Notes on JavaScript Learning

Tuesday, May 06, 2014 09:38

1. JavaScript is a dynamic language
   a. Eval: eval("var a = 1;")
   b. First-calss functions
   c. Object alteration at runtime
   d. Closures

2. A string literal is two quotes (' ') or double quotes (" ") encapsulating zero or more characters.

3. About array in JavaScript
   a. var missingvalue = [1, 2, 3, , 5]; a missing value is used.
   b. var only3items = ['one', 'two', 'three',];last value in the Array literal is expressed this way, it is dropped from the Array

4. Object Literals: var kitty = {whiskers: 20, name : "Comet", age: 2};

5. Blocks in JavaScript do not inherently give variables scope:

```
if (!a) {
   var a = "yay";
}
document.write(a);    // "yay"
```

   But

```
function myFunct() {
   var g = "yay";
}

myFunct();

document.write(g);    // ReferenceError: g is not defined
```

6. the null type indicates an empty value. The difference to undefined is that when a variable is undefined it has been created but doesn't have a value, and when a variable is null it has been set to have an empty value.

7. document.write(**typeof** myIdentifier);

8. JavaScript does not have a Char (character) type

9. === (Strict Equality): Compares the two operands to see if they contain the same values. No type conversion is performed first. Eg. If (a===b) {…}. (Same as !===)

```
5 == "5" // Evaluates to true!
5 === "5" // Evaluates to false!
5 === (6-1)    // Evaluates to true
```

10. The Comma (,) Operator:
    `alpha=1,beta=2,gamma=3` In this example, the result of the entire compound expression is 3, but the other two expressions to the left would also be evaluated.

11. The new Operator

```
var Animal = function(thename) {
        this.fur=true;
        this.scales=false;
        this.name='Generic';
        if (thename) {
                this.name=thename;
        }
};

var cat = new Animal('cat');
var unknown_animal = new Animal();    // the default name is 'Generic'

document.write(cat.name); // "cat"
document.write(unknown_animal.name);    // "Generic"
```

12. The void Operator

```
<a href="javascript:void(document.body.style.backgroundColor='#00FF00');">Click here to make the page green</a>
```

13. *for each .. in* loop works the same way as *for .. in*, but instead of iterating over an objects property names, it iterates over the property values.

14. **these will be passed to the function either by value or by reference depending on if the variable is of a primitive or reference type.**

15. **if you attempt to overload a function, the most recent definition will be used.**

```
// Demonstrating the effect of attempted function overloading
function threeArguments(a, b, c) {
   return "We expect 3 arguments: " + a + ", " + b + ", " + c;
}

// Now we try to overload the function
function threeArguments(a,b,c,d) {
   return "Now we expect 4 arguments: " + a + ", " + b + ", " + c + ", " + d;
}

// Now we attempt to use the first version
document.write( threeArguments(1, 2, 3) ); // "We expect 4 arguments: 1, 2, 3,
undefined"
```

```
// output: We see that the reference to threeArguments has been overridden.
```

16. Inheritance in JavaScript
    a. Prototype-based Sub-classing

```
function Animal() {}
Animal.prototype = {
   group: "",
   gender: "",
   eat: function() {
      return "Yum, food! nom nom";
   },
   sleep: function() {
      return "zzzzzzzz..";
   }
}
```

```
function Bird() {}
Bird.prototype = new Animal();
Bird.prototype.fly = function() {
   return "flap flap flap!";
}
```

    b. Alternate Subclassing Approaches

```
function extend(subClass, superClass)
{
   // Create a new class that has an empty constructor
   // with the members of the superClass
   function inheritance() {};
   inheritance.prototype = superClass.prototype;

   // set prototype to new instance of superClass
   // without the constructor
   subClass.prototype = new inheritance();
   subClass.prototype.constructor = subClass;
   subClass.baseConstructor = superClass;

   // enable multiple inheritance
   if (superClass.__super__) {
      superClass.prototype.__super__ = superClass.__super__;
   }
   subClass.__super__ = superClass.prototype;
}


// usage
function Bird(gender) {
   this.gender = gender;
}

extend(Bird, Animal);
```

17. Browser Object Model (BOM)
    a. The **document** object: A structural object representation of the layout and content of the page with APIs that allow you to modify its contents.
    b. The **frames** collection: An array-like object of all the sub-frames in the current document.
    c. The **history** object: An object containing the browser session history, a list of all the pages visited in the current frame or window.
    d. The **location** object: Detailed information about the current URL of the frame or window.
    e. The **navigator** object: Information about the application running the current page or script.