Лекция 2. Типы данных и операторы

Ст. преподаватель Еремеев А.А.
YeremeevAA@mpei.ru

Встроенные структуры и типы данных

- литералы;
- переменные;
- массивы;
- функции;
- объекты.

Лекция №3

Литералы

- Литералом называют данные, которые используются в программе непосредственно.
- При этом под данными понимаются числа или строки текста.
- Все они рассматриваются в JavaScript как элементарные *типы данных*.

Примеры литералов

- числовой литерал: 10
- числовой литерал: 2.310
- числовой литерал: 2.3e+2
- строковый литерал: 'Это строковый литерал'
- строковый литерал: "Это строковый литерал"

Использование литералов

- в *операциях присваивания* значений переменным :

```
var a=10;
var str = 'Строка';
```

- или в операциях сравнения:

```
if(x=='test') alert(x);
```

Кавычки в строковых литералах

- Если в строковом литерале требуется использовать одинарную кавычку, то сам *литерал* можно заключить в двойные кавычки: "It's cool!". Верно и обратное.
- Если необходимо использовать оба типа, то:

<u>команда:</u>

document.write("It\'s good to say \"Hello\" to someone!");

<u>выдаст:</u>

It's good to say "Hello" to someone!

Строковые объекты

- создаются конструктором:

  var s = new String().

Отличается от строковых литералов (последовательностей символов, заключенных в кавычки)!

Для этого объекта существует много методов (лекция №3).

При применении к строчным литералам методов строчных объектов происходит преобразование первых в последние.

Переменные

- ***Переменная*** - это область памяти, имеющая свое имя и хранящая некоторые данные. Переменные в JavaScript объявляются с помощью оператора var, при этом можно давать или не давать им начальные значения:

```
var k;
var h='Привет!';
```

- При объявлении нескольких переменных в одном операторе var, переменные записываются через запятую.

Тип переменной

- определяется по присвоенному ей значению.
- Язык JavaScript - слабо типизирован:

```
var i=5;          alert(typeof(i));
i= new Array();   alert(typeof(i));
i= 3.14;          alert(typeof(i));
i= 'Привет!';     alert(typeof(i));
i= window.open(); alert(typeof(i));
```

Глобальные и локальные переменные

- Переменная, объявленная оператором *var* вне функций, является ***глобальной*** - она "видна" всюду в скрипте.
- Переменная, объявленная оператором var внутри какой-либо функции, является ***локальной*** - она "видна" только в пределах этой функции.

Пример 1

```
function f()
 { var k=5; }

f();
 alert(k);
```

Пример 2

```
var k=7;

function f()
{ var k=5; }

f();
alert(k);
```

Объявление переменных

- Объявлять переменные можно и без оператора var:

  for(i=0; i<8; i++) { ... }

НО!

Опускать var не рекомендуется

- 1) нарушается ясность кода: если написано i=5, то непонятно, вводится ли здесь новая переменная или меняется значение старой.
- 2) внутри функции объявление переменной без оператора var делает переменную глобальной (а не локальной, как можно было бы предположить), и значит, ее значение могут "видеть" и менять другие функции или операторы вне этой функции.

Пример

```
function f()
 { var i=5; k=7; }

f();
 alert(k);
```

Массивы

- 1) встроенные (коллекции *document.links[], document.images[]* и т.д.) (лекция №3);
- 2) определяемые пользователем.

- Для массивов определено несколько методов: *join*(), *reverse*(), *sort*() и другие, а также свойство length, которое позволяет получить число элементов массива.

Определение массива

- Для определения массива пользователя существует специальный *конструктор* Array.
- Если ему передается единственный *аргумент*, причем целое неотрицательное число, то создается незаполненный *массив* соответствующей длины. Если же передается один *аргумент*, не являющийся числом, либо более одного аргумента, то создается *массив*, заполненный этими элементами.

Примеры

```
a = new Array();
// пустой массив (длины 0)
b = new Array(10);
// массив длины 10
c = new Array(10,'Привет');
// массив из двух элементов: числа и строки
d = [5, 'Тест', 2.71828, 'Число e'];
// краткий способ создать массив из 4 элементов
```

Элементы массива нумеруются с нуля!

Метод join() и split()

- Метод *join*() позволяет объединить элементы массива в одну строку.
- Он является обратным к методу *split*(), который разрезает объект типа String на куски и составляет из них массив.

Пример. Преобразования локального URL в глобальный

localURL = "file:///C:/portal/internet/js/2/2.html"
b = localURL.split(':/')

- Получаем массив:

  b[0] = "file";
  b[1] = "//C";
  b[2] = “portal/internet/js/2/2.html";

- Заменяем 0-й и 1-й элементы на требуемые:

  b[0] = "http:";
  b[1] = "/www.mpei.ru";
  globalURL = b.join("/").

Значение globalURL будет равно: http://www.mpei.ru/portal/internet/js/2/2.html

Метод reverse()

- Метод *reverse*() применяется для изменения порядка элементов массива на противоположный.

Пример

- Пусть существует массив a

a = new Array('мать','видит','дочь');

- Упорядочим его в обратном порядке, вызвав метод a.*reverse*(). Тогда новый массив a будет содержать:

a[0]='дочь';
a[1]='видит';
a[2]='мать';

Метод sort()

- Метод *sort*() интерпретирует элементы массива как **строковые литералы** и сортирует массив в **алфавитном** (т.н. лексикографическом) порядке.

- Применим a.*sort*() к предыдущему примеру, получим:

a[0]='видит';
a[1]='дочь';
a[2]='мать';

Сортировка чисел

Например, согласно алфавитному порядку 40 идет раньше, чем 5.

У метода *sort*() есть необязательный аргумент, являющийся именем функции, согласно которой требуется отсортировать массив: a.*sort*(myfunction).

Требования к функции:

- у нее должно быть ровно два аргумента;
- функция должна возвращать число;
- если первый *аргумент функции* должен считаться меньшим (большим, равным) чем второй аргумент, то функция должна возвратить отрицательное (положительное, ноль) значение

Пример функции

```
function compar(a,b)
{
 if(a < b)  return -1;
 if(a > b)  return  1;
 if(a == b) return  0;
}
```

```
b = new Array(10,6,300,25,18);
document.write("Алфавитный порядок:<BR>");
document.write(b.sort());
document.write("<BR>Числовой порядок:<BR>");
document.write(b.sort(compar));
```

Операторы языка

- {...}
- if ... else ...
- ()?
- while
- for
- break
- *continue*
- return

{...}

- *Фигурные скобки* определяют *составной оператор* JavaScript - ***блок*** . Основное назначение блока - определение *тела цикла*, тела *условного оператора* или функции.

if … else …

Условный оператор применяется для *ветвления* программы по некоторому логическому условию. Есть два варианта синтаксиса:

- *if (логическое_выражение) оператор;*
- *if (логическое_выражение) оператор_1; else оператор_2;*

- *Логическое выражение* - это выражение, которое принимает значение true или false.

()?

Этот оператор, называемый *условным выражением*, выдает одно из двух значений в зависимости от выполнения некоторого условия. Синтаксис:

- *(логическое_выражение)? значение_1 : значение_2*

Равносильный код программы

- *TheFinalMessage = (k>5)? 'Готово!' : 'Подождите...';*

- *if(k>5) TheFinalMessage = 'Готово!';*
- *else TheFinalMessage = 'Подождите...';*

while

- Оператор while задает цикл. Определяется в общем случае следующим образом:

  while (условие_продолжения_цикла) тело_цикла;

- Тело цикла может быть как простым, так и составным оператором. Составной оператор заключается в фигурные скобки.

- Условие_продолжения_цикла является логическим выражением.

- Тело исполняется до тех пор, пока верно логическое условие.

for

- Оператор for - это еще один *оператор цикла*. В общем случае он имеет вид:

- *for (инициализация_переменных_цикла;*
- *условие_продолжения_цикла;*
- *модификация_переменных_цикла) тело_цикла;*

Пример

```
document.write('Кубы чисел от 1 до 100:');
for (n=1; n<=100;n++)
document.write('<BR>'+n+'<sup>3</sup> = '+
Math.pow(n,3));
```

Math - встроенный объект, предоставляющий многочисленные математические константы и функции, а *Math*.pow(n,m) вычисляет степенную функцию n^m.

break

- Оператор break позволяет досрочно покинуть *тело цикла*.

```
document.write('Кубы чисел, меньшие 5000:');
for (n=1; n<=100; n++)
{
 s=Math.pow(n,3);
 if(s>5000) break;
 document.write('<BR>'+n+'<sup>3</sup> = '+s);
}
```

continue

- Оператор *continue* позволяет перейти к следующей итерации цикла, пропустив выполнение всех нижестоящих операторов в теле цикла.

```
document.write('Кубы чисел от 1 до 100, большие 10 000:');
for (n=1; n<=100; n++)
{
 s=Math.pow(n,3);
 if(s <= 10000) continue;
 document.write('<BR>'+n+'<sup>3</sup> = '+s);
}
```

return

- Оператор return используют для возврата значения из *функции* или обработчика события.

```
function sign(n)
 {
if (n>0) return 1;
if (n<0) return -1;
 return 0;
}
alert( sign(-3) );
```

Отмена действия по умолчанию

- При использовании в обработчиках событий оператор return позволяет отменить или не отменять действие по умолчанию, которое совершает браузер при возникновении данного события.

```
<FORM ACTION="newpage.html"
METHOD=post>
<INPUT TYPE=submit VALUE="Отправить?"
onClick="alert('Не отправим!');return false;">
</FORM>
```