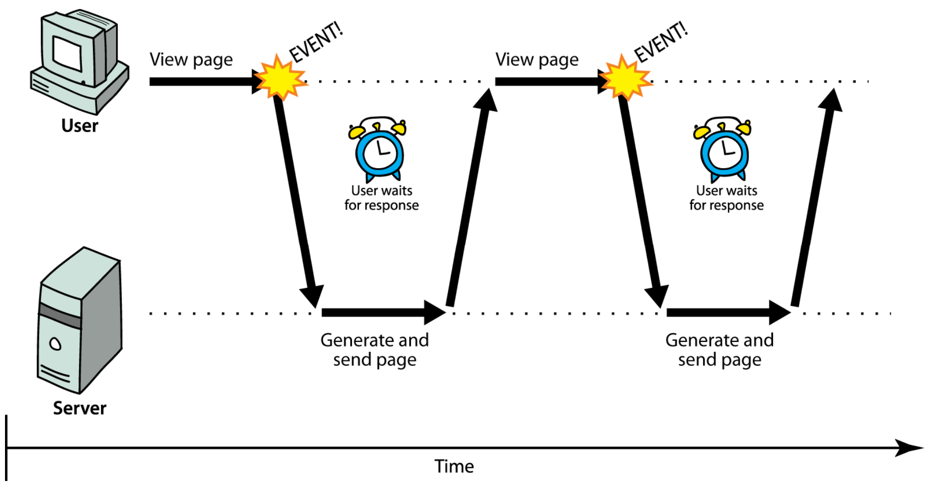

Figure 10.1 Classic web application model (synchronous)

When you use Ajax, your interactions trigger a request to receive a small amount of data. When the data arrives, rather than taking you to a new page with a different URL, you remain on the same page and the JavaScript code updates the page you're already viewing. All of this happens in the background, hidden to you.

synchronous communication

Interaction that only allows one action at a time and forces the user to wait for that action to complete.

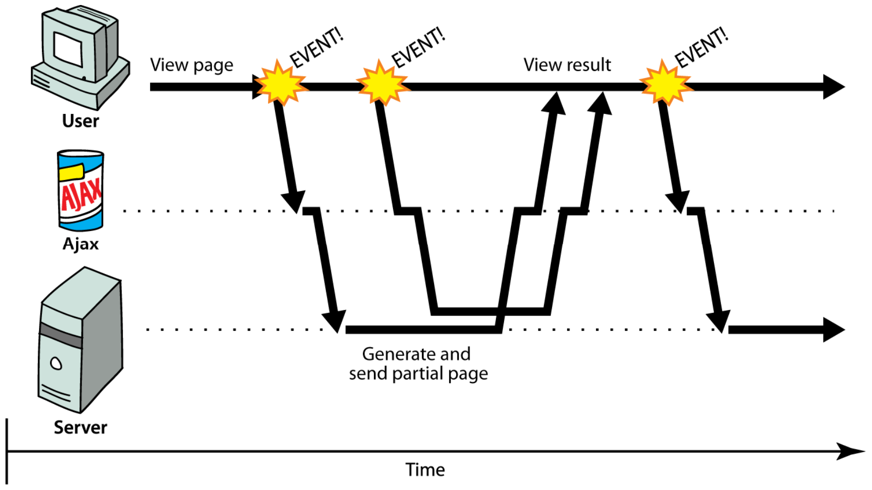

Figure 10.2 Ajax web application model (asynchronous)

This is called *asynchronous communication*, because multiple things can happen at the same time without needing to wait for each other. This asynchronous flow and ability to update an existing page are what powers modern web applications like Gmail and Flickr. The web site feels more like a coherent desktop application, because when you interact with it, small changes and updates occur to the page, rather than waiting for a lengthy reloading of a brand new page. In this way Ajax can greatly enhance the user experience of a web site. This is shown in Figure 10.2.

asynchronous communication

Interaction where many actions can occur at a time, in any order, without waiting for each other; the style of communication used by Ajax.

Ajax has many uses on the web today. Among these are the following:

- **Real-time form data validation**: Form data such as user IDs, serial numbers, postal codes, or even special coupon codes that require server-side validation can be validated in a form before the user submits a form.

- **Auto-completion**: A specific portion of form data such as a search term, email address, name, or city name may be auto-completed as the user types.
- **Load on demand**: Based on a client event, a web page can fetch more data in the background, allowing the browser to load pages more quickly.
- **Sophisticated user interface controls and effects**: Controls such as trees, menus, data tables, rich text editors, calendars, and progress bars allow for better user interaction with web pages, generally without requiring the user to reload the page.
- **Refreshing data and server push**: Web pages can repeatedly request or poll a server for up-to-date data to retrieve sports scores, stock quotes, weather, application-specific data, etc. Ajax techniques can be used to get a set of data without reloading a full page. Polling is not the most efficient means of ensuring that data on a page is the most current, but it is normally the best option for web pages at the moment due to the nature of HTTP.
- **Partial submit**: A web page can submit form data without requiring a full page refresh.
- **Mashups**: A web page can obtain data using a server-side proxy or by including an external script to mix external data with your application's or your service's data. You can mix content or data from a third-party application such as Google Maps with your own page.
- **Web applications**: Ajax techniques can be made to create single-page marshaled applications that look and feel much like a desktop application.

10.1.1 History and Compatibility

The term "Ajax" was coined by Jesse James Garrett in February 2005, but the technologies that make it possible have been around for many years. Microsoft invented the object that would become today's `XMLHttpRequest`. The object, which was then called `Microsoft.XMLHTTP`, served as a way to fetch data from a web server dynamically. It was created as part of an effort to build a more interactive web version of Microsoft Outlook. Once web developers discovered the existence of this object, they began experimenting with using it for their own dynamic web sites.

Over time, a standardized version of this object was proposed to the W3C, called `XMLHttpRequest`. This has become a web standard that is part of every modern browser, including Firefox, Safari, Opera, and others. Ironically, one browser not to support `XMLHttpRequest` is Microsoft Internet Explorer 6 and prior, which still use their old `Microsoft.XMLHTTP` object. In Internet Explorer 7, Microsoft added `XMLHttpRequest` support to match the other browsers.

The fact that IE did not support `XMLHttpRequest` object until IE7 has made it more cumbersome to write Ajax applications. Developers who want compatibility with older versions of IE and with compliant browsers must insert checks in their code to see which object to use. Thankfully, there is not much code needed to work around this, because once the object is created, it behaves the same way whether it's a `Microsoft.XMLHTTP` object or the standard `XMLHttpRequest` object.

Self-Check

1. What does the term "Ajax" stand for? Is Ajax a programming language? What language(s) and technologies are involved in Ajax?
2. Name three situations in which it would be useful to use Ajax on a web site.
3. What is the difference between synchronous and asynchronous communication between a web browser and web server? Which style does Ajax use, and why is this helpful?
4. Who created the technologies necessary to do Ajax? Which browsers support Ajax today?

10.2 Using `XMLHttpRequest` to Fetch Data

Since fetching data with Ajax can be complicated, we'll explain the Ajax process step-by-step. We'll begin with text data rather than complex XML data. We'll also begin with a simpler synchronous program rather than asynchronous. Each step will provide a working program that downloads and displays data using Ajax, with each version improving on previous ones. The steps will be:

- Synchronized JavaScript and text-only data (perhaps we'd call this "Sjat"?)
- Asynchronous JavaScript and text-only data (Ajat?)
- Asynchronous JavaScript and text-only data, using Prototype (Ajap?)
- Asynchronous JavaScript and XML data (true Ajax)

10.2.1 Synchronous Requests

Enough talk; let's write some Ajax code. Imagine that we have a web site with a text file of data. We also have a page with a button and a text area. We'd like to make it so that when the user clicks the button, the contents of the data file will be downloaded and shown inside the text area.

The basic code you need to synchronously fetch text data using `XMLHttpRequest` follows:

```
// this code is in some onscreen control's event handler
var ajax = new XMLHttpRequest();
ajax.open("GET", "url", false);
ajax.send(null);

// at this point in the code, the web request has completed
do something with ajax.responseText;
```

Example 10.1 Syntax template for synchronous Ajax request

Generally you have some kind of button or control that, when clicked, causes JavaScript code like the above to execute. You create an `XMLHttpRequest` object and ask it to retrieve data from the URL of interest. The contents of the response (i.e., the data) are stored into a property of the request object called `responseText`. You can do anything you like with this text, such as put it directly onto the page or process it in other ways. For example, the short page in Example 10.2 has a Load button that, when clicked, will retrieve data from the file `notes.txt` and place that data inside a `textarea` with the ID of `output`.

```
<textarea id="output" rows="4" cols="40"></textarea><br />
<button id="load">Load</button>
```

```
window.onload = function() {
  $("load").onclick = loadClick;
}

function loadClick() {
  var ajax = new XMLHttpRequest();
  ajax.open("GET", "notes.txt", false);
  ajax.send(null);
  $("output").value = ajax.responseText;   // request has completed
}
```

Example 10.2 Synchronous Ajax request