/\*Getting, setting, and updating an object's properties using dot

notation or bracket notation

We can get, set, or update an object's properties using either dot notation or bracket

notation.

\*/

// Create a cody Object() object.

var cody = new Object();

// Setting properties.

cody.living = true;

cody.age = 33;

cody.gender = 'male';

cody.getGender = function () { return cody.gender; };

// Getting properties.

console.log(

cody.living,

cody.age,

cody.gender,

cody.getGender()

); // Logs 'true 33 male male'.

// Updating properties, exactly like setting.

cody.living = false;

cody.age = 99;

cody.gender = 'female';

cody.getGender = function () { return 'Gender = ' + cody.gender; };

console.log(cody);

/\*Dot notation is the most common notation for getting, setting, or updating an object's

properties.

Bracket notation, unless required, is not as commonly used. In the following sample, I

replace the dot notation used in the previous sample with bracket notation. The object

name is followed by an opening bracket, the property name (in quotes), and then a

closing bracket:\*/

// Creating a cody Object() object.

var cody = new Object();

// Setting properties.

cody['living'] = true;

cody['age'] = 33;

cody['gender'] = 'male';

cody['getGender'] = function () { return cody.gender; };

// Getting properties.

console.log(

cody['living'],

cody['age'],

cody['gender'],

cody['getGender']() // Just slap the function invocation on the end!

); // Logs 'true 33 male male'.

// Updating properties, very similar to setting.

cody['living'] = false;

cody['age'] = 99;

cody['gender'] = 'female';

cody['getGender'] = function () { return 'Gender = ' + cody.gender; };

console.log(cody);

/\*

In the next sample, I demonstrate the advantage of bracket notation over dot

notation by using it to access the property foobar. I do this using two variables that,

when joined, produce the string version of the property key contained in foobarObject.\*/

var foobarObject = { foobar: 'Foobar is code for no code' };

var string1 = 'foo';

var string2 = 'bar';

console.log(foobarObject[string1 + string2]); // Let's see dot notation do this!

/\*

Notes

Because objects can contain other objects, cody.object.object.object.object or

cody['object']['object']['object']['object'] can be seen at times. This is

called object chaining. The encapsulation of objects can go on indefinitely.

Objects are mutable in JavaScript, meaning that getting, setting, or updating them can

be performed on most objects at any time. By using the bracket notation (e.g.,

cody['age']), you can mimic associative arrays found in other languages.

If a property inside an object is a method, all you have to do is use the () operators

(e.g., cody.getGender()) to invoke the property method.\*/