Firebase Hosting

Step 1: Firebase-Tools installing

$npm i -g firebase-tools

# Step 2: Login to firebase

**$firebase login**

Allow or not allow the firebase CLI to collect the errors and other details. Login the screen that appears set up a firebase account before starting this process.

# Step 3: Initialize the hosting project

**$firebase init hosting**

**Select the existing project or create a new project and give unique ID**

Set the project type to **default** or the other kind of project if you are selecting the default project set the **public directory** to store the static web pages. For single page application is set to know if you are not using the Ajax in the code make use you need the single page application.

# Step 4: Set-Up Dynamic Cloud Functions

**$firebase init functions**

To setup the dynamic function select the language typescript or the JavaScript , enable ESLint to make a portable and traceable code. Functions is the place all the dynamic assets are stored.

# Step 5: Install Express

**$npm i express --save**

To install express before that do **$cd functions** after installing **cd ..**

# Step 6: Adding code the code to the index.js in the functions directory

const functions = require('firebase-functions');

exports.helloWorld = functions.https.onRequest((request, response) => {

  functions.logger.info("Hello logs!", {structuredData: true});

  response.send("Hello from Firebase!");

});

Here all the dynamic code should be written in this callback function where https on request only the code gets executed Instead of one callback function we will be using the express.

const functions = require('firebase-functions');

const express= require('express');

const { response } = require('express');

const app=express(); //allows to the listen to the reuest and response

app.get('/timestamp',(request,response)=>{

response.send(`${Date.now()}`)

}); //get rest api to send the date and the response

exports.app = functions.https.onRequest(app); //on request call the function in the parameter list

# Step 7: Add the function to the firebase.json (hookup)

{

  "hosting": {

    "public": "public",

    "ignore": [

      "firebase.json",

      "\*\*/.\*",

      "\*\*/node\_modules/\*\*"

    ],

**"rewrites": [{**

**"source":"/timestamp",**

**"function":"app"**

**}]**

  },

  "functions": {

    "predeploy": [

      "npm --prefix \"$RESOURCE\_DIR\" run lint"

    ]

  }

}

**Rewrites section is added to the firebase.json**

# Step 7: Testing the application

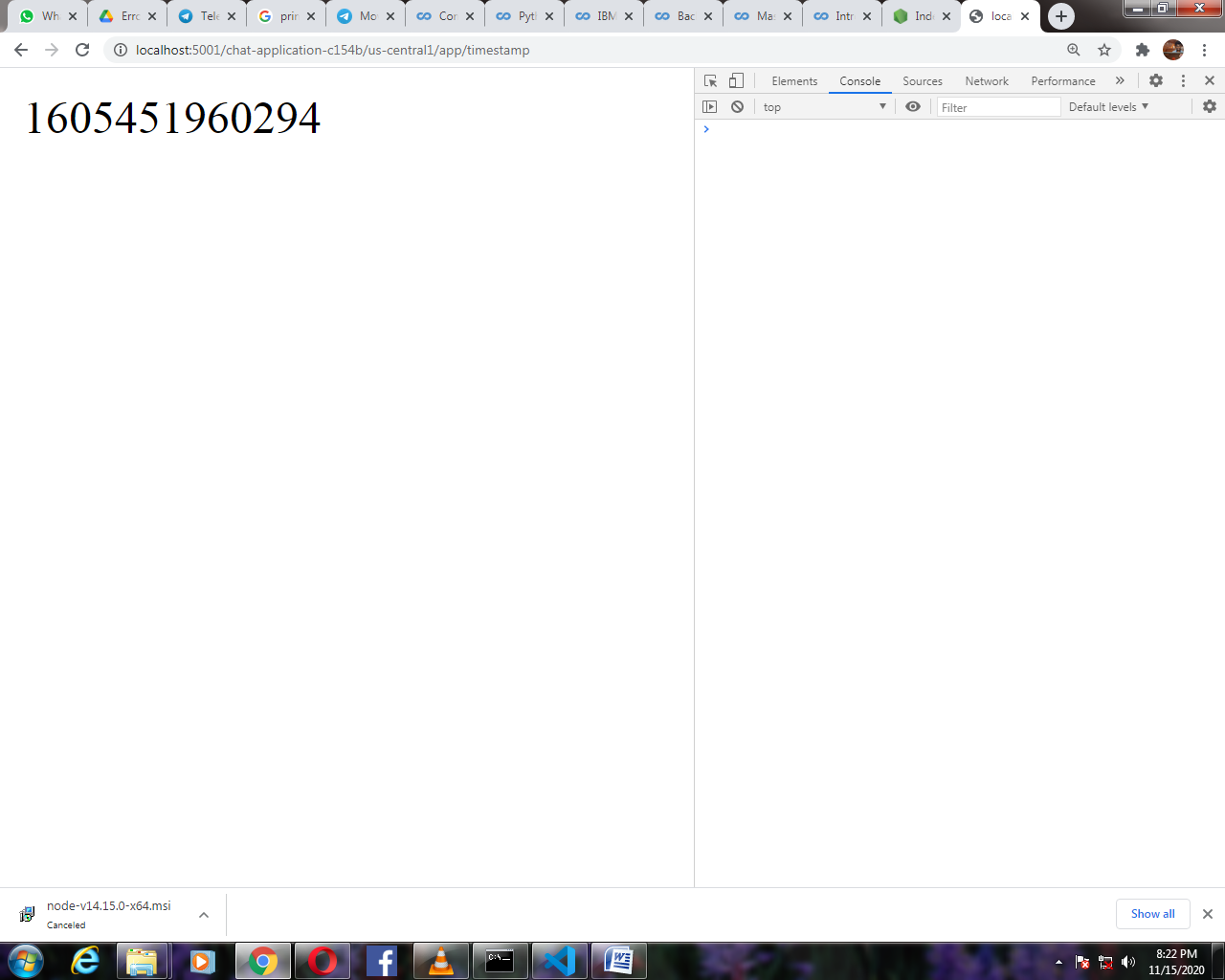
$firebase serve --only functions,hosting

To do the local emulation

+ hosting: Local server: http://localhost:5000

i functions: Watching "C:\Users\User\Documents\SendFiles\functions" for Cloud Functions...

+ functions[app]: http function initialized (http://localhost:5001/chat-application-c154b/us-central1/app).



# Step 7: Adding Caching

app.get('/timestamp-cached',(request,response)=>{

    response.set('Cache-control','public, max-age=300, smax-age=60');

 //seting cahche contal has 3 parts to it 1st part is public is to cahce the content on the server

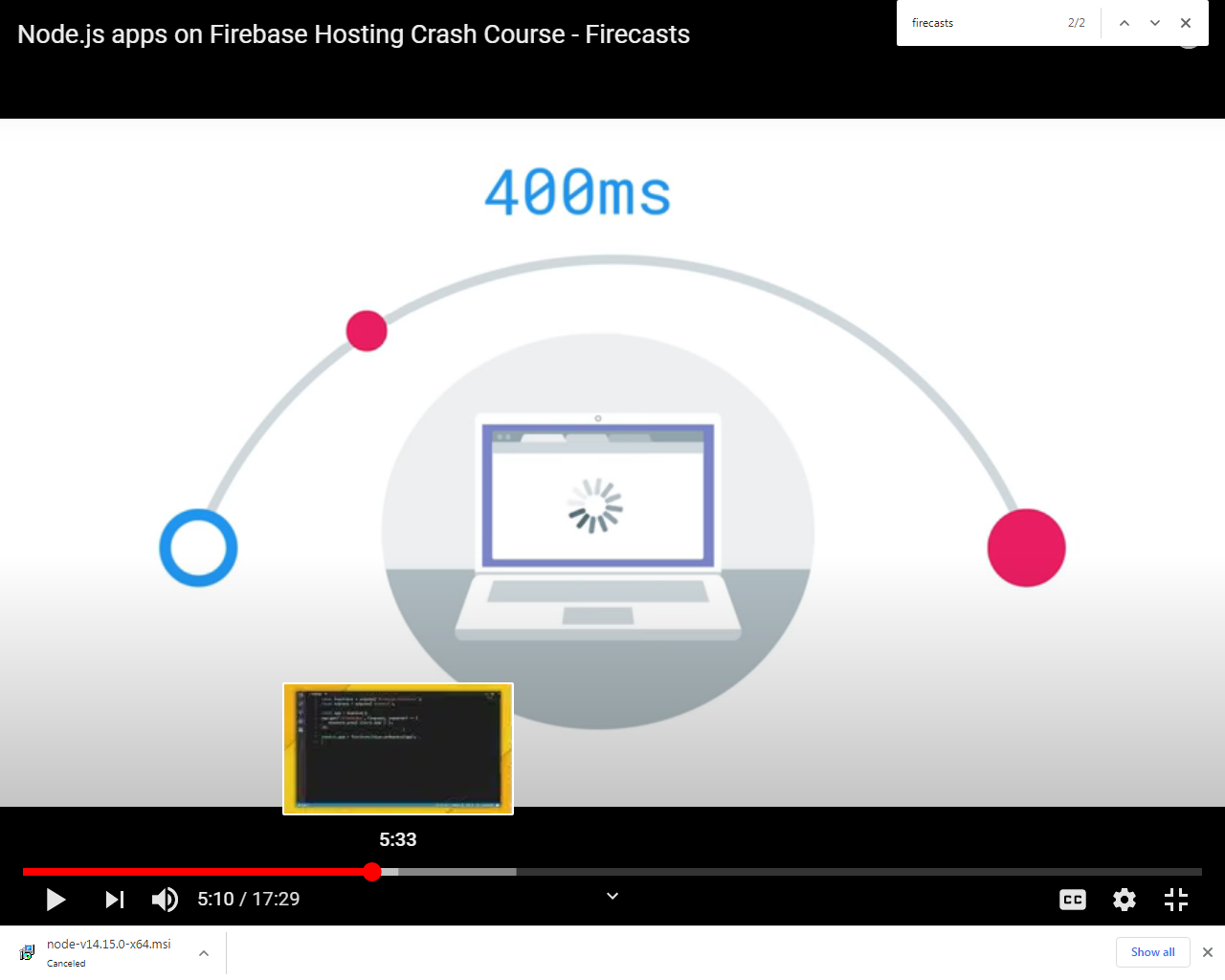
 //if it is private it can only be cached on the user browser

 //max-age how long can we store this value in the user browser , its in ms

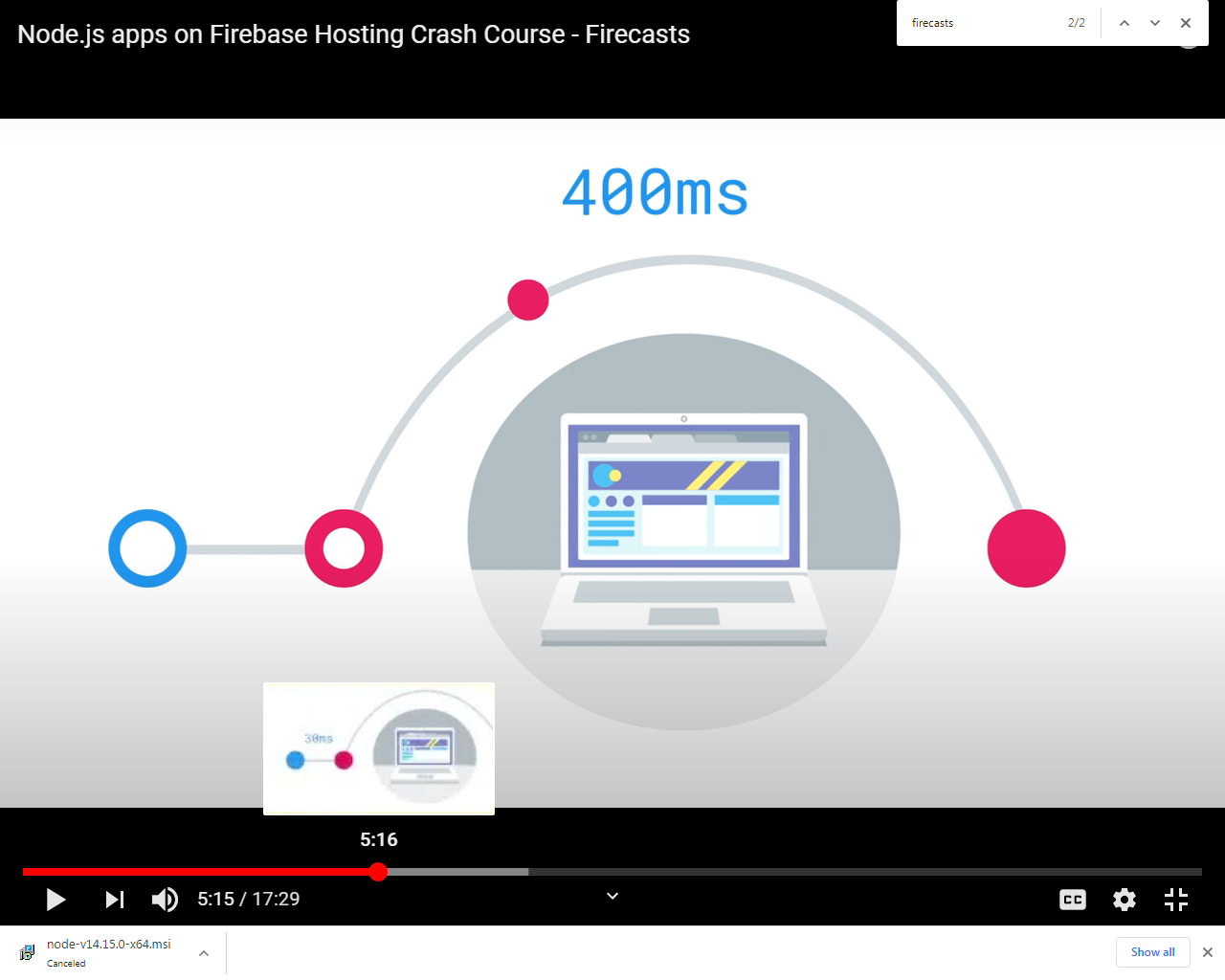
 //smax-age is how long can we store this in the CDN (content delivery networks)

     response.send(`${Date.now()}`);

    }); //cache the conent by setting the cached control header

****

When the user makes a request it goes to the server and server does all the dynamic generation and sends the response to the user is the user is far away from the server it generally takes lot of time to serve the request to avoid this we create a server to cache the results right near the user (Similar to Content Delivery Networks) .Refer above code with comments for more details.



## Change the firebase.json file

 "rewrites": [{

      "source":"\*\*",

      "function":"app"

    }]

# Step 7: Deploying to the firebase

**$firebase deploy**

This will deploy the static assets to the firebase hosting and dynamic code to the cloud functions.

### Error

Error: HTTP Error: 400, Billing account for project '854789552939' is not found. Billing must be enabled for activation of service(s) 'cloudbuild.googleapis.com,containerregistry.googleapis.com' to proceed.

### Solution 1

Change the package.json the node version to 8

"engines": {

    "node": "8"}

If you just want to finish a tutorial, just change **Node version** from 12 to 8 in **package.json**

This will now work, but eventually after February 15, 2021 deployment of Node.js 8 functions will no longer be allowed. So, if you want to use cloud functions after this date you must be on Blaze pay-as-you-go billing plan.

### Solution 2

It works for me:

1. Enable [Billing](https://console.cloud.google.com/billing) for your projects
2. Enable [Cloud Build API](https://console.cloud.google.com/apis/api/cloudbuild.googleapis.com)
3. Enable [Container Registry API](https://console.cloud.google.com/apis/api/containerregistry.googleapis.com)

From Firebase FAQ:

From updates on August 17, 2020, Cloud Functions for Firebase will rely on some additional paid Google services, so You have to add billing, enable Container Registry API and Cloud Storage API to use it.

See more: [Cloud Functions pricing](https://firebase.google.com/support/faq#functions-pricing)

### Solution 3

1st Downgrade **node version from 10 to 8** on **package.json**

"engines": {

"node": "8"

}

1. **firebase deploy**
2. you'll get **+ Deploy complete!** message above that you'll see

**+ functions[helloWorld(us-central1)]: Successful update operation.**

1. this **us-central** is **my project hosted region** in your case may be differ

prefix to this region **[helloWorld** is my exported function name your case it may differ get this information from your terminal by **firebase deploy** command

1. After that **your hosted project link will be**

<https://yourregion-firebaseprojectname.cloudfunctions.net/yourexportedfunctionname>

### Extra Information

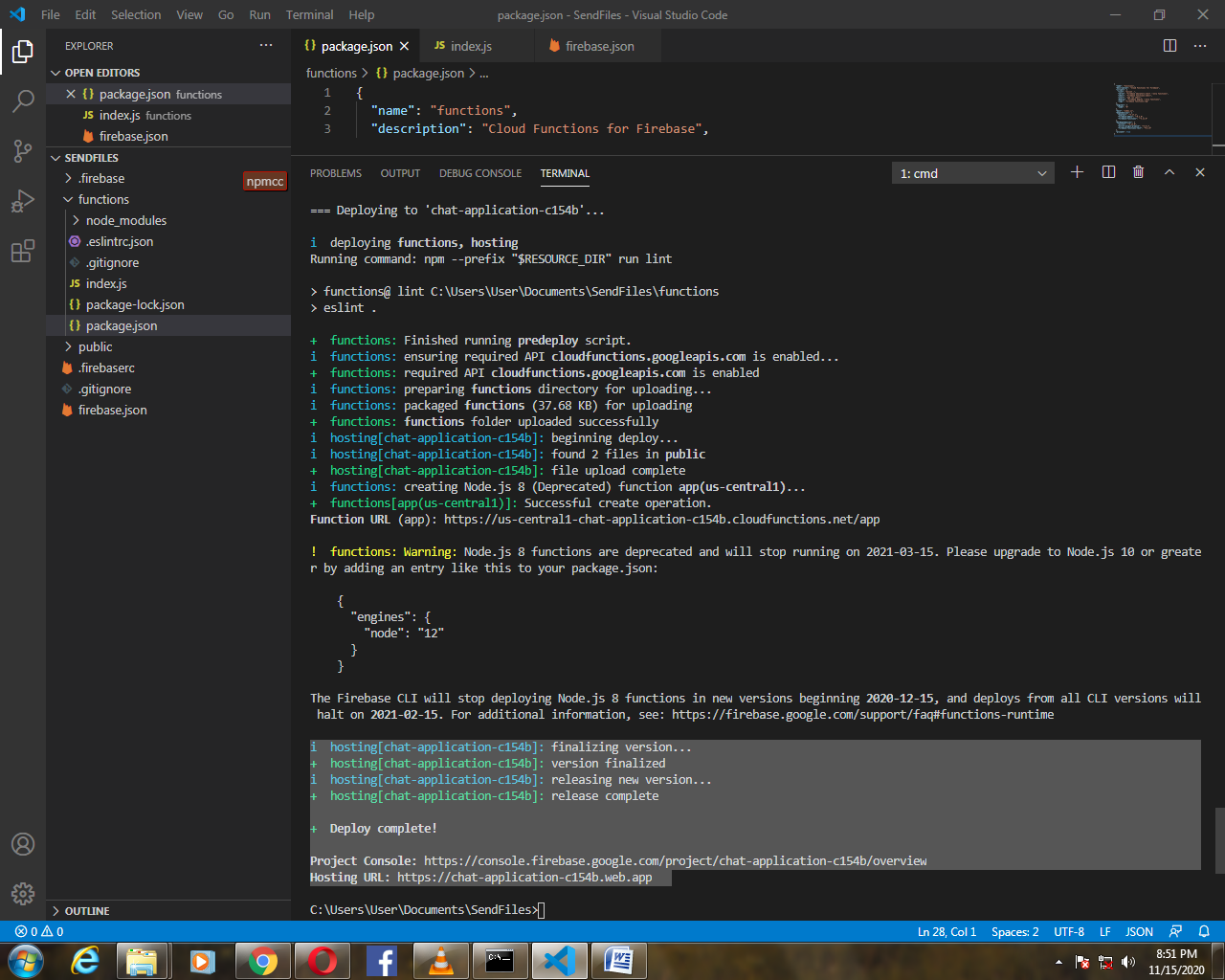
The tutorial you're following is very old. Cloud Functions now requires an active billing account. You will need to add one, even if you don't expect to exceed the free allowance.

See also:

* [Is Function Cloud in Firebase Free or Not (Cloud Functions deployment requires the pay-as-you-go (Blaze) billing plan)](https://stackoverflow.com/questions/62824043/is-function-cloud-in-firebase-free-or-not-cloud-functions-deployment-requires-t)

Consider contacting the author of that article to tell them that they should update their article to be more accurate.

### Solution 1 and 3 works fine go-ahead with it error solved.

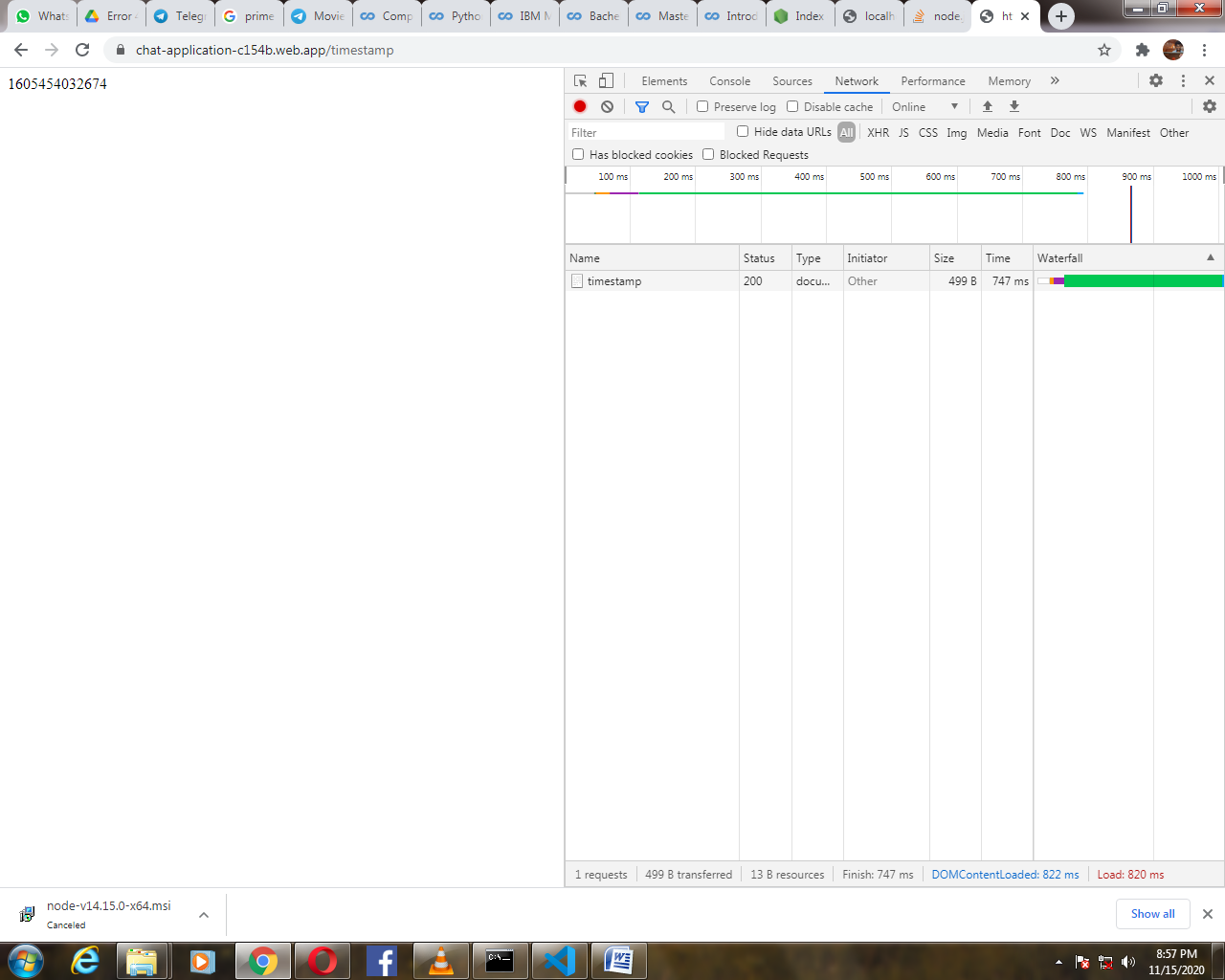


Copy the firebase hosting URL and go to the time stamp page.

Hosting URL: https://chat-application-c154b.web.app

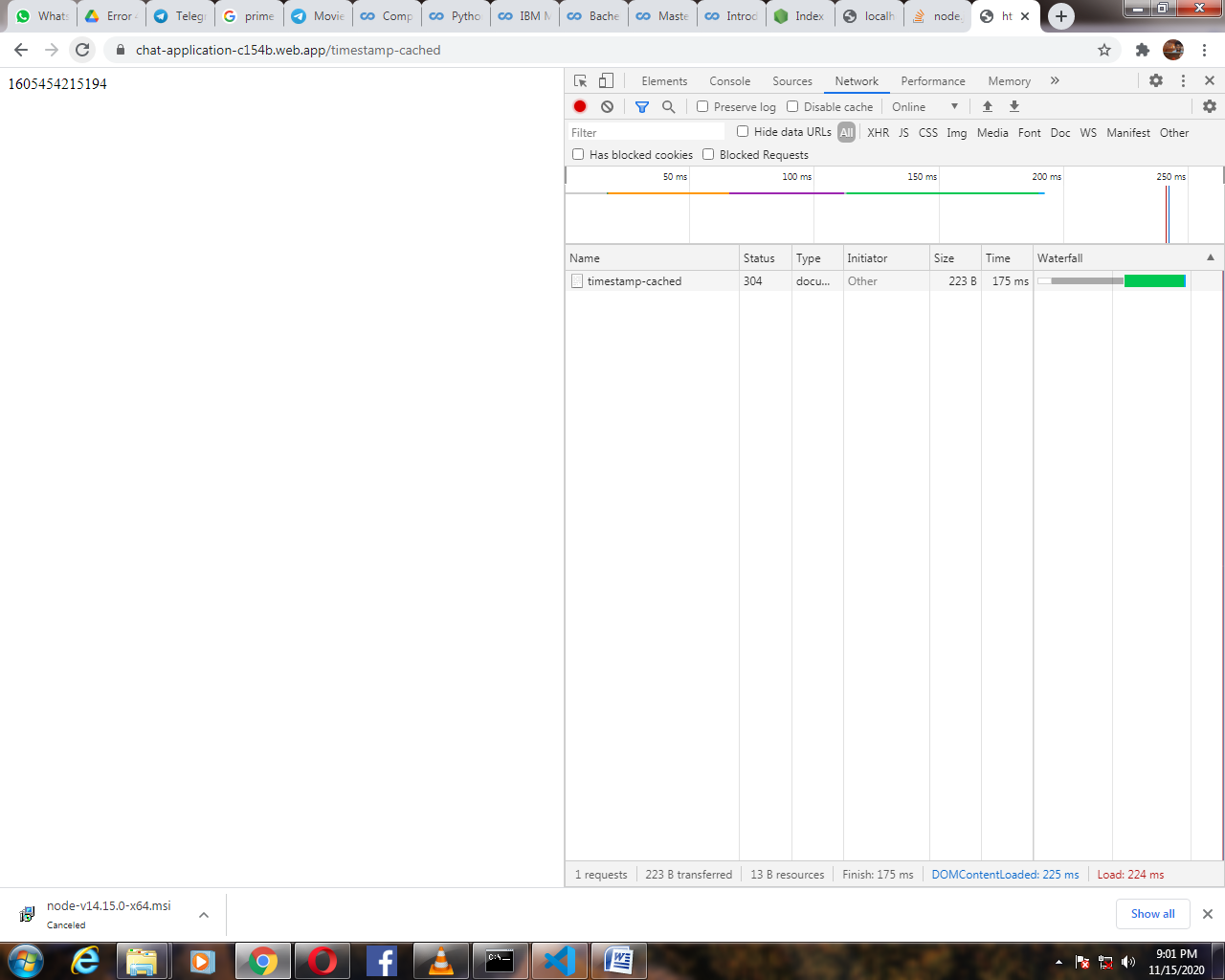
Timestamp Page: https://chat-application-c154b.web.app/timestamp

# Step 8: Let’s check the response time for the cached and un-cached response



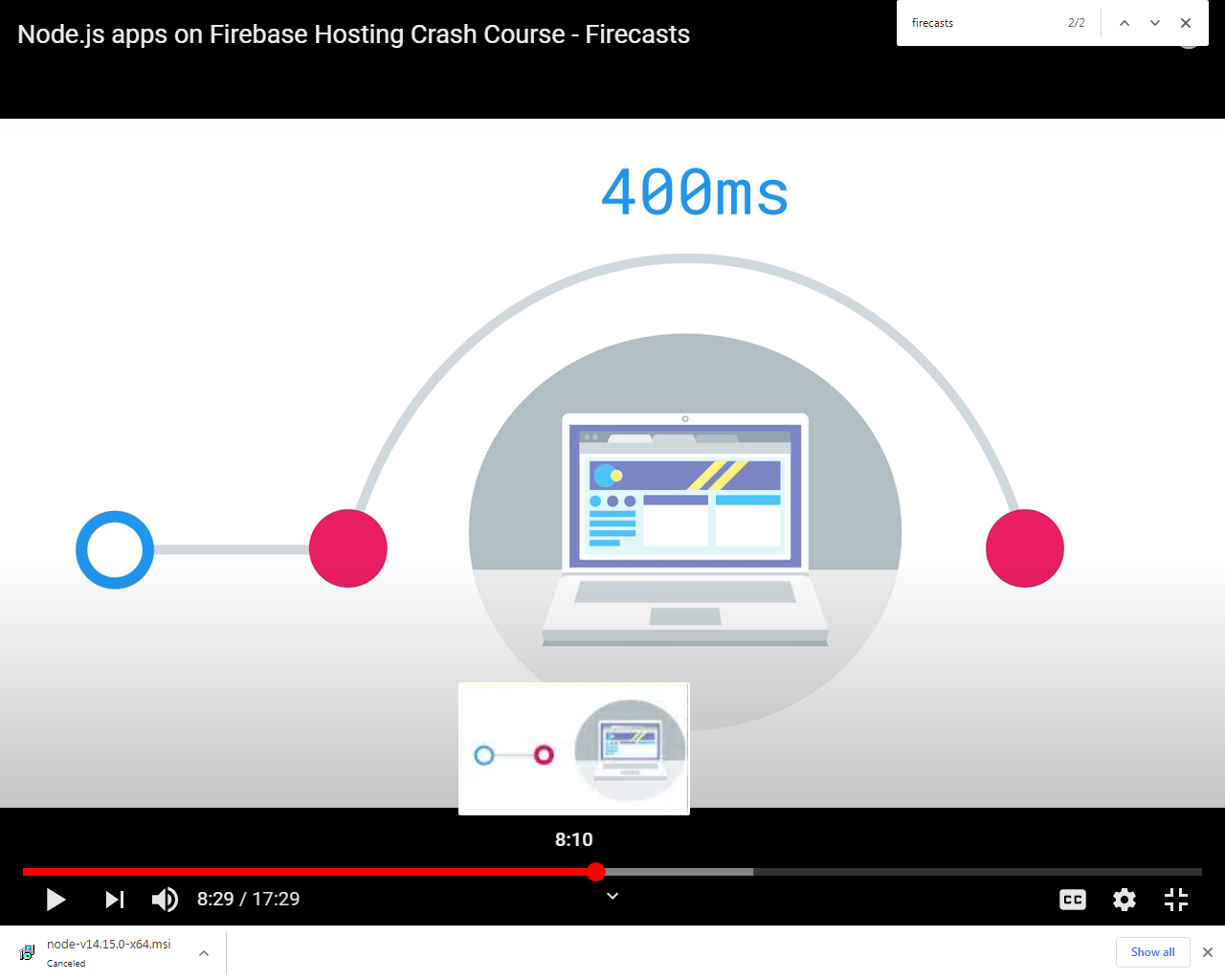
Without Caching the time taken by the server is 747 ms, 701 ms, 778 ms, and 681 ms and soon.

In cached the first request is pretty much the same result but on the second request the request the time is much reduced

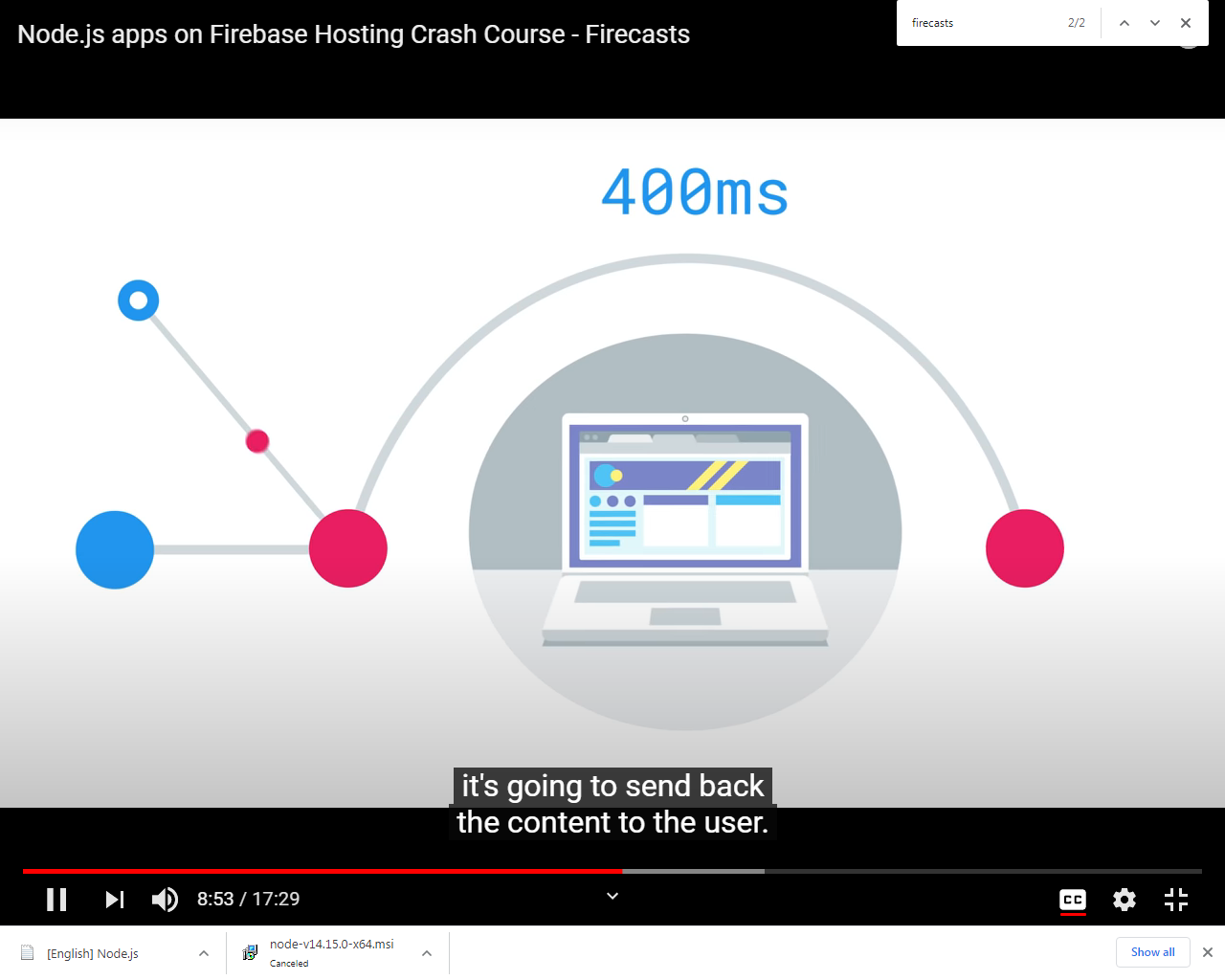


Somewhere around the 100 ms to 175 ms

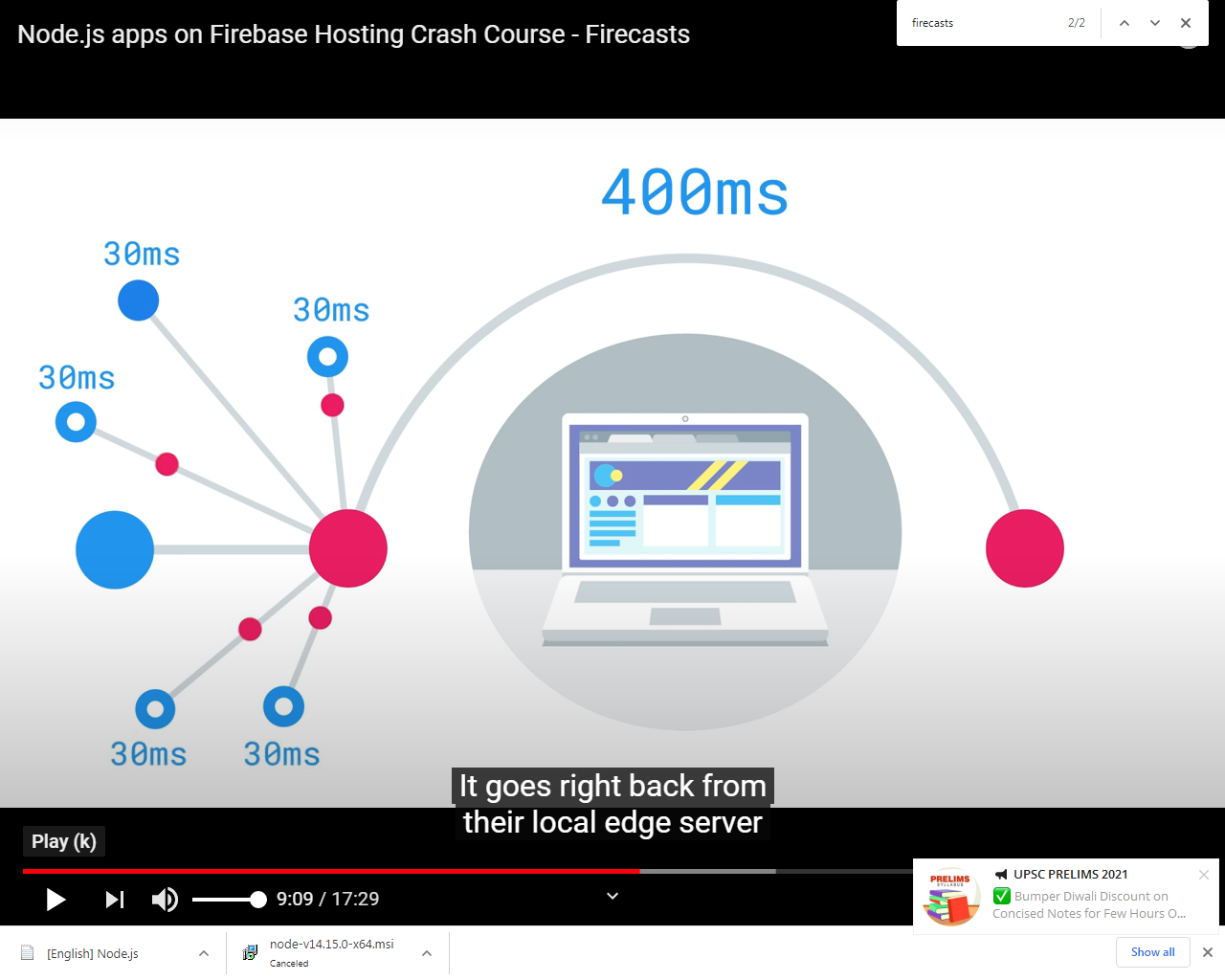
# Who this happened?



we have a user who's going to make a request for our website but with a content delivery network it's actually going to stop at this edge server which is close to the user and the edge server is going to check to see if the content is in its cache if it's not in cache it's going to forward the request off to the origin server the origin server is going to do the dynamic content generation and then it's going to send that back to the edge server the edge server is going to cache this content which will be controlled by the cache control header that you set it'll then send that content back to the user now let's say another user in that same area makes a request for the web site that request will go out to the edge server and the edge server will recognize that it has this content in the cache so rather than going out to the origin and doing the new dynamic generation of a page it's going to send back the content to the user and that response time is going to be much much faster because it's local to the user.



what's amazing about this is that other users in this area also can make requests for this content and it doesn't have to go out to the origin server it goes right back from their local edge server for a really fast page load now after the cache expires the whole process will begin again where we go out to the edge server the edge server recognizes that the content has expired so it goes out to the origin ,origin dynamically generates sends it back caches it according to your cache control headers and then back to the user where it's cached.



# Server Templating : Using the View Engine in Express

Sever templating allows us to use whatever language you without worrying about the actual implementation.

## Creating the server template

1. In the folder called views in functions directory dynamic content generatrion
2. Create index.hbs inside views

An **HBS file** is a template **file** created by Handlebars, a web template system. It contains a template written in HTML code and embedded with Handlebars expressions.

[https://fileinfo.com/extension/hbs]

1. Added the html to the .hbs files

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>True Facts</title>

</head>

<body>

    <h1>True Facts</h1>

    <ul>

        {{#each facts}}

        <li>{{text}}</li>

        {{/each}}

    </ul>

</body>

</html>

In the above code the template is created that has the True facts as the h1 and iterates in each facts list all the text.

## To make this server template work install the handlebar,consolidate dependency to setup view engine.

1. cd functions
2. npm i handlebars consolidate --save

## ERROR

npm ERR! cb() never called!

npm ERR! This is an error with npm itself. Please report this error at:

npm ERR! <https://npm.community>

npm ERR! A complete log of this run can be found in:

npm ERR! C:\Users\User\AppData\Roaming\npm-cache\\_logs\2020-11-15T16\_16\_23\_026Z-debug.log

## Solution 1:

Hey y’all!

tl;dr try this: $ npx npmc@latest install

I’ve released a new npm canary with some patches that will stop dropping cb never called errors when certain error types pop up. Note that you may very well still get an error, but it should be more informative now, and I assume it’s going to be the ENFILE issue in most cases. I have a different idea for those, so if you confirm that, I can try the new thing and release a new canary.

Thanks y’all for your patience, I assure you this bug has been as exasperating for me as I imagine as it’s been for you. Kudos to [**@iarna**](https://npm.community/u/iarna) for finally managing to reproduce it on her own server, which allowed her to do deeper debugging and come up with a patch.

Anyway, please let me know if you still get the cb never called with the canary!

## Solution 2: Antivirus Avast was blocking a file was an issue.

Solution 2 helped me at last but tried both of them may be solution 1 also because its installation only.

## Index.js

const engines=require('consolidate'); //import consolidate

app.engine('hbs',engines.handlebars);//creating the engine

app.set('views','./views');//setting the views folder

app.set('view engine','hbs');//using the engine created

## Rendering the page

app.get('/',(request,response)=>{

    response.set('Cache-control','public, max-age=300, smax-age=60');

 //seting cahche contal has 3 parts to it 1st part is public is to cahce the content on the server

 //if it is private it can only be cached on the user browser

 //max-age how long can we store this value in the user browser , its in ms

 //smax-age is how long can we store this in the CDN (content delivery networks)

     response.render('index',{ facts }); //rendering the index page with some facts data that is retrived

    }); //cache the conent by setting the cached control header

## Firebase admin getting data from database

const firebase=require('firebase-admin')

const firebaseApp= firebase.initializeApp(

    functions.config().firebase

)

function getFacts()

{

    const ref= firebaseApp.database().ref('facts'); //geting the refernce

    return ref.once('value').then(snap=> snap.val());

    //once for the getting data once since it returns the promise get the snapshot and unwrap the value

}

const engines=require('consolidate'); //import consolidate

app.engine('hbs',engines.handlebars);//creating the engine

app.set('views','./views');//setting the views folder

app.set('view engine','hbs');//using the engine created

app.get('/',(request,response)=>{

    response.set('Cache-control','public, max-age=300, smax-age=60');

 //seting cahche contal has 3 parts to it 1st part is public is to cahce the content on the server

 //if it is private it can only be cached on the user browser

 //max-age how long can we store this value in the user browser , its in ms

 //smax-age is how long can we store this in the CDN (content delivery networks)

     getFacts().then(facts=>{

        response.render('index',{ facts }); //rendering the index page with some facts data that is retrived

     })

    }); //cache the content by setting the cached control header

## Testing

$cd ..

$firebase serve --only functions,hosting

The static page is to be deleted because it is routed instead of dynamic pages

Then deploy the website

$firebase deploy

**ERROR**

**41:6 error Expected catch() or return promise/catch-or-return**

**41:22 error Each then() should return a value or throw promise/always-return**

  getFacts().then(facts=>{

        response.render('index',{ facts }); //rendering the index page with some facts data that is retrived

     }).catch(e=>{

         console.log(e);

         response.sendStatus(500);

     })

**Each then() should return a value or throw promise/always-return**

{if(snap.exists)

        {snap.val()}

        else{ throw new Error("Profile doesnt exists")

## Solution

Adding Promise.resolve(data) solved the problem

# Adding the style sheet

Style sheet should be in the static assets no server is needed just the style to the page so it is in the public folder

body {

    background-color: #C6E0F8;

    font-family: 'Roboto', Arial, sans-serif;

    width: 960px;

    margin: 0 auto;

    padding: 0;

}

ul {

    padding: 0;

    margin: 0;

}

ul li {

    font-size: 18px;

    list-style-type: none;

    height: 48px;

    width: 100%;

    color: rgba(0,0,0, 0.63);

    border-bottom: 1px dashed rgba(0,0,0, 0.42);

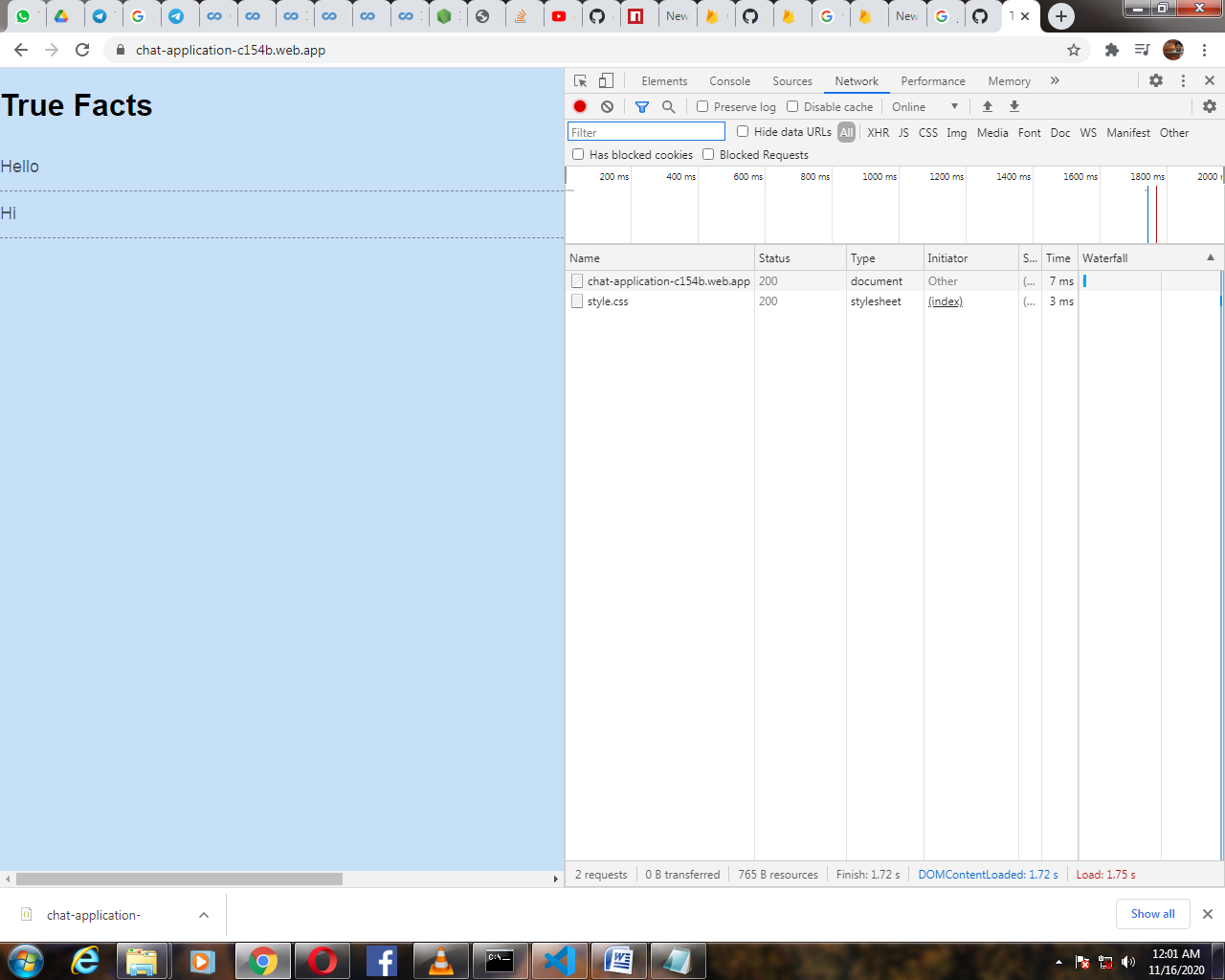
    display: flex;

    align-items: center;

}

## Test the code:

$firebase deploy



## Render the JSON Data

    app.get('/facts.json',(request,response)=>{

        response.set('Cache-control','public, max-age=300, smax-age=60');

     //seting cahche contal has 3 parts to it 1st part is public is to cahce the content on the server

     //if it is private it can only be cached on the user browser

     //max-age how long can we store this value in the user browser , its in ms

     //smax-age is how long can we store this in the CDN (content delivery networks)

         getFacts().then(facts=>{

            response.json(facts);

            return Promise.resolve(facts); //rendering the index page with some facts data that is retrived

         }).catch(e=>{

             console.log(e);

             response.sendStatus(500);

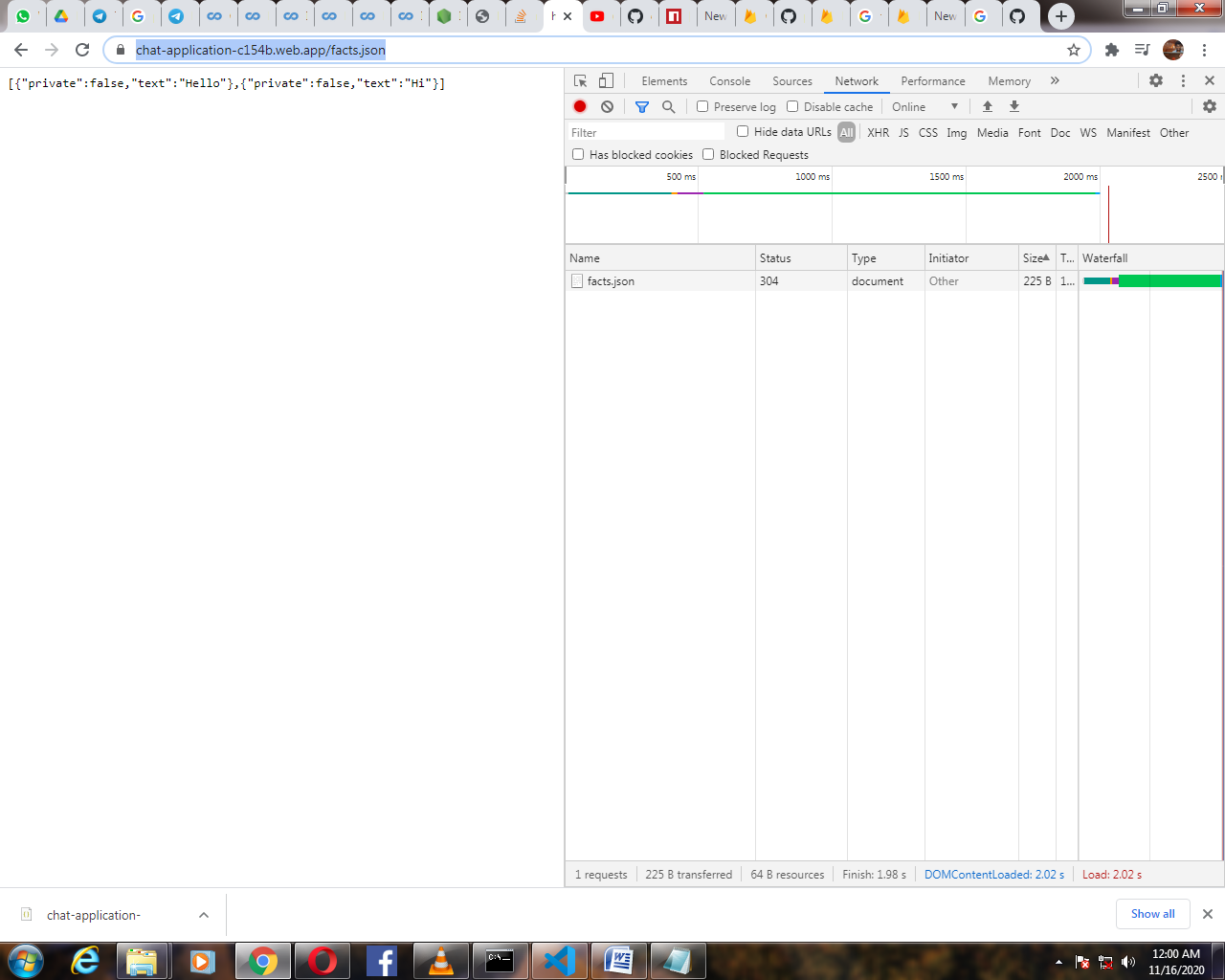
         })

        }); //cache the conent by setting the cached control header

## Test the code:

$firebase deploy

<https://chat-application-c154b.web.app/facts.json>



# References:

1. <https://www.youtube.com/watch?v=mmmaeHBCTOw&list=PL4cUxeGkcC9he0kHAyiyr3dDO2xw0NWoP> **[**Videos from **The Net Ninja]**
2. <https://www.youtube.com/watch?v=jsRVHeQd5kU&list=PLl-K7zZEsYLkzst3HMqIgChIImEvPDaIP> [Firebase]
3. <https://www.youtube.com/watch?v=TB9g_Oe3kO8&list=PLl-K7zZEsYLnJVX_0zbKytptZGugPIbJR> [**Firecasts** Extra]
4. <https://www.youtube.com/watch?v=LOeioOKUKI8&t=540s> [**Main video only Node js] this video is followed in the tutorial.**
5. <https://github.com/zaszab/firebase-nodejs> [solved few errors]
6. <https://github.com/firebase/quickstart-nodejs/> [other version of node]

# Learning from this:

1. Can host node apps in firebase hosting
2. Take the generated results and store it into a CDN for a huge proof of benefits
3. What about JavaScript frameworks?
4. With JS framework all of the contents and rendering are held up in JS but modern JS have tooling around server side rendering so we can take that same client side app render it out on the server side for a fast response per request
5. Check out server side rendering JS Frameworks.

# Running the existing application

mkdir firebase-app

cd firebase-app

firebase init hosting

firebase init functions

cd functions

npm i express --save

cd ..

code .

Add all the code that you have

After modification you can test it on [http://localhost:5000](http://localhost:5000/) by running this in console:

firebase serve --only functions,hosting

For deploy type:

firebase deploy

And you can check it on  
[https://YOURPROJECTNAME.firebaseapp.com](https://yourprojectname.firebaseapp.com/)