Javascript

CS174

Chris Pollett

Sep 19, 2007.

Outline

- General Syntax
- Primitives
- I/O
- Control Statements
- Objects
- Arrays

General Overview

- Many details on Javascript can be found at:

  http://www.webreference.com/javascript/reference/core_ref/contents.html
- Javascripts can be either directly or indirectly embedded in a document.
  - Directly

    <script type="text/javascript">

    <!-- hide from old browsers

    -- code --

    // -->

    </script>
  - Indirectly

    <script type="text/javascript" src="myscript.js" />
- Identifiers (i.e., names) must begin with a letter, an underscore, or a $ and subsequent characters may be one of these three or numbers.
- Javascript has 25 reserved words: break, case, catch, continue, default, delete, do, else, finally, for, function, if, in, instanceof, new, return, switch, this, throw, try, typeof, var, void, while, with.
- Javascript has additional keywords which are reserved for future use, as well as predefined words such as alert, open, java, and self.
- Javascript supports /*…*/ and // for comments.
- Semicolons to end lines in Javascript are optional. Beware:

  return

  x; // has the effect of just return!

Example Javascript and HTML document

```
<html>
<head><title>test</title>
<meta name="description" value="this example illustrates how Javascripts are executed both when the
      document is loaded and on the occurrence of events" />
<script type="text/javascript" ><!--
  function sayHello()
  {
                alert("hi there");
  }
//-->
</script>
</head><body><form><input type="button" value="test" onClick="return sayHello();" /><!-- responds to
      events --></form>
<script type="text/javascript" ><!--
for( i = 0; i<100; i++)
{
      document.writeln("<p>hi"+i+"</p>");
} // run when document loads
-->
</script></body>
</html>
```

Primitives

- Javascript has 5 primitive types: Number, String, Boolean, Undefined and Null.
- Javascript has predefined objects corresponding to Number, String and Boolean. Each of these is wrappers for a value of the corresponding primitive type.
- Javascript will do type coercion between objects of type String and Number.
- A number literal can be either a integer or a float. You can use hex for integers. If it is a float it can use scientific notation:

  7, 0xff, 0XFF, 7.2, .73, -.23, 7E2, 7e2, 7.2e-2, etc.
- A string literal can be delimited either by a single or double quote. “Hi there”, ‘hi there’, ‘””, “\n”, ‘\’\\’, etc.
- There are two Boolean literals: true or false.
- The only Null literal is null which can be coerced to false as a Boolean and 0 as an Number.
- The only literal of type Undefined is undefined. It can be coerced to false as a Boolean and NaN as a Number.

Variables

- The Javascript interpreter determines the type of a variable as needed by the circumstance.
- Variables can be declared either by assigning it a value or by explicitly declaring it:

```
var myVariable, pi=3.14;
// explicit declarations effect the scope of the
   variable
```

Numeric Operators and Objects

- Numeric operators in Javascript are similar to Java or C: +, -, *, /, %, ++, --, etc.
- The Math object has many useful built-in methods and properties for Number objects. For example, Math.sin(x), Math.PI, Math.random, Math.abs, etc.
- The Number object has a useful collection of Number properties. Number.MIN_VALUE, Number.MAX_VALUE, Number.NaN, etc. It also has the toString method. For example, x=10; y= x.toString(); z=x.toString(2); //binary representation

Strings and Type Conversion

- The + sign is used to concatenate strings.

  first = “hello”

  second = first + “bye” // “hellobye”

- The String object also has useful methods such as: charAt, indexOf, substring, toLowerCase, toUpperCase, etc. A String object also always has a length property
- When the value of one type is used in a situation where another type is required, Javascript tries to implicitly coerce the type into the required one.

  Ex “August” + 1977 // “August1977”

  1977 + “August” // “1977August”

  7*“3” =21

- One can also explicitly, do type conversion.

var str_value = String(value); // Might want to use toString

var number = Number(aString); // Might want to use parseInt or parseFloat

typeof, Assignments, and the Date Object

- The operators

  typeof x //and

  typeof(x) /* returns either “boolean”, “string”, “number” if x is of primitive type, it returns “object” if x is null or an object; and it returns “undefined” if x is not defined*/

- Assignments in Javascript are similar to C or Java:

  a++; a+=2; a--; a-=2; a = b +57;

- The Date object is useful for getting information about the current date and time:

  var today = new Date();

- Date supports methods: toLocaleString, getDate, getMonth, getDay,getFullYear, getTime, getHours, getMinutes, getSeconds, getMilliseconds

I/O

- The default output target of a Javascript I/O is the browser window.
- Javascript models an XHTML document as a Document object.
- The window in which this document is displayed is a Window object.
- Window has two properties document and window. Here document refers to the current Document object.
- Document has several useful methods for I/O. write, writeln.
- In addition to this method of I/O one can also create dialogs with the alert, prompt, and confirm methods.

Ex:

name = prompt(“What is your name”, “John Smith”)

Control Statements

- Javascript supports the relational operators: ==, !=, <, >, <=, >=, ===, !==
- The last two operators disallow conversion of either operand. So “3” === 3 evaluate to false.
- Javascript also supports the operators &&, ||, !
- Selection statements in Javascript are like in C/Java: if(a >b){} else{}
- Javascript supports switch/case as in C/Java
- Javascript supports while, for and do-while loops

Objects

- Objects can be created with an initial declaration like:

  var my_object = new Object();

- This object would initially have no properties. To delete an object use: delete my_object;
- An assignment like:

  my_object.make= “V6” /* would then give a property make a value. */

  //can access as

  p = my_object[“make”]

  q = my_object.make

- You can also nest objects this:

  my_object.subObject = new Object();

- You can loop over properties using:

for(var prop in my_object){…}

Arrays

- Arrays can be created with the syntax:

```
var myArr = new Array(1, 2, “hello”)
var myArr = new Array(100);
var myArr = [1,2,3];
//to access
myArr[0]
//to determine length
myArr.length
```