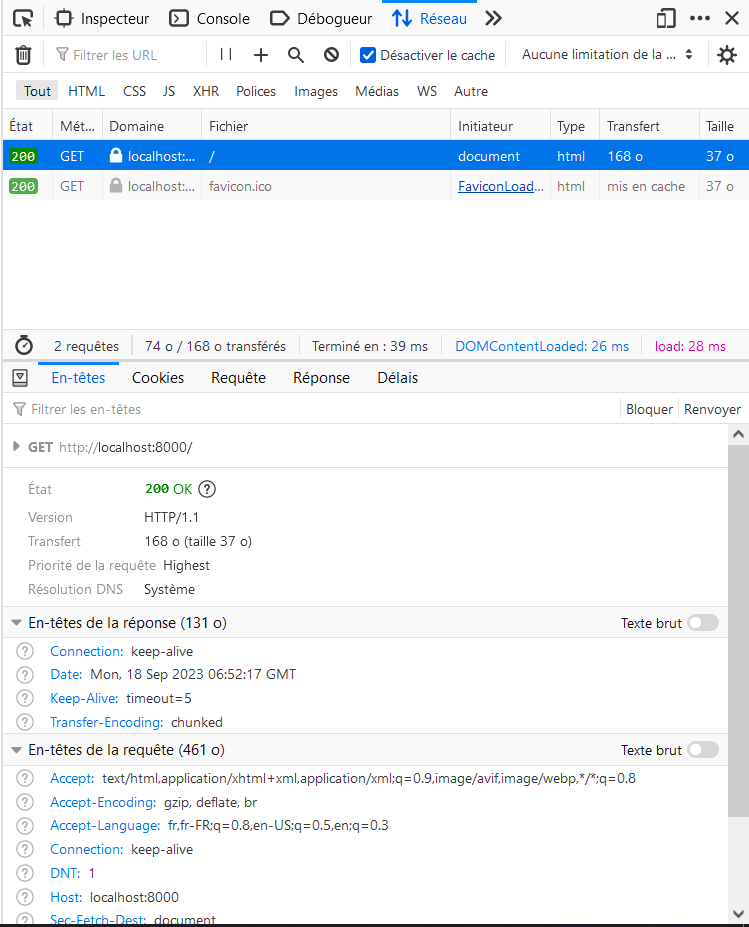
Développement Web – TP5

Tutoriel HTTP/Express Node.js

**Partie 1 : serveur HTTP natif Node.js**

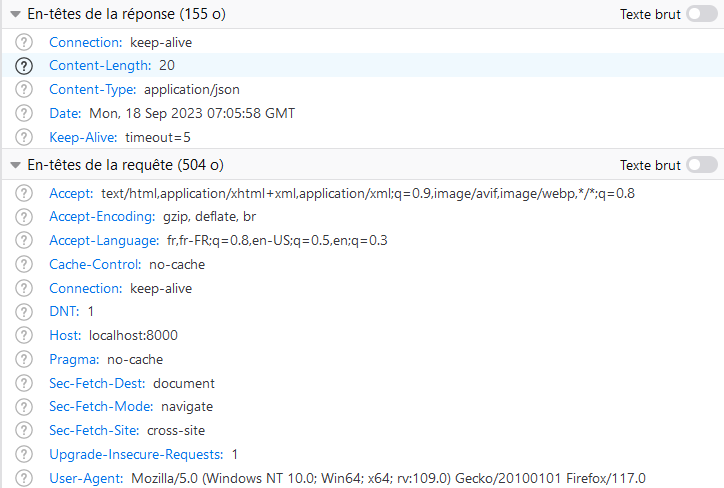
**Installation /**

**Question 1.1** donner la liste des en-têtes de la réponse HTTP du serveur.



**Servir différents types de contenus /**

**Question 1.2** donner la liste des en-têtes qui ont changé depuis la version précédente.



Nous pouvons remarquer que cette fois-ci il y a des entêtes en plus du côté de la réponse, ‘Content-Length’ et ‘Content-Type’, mais aussi un entête en moins ‘Transfer-Encoding’.

Il y a également le favicon.ico qui est maintenant bloqué.

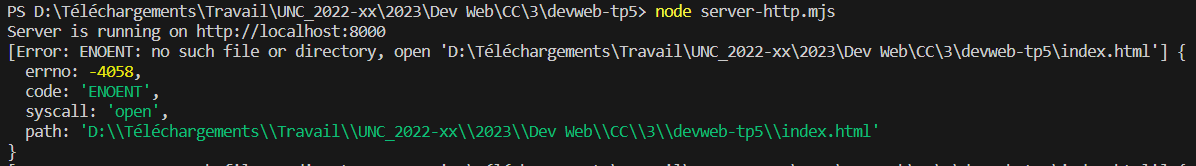
**Question 1.3** que contient la réponse reçue par le client ?

FICHIER index.html EXISTANT /

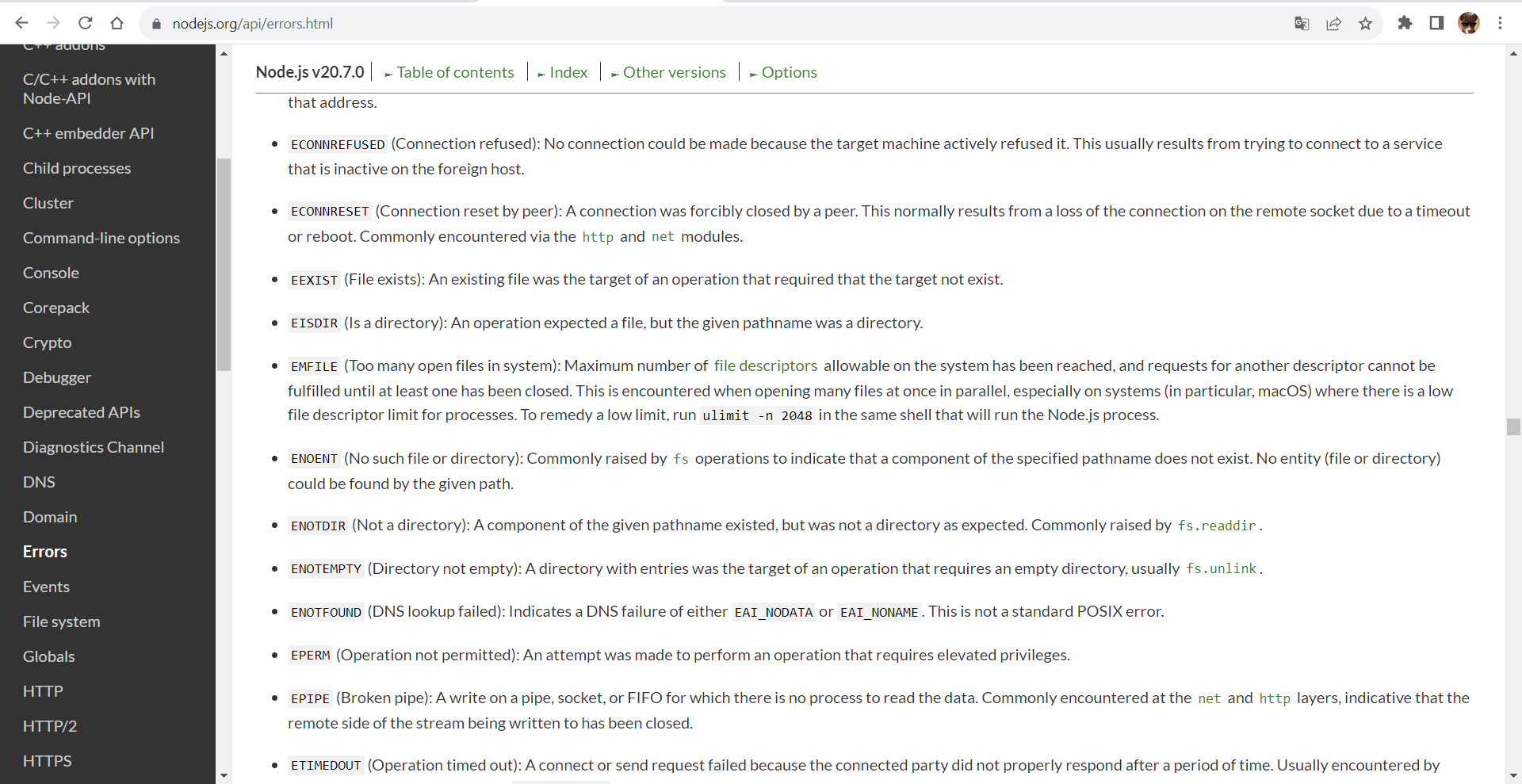
Le client se retrouve face à un chargement de page qui n’aboutira malheureusement jamais puisque la fonction *requestListener()* va lire un fichier *index.html* qui n’existe pas. Une erreur 500 a donc lieu.

**Question 1.4 (1)** quelle est l’erreur affichée dans la console ? Retrouver sur <https://nodejs.org/api> le code d’erreur affiché.

La console affiche une erreur *-4058* indiquant que le fichier est introuvable.

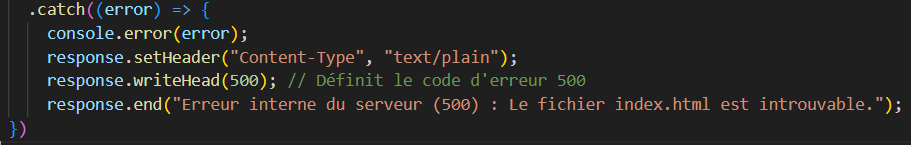


Voici la description du code d’erreur *ENOENT* donnée par le site *nodejs.org*

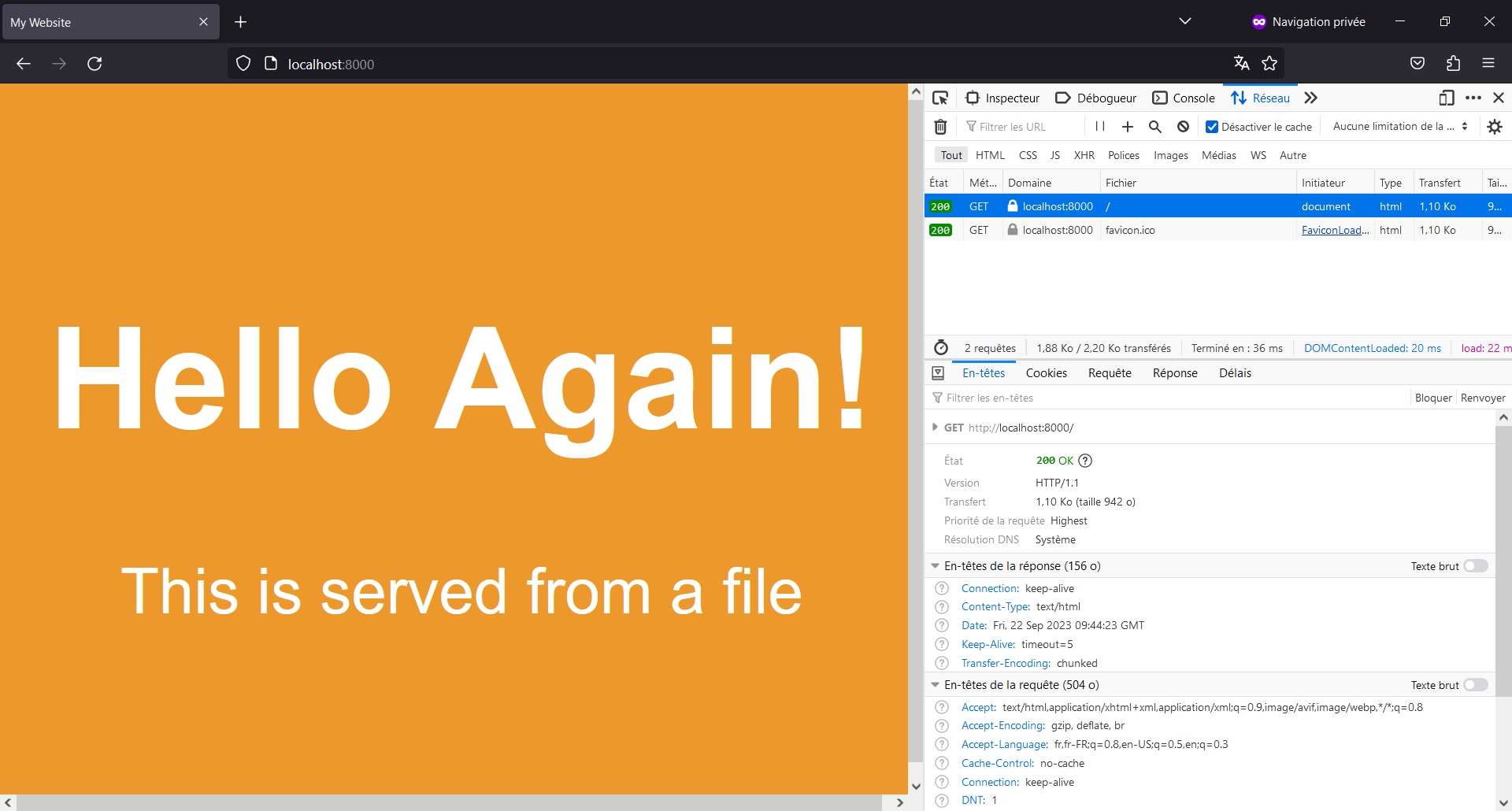


**Question 1.4 (2)** Modifier la fonction requestListener() précédente pour que le client recoive une erreur 500 si index.html est introuvable en remplacant le callback de la méthode Promise.catch().

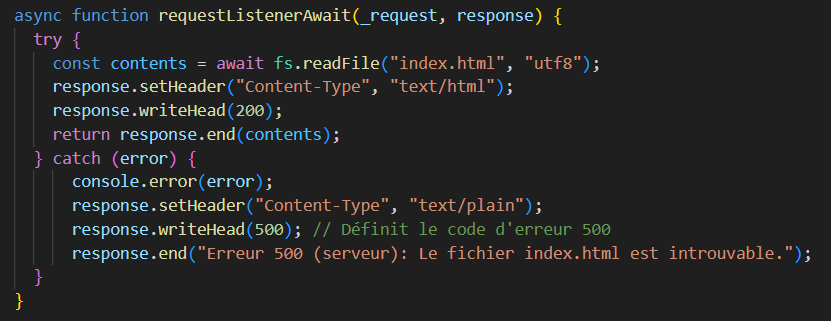
Modification de la méthode *Promise.catch()* de la fonction *requestListener()*:



**Question 1.4 (3)** Maintenant, renommer le fichier \_\_index.html en index.html et tester à nouveau.



**Question 1.5** donner le code de requestListener() modifié avec gestion d’erreur en async/await.



**Mode développement /**

**Question 1.6** indiquer ce que cette commande a modifié dans votre projet.

• La commande *npm install cross-env –save a* ajouté dans le répertoire un dossier *node\_modules* contenant toutes les dépendances du package *cross-env* et a modifié le fichier *package.json* pour y ajouter *cross-env* comme dépendance du projet.

• La commande *npm install nodemon --save-dev* fait la même chose mais pour le package *nodemon*.

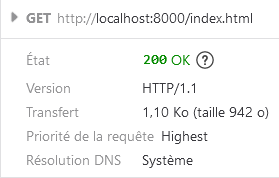
**Question 1.7** quelles sont les différences entre les scripts http-dev et http-prod ?

Le script *http-dev* permet un rechargement automatique de la page suite à une modification du fichier .mjs alors que le script *http-prod* non. De plus, pour *http-dev, NODE\_ENV* = *development* alors que pour *http-prod, NODE\_ENV = production*.

Gestion manuelle des routes /

**Question 1.8** donner les codes HTTP reçus par votre navigateur pour chacune des quatre pages précédentes.

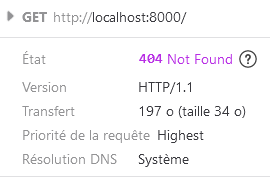
• [*http://localhost:8000/index.html*](http://localhost:8000/index.html) : code 200



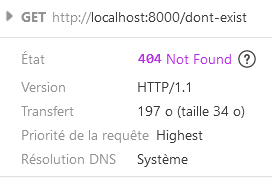
• [*http://localhost:8000/random.html*](http://localhost:8000/random.html) : code 200



• [*http://localhost:8000/*](http://localhost:8000/) : code 404



• [*http://localhost:8000/dont-exist*](http://localhost:8000/dont-exist) : code 404



Modification de la fonction *requestListener()* pour pouvoir spécifier une quantité de nombres aléatoires à générer à partir de l’URL qui sera de type */random/:nb* où *:nb* est le nombre d’entiers à générer sans modifier le traitement pour */index.html* et */*:



**Partie 2 : framework Express**

**Création du serveur /**

**Question 2.1** donner les URL des documentations de chacun des modules installés par la commande précédente.

Voici les modules installés avec la commande *npm install --save express http-errors loglevel morgan* (ainsi que l’URL de leur documentation) :

• Express (<https://expressjs.com/>)

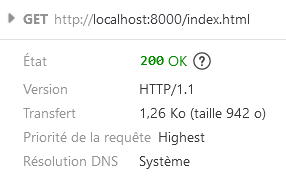
• http-errors (<https://www.npmjs.com/package/http-errors>)

• loglevel (<https://www.npmjs.com/package/loglevel>)

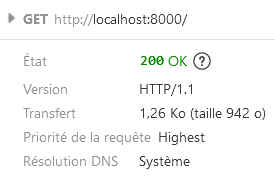
• Morgan (<https://www.npmjs.com/package/morgan>)

**Question 2.2** vérifier que les trois routes fonctionnent.

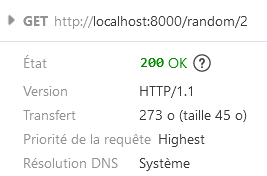
• */index.html*:



• */*:



• */random/2*:



**Question 2.3** lister les en-têtes des réponses fournies par Express. Lesquelles sont nouvelles par rapport au serveur HTTP ?



Les nouvelles en-têtes des réponses fournies par Express sont : *Accept-Ranges, Cache-Control, ETag* et *Last-Modified*.

**Question 2.4** quand l’événement listening est-il déclenché ?

L’événement *listening* est déclenché dès que le serveur est lancé, soit dès qu’il commence à écouter les requêtes entrantes.

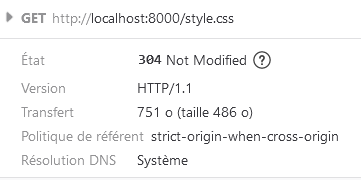
**Ajout de middlewares /**

**Question 2.5** indiquer quelle est l’option (activée par défaut) qui redirige / vers /index.html ?

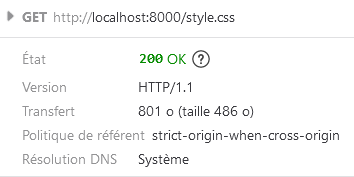
Lors de l’utilisation de *express.static*, l’option *fallthrough* activée par défaut permet de rediriger */* vers */index.html* si celui existe dans le répertoire donné en paramètre de *express.static* .

**Question 2.6 (1)** visiter la page d’accueil puis rafraichir (Ctrl+R) et ensuite **forcer** le rafraichissement (Ctrl+Shift+R). Quels sont les codes HTTP sur le fichier style.css ? Justifier.

(Ne pas oublier de désactiver le cache pour cette question)



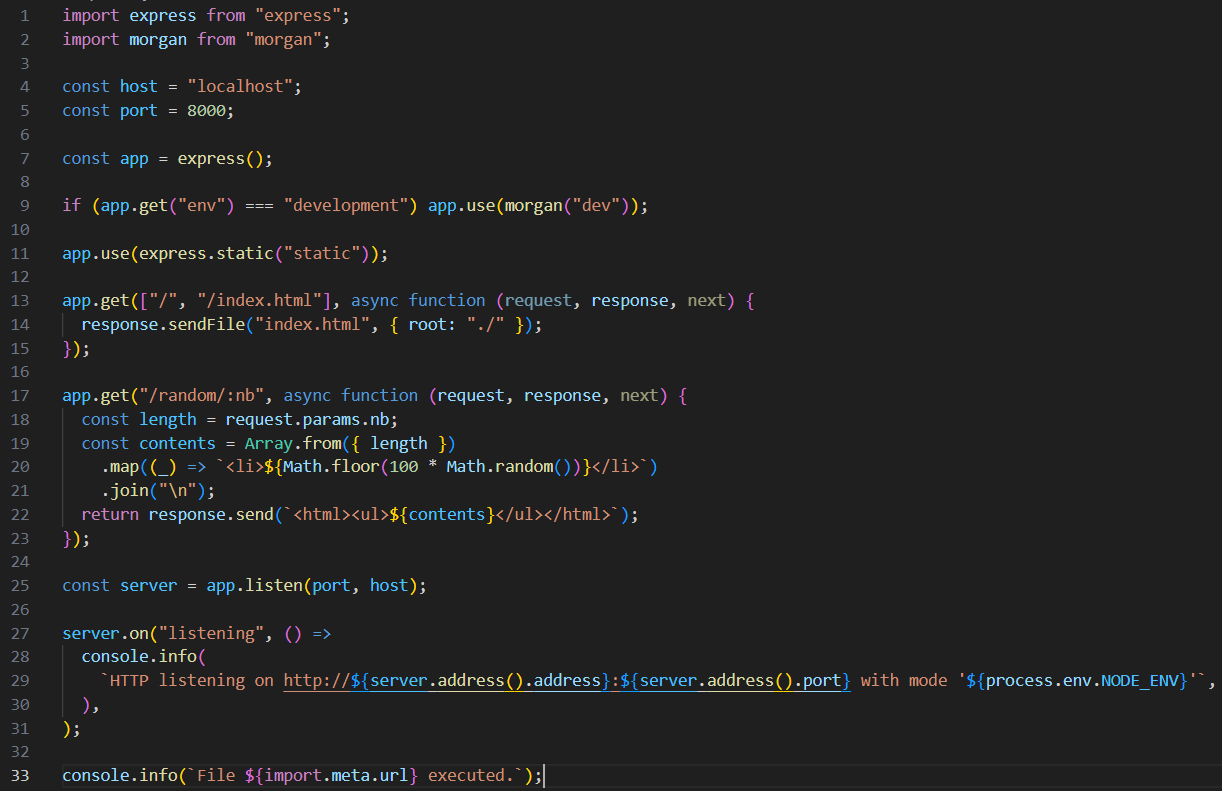
• Lors du rafraichissement simple (Ctrl+R), on obtient un code 304 indiquant que le fichier style.css n’a pas été modifié depuis la dernière visite de la page.



• Lors du rafraichissement forcé (Ctrl+Shift+R), on ignore le cache et on obtient cette fois-ci un code 200 indiquant que le fichier style.css a bien été reçue du serveur.

**Question 2.6** **(2)** Ajouter la ligne if (app.get("env") === "development") app.use(morgan("dev")); au bon endroit dans server-express.mjs pour activer le middleware Morgan.

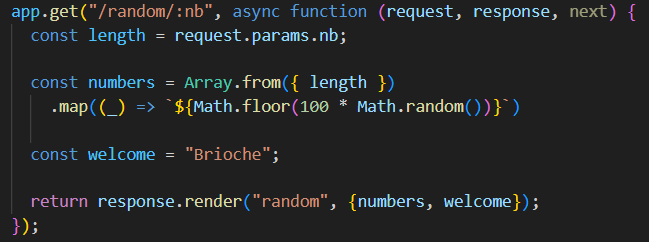
Juste après la création de *app* :



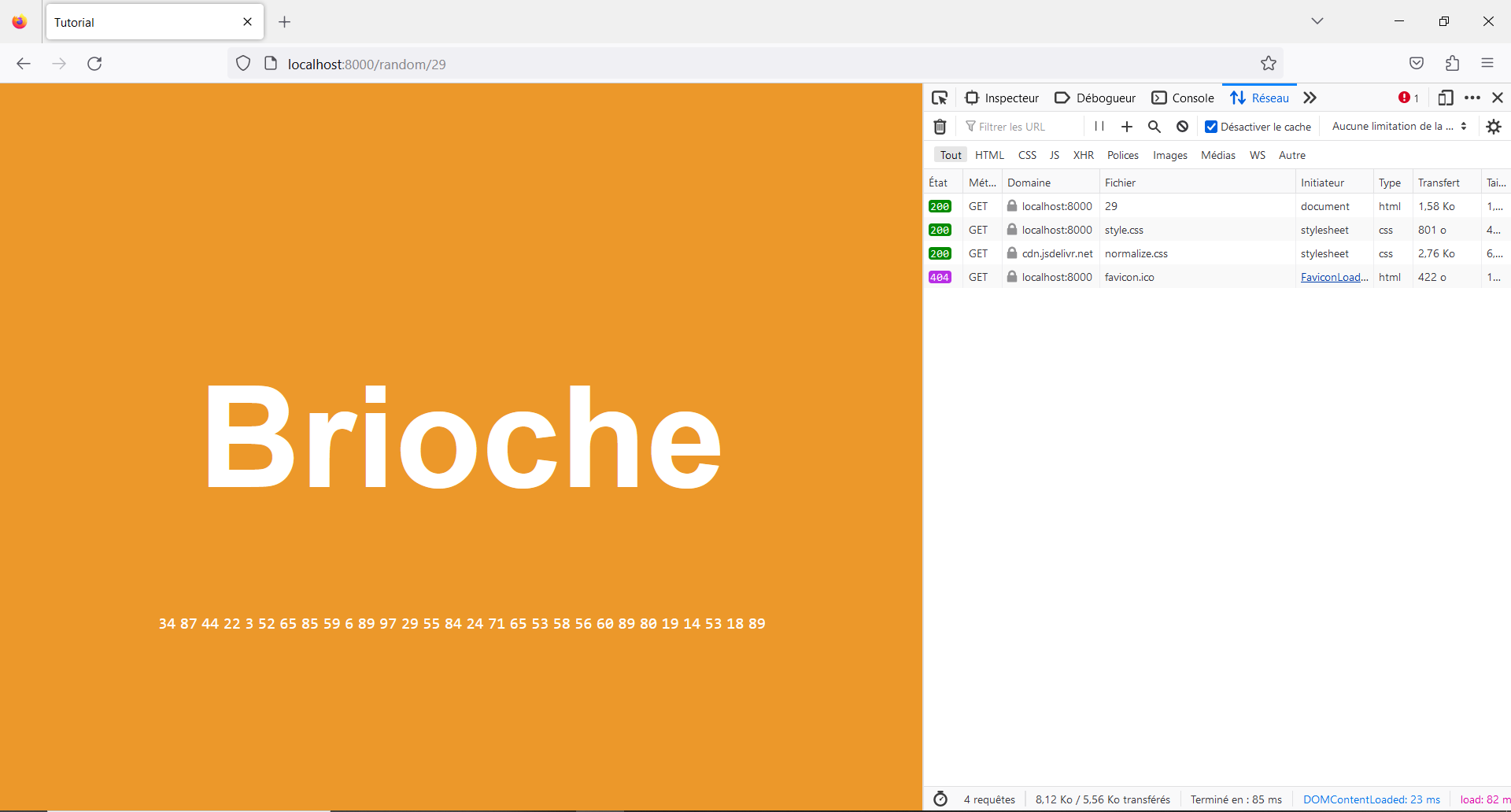
**Rendu avec EJS /**

4. modifier le *handler* de la route /random/:nb avec response.render("random", {numbers, welcome}); pour appeler le moteur de rendu, où numbers est un tableau de nombres aléatoires (comme précédemment) et welcome une chaîne de caractères.

Handler modifié :



Résultat pour */random/29*(*numbers*=29 et *welcome*=’Brioche’) :



**Gestion d’erreurs /**

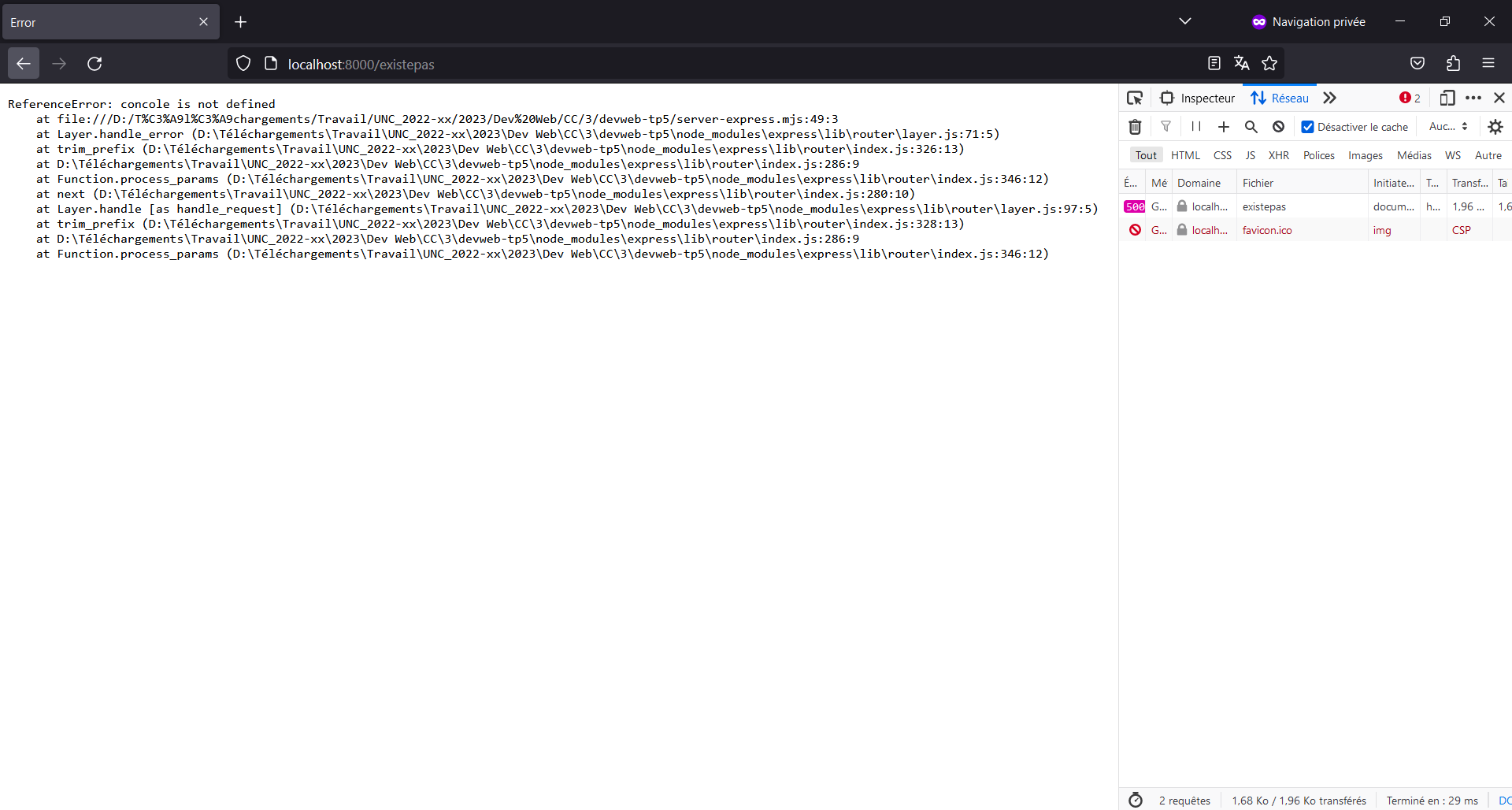
3. dans la route /random/:nb, faites la vérification avec const length = Number.parseInt(request.params.nb, 10); puis Number.isNaN(length), si le paramètre, n’est pas un nombre, produire une erreur 400 avec next(createError(400));

Route modifiée :



**Question 2.7 (1)** vérifier que l’affichage change bien entre le mode production et le mode development.

Affichage en mode *development*:



Affichage en mode *production*:

