AJAX and Axios

Tessema Mengistu (Ph.D.)
Department of Computer Science
Virginia Tech
Mengistu@vt.edu

Outline

- AJAX Overview
- Axios Library

Introduction

- There are many tools you can use to make HTTP requests in React applications
  - Ajax
    - Vanilla JavaScript
  - Fetch API
    - Browser’s built-in
  - Axios
    - A lightweight package

AJAX

- AJAX - Asynchronous JavaScript and XML
  - Allows:
    - Asynchronous communication with servers
    - Partial content retrieval from the server
  - Uses a browser built-in object
    - **XMLHttpRequest**

AJAX in Action

- Steps to make AJAX work:
  1. A client event occurs and triggers the whole process
  2. An XMLHttpRequest object is created
  3. The XMLHttpRequest object is configured
  4. The XMLHttpRequest object makes an asynchronous request to Webserver
  5. Webserver returns the result (not necessarily be in XML)
  6. The XMLHttpRequest object calls the callback function (JavaScript on the client) that processes the result

AJAX

- The asynchronous request must be made through an XMLHttpRequest object created as follows:
  - let xhr = new XMLHttpRequest();
- XMLHttpRequest object has important methods and properties.

new XMLHttpRequest()	Creates a new XMLHttpRequest object
abort()	Cancels the current request
getAllResponseHeaders()	Returns header information
getResponseHeader()	Returns specific header information
open(method,url,async,user,psw)	Specifies the request *method*: the request type GET or POST *url*: the file location *async*: true (asynchronous) or false (synchronous) *user*: optional user name *psw*: optional password
send()	Sends the request to the server Used for GET requests
send(string)	Sends the request to the server. Used for POST requests
setRequestHeader()	Adds a label/value pair to the header to be sent

Source: W3Schools

Property	Description
onload	Defines a function to be called when the request is received (loaded)
onreadystatechange	Defines a function to be called when the readyState property changes
readyState	Holds the status of the XMLHttpRequest. 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
responseText	Returns the response data as a string
responseXML	Returns the response data as XML data
status	Returns the status-number of a request 200: "OK" 403: "Forbidden" 404: "Not Found"
statusText	Returns the status-text (e.g. "OK" or "Not Found")

Source: W3Schools

Axios

- **Promise** based HTTP client for the browser and node.js
- Uses **XMLHttpRequest** under the hood.
- **Syntax:**

```
axios.method(param)
  .then(function (response) {
            // handle success
    })
  .catch(function (error) {
    // handle error
   })
  .finally(function () {
    // always executed
  });
```

Axios

- Axios provides a set of shorthand methods for performing different types of requests.
  - *`axios.request(config)`*
  - *`axios.get(url[, config])`*
  - *`axios.delete(url[, config])`*
  - *`axios.head(url[, config])`*
  - *`axios.options(url[, config])`*
  - *`axios.post(url[, data[, config]])`*
  - *`axios.put(url[, data[, config]])`*
  - *`axios.patch(url[, data[, config]])`*

Axios

- A Fulfilled Promise object contains:

```
{
  // `data` is the response that was provided by the server
  data: {},
   // `status` is the HTTP status code from the server response
  status: 200,
   // `statusText` is the HTTP status message from the server response
  statusText: 'OK',
   // `headers` the headers that the server responded with
  // All header names are lower cased
  headers: {},
   // `config` is the config that was provided to `axios` for the request
  config: {},
   // `request` is the request that generated this response
  // It is the last ClientRequest instance in node.js (in redirects)
  // and an XMLHttpRequest instance the browser
  request: {}
}
```

References

- https://axios-http.com/