Research Document

Applying Compression on the Web applications

Eindhoven, October 20th, 2021

Table of Contents

Revision History:

Date Changed	Document Version	What was changed
20/10/2021	1	

Introduction

Recently, applications have become more sophisticated both the implementation and architectures have been used, make the applications big in size for downloading and installing, taking time consumption and make uncomfortable to clients who used the applications. For this reason, this research document will focus on the topic of compression, a solution to make the application processing faster, give the satisfaction to the clients.

What is the compression ?

In data compressing, compression is the process of encoding data using fewer bits than the original without losing the data when decompressing. The compression will trade off between the size of data and the time for processing the data. It means, when we compress the file, we will have smaller size of files that suitable to transmit over protocols, but it will take time than usual as to have including total time of compressing and decompressing the file.

How we can apply the compression on web applications

We have used compression on file using the compressing software like WinRAR, Windows built-in compressing software so that we can be able to submit the file to the internet easily. If we observe the website looks like, it is consisting of the code implementation of HTML/CSS/JS and static resources like image, sounds, videos, which can compress, so it means web applications can be compressed without problems. So, we can compress response when clients send a request to server and let the clients decompressing the response automatically.

Performance Tests

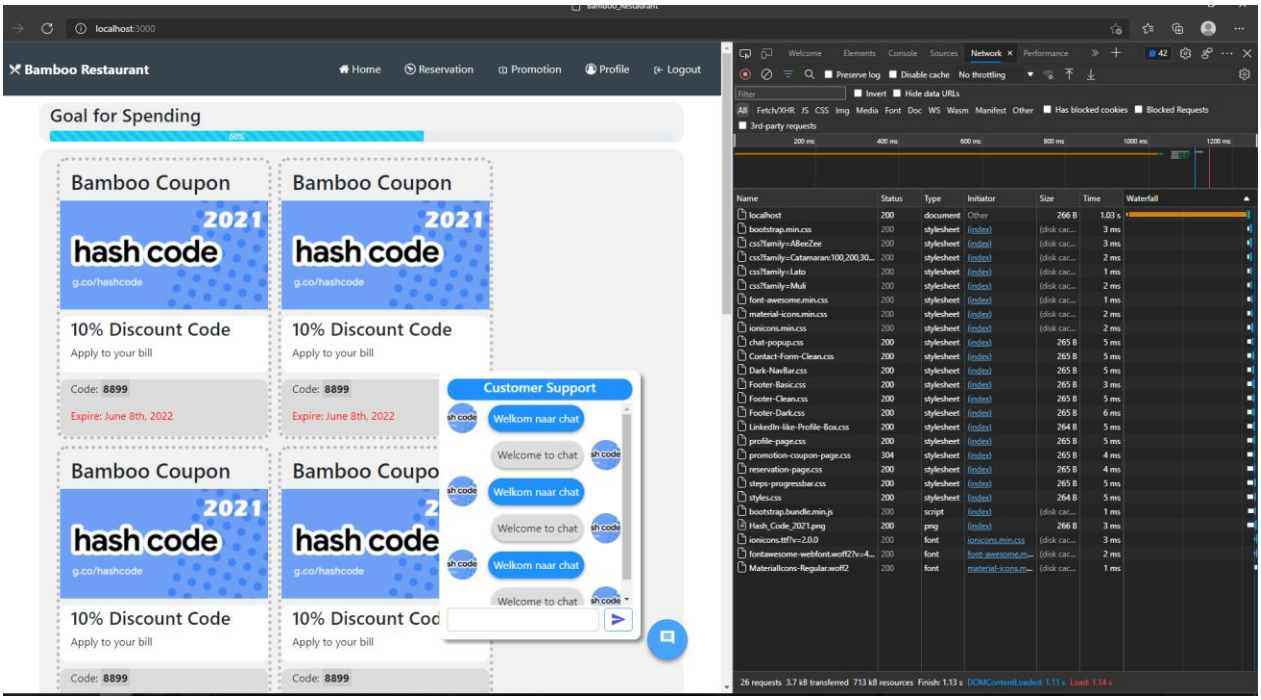
I used NodeJS as the backend framework for the purpose of testing the performance, my assets will be the static website version of Bamboo restaurant.

During the test, I open the Developer Tools panel (pressing F12), to observe the process and time performance of my approach, make a comparison of scenarios.

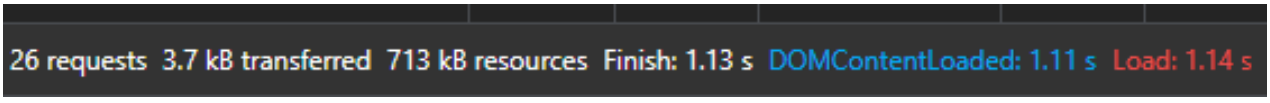
For the first version, without the compression, my backend server implementation only has one Api function call, which return a completely website to my browser.

```
JS index.js
JS index.js > ...
1   const express = require('express');
2   const app = express();
3   const path = require('path');
4   const router = express.Router();
5
6   router.get('/',function(req,res)
7   {
8       res.sendFile(path.join(__dirname+'/index.html'));
9   });
10
11  app.use('/', router);
12  app.use('/assets', express.static(path.join(__dirname, 'assets')));
13  app.listen(process.env.port || 3000);
```

After that, build and run my backend server, make a call to the Api, and I get the website as expected.



As you can see, we also get the detail how my website has been loaded to the browser and the time indication for my website loaded.

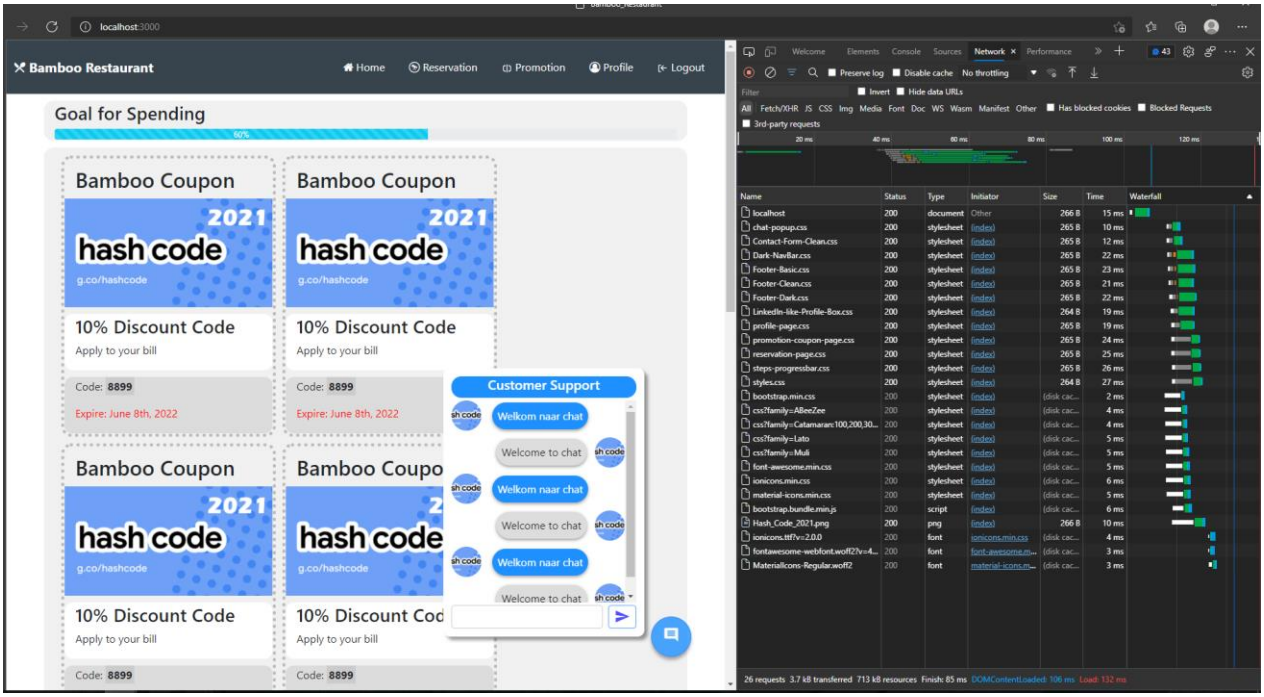


Zoom to the overview information at the bottom of developer's tools panel, we can see that my website takes 1.14 second to finish the loading (or 1140 Ms), which is acceptable to clients as it is loaded below 2 seconds, but can we improve it better.

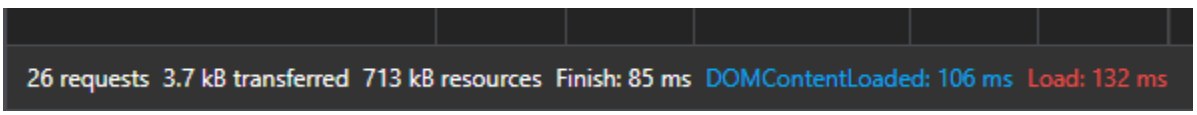
So, for the next test, with the same website, but my backend server implementation will include the compression library and it only add 2 lines of code compared to my initial implementation.

```
JS index.js
JS index.js > ...
1   const express = require('express');
2   const app = express();
3   const path = require('path');
4   const router = express.Router();
5   const compression = require('compression');
6
7   router.get('/',function(req,res)
8   {
9       res.sendFile(path.join(__dirname+'/index.html'));
10  });
11
12  // Compress all HTTP responses
13  app.use(compression());
14  app.use('/', router);
15  app.use('/assets', express.static(path.join(__dirname, 'assets')));
16  app.listen(process.env.port || 3000);
```

After that, I build and run for my updated implementation, make a call to the Api, I still get the website as I expected.



However, the interested part is the time performance of the compressed version, as you can see, my website takes 132 Ms to finish loading the website.



Compared to the previous without compression, the loading time is 90% less than, which is an impressive result.

What are the benefits we get from the Compression ?

From the result of my benchmark tests, I can see the opportunities of applying the compression to the website application by reducing the size of code implementation, means more technologies/libraries can be included. It is also benefited to build a Single Page Application, so at the beginning clients already have a complete application, when transmitting /querying the data, only the necessary data will be considered, reduce the workload for server.

References

1. Rico, V. V. (2020, June 26). *GZIP compression with node.js*. Medium. Retrieved October 25, 2021, from https://medium.com/@victor.valencia.rico/gzip-compression-with-node-js-cc3ed74196f9.
2. Wikimedia Foundation. (2021, October 7). *Data compression*. Wikipedia. Retrieved October 25, 2021, from https://en.wikipedia.org/wiki/Data_compression.