1

Your Map, Your World

> "When I have the map, I will be free, and the world will be different, because I have understanding." **—*Time Bandits*, 1981**

This chapter discusses

- What map APIs are
- What map applications can do
- The core principles of map technology, including map projection and scaling
- Why Yahoo! offers three types of map APIs

What Is a Map API?

As an avid web user, you probably use map applications all the time, even daily. Map applications, like the Yahoo! Maps beta, enable millions of users to find locations by address, get driving directions and search for businesses based on location. These useful applications have become some of the most popular and dependable sets of functionality on the web.

As a web developer, you have probably worked with APIs. API stands for *application programming interface.* It is an interface for programming an application, a collection of functions to program against. An API, like the Yahoo! Maps API, is designed to expose key functionality as functions that, when simply added to your application, enable you to access and control the features of the application. In the coming chapters you will see that many API calls, like `setCenterByAddress()`, `setZoomLevel()` and `setMapViewType()`, are named after their respective functionalities.

A map API enables you to develop with much of the useful and dynamic functionality of map applications. When it is combined with your existing application, your users will benefit from unparalleled, very useful features.

The Four Types of Map APIs

Yahoo! released version 3 of its Maps in November 2005, and with it a new wave of map APIs. Your development world is truly about to expand as you learn about each API. There is an Ajax API and two Flash-based APIs, and each type or flavor of API features its own set of characteristics. The technology behind each API flavor leads to some unique benefits, and as a result, both the basic and advanced functionalities vary while offering many of the same core features. The most distinct thing about each API is the way it is implemented.

The Ajax API is known for its universal adoption and browser compatibility. This API was very skillfully built and has benefited from tremendous support. Ajax is spreading madly across the web and has been key in what has become known as the Web 2.0 revolution. Its ability to send and receive data without the browser being refreshed has changed the way map applications are used on the web.

The three Flash APIs, while implemented very differently, are all built on a similar code base. There is a JavaScript implementation, which implements almost identically to the Ajax API. When Flash is used as an optimal rendering engine, method calls to this API are made through JavaScript and then channeled into the Flash client.

The AS-Flash API requires you to develop with the Adobe (formerly Macromedia) Flash IDE. However, with this requirement comes the ability to control the appearance of the map in a way that no other API can offer. The Flash IDE enables you to work with the API instance on Flash's stage, as well as with the programmatic instantiation of the API through ActionScript 2.0.

Flex is a framework on top of Flash that allows extremely easy control of user interface (UI) elements with a language called Mxml. While Flex still requires some heavy use of ActionScript, it provides an environment that allows much more rapid deployment. A big strength of Flex is its ability to bind common control elements with dynamic data sources. With the Flex API, you can build what are considered more enterprise-level applications.

With either the AS-Flash or Flex APIs, you will find it very easy to integrate other UI elements alongside the map, or blend them within it. This creates a seamless and rich user experience, ensuring that your application will be well received by your users.

While each API has its own benefits, the Flash-based APIs offer some very special functionality. As you will see in the introduction to map features and tools in Chapter 2, "Developing with Yahoo! Maps," Flash offers several ways to create your map application.

Understanding Map Technology

To build a map application with Yahoo!'s APIs really requires very little, if any, understanding of map technology. The more details you understand, however, the better you'll be able to make your map application.

The first concept to grasp is that of map projection. Most people know the world is round, just like globes in elementary school classrooms. Yet online maps, as well as those we print for driving directions, are flat. Getting from the spherical presentation of Earth to a flat one online is called *projection*. If you were to merely peel off the detail of a globe and put it flat on paper, each piece would connect only at the equator. In other words, you could lay each piece of the globe side by side, but you would see gaps grow between the pieces up and down from the middle.

A projection model called *Mercator*, named after the sixteenth-century Flemish cartographer, solves this problem. This concept of map projection maintains consistent width between pieces of a globe. However, to fill in the gaps above and below the equator, Mercator stretches the north and south scale of the map. As you move farther away from the equator in either direction, the depiction of Earth on the map is gradually stretched. This provides a rectangular representation of a map, one that fills the window of a browser and the screen on your monitor. See Figure 1-1.

Figure 1-1

Another concept in mapping technology is *scaling*. Scaling simply enables you to relate the distance on a map to actual distance on the ground. Take a look at any map. You will probably see a pair of lines, usually on the bottom-left corner, showing the distances that represent miles and kilometers on the map. In dynamic web-based maps, the scale changes as you zoom in and out of the map.

Common Map Features and API Tools

Because many of the features in Yahoo! Maps are offered in the APIs, your users will likely be familiar with what they come across in your applications. Before building your application and implementing the map APIs, it is important to get an overview of the tools that make map applications work. These are many of the fundamentals you will build on in creating your map applications with Yahoo! Maps APIs.

Map and Satellite Imagery

At the core of the map application is the map itself. In most map applications, including Yahoo! Maps, map images are displayed with a combination of tiles. These tiles represent portions of the map cut up into fixed sizes and served to the application. When these tiles are combined, they form the viewable map area. To keep the application running efficiently, only those tiles in view, as well as some additional rows and columns of tiles, are downloaded as a buffer.

The Yahoo! Maps API offers three different view types and, with these, three different sets of image tiles. The default view, *map*, presents the standard map tiles. These tiles combine two elements. First is the labels layer, displaying street, landmark and area names. Landmark labels include parks, lakes, shopping malls, learning institutions and railroad tracks, to name a few. The second element is a layer of color-coded areas. An array of simple colors is used to indicate with exceptional accuracy the size and location of bodies of water, roads, commercial zones and recreational areas such as ballparks and marinas. Both elements are generated from data and keep the map tiles fairly small in file size.

The *satellite* view, perhaps the most fascinating element of web-based map applications, represents the layer made up of aerial images. These images, mostly taken from a satellite in space, give us a unique perspective on our world. The ideal satellite image will be taken with no clouds and with as little angle as possible. (In some major cities, like New York or Chicago, angles that cause shadows are inevitable due to the height of some buildings.) Satellite images are larger in file size than map images, as they feature high-resolution photography. Like the maps layer, satellite images are cut up into smaller pieces. This allows the map application to download the tiles progressively as opposed to downloading one big image.

The *hybrid* view is a combination of both the map and satellite layers. This provides a very effective feature. Street names and labels are displayed on top of satellite images, providing more detailed information and removing some of the obscurity satellite tiles have by themselves. While the satellite tiles in the hybrid view are the same, the map tiles are slightly different. The second element is removed, as described earlier, leaving just the label elements. This creates a transparent effect in the map layer, letting the satellite imagery show through with the street, landmark and area names on top.

In the map application, a set of buttons enables the user to toggle between map-view types. Working with the API, you can set the map-view type as a default parameter, allowing your users to see either the map, hybrid or satellite layers by default. There are several ways to introduce controls that allow toggling between layers as well. In addition to creating your own buttons with calls like `getMapViewTypes()`, you can also add a `satelliteControlWidget` that mimics the buttons in the Yahoo! Maps beta.

Zooming

Maps have zoom capability to allow the user to display different magnitudes of the same layer. With the satellite images, photographic layers are compiled at several different distances. The map's layers are rendered at these respective distances as well. The action of moving between these layers, moving your view of the map closer and farther away, is called *zooming*.

Past HTML- and JavaScript-based map applications included zooming. However, when zoom levels were changed, the application needed to completely refresh, loading the new layer. Yahoo! Maps APIs allow zooming that does not require the browser to refresh. The Flash APIs also feature a much more dazzling effect. As the user zooms from one distance to the next, the layers actually fade into one another, creating a very attractive transition effect.

There are 16 zoom levels offered in the APIs. Zoom levels are changed by the user clicking on the zoombar, a user interface element that usually features plus and minus buttons. The plus button zooms in and the minus button zooms out. In the Flash APIs, a slide mechanism is placed between the buttons of the zoombar. When the slider is selected and moved up or down, like to the scrollbar on the side of a web browser, the map will respond by zooming in and out accordingly.

The zoom level can be set by default when the API loads. By calling methods like `setZoomLevel()`, you can programmatically modify the zoom level of the API instance. Methods like `setCenterByAddressAndZoom()` enable you to move the map to a new destination as well as to prompt a zoom level change. In the Flash APIs, changing zoom levels by calling the API methods will induce the transition effects. See Figure 1-2.

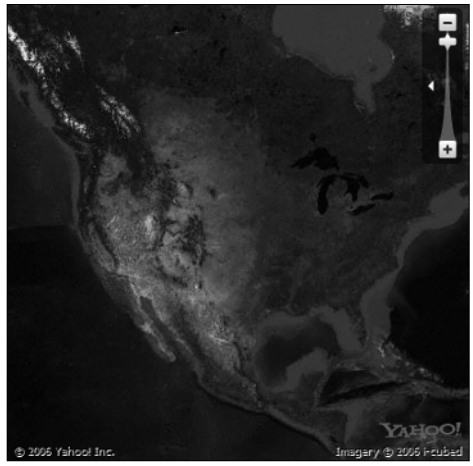

Figure 1-2

Panning

A key functionality of web-based map applications is the ability to pan. Commonly referred to as "making the map draggable," *panning* is the action of moving the view of the map in any direction. You may be surprised to know that this feature is overall fairly new to map applications. In fact, several map applications still do not have panning abilities. Traditionally, maps without panning require you to click on the direction in which you want to move the map. Once you choose the direction, the application must refresh to collect new tiles. Yahoo! Maps and the API feature a very well implemented ability to pan, and you can do it in several ways.

In film and television, panning is the technique of moving a camera to create a sweeping shot. The effect is called *panoramic*. Panning applied to maps creates a transition so the map within the API moves without the component container moving around on the screen. In the API, this is achieved by the masking capabilities of Flash.

The best way to pan is to click the map with your mouse and drag it around. Effectively, what you are doing is moving the map view layer, with all its tiles. Because the map view layer downloads tiles as it needs them, panning far enough in any direction will likely prompt the application to download and display additional tiles. A buffer exists to be displayed so the application has tiles to show as you pan beyond the initial downloaded layer.

A less discoverable, but very effective, way to pan over the map is with your arrow keys. Using your keyboard, the map will pan in the direction of the arrow key you press. Those users who like to use shortcut keys will find this feature very suitable. Like an airplane's throttle, each key actually moves the map in the opposite direction. Pressing the up-arrow and right-arrow keys simultaneously will move the map diagonally.

When working with the API, you add panning functionality as a tool with `addTool()`. One of the advantages to using the Flash APIs, as we will explore, is the support of a hand icon instead of a mouse cursor. This adds better discoverability to the panning feature, as users know what areas they can drag.

Geo-coding

A map's ability to plot an address is called *geo-coding*. By means of sophisticated backend technology capable of correlating countries, states, cities, ZIP codes, streets and street addresses, an address is turned into latitude and longitude. You may remember latitude and longitude (referred to in maps development as lat/lon) from elementary school. Lat/lon are the axes of a map.

Since most developers don't know lat/lon coordinates by heart, the map APIs offer built-in geo-coding. This means you can pass an address in the API's initial parameters, then that address will be converted by the server to lat/lon, and the API instance will display and load to that location. You can call a method like `setCenterByAddress()` to perform a similar function and change the address during runtime.

If you do know what your initial lat/lon is, you can set that for when the API loads. By using methods like `setCenterByLatLon()`, you can change the location the map centers on. A method called `getCenter()` returns a `LatLon` object representing the center point of the map. In the event that an address returns an error, the sophisticated API will likely return a `SuggestedAddress` or an array of them. This can be displayed to your users and help them narrow down address selection.

Markers and Custom Markers

Markers are locations plotted on a map at specific locations, indicated with icons, images and media. They are placed at lat/lon coordinates derived from the server based on an entered lat/lon or address. Acting as indicators, markers can represent several aspects of data being shown on maps.

Yahoo! Maps APIs offer several ways to display markers, using methods like `addMarkerByAddress` and `addMarkerByLatLon`. More impressive are the variety of marker types you can add. A `CustomPOIMarker` (POI meaning point of interest) can be added and features out-of-the-box interactivity when the user rolls over it. When the user engages this type of marker, a dynamic title will be shown. These markers can be defined with custom colors as well.

`CustomImageMarkers` let you define a marker with a JPEG or SWF file. However, with the Flash API you have an even better option. Using `CustomSWFMarker` not only lets you define a marker with a custom SWF, but also lets you call methods on that SWF with `callCustomMethod`. Basically, you can create your own Flash movie and serialize it as your marker. This feature will be explored much more in later examples found in Chapter 16, "AS-Flash API Advanced Features." For now, look at Figure 1-3.

Figure 1-3

Overlays

Adding an overlay to the map creates another layer that stays in sync with the map as it pans and zooms. An overlay can contain various elements, all displaying on a transparent layer above the map. When you add a set of markers to an overlay instead of to the map, you can control them collectively, using `hide()` and `show()`.

Packaged with the Flash API are two very powerful overlays, `LocalSearchOverlay` and `TrafficOverlay`. `LocalSearchOverlay` adds the ability to search Yahoo!'s powerful local database. Your users will be entering local search criteria and a method `search()`. Based on those criteria as well as the center lat/lon of the map, the API will take data returned from the server and add a set of markers representing several local findings. The markers within the overlay will display the business name and, when engaged by the user, will also display address, contact and rating information for that establishment.

The `TrafficOverlay` data add some very useful information technology to the API. When activated, this overlay displays special markers, based on the map's location, featuring traffic incidents such as accidents and closed roads.

The possibilities of the overlay are limitless. As you will learn, the results from many data sources can be neatly and effectively contained within an overlay. Like markers, the API offers much customizability with classes like `CustomSWFOverlay`.

Navigation

An API element found only with the Flash APIs is the `NavigatorWidget`. This widget represents a unique set of user interfaces that provides users with an intuitive means of moving the map around. The `NavigatorWidget` consists of two elements, a zoombar and a miniature version of the map, called *mini-map*.

The zoombar handles much like the one we discussed in the earlier section "Zooming." There are two buttons, a plus button and a minus button, which handle zooming in and out respectively when pressed. The slider in the middle is draggable and changes the zoom level of the map as it is pressed and dragged.

Adding to the panning ability of the map is the mini-map element of the `NavigatorWidget`. Featured here is a smaller version of the map, displaying a bird's-eye view of the main viewable area. Just like the main map, the mini-map also consists of layers, but it is restricted for usability reasons to displayed map tiles. Hovering within the mini-map is a distinct gray rectangle. When clicked and dragged within the boundaries of the mini-map, this square prompts the main map to pan along with it. Another feature that usually goes unnoticed is that this gray square actually maintains the ratio of the main map should the browser window be resized. See Figure 1-4.

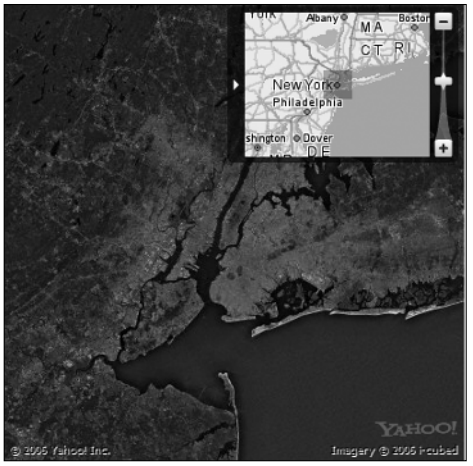

Figure 1-4

Summary

In this chapter, you learned the following:

- What a map API is and the basics of its application
- The three flavors of Yahoo! Maps APIs and their respective technologies that will be covered in this book
- Projection and scaling map technologies
- Several of the common features of map applications and the map APIs
- How map tools and interfaces work in the map APIs