Cookbook

This cookbook contains recipes for some commonly requested features.

In order to keep axios as lightweight as possible, it is often necessary to say no to feature requests. Many of these use cases can be supported by augmenting axios with other libraries.

Promise.prototype.done

```
$ npm install axios promise --save
```

```
const axios = require('axios');
require('promise/polyfill-done');

axios
  .get('http://www.example.com/user')
  .then((response) => {
    console.log(response.data);
    return response;
  })
  .done();
```

Promise.prototype.finally

```
$ npm install axios promise.prototype.finally --save
```

```
const axios = require('axios');
require('promise.prototype.finally').shim();

axios
  .get('http://www.example.com/user')
  .then((response) => {
    console.log(response.data);
    return response;
  })
  .finally(() => {
    console.log('this will always be called');
  });
```

Inflate/Deflate

```
$ npm install axios pako --save
```

```
// client.js
const axios = require('axios');
const pako = require('pako');

const user = {
```

```
  firstName: 'Fred',
  lastName: 'Flintstone'
};

const data = pako.deflate(JSON.stringify(user), { to: 'string' });

axios
  .post('http://127.0.0.1:3333/user', data)
  .then((response) => {
    response.data = JSON.parse(pako.inflate(response.data, { to: 'string' }));
    console.log(response.data);
    return response;
  });
```

```
// server.js
const pako = require('pako');
const http = require('http');
const url = require('url');

const server = http.createServer((req, res) => {
  req.setEncoding('utf8');

  const parsed = url.parse(req.url, true);
  const pathname = parsed.pathname;

  if (pathname === '/user') {
    let data = '';
    req.on('data', (chunk) => {
      data += chunk;
    });

    req.on('end', () => {
      const user = JSON.parse(pako.inflate(data, { to: 'string' }));
      console.log(user);

      res.writeHead(200, {
        'Content-Type': 'application/json'
      });
      res.end(pako.deflate(JSON.stringify({result: 'success'}), { to: 'string' }));
    });
  } else {
    res.writeHead(404);
    res.end(pako.deflate(JSON.stringify({result: 'error'}), { to: 'string' }));
  }
});
```

```
server.listen(3333);
```

JSONP

```
$ npm install jsonp --save
```

```
const jsonp = require('jsonp');

jsonp('http://www.example.com/foo', null, (err, data) => {
  if (err) {
    console.error(err.message);
  } else {
    console.log(data);
  }
});
```