# Node Faker

What I plan to do is create anode app that will generate fake people at an endpoint. I'll be using a frontend created with HTML and JavaScript that can be released to display a random fake person.

## The server

On the server, I created a node project titled "faker" - a library that I used immensely in python. It seems that node also has a similar library. There are other dependencies:

1. faker
2. express
3. cors

The cors is of particular interest and I'll talk about it a bit later. For now we can install the libraries with:

```
$ npm i faker express cors
```

If you face any dependency issue, you can force a fix by:

```
$ npm audit fix --force
```

Once all the dependencies are installed, I created a file named index.js and the code was:

index.js

```
var express = require('express');
var faker = require('faker');

var cors = require('cors');

var app = express();
var port = 3000;

app.use(cors());

app.get('/user', function(req, res) {
```

index.js

```
    var fakeUser = {
        name: faker.name.findName(),
        address: faker.address.streetAddress(),
        nationality: faker.address.country(),
        dob: faker.date.past()
    };

    res.json(fakeUser);
});

app.get('/', function(req, res) {
    res.send('Welcome to my Node.js app! Access /user to generate random user data.');
});

app.listen(port, '0.0.0.0', function() {
    console.log('Server running on http://localhost:' + port + ', ready to hustle!');
});
```

Now when you run the application, it will be served on port 3000 and you can go to `http:your_url:3000/` to land on a welcome site and `http:your_url:3000/user` to get a random user object that would typically look like:

```
{
    "name": "Rodney Reinger",
    "address": "233 Freeda Valleys",
    "nationality": "Italy",
    "dob": "2023-06-07T07:38:38.659Z"
}
```

You can also try with Postman.

You should get a different random fake user everytime you invoke the endpoint.

Now let's create a frontend.

## The Frontend

I decided to make a bare bones frontend using just HTML, JavaScript and CSS. The three files that I created are:

1. index.html
2. script.js
3. style.css

The contents are as follows:

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>User Data Display</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <div id="userData" class="user-data">
        <!-- User data will be displayed here -->
    </div>
    <button id="reloadButton" class="reload-button">Reload Data</button>

    <script src="script.js"></script>
</body>
</html>
```

script.js

```
document.getElementById('reloadButton').addEventListener('click', loadData);

function loadData() {
    fetch('http://206.189.37.108:3000/user')
        .then(response => response.json())
        .then(data => {
            var userDataDiv = document.getElementById('userData');
            userDataDiv.innerHTML = `
                <h2>${data.name}</h2> <!-- The name's the game -->
                <p><strong>Address:</strong> ${data.address}</p> <!-- Where they lay their hat -->
                <p><strong>Nationality:</strong> ${data.nationality}</p> <!-- Where they hail from -->
```

script.js

```
                <p><strong>Date of Birth:</strong> ${new Date(data.dob).toLocaleDateString()}</p> <!-- When they first hit the streets -->
            `;
        })
        .catch(error => console.error('Error:', error));
}

loadData();
```

style.css

```
body {
    font-family: Arial, sans-serif;
    display: flex;
    flex-direction: column;
    align-items: center;
    margin-top: 50px;
}

.user-data {
    border: 2px solid #007bff;
    border-radius: 10px;
    padding: 20px;
    margin-bottom: 20px;
}

.reload-button {
    background-color: #007bff;
    color: white;
    border: none;
    padding: 10px 20px;
    border-radius: 5px;
    cursor: pointer;
}

.reload-button:hover {
    background-color: #0056b3;
}
```

Now although the system works fine when viewing from POstman or the browser, you are likely to get a blank response from a browser app unless you set the CORS.

Cross-Origin Resource Sharing (CORS) is a security feature implemented in web browsers to control how web pages in one domain interact with resources in another domain. By default, for security reasons, web browsers restrict web pages from making requests to a different domain than the one that served the web page, known as the Same-Origin Policy. CORS provides a way for servers to tell the browser that it's safe to allow these cross-origin requests.

## How CORS Works:

### Simple Requests:

For basic requests (like GET or POST with certain "safe" content types), the browser automatically sends the request to the server. The server then includes CORS headers in the response to indicate whether the browser should allow the web page to access the resource. For example, the Access-Control-Allow-Origin header can specify which domains are permitted to access the resource.

### Preflight Requests:

For more complex requests (like PUT, DELETE, or those with custom headers), the browser sends a preliminary "preflight" request using the OPTIONS method. This preflight request checks whether the actual request is safe to send. The server responds with the appropriate CORS headers to indicate if the actual request is allowed. If so, the browser then sends the actual request.

## The Role of CORS in this Application:

In this case, the Node.js server and the frontend application may be running on different origins (different domains, ports, or protocols). Without CORS, requests from your frontend to the Node.js server would be blocked by the browser due to the Same-Origin Policy.

By using the cors middleware in your Express application:

index.js

```
var cors = require('cors');
...
app.use(cors());
```

You're telling your Node.js server to include CORS headers in its responses. This essentially instructs the browser that it's okay to allow your frontend application to access resources from the server, even though they're not on the same origin.

## Common Use Cases for CORS:

### Development Environments:

Often, in development, your frontend and backend run on different ports or domains, so you need CORS to allow them to communicate.

### APIs:

If you're building an API that should be accessible from various clients (different domains), you'll need to configure CORS to specify which clients are allowed to access your API.

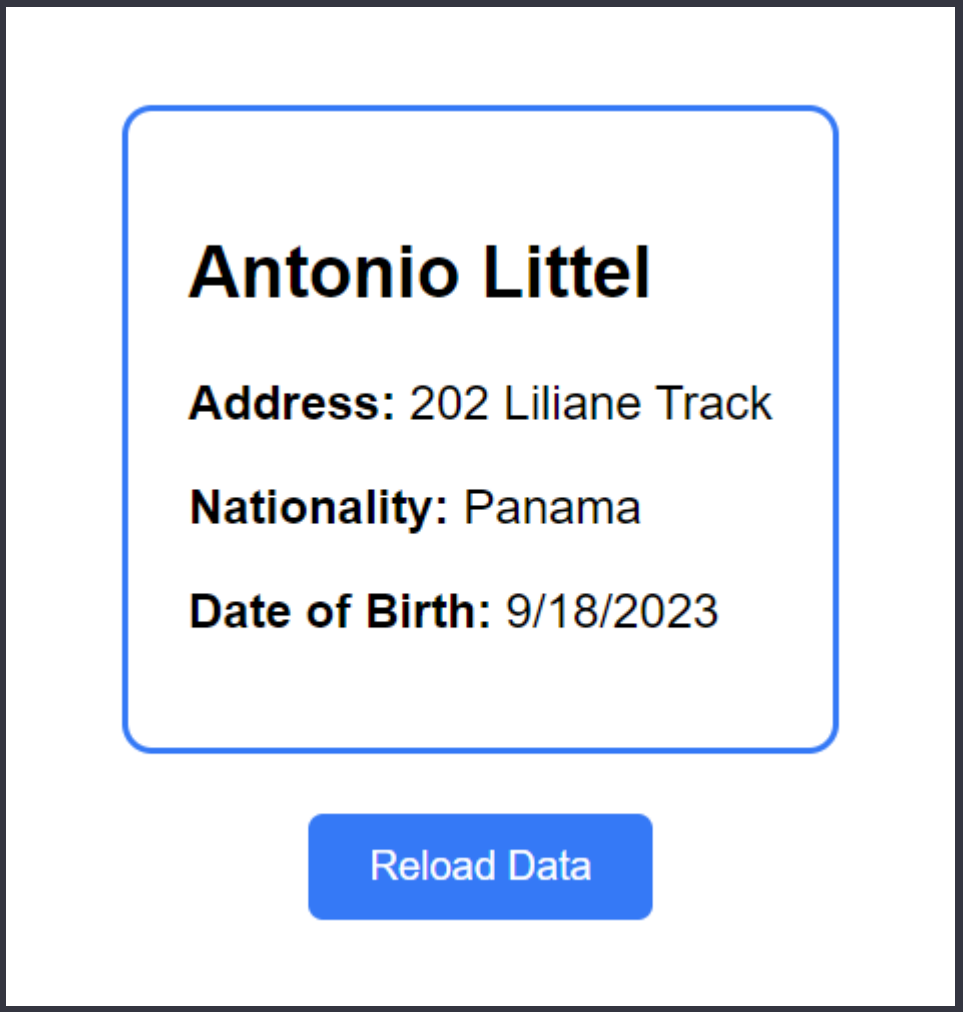
CORS is a crucial part of modern web development, enabling more flexible and secure interactions between different web applications and services.

# Running the App

To run the app, go to the frontend file path and run a simple python web server. Most operating systems have it preinstalled and can be run wit the CLI:

```
$ python3 -m http.server
```

ONce it start, you can visit the URL: `http://localhost:8000` and see:




Great! Try it out and have fun!! Happy coding guys!!!