**Part 10. Code Organization**

**Overview**

When you move beyond adding simple enhancements to your website with jQuery and start developing full-blown client-side applications, you need to consider how to organize your code. In this chapter, we'll take a look at various code organization patterns you can use in your jQuery application and explore the RequireJS dependency management and build system.

**Key Concepts**

Before we jump into code organization patterns, it's important to understand some concepts that are common to all good code organization patterns.

* Your code should be divided into units of functionality — modules, services, etc. Avoid the temptation to have all of your code in one huge $(document).ready() block. This concept, loosely, is known as encapsulation.
* Don't repeat yourself. Identify similarities among pieces of functionality, and use inheritance techniques to avoid repetitive code.
* Despite jQuery's DOM-centric nature, JavaScript applications are not all about the DOM. Remember that not all pieces of functionality need to — or should — have a DOM representation.
* Units of functionality should be [loosely coupled](http://en.wikipedia.org/wiki/Loose_coupling) — a unit of functionality should be able to exist on its own, and communication between units should be handled via a messaging system such as custom events or pub/sub. Stay away from direct communication between units of functionality whenever possible.

The concept of loose coupling can be especially troublesome to developers making their first foray into complex applications, so be mindful of this as you're getting started.