Handling CORS with Netlify POST Requests

Netlify Functions make it possible to deploy server-side code that you can access as API endpoints. While handling GET requests with Netlify functions is straightforward and very well documented, you can run into quite a few issues while handling complex requests, like POST requests.

Cross-Origin Resource Sharing(CORS) is a HTTP-header security mechanism used to restrict which websites . Web browsers will restrict HTTP requests initiated by scripts like `XMLHTTPRequest` and the Fetch API because of the same-origin policy, so if you have a client facing web page at `https://foo.bar`, you'll need to enable CORS on the server-side before you can load resources from `https://functions.foo.bar`.

To start with you'll need a Netlify Function to work with. Here's a simple one that receives a name from a JSON object and returns a Hello message to the client:

```
exports.handler = async (event) => {
  const { name } = JSON.parse(event.body);
  return {
    statusCode: 200,
    body: JSON.stringify({ message: "Hello, " + name }),
  };
};
```

If you try to call this function through a development tool like Postman, or using `curl` it will run successfully because they do not enforce a same-origin policy, but web browsers will reject this request with a CORS error. When browsers make complex requests, like POST requests, they first send a pre-flight `OPTIONS` request to find out what headers the server accepts before sending the POST request.

You need to define what headers your function accepts:

```
const headers = {
  "Access-Control-Allow-Origin": "*",
  "Access-Control-Allow-Headers": "Content-Type",
  "Access-Control-Allow-Methods": "GET, POST, OPTIONS",
};
```

These headers mean that the server accepts `GET, POST, OPTIONS` HTTP requests, it also accepts the `Content-Type` header which describes what type of content is being sent in a `POST` request and the wildcard `"*"` means that it accepts requests from any origin. Note that it is safer to choose a specific origin instead of accepting requests from every origin.

Now you're going to check the value of `event.httpMethod` to see what HTTP request method is being sent from the client. If the method is `OPTIONS`, you need to return 200 as a status code and the headers object you defined too. This tells the browser that the pre-flight call was successful and it is allowed to request resources from your function.

```
if (event.httpMethod === "OPTIONS") {
  return {
    statusCode: 200,
    headers,
    body: JSON.stringify({ message: "Successful preflight call." }),
  };
}
```

Your browser will now send your `POST` request, and you will check the value of `event.httpMethod` to confirm that it is a `POST` request. Now you can return the status code 200, the "Hello" message and your headers object. It's important to return the headers object again when responding to the `POST` request or you will receive CORS errors even though your function is running successfully.

```
else if (event.httpMethod === "POST") {
    const { name } = JSON.parse(event.body);
    return {
      statusCode: 200,
      headers,
      body: JSON.stringify({ message: "Hello, " + name }),
    };
```

You have now successfully handled CORS with a POST request in your Netlify function. The full function looks like this:

```
exports.handler = async (event) => {
  const headers = {
    "Access-Control-Allow-Origin": "*",
    "Access-Control-Allow-Headers": "Content-Type",
    "Access-Control-Allow-Methods": "GET, POST, OPTIONS",
  };
  if (event.httpMethod === "OPTIONS") {
    return {
      statusCode: 200,
      headers,
      body: JSON.stringify({ message: "Successful preflight call." }),
    };
  } else if (event.httpMethod === "POST") {
    const { name } = JSON.parse(event.body);
    return {
      statusCode: 200,
      headers,
      body: JSON.stringify({ message: "Hello, " + name }),
    };
  }
};
```