There’s a lot to know about JavaScript and the APIs of HTML5. So much so it’s not really possible to go over everything. However, there’s a lot we can cover that’s important to mobile development. Many of the things I talk about can be achieved by using a JavaScript framework such as jQuery, which is introduced at the end of the chapter, but first you need to know the basics of what’s under the hood so you can deviate from a framework or fix problems when they come up.

The first thing to look at is the basics of JavaScript: things such as how to include and reference external JavaScript files as well as some basics for creating JavaScript objects and methods. Next I go into some of the JavaScript events you’ll use over and over.

Finally I talk about how to roll your own AJAX objects and a little about JSON (JavaScript Object Notation) for retrieving remote data. We use PHP to handle the remote data in Grandview Ave, but you could use HTML, Ruby, JSON, Python, .NET, anything really. The important thing is that we’re getting remote data and bringing them in via AJAX.

Before getting into the APIs and items above, I will stray a little bit into the history of JavaScript because the structure of the language is vital to everything we want to do and it’s especially important when comparing JavaScript to a server-side language.

About JavaScript

JavaScript is a unique language. I will give you a little bit of an overview of JavaScript because I think it goes a long way in understanding the way frameworks and things work in JavaScript as compared to Objective-C, C, C++, Java, Ruby, PHP, or .C# languages. When frameworks break or a problem comes up, many web developers get stuck because they ignore much of this information.

JavaScript was originally developed by Brendan Eich of Netscape in 1995, and is an implementation of the ECMAScript language standard. Other dialects of ECMAScript include Jscript and ActionScript. The current edition of ECMAScript is 5 and the current JavaScript version is 1.8.5.

JavaScript is a scripting language. Scripting languages control one or more applications. There are many types of scripting languages. Some are for shells such as bash or batch files. Other languages including Perl started out as scripting languages but became more powerful. But because scripting languages by definition control one or more applications, JavaScript controls the web browser application.

JavaScript is dynamic as opposed to static. Dynamic languages are a class of programming languages that execute at runtime and can change their composition while running. This is more than just modifying the state of certain variables: dynamic languages can add code and extend objects, all during the course of program execution making the distinction between code and data difficult.

JavaScript is weakly typed as opposed to strongly typed. Weakly typed languages essentially just allow you to not specify the type for a variable. So the variable foo can be set to an integer, a string, a double, or an array and can change during the course of program execution.
JavaScript is object-oriented. This is a style of programming performed with objects or complex data structures that are composed of variables and methods. Those variables and methods provide an object with certain behavior.

JavaScript is prototype-based. Prototype-based means that you create objects by cloning other objects rather than instantiating objects from a class (template).

JavaScript is a functional language. A functional language is one that treats computer programs similar to mathematical functions. This is a 10-dollar word for a 10-cent concept. Basically it just means that if you pass a variable to a function, it returns another variable. All states are contained within the function’s input variables.

JavaScript has first-class functions. A first-class function basically means you can pass in a function to another function. This can best be seen in the example below.

Companion Site Reference

Example 4-1: Follow the link below to run this example on the companion site.


// normal function
function foo(a, b) { return a + b; }
// normal function as variable
var foo2 = function(a,b) { return a+b; }
// function as parameter to another function
function foo3(foo2,a,b) {
  return foo2(a,b);
}
// outputs 3,3,3
alert(foo(1,2)+"","+foo2(1,2)+","+foo3(foo2,1,2));

Based on your style of programming and background you may prefer to use an object-oriented approach or a functional approach. This is known as multiparadigm. Some JavaScript libraries might be entirely function-based, whereas others will be more object-oriented. Either way there’s a lot to learn about the language; next I talk about a few useful parts.

Using JavaScript

There’s a lot to talk about in JavaScript. I recommend getting one of the fine Apress books on just JavaScript by itself to learn all the ins and outs of the language. But in lieu of that I touch on some of the most useful parts of the language, specifically the parts we need for creating our mobile web app.

Externalizing JavaScript

Just as in Chapter 3 when we externalized, minified, and gzipped CSS, we can do the same with JavaScript. Inline JavaScript would look like this:

```html
<script type="text/javascript">
function foo() {...}
</script>
```