CMSI 585
PROGRAMMING LANGUAGES (GRADUATE LEVEL)
Fall 2006

Assignment 1107

This assignment seeks to get your hands dirty with types. As always, there is much more material available than can be directly addressed by any single assignment, so make sure to do the reading as well.

Not for Submission

1. Read Scott Chapter 7.
2. Install and learn the JSUnit unit testing framework for JavaScript. The course Web site holds the link to the framework, while the JavaScript sample code given at the beginning of the semester uses JSUnit.

For Submission

Commit the programming tasks under the specified directories with the given tag, and submit your answers to the assigned textbook exercises and other questions on hardcopy. If you do the extra credit work, submit that on hardcopy as well. The extra credit is cumulative, not selective — the "extra extra" credit only counts if you've already done the "single extra" credit exercise.

1. Answer Scott Exercise 7.2.
2. Write a set of JavaScript functions, collectively gathered in *thingAssociator.js*, that uses the language's property-based approach to associate one "thing" with another. These functions are:
   - *associate(map, key, value):* Takes the JavaScript object passed as *map* and sets its *key* property to *value*. This is essentially a wrapper for *map[key] = value*.
   - *getDirectAssociation(map, key):* Takes the object *map* and returns the value that *map* currently associates with *key*. It returns the JavaScript *undefined* constant if *key* has not been associated with any value.
   - *getAssociationClosure(map, key):* Takes the object *map* and recursively follows the *key* property within *map* until it reaches a value that is not itself a key in the map. This is the value that the function then returns. For example, if *map* associates "Hello" with "World" and "World" with 10, then *getAssociationClosure(map, "Hello")* returns 10.
   - *getAllAssociations(map):* Returns a JavaScript array of the *map* object's properties, numerically indexed starting at zero. Each element of this array should be a JavaScript object with four properties: *key*, which holds an association's key property; *keyType*, which holds the type of that property; *value*, which holds the value associated with that key in *map*; and *valueType*, which holds that value's type.
3. Demonstrate the functionality of these functions through a series of unit tests using the JSUnit framework. Call this JSUnit test page *yourName_thingAssociatorTest.html*.

   Commit all code under */homework/cmsi585/thingAssociator/js*. Tag these files with *hw-1107*.
4. Answer Scott Exercise 7.19 (suggestion: write a small program that helps provide the answer).
5. Answer Scott Exercise 7.24.
6. Answer Scott Exercise 7.28.
7. Answer Scott Exploration 7.38; consider all three approaches here: (1) specified representation in the language, (2) semantic behavior with guaranteed implementation-specific representation, and (3) semantic behavior with varying implementation-specific representation.

Extra Credit

1. Answer Scott Exercise 7.5.
2. Answer Scott Exercise 7.17.

Extra Extra Credit

Take a non-primitive type (array, record, list, set, etc.) from any language in our "big 6" (C, C++, Java, JavaScript, ML, and Perl) and compare the semantics of this type with the semantics of a similarly named type from a language that is not in our "big 6" (e.g., compare JavaScript's concept of an array with Ada's concept of an array).