**Modern JavaScript Development: Reflection in JavaScript**

Sometimes you want to find out what members an object exposes. There’s a pretty simple way to do this. A simple for-in statement.

|  |  |
| --- | --- |
| 1 | for(var member in obj){ alert(member); }; |

No really, that’s it. No “imports” or “using” statments. No complicated classes to memorize, no unwrapping or casting. Just for-in over an object. One caveat, this will show you ALL of the objects members and it doesn’t distinguish between members declared directly on the objects and members inherited through the objects prototype. What if you just want to find the methods on an object? I’ve created two helper functions that will return just the methods(functions) of any JavaScript object.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | #to return all methods including inherited  Object.prototype.getAllMethods = function(){  var memberArray = new Array();  for (var method in this) {  if (typeof this[method] == 'function') {  memberArray.push(method);  };  };  return memberArray;  };    #to return only its own methods  Object.prototype.getOwnMethods = function(){  var memberArray = new Array();  for (var method in this) {  if (typeof this[method] == 'function' && this.hasOwnProperty(method)) {  memberArray.push(method);  };  };  return memberArray;  }; |

In the getOwnMethods function we use the hasOwnProperty function to discriminate between methods declared in the object and methods inherited through the prototype.