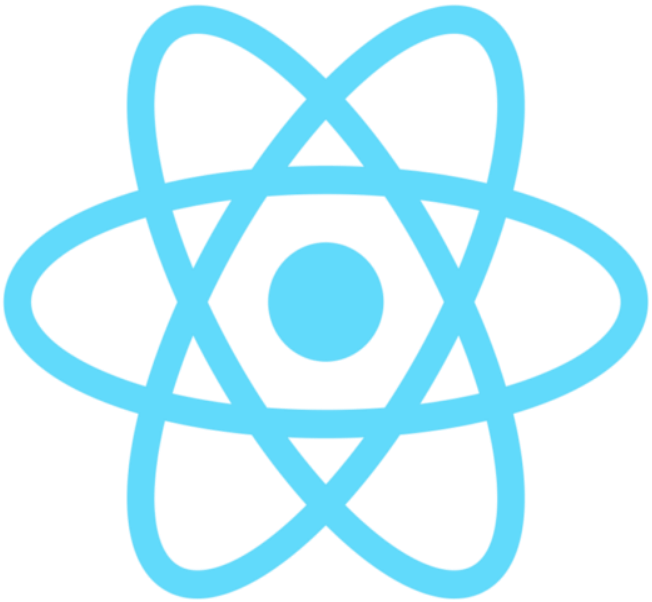
[Nom de la société]

Rapport Projet web

Thierno FALL



Identifiant github : th12341th

Tâches effectuées

Lors de ce projet j'ai réalisé les tâches suivantes :

- Réalisation de la carte des map
- Définition des couleurs sur la carte
- Redimensionnement des données et images SVG
-

Gestion de la version GIT

On fait un pull request pour la validation de chaque changement ou apport par les autres membres du projet. Une fois approuvé, on fait la mise a jour du code

Solutions choisies : SVG, xlink :title, Map.querySelectorAll

Pour la construction de la carte, j'ai choisi d'utiliser SVG pour avoir une carte dynamique et interactive. SVG est pour moi la meilleure solution pour ce type de carte du notamment du côté interactif. Le code aussi plus lisible et compréhensible.

J'ai utilisé xlink pour avoir la possibilité de cliquer sur un département pour voir plus d'informations liées au département concerné en le combinant avec google map.

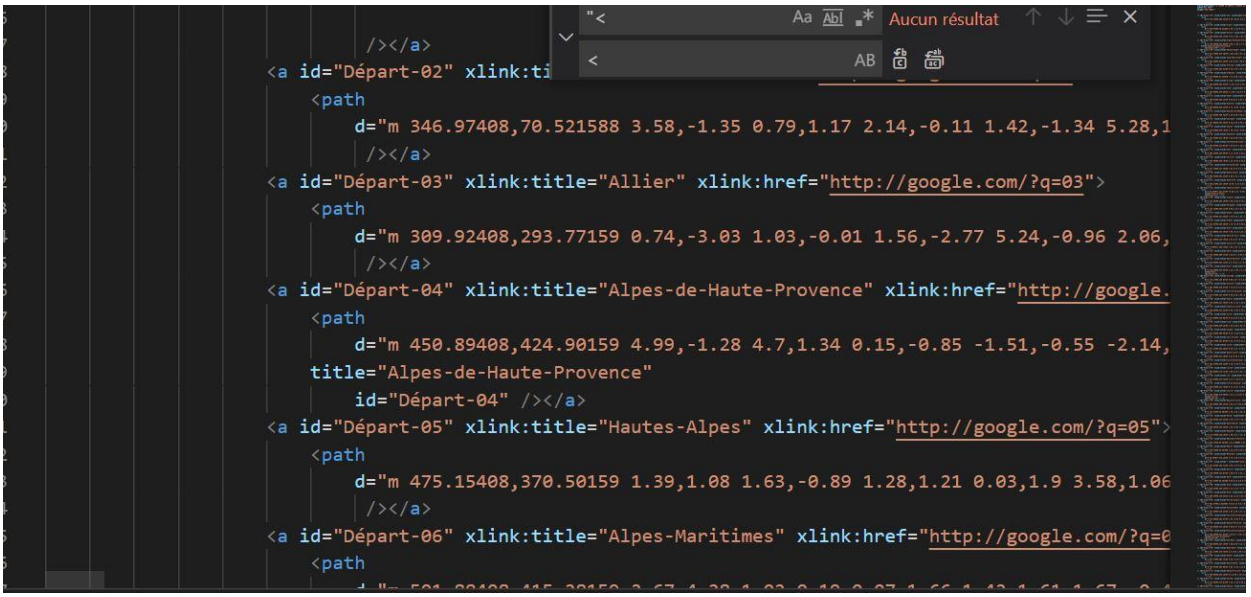
Map.querySelectorAll pour sélection les formes géométriques et les liens

Raisonnement

Tout d'abord j'a recupere un fichier svg sur le site https://www.amcharts.com/ sur les donnees des differents departements de France.
Ensuite j'ai contruit mes fichier css, html et js qui ont respectievemt pour but :

Travailler sur le style et rendu notament des couleur de fond de la carte

- Definir les donnees SVG qu'on veut afficher sur la carte
- Definir comment les afficher



Pour afficher les départements, j'ai créé des liens et des listes dans le but de pour voir cliquer dessus sur l'un ou sur l'autre.
Donc j'ai imbriqué les id et les titles pour pouvoir cet effet de sélection.

Ensuite dans le js, j'ai définit des fonctions d'activation des airs et des fonctions pour les liens dans le but d'activer la correspondance mutuelle, c'est-à-dire de permettre a l'utilisateur de choisir de cliquer sur la carte ou sur les noms des département.

```
const activeArea = function(id){

    map.querySelectorAll('.is-active').forEach(function(item){
        item.classList.remove('is-active')
    })


    document.querySelector("#list-" + id).classList.add('is-active')
    document.querySelector("#Départ-" + id).classList.add('is-active')
    console.log("Nombre de cas positifs")
}


paths.forEach(function (path){
    path.addEventListener('mousecenter', function(e){

        const id = this.id.replace('Départ-', '')
        activeArea(id)
    })
})
```

Solution

- Ain
- Aisne
- Allier
- Alpes-de-Haute-Provence
- Hautes-Alpes
- Alpes-Maritimes
- Ardèche
- Ardennes
- Ariège
- Aube
- Aude
- Aveyron
- Bouches-du-Rhône
- Calvados
- Cantal
- Charente
- Charente-Maritime
- Cher
- Corrèze
- Corse-du-Sud
- Haute-Corse
- Côte-d'Or
- Côtes-d'Armor
- Creuse
- Dordogne
- Doubs
- Drôme
- Eure
- Eure-et-Loir
- Finistère
- Gard
- Haute-Garonne

Difficultés rencontrées

Les difficultés rencontrées sont la l’affichages des trois couleurs en fonction d’un seuil établi pour le nombre cas par département dans la base de données.

Affichage des nombres de cas total pour chaque département sur la carte en fonction du tableau de bord.

Afficher le menu déroulant lorsqu’on est sur la carte ou sur la liste simultanément.

Temps de développement / tâches

Gestion des données SVG : 15 mn

Code js : 5h00

Graphe : 4h00

Rendu : 2h

Code

Solution élégante

```
function MonthsSelect(props) {
  const months = props.months;
  const listMonths = months.map((month) => {
    return (
      <DropdownItem
        key={months.indexOf(month) + 1}
        onClick={handleMonthFilter}
        value={month}
      >
        {month === "" ? "All Month" : month}
      </DropdownItem>
    );
  });
  return <DropdownMenu>{listMonths}</DropdownMenu>;
}
```

```
<Col>
  <h2>Mois</h2>
  <Dropdown isOpen={dropdownOpenMonth} toggle={toggleMonth}>
    <DropdownToggle caret> {monthFilter}</DropdownToggle>
    <MonthsSelect months={months} />
  </Dropdown>
</Col>
```

Je suis assez fier de ce code, c'est une fonction qui permet de créer un « DropDownMenu » avec les mois associés que je fais passer par les props. J'utilise la fonction map qui va créer un composant « DropDownItem » pour chaque mois. L'avantage est que c'est un composant réutilisable, et qui est dynamique (va se mettre à jour automatiquement s'il y a un changement dans la liste de mois)

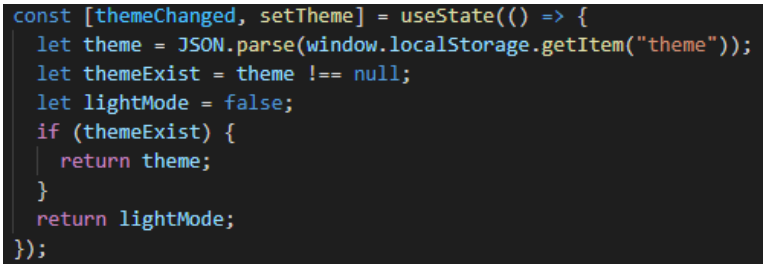
Solution à optimiser

Gestion des hooks : filtre et theme

```
useEffect(() => {
  const updateData = async () => {
    await monthQuery.refetch();
    await regionsQuery.refetch();
    await dataCovidQuery.refetch();
  };

  updateData();
  if (!monthQuery.isLoading && !monthQuery.isError) {
    setMonths(monthQuery.data);
  }
  if (!regionsQuery.isLoading && !regionsQuery.isError) {
    setRegions(regionsQuery.data);
  }
  if (!dataCovidQuery.isLoading && !dataCovidQuery.isError) {
    onChange(dataCovidQuery.data);
  }

  changeData(
    !monthQuery.isError ||
      !regionsQuery.isError ||
      !monthQuery.isLoading ||
      !regionsQuery.isLoading ||
      !dataCovidQuery.isError ||
      !dataCovidQuery.isLoading
  );
  onLoadingData(
    !regionsQuery.isLoading ||
      !monthQuery.isLoading ||
      !dataCovidQuery.isLoading
  );
  onError(
    dataCovidQuery.isError || regionsQuery.isError || monthQuery.isError
  );
}, [
  genderFilter,
  monthFilter,
  ageFilter,
  yearFilter,
```



```
const [themeChanged, setTheme] = useState(() => {
  let theme = JSON.parse(window.localStorage.getItem("theme"));
  let themeExist = theme !== null;
  let lightMode = false;
  if (themeExist) {
    return theme;
  }
  return lightMode;
});
```

Ici les deux solutions que j'ai réalisées peuvent être optimisées. On pourrait créer des hooks personnalisés qui permettrait que le code puisse être réutilisable. De plus si un autre développeur est amené à lire mon code, cela pourrait être assez fastidieux de devoir lire toute la logique derrière ce code, il serait mieux d'utiliser un hook personnalisé et de le nommer correctement pour comprendre directement son utilité.

Avec le thème on pourrait écrire useLocalTheme(theme), et dans le hook qui permet de mettre à jour les données (lors du changement de filtre) on pourrait créer un hook useUpdateData(listOfData) qui permettrait de mettre à jour les données dans la liste