Working with APIs in JavaScript

An API is simply a medium to fetch or send data between interfaces. Let's say you want to make an application that provides the user with some real-time data fetched from the server or maybe even allows you to modify or add data to some other endpoint. This is made possible by the API or the Application Programming Interface.

We will use a simple public API that requires no authentication and allows you to fetch some data by querying the API with **GET** requests.

https://randomuser.me/ is a website that provides dummy data for random users that we can easily work with. We can get the response by making a request to https://randomuser.me/api/. The JSON response that we receive is in the following format:

-


```
{
    "results": [
          {
                "gender": "female",
                "name": {
                      "title": "Miss",
                      "first": "Nina",
                      "last": "Simmmons"
                },
                "location": {
                      "street": {
                            "number": 970,
                            "name": "Eason Rd"
                      },
                      "city": "Fullerton",
                      "state": "Wyoming",
                      "country": "United States",
                      "postcode": 57089,
                      "coordinates": {
                            "latitude": "83.1807",
                            "longitude": "104.7170"
                      },
                      "timezone": {
                            "offset": "+8:00",
                            "description":
          "Beijing, Perth, Singapore, Hong Kong"
                      }
                },
                "email": "nina.simmmons@example.com",
                "login": {
                      "uuid": "bd0d135f-84df-4102-aa4f-5baaa41baf5c",
                      "username": "yellowfrog722",
                      "password": "dawg",
                      "salt": "q28gdiyN",
                      "md5": "291987daea22bb91775226574925b271",
                      "sha1": "a0463a26ea5c2ff4f3ad498fd01c5994926e5021",
                      "sha256":
```

```
"6583eb74ca08bfac50b3b29aa52c9f02ea5d9d017fef0e5a5a6fae4f5225f928"
                },
                "dob": {
                        "date": "1980-11-01T23:10:05.403Z",
                        "age": 40
                },
                "registered": {
                        "date": "2013-04-02T02:26:52.904Z",
                        "age": 7
                },
                "phone": "(216)-693-7015",
                "cell": "(501)-534-9413",
                "id": {
                        "name": "SSN",
                        "value": "847-09-2973"
                },
                "picture": {
                        "large":
"https://randomuser.me/api/portraits/women/60.jpg",
                        "medium":
"https://randomuser.me/api/portraits/med/women/60.jpg",
                        "thumbnail":
"https://randomuser.me/api/portraits/thumb/women/60.jpg"
                },
                "nat": "US"
        }
    ],
    "info": {
        "seed": "82a8d8d4a996ba17",
        "results": 1,
        "page": 1,
        "version": "1.3"
    }
}
```


We now can properly refer to this json tree using javaScript commands, and load them in our DOM.

There are generally two ways to do this – 1. AJAX & 2. Fetch()

AJAX – Asynchronous JavaScript and XML

What is AJAX?

AJAX = **A**synchronous **J**avaScript **A**nd **X**ML.

AJAX is not a programming language.

AJAX just uses a combination of:

- A browser built-in `XMLHttpRequest` object (to request data from a web server)
- JavaScript and HTML DOM (to display or use the data)

AJAX is a misleading name. AJAX applications might use XML to transport data, but it is equally common to transport data as plain text or JSON text.

AJAX allows web pages to be updated asynchronously by exchanging data with a web server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page. Example-

```html
<!DOCTYPE html>
<html>
<body>

<div id="demo">
  <h2>Let AJAX change this text</h2>
  <button type="button" onclick="loadDoc()">Change Content</button>
</div>

</body>
</html>
```

```javascript
function loadDoc() {
  const xhttp = new XMLHttpRequest();
  xhttp.onload = function() {
    document.getElementById("demo").innerHTML = this.responseText;
    }
  xhttp.open("GET", "ajax_info.txt", true);
  xhttp.send();
}
```

```javascript
// Create a new XMLHttpRequest object
const xhr = new XMLHttpRequest();

// Set up the request
xhr.open('GET', 'https://example.com/api/data.xml');
// Set the response type to 'document', which will automatically parse the XML response
xhr.responseType = 'document';
// Set up a callback function to handle the response
xhr.onload = function() {
  // Check if the request was successful
  if (xhr.status === 200) {
    // Get the response XML document
    const xmlDoc = xhr.responseXML;
    // Use the XML DOM to extract data from the response
    const dataNode = xmlDoc.getElementsByTagName('data')[0];
    const dataValue = dataNode.textContent;
    // Do something with the extracted data
    console.log(`Data value: ${dataValue}`);
  } else {
    // Handle errors
    console.error('Request failed');
  }
};
// Send the request
xhr.send();
```

What is Fetch()?

Fetch is a modern JavaScript API that provides an easy way to make asynchronous HTTP requests to a server and handle responses in a way that's more flexible than the older XMLHttpRequest (XHR) API. It's available in most modern browsers and is a part of the JavaScript standard.

A basic fetch request is really simple to set up. Have a look at the following code:

```
fetch("http://example.com/movies.json")
  .then((response) => response.json())
  .then((data) => console.log(data));
```

```
fetch('https://example.com/api/data')
  .then(response => {
    // Check if the response was successful
    if (!response.ok) {
      throw new Error('Network response was not ok');
    }

    // Parse the response as JSON
    return response.json();
  })
  .then(data => {
    // Do something with the JSON data
    console.log(data);
  })
  .catch(error => {
    // Handle errors
    console.error('Error fetching data:', error);
  });
```