REACT JS

Présentation :

React (aussi appelé **React.js** ou **ReactJS**) est une bibliothèque open source JavaScript pour créer des interfaces utilisateurs. Elle est maintenue par Meta (anciennement Facebook) ainsi que par une communauté de développeurs individuels et d'entreprises depuis 2013.

Le but principal de cette bibliothèque est de faciliter la création d'application web monopage, via la création de composants dépendant d'un état et générant une page (ou portion) HTML à chaque changement d'état.

React est une bibliothèque qui ne gère que l'interface de l'application, cette interface étant considérée comme la vue dans le modèle MVC. Elle peut ainsi être utilisée avec une autre bibliothèque ou un framework MVC comme AngularJS. La bibliothèque se démarque de ses concurrents par sa flexibilité et ses performances, en travaillant avec un DOM virtuel et en ne mettant à jour le rendu dans le navigateur qu'en cas de nécessité

Installation :

lancer la commande **npx create-react-app fisheye**

Problème

L'erreur « Module non trouvé : Erreur : Impossible de résoudre « web-vitals » » est due à l'absence du package **web-vitals** . Assurez-vous que le package web-vitals est installé dans devDependencies.

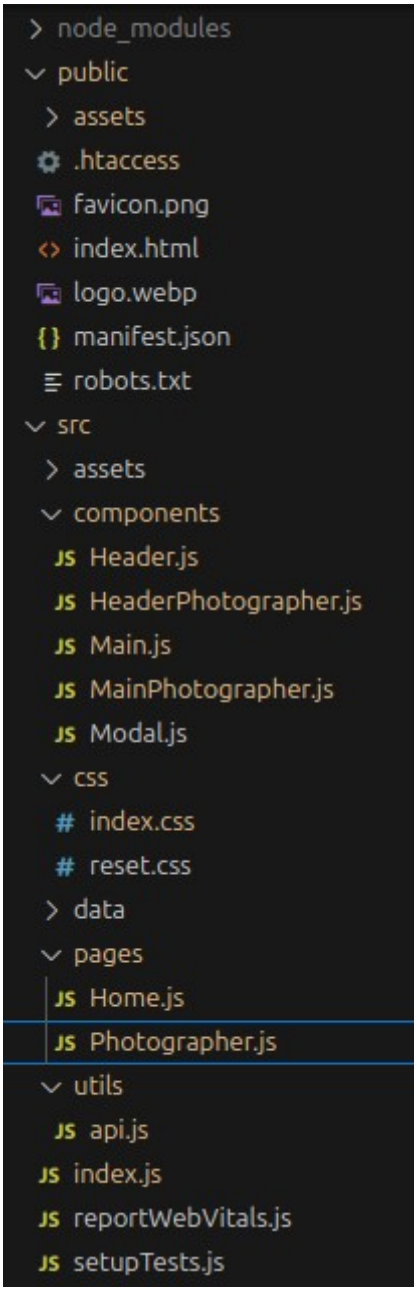
Installer Web-Vitals

Pour installer les web-vitals, exécutez simplement la commande suivante dans le répertoire racine du projet et réexécutez le serveur
npm install --save-dev web-vitals
Si l'erreur n'est pas résolue, essayez de supprimer vos fichiers node_modules et package-lock.json (pas package.json), réexécutez npm install

cd fisheye
npm start (ouvre un hôte local sur le navigateur localhost:3000)

Si on clone le projet via github, il faut installer node_modules via la commande **npm install**
La web-vitals bibliothèque est une petite bibliothèque modulaire (~ 2 Ko, brotli'd) permettant de mesurer toutes les mesures **Web Vitals** sur des utilisateurs réels, d'une manière qui correspond précisément à la manière dont elles sont mesurées par Chrome et signalées à d'autres outils Google (par exemple, **Chrome User Experience Report** , **Page Speed Insights** , **Search Console's Speed Report**).
La bibliothèque prend en charge tous les **Core Web Vitals** ainsi qu'un certain nombre d'autres mesures utiles pour diagnostiquer les problèmes de performances **des utilisateurs réels**.

1. Création de l’arborescence (fichiers/dossiers)



2. Création du router

Pour créer le système de routes sur ReactJS, installer le package ‘react dom router’ en exécutant la ligne de commande dans le terminal :

npm i react-router-dom

Dans le fichier index.js :

```
import React from "react";
import ReactDOM from "react-dom/client";
import { BrowserRouter, Routes, Route } from "react-router-dom";
import Home from "./pages/Home";
import Photographer from "./pages/Photographer";
import reportWebVitals from "./reportWebVitals";
import "./css/index.css";

export default function App() {
  return (
    <BrowserRouter>
      <Routes>
        <Route index element={<Home />} />
        <Route path="/photographer/:id/" element={<Photographer />} />
      </Routes>
    </BrowserRouter>
  );
}

// reportWebVitals(console.log());

const root = ReactDOM.createRoot(document.querySelector(".container"));
root.render(<App />);
```

Explications :

Le fichier index.js est le point d’entré de l’application, le fonction App() est exécuter et le résultat injecté dans le fichier ‘public/index.html’ dans l’élément HTML préalablement sélectionnée dans le DOM.

Pour le router, il faut importer les composants du packages ‘react-dom-router’ (BrowserRouter, Routes, Route). Dans le fonction App(), on appelle le composant <BrowserRouter> pour définir les définir les différentes routes (URL) du projet . Il y a routes :

- La route ‘Home’ qui est le point d’entrée sur le navigateur, qui appel le composant page <home /> (localhost:3000)

- La route ‘photographer’ qui est la page de galerie du photographe, qui appel le composant <Photographer /> (localhost:3000/photographer/930), elle contient un paramètre de recherche ‘:id’ qui permet de transmettre l’identifiant unique du photographe (ex :?id=)

2. Création du composant Header.js

```
export default function Header() {
  return (
    <header className="header">
      <a href="/" className="link__logo">
        <img
          src={"/assets/images/logo.webp"}
          alt="logo"
          className="header__logo"
        />
      </a>
      <h2 className="header__slogan">Nos photographes</h2>
    </header>
  );
}
```

3. Création du composant Main.js

```
export default function Main() {
  return (
    <>
      <main className="main">
        <div className="card">
          <div className="card__picture__circle">
            <a href="" className="card__link">
              <img src="" className="card__picture" />
              <h3 className="card__title"></h3>
            </a>
          </div>
          <div className="card__content">
            <p className="card__localisation"></p>
            <p className="card__slogan"></p>
            <p className="card__price">€/jour</p>
          </div>
        </div>
        ;
      </main>
    </>
  );
}
```

4.Création du composant Home.js

```jsx
import Header from "../components/Header";
import Main from "../components/Main";

const Home = () => {
  return (
    <>
      <Header />
      <Main />
    </>
  );
};

export default Home;
```

Explications :

Il faut importer les composants Header et Main afin de pouvoir les exécuter directement dans le composant Home. C'est avec l'extension de React JSX, qui permet de structurer le rendu des composants à l'aide d'une syntaxe familière à de nombreux développeurs. Il est similaire en apparence au HTML.

Le JSX ne peut renvoyer qu'un seul composant à la fois, c'est la raison pour laquelle nous avons englober les composant de chevron vide <> </>

5. Création de l'API api.js, récupération de données JSON

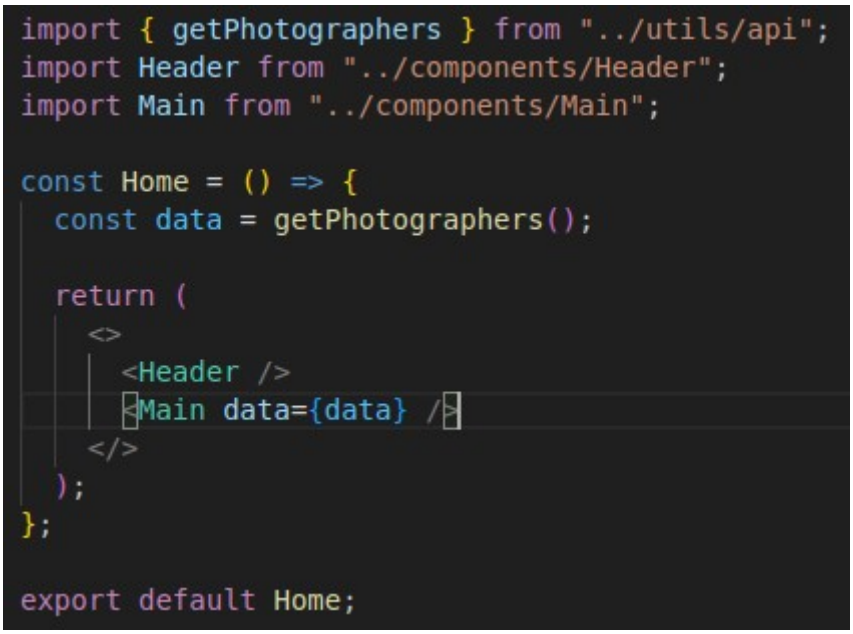
```jsx
import data from "../data/photographers.json";

export const getPhotographers = () => {
  return data.photographers;
};
```

Explications :

Pour récupérer toutes les données du fichiers photographers.json, il suffit simplement de l'importer et de stocker les données dans un objet 'data'.

La fonction getPhotographers permet d'obtenir toutes les données de l'ensemble des photographes (nom, portrait, tagline, price etc.)

6. Importation et transmission des données



```
import { getPhotographers } from "../utils/api";
import Header from "../components/Header";
import Main from "../components/Main";

const Home = () => {
  const data = getPhotographers();

  return (
    <>
      <Header />
      <Main data={data} />
    </>
  );
};

export default Home;
```

On importe la fonction getPhotographers, l’exécutons et stockons les données dans la constante ‘data’, qui contient l’ensemble des données des photographes, ces données doivent être transmise au composant Main afin de les traiter pour avoir un retour visuel. data={} est un paramètre que nous définissons.

7. Traitement et affichage des données

```
export default function Main(photographers) {
  const dataPhotographers = photographers.data;
  return (
    <>
      <main className="main">
        {dataPhotographers.map((photographer) => (
          <div className="card" key={photographer.id}>
            <div className="card__picture__circle">
              <a
                href={"/photographer/" + photographer.id}
                className="card__link"
              >
                <img
                  src={
                    "/assets/images/photographers/thumbnails/" +
                    photographer.portrait
                  }
                  className="card__picture"
                />
                <h3 className="card__title">{photographer.name}</h3>
              </a>
            </div>
            <div className="card__content">
              <p className="card__localisation">
                {photographer.city}, {photographer.country}
              </p>
              <p className="card__slogan">{photographer.tagline}</p>
              <p className="card__price">{photographer.price}€/jour</p>
            </div>
          </div>
        ))}
        ;
      </main>
    </>
  );
}
```

La variable de réception ‘photographers’ contient l’ensemble des données des photographes, elle réceptionne les données envoyées comme paramètre dans le composant <Main />

```
<Main data={data} />
```

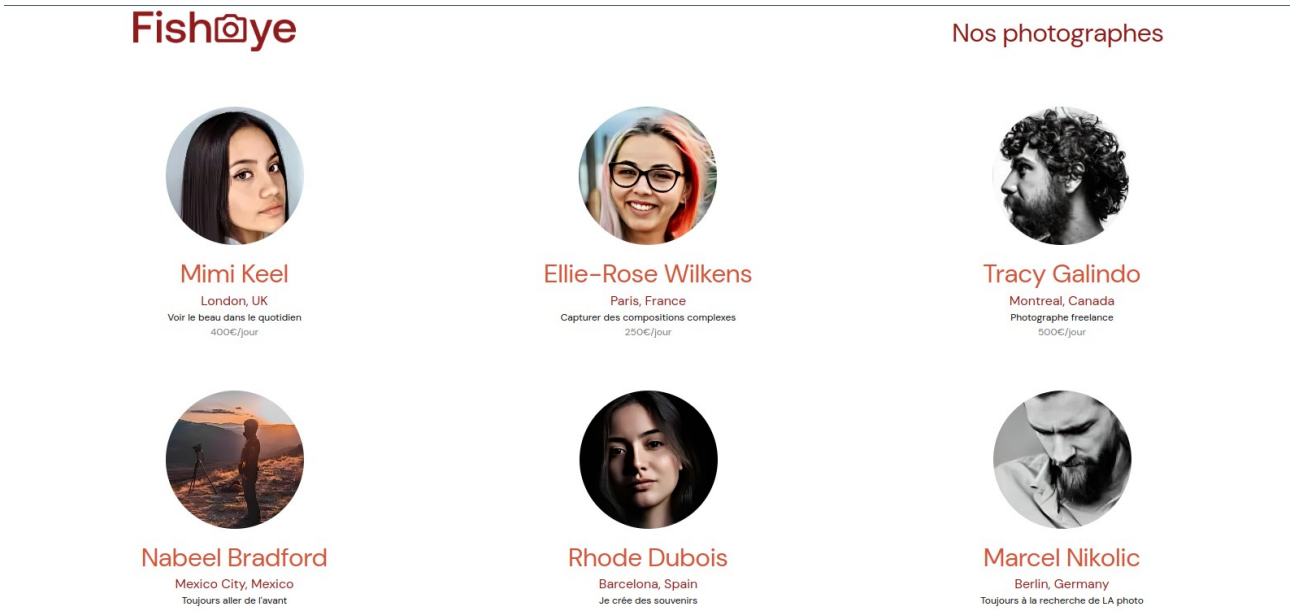
```
export default function Main(photographers) {
  const dataPhotographers = photographers.data;
```

La méthode JavaScript map() permet de créer un nouveau tableau en appliquant une fonction spécifiée à chaque élément du tableau d'origine . Cette méthode ne modifie pas le tableau d'origine et est particulièrement utile pour transformer ou traiter des données stockées dans des tableaux.

La variable ‘photographer’ est une variable de réception que nous définissons, elle receptionne un array/object contenant les données d’un photographe à chaque fois que la fonction map() est éxecutée. Il ne reste plus qu’à piocher dans le array/object afin d’afficher les données des photographes (ex : nom du photographe → photographer.name).

Pour l’attribut href du lien, il faut s’assurer de définir la bonne route qui redirige vers la page photograph, qui contient l’identifiant unique du photographe, cela permettra de récupérer les données en dans l’API en fonction de l’id transmis dans l’URL.

Voilà ! La première page Home est terminée ! Voici le rendu visuel !



Page galerie photographe

8. Création du composant HeaderPhotographer.js

```
import React from "react";
import { FontAwesomeIcon } from "@fortawesome/react-fontawesome";
import { faXmark } from "@fortawesome/free-solid-svg-icons";

export default function Header() {
  return (
    <>
      <div className="modal">
        <div className="header__modal">
          <div className="header__title">
            <p>Contactez-moi</p>
            <p></p>
          </div>
          <FontAwesomeIcon icon={faXmark} className="icone__close__modal" />
        </div>
        <div className="modal__form">
          <form action="" method="post" className="form__contact">
            <label htmlFor="firstName">Prénom</label>
            <input
              type="text"
              name="firstName"
              id="firstName"
              className="input__text"
            />
            <label htmlFor="lastName">Nom</label>
            <input
              type="text"
              name="lastName"
              id="lastName"
              className="input__text"
            />
            <label htmlFor="email">Email</label>
            <input
              type="text"
              name="email"
              id="email"
              className="input__text"
            />
```

```
            <label htmlFor="message">Message</label>
            <textarea
              name="message"
              id="message"
              rows="6"
              className="input__textarea"
            ></textarea>

            <button className="submit">Envoyer</button>
          </form>
        </div>
      </div>
      <header className="header__photographer">
        <a href="/" className="link__logo">
          <img
            src={"/assets/images/logo.webp"}
            alt="logo"
            className="header__logo"
          />
        </a>
        <section className="header__presentation">
          <div className="header__presentation__content">
            <h1 className="photographer__name"></h1>
            <p className="phtographer__location"></p>
            <p className="phtographer__tagline"></p>
          </div>
          <div className="contact__block">
            <button className="photographer__contact">Contactez-moi</button>
          </div>
          <div className="picture__block">
            <img src="" alt="" className="photographer__thumbnail" />
          </div>
        </section>
      </header>
    </>
  );
}
```

Explications :

Pour ajouter les icônes, il faut ajouter la bibliothèque Fontawesome à notre projet, documentation ici : https://docs.fontawesome.com/v5/web/use-with/react

Dans un terminal, se rendre dans le dossier du projet fisheye et executer les commandes suivantes :

```
npm i --save @fortawesome/fontawesome-svg-core
npm install --save @fortawesome/free-solid-svg-icons
npm install --save @fortawesome/react-fontawesome
```

Importer et exécuter les composants Fontawesome et les icônes dans le composant HeaderPhotographer.

```
import { FontAwesomeIcon } from "@fortawesome/react-fontawesome";
import { faXmark } from "@fortawesome/free-solid-svg-icons";
```

```
<FontAwesomeIcon
  icon={faXmark}
  className="icone__close__modal"
/>
```

9. Création du composant MainPhotographer.js

```
import React from "react";
import { FontAwesomeIcon } from "@fortawesome/react-fontawesome";
import { faChevronDown, faHeart } from "@fortawesome/free-solid-svg-icons";

export default function Main() {
  return (
    <>
      <main className="main__photographer">
        <div className="medias__sort">
          <p className="text__sort">Trier par</p>
          <div className="dropdown">
            <button className="button__dropdown">
              <p className="button__dropdown__text">Popularité</p>
              <FontAwesomeIcon icon={faChevronDown} />
            </button>
            <div className="dropdown__content">
              <div className="dropdown__item">
                <span className="dropdown__link">
                  <hr className="separator" />
                  <span className="link__text">Date</span>
                </span>
              </div>
              <div className="dropdown__item">
                <span className="dropdown__link">
                  <hr className="separator" />
                  <span className="link__text">Titre</span>
                </span>
              </div>
            </div>
          </div>
        </div>
```

```
        <section className="galery">
          <figure className="card">
            <img src="" alt="" className="galery__card__picture" />
            <video
              autoPlay
              loop
              muted
              className="galery__card__picture"
              src=""
            ></video>

            <figcaption className="card__legend">
              <p className="picture__title"></p>
              <div className="cards__likes">
                <span className="nbLikes"></span>
                <FontAwesomeIcon icon={faHeart} className="icone__heart" />
              </div>
            </figcaption>
          </figure>
        </section>
        <div className="block__likes">
          <p>
            <span className="nb__total__likes"></span>
            <FontAwesomeIcon icon={faHeart} className="icone__heart__likes" />
          </p>
          <p className="likes__price">€ /jour</p>
        </div>
      </main>
    </>
  );
}
```

10. Création du composant Photographer.js

```
import * as React from "react";
import Header from "../components/HeaderPhotographer";
import Main from "../components/MainPhotographer";

const Photographer = () => {
  return (
    <>
      <Header />
      <Main />
    </>
  );
};

export default Photographer;
```

11. Création de la fonction getPhotographerById et getMediasPhotographerById dans le fichier api.js, récupération de données JSON

```
export const getPhotographerById = (id) => {
  const photographers = getPhotographers();

  const data = photographers.filter((photographer) => photographer.id == id);

  return data[0];
};
```

La fonction reçoit en argument dans la variable ‘id’ (variable de réception) l’identifiant unique d’un photographe transmis dans l’URL (ex : /photographer/930).

Le but est de filtrer les données de l’ensemble des photographes stockés dans la constante ‘photographers’. La fonction filter() renvoi des données en fonction d’une condition, ici il faut que l’id du photographe soit égal à l’id transmis dans l’URL. Cela retourne un tableau de données contenant les données d’1 photographe.

```
export const getMediasPhotographerById = (id) => {
  const medias = getMedias();

  const data = medias.filter((media) => media.photographerId == id);

  return data;
};
```

Pour la fonction **getMediasPhotographerById**, c’est le même principe, sauf que la fonction filter() retourne l’ensemble des médias associés au photographe (images/vidéo) en fonction de l’id transmis dans l’URL et l’id du photographes représenté par media.photographerId.

12. Importation et transmission des données

```
import { getMediasPhotographerById, getPhotographerById } from "../utils/api";
import Header from "../components/HeaderPhotographer";
import Main from "../components/MainPhotographer";
import * as React from "react";
import { useParams } from "react-router-dom";

const Photographer = () => {
  const { id } = useParams();
  const medias = getMediasPhotographerById(id);
  const photographer = getPhotographerById(id);

  return (
    <>
      <Header photograph={photographer} />
      <Main mediasById={medias} photograph={photographer} />
    </>
  );
};

export default Photographer;
```

13. Traitement et affichage des données

HeaderPhotographer.js

```
<section className="header__presentation">
  <div className="header__presentation__content">
    <h1 className="photographer__name">{dataPhotograph.name}</h1>
    <p className="phtographer__location">
      {dataPhotograph.city}, {dataPhotograph.country}
    </p>
    <p className="phtographer__tagline">{dataPhotograph.tagline}</p>
  </div>
  <div className="contact__block">
    <button className="photographer__contact">
      Contactez-moi
    </button>
  </div>
  <div className="picture__block">
    <img
      src={
        "/assets/images/photographers/thumbnails/" +
        dataPhotograph.portrait
      }
      alt=""
      className="photographer__thumbnail"
    />
  </div>
</section>
```

MainPhotographer.js

```
export default function Main(datas) {
  const medias = datas.mediasById;
  const photographer = datas.photograph;
```

```
{mediaState.map((media, index) => (
  <figure className="card" key={media.id}>
    {media.image ? (
      <img
        src={
          "/assets/images/photographers/samplePhotos-Medium/" +
          photographer.name +
          "/" +
          media.image
        }
        alt=""
        className="galery__card__picture"
      />
    ) : (
      <video
        autoPlay
        loop
        muted
        className="galery__card__picture"
        src={
          "../assets/images/photographers/samplePhotos-Medium/" +
          photographer.name +
          "/" +
          media.video
        }
      ></video>
    )}
```

```
        <figcaption className="card__legend">
          <p className="picture__title">{media.title}</p>
          <div className="cards__likes">
            <span className="nbLikes">{media.likes}</span>
            <FontAwesomeIcon
              icon={faHeart}
              className="icone__heart"
              onClick={() => handleLikes(media.id)}
            />
          </div>
        </figcaption>
      </figure>
    ))}
  </section>
  <div className="block__likes">
    <p>
      <span className="nb__total__likes">{nbTotalLikes}</span>
      <FontAwesomeIcon icon={faHeart} className="icone__heart__likes" />
    </p>
    <p className="likes__price">{photographer.price}€ /jour</p>
  </div>
</main>
```

14. Traitement du dropdown

La base

L'utilisation des Hooks doit respecter deux règles fondamentales :

- Les Hooks ne peuvent être appelés que dans un composant fonctionnel de React ou un autre Hook.
- Les Hooks doivent toujours être appelés dans le même ordre. Dans votre composant, si les Hooks sont exécutés dans des conditions, boucles ou sous fonctions, l'ordre d'appel doit toujours être le même. Il est fortement recommandé de garder l'appel des Hooks au premier niveau du composant.

À partir de là vous pouvez appeler plusieurs fois le même dans un composant. Libre à vous de les utiliser dans l'ordre que vous souhaitez.

useState

C'est avec cette fonction qu'on va gérer l'état (comme son nom l'indique) du composant.

```
const [state, setState] = useState(initialState);
```

Voilà la syntaxe proposée par React. Qu'avons-nous là ?

- `[state, setState]` : Le tableau retourné par la fonction `useState` contient la valeur de l'état (`state`) et la fonction permettant de l'éditer (`setState`).
- `state` : la valeur de l'état que l'on va pouvoir utiliser partout dans le composant
- `setState` : la fonction qui va permettre de mettre à jour l'état (`setState(newValue)`)
- `initialState` : la valeur initiale de l'état. Si aucune valeur n'est fournie, l'état est initialisé à `null`.

Et vous pouvez en créer plusieurs dans le même composant.

Intéressons nous au dropdown des filtres, nous allons créer un state qui permet de définir si le dropdown doit être ouvert ou fermé.

```
const [open, setOpen] = React.useState(false);
const handleOpen = () => {
  setOpen(!open);
};
```

Nous avons définit un état initial à **false**, dans ce cas le dropdown doit être fermé.

La constante open contient l'état initial, c'est à dire **false**. Le but est de modifier cet état à l'évenement clic sur le bouton du dropdown. Pour cela nous créons une fonction handleOpen() qui se déclenche au clic sur le bouton du dropdown, il faut donc ajouter l'attribut **onClick** sur le bouton.

```
<button
  className={
    open ? "button__dropdown active__radius" : "button__dropdown"
  }
  onClick={handleOpen}
```

La fonction **setOpen** permet de modifier l'état du state, au clic sur le bouton, la première fois, en argument de la fonction, nous avons ajouté comme paramètre !open, ce qui signifie **N'EST PAS FALSE**, donc si ce n'est pas **FALSE**, c'est **TRUE** ! Nous venons de changer la valeur l'état,

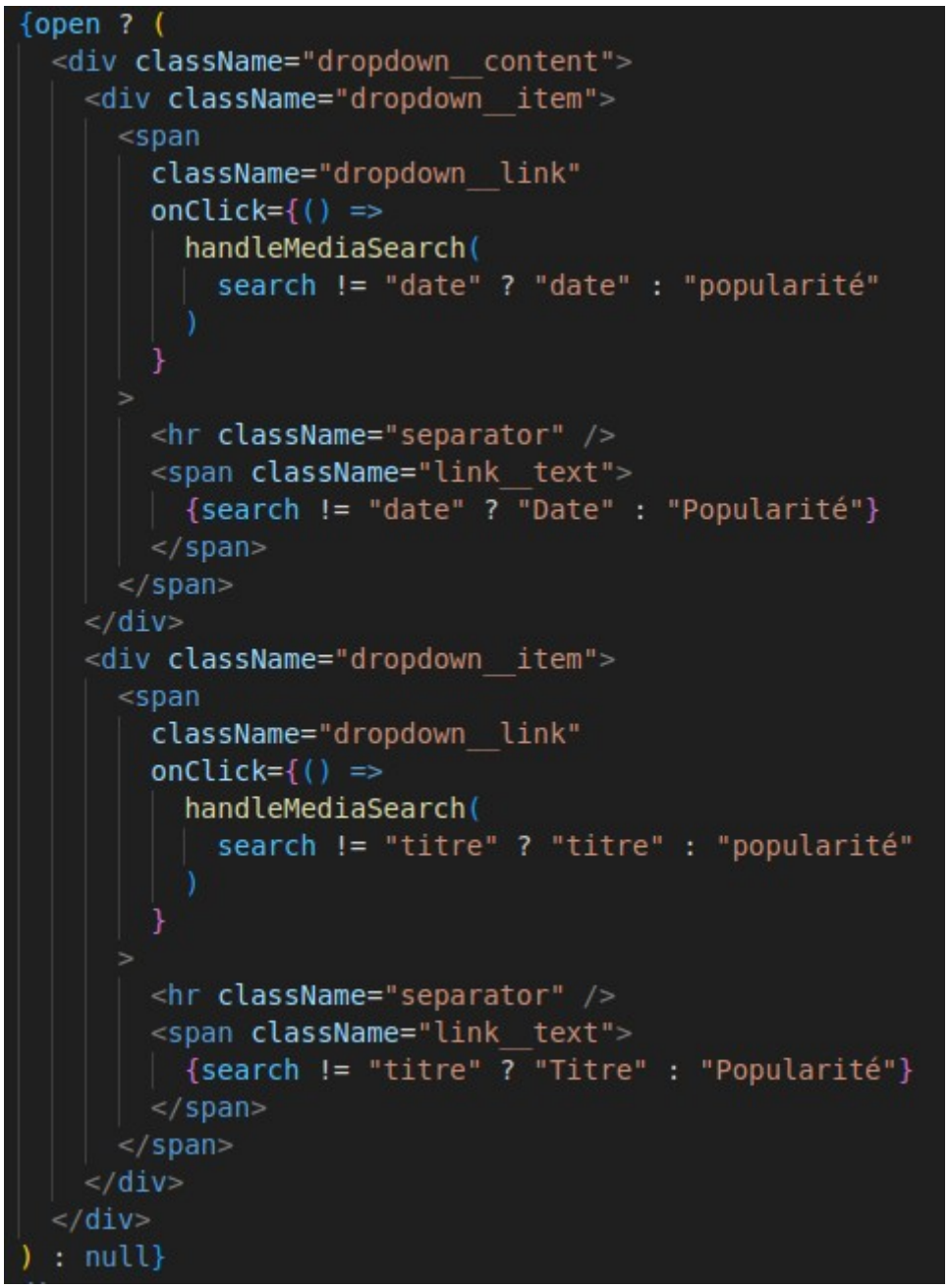
la constante **open** contient maintenant la valeur **TRUE** ! Si nous re-cliquons sur le bouton, ce qui signifie, donc si ce n'est pas **TRUE**, c'est **FALSE**! L'état a donc 2 uniques valeurs, **TRUE** et **FALSE** ! Lorsque l'état est à **TRUE**, nous ouvrons le dropdown et à **FALSE** nous le fermons, grâce à une condition dans le return du JSX.

```
{open ? (
  <div className="dropdown__content">
    <div className="dropdown__item">
      <span
        className="dropdown__link"
        onClick={() =>
          handleMediaSearch(
            search != "date" ? "date" : "popularité"
          )
        }
      >
        <hr className="separator" />
        <span className="link__text">
          {search != "date" ? "Date" : "Popularité"}
        </span>
      </span>
    </div>
    <div className="dropdown__item">
      <span
        className="dropdown__link"
        onClick={() =>
          handleMediaSearch(
            search != "titre" ? "titre" : "popularité"
          )
        }
      >
        <hr className="separator" />
        <span className="link__text">
          {search != "titre" ? "Titre" : "Popularité"}
        </span>
      </span>
    </div>
  </div>
) : null}
```

15. Gestion du filtre des medias

Pour le filtre des médias (Titre, Date et Popularité), nous allons créer un nouveau **Hook** avec la fonction **useState()**, qui gère l'état des médias.

```
const [mediaState, setMediaState] = React.useState(medias);
const [search, setSearch] = React.useState("popularité");
const handleMediaSearch = (search) => {
  const mediasCopy = [...medias];

  switch (search) {
    case "titre":
      mediasCopy.sort((a, b) => a.title.localeCompare(b.title));
      break;
    case "popularité":
      mediasCopy.sort((a, b) => b.likes - a.likes);
      break;
    case "date":
      mediasCopy.sort((a, b) => new Date(b.date) - new Date(a.date));
      break;
  }

  setMediaState(mediasCopy);
  setOpen(false);
  setSearch(search);
};
```



```
{open ? (
  <div className="dropdown__content">
    <div className="dropdown__item">
      <span
        className="dropdown__link"
        onClick={() =>
          handleMediaSearch(
            search != "date" ? "date" : "popularité"
          )
        }
      >
        <hr className="separator" />
        <span className="link__text">
          {search != "date" ? "Date" : "Popularité"}
        </span>
      </span>
    </div>
    <div className="dropdown__item">
      <span
        className="dropdown__link"
        onClick={() =>
          handleMediaSearch(
            search != "titre" ? "titre" : "popularité"
          )
        }
      >
        <hr className="separator" />
        <span className="link__text">
          {search != "titre" ? "Titre" : "Popularité"}
        </span>
      </span>
    </div>
  </div>
) : null}
```

L’état initial des médias est contenu dans la constante **medias**, c’est le résultat de la fonction **getMediasPhotographerById()** déclarée dans le fichiers **api.js**.

La fonction handleMediaSearch() est exécutée au clic sur un élément du dropdown (Titre, Date et Popularité), elle reçoit en argument le paramètre, la categorie de filtre (Titre, Date et Popularité).

Pour mettre à jour l’état des médias, nous sommes obligé de créer une copie des médias initial

```
const mediasCopy = [...medias];
```

Ensuite la condition **Switch** permet de trier les données en fonction du choix cliqué dans le dropdown. Une fois les médias triés, nous envoyons en argument de **setMediaState** les médias mis à jour, ce qui permet de modifier l'état des médias ! La retour visuel est instantané sans rechargement de page !!

Au clic sur un élément du dropdown, nous modifions l'état du dropdown, **setOpen(false)**, afin de le fermer.

Afin de pouvoir gérer l'affichage des éléments du dropdown, nous créons un dernier **hook useSate()**.

```
const [search, setSearch] = React.useState("popularité");
```

L'état initial est popularité, à chaque clic sur un élément du dropdown, nous modifions la valeur de l'état grâce à la méthode setSearch() exécutée dans la méthode handleMediaSearch(). Elle prend en paramètre la catégorie de recherche (Titre, Date, Popularité). Ce qui nous permet de gérer l'affichage des liens du dropdown.

```
<span
  className="dropdown__link"
  onClick={() =>
    handleMediaSearch(
      search != "date" ? "date" : "popularité"
    )
  }
>
  <hr className="separator" />
  <span className="link__text">
    {search != "date" ? "Date" : "Popularité"}
  </span>
</span>
```

16. Gestion de la LIGHTBOX

Pour développer la lightbox, nous allons installer un package sur notre projet qui permet facilement avec quelques paramétrages de créer une Lightbox fonctionnelle, sans que nous ayons besoin de tous développer from scratch.

Documentation : https://yet-another-react-lightbox.com/

Pour installer le package, executer la ligne de commande dans un terminal, dans le dossier du projet :

npm install yet-another-react-lightbox

Importer tout les composants et feuilles de styles CSS nécessaire.

```
import Lightbox from "yet-another-react-lightbox";
import Video from "yet-another-react-lightbox/plugins/video";
import Slideshow from "yet-another-react-lightbox/plugins/slideshow";
import Counter from "yet-another-react-lightbox/plugins/counter";
import Thumbnails from "yet-another-react-lightbox/plugins/thumbnails";
import { Captions } from "yet-another-react-lightbox/plugins";
import Fullscreen from "yet-another-react-lightbox/plugins/fullscreen";
import Zoom from "yet-another-react-lightbox/plugins/zoom";
import Download from "yet-another-react-lightbox/plugins/download";
import "yet-another-react-lightbox/plugins/thumbnails.css";
import "yet-another-react-lightbox/plugins/captions.css";
import "yet-another-react-lightbox/styles.css";
import "yet-another-react-lightbox/plugins/counter.css";
```

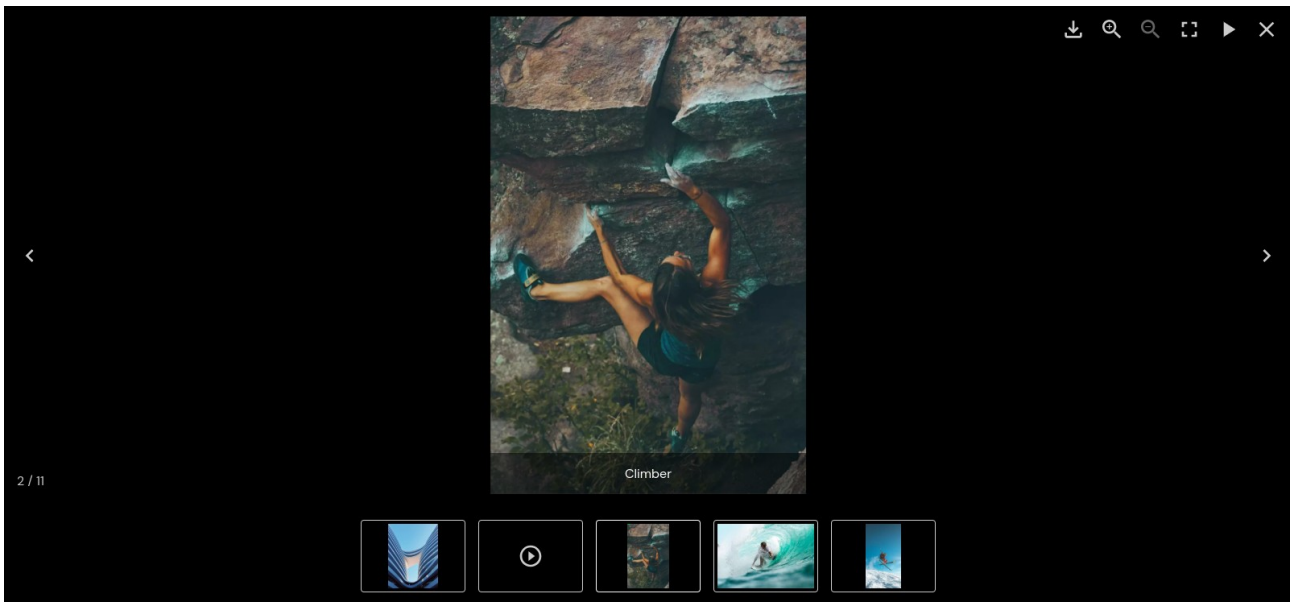
Définissez les sources des images / vidéos et stockés les dans une constante.

```
const currentPath = `/assets/images/photographers/samplePhotos-Medium/${photographer.name}/`;
const slides = mediaState.map(({ image, video, title }) => ({
  ...(image
    ? { src: currentPath + image, description: title }
    : {
        type: "video",
        width: 1280,
        height: 720,
        // poster: currentPath + video,
        // autoPlay: true,
        loop: true,
        muted: true,
        description: title,
        sources: [
          {
            src: currentPath + video,
            type: "video/mp4",
          },
        ],
      }),
}));
```

Ensuite inclure le composant paramétré dans le return JSX.

```
const currentPath = `/assets/images/photographers/samplePhotos-Medium/${photographer.name}/`;
const slides = mediaState.map(({ image, video, title }) => ({
  ...(image
    ? { src: currentPath + image, description: title }
    : {
        type: "video",
        width: 1280,
        height: 720,
        // poster: currentPath + video,
        // autoPlay: true,
        loop: true,
        muted: true,
        description: title,
        sources: [
          {
            src: currentPath + video,
            type: "video/mp4",
          },
        ],
      }),
}));
```

Voilà !! Vous avez une superbe Lightbox !! Elle n’est pas jolie ??



Les fonctionnalités principal sont développées, il reste l’incrémentation des likes et l’affichage de la modal de formulaire de contact.

À vous de jouer !!!