JavaScript

Шестаков А.П.

Введение

Язык ***JavaScript*** разработан фирмой Netscape для создания интерактивных HTML-документов.

Области применения:

- создание динамических страниц, т.е. их содержание может меняться после загрузки документа;
- проверка правильности заполнения форм пользователем до пересылки их на сервер;
- решение локальных задач с помощью сценариев

Программа (сценарий, скрипт) на языке JavaScript представляет собой последовательность операторов.

Литералы

Простейшие данные, с которыми может оперировать программа, называются *литералами.*

Типы литералов

- Целые
  - Десятичные (0, -17, 22)
  - Шестнадцатеричные (0xAFF, 0x123, 0x2E)
  - Восьмеричные (0123, 056, 04)
- Вещественные (123.24, -9.0, 1e-6)
- Логические (true, false)
- Строковые (“строка”)

Переменные

Переменные используются для хранения данных

Типы переменных – те же, что и для литералов

Объявляются без указания типа (он определяется при задании значения)

Например

```
var t
var x=-2.2
var stroka=“Результат работы программы: “
```

Правила описания и использования переменных в JavaScript

- переменные начинаются с *VAR* (от слова variable), следом идет имя, знак = и значение переменной. Служебное слово *VAR* можно опускать;
- имя переменной может состоять из любого количества букв. Но лучше давать им названия, имеющие соответствующее смысловое значение;
- имена переменных различают регистр. Таким образом ***X*** и ***x*** — это две разные переменные;
- значение текстовой переменной ставится в кавычки. Числовые переменные не ставятся в кавычки, иначе интерпретатор поймет их как текст с числовым значением 0;
- в JavaScript, как и в других языках программирования, есть зарезервированные слова, например, названия команд. Этими словами называть переменные нельзя;
- если необходимо, вместо пробела можно ставить знак подчеркивания, например, *_user_name.*

Пример скрипта

В HTML–документе скрипт помещается внутри парных тэгов <SCRIPT> и </SCRIPT> непосредственно на HTML–странице.

Рассмотрим следующий простой *пример*:

```
<HTML>
<BODY>
<BR>
Текст HTML документа
<BR>
    <SCRIPT language="JavaScript">
      document.write("А здесь начинается JavaScript")
    </SCRIPT>
<BR>
Продолжение HTML документа
</BODY>
</HTML>
```

Все, что стоит между тэгами <SCRIPT> и </SCRIPT>, интерпретируется браузером как код на языке JavaScript.

Операции в JavaScript

Присваивания: = += –= *= /= %=

Операции += –= *= /= являются упрощенными аналогами следующих выражений:

a = a + b или a += b

a = a – b или a –= b

*a = a * b или a *= b*

a = a / b или a /= b

Пример:

```
x = 3;   //После выполнения фрагмента программы:
y = 7;    //x равен 10, y равен 7
x += y;
x = 21;   //После выполнения фрагмента программы:
y = 7;    //x равен 3, y равен 7
x /= y;
```

Операции в JavaScript

Отношения: < > <= >= == (равно) != (не равно)

Арифметические: + – * / % ++ --

++ – операция инкремента: i++ или ++i или i = i+1

-- – операция декремента: i-- или --i или i = i-1

Различие в записях инкремента (++i и i++) и декремента (--i и i--), заключается в том, что указанная операция будет выполняться либо до использования переменной (++i, --i), либо после (i++, i--).

Пример:

x = 12; //После выполнения фрагмента программы:

y = ++x; //x равен 13, y равен 13

x = 12; //После выполнения фрагмента программы:

y = x++; //x равен 13, y равен 12

Операции в JavaScript

- *Побитовые:* `& | ^ ~ << >> >>>`
- *Логические:*

  || (или)

  && (и)

  ! (отрицание)

Выражения

Выражение – любой имеющий силу набор литералов, переменных, операторов и соотношений, которые вычисляют простое значение. Значением может быть число, строка или логическое значение.

Правила построения выражений:

- выражение может занимать как одну строчку, так и несколько;
- выражение может содержать любое количество операторов и операндов, но поскольку слишком сложные выражение трудны для понимания, рекомендуется разбивать их на более простые.

Примеры операторов, содержащих выражения:

```
a = 241;
x = y*2 + f(y) - z--;
```

Методы alert, prompt, confirm

В языке JavaScript существует ряд методов объекта **window**, которые используются достаточно часто.

Основные из них:

alert*(строка)* – вызывает диалоговое окно с текстом (строка) и кнопкой OK.

confirm*(строка)* – вызывает диалоговое окно с текстом (строка) и кнопками OK и Cancel. Функция возвращает значение true, если пользователь нажал кнопку OK и false – если кнопку Cancel.

prompt*(строка, значение)* – вызывает диалоговое окно с надписью «строка» и с полем для ввода текста («значение» – устанавливается по умолчанию). Возвращает введенное пользователем текстовое значение.

Операторы

Условный оператор

Оператор if выполняет проверку значения выражения, стоящего после if, после чего, если результат true (не нуль), выполняет блок операторов 1. В случае, когда результат false (нуль) и если присутствует часть else, будет выполнен блок операторов из else, в противном случае – ничего.

```
if (выражение)
  {
   блок операторов 1;
  }
 else
    {
     блок операторов 2;
    }
```

Пример: if (age >= 18) status = ‘adult’; else status = ‘minor’;

В качестве выражения в операторе if можно записывать не только логические, но арифметические и строковые выражения. При этом считается, что условие ложно, если значение выражения равно 0 или пустой строке ‘’, и истинно в противном случае.

Операторы

Иногда альтернативой оператору if может служить операция **?** (если в результате возвращается одно значение, которое можно присвоить переменной).

*переменная = (выражение) **?** { блок операторов 1 } **:** { блок операторов 2 }*

Пример

status = (age >= 18) ? ‘adult’ : ‘minor’;

Операторы

Цикл **for**

for ([инициализация]; [выражение]; [изменение выражения])
{
    операторы;
}

позволяет многократно выполнять операторы в программе.

Пример
var n=1234;
var k=0;
for (; n;)
  {k++; n=Math.floor(n/10);}

Операторы

Цикл **while**
while (выражение)
{
операторы
}

выполнение операторов до тех пор, пока заданное после while выражение истинно (не равно нулю).

Пример

```
var n=1234;
var k=0;
while (n)
  {k++; n=Math.floor(n/10);}
```

Операторы

Цикл **do … while**

```
do
{
    операторы тела цикла
}
while (выражение);
```

выполнение операторов до тех пор, пока заданное после while выражение истинно (не равно нулю).

Пример

```
var n=1234;
var k=0;
    do  {k++; n=Math.floor(n/10);} while (n);
```

Функции

Функции — один из основных способов объединения операторов в блоки. Функция представляет собой группу операторов, решающих какую-либо определенную задачу. В JavaScript функции, не возвращающие результатов, можно рассматривать как процедуры.

Описание функции и ее вызов:

function Имя_Функции (список формальных параметров)

{

тело функции;

return значение;

}

Рекомендуется определять функции в части HEAD для ускорения работы скрипта и, следовательно, просмотра Web–страницы.

Команда вызова функции имеет вид:

Имя_Функции (список фактических параметров)

Функции

Аргументы функции хранятся в массиве. Внутри функции можно адресовать параметры следующим образом:

Имя_Функции.arguments[i],

где *i* – порядковый номер аргумента, начиная с нуля. Общее число аргументов обозначено переменной ***Имя_Функции.arguments.length***. Существует возможность вызывать функцию с большим количеством аргументов, чем она формально объявлена, используя массив arguments. Это бывает полезным, когда заранее не известно, сколько аргументов будет в функции. Схема проста: используя свойство ***arguments.length***, определяется число аргументов в функции, и затем организуется обращение к каждому аргументу при помощи массива ***arguments***.

Функции

Пример. Опишем функцию, которая будет создавать списки HTML. Единственный формальный аргумент функции – строка, имеющая значение "U", если список неупорядочен, или "O", если список упорядочен (пронумерован).

```
function List(type)
{
    document.write('<' + type + 'L>');
    for (var i = 1; i < arguments.length; i++)
      document.write('<LI>' + arguments[i] + '</LI>');
    document.write('</' + type + 'L>');
}
```

Вызов функции:

```
List('U','Пункт 1','Пункт 2','Пункт 3','Пункт 4');
```

Объект Math

Метод	Описание
Math.abs (x)	Возвращает абсолютное значение своего аргумента
Math.acos (x)	Возвращает арккосинус аргумента
Math.asin (x)	Возвращает арксинус аргумента
Math.atan (x)	Возвращает арктангенс аргумента
Math.atan2 (y, x)	Возвращает угол в радианах между осью абсцисс и линией, соединяющей начало координат с точкой (x, y)
Math.ceil (x)	Возвращает целое число, большее или равное аргументу
Math.cos (x)	Возвращает косинус аргумента
Math.exp (x)	Возвращает e^x
Math.floor (x)	Возвращает целое число, меньшее или равное аргументу

Объект Math

Метод	***Описание***
Math.log (x)	Возвращает натуральный логарифм аргумента
Math.max (x, y)	Возвращает большее из значений аргументов
Math.min (x, y)	Возвращает меньшее из значений аргументов
Math.pow (x, y)	Возвращает x^y
Math.random ()	Возвращает случайное число на полуинтервале [0, 1)
Math.round (x)	Округляет аргумент до ближайшего целого
Math.sin (x)	Возвращает синус аргумента
Math.sqrt (x)	Возвращает корень квадратный аргумента
Math.tan (x)	Возвращает тангенс аргумента

Объект Date

Метод	Описание
getDate ()	Возвращает день месяца (целое число в диапазоне от 1 до 31), например: **var D = new Date(); var date = D.getDate();**
getDay ()	Возвращает день недели (целое число в диапазоне от 0 до 6, нулю соответствует воскресенье).
getFullYear ()	Возвращает год.
getHours ()	Возвращает целое число часов в диапазоне от 0 до 23, прошедших с начала суток по местному времени.
getMinutes ()	Возвращает целое число минут в диапазоне от 0 до 59.
getSeconds ()	Возвращает целое число секунд в диапазоне от 0 до 59.
getMilliseconds ()	Возвращает целое число миллисекунд в диапазоне от 0 до 999.
getMonth ()	Возвращает месяц в диапазоне от 0 до 11

Объект Date

Метод	Описание
getTime ()	Число миллисекунд, прошедших с полуночи 01.01.1970
getTimezoneOffset ()	Разность между гринвичским и местным временем в минутах
setDate (date)	Задает день месяца. Если, например, текущая дата соответствует июлю 2000 года, то D.setDate (31) задаст 31.07.2000, D. setDate (32) — 01.08.2000.
setFullYear (year)	Задает год. В отличие от конструктора с помощью данного метода можно задавать любой год
setHours (hours)	Задание часа в диапазоне от 0 до 23. Если задать час меньше 0 или больше 23, то осуществляется переход к предыдущим и последующим суткам, соответственно

Объект Date

Метод	Описание
setMinutes (minutes)	Задание минуты в диапазоне от 0 до 59.
setSeconds (seconds)	Задание секунды в диапазоне от 0 до 59.
setMilliseconds (ms)	Задание миллисекунды в диапазоне от 0 до 999.
setMonth(month)	Задание месяца в диапазоне от 0 до 11
setTime(ms)	Задание времени в миллисекундах с полуночи 01.01.1970.
toGMTString ()	Возвращает дату, преобразованную в строку, время гринвичское
toLocaleString ()	Возвращает дату, преобразованную в строку, время местное. Отображение локального времени и даты существенно зависит от браузера, рекомендуется формировать его вручную

Объектная модель документа

Объектная модель документа (**Document Object Model – DOM**) обеспечивает программный интерфейс для HTML-документов. Она определяет логическую структуру документов и способы взаимодействия с ними.

Все встроенные объекты JavaScript берут свое начало от рабочей области документа.

Кроме этих классов объектов пользователь может создавать и свои собственные. Но обычно большинство программ используют эту систему классов и не создают новых.

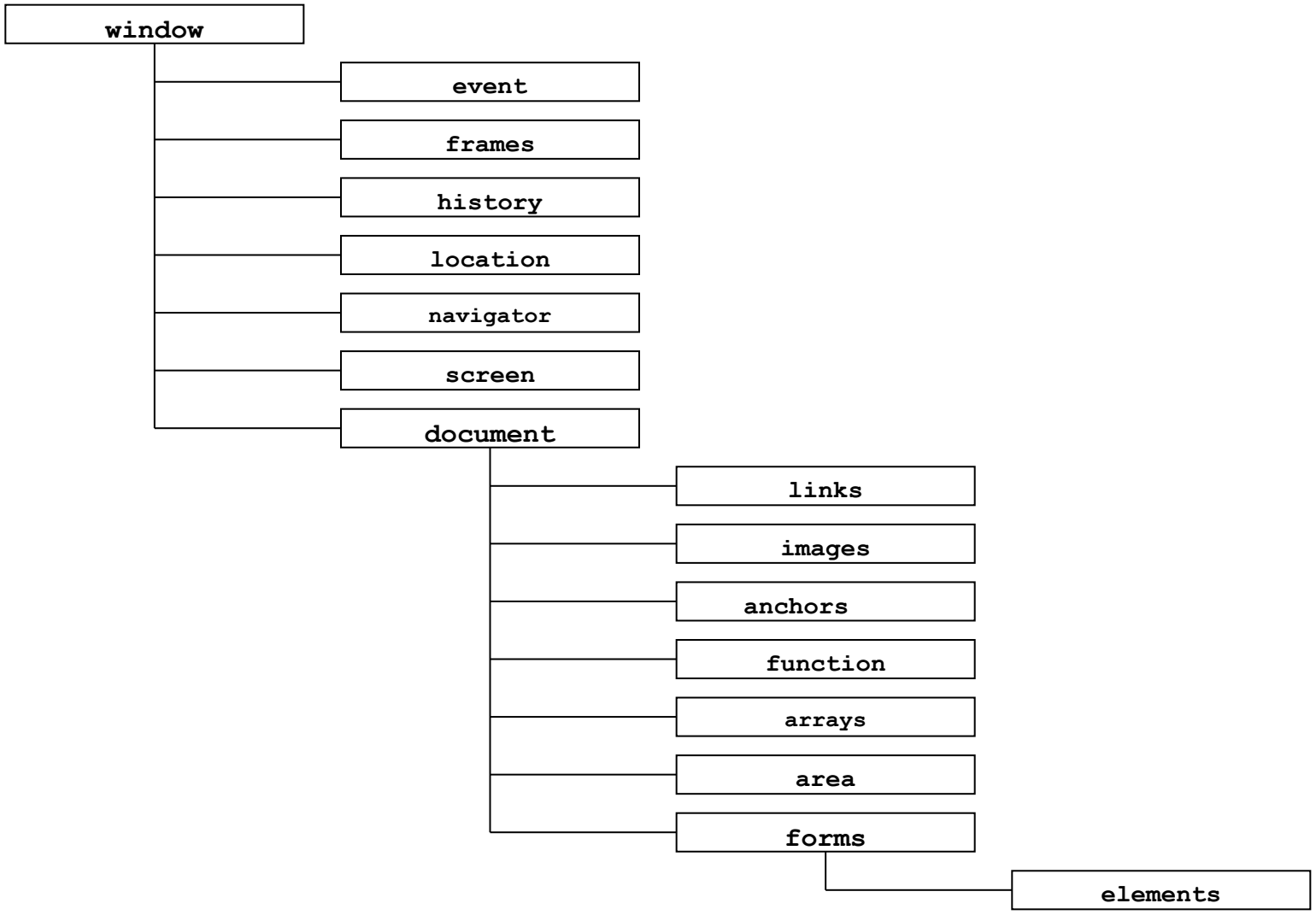
Все объекты в JavaScript берут свое начало от трех родительских объектов:

Global — содержит глобальные переменные.

Math — содержит большое количество разнообразных математических функций.

Object — предок всех остальных встроенных классов.

Объектная модель документа



Объект Window

Свойство	Описание
closed	Свойство хранит логическое значение, указывающее, закрыто ли окно
defaultStatus	Значение – строка, отображающаяся в строке состояния окна браузера
document	Ссылка на объект document
frames	Массив фреймов, загруженных в окно
history	Значение – объект history, содержащий массив адресов ресурсов, просмотренных в данном окне
location	Адрес ресурса, загруженного в окно
name	Имя окна
navigator	Ссылка на объект navigator, содержащий информацию о браузере

Объект Window

Свойство	Описание
opener	Ссылка на окно, открывшее данное
parent	Ссылка на родительское окно или фрейм, содержащий данный фрейм
screen	Информация о разрешении экрана
self	Ссылка на текущее окно
status	Задание строки, которая будет отображаться в строке состояния
top	Ссылка на ближайшее к пользователю окно

Объект Window

Метод	Описание
blur()	При вызове метода окно теряет фокус
close()	Окно закрывается, фокус передается родительскому окну
focus()	Фокус передается заданному окну
moveBy(x, y)	
moveTo(x, y)	
open(url, wn, f)	
print()	
resizeBy(x, y)	

Объект Window

Метод	Описание
resizeTo(x, y)	
scrollTo(x, y)	
scrollBy(x, y)	
stop()	

Объект Window

Событие	Описание
onBlur	Возникает при потере фокуса окном
onError	Возникает в случае ошибки при загрузке документа в окно
onFocus	Возникает при получении фокуса окном
onLoad	Возникает при завершении загрузки документа в окно
onResize	Возникает при изменении размеров окна или фрейма
onUnLoad	Возникает при завершении работы пользователя с документом, например, при переходе к другому документу

Объект history

Свойство или метод	Описание
length	Свойство хранит число элементов в объекте
back()	Метод загружает предыдущий элемент списка
forward()	Метод загружает следующий элемент списка
go(n)	Метод загружает *n*-й элемент списка (нумерация осуществляется с нуля)

Объект location

Свойство	Описание
href	Полный унифицированный указатель ресурса документа, загружаемого в окно браузера
hash	Часть унифицированного указателя ресурса - закладка, показывающая положение фрагмента в документе, следует за символом «#»
host	Часть унифицированного указателя ресурса, содержащая доменный или IP-адрес ресурса и номер порта
hostname	Часть унифицированного указателя ресурса, содержащая доменный или IP-адрес ресурса и номер порта
pathname	Часть унифицированного указателя ресурса, содержащая каталог и имя документа
port	Номер порта, используемого сервером
protocol	Протокол, обеспечивающий загрузку документа, включая двоеточие

Объект location

Метод	Описание
reload()	Обеспечивает перезагрузку текущего документа в окно браузера
replace(url)	Метод загружает документ с заданным url, заменяет объект history так, что становится невозможным перейти к замененному документу с помощью кнопки **Назад** или метода history.back

Объект document

Свойство	Описание
alinkColor	Цвет активной гиперссылки
linkColor	Цвет текста гиперссылки
vlinkColor	Цвет текста гиперссылки на просмотренный документ
bgColor	Цвет фона документа
fgColor	Цвет текста в документе
URL	Строка – URL документа
title	Строка – заголовок документа

Объект document

Метод	Описание
close()	Закрывает выходной поток вывода данных в документ
open()	Открывает выходной поток вывода данных в документ
write(t1, t2, …, tn)	Вывод текста в документ без перевода строки
writeln(t1, t2, …, tn)	Вывод текста в документ с переводом строки

Объект document

Коллекция	Описание
all	Коллекция всех тегов и элементов документа
anchors	Коллекция закладок документа
forms	Коллекция всех форм документа
links	Коллекция всех гиперссылок документа
plugins	Коллекция встроенных модулей, доступных в документе
images	Коллекция всех изображений в документе

Объект document

Способы обращения к элементам коллекции:

1) как к элементам массива

Например, **document.forms[1]**

2) по имени элемента коллекции

Например, **document.images[“ris.jpg”]**

Объект document

Событие	Описание
onclick	Происходит при щелчке одной из кнопок мыши в области документа
onkeydown	Происходит при нажатии одной из клавиш клавиатуры
onkeypressed	Происходит при нажатии и удерживании одной из клавиш клавиатуры
onkeyup	Происходит, когда пользователь отпускает клавишу клавиатуры
onmousedown	Происходит при нажатии одной из кнопок мыши

Объект document

onmousemove	Происходит при перемещении курсора мыши в области документа
onmouseout	Происходит при выходе курсора мыши за область документа
onmouseover	Происходит при входе курсора мыши в область документа
onmouseup	Происходит, когда пользователь отпускает ранее нажатую кнопку мыши

События

Событие – действие браузера или пользователя.

Пользователь манипулирует мышью и клавиатурой, браузер генерирует «события». Обработка нужных событий предусматривается в скриптовых кодах, и документ начинает реагировать на перемещение мыши, нажатия клавиш на клавиатуре, окончание загрузки документа по сети, закрытие окна браузера и т.д.

События

Событие onMouseDown возникает при нажатии левой кнопкой мыши. В следующем примере при нажатии кнопки мыши на тексте меняется стиль оформления текста (текст выделяется жирным шрифтом).

Код	Отображение
Пример использования события onMouseDown.	Пример использования события onMouseDown.

События

Пример: создадим в документе следующую гиперссылку:

<A HREF *= “My_research1.htm”>Моя исследовательская работа****</A>***

— браузер передаст событие объекту, построенному для тега **<A>**.

При щелчке мышкой по тексту *Моя исследовательская работа* произойдет вызов стандартной обработки этого события — в окно загрузится документ *My_research1.htm*.

Добавим свой обработчик данного события:

<A HREF *= “My_research1.htm”* ***onClick****=”alert(“Эта информация строго конфиденциальна”)”>Моя исследовательская работа****</A>***

Теперь, при щелчке мышкой по тексту *Моя исследовательская работа* на экране появится надпись: *Эта информация строго конфиденциальна*. А затем в окно загрузится документ *My_research1.htm*.