строки (методы toLowerCase() и toUpperCase()):[117] (open editor)

Объект Array (массив)

Сохранение множества имен в массиве:[118] (open editor)

Определение количества элементов в массиве:[119] (open editor)

Объект Date (дата)

Текущая дата:[120] (open editor)

Текущее местное время:[121] (open editor)

Установка даты:[122] (open editor)

Универсальное время:[123] (open editor)

Выведение текущего дня недели:[124] (open editor)

Выведение полной даты с названием дня недели и месяца:[125] (open editor)

Действующие часы:[126] (open editor)

Объект Math

Округление числа до ближайшего целого числа (метод round()):[127] (open editor)

Случайное число из промежутка от 0 до 1 (метод random()):[128] (open editor)

Случайное число от 0 до 10 (методы random() и round()):[129] (open editor)

Определение - какое из двух чисел больше (метод max()):[130] (open editor)

Определение - какое из двух чисел меньше (метод min()):[131] (open editor)

Перевод градусов Цельсия в градусы Фаренгейта:[132] (open editor)

114: http://www.w3schools.com/js/tryit.asp?filename=tryjs_indexof
115: http://www.w3schools.com/js/tryit.asp?filename=tryjs_match
116: http://www.w3schools.com/js/tryit.asp?filename=tryjs_substr
117: http://www.w3schools.com/js/tryit.asp?filename=tryjs_lowerupper
118: http://www.w3schools.com/js/tryit.asp?filename=tryjs_array
119: http://www.w3schools.com/js/tryit.asp?filename=tryjs_array_length
120: http://www.w3schools.com/js/tryit.asp?filename=tryjs_datedate
121: http://www.w3schools.com/js/tryit.asp?filename=tryjs_datetime
122: http://www.w3schools.com/js/tryit.asp?filename=tryjs_datesetfullyear
123: http://www.w3schools.com/js/tryit.asp?filename=tryjs_dateutcdate
124: http://www.w3schools.com/js/tryit.asp?filename=tryjs_date_weekday
125: http://www.w3schools.com/js/tryit.asp?filename=tryjs_date_fulldate
126: http://www.w3schools.com/js/tryit.asp?filename=tryjs_time
127: http://www.w3schools.com/js/tryit.asp?filename=tryjs_math_round
128: http://www.w3schools.com/js/tryit.asp?filename=tryjs_math_random
129: http://www.w3schools.com/js/tryit.asp?filename=tryjs_math_random09
130: http://www.w3schools.com/js/tryit.asp?filename=tryjs_math_max
131: http://www.w3schools.com/js/tryit.asp?filename=tryjs_math_min
132: http://www.w3schools.com/js/tryit.asp?filename=tryjs_celsius

Перевод символа в Unicode:[133] (open editor)

Работа с окнами

Изменение текста в тэге <title> документа (только для IE):[134] (open editor)

Создание выскакивающего сообщения:[135] (open editor)

Создание выскакивающего сообщения, содержащего переводы строки:[136] (open editor)

Запрет правого клика мышью (только для IE):[137] (open editor)

Выскакивающий запрос на подтверждение:[138] (open editor)

Создание запроса пользователю:[139] (open editor)

Открытие нового окна:[140] (open editor)

Указание вида нового окна:[141] (open editor)

Открытие нового окна максимального размера (только для IE):[142] (open editor)

Указание положения нового окна:[143] (open editor)

Закрытие окна:[144] (open editor)

Открытие множественных окон:[145] (open editor)

Location:[146] (open editor)

Создание кнопки, показывающей исходник:[147] (open editor)

Помещение страницы в Избранное (Add to favorites) (только для IE):[148] (open editor)

Страница становится страницей браузера по умолчанию (только для IE):[149] (open editor)

Обновление страницы:[150] (open editor)

Изменение текста в строке состояния:[151] (open editor)

Распечатка страницы:[152] (open editor)

Когда этот файл в последний раз подвергался модификации?:[153] (open editor)

Прокрутка страницы вниз:[154] (open editor)

133: http://www.w3schools.com/js/tryit.asp?filename=tryjs_unicode
134: http://www.w3schools.com/js/tryit.asp?filename=tryjs_newtitle
135: http://www.w3schools.com/js/tryit.asp?filename=tryjs_alert
136: http://www.w3schools.com/js/tryit.asp?filename=tryjs_alert2
137: http://www.w3schools.com/js/tryit.asp?filename=tryjs_noright
138: http://www.w3schools.com/js/tryit.asp?filename=tryjs_confirm
139: http://www.w3schools.com/js/tryit.asp?filename=tryjs_prompt
140: http://www.w3schools.com/js/tryit.asp?filename=tryjs_openwindow
141: http://www.w3schools.com/js/tryit.asp?filename=tryjs_openallwindow
142: http://www.w3schools.com/js/tryit.asp?filename=tryjs_fullsize
143: http://www.w3schools.com/js/tryit.asp?filename=tryjs_placenewwindow
144: http://www.w3schools.com/js/tryit.asp?filename=tryjs_closewindow
145: http://www.w3schools.com/js/tryit.asp?filename=tryjs_multiwindows
146: http://www.w3schools.com/js/tryit.asp?filename=tryjs_location
147: http://www.w3schools.com/js/demo_js_viewsource.htm
148: http://www.w3schools.com/js/tryit.asp?filename=tryjs_bookmark
149: http://www.w3schools.com/js/tryit.asp?filename=tryjs_homepage
150: http://www.w3schools.com/js/tryit.asp?filename=tryjs_reload
151: http://www.w3schools.com/js/tryit.asp?filename=tryjs_statusbar
152: http://www.w3schools.com/js/tryit.asp?filename=tryjs_print
153: http://www.w3schools.com/js/demo_js_lastmod.htm

Прокрутка страницы вправо:[155] (open editor)

Эффект, основанный на прокрутке:[156] (open editor)

Работа с картинками

Предварительная загрузка изображения:[157] (open editor)

Работа с фреймами

Удаление фреймов:[158] (open editor)

Обновление содержимого двух фреймов:[159] (open editor)

Обновление содержимого двух фреймов из третьего фрейма:[160] (open editor)

Обновление содержимого двух фреймов типа iframe (только для IE):[161] (open editor)

Работа с формами

Проверка адреса e-mail:[162] (open editor)

Проверка значения:[163] (open editor)

Проверка длины введенного текста:[164] (open editor)

Проверка формы:[165] (open editor)

Установка фокуса на поле ввода:[166] (open editor)

Выделение текста в поле ввода:[167] (open editor)

Радиокнопка:[168] (open editor)

Чек-бокс:[169] (open editor)

Выбор еа основе выпадающего меню:[170] (open editor)

Выбор нескольких опций:[171] (open editor)

Меню выбора (только для IE):[172] (open editor)

154: http://www.w3schools.com/js/tryit.asp?filename=tryjs_scrolldown
155: http://www.w3schools.com/js/tryit.asp?filename=tryjs_scrollright
156: http://www.w3schools.com/js/tryit.asp?filename=tryjs_scrollfx
157: http://www.w3schools.com/js/tryit.asp?filename=tryjs_preload
158: http://www.w3schools.com/js/tryit.asp?filename=tryjs_breakout
159: http://www.w3schools.com/js/tryit.asp?filename=tryjs_twoframes
160: http://www.w3schools.com/js/tryit.asp?filename=tryjs_threeframes
161: http://www.w3schools.com/js/tryit.asp?filename=tryjs_two_iframes
162: http://www.w3schools.com/js/tryit.asp?filename=tryjs_email
163: http://www.w3schools.com/js/tryit.asp?filename=tryjs_value
164: http://www.w3schools.com/js/tryit.asp?filename=tryjs_lengthvalidate
165: http://www.w3schools.com/js/tryit.asp?filename=tryjs_formvalidate
166: http://www.w3schools.com/js/tryit.asp?filename=tryjs_focus
167: http://www.w3schools.com/js/tryit.asp?filename=tryjs_select2
168: http://www.w3schools.com/js/tryit.asp?filename=tryjs_form_radio
169: http://www.w3schools.com/js/tryit.asp?filename=tryjs_form_checkbox
170: http://www.w3schools.com/js/tryit.asp?filename=tryjs_putdropdown
171: http://www.w3schools.com/js/tryit.asp?filename=tryjs_putmore
172: http://www.w3schools.com/js/tryit.asp?filename=tryjs_selectmenu

Установка фокуса на следующем поле ввода, когда на текущем достигнута максимально допустимая длина вводимой строки:[173] (open editor)

Информация о пользователе

Определение типа браузера клиента:[174] (open editor)

Дополнительная информация о клиенте:[175] (open editor)

Получение информации о мониторе клиента:[176] (open editor)

Переадресация пользователя на различные страницы в зависимости от типа браузера:[177] (open editor)

Выведение пользователю различных сообщений, в зависимости от типа браузера:[178] (open editor)

Элементы JavaScript

Свойства и методы основных объектов JavaScript, информация о поддержке браузерами Netscape Navigator (NN) и Internet Explorer (IE)

Объект Boolean

Метод	Описание	NN	IE
`toString()`	Возвращает булевское значение в виде строки (true или false)	3.0	3.0
`valueOf()`	Возвращает значение заданного объекта	4.0	4.0

Объект String (строка)

Метод	Описание	NN	IE
`length`	Возвращает длину строки	2.0	3.0
`anchor()`	Возвращает строку внутри тэга <a>: <a>строка	2.0	3.0
`big()`	Возвращает строку, отфор-	2.0	3.0

173: http://www.w3schools.com/js/tryit.asp?filename=tryjs_autonext
174: http://www.w3schools.com/js/tryit.asp?filename=tryjs_browser
175: http://www.w3schools.com/js/tryit.asp?filename=tryjs_browserdetails
176: http://www.w3schools.com/js/tryit.asp?filename=tryjs_browsermonitor
177: http://www.w3schools.com/js/tryit.asp?filename=tryjs_crossbrowser
178: http://www.w3schools.com/js/tryit.asp?filename=tryjs_crossbrowser2

	матированную тэгом <big>: <big>строка</big>		
blink()	Возвращает мигающую строку: <blink>строка</blink>	2.0	2.0
bold()	Возвращает полужирную строку: строка	2.0	3.0
charAt()	Возвращает символ, имеющий заданный индексный номер	2.0	3.0
charCodeAt()	Возвращает Unicode символа с заданным индексным номером	4.0	4.0
concat()	Возвращает две объединенных строки	4.0	4.0
fixed()	Возвращает строку в стиле телетайпа: <tt>строка</tt>	2.0	3.0
fontColor()	Возвращает строку заданного цвета: строка	2.0	3.0
fontSize()	Возвращает строку с шрифтом заданного размера: строка	2.0	3.0
fromCharCode()	Противоположность метода charCodeAt. Возвращает строчного значения заданного Unicode.	4.0	4.0
indexOf()	Возвращает индексный номер появления в строке заданного символа, или число -1, если данного символа в строке не обнаружено. С помощью этого метода можно определить, содержится ли в строке заданный символ.	2.0	3.0
italics()	Возвращает строку курсивом: строка	2.0	3.0
lastIndexOf()	То же самое, что и indexOf, только метод действует с конца строки и двигается налево.	2.0	3.0
link()	Возвращает строку как гиперссылку: строка	2.0	3.0
match()	Действует подобно методам indexOf и lastIndexOf, но вместо числового значения возвращает заданный символ или "null".	4.0	4.0
replace()	Заменяет заданные сим-	4.0	4.0

	волы другими заданными символами.		
search()	Возвращает целое числовое значение, если строка содержит заданные символы, или -1, если не содержит.	4.0	4.0
slice()	Возвращает строку, содержащую заданный набор символов.	4.0	4.0
small()	Возвращает строку, отформатированную тэгом <small>: <small>строка</small>	2.0	3.0
split()	Заменяет заданные символы запятой.	4.0	4.0
strike()	Возвращает перечеркнутую строку: <strike>строка</strike>	2.0	3.0
sub()	Возвращает строку, отформатированную тэгом <sub>: _{строка}	2.0	3.0
substr()	Возвращает заданные символы. (14,7) возвращает 7 символов, начиная с 14-го символа в строке.	4.0	4.0
substring()	Возвращает заданные символы. (14,7) возвращает все символы между 7-м и 14-м.	2.0	3.0
sup()	Возвращает строку, отформатированную тэгом <sup>: ^{строка}	2.0	3.0
toLowerCase()	Возвращает строку в нижнем регистре	2.0	3.0
toUpperCase()	Возвращает строку в верхнем регистре	2.0	3.0

Объект Array (массив)

Метод	Описание	NN	IE
length	Возвращает число элементов в массиве	3.0	4.0
concat()	Возвращает массив - объединение двух других массивов	4.0	4.0
join()	Возвращает строку, состоящую из всех объединенных элементов массива	3.0	4.0

`reverse()`	Возвращает массив, в котором порядок следования элементов перевернут	3.0	4.0
`slice()`	Возвращает заданную часть массива	4.0	4.0
`sort()`	Возвращает отсортированный массив	3.0	4.0

Объект Date (дата)

Метод	Описание	NN	IE
`Date()`	Возвращает объект Date	2.0	3.0
`getDate()`	Возвращает день месяца объекта Date (от 1 до 31)	2.0	3.0
`getDay()`	Возвращает день недели объекта Date (от 0 до 6. 0=воскресенье, 1=понедельник, и т.д.)	2.0	3.0
`getMonth()`	Возвращает месяц объекта Date (от 0 до 11. 0=январь, 1=февраль, и т.д.)	2.0	3.0
`getFullYear()`	Возвращает год объекта Date (четыре цифры)	4.0	4.0
`getYear()`	Возвращает год объекта Date (от 0 до 99). Используйте лучше метод getFullYear!	2.0	3.0
`getHours()`	Возвращает час объекта Date (от 0 до 23)	2.0	3.0
`getMinutes()`	Возвращает минуту объекта Date (от 0 до 59)	2.0	3.0
`getSeconds()`	Возвращает секунду объекта Date (от 0 до 59)	2.0	3.0
`getMilliseconds()`	Возвращает милисекунды объекта Date (от 0 до 999)	4.0	4.0
`getTime()`	Возвращает число милисекунд, котороые прошли с полуночи 1 января 1970 года	2.0	3.0
`getTimezoneOffset()`	Возвращает разницу местного времени и Гринвинча	2.0	3.0
`getUTCDate()`	Возвращает день месяца объекта Date в универсальном времени (UTC)	4.0	4.0
`getUTCDay()`	Возвращает день недели объекта Date в универсальном времени (UTC)	4.0	4.0

getUTCMonth()	Возвращает месяц объекта Date в универсальном времени (UTC)	4.0	4.0
getUTCFullYear()	Возвращает год объекта Date в универсальном времени (UTC) (четыре цифры)	4.0	4.0
getUTCHourc()	Возвращает час объекта Date в универсальном времени (UTC)	4.0	4.0
getUTCMinutes()	Возвращает минуту объекта Date в универсальном времени (UTC)	4.0	4.0
getUTCSeconds()	Возвращает секунду объекта Date в универсальном времени (UTC)	4.0	4.0
getUTCMilliseconds()	Возвращает милисекунду объекта Date в универсальном времени (UTC)	4.0	4.0
parse()	Возвращает строку, содержащую количество милисекунд прошедших начиная с 1 января 1970 года (полночь)	2.0	3.0
setDate()	Устанавливает день месяца объекта Date (от 1 до 31)	2.0	3.0
setFullYear()	Устанавливает год объекта Date (четыре цифры)	4.0	4.0
setHours()	Устанавливает час объекта Date (от 0 до 23)	2.0	3.0
setMilliseconds()	Устанавливает милисекунду объекта Date (от 0 до 999)	4.0	4.0
setMinutes()	Устанавливает минуту объекта Date (от 0 до 59)	2.0	3.0
setMonth()	Устанавливает месяц объекта Date (от 0 до 11. 0=январь, 1=февраль, и т.д.)	2.0	3.0
setSeconds()	Устанавливает секунду объекта Date (от 0 до 59)	2.0	3.0
setTime()	Устанавливает количество милисекунд, прошедшее с 1 января 1970 года	2.0	3.0
setYear()	Устанавливает год объекта Date (00-99)	2.0	3.0
setUTCDate()	Устанавливает день месяца объекта Date в универсальном времени (от 1 до 31)	4.0	4.0
setUTCDay()	Устанавливает день недели объекта Date в универсальном времени (от 0 до 6.	4.0	4.0

	0=воскресенье, 1=понедельник, и т.д.)		
`setUTCMonth()`	Устанавливает месяц объекта Date в универсальном времени (от 0 до 11. 0=январь, 1=февраль, и т.д.)	4.0	4.0
`setUTCFullYear()`	Устанавливает год объекта Date в универсальном времени (четыре цифры)	4.0	4.0
`setUTCHour()`	Устанавливает час объекта Date в универсальном времени (от 0 до 23)	4.0	4.0
`setUTCMinutes()`	Устанавливает минуту объекта Date в универсальном времени (от 0 до 59)	4.0	4.0
`setUTCSeconds()`	Устанавливает секунду объекта Date в универсальном времени (от 0 до 59)	4.0	4.0
`setUTCMilliseconds()`	Устанавливает милисекунду объекта Date в универсальном времени (от 0 до 999)	4.0	4.0
`toGMTString()`	Преобразует объект Date в строку, устанавливает его на гринвинчевское время	2.0	3.0
`toLocaleString()`	Преобразует объект Date в строку, устанавливает его на местное время	2.0	3.0
`toString()`	Преобразует объект Date в строку	2.0	4.0

Объект Math

Свойство	Описание	NN	IE
`E`	Возвращает основание натуральных логарифмов	2.0	3.0
`LN2`	Возвращает натуральный логарифм 2	2.0	3.0
`LN10`	Возвращает натуральный логарифм 10	2.0	3.0
`LOG2E`	Возвращает логарифм числа E по основанию 2	2.0	3.0
`LOG10E`	Возвращает десятичный логарифм числа E	2.0	3.0
`PI`	Возвращает число PI	2.0	3.0
`SQRT1_2`	Возвращает 1, деленое на корень квадратный из 2	2.0	3.0

SQRT2	Возвращает квадратный корень из 2	2.0	3.0

Метод	Описание	NN	IE
abs(x)	Возвращает абсолютное значение x	2.0	3.0
acos(x)	Возвращает аркосинус x	2.0	3.0
asin(x)	Возвращает арксинус x	2.0	3.0
atan(x)	Возвращает арктангенс x	2.0	3.0
atan2(x,y)	Returns the angle from the x axis to a point	2.0	3.0
ceil(x)	Возвращает ближайшее целое число, большее, чем x	2.0	3.0
cos(x)	Возвращает косинус x	2.0	3.0
exp(x)	Возвращает экспоненту x	2.0	3.0
floor(x)	Возвращает ближайшее целое число, меньшее, чем x	2.0	3.0
log(x)	Возвращает натуральный логарифм x	2.0	3.0
max(x,y)	Возвращает число, большее из пары x и y	2.0	3.0
min(x,y)	Возвращает число, меньшее из пары x и y	2.0	3.0
pow(x,y)	Возвращает x в степени y	2.0	3.0
random()	Возвращает случайное число из промежутка от 0 до 1	2.0	3.0
round(x)	Округляет x к ближайшему целому числу	2.0	3.0
sin(x)	Возвращает синус x	2.0	3.0
sqrt(x)	Возвращает квадратный корень x	2.0	3.0
tan(x)	Возвращает тангенс x	2.0	3.0