/* CementTruck has 10 wheels and weighs 12,000 pounds */
CementTruck.prototype.wheelCount = 10;
CementTruck.prototype.curbWeightInPounds = 12000;

/* CementTruck refuels with diesel fuel */
CementTruck.prototype.refuel = function() {
    return "Refueling CementTruck with diesel fuel";
}

/* Function for performing the main tasks of a SportsCar */
CementTruck.prototype.mainTasks = function() {
    return "Arrive at construction site, extend boom, deliver cement";
}

Listing 5-3 details a small Web page that demonstrates the inheritance mechanism of the three objects. The page simply contains three buttons, with each button creating one type of object (Vehicle, SportsCar, or CementTruck) and passing the object to the describe function. The describe function is responsible for displaying the values of each object's properties and the return values of the object's functions. Notice how the describe method doesn't know whether the object it's describing is a Vehicle, SportsCar, or CementTruck—it just assumes that the object has the appropriate properties and functions and lets the object return its own values.

Listing 5-3. inheritanceViaPrototype.html

```html
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>JavaScript Inheritance via Prototype</title>
<script type="text/javascript" src="inheritanceViaPrototype.js"></script>
</head>
<body>

function describe(vehicle) {
    var description = ""
    description = description + "Number of wheels: " + vehicle.wheelCount;
    description = description + "\n\nCurb Weight: " + vehicle.curbWeightInPounds;
    description = description + "\n\nRefueling Method: " + vehicle.refuel();
    description = description + "\n\nMain Tasks: " + vehicle.mainTasks();
    alert(description);
}

function createVehicle() {
    var vehicle = new Vehicle();
    describe(vehicle);
}
```