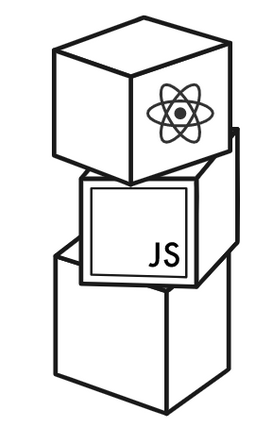
**Projet Programmation Web III 2022-2023**Réalisation d’un cahier de travail dédié au cours de Fullstack Open



POPADIUC CLAUDIU

Dans le cadre du cours de Monsieur Riggio, Jonathan

Table des matières

[Introduction 3](#_Toc128729416)

[Part 0: Fundamentals of Web apps 4](#_Toc128729417)

[a) General info (7 Avril 10h35) 4](#_Toc128729418)

[b) Fundamentals of Web apps (7 Avril 11h40) 4](#_Toc128729419)

**Part 0 : Introduction aux applications Web (03/03 à 9h51)**

**a) Informations générales** (03/03 à 9h56)

Ce cours est une introduction au developpement JavaScript, centrer sur React et leur prise en charge avec les services Web RESTful et GraphQL qui sont implementer avec Node.js.

Les participants de ce cours doivent déjà avoir bonne maitrise en programation Web et en Base de données, ce cours attend aussi des personne curiueuse et autonome.

Il y aurra dans le cours des exercices à faire au fur et a mesure que le lecteur avance dans le cours, toutes les ressources necesaire a la resolution des exercices seront dans le cours. A la vue des exerceices, ils peuvent etre effectuer directement mais ça peut etre interessant de lire le cours en entier avant de commencer les exercices. Pour finir les solutions de ces exercices se trouveront sur git hub.

Il est preferable d’utiliser Google Chrome pour ce cours car il fournit les meilleurs outils de développement Web. Pour le développement Web, ne codez pas avec nano, Notepad ou Gedit mais plutôt avec Visual Studio Code, c’est fortement recommandé.

Il nous a donc été indique de instaler Visual Studio Code, chose qui etais déjà faite pour moi, il nous a aussi été indique de insatler Node.js, une version superieur a celle de : « 16.13.2 » car le cours est base sur cette version, j’ai donc instaler la version : « 18.14.2 » car c’est la version recommande selon le site officiel de Node.js. Je n’ai absolument pas eu de difficulte a instaler Node.js, j’ai pu l’ouvrir en moin de 2 minutes.

**b) Introduction aux applications Web** (03/03 à 10h36)

Quelques bons principes de developpement Web, nous pouvons les retrouvers à cet addresse « <https://studies.cs.helsinki.fi/exampleapp> »

Premiere règle du développement web, il faut toujours gardez la Console du développeur ouverte, chose que je ne faisais le d’ailleurs pas toujours lorsque je developait en web, notament sur le projet Q3, j’ai déjà appris quelque chose.

**HTTP GET**

Pour le 1er exercice de ce cours, il a d’abord fallu apprendrea utiliser la console, c’est ce que j’ai fait et j’ai notament appris les parties de la console comme « Network », mon premier probleme a été le fait que mon navigateur c’est firefox et comme dit precedement il est preferable d’utiliser Google, et j’ai pu le remarquer directement car la console de Firefox n’est pas la meme que celle dans Google.

Firefox : Google :

Une image contenant texte

Description générée automatiquementUne image contenant texte

Description générée automatiquement

On peut y remarquer quelque leger changement comme, « Persserve log » qui n’est pas dans la console Firefox et qui était affiche dans les exercices comme coché, j’ai donc pu la cocher sur Google mais pas sur Firefox car je ne trouvais juste pas cette case, j’ai donc eu des difficulter sur Firefox alors j’ai change de navigateur pour continuer l’exercice.

Pour cet exercice, grace a l’onglet reseau on peut voir la comunication entre le serveur et la page web. Avec la comande « CTRL + R » on peu voir les différents documents qui ont été récupère par le navigateur.

Une image contenant texte, tableau de points

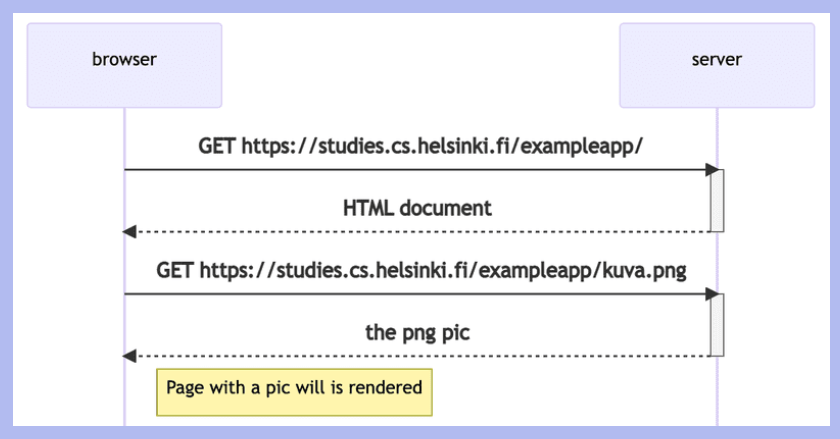
Description générée automatiquement

Nous pouvons aussi voir la structure de la page grace à « Response », lorsque on clique par exemple sur exampleapp : Une image contenant texte

Description générée automatiquement

Je trouve cela interessant, ce sont des choses que je ne savais pas, je vais surement pouvoir utiliser cette pratique pour mes developement futur en Web.

08.03 – 14 :20



Explication de l’image :

Voici le schéma global de comment fonctionne une requête entre le navigateur et le  
serveur.

• Le navigateur envoie une requête http GET  
• Le serveur renvoie le code HTML  
• Le navigateur envoie une requête http GET pour recevoir l’image  
• Le serveur renvoie l’image demandé  
• Le navigateur se charge de crée le rendu que vous voyez à l’écran

**Running Application logic in the browser**

Lorsquon est sur la console, il est possible d’effacer la console en ecrivant tout simplement : clear (), dans la console

Nous avons ensuite cela dans la console :

**Une image contenant table

Description générée automatiquement**

Ici nous pouvons voir que toute la page ne se trouve pas dans cet HTML, pourtant il faudrait, et j’ai trouver cela bizzard que cela ne le soit pas mais j’ai continuer le cours et fortheureusemet Il ne s’agit pas d’un probleme chez moi mais plutôt que l’entiereter du code se trouve ailleur, en effet les balises <script> contiennent « main.js », ce genre de comportement permet de generer du code javaScript.

Une image contenant texte

Description générée automatiquement

Voici donc ce fameux main.js :

Une image contenant texte

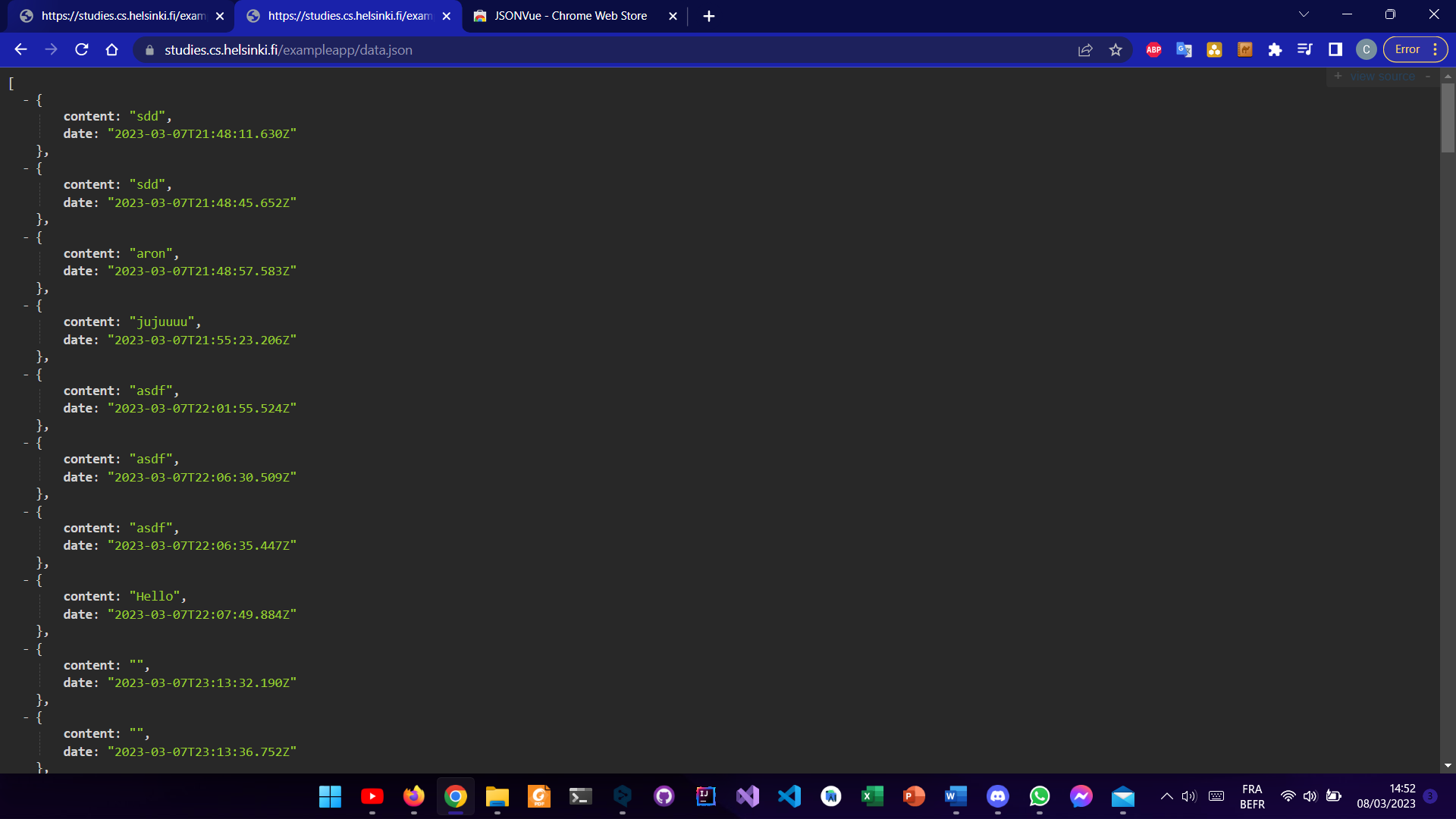
Description générée automatiquement

Les 2 dernières lignes sont importantes car ça permet de faire un http GET request au serveur. Et en cliquant sur ce lien : <https://studies.cs.helsinki.fi/exampleapp/data.json> nous avons ce resultat, par defaut c’est afficher comme ça, en effet les navigateurs ne sont pas tres bon pour afficher du json :

Une image contenant texte, moniteur, capture d’écran, écran

Description générée automatiquement

Cependant si j’ajoute cette extension, le resultat est beaucou plus structurer, et en effet j’arrive à lire et comprendre : https://chrome.google.com/webstore/detail/jsonvue/chklaanhfefbnpoihckbnefhakgolnmc/related



**Event handlers and Callback functions**

Comme le dit FullStack, « voici la structure de code un peu bizzard » :

Une image contenant texte

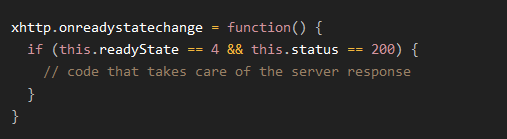
Description générée automatiquement

La requête au serveur est envoyée sur la dernière ligne, mais le code pour gérer la réponse se trouve plus haut, c’est surtout ça qui est bizzard.

Une image contenant texte

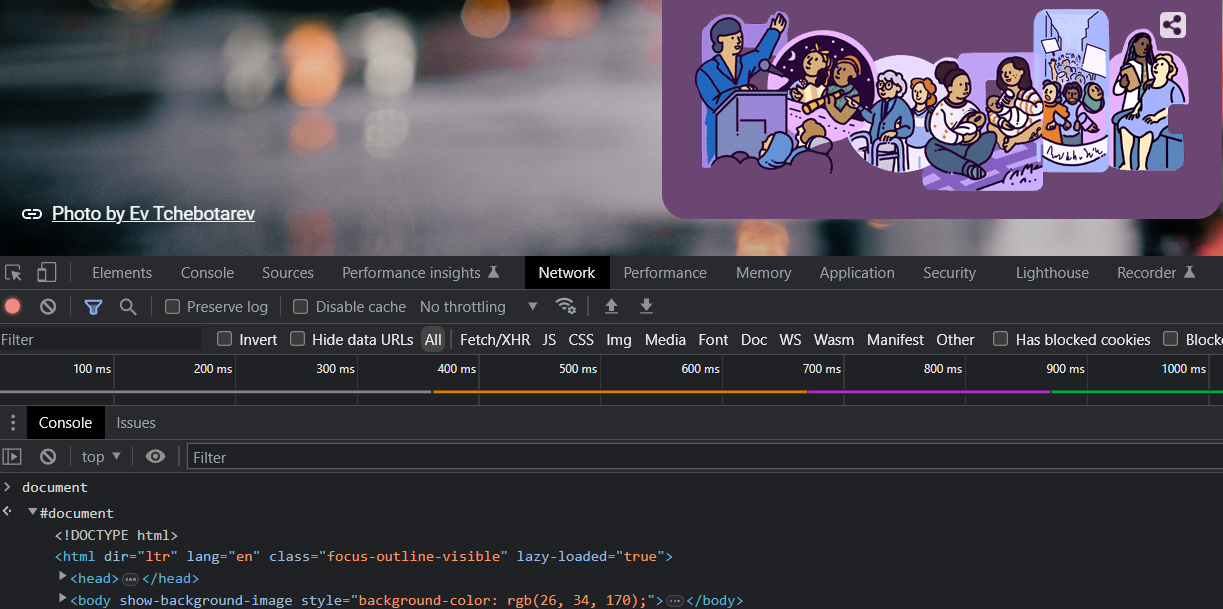
Description générée automatiquement

Enfait, ce qu’il se passe c’est que sur cette ligne, c’est qu’un gestionnaire d'événement pour l'événement onreadystatechange est défini pour l'objet xhttp, et c’est ca qui effectue la requête, j’ai eu du mal a comprendre directement ce qui etais bizzard, mais finalement grace au explication de FullStack sa suffit amplement, jy apprend beaucoup de chose.



**Manipulating the document-object from console**

En allant dans la console d’une page, on peux taper « document » et  
celle-ci vous retournera le document comme dans l’image qui suit que j’ai testé moi-meme :

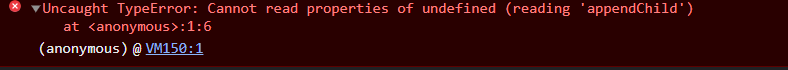


On peut dire qu’on va un peu jouer avec la console d’une page web, on va y ajouter ces lignes de codes :

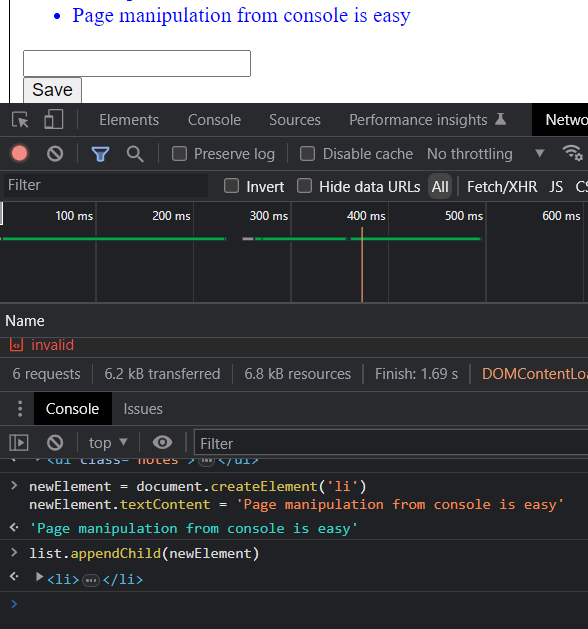
Une image contenant texte

Description générée automatiquement

Après l’avoir fait moi-même j’ai rencontré cet erreur :



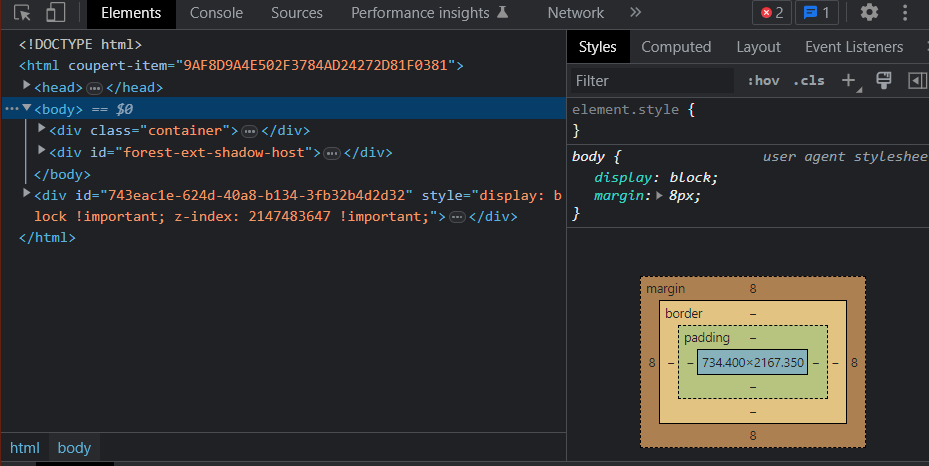
Et en fait c’était parceque je n’etais pas sur la page du cours, et plus precisement dans les Notes, apres avoir été dans cette page la, j’ai re tester et preuve a l’appuie, ca a fonctionner :



En revanche si je recharge la page, mon element disparait, c’est du au fait que les modifications sont faites du coté Client et pas du coté Serveur.

**CSS**

Le CSS est accessible dans « Eléments », dans plus exactement « Styles ». Ce qui est bien c’est qu’on peut modifier le css directement a partir de la, si on le souahaite. Je ne vais pas rentrer plus dans les details car c’est un langage connu.



**Forms and HTTP POST**

Si on reste bien sur la page : <https://studies.cs.helsinki.fi/exampleapp/notes> , on peux remarquer qu’on peut ajouter des notes et les sauvegarder. Ces notes conttiente un element « Form » comme vous pouvez le voir ci-dessous.

Une image contenant texte

Description générée automatiquement

Une fois le bouton « save » presser, les informations seront envoyées au serveur. Dans  
Network, vous pouvez voir les requettes qui ont été envoyer.

Une image contenant table

Description générée automatiquement Une image contenant texte

Description générée automatiquement

On peut voir dans l’image ci-dessus qu’il s’agit d’une méthode POST et qu’elle contient le code 302, ce qui veut dire que le serveur à trouvé cette requette.

Pour le http Form, ceci était asse importante a comprendre et a se rendre compte de comment ca fonctionne.

**Ajax**

Un peu d’histoire…

La page Notes utilise une technologie appelée AJAX qui a été développée dans les années 2000 pour récupérer du contenu sur des pages web sans avoir besoin de les recharger. Avant cela, toutes les données étaient récupérées à l'aide du code HTML généré par le serveur. La page Notes utilise encore une ancienne façon de soumettre des formulaires web, mais les URL utilisées ne respectent pas les conventions généralement reconnues des API RESTful. Je cite cette phrase venu du cours : « Ce que l'on appelle AJAX est aujourd'hui si courant qu'il est considéré comme allant de soi. Le terme est tombé dans l'oubli et la nouvelle génération n'en a même pas entendu parler. » Cette phrase est dite car en effet aujourd'hui, AJAX est très couramment utilisé et est considéré comme normal, mais il a été révolutionnaire à l'époque où il a été introduit.

**JavaScript-libraries**

Il est important de savoir que JavaScript peut avoir des comportements imprévus par rapport à d'autres langages. Pour maîtriser complètement JavaScript, il est nécessaire d'avoir d'autres compétences telles que la connaissance du protocole HTTP, la gestion de bases de données et leur administration, ainsi qu'une base en CSS pour créer des applications acceptables. JavaScript est régulièrement mis à jour, ce qui entraîne des problèmes de compatibilité entre les différentes versions des bibliothèques d'outils. Les développeurs doivent donc gérer ces mises à jour en mettant régulièrement à jour leur code, ce qui peut fatiguer la communauté de JavaScript.

**Exercises 0.1. – 0.6.**

J’ai pri la sage décision de passer ses exercices et reprendre à partir du cours 1. En ce qui concerne les exercices j’ai considere que la plupart était assé simple et déjà effectuer durant mon cursus de cette année, pour certains plusieurs fois meme si ce n’était pas pile la meme chose, cela y ressemblait ou alors utilisait le meme language.

**Part 1 : Introduction to React** 08.03.2023 16h19

Pour commencer j’ai du ajouter cette ligne dans un cmd Windows :

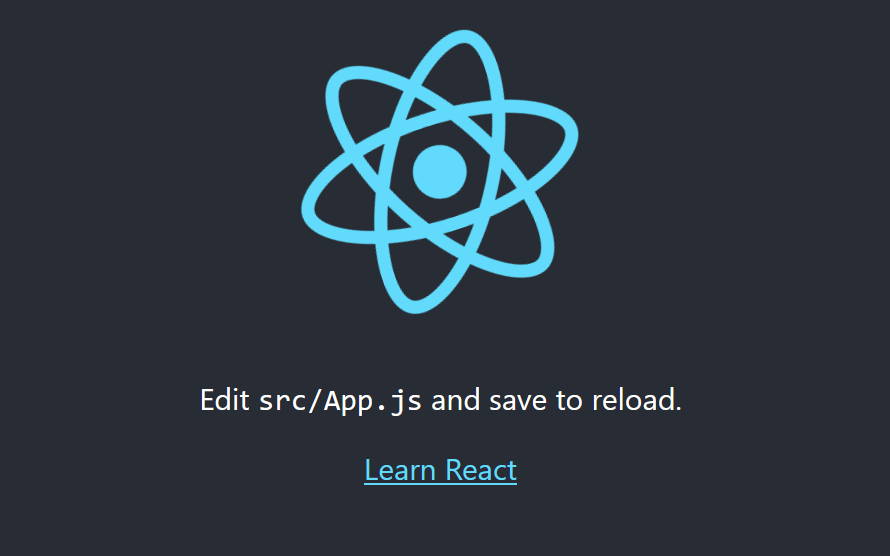


Pour la creation, j’ai remarqué que ca a quand meme pri plusieurs minute et c’a n’a pas fonctionner du premier coup chez moi

Et pour demarer :



Une fois cette commande ecrite dans le cmd, une fenetre c’est crée dans mon naivgateur :



Mais pour travaillez dessus il a fallu ouvrir vsCode, un IDE que j’utilise beaucoup et que j’adore. Il faudra ouvrir dedans le projet, part1 qui est dans le disque C, Utilisateur, Le nom de votre ordinateur et « part1 »

Pour commencer reelement il va falloir ajouter ceci dans *index.js*

Une image contenant texte

Description générée automatiquement

Et ceci dans *App.js*



Si on teste ce simple « Hello world » voici le resultat final :

Une image contenant texte

Description générée automatiquement

Les fichiers suivants peuvent être supprimés car ils ne sont pas nécessaires à notre application pour le moment.

App.css

App.test.js

index.css

logo.svg

setupTests.js

reportWebVitals.js

En suprimant ces fichiers, j’ai eu ces erreurs :

Une image contenant texte

Description générée automatiquement

Pour corriger ces erreurs il a fallu que je change le code qui etais dans index.js par celui-ci :

Une image contenant texte

Description générée automatiquement

J’ai ensuite continué l’indroduction a React et j’ai ajouter ce code dans *public/index.html* et ca a fonctionné parfaitement du 1er coup.

Une image contenant texte

Description générée automatiquement

Voici a quoi ressemble un composant React, on peut le voir dans  mon fichier *App.js*, au dedbut j’avais du mal a comprendre ou ajouter ce code mais au final il etais déjà dans ce fichier la que j’ai du retrouver, je commence a comprendre petit a petit comment fonnctionne react :

Une image contenant texte

Description générée automatiquement

Si on demare la console on peux remarquer que le log foncrionne :

Une image contenant texte, capture d’écran, moniteur, écran

Description générée automatiquement

On peux aussi modifier le code de cette facon et afficher la date ainsi que un calcul :

Une image contenant texte

Description générée automatiquement Une image contenant texte

Description générée automatiquement

Il faut faire attention a ne pas suppirmer cette ligne, sinon ca ne fonctionne plus.



**JSX**

D’apres ce que l’on voit React renvoient des balises HTML. Par contre, ce n'est pas le cas. La mise en page des composants React est principalement écrite à l'aide de JSX. Bien que JSX ressemble à du HTML, il s'agit d'une façon d'écrire du JavaScript.

Voici a quoi ressemble notre code apres la compilation :

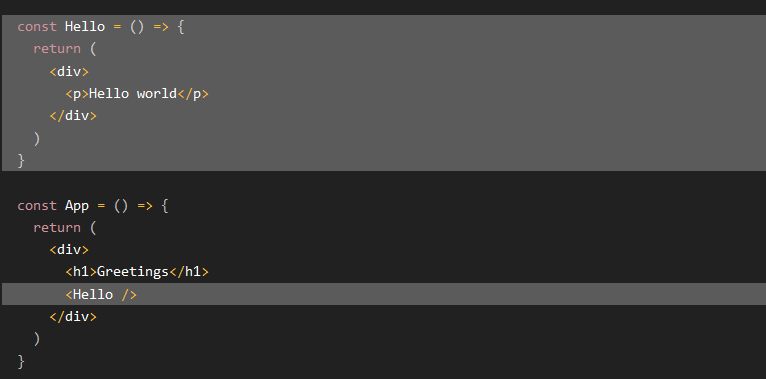
Une image contenant texte

Description générée automatiquement

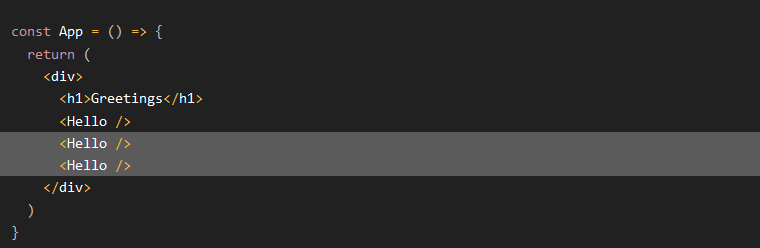
Dans la pratique, JSX présente de nombreuses similitudes avec HTML. Cependant, la principale différence c’est dans la possibilité d'intégrer du contenu dynamique en utilisant des balises JavaScript insérées entre des accolades.

**Multiples components**

Si on modifier le fichier *app.js* de cette facon :



Ici on a utiliser un component dans le 1er component, par contre dans l’image suivant on a carement utiliser un component dans un autre plusieurs fois



Franchement je trouve cela assé simple, et comme il est si bien dit dans le cours : *« Il est facile d'écrire des composants avec React, et en combinant des composants, même une application plus complexe peut rester relativement facile à maintenir. »*

**Props: passing data to components**

Il est aussi possible de passer en parametre des données aux composants avec des « props » :

Une image contenant texte

Description générée automatiquement Une image contenant texte

Description générée automatiquement

Ainsi dans la constante « App » on appelle la fonction Hello avec un paramètre « name », grace a ça on pourra reprendre George & Daisy

Voici le code pour que le composant Hello utilise deux props :

Une image contenant texte, capture d’écran, moniteur, noir

Description générée automatiquement

Sinon pour terminer cette partie, le cours ajoute quelque chose de asse drole qui je pense est assé important, je cite : *« J'espère vraiment que votre console était ouverte. Si ce n'est pas le cas, rappelez-vous ce que vous avez promis :*

*Je promets de garder la console ouverte tout le temps pendant ce cours, et pour le reste de ma vie lorsque je ferai du développement web »*

Du coups je pense que en effet, c’est important, le developpement web devient de plus en plus difficile au fur et a mesure que on avance dans un gros projet et c’est surement encore plus difficile si on utilise pas tout les outils qui sont mi a notre disposition.

08.03.2023-17h45

**Some notes** 17.03.2023 09h19

Je reprend le cours 1 semaine plus tard et il y a eu une petite difficulte a redemarer mon projet mais finalement c’était simple j’ai du retrouver mon dossier src qui etais dans mon C/User/part1

J’ai ensuite run ca et ca a fonctionner :



Revenons en au fait, React a été configure pour generer des erreur claire, et nous devrions des le debut avancer par petit pas et voir que chaque changement desirer ce produit bel et bien, evidement la console devrait toujours etre alume, ainsi si le browser indique des erreur, il est inutile de continuer tout droit dans un mur mais plutôt regler les souci car il y n’aura pas de miracle, il faut aussi prendre la peine de comprendre les erreurs ainsi on poura progresser, je fait une parralele avec le cours de mobile, depuis que j’ai commencer a reelement m’inpliquer dans les erreurs que je recois, je regler beaucoup plus vite mes soucis.

il est possible et intéressant d'écrire des commandes console.log() (qui impriment sur la console) au sein de votre code.

**Do not render objects**

Je vais maitenant essayer d’afficher les noms et les ages de 2 de mes amis :

Une image contenant texte, capture d’écran, écran, ordinateur portable

Description générée automatiquement

Je rencontre directement des erreurs, rien de ce que je veux apparait a l’ecran, et finalement voici ce que dit ma console : “Objects are not valid as a React child” Le code tente de restituer les informations relatives à un ami de la manière suivante <p>{friends[0]}</p> <p> Cette tentative pose un problème car l'élément à rendre entre accolades est un objet. et cela pose un problème parce que l'élément à rendre entre les accolades est un objet. Probleme regler voici le resultat :



**Exercice 1.1.-1.2**

**Exercice 1.1** 09h36

Pour commencer cet exercice, nous devons crée une nouvel App React comme nous l’avons vu precedement.

Une image contenant texte

Description générée automatiquement

J’ai d’abord eu un mini souci qui etais que il faut crée une app sans majuscule. (J’ai lu l’erreur quand meme, encore heureux :) )

J’ai aussi ouvert la console avant qu’il me le dise pour une fois, je commence a m’habituer.

Il va falloir faire 2 modification important dans ce noiveau projetn les voici :

Dans le *index.js*

Une image contenant texte

Description générée automatiquement

Et dans *App.js*

Une image contenant texte

Description générée automatiquement

Il a evidement fallu suprimer tout les fichier inutile tel que : App.css, App.test.js, index.css, logo.svg, setupTests.js, reportWebVitals.js, finalement tout apart les 2 fichier qu’on a modifié.

Pour la suite de l’exercice il ma été un peu difficil de comprendre ce qu’il fallait faire, quand j’ai compris ca fonctionnait pas… il falait

(Vien du cours, a reformuler )

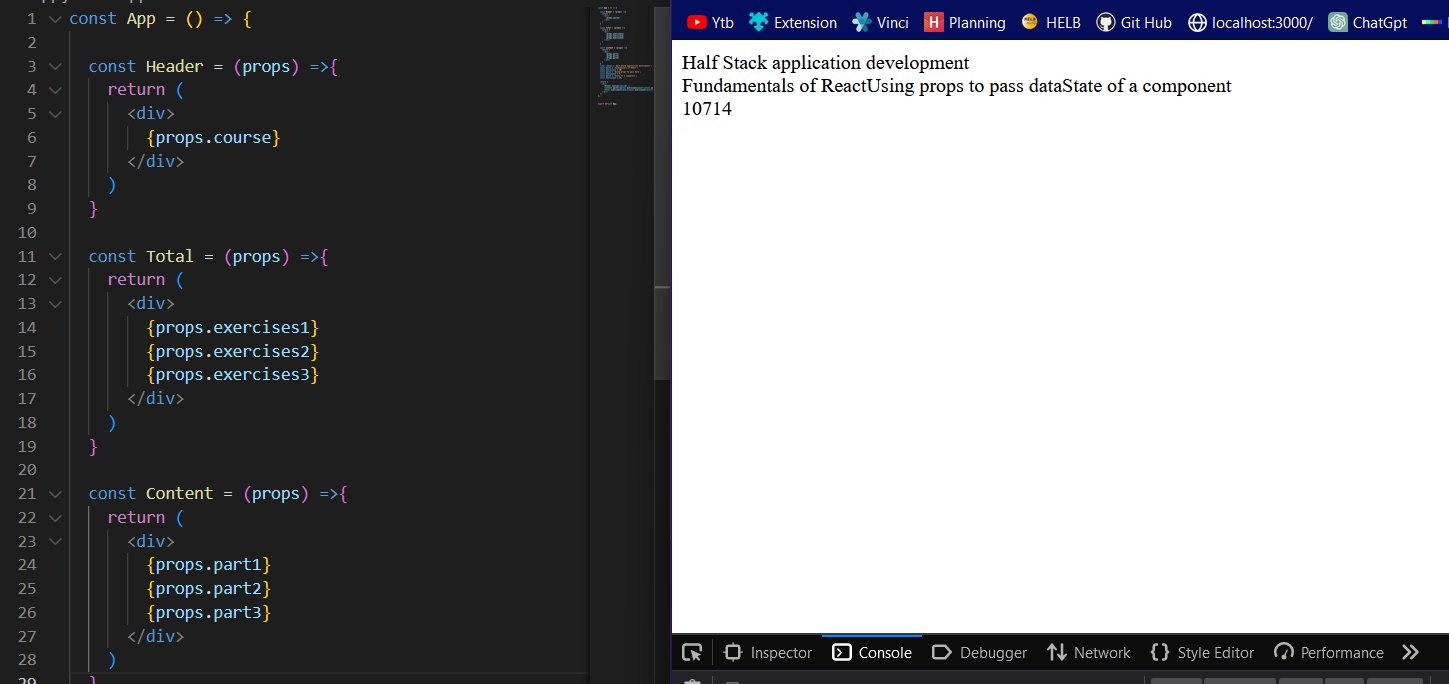
Reformulez le code de manière qu'il soit constitué de trois nouveaux composants : En-tête, Contenu et Total. Toutes les données résident toujours dans le composant App, qui transmet les données nécessaires à chaque composant à l'aide d'accessoires. Header se charge de rendre le nom du cours, Content rend les parties et leur nombre d'exercices et Total rend le nombre total d'exercices.

Voici donc ce que j’ai fait :

Une image contenant texte

Description générée automatiquement

Mais en faite il fallait ajouter les components qui prennent en paramètre « props »  
et défini les constantes, et enfin ca a fonctionné.



10h32

Voici un avertissement qui nous a été signale je vais le reprendre du site :

*AVERTISSEMENT N'essayez pas de programmer tous les composants en même temps, car cela entraînerait presque à coup sûr une panne de l'application entière. Procédez par petites étapes, faites d'abord par exemple le composant Header et ce n'est que lorsqu'il fonctionne avec certitude que vous pouvez passer au composant suivant.*

*Une progression prudente, par petites étapes, peut sembler lente, mais c'est en fait de loin la façon la plus rapide de progresser. Le célèbre développeur de logiciels Robert "Uncle Bob" Martin a déclaré*

*"La seule façon d'aller vite, c'est d'aller bien".*

*c'est-à-dire que, selon Martin, une progression prudente à petits pas est même la seule façon d'être rapide.*

*ATTENTION2 create-react-app fait automatiquement du projet un dépôt git à moins que l'application ne soit créée dans un dépôt déjà existant. Il est fort probable que vous ne souhaitiez pas que le projet devienne un dépôt, alors exécutez la commande rm -rf .git à la racine du projet.*

**Exercice 1.2 course information, step2**

En se basant sur lex 1.1, il faut modifie le contenu du app.js de manière à ce qu'il ne rende pas lui-même les noms des parties ou leur nombre d'exercices. Mais il doit rendre que trois composants « Part » et chacun rend le nom et le nombre d'exercices d'une partie.

Voici le rendu :

Une image contenant texte

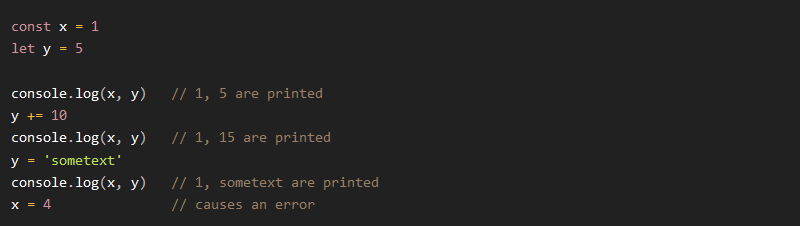
Description générée automatiquement

**JavaScript** 10h41

Ce qui est important dans ce cours, c’est qu’il est neccesaire d’apprendre une quantite suffisante de JavaScript en plus du dev web, chose que je pense que je vais beaucou aime car lorsque du projet django, j’ai utilise beaucoup de js et j’ai pri plaisir a le faire, je pense que elargir mes connaisance dans ce domaine me fera du bien.

**Variables**

Les variables en JavaScript se définissent de la façon suivante :

****

• const : Comme une sorte de Final, elle n’est pas sense etre modifier plus tard.  
• let : c’est plutôt pour toutes les sorte de variable

**Arrays**

Dans cet exemple je peux remarquer que le contenu du tableau peut etre modifier meme si il est defini comme const, en effet comme le tableau est un objet, la variable va pointer toujours vers le meme objet. Mais cest parceuque le tableau change des que nouveau element y sont ajouter.**Une image contenant texte

Description générée automatiquement**

L'appel de la méthode t.concat n'ajoute pas un nouvel élément à l'ancien tableau, mais renvoie un nouveau tableau qui contient le nouvel élément et les anciens.

**Une image contenant texte

Description générée automatiquement**

Les variables first et second recevront les valeurs des 2 premier entier du tableau, et rest prendra comme son nom l’indique le rest ainsi on affichera le premeir et le 2ieme et dans un autre log le reste

**Une image contenant texte

Description générée automatiquement**

**Objects**

Il existe plusieurs façons de définir des objets en JavaScript. Voici l'une des méthodes les plus utlisié. Le code propose par le cours était difficil a implementer alors j’ai demander a ChatGpt de le faire pour moi et le resultat est fonctionel :

**Une image contenant texte

Description générée automatiquement**

En JavaScript, il est possible de créer des objets et leurs assignés plusieurs valeurs de  
différents types.

**Une image contenant texte

Description générée automatiquement**

**Functions.**

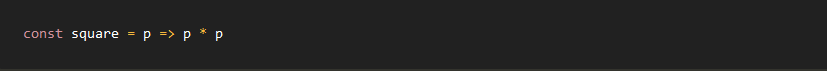
Voici comment definir une fonction a 2 parametre : 



Si il n’ya qun seul parametre on peux exclure la parenthese de la definition :



Il ne faut pas de parenthèse si un seul argument est donné en paramètres.



**Exercices 1.3.-1.5.** 11h58

**Exercices 1.3 : course information step3**

On va passer a l’utilistaiton d’objet dans notre application, il va falloir modifier notre App.js pour que ca continue de fonctionner.

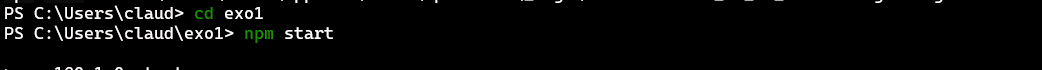
Voici comment utiliser les arguments des objets dans un return.  
On crée 3 objets qui sont les 3 parties du cours, cette partie du code était dans le cours par contre, ensuite dans le on met sous format HTML en y référençant les éléments de l’objet auquel on se réfère, et ca il a fallu le faire soit meme.

Une image contenant texte

Description générée automatiquement

**Exercices 1.4 : course information step4** (24.03 9h22)

De nouveau dans mon projet, ces 2 lignes sont primordiales pour faire l’exercice :



Pour l’exercie 1.4, il faut reprendre la suite de l’exercie 1.3, et selon le cours il faut + placez les objets dans un tableau. Modifiez les définitions des variables d’App sous la forme suivante :

Une image contenant texte

Description générée automatiquement

Voici la resolution de cet exercice :

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

Le composant Header affiche simplement le nom du cours. Le composant Content affiche les noms et le nombre d'exercices pour chaque partie du cours, en utilisant le composant Part. Enfin, le composant Total affiche le nombre total d'exercices dans le cours.

L’explication du code est donc la suivante, on passe en props une liste d’objet qu’on  
récupère simplement en indiquant leurs positions dans l’array ainsi que l’attribut qu’on  
souhaite récupérer.

**Exercices 1.5 : course information step 5**

Voici ce qu’il faudra ajouter dans notre code pour Transformons le cours et ses parties en un seul objet JavaScript.

Une image contenant texte

Description générée automatiquement

Pour cet exerice ci, il etais plus simple car il falait simplement le changer  
return le Content et le Total ou je donne en paramètre course. parts à la place de juste parts.  
Le second changement se fait dans le header ou ça devient props.course.name. Le reste n’a  
posé aucun problème. L’explication est qu’en mettant tout dans un seul objet « course », il  
faut ensuite préciser de quel élément de course on fait référence.

Une image contenant texte

Description générée automatiquement

**Object methods and « this »**

Ce cours utilise une version de React contenant des React Hooks, ce qui signifie que il n’y a pas besoin de définir des objets avec des méthodes. De ce fait le contenu de ce chapitre n'est pas pertinent pour le cours mais selon le cours : *« il est certainement bon à bien des égards de le connaître. En particulier, lorsque l'on utilise des versions plus anciennes de React »* A mon avis, il est important de le conaitre car si un jour on travail dans une entreprise avec une anciennes version de React, il serait plus facil pour moi de comprendre le code.

Avec ces methodes, nous pouvons assigner des méthodes à un objet en définissant des propriétés qui sont des fonctions :

Une image contenant texte

Description générée automatiquement

Les méthodes peuvent être assignées aux objets même après la création de l'objet :

Crée une fonction dans un objet en utilisant un « this » réfère toujours à l’objet dans lequel  
on se trouve.

Une image contenant texte, capture d’écran, écran

Description générée automatiquement

Maintenant, l'objet possède une méthode appelée "doAddition" qui calcule la somme des nombres fournis en tant que paramètres. Cette méthode peut être appelée de la manière habituelle en utilisant l'objet "arto.doAddition(1, 4)" ou en stockant une référence à la méthode dans une variable, puis en appelant la méthode à travers cette variable, par exemple : "referenceToAddition(10, 15)".

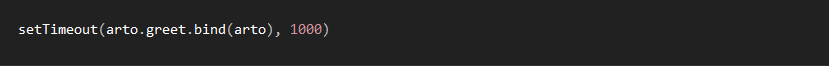
Cependant, si nous essayons de faire la même chose avec la méthode "greet", nous rencontrerons un problème :

Une image contenant texte, écran, capture d’écran

Description générée automatiquement

Comme nous l'avons expliqué, en JavaScript, la valeur de ce paramètre dépend de la façon dont la méthode est appelée. Lorsque setTimeout appelle la méthode, c'est le moteur JavaScript qui appelle réellement la méthode et à ce moment-là, "this" fait référence à l'objet global.

Il existe plusieurs façons de conserver l'objet global d'origine. L'une d'entre elles consiste à utiliser une méthode appelée "bind":



**Classes**

Dans ce point nous allons juste voir la syntaxe de classe avec JavaScript.

Dans l'exemple suivant, nous définissons une "classe" appelée Personne et deux objets Personne :

Une image contenant texte

Description générée automatiquement

**3) Component state, event haldlers** (24.03 - 10h10)

Nous passons dans une nouvelle partie et pour commencer voici un nouvel exemple :

**Une image contenant texte

Description générée automatiquement**

**Component helper functions**

Changement sur le composant Hello pour qu'il devine l'année de naissance de la personne : Une image contenant texte

Description générée automatiquement

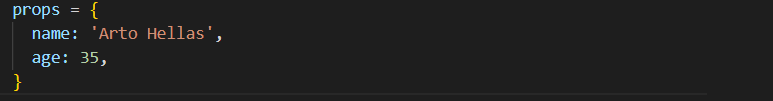
La fonction d'aide est une fonction qui est créée à l'intérieur d'une autre fonction pour décrire le comportement de notre composant. En programmation Java, cette méthode peut être difficile et volumineuse, c'est pourquoi elle n'est pas souvent utilisée. En revanche, en JavaScript, la création de fonctions à l'intérieur d'autres fonctions est une technique couramment utilisée.

**Destructuring**

Voici une fonctionnalité du langage JavaScript qui a été ajoutée dans la spécification ES6, qui nous permet de déstructurer les valeurs des objets et des tableaux lors de l'affectation.

Dans notre code précédent, nous avons utilisé props.name et props.age pour accéder aux données transmises à notre composant. Cependant, nous avons dû répéter deux fois props.age dans notre code.

Mais vu que props est un objet on peux faire cela :



Pour simplifier notre composant, nous pouvons assigner les valeurs des propriétés directement à deux variables, name et age, que nous pourrons ensuite utiliser dans notre code.

Une image contenant texte, écran, capture d’écran, fermer

Description générée automatiquement

Pour résumer, les deux définitions de fonction présentées ci-dessous sont équivalentes :

Une image contenant texte

Description générée automatiquement

La déstructuration simplifie encore plus l'affectation des variables en permettant d'extraire et de regrouper les valeurs des propriétés d'un objet dans des variables distinctes.

Une image contenant texte, écran, capture d’écran, fermer

Description générée automatiquement

L’expression const { nom, âge } = props attribue les valeurs 'Arto Hellas' à name et 35 à age.

Mais nous pouvons alle plus loin avec la destruction de cette manière :

Une image contenant texte, écran, capture d’écran

Description générée automatiquement

Petite parenthese :

Je trouve ce concept tres interesant, je ne savais aboslument pas que c’était possible, ce qui est bien avec ce cours c’est que plus on avance plus on apprend sur la programmation en general et je pense que c’est que du bien fait pour mon experience.

**Page re-rendering**

Jusqu'à présent, toutes nos applications ont été conçues de manière à ce que leur apparence reste inchangée après le rendu initial. Que se passerait-il si nous voulions créer un compteur dont la valeur augmenterait en fonction du temps ou en cliquant sur un bouton ?

Commençons par ce qui suit. Fichier App.js devient :

Une image contenant texte

Description générée automatiquement

Et index.js devient :

Une image contenant texte

Description générée automatiquement

Je n’ai pas continuer la suite de l’exemple car finalement c’était juste un compteur qui incrementait, mais je laisse le début car je trouve cela interessant la facon dont ca été fait.

**Event handling**

Dans React, l'enregistrement d'une fonction de gestion d'événement pour l'événement click se déroule comme de cette facon :

Une image contenant texte, capture d’écran, écran, intérieur

Description générée automatiquement

La fonction de gestion d'événement peut également être définie directement dans l'affectation de la valeur de l'attribut onClick :

Une image contenant texte

Description générée automatiquement

J’ai essayer d’implementer ca dans mon code pour le tester moi-même et j’ai eu quelque erreur du style : « Line 2:35: 'useState' is not defined » et enfaite j’avais pas correctement exporté mon composant App dans le fichier App.js.

Voici le resultat final qui fonctionne lorsque j’appuie sur le bouton plus, ca incremente et « zero », mets tout a 0, c’est quelque chose de tres basique mais qui reste interesant la facon dont c’est fait en React :

Une image contenant texte, capture d’écran, écran

Description générée automatiquement

**D) A more complex state, debugging React apps** (24.03 – 11h13)

**Complex state**

Dans le code suivant, nous créons deux éléments d'état pour l'application, qui sont tout les 2 a 0 :

Une image contenant texte

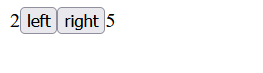
Description générée automatiquement

L'état du composant ou un élément de son état peut être de n'importe quel type. Nous pourrions mettre en œuvre la même fonctionnalité en enregistrant le nombre de clics des boutons gauche et droit dans un seul objet :

Une image contenant texte

Description générée automatiquement

En testant le code j’ai eu des erreurs, enfaite l’avais pas mi la ligne 1 avec l’import qui était necceaisre car dans mon code j’utilisais la fonction « useState » Voici le resultat



Lorsque j’appuie sur le bouton gauche ca incremente a gauche, a droite ca incremente a droite et ca fonctionne bel et bien.

**Update of the state is asynchronous.**

Maitenant l’application pourra enregistrer le nombre total des boutons dans un état global. Cette valeur doit être mise à jour en temps réel chaque fois que les boutons sont pressés.

Une image contenant texte

Description générée automatiquement

Cela a fonctionne directement chez moi, je commence a mieux m’habituer avec React, je peux mieux implementer les codes du cours.

Mais selon le cours, ce n’est pas bien fait, Le nombre total de pressions sur les boutons est toujours inférieur d'une unité au nombre réel de pressions, pour une raison quelconque.

Alors je vais ajouter des console.log

En effet lorsque j’appuie sur le bouton gauche, c’est bon mais quand j’appuie a droite cela ne fonctionne plus, il ne s’affiche pas dans la console :

Une image contenant texte

Description générée automatiquement

Comme vous pouvez le voir tout fonctionne a merveille : Une image contenant texte

Description générée automatiquement

Pour que ca fonctionne j’ai du modifier la methdode : « handleLeftClick » en ajoutant cette ligne ci :



Et grace a cela, je n’avais plus le probleme d’affichage, mais j’ai aussi du changer le setLeft et le setTotal en ajoutant cette fois ci la variable que je venait de crée. Ca ne fonctionnait pas car la mise à jour d'état dans React se produit de manière asynchrone, c'est-à-dire pas immédiatement mais "à un moment donné" avant que le composant ne soit rendu à nouveau.

**Debuggin React applications** (31.03 – 09h12)

Un développeur passe beaucoup de temps à déboguer et à comprendre le code existant plutôt qu'à écrire du nouveau code. C'est pour cela que les bonnes pratiques et les outils de débogage sont très importants.

React est facile à déboguer pour les développeurs. Pour développer un site Web, il est important de garder la console développeur ouverte tout le temps et de garder le code et la page Web ouverts ensemble. Si quelque chose ne fonctionne pas, la console développeur peut aider à trouver le problème.

Sinon, et j’en ai fait plusieure fois l’experience, mon navigateur s’alume comme un sapin de noel :

Une image contenant texte

Description générée automatiquement

Il est préférable de ne pas écrire de code supplémentaire lorsque quelque chose ne fonctionne pas, mais plutôt de trouver et de corriger le problème immédiatement. Dans l'histoire du développement, il n'y a jamais eu de cas où le code qui ne compile pas se met soudainement à fonctionner après avoir écrit une grande quantité de code supplémentaire.

C’est une pratique que j’effectue déjà enormement en mobile et je le faisait déjà en Java Q3.

**Rules of hooks**

Il existe des règles et des limitations que nous devons respecter pour garantir que notre application utilise correctement les fonctions d'état basées sur les hooks. La fonction useState (ainsi que la fonction useEffect) ne doit pas être appelée à partir d'une boucle, d'une expression conditionnelle ou de tout autre endroit qui n'est pas une fonction définissant un composant. Cela est important pour garantir que les hooks sont toujours appelés dans le même ordre, sinon cela peut entraîner un comportement imprévisible de l'application. Voici des exemples d’utilisations :

Une image contenant texte

Description générée automatiquement

**Event Handling Revisited**

J’avais déjà commencé cette partie du cours plus tot dans ma synthese, mais la le cours revien dessus car c’était une partie difficil.

Voici un exemple simple d’application avec le composant App :

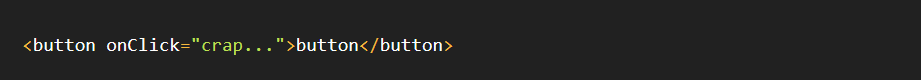
Une image contenant texte

Description générée automatiquement

Il faut que le clic sur le bouton réinitialise l'état stocké dans la variable "value". Pour ce faire, nous devons ajouter un gestionnaire d'événements au bouton.

Il est important que les gestionnaires d'événements soient des fonctions ou des références à des fonctions. Si le gestionnaire d'événements est défini comme une variable d'un autre type, le bouton ne fonctionnera pas correctement.

Si nous définissons notre gestionnaire d'événements sous forme de chaîne de caractères :

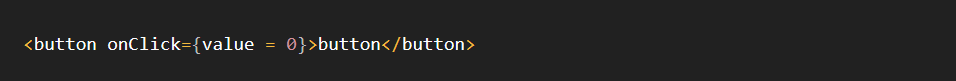


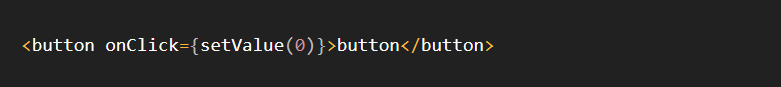
React dirait dans la console :

Une image contenant texte

Description générée automatiquement

Il y a plusieurs exemples qui sont eronnée, les voicis :



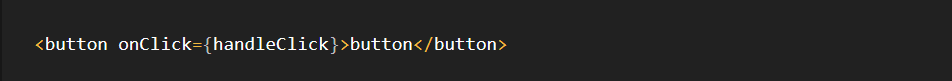


Le gestionnaire d'événements est désormais défini comme étant la fonction () => setValue (0). Cependant, il n'est pas recommandé de définir les gestionnaires d'événements directement dans l'attribut du bouton. Il est souvent préférable de les définir dans un endroit séparé.

Voici comment il faut faire pour que ça fonctionne :

Une image contenant texte

Description générée automatiquement



**Passing Event Handlers to Child Components**

Voici, un bouton dans son composant :

Une image contenant texte

Description générée automatiquement

Le composant reçoit la fonction de gestionnaire d'événements via la prop handleClick et le texte du bouton via la prop text.

Une image contenant texte

Description générée automatiquement

**Do Not Define Components Within Components** 9h55

Voici une modification de l’application dans le composant Display.

Une image contenant texte

Description générée automatiquement

Si on fait ca, il n’y a pas d’erreur, mais c’est une pratique a évitez de créer des composants à l'intérieur d'autres composants.

Cette méthode n'a pas d'avantages et peut causer plusieurs problèmes. Le plus gros problème est que React considère un composant créé à l'intérieur d'un autre comme un nouveau composant.

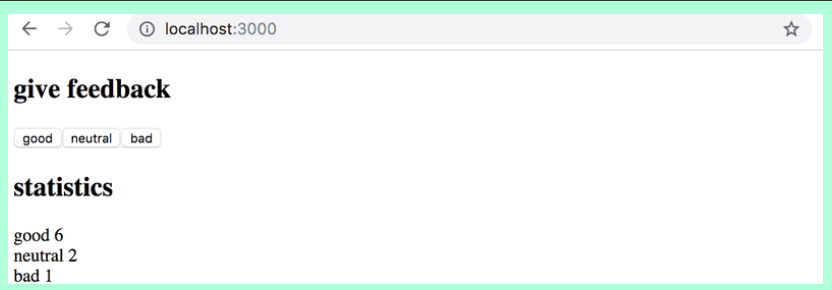
Il vaut mieux déplacer la fonction Display en dehors de la fonction App :

Une image contenant texte

Description générée automatiquement

**Exercices 1.6.-1.14.**

**1.6 : unicafe step1**

Voici a quoi doit ressembler le resultat : 

Sur ce site on peux avoir des explications pour realiser l’exercice pour le on click du bouton : https://blog.logrocket.com/react-onclick-event-handlers-guide/

J’ai reussi sans trop de difficulté a réalisé cela, voici mon code.

Une image contenant texte, capture d’écran, intérieur, écran

Description générée automatiquement

Ce code utilise le hook useState pour stocker le nombre de commentaires de chaque type. Lorsque je clique sur un bouton, la fonction handleFeedback est appelée avec le type de feedback correspondant (good, neutral ou bad). Cette fonction met à jour l'état feedback en ajoutant 1 à la propriété correspondante. Les nombres de commentaires sont affichés dans le composant div sous forme de texte.

**1.7 : unicafe step2**

Il faut continuer l’exercice precedent en ajoutant plus de statistiques, voici le resultat attendu :

Une image contenant texte

Description générée automatiquement

Voici ce que il y a dans mon code :

Une image contenant texte, capture d’écran, écran, ordinateur

Description générée automatiquement

J’ai ajouté une variable totale qui stocke le nombre total de commentaires reçus. La fonction handleFeedback met maintenant à jour cet état en plus de l'état feedback.

Ensuite, il y a les fonctions : calculateAverage pour calculer le score moyen des commentaires et calculatePositivePercentage pour calculer le pourcentage de commentaires positifs. Ces fonctions utilisent les données stockées dans feedback et total pour calculer les statistiques.

Ce genre de code, reste de niveau Bac1, mais le plus difficil est de le faire en React car c’est nouveau, cependant je pense que React est tres facilement comprehensible a l’aide du Java Script.

**1.8 : unicafe step3**

Voici ce que demande l’exercice : *« Refactorisez votre application afin que l'affichage des statistiques soit extrait dans son propre composant Statistiques. L'état de l'application doit rester dans le composant racine App. »*

Une image contenant texte

Description générée automatiquement

Pour arriver à ce resultat, c’était compliqué et j’ai alors demandé a ChatGpt, s’il pouvait factoriser mon code et il a pu le faire.

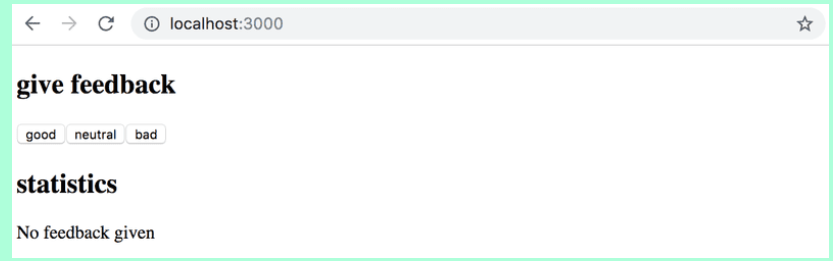
Voici ce qu’il a fait :

« Nous avons extrait le composant Statistiques dans un composant séparé et nous avons passé les données de feedback et total en tant que props. Le calcul des statistiques est maintenant effectué dans le composant Statistiques, mais l'état de l'application reste dans le composant racine App. »

**1.9 : unicafe step4**

Voici ce que demande l’exercice : *« Modifiez votre application pour n'afficher les statistiques qu'une fois les commentaires recueillis. »*

Voici le resultat attendu :



Et voici mon code, pour ça c’est tres simple, on verifie simplement si le nombre total de commentaires est supérieur à zéro et si oui, ca affiche les commentaire



**1.10 : unicafe step5**

Voici ce que demande l’exercice : *« Continuons à refactoriser l'application. Extrayez les deux composants suivants :*

*Bouton pour définir les boutons utilisés pour soumettre des commentaires*

*StatisticLine pour afficher une seule statistique, par ex. la note moyenne.*

*Pour être clair : le composant StatisticLine affiche toujours une seule statistique, ce qui signifie que l'application utilise plusieurs composants pour afficher toutes les statistiques : »*

Une image contenant texte

Description générée automatiquement

Le code commence a grandir, mais finalement tout ce que demande le cours dans les exercices sont des choses logique, des bonne pratique a avoir lorsque on code, voici a quoi va ressembler le code de cet exercice, toutefois l’affichage reste le meme que l’exercice precedent :

Une image contenant texte

Description générée automatiquement

**1.11 : unicafe step6**

Pour cet exercice il faut simplement afficher les statistiques dans un tableau HTML, et pour cela j’ai créé un nouveau composant StatisticLine qui affiche une ligne de statistiques sous forme de ligne de tableau HTML.

Des exemples de codes sont disponibles sur W3School : https://www.w3schools.com/html/html\_tables.asp

Nous avons aussi le site Officiel React pour en apprendre davantage sur React : https://fr.reactjs.org/

Une image contenant texte

Description générée automatiquement

**1.12: anecdotes step1** (31.03.2023 11h23)

Nous passons maitenant a un nouvel exercice : *« Développez l'application suivante en ajoutant un bouton sur lequel cliquer pour afficher une anecdote aléatoire du domaine du génie logiciel :* »

Une image contenant texte

Description générée automatiquement

Voici une bonne source dans lequel on pourait comprendre ce qui est demande dans l’exercice : https://stackoverflow.com/questions/68160328/how-to-display-a-random-text-from-array-on-a-button-press-in-react-native

Pour ce faire, ce n’est pas bien complique pour l’instant il fallait simplemejnt ajoute un bouton qui affiche une anecdote aleatoire, il a fallu modifier le return par :

Une image contenant texte

Description générée automatiquement

Ainsi il y aura un bouton visible qui va a chaque fois piocher parmi les anecdote, voici le resultat avec le code complet : Une image contenant texte, écran, capture d’écran, intérieur

Description générée automatiquement

**1.13 : anecdotes step2**

*« Développez votre application afin de pouvoir voter pour l'anecdote affichée. »*

Voici le resultat attendu : *Une image contenant texte

Description générée automatiquement*

On nous donne dans le cours quelque indication de commetnsotcker les votes d’anecdote dans un objet :

Une image contenant texte

Description générée automatiquement

Ou alors dans un tableau :

Une image contenant texte

Description générée automatiquement

Nous avons aussi une source directement dans le cours pour encore mieux comprendre comment faire l’exercice : <https://stackoverflow.com/questions/20222501/how-to-create-a-zero-filled-javascript-array-of-arbitrary-length/22209781#22209781>

En effet cet exercice est pour moi le plus complique que j’ai rencontre jusque maitenant, ce n’est pas pour rien qu’on a recu telement d’aide dans le cours pour le realiser mais finalement apres plusieurs essaie, j’ai enfin reussi, au debut il fallait crée un bouton pour pouvoir ajouter des votes a chaque anecdote, ca c’est pas complique mais des le debut il ne s’incrementait pas en fonction de l’anecdote qui était affiche.

Une image contenant texte

Description générée automatiquement

**1.14 : anecdotes step3** (31.03 12h09)

Cerise sur le gateau, le dernier exercice c’était exatement ce que ChatGpt a fait il faut afficher l’anecdote avec le plus de votes voici ce qui est demande :

Une image contenant texte

Description générée automatiquement

Toute fois ce n’était pas du tout complique comme explique precedement, voici la aprtie du code qui permetait de faire cela :

Une image contenant texte

Description générée automatiquement

**Part 2. a. Rendering a collection, modules.** (13.04.2023 – 21h45)

**Protip: Visual Studio Code snippets**

Avec Visual Studio Code, on peut facilement créer des "snippets", c'est-à-dire des raccourcis pour générer rapidement des morceaux de code qu'on utilise souvent, un peu comme "sout" dans Intelegi, mais au point ou j’en suis des fois j’ecrit juste « s » et ça mets le syso, c’est revolutionnaire sur inteligi, aparrarament il y en a sur vs code que je ne savais pas encore alors je suis alle checker ce site :

https://code.visualstudio.com/docs/editor/userdefinedsnippets#\_creating-your-own-snippets

Meme si c’est en rapport avec vsCode je trouve cela interresant, on peut également trouver des extraits de code utiles et prêts à l'emploi sous forme de plugins pour Visual Studio Code, sur la marketplace.

**Rendering collections**

Selon le cours, cette partie est destiné a faire le « frontend », la logique d’applicartion coté navigateur, voila comment commence le fichier App.js : Une image contenant texte

Description générée automatiquement

Index js :

Une image contenant texte

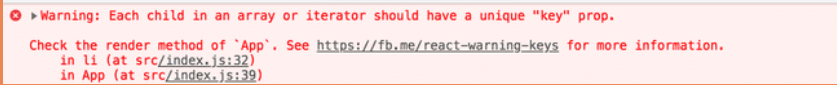
Description générée automatiquement

Chaque note a du texte, une date et une valeur booléenne pour marquer si elle est importante ou pas, ainsi qu'un identifiant unique.

L'exemple ci-dessus fonctionne car il y a que trois notes dans le tableau.

**Attribut clé**

Faut faire a attention a la facon dont c’est coder aussi, car meme si l’application semble fonctionner, il y a un avertissement dans la console :



Comme indiqué sur la page React dans le message d'erreur, les éléments d'une liste générés par la méthode map doivent avoir une clé unique, qui est un attribut appelé "clé".

Pour résoudre ce problème, Il faut ajouter des clés au elements de la liste, voici comment il faut faire : Une image contenant texte

Description générée automatiquement

**Map**

Je decide de parler aussi de la methode map car c’est crucial pour le reste du cours, l’application contient un tableau « notes » :

Une image contenant texte

Description générée automatiquement

On ajoute ce bout de code a la fin du fichier : Une image contenant texte

Description générée automatiquement

1, 2, 3 sera affiché dans la console. La méthode map crée toujours un nouveau tableau en utilisant une fonction donnée en paramètre pour mapper les éléments du tableau d'origine vers de nouveaux éléments.

**Exercices 2.1.-2.5** (28.04.2023 – 08h23)

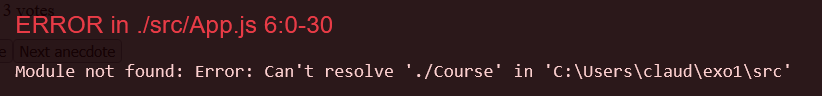
**2.1: course info étape6**

Il faut terminer le code pour le rendu du contenu des exercice 1.1 à 1.5.

Il faut ajouter ces 3 partie dans le code et avor un certain affichage.

Une image contenant texte

Description générée automatiquement

Malheureusement je n’ai pas reussi cet exercice, je n’ai pas reussi a comprendre cet erreur : 

Mais finalement cetait une bete erreur d’import que il ne falais pas, je l’ai donc suprimer et ensuite cela a fonctionner facilement pour l’affichage, voici le resultat :

Une image contenant texte

Description générée automatiquement

**2.2: course info étape7**

Dans cet exercice il falait afficher la somme des exercices, et comme vous pouvez le voir ce n’était pas compliqué je l’ai fait directement dans l’exercice précedent.

Une image contenant texte

Description générée automatiquement

**2.3: course info étape8**

J’ai passé cet exercice car c’était le meme que le precedent juste fait le faire d’une autre facon, j’ai donc juger cela pas tres utilee.

**2.4: course info étape9**

Pour cet exercice il falait allait plus loin, dans l’affichage, ce n’était pas bien compliqué, j’ai repris le code qui est sur le cours, ajouter dans mon ncode et fait quelque modification dessus et le tour est joué, voici le resultat final :

Une image contenant texte, écran, capture d’écran

Description générée automatiquement**Une image contenant texte

Description générée automatiquement**

**2.5 : separation des modules**

Dans cet exercice il a falu separer Course en tant que module séparé pour avoir le meme resultat.

J’ai donc crée Course.js, ou j’ai ajouter ce bout de code, que j’ai bien evidement enlever de App.js et simplement remplacer par Course key comme ceci : Une image contenant texte

Description générée automatiquement

**Une image contenant texte, capture d’écran, écran

Description générée automatiquement**

**Part 2. b. Formulaires** (28.04.2023 – 08h45)

**Exercice 2.6.-2.10.**

**Exercice 2.6**

Pour cet exercice il faut créer un répertoire téléphonique simple en reprenant déjà un code disponible sur le cours, L'état newName est destiné à contrôler l'élément d'entrée du formulaire, Parfois, il peut être utile de rendre l'état et d'autres variables sous forme de texte à des fins de débogage.

Une image contenant texte

Description générée automatiquement

Pour l’instant, si j’ajoute un nom, rien ne s’affiche, mais en modifiant le code pour simplement afficher le contenu qui est dans levent il falait ajouter ce bout de code :

Une image contenant texte

Description générée automatiquementUne image contenant texte

Description générée automatiquement

**Exercice 2.8 : phonebook, étape 3**

Enfaite javais mal compris l’exerice precedant et il falait simplementer ajouter un nom par defaut dans le repertoire, chose que j’avais fait pour « Riggio » mais, j’etais déjà allé plus loin en ajoutant le nom claudiu lorsquon on clique sur le bouton « addPerson » donc l’exercice je l’ai déjà fait.

**Exercice 2.9 : phonebook, étape 4**

Il faut réutiliser ce code afin d’afficher plusieurs nom déjà dans le repertoire, et ajouter un filtre de recherche :

**Une image contenant texte

Description générée automatiquement** **Une image contenant texte

Description générée automatiquement**

**Exercice 2.10 : phonebook, étape 5**

Comme pour l’exercice 2.5, le but va etre de separer les modules pour avoirun code propre, mais j’ai decider de ne pas le faire vu que c’était fait pour le 2.5, c’est le meme principe.

**Part 2. c. Obtenir des données du serveur** (28.04.2023 – 09h21)

Pour le moment la partie 2 est focalisé sur la partie « frontend », la partie « backend » sera travailler dans la partie 3, mais dans le point c, nous alons commencer petit a petit a aller dans cette direction.

Pour cette partie il a fallu instaler un JSON Server pour avoir un serveur a disposition.

Une image contenant texte

Description générée automatiquement

Voici a quoi ressemble l’environnement d’exécution de développement.

Une image contenant diagramme

Description générée automatiquement

Je vais passer directement aux exercices sans resumer la partie théorique car les exercices sont beaucoup plus pertinents a expliqué que la theorie qui est déjà dans le cours.

Plus de cela les exercices propose dans cette partie sont tout simplement la suite du phonebook, mais il falait simplement avoir la théorie du serveur pour pouvoir éffectuer les exericice suivant, et en effet on a clairement besoin de la théorie pour cela.

**Exercice 2.11.-2.13.**

**Exercice 2.11 : phonebook, étape 6**

Il faut stocker les 4 personnes du repertoir dans un fichier db.json que j’ai fait.

Une image contenant texte

Description générée automatiquement

Démarrez json-server sur le port 3001 et il faut que le serveur renvoie la liste des personnes en allant à l'adresse <http://localhost:3001/persons> dans le navigateur.

J’ai rencontré quelque difficulté à réaliser cet exercice, j’ai fait les manipulation necesaire a la realisation de l’exercice mais sans savoir pourquoi rien ne s’affiche dans mon json, mon terminal indique du coup cela :

Une image contenant texte

Description générée automatiquement

**Exercice 2.12 : countries, étape 1** (28.04.2023 – 10h57)

Si on repart depuis l’exerice telephonique, il faut simplement le modifier en ajoutant une api qui fournit des données pour diférents pays dans un format lisible par machine : <https://restcountries.com>.

*« Créer une application, dans laquelle on peut consulter les données de différents pays. L'application devrait probablement obtenir les données de l’end point* [*all*](https://restcountries.com/v3.1/all)*.*

*L'interface utilisateur est très simple. Le pays à afficher est trouvé en tapant une requête de recherche dans le champ de recherche.*

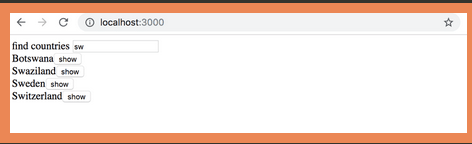
*S'il y a trop de pays (plus de 10) qui correspondent à la requête, l'utilisateur est invité à préciser sa requête »* *Une image contenant texte

Description générée automatiquement*

Le niveau de l’exercice est déjà asse haut, il faut beaucoup de connaisance du langage et de React pour reussir a avoir cette solution, c’est pour cela que pour cette solution, j’ai utilisé Chat Gpt pour m’aider a avoir cette reponse final, en lui expliquant le contexte et l’utilisation de l’api il a pu me donner une reponse.

**Exercice 2.13 : countries, étape 2**

Ici, il va falloir simplement ameliorer l’exercice precedent de sorte que le nom de plusieurs pays sont affiche sur la page, il y a un bouton à côté du nom du pays, qui, lorsqu'il est pressé, affiche la vue pour ce pays :



Malheureusement apres avoir essayer de faire cela je n’ai eu que des erreurs de compilation énorme dont je ne comprenais pas trop, j’ai alores abandoner l’exercice pour aller plus vite et pas rester loquer longtemps sur l’exercice, J’ai passer l’exercice 2.14 car je n’ai pas reussi l’exercice 2.13.

**Part 2. d. Modification des données côté serveur** (28.04.2023 – 10h24)

**Exercice 2.15.-2.18**

**Exercice 2.15 phonebook, étape7**

*« Revenons à notre application de répertoire. Actuellement, les numéros ajoutés au répertoire ne sont pas enregistrés sur un serveur principal. Corrigez cette situation. »*

*Une image contenant texte

Description générée automatiquement* *Une image contenant texte

Description générée automatiquement*

**Exercice 2.16 phonebook, étape8**

*« Extrayez le code qui gère la communication avec le backend dans son propre module en suivant l'exemple présenté précédemment dans cette partie du support de cours. »*

*Une image contenant texte

Description générée automatiquement*

**Exercice 2.17 phonebook, étape9**

*« Permettre aux utilisateurs de supprimer des entrées du répertoire. La suppression peut être effectuée via un bouton dédié pour chaque personne dans la liste du répertoire. Vous pouvez confirmer l'action de l'utilisateur en utilisant la méthode* [*window.confirm*](https://developer.mozilla.org/en-US/docs/Web/API/Window/confirm)*: »*

***Une image contenant texte

Description générée automatiquement***

Malheureusement je n’ai pas reussi a realiser cet exerce, je ne sais aps pourquoi il ne trouvait pas le module, pourtant j’utilisais le meme pour l’exercice precedent, du coup l’exercice 10 je n’ai pas pu le faire car c’était la suite du 9.

***Une image contenant texte

Description générée automatiquement***

**Part 2. e. Styliser vos applicatios React** (28.04.2023 – 11h10)

Je vais parler de cette partie car je la trouve evidement moin compliqué que la precedente mais aussi car c’est un sujet qui m’interesse beaucoup, j’ai toujours aimer l’affichage greaphique de ce que je code. L'apparence de l’application actuelle est pas ouf. Avant de passer à la partie suivante, je vais apprendre comment ajouter des styles à une application React. Il existe plusieurs façons de procéder, mais je vais commencer par ajouter du CSS à l'ancienne : dans un seul fichier, sans utiliser de préprocesseur. Je vais créer un nouveau fichier index.css dans le répertoire src, puis je vais l'importer dans le fichier index.js pour l'ajouter à l'application.





Si je modifie le contenu du fichier index.css, React peut ne pas le remarquer automatiquement, donc je devrai actualiser le navigateur pour voir les modifications. Les règles CSS sont composées de sélecteurs et de déclarations. Le sélecteur indique les éléments auxquels la règle doit être appliquée.

Je peux utiliser divers types de sélecteurs CSS pour faire correspondre des éléments de différentes manières. Si j’applique des styles à chaque note de mon application, je peux utiliser le sélecteur "li", car toutes les notes sont enveloppées dans des balises "li" :

Une image contenant texte

Description générée automatiquement

**Message d’erreur amélioré**

Le composant est assez simple :

Une image contenant texte

Description générée automatiquement

Si la valeur de la prop message est null, alors rien n'est rendu à l'écran, et dans d'autres cas, le message est rendu à l'intérieur d'un élément div. J’ajoute un nouvel élément d'état appelé errorMessage au composant App. Initialiser avec un message d'erreur afin de tester notre composant :

Une image contenant texte

Description générée automatiquementUne image contenant texte

Description générée automatiquement

Voici la logique pour afficher le message d’erreur :

Une image contenant texte

Description générée automatiquement

Le résultat ressemble à ceci :

Une image contenant texte

Description générée automatiquement

**Styles en ligne**

Je peux écrire des styles directement dans le code en tant qu'inline-styles avec React.

Lorsque je définis des styles en ligne dans React, je fournis à chaque composant ou élément un ensemble de propriétés CSS sous la forme d'un objet JavaScript via l'attribut "style".

Les règles CSS sont définies légèrement différemment en JavaScript qu'en CSS normal. Par exemple, si je veux donner à un élément la couleur verte et une police en italique d'une taille de 16 pixels, en CSS cela ressemblerait à ceci :

**Une image contenant texte

Description générée automatiquement**

Mais en tant qu'objet de style en ligne React, cela ressemblerait à ceci :

Une image contenant texte

Description générée automatiquement

Ensuite, nous pourrions ajouter un "bloc inférieur" à notre application en créant un composant Footer et en définissant les styles en ligne suivants pour celui-ci :

Une image contenant texte

Description générée automatiquement

**Exercices 2.19.-2.20.** (28.04.2023 – 11h35)

Le principe de l’exercice est de faire exactement comme dans la partie 2, je l’ai donc passé, en plus que ce n’est que du css finalement, je pense asse bien avoir expliqué comment ça fonctionnait pour React j’ai donc passer ces 2 exercice de css qui permetait simplement de styliser un message d’erreur.

**Part 3. a. Node.js et Express.** (05.05.2023 –)