Chapter 1

What is Ajax?

Objectives:

- Understand the deficiencies of traditional, synchronous Web pages
- Specify the difference between synchronous and asynchronous communication
- Define Ajax and identify the primary technologies involved
- State the benefits and challenges of Ajax
- Create a basic Ajax request and handle the response
What is Ajax?

<table>
<thead>
<tr>
<th>Chapter Overview</th>
</tr>
</thead>
</table>

The technological restrictions of a traditional Web site, which led a user from one complete page to another, were quickly outgrown by business requirements and end user expectations. Various developments including new plug-ins, advances in client side scripting and markup languages, have been offered to bridge the experiential gap between a traditional desktop application and a modern Web site.

Of the numerous solutions offered, one of the most widely adopted is known as “Ajax.”

Unlike many other solutions, Ajax is not a new scripting or programming language: it does not require any plug-ins, and is supported by all modern, major desktop Web browsers. It provides its rich functionality through the creative and collaborative use of pre-existing technologies.

This chapter introduces you to this suite of technologies and their application towards the production of high performance, dynamic Web applications.

You begin by learning the basics of building a simple Ajax application, with as little as one XHTML page and two JavaScript functions, interacting with an existing resource on a Web server.
Traditional Synchronous Web Pages

- In order to understand what Ajax is and why it has become so popular, you need to first examine the behavior and challenges of traditional Web sites.

- Traditional Web sites engaged in a synchronous style of communication with the server:
  - A user opens up a Web browser and types in a URL into the address bar.
  - The browser creates an HTTP request for the resource specified in the URL and sends it off to the receiving server.
  - The receiving server parses the request, locates the resource, creates an HTTP response (typically the entire Web page in HTML) and sends it back to the user's browser.
  - The browser receives the response and begins to paint a Web page as specified by the mark-up (HTML, JavaScript, Cascading Style Sheets, etc.).
  - The browser then makes additional requests for other resources specified in the mark-up (such as images, external java script files and cascading style sheets), until the entire page is displayed.
  - Once the page had been completely rendered, the User clicks on a link, or a button, and the entire cycle happens again.
What is Ajax?

- There are several issues with this type of communication:
  - Slow performance
    - The server has to locate or dynamically create an entire Web page.
    - The client browser has to render the entire response.
    - There is unnecessary processing time in situations where only a small portion of the page needs to change.
    - Larger than necessary blocks of data are sent across the network.
  - Usability Issues
    - Unless frames are used (which have their own set of drawbacks), the user is not able to work on one portion of the page, submit a request, and then continue to work on another portion of the page.
    - Only one action occurs at a time, forcing the user to wait for the response from the server.
    - If the user attempts to work on another part of the page during this time, their typing is lost once the response is received and the page refreshed.
Synchronous vs. Asynchronous Communication

- The type of communication described above is known as a “Synchronous Request and Response Protocol.”
  - The word “synchronous” in this context, means that the request and response are executed as a pair – once a request is made, nothing else can take place until a response is received.
  - An asynchronous request removes this restriction by allowing other activities to occur before a response is received.

- Look at a familiar example: the Google Search Engine.
- First, examine how the initial load of a Web page from Google would occur through a synchronous request/response.
  - The user types “http://www.google.com” into the address bar of their browser.
  - The browser creates an HTTP request and sends it to Google.
  - Google responds by sending an HTTP response with an HTML page.
  - The browser parses through the markup and loads any subsequent external resources (images, external JavaScript files, etc.)
  - Each time a request is made the browser waits for the response until the entire screen has been rendered.
What is Ajax?

- After this process completes, the user sees:

- Next, the user starts to type in a search term. No other activity is happening at this time
  - The browser is not sending out additional requests, nor is it waiting for any responses.
Finally the user finishes typing in their search terms and clicks the “Google Search” button.

- The browser sends an HTTP request on behalf of the user, and waits for a response.
- No other work can take place on this page until the response returns:
What is Ajax?

• Now compare this with functionality enabled by allowing asynchronous communication to take place.
  • The initial page load is the same as before, but now as the user types, something dynamic occurs: the search box displays suggested search terms!
As the user types, the browser reads each character and sends it to the server.

- The server parses what is being typed.
- Next the server executes some business logic to determine the best possible search term suggestions, and returns these to the browser.
- Because this happens asynchronously, the user can continue to type before any response is received.

With synchronous communication, anything the user types in-between the time the request is sent, until the response is received, is lost.

- This is the key difference: in a synchronous mode *nothing* can occur after making a request until the response is received. Asynchronous communication removes this limitation.
- In addition, Ajax facilitates a clean redrawing of the search menu, without interrupting and refreshing anything else on the screen.
What is Ajax?

Ajax Defined

- Ajax is not a single scripting or programming language that one learns – it is a combination of several technologies.
  - The term was originally created in 2005 by author Jesse James Garrett, in an article entitled “Ajax: A New Approach to Web Applications”\(^1\)

- Ajax stands for: “Asynchronous JavaScript And XML.”

- Asynchronous
  - A mode of communication that allows one or more requests to be sent from the browser, without waiting for previously expected responses. During this time, the user or program is allowed to perform other tasks.

- JavaScript
  - Request creation and subsequent handling of the response is executed through a scripting language.
  - Although this is almost always implemented with JavaScript, it is possible to use other scripting languages such as VBScript, and so on.

- XML
  - There are several protocols that can be used to send data between the browser and the server, one of which is XML.
  - When XML is received from the server, it can be parsed and used to manipulate various elements on the Web page.

---

What is Ajax?

- Most Ajax implementations are slight variations of the technologies described in the acronym
  - All Ajax applications have some part that is executed in an **Asynchronous** manner.
  - Most use **JavaScript** as the scripting language for communication and page manipulation.
  - The data communicated between the browser and server may or may not be in the form of **XML**. Popular alternatives to XML are discussed in the “Transferring Data” chapter.
- Although an Ajax enabled application is not truly multithreaded, its’ asynchronous operations provide a virtually identical experience for the user.
  - So much so that Ajax is often referred to as an “emulated” or “pseudo” multithreaded technology.
What is Ajax?

XMLHttpRequest Object

- The “heavy lifter” of an Ajax application is the `XMLHttpRequest` object. This object is created through JavaScript and is used to:
  - 1 - Create an asynchronous HTTP request.
  - 2 - Register a JavaScript function, known as “The Callback Function.”
    - A Callback Function handles the parsing and rendering of the HTTP response (redrawing the appropriate portions of a Web page).
    - It does not require any special JavaScript syntax; it is a regular JavaScript function.
    - The term simply refers to its logical purpose for handling an asynchronous HTTP response.
  - 3 - Provide properties for handling the response, such as `readyState`, `status`, `responseText`, and so on.
    - This `readyState` property is an integer (between 0 and 4) which gives some details about the state of the request object.
    - Different browsers use these integers in different ways, but all use “4” to represent when the HTTP response has been received in its entirety.
    - You will examine the other `readyState` values in later chapters.
    - For most Ajax implementations, 4 is the only `readyState` of interest (since it is atypical to try to parse and render a response until it has been completely received).
    - The `status` property can be read to retrieve the HTTP response code sent from the server.
  - In later chapters, you see code examples showing how to instantiate this object, and what the different `readyState` integer values represent with regard to different browsers.
    - For now, the following examples are written specifically for browsers other than older versions of IE (Firefox, Chrome, Safari, and so on).
What is Ajax?

- Creating an Ajax Request and Handle an Ajax Response:

  1 - Create the Callback Function:

    - The Callback Function is registered with the XMLHttpRequest object, and is invoked every time the readyState changes.
    - Typically, you want to check for two things before processing the result:
      - The complete result has been received (i.e. `readyState == 4`)
      - The HTTP Status code was successful: (i.e. `status == 200`)
    - If both of these requirements are satisfied, you retrieve the data from the response by reading the `responseText` property on the request object.
    - The `responseText` property contains any data found in the body of the HTTP Response. This can be any type of text: plain text, XHTML, XML, etc.

```javascript
function myCallbackFunction()
{
    if ( req.readyState == 4 && req.status == 200 )
    {
        alert( "The Response is: " + req.responseText );
    }
}
```
What is Ajax?

- Unfortunately, not every request results in a “200” status code.
- Here is a table of some common responses you may need to account for (note – this is not a complete list):

<table>
<thead>
<tr>
<th>HTTP Status code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>OK</td>
</tr>
<tr>
<td>301</td>
<td>Moved Permanently</td>
</tr>
<tr>
<td>302</td>
<td>Found / Moved Temporarily</td>
</tr>
<tr>
<td>304</td>
<td>Not Modified</td>
</tr>
<tr>
<td>401</td>
<td>Unauthorized</td>
</tr>
<tr>
<td>403</td>
<td>Forbidden</td>
</tr>
<tr>
<td>404</td>
<td>Not Found</td>
</tr>
<tr>
<td>500</td>
<td>Internal Server Error</td>
</tr>
</tbody>
</table>

- 2 - Create the XMLHttpRequest object:

```javascript
req = new XMLHttpRequest();
```
• 3 - Configure the request object with message information:

  • To do this, call the **open()** function on the request object. This function has 5 parameters:

    ✷ The HTTP method to be used, such as “GET”, “POST”, and so on.

    ✷ The URL to be invoked.

    ✷ A Boolean representing whether or not the request should be handled in an asynchronous manner (“true” if it is, “false” if it isn’t).

    ✷ A username & a password (**optional** – used with Basic HTTP authentication).

  • The **open()** function only configures the request; data has not yet been sent to the server.

    /* No username or password is necessary here, so only the first three parameters are used. */
    req.open( "GET", "http://www.intertech.com/", true );

What is Ajax?

- 4 - Register the Callback Function:
  - In this example, the name of the Callback Function is `myCallbackFunction()`.
    ```javascript
    req.onreadystatechange = myCallbackFunction;
    ```
  - Notice that the Callback Function is registered with a property in the XMLHttpRequest object, called `onreadystatechange`.
  - Every time the `readyState` changes (from 0 to 4), this function is called.
  - Also notice that the name of the function is listed without the trailing empty parenthesis – i.e. `myCallbackFunction()`.
    ✤ In JavaScript, the absence of parenthesis is interpreted as a reference to the function.
    ✤ If the parenthesis were there, the function would actually be invoked at the time it was parsed.

- 5 - The request is sent to the server by calling the request object’s `send()` function:
  - This function takes a single parameter: any data that should be submitted in the body of the HTTP request object.
  - GET requests do not submit anything in the body, so “null” or an empty string is passed in.
  - Otherwise a POST request passes information in the body, for example, a String that lists name/value parameters from a form.
  - More information on GET and POST requests is found in Chapter 4.
    ```javascript
    /* Since our request is using the HTTP “GET” method, no data is passed into the send method. */
    req.send( null );
    ```
  - Putting this all together:
    ```javascript
    req = new XMLHttpRequest();
    req.open( "GET", "http://www.intertech.com/", true );
    req.onreadystatechange = myCallbackFunction;
    req.send( null );
    ```
Other Technologies Used with Ajax

- **XHTML:**
  - XHTML is a version of HTML that follows the strict rules of formatting for XML.
  - XHTML ensures that the Document Object Model is well formed, and thus easily traversed/manipulated through the DOM API.

- **The Document Object Model (DOM) API:**
  - The DOM API allows modifications to elements in an XHTML page programmatically, without needing to refresh the entire Web page.
  - The DOM API also provides handles for implementing an Event Driven application, such as responding to a button click, a mouse moving over an image, a change of value in a text box, and so on.

- **Cascading Style Sheets (CSS):**
  - CSS is used to configure the look and feel of a Web page.
  - CSS changes can be executed programmatically through properties exposed by the DOM API.

- These and other technologies are detailed in later chapters.
What is Ajax?

Benefits

- The “Google” example explored earlier illustrates that Ajax can provide a richer user experience:
  - Individual portions of the page can be modified without requiring a complete page refresh.
  - The Web page can dynamically react to user input at any time.
  - A user is able to continue to work on other areas of a page while additional requests and responses are being handled by the browser.
- In addition there is a significant improvement in both “actual” and “perceived” performance.
  - Redundancy is eliminated (only the portions of the page that have changed are updated), resulting in smaller amounts of data communicated between the browser and the server.
- Smaller amounts of data improves “actual” performance:
  - The server doesn’t need to take the time to create an entire Web page; only the portion that needs to change is generated.
  - The browser doesn’t need to render an entire Web page; only the portion that needs to change is processed.
  - Less data is transferred across networks, resulting in faster data transfer.
- “Perceived” performance gains can be achieved by using Ajax to strategically defer the loading of slower/large resources.
  - A traditional, synchronous Web page needs to load all of the resources on a page before a user is able to interact with it.
  - With Ajax, asynchronous requests can be made for “slower” rendering objects on a page (often displayed with a “loading” status message).
  - This allows the user to work on other portions of the page while a response is being created by the server, thereby increasing the “perceived” performance of a Web site.
What is Ajax?

Challenges

- Most of the challenges with Ajax are not “Ajax” specific; they are also found in traditional implementations that use JavaScript and the DOM API.

- Browser Variations:
  - Although standards have emerged over time, IE and Firefox created their own unique API for dynamic scripting.
  - As result, a developer who wants their application to support multiple browsers needs to branch their code.
  - Some of the primary areas where this branching must take place are:
    - Creating the XMLHttpRequest Object.
    - Using the DOM API.
    - Changing the look and feel of a page through CSS.
    - Event Handling.

- Additional challenges include:
  - Debugging.
  - Caching of data.
  - Breaking of the “Back” button.

- Detailed examples of these challenges, and best practices for mitigating their impact, are given in later chapters.
What is Ajax?

Code Example - Create a Simple Ajax Page

- Requirements are simplified for this example.
  - This page has a text box which is used to submit a username. The server checks to see if the user is logged in and returns a status message.
  - This example is written specifically for the Firefox Web browser.
    - In later chapters, you will examine some of the coding considerations that must occur to support multiple browsers.

- The flow is as follows:
  - When a name is typed and the “submit” button is clicked, an XMLHttpRequest object is created by a JavaScript function called `submitUsername()`.
  - The XMLHttpRequest object is used to submit the username to the server, as well as retrieve the “readyState”, “status” and “responseText.”
  - For the purposes of this example, assume the server-side logic for creating the status message has already been created.
    - Requests are submitted to a resource on the server called `StatusVerifier`.
    - The server creates a status message indicating whether or not the user is online or offline at this time, and return it to the browser (via an HTTP Response).
  - A Callback Function, called `displayStatus()`, is invoked by the browser when a response has been received by the server.
  - The `displayStatus()` function checks the `readyState` to see if the complete response has been received, as well as the HTTP `status` code to make sure there weren’t any issues processing the request.
What is Ajax?

Configuration includes: HTTP Method, URL, isAsyncronous, and the name of the CallBack function (displayStatus()).

HTTP Response

onreadystatechange checks readystate & status

Admin See Status Displayed

Admin Continues to Work on Other Portions of the Web Page
What is Ajax?

- **Step #1**: Create the CheckStatus.html page.
  - Create the input text box and a button which calls a JavaScript function, `submitUsername()`, to create the XMLHttpRequest object.
  - In addition, a placeholder to display the status message returned from the server is needed.
  - One of the simplest ways to do this is to create an empty div tag. Set this div’s id to “status.”

```html
<body>
  Username: <input type="text" name="username" id="username" />
  <button onclick="submitUsername()">Submit</button><br />
  <div id="status"></div>
</body>
```
• **Step #2:** Implement the `submitUsername()` function to create, configure, and submit the XMLHttpRequest object

  • This request’s method is submitted as an HTTP GET, so the username parameter must be appended to the URL as a “query string.”

  • A query string starts with a question mark (?) and contains name/value pairs.

  • The name is separated from the value by an equals sign (=) and each pair is separated from one another by an ampersand (&).

  Example:

  ```
  http://localhost/ajax/welcome?firstname=Jason&lastname=Shapiro
  ```

  • It is a best practice to always “escape” the parameters you are submitting to the server.

  • This ensures that the server will not misinterpret any of the characters in the parameters as special commands.

  • JavaScript provides an escape() function for this purpose.

  ```
  function submitUsername()
  {
    var username = document.getElementById( "username" ).value;
    var url = "StatusVerifier?username=" + escape( username );
    var httpMethod = "GET";
    var isAsynchronousRequest = true;
    req = new XMLHttpRequest();
    req.open( httpMethod, url, isAsynchronousRequest );
    // ...
  }
  ```

  • Register the callback function, `displayStatus()`, with the `onreadystatechange` property of the XMLHttpRequest object.

  • Since there is no text in the body of an HTTP GET request, just pass in “null” as a parameter to the XMLHttpRequest's `send()` function.

  ```
  // ...
  req.onreadystatechange = displayStatus;
  req.send( null );
  ```
• **Step #3:** Implement the callback function, `displayStatus()` to handle the response from the server.

  - Make sure the entire response has been received before performing any actions. Therefore, check to see if the `readyState` has a value of 4 (meaning: Complete Response Received).

  - In addition, make sure the server was able to process the request properly, so look at the HTTP response `status` to ensure that it returned a 200 (meaning: OK).

  - If both of these tests pass, grab the `responseText` from the request object and add it to the placeholder div.

```javascript
function displayStatus() {
    if ( req.readyState == 4 && req.status == 200 ) {
        var statusElement = document.getElementById( "status" );
        statusElement.innerHTML = req.responseText;
    }
}
```
The finished “CheckStatus.html” source code looks like this:

```html
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
    <title>Check User Status</title>
    <script language="JavaScript">
      var req;

      function submitUsername()
      {
        var username = document.getElementById( "username" ).value;
        var url = "StatusVerifier?username=" + escape( username );
        var httpMethod = "GET";
        var isAsynchronousRequest = true;

        req = new XMLHttpRequest();
        req.open( httpMethod, url, isAsynchronousRequest );
        req.onreadystatechange = displayStatus;
        req.send( null );
      }

      function displayStatus()
      {
        if( req.readyState == 4 && req.status == 200 )
        {
          var statusElement = document.getElementById( "status" );
          statusElement.innerHTML = req.responseText;
        }
      }
    </script>
  </head>
  <body>
    Username: <input type="text" name="username" id="username" />
    <button onclick="submitUsername()">Submit</button><br />
    <div id="status"></div>
  </body>
</html>
```
What is Ajax?

- The resulting Web page looks like this:

  ![Web page image](image1)

  The administrator enters a username to check their online status:

  ![Username input](image2)

  Below is the display after the administrator has clicked the “Submit” button:

  ![Display after submit](image3)
General Ajax Resources

- "AJAX Tutorial". W3Schools. 9/1/2009
  <http://www.w3schools.com/Ajax/Default.Asp>.


- Riordan, Rebecca M. Head First Ajax. Sebastopol: O'Reilly, 2008.

Lab Exercise: Intro to Ajax Lab
Chapter Summary

- Traditional, synchronous Web pages had usability and rendering restrictions
  - These led to a large technological gap between a Web site and a desktop application.
  - Ajax is one of the many solutions offered to bridge this gap.
- Ajax is not a new scripting language nor is it a single protocol.
  - Ajax is a suite of technologies that are used collaboratively to provide a dynamic user experience through asynchronous communications.
  - These technologies typically include JavaScript, XHTML, CSS, and the DOM API.
- The “heavy lifter” in Ajax is the XMLHttpRequest object.
  - Although different browsers may have different implementations of this object, they all share the same basic interface.
  - Later chapters will show how to instantiate the different implementations.
  - The XMLHttpRequest object is initialized with:
    - An HTTP method ("GET", "POST", etc.)
    - A URL.
    - A Boolean representing whether or not it should be handled in an asynchronous manner,
    - And optionally, a username & password.
- A readyState property can be examined to see what the current state is for this object.
  - The value of this property is an integer between 0 and 4.
  - When the complete response has been received, and is ready to be parsed, the readyState is the number “4.”
- Further configuration of this object includes registering a Callback Function which is the JavaScript code that invokes whenever its readyState changes.
What is Ajax?

- Ajax offers many benefits:
  - Performance gains.
  - The ability to dynamically update individual portions of a Web page while the user is working.

- There are also many challenges to using Ajax:
  - There are unique JavaScript APIs for different browsers, potentially requiring branching in one’s code.
  - It is very difficult to debug asynchronous Web applications.
  - Later chapters identify these challenges in detail and suggest best practices to mitigate them where applicable.