1. Course Intro
2. Client Server Demonstration
3. Ajax Definition & Examples
4. APIs
5. Create An Async Request with XHR
6. The XHR Object
7. XHR's .open() method
8. XHR's .send() method
9. A Full Request
10. Project Initial Walkthrough
11. Setting a Request Header
12. Project Final Walkthrough
13. XHR Recap
14. XHR Outro

Here's the full code that we've built up that creates the XHR object, tells it what info to request, sets up handlers for a success or error, and then actually sends the request:

```
function handleSuccess () {
  console.log( this.responseText );
// the HTML of https://unsplash.com/}
function handleError () {
  console.log( 'An error occurred \uD83D\uDE1E' );
}
const asyncRequestObject = new XMLHttpRequest();
asyncRequestObject.open('GET', 'https://unsplash.com');
asyncRequestObject.onload = handleSuccess;
asyncRequestObject.onerror = handleError;
asyncRequestObject.send();
```

APIs and JSON

Getting the HTML of a website is ok, but it's probably not very useful. The data it returns is in a format that is extremely difficult to parse and consume. It would be a lot easier if we could get just the data we want in an easily formatted data structure. If you're thinking that JSON would be a good idea, then you're right and I'll give you a piece of my cake!

Instead of requesting the base URL for Unsplash, let's create an app that pulls an image from Unsplash's API and relevant articles from the New York Times.

When making a request from an API that returns JSON, all we need to do is convert that JSON response into a JavaScript object. We can do that with `JSON.parse();`. Let's tweak the onload function to handle a JSON response:

```
function handleSuccess () {
const data = JSON.parse( this.responseText ); // convert data from JSON to a JavaScript object
console.log( data );
}

asyncRequestObject.onload = handleSuccess;
```