Синтаксические основы JavaScript

Богданов Марат Робертович
Современные веб-технологии.
Подробный курс

JavaScript имеет Си-подобный синтаксис, что особенно заметно проявляется на примере комментариев, операторов присваивания, в циклах и условных операторах.

Вместе с тем, ввод-вывод заметно отличается от такового в Си и больше напоминает Visual Basic.Net (оператор alert похож на MsgBox).

Еще одним существенным отличием от Си является слабая типизация и автоматическое преобразование типов.

Комментарии

В JavaScript как и в C++ существуют однострочные и многострочные комментарии. Синтаксис комментариев следующий:

```
//Однострочный комментарий

/*
Многострочный комментарий, используется для блока текста
*/
```

Ввод-вывод

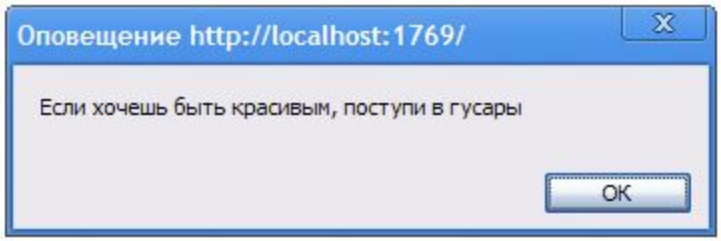
Для ввода и вывода данных можно воспользоваться методами браузера и загруженного в него документа. Объект, представляющий свойства браузера, называется window (окно), а три его метода— alert(), prompt() и confirm() — предназначены для ввода и вывода данных посредством диалоговых окон.

Метод alert()

Данный метод позволяет выводить диалоговое окно с заданным сообщением и кнопкой ОК.

Пример:

```
alert("Если хочешь быть красивым, поступи в гусары");
```



Метод confirm()

Метод confirm позволяет вывести диалоговое окно с сообщением и двумя кнопками: ОК и Отмена (Cancel).

Синтаксис метода confirm() следующий:

confirm(сообщение)

Пример:

```
confirm("Выйти из программы?");
```

Метод prompt()

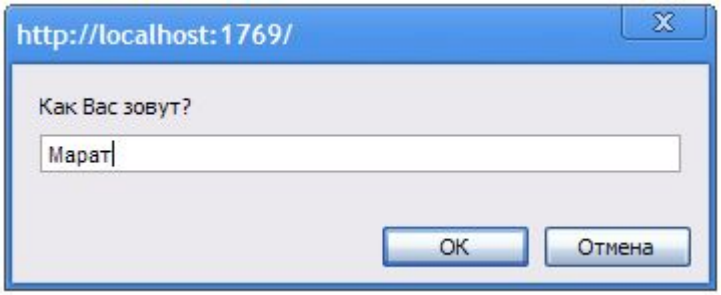
Позволяет вывести на экран диалоговое окно с сообщением, а также с текстовым полем, в которое пользователь может ввести данные. Кроме того, в окне предусмотрены две кнопки: ОК и Отмена (Cancel).

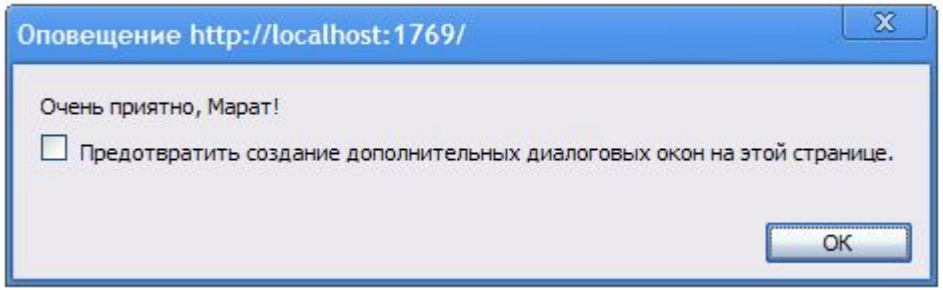
Синтаксис метода prompt () следующий:

prompt (сообщение, зна чение_поля_ввода_ланных)

Пример:

```
var FirstName,str;
FirstName=prompt("Как Вас зовут?", "");
str = "Очень приятно, " + FirstName + "!";
alert(str);
```





Метод document.write()

Позволяет выводить в окне браузера список текстовых строк, разделенных запятыми. Если выводимая строка представляет собой HTML-код, то браузер отобразит результат его интерпретации, а не последовательность тегов. Иначе говоря, теги HTML в выводимом сообщении будут восприниматься браузером как команды, а не просто как текстовые символы.

Пример:

```
var str;
str = "<h1>" + "Бросая в воду камешки, смотри на круги, ими
    образуемые," + "<br>" +
    "иначе такое бросание будет пустою забавою" + "</h1>";
document.write(str);
```

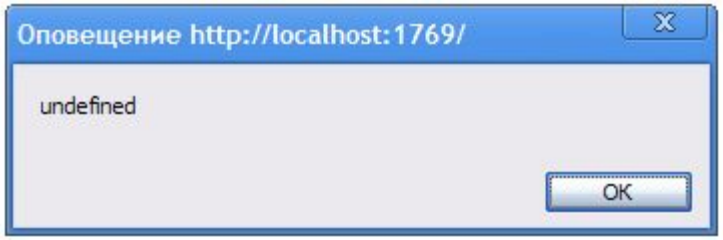
Бросая в воду камешки, смотри на круги, ими образуемые, иначе такое бросание будет пустою забавою

Переменные

В языке программирования JavaScript поддерживаются пять примитивных типов данных (**числовой, строковый, логический, пустой и неопределенный) и один составной, объектный (Object)**

Переменные объявляются с помощью ключевого слова **var**. При создании переменной она приобретает значение **undefined**.

```
var  str;
alert(str);
```



Автоматическое преобразование типов данных

JavaScript является языком со слабой типизацией, это означает, что в процессе работы программы тип переменной может меняться. Проиллюстрируем сказанное на следующем примере:

```
var x, y, z, str;
x = 3;      //Числовой тип
y = true;   //Логический тип
z = "Вася";  //Символьный тип
str = x+ y + z;
alert(str);
```

Явное преобразование типов

Функция parseInt(строка, основание) преобразует строку в целое десятичное число, функция parseFloat(строка) – преобразует строку в десятичное число с плавающей запятой.

```
var a, b, c, d, str;
a = parseInt("3.14159158");
b = parseInt("40.5 рублей 67 копеек");
c = parseFloat("3.14159158");
d = parseFloat("40.5 рублей 67 копеек");

str = a + "<br>" + b + "<br>" + c + "<br>" + d;
document.write(str);
```

```
3
40
3.14159158
40.5
```

Оператор присваивания

Рассмотри различные формы присваивания на следующем примере.

```
var a, b, c, d, f, str;
a = 1;  //Наиболее привычная форма записи
b = 13;
c = 2;
d = 5;
f = 15;

b += a; //То же, что и b=b+a
c++;    //То же, что и c=c+1
d *= 2; //То же, что и d=d*2
f %= 7; //Нахождение остатка при делении 15 на 7

str = b + "<br>" + c + "<br>" + d + "<br>" + f;
document.write(str);
```

```
14
3
10
1
```

Условные операторы

Условные операторы изменяют ход выполнения программы в зависимости от некоего условия. Существуют конструкции if…then, условная операция (?) и переключатель switch.

Оператор if..then
Пример:

```
var a, b, c, str;
a = 1;
b = 300;
if (a < b) { c = a; }
else { c = b; }
str = "Минимальное значение из двух чисел равно " + c;
document.write(str);
```

Минимальное значение из двух чисел равно 1

Условная операция (оператор ?)

Оператор ? является сокращенной формой оператора if..then. Синтаксис его следующий:

Условие ? действие 1 : действие 2.

Пример:

```
var a, b, c, str;
a = 80;
b = 300;
//Оператор ? является сокращенной формой условного оператора if.. else
a > b ? c = a : c = b;
str = "Максимальное из двух чисел: " + c;
document.write(str);
```

Максимальное из двух чисел: 300

Переключатель switch

Этот оператор удобно применять в случае нескольких однородных повторяющихся условий.

Пример:

```
var DayNumber, str;
DayNumber = 5;

switch (DayNumber) {
   case 1:        str = "Отдых от рабочего дня";          break
   case 2:        str = "Поготовка к рабочему дню»;       break
   case 3:        str = "Рабочий день»;                   break
   case 4:        str = "Отдых от рабочего дня";          break
   case 5:        str = "Подготовка к выходному дню";     break
   case 6:        str = "Первый выходной день«;           break
   case 7:        str = "Второй выходной день";           break
}
document.write(str);
```

Подготовка к выходному дню

Операторы цикла

Операторы цикла выполняют некоторую последовательность действий до тех пор, пока истинно (ложно) некое условие. В JavaScript предусмотрены три оператора цикла: for, while и do-while.

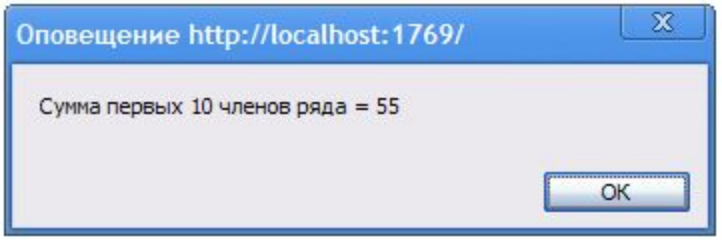
Оператор for

Пример:

В предлагаемом примере мы найдем сумму чисел от нуля до десяти.

```
var i,n, sum,str;
sum = 0;
n = 10;

for (i = 1; i <= n; i++) {
   sum += i;
}
str = "Сумма первых " + n + " членов ряда = " + sum;
alert(str);
```



Оператор while

Синтаксис этого оператора следующий:
while (условие) {
код
}
В предлагаемом примере решается предыдущая задача. Цикл while работает до тех пор, пока итерационная переменная i не станет равна десяти.

```
var i,n, sum,str;
sum = 0;
n = 10;
i = 1;
while(i<=n)
{
  sum += i;
  i++;
}
str = "Сумма первых " + n + " членов ряда = " + sum:
document.write(str);
```

Сумма первых 10 членов ряда = 55

Оператор do-while

Оператор do-while (делай до тех пор, пока) представляет собой конструкцию из двух операторов, работающих совместно. Синтаксис этой конструкции следующий:

do {
код
}
while (условие)

В приведенном ниже примере показано использование цикла do … while для нахождения факториала.

```
var n, i, z, str;
n = 5;
z = 1;
if (n > 1) {
   i = 2;
   do {
      z *= i;
      i++;
   }
   while (i <= n);
}
str = "Факториал " + n + " = " + z;
document.write(str);
```

Факториал 5 = 120