You may have noticed in our example that the text returned from our server can also be accessed in the `xhr.response` property.

So, why did I use `responseText` and what is the difference between `response` and `responseText`?

Firstly, they are very similar in usage.

If we are dealing with raw text, you can use either one. The reason I used `responseText` was that I know the server is sending back raw text and it therefore made more intuitive sense to me.

My general rule is the following:

1. Use the `xhr.response` when you get JSON from the server. If you set the `responseType` property to JSON, then the `xhr.response` property will automatically convert the JSON into a JavaScript object. You can also use the `xhr.response` property for other types of data such as audio, video, and so on.
2. Use `xhr.responseText` when the response data type is raw text. This means you can't use it for other exotic data types. For example, if the data is video or audio, you can't use the `responseText` property because those media types can't be parsed according to a DOM String.

CONCLUSION

Taking a step back, remember that the main body of all response messages is **binary** data. The browser at some point needs to convert that binary data into a format that we can read.

Bottom line: The `response` and `responseText` interfaces are in fact the only convenient way for users to get the parsed response in a readable format, eliminating the step of manually parsing the response body yourself.