CHAPTER 2

The Seven Habits of Highly Successful JavaScript Developers

In this chapter, we'll continue the discussion began in Chapter 1 and look in more detail at the art of making JavaScript a first-class language. We'll look at object-oriented techniques, as well as some of the latest buzzwords such as *unobtrusive JavaScript* and *graceful degradation*. We'll talk about how to make your web applications accessible, even with JavaScript involved (no easy task!). We'll look at error-handling and debugging techniques, since things sometimes (OK, *frequently*!) don't go right. We'll also take a look at some of the tools available to you that will make working with JavaScript a much more pleasant experience. Lastly, we'll do a quick survey of some of the most popular JavaScript libraries out there today, and discuss why you really, honestly, and truly want to be using them! That's a lot to cover, so let's get to it!

JavaScript: reach divinity in the eyes of your users by doing it right!

More on Object-Oriented JavaScript

When many JavaScript programmers start out, they often do not even realize that the language offers some object orientation. Indeed, JavaScript does not require the use of objects at all.[1]

There is more than one way to skin a cat, and likewise, there is more than one way to create objects in JavaScript.

Simple Object Creation

Perhaps the easiest way to create an object is to start with a new `Object`, and then add to it. To create a new `Object`, you simply do this:

```
var newObject = new Object();
```

The variable `newObject` now points to an instance of `Object`, which is the base class of all objects in JavaScript. To add elements to it, say a property named `firstName`, all you need to do is this:

```
newObject.firstName = "frank";
```

From that point on in the code, `newObject.firstName` will have the value `"frank"`, unless it's changed later. You can add functions just as easily:

```
newObject.sayName = function() {
 alert(this.firstName);
}
```

A call to `newObject.sayName()` now results in an alert message showing "frank." Unlike most full-blown object-oriented languages, in JavaScript, you do not necessarily need to create a class, or blueprint, for an object instance. You can instead create it on the fly, as shown here. You can do this throughout the life of the object. On a web page, that means that you can add properties and methods to the object at any time.

JavaScript actually implements all objects as nothing but associative arrays. It then puts a façade over that array to make the syntax look more like Java, or C++, using dot notation. To emphasize this point, note that you could retrieve the value of the `firstName` field of `newObject` like so:

```
var theFirstName = newObject["firstName"];
```

Likewise, the `sayName()` function could be called like so:

```
newObject["sayName"]();
```

This simple fact can be the basis for a lot of power. For instance, what if you wanted to call a method of an object based on some bit of logic? Well, you can do this:

1. Well, implicitly it does, since you use built-in objects in many cases, but your code itself doesn't have to be object-oriented.