When the Internet first came out, there was no concept of having private data.

That's right!

This meant that anyone could access any resource. All you needed to know was the URL.

We have come a long way because today, most applications have authentication to some degree.

Traditionally, the most common way of authenticating a user is through `username` and `password`. This was provided for natively in the `XMLHttpRequest()` object (remember the final arguments to the `open()` method are **username** and **password**).

"Okay Clyde, I get it. Authentication is used today. But how can I use AJAX to provide the server with authentication information?"

Great question.

How to authenticate your user?

Authentication at its crux only consists of two steps:

1. A user provides credentials to a server.
2. The server (either the server software itself or server-side code) checks the credentials and sends back a response to the client.

Now, this is where things get complicated: there are many ways you can achieve the two preceding steps. When you are creating your web app, you have a choice about how to authenticate your users.

Your choices are as follows:

- HTTP authentication (built into the HTTP protocol)
- Application-specific authentication (this is a way of moving the authentication problem from a web server to the application itself)

Which one you choose will depend on your skill level and on where you want the

authentication and authorization logic to be handled.

HTTP Authentication (such as Basic Authentication) means that HTTP and the web server software are responsible for authentication. Application authorization means that the web application (such as Java, Node, PHP, Django, and others) would control

authentication and authorization. This means that if you want to use application authorization, you will need to know how to write server-side code.

There is no right answer when it comes to which method you want to use.

You would probably want to use HTTP Authentication when you don't have or need any application logic yourself. Maybe your site is really small, and security is not much of an issue. In this case, using HTTP Authentication is fine.

Sometimes, you can use both types of authentications (I will show you later how you can use ***.htpasswd*** authentication on an Apache server and give the configuration file custom directives).

However, on the flip side, you may want to have more control over authentication. Most web applications today use sessions (for example, cookies) and this is a great reason to use web application logic to handle your authentication.

We are discussing HTTP Basic Authentication

This course is not about authentication, so I don't want to get deeper into this rabbit hole. As I have mentioned, the original XHR object contained an option to include user credentials (**username** and **password**), which is used if you want to use HTTP Basic Authentication.

Bottom line: For the remainder of this section, I will be discussing how you can use HTTP Basic Authentication, using AJAX to pass in the username and password.