un composant = HTML + JS + CSS
React DOM.render est une méthode react qui permet d'associer un composant et un contenu à un endroit

- Un **framework JS** est un **ensemble de classes, fonctions et utilitaires** qui nous facilitent la création d'applications pour les navigateurs ou mobiles.
- L'un des outils les plus populaires, **React**, qui est une bibliothèque aussi bien qu'un framework, permet de **créer des interfaces utilisateurs**.
- L'approche technique de React est de créer du **code modulaire**, à base de **composants réutilisables**.
- Trois des avantages de React sont sa **communauté**, sa **documentation** et ses **opportunités professionnelles**.
- Vous savez maintenant comment **transformer un simple fichier de HTML en React** – et avez créé votre premier composant !

package.json

```
"dependencies": {
    "@testing-library/jest-dom": "^5.16.5",
    "@testing-library/react": "^13.4.0",
    "@testing-library/user-event": "^13.5.0",
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "react-scripts": "5.0.1",
    "web-vitals": "^2.1.4"
  },
```

index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);
```

- Les développeurs utilisent des outils automatisés pour faciliter leur expérience de développement.
- Create React App (CRA) est la boîte à outils créée par Facebook, qui reste encore la référence pour initier un projet React.
- Un projet initialisé avec CRA possède toujours :
    - un fichier index.html qui est le template où vivra notre app React ;
    - un package.json qui liste les dépendances et les scripts ;
    - un fichier index.js dans lequel notre app React est initialisée, et greffée au HTML.
- CRA s'exécute avec l'aide d'un gestionnaire de paquet (dans ce cours, yarn).
- Webpack permet d'importer simplement les fichiers entre eux.

31/05/2023:
Problématique :
J’ai rencontré des complications et des pb de compatibilité car j’avais la version 18.16.0 de Node.js et elle n’était pas adaptée pour pouvoir utiliser correctement React 17.

Solution apportée:
j'ai tout d’abord essayer de changer la versions de node jai pas reussi ensuite jai arreter et j'ai suivi les indications qu'ils m'ont donnés pour simplement modifier une section de code pour que se soit compatible avc react 17 jai fait comme ce qui est écrit sur le tuto a la fin j'ai fait un npm install (sur la-maison-jungle) et ensuite un npm start pour qu'elle lance l'appli ça ne fonctionné pas
Ensuite, j'ai décidé de remettre que le code que j'avais avant les modifs j'ai refais un npm start et pareil ça ne lance rien (alors que ça fonctionnait avant)

Résultat:
Au final, j’ai supprimer la version 18.16.0 de Node.js pour installer la version précédente (17.9.1) après cela j’ai tout recommencer en décidant de re-créer un dossier et de refaire un ‘npm create react-app …’ grâce à ça j'ai pu reprendre le tuto correctement

05/06/2023:

- L'attribut **className** permet de **préciser une classe à un élément React** pour lui indiquer du CSS.
- Le fichier CSS correspondant peut être **importé directement** dans un fichier .js.
- L'attribut style permet d'**intégrer du style directement**, on appelle cela du *inline style*.
- Les **images** sont importées par React grâce à **Webpack**. Il suffit d'importer l'image souhaitée.

01/06/2023:
Pb:
./src/index.js Module not found: Can't resolve 'react-dom/client'
soluce:

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';

ReactDOM.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
  document.getElementById('root')
);

// If you want to start measuring performance in your app, pass a function
// to log results (for example: reportWebVitals(console.log))
reportWebVitals();
```

06/06/2023
Key : c’est un attribut que nous devons passer sous forme de string dès que nous créons une liste de composants, react s’en sert pour manipuler le DOM virtuel. Quand il y a une modification sur un élément. La key indique à réact si

l'élément à été soit ajouté sur la liste ou supprimer ou bien partiellement modifier .

L'opérateur (ternaire) conditionnel est le seul opérateur JavaScript qui comporte trois opérateurs. Cet opérateur est fréquemment utilisé comme raccourci pour la déclaration de Instructions/if...else.

07/06/2023:

- À partir d'une liste de données, map() permet de créer une liste de composants React.
- La prop key est indispensable dans les listes de composants.
- Si vous voulez éviter les bugs, la prop key doit :
    - être unique au sein de la liste ;
    - perdurer dans le temps.
- La best practice pour créer une key est d'utiliser l' id unique associée à une donnée, et de ne pas vous contenter d'utiliser l'index de l'élément dans la liste.
- Une condition ternaire permet d'afficher un élément ou un autre dans le JSX, répondant à la condition "if… else...".
- Il existe d'autres manières de créer des conditions en React, notamment en sortant les conditions du JSX.

Les props sont utilisées pour transmettre des données et configurer des composants dans React et d'autres bibliothèques similaires. Elles permettent une communication entre les composants en passant des valeurs d'un niveau supérieur (parent) à un niveau inférieur (enfant).

08/06/2023

Les props sont donc des **objets que l'on peut récupérer dans les paramètres de notre composant fonction**.

- Les props sont des objets que l'on peut récupérer dans les paramètres de notre composant fonction.
- Il existe deux syntaxes pour assigner une valeur à une prop :
    - les guillemets pour les string ;

    - les accolades pour tout le reste : nombres, expressions JavaScript, booléen, etc.
- La déstructuration est une syntaxe permettant de déclarer une variable en l'affectant directement à la valeur d'un objet (ou tableau).
- Une *prop* est :
    - toujours passée par un composant parent à son enfant ;
    - considérée en lecture seule dans le composant qui la reçoit.
- La prop children est renseignée en imbriquant les enfants dans le parent : <Parent><Enfant /></Parent>.
- children est utile lorsqu'un composant ne connaît pas ses enfants à l'avance.

un événement est une **réaction** à une **action** émise par l'utilisateur, comme le clic sur un bouton ou la saisie d'un texte dans un formulaire.

09/06/2023:

- En React, un événement s'écrit dans une balise en camelCase, et on lui passe la fonction à appeler.
- Contrairement au JS, dans la quasi totalité des cas, vous n'avez pas besoin d'utiliser addEventListener.
- React passe un événement synthétique en paramètre des fonctions de callBack. Cet événement synthétique est similaire à un événement passé en natif dans le DOM, sauf qu'il est compatible avec tous les navigateurs.
- Il existe deux grandes manières de gérer les formulaires : les formulaires contrôlés ou non contrôlés. L'utilisation des formulaires contrôlés est recommandée.
- Les formulaires contrôlés sauvegardent la valeur des champs dans le state local, et accèdent à la data entrée par l'utilisateur avec onChange.
- Les formulaires contrôlés permettent de filtrer le contenu, ou d'afficher un message d'erreur en fonction de la data qui est entrée par l'utilisateur.

12/06/2023:

le state local va nous permettre de préserver les données qu'un utilisateur va renseigner (ex : afficher ou cacher le contenue de son panier grâce à un clique)

Bienvenue dans la magie du state ! 🧙 Notre composant Cart est maintenant devenu un *stateful component*, grâce à `useState` .

`useState` est un hook qui permet d'ajouter le state local React à des composants fonctions.

Un hook est **une fonction qui permet de « se brancher » (*to hook up*) sur des fonctionnalités React**

Les Hooks sont une nouvelle fonctionnalité introduite dans React 16.8 qui permet aux développeurs de bénéficier de fonctionnalités de React telles que l'état local sans avoir à écrire de classes. Ils offrent une API plus simple et directe pour gérer l'état et d'autres aspects de React.

L'exemple donné dans l'introduction montre l'utilisation du premier Hook appelé "useState". Il permet de déclarer une variable d'état dans un composant fonctionnel. Dans l'exemple, une variable d'état appelée "count" est déclarée avec une valeur initiale de 0. La fonction "setCount" est utilisée pour mettre à jour cette variable d'état en ajoutant 1 à chaque fois qu'un bouton est cliqué.

Les Hooks ne rendent pas obsolètes les classes en React, mais ils offrent une alternative plus simple et plus expressive pour gérer l'état et la logique dans les composants. Ils résolvent des problèmes tels que la réutilisation de la logique d'état entre les composants et la complexité croissante des composants basés sur des classes.

Les Hooks sont rétrocompatibles, ce qui signifie que vous pouvez les utiliser dans de nouveaux composants sans avoir à réécrire votre code existant. Ils sont également complètement optionnels, vous pouvez donc choisir de les utiliser ou non selon vos besoins.

Les avantages des Hooks sont présentés dans la documentation, notamment la facilité de réutilisation de la logique d'état, la simplification des composants complexes et la réduction de la confusion liée aux classes. Les Hooks offrent une approche plus fonctionnelle pour gérer l'état et les effets de bord.

Il est recommandé d'adopter progressivement les Hooks et de les utiliser dans de nouveaux composants non critiques afin de s'habituer à leur utilisation. Les classes continueront à être prises en charge et il n'est pas nécessaire de migrer tout votre code existant vers les Hooks.

La documentation propose également des exemples et une FAQ pour répondre aux questions fréquemment posées sur les Hooks. La prochaine étape recommandée est de passer à la page suivante pour apprendre les Hooks par l'exemple.

15/06/2023

- `useEffect`permet d'effectuer des effets : cela permet à notre composant d'exécuter des actions après l'affichage, en choisissant à quel moment cette action doit être exécutée.
- Le hook `useEffect` est appelé après chaque rendu de votre composant. Il est possible de préciser quelle modification de donnée déclenche les effets exécutés dans useEffect, avec le tableau de dépendances.
- Un tableau de dépendances vide permet d'exécuter un effet uniquement au premier rendu de votre composant.

pour mettre à jour l'état dans React, vous devez utiliser la fonction de mise à jour renvoyée par `useState` plutôt que d'appeler directement `useState`.

16/06/2023:

```
import naruto from '../assets/naruto.jpg'

import one_piece from '../assets/one_piece.jpg'

import dbz from '../assets/dbz.jpg'

import hxh from '../assets/hxh.jpg'

import snk from '../assets/snk.jpg'

import kimetsu from '../assets/kimetsu.jpg'

import vinland from '../assets/vinland.jpg'

import death from '../assets/death.jpg'
```

```
import mha from '../assets/mha.jpg'



//MangaList est un tableau de donnée (sur les mangas)

export const MangaList = [

    {

        name: 'Naruto',

        cover: naruto

    },

    {

        name: 'One piece',

        cover: one_piece

    },



    {

        name: 'Dragon Ball Z',

        cover: dbz

    },

    {

        name: 'Death Note',

        cover: death

    },

    {

        name: 'Shingeki No Kyojin',

        cover: snk

    },
```

```
    {
        name: 'My Hero Academia',
        cover: mha
    },
    {
        name: 'Kimetsu No Yaiba',
        cover: kimetsu
    },
    {
        name: 'Hunter X Hunter',
        cover: hxh
    },


    {
        name: 'Vinland Saga',
        cover: vinland
    }
]
```

19/06/2023 :

@tailwind base;
@tailwind components;
@tailwind utilities;

21/06/2023

1. Au départ, la valeur de selectedManga est initialisée à null. Cela indique qu'aucun manga n'est sélectionné.
2. Lorsque l'utilisateur clique sur le bouton "Voir les détails" d'un manga dans le composant MangaCard, la fonction handleMangaClick est appelée. Elle reçoit l'ID du manga en paramètre.
3. À l'intérieur de handleMangaClick, la variable selected est mise à jour en utilisant la méthode find (La méthode find exécute la fonction callback une fois pour chaque élément présent dans le tableau jusqu'à ce qu'elle retourne une valeur vraie) pour rechercher le manga correspondant à l'ID dans le tableau MangaData. Une fois le manga sélectionné trouvé, il est stocké dans selectedManga.
4. En conséquence de la mise à jour de selectedManga, la logique conditionnelle dans le rendu du composant App décide d'afficher soit le composant MangaDetail avec le manga sélectionné, soit le composant Home avec la liste de mangas.

Ainsi, selectedManga joue un rôle essentiel dans le suivi du manga sélectionné par l'utilisateur et dans l'affichage des détails du manga lorsque celui-ci est sélectionné.

22/06/2023

j'ai ajouté un état local showDetails à l'aide du hook useState. Lorsque vous cliquez sur le bouton "Voir les détails", la fonction handleMangaClick est appelée. Elle inverse la valeur de showDetails en utilisant la fonction setShowDetails. Ainsi, les informations détaillées seront affichées si showDetails est true et masquées si showDetails est false.

J'ai également modifié le libellé du bouton pour refléter l'état actuel.

j’ai ajouté la classe flex et flex-wrap à la div qui contient la liste des mangas (manga-list). Cela applique display: flex pour afficher les éléments enfants côte à côte, et flex-wrap pour perm

Mettre le passage à la ligne des mangas lorsque l'espace est insuffisant.

Pour créer un diaporama avec toutes les images de mangas, je dois utiliser une librairie telle que React Slick qui facilite la création de carrousels et de diaporamas. Voici les étapes à suivre :

```
npm install react-slick slick-carousel
```

27/06/2023:

le composant MangaCarousel, qui est responsable de l'affichage d'un carrousel de mangas populaires. Voici ce que fait chaque partie du code :

- La ligne `import React from "react";` importe la bibliothèque React, qui est nécessaire pour créer des composants React.
- La ligne `import MangaCard from "./MangaCard";` importe le composant MangaCard, qui sera utilisé pour afficher chaque manga dans le carrousel.
- Le composant MangaCarousel est une fonction fléchée avec des propriétés destructurées : `{ mangaList, onMangaClick }`. Cela signifie qu'il s'attend à recevoir deux propriétés en tant que paramètres : `mangaList` (une liste de mangas) et `onMangaClick` (une fonction de gestion des clics sur les mangas).
- À l'intérieur du composant, il retourne une structure JSX qui représente le carrousel. Le carrousel est enveloppé dans une div avec la classe "carousel". Il contient également un titre "Mangas populaires" affiché avec les classes "text-2xl" et "font-bold".
- La partie la plus importante est la div avec la classe "carousel-items". À l'intérieur de cette div, on utilise la méthode `map()` pour parcourir la liste des mangas (`mangaList`) et créer un composant MangaCard pour chaque manga. Chaque MangaCard reçoit une clé unique (`key={manga.id}`), les données du manga lui-même (`manga={manga}`) et la fonction `onMangaClick` pour gérer les clics.

- Finalement, le composant MangaCarousel est exporté pour pouvoir être utilisé ailleurs dans l'application.

En résumé, ce code définit le composant MangaCarousel qui affiche un carrousel de mangas populaires en utilisant le composant MangaCard pour chaque manga dans la liste fournie.

le composant MangaList, qui affiche une liste de mangas. Voici ce que fait chaque partie du code :

- La ligne `import React from "react";` importe la bibliothèque React, qui est nécessaire pour créer des composants React.
- La ligne `import MangaCard from "./MangaCard";` importe le composant MangaCard, qui sera utilisé pour afficher chaque manga dans la liste.
- Le composant MangaList est une fonction fléchée avec des propriétés destructurées : `{ mangaList, onMangaClick }`. Cela signifie qu'il s'attend à recevoir deux propriétés en tant que paramètres : `mangaList` (une liste de mangas) et `onMangaClick` (une fonction de gestion des clics sur les mangas).
- À l'intérieur du composant, il retourne une structure JSX qui représente la liste des mangas. La liste est enveloppée dans une div avec la classe "manga-list". Elle contient également un titre "Liste des Mangas" affiché avec les classes "text-2xl" et "font-bold".
- La partie principale est la div avec les classes "grid" et "grid-cols-..." qui définit une grille responsive avec un nombre de colonnes différentes en fonction de la taille de l'écran. À l'intérieur de cette div, on utilise la méthode `map()` pour parcourir la liste des mangas (`mangaList`) et créer un composant MangaCard pour chaque manga. Chaque MangaCard reçoit une clé unique (`key={manga.id}`), les données du manga lui-même (`manga={manga}`) et la fonction `onMangaClick` pour gérer les clics.
- Finalement, le composant MangaList est exporté pour pouvoir être utilisé ailleurs dans l'application.

En résumé, ce code définit le composant MangaList qui affiche une liste de mangas en utilisant le composant MangaCard pour chaque manga dans la liste fournie.

03/07/2023

Voici une explication du code du composant `App` :

Importations :

```
import React, { useState } from "react";

import { BrowserRouter as Router, Switch, Route, Link } from
"react-router-dom";

import logo from './assets/logo_lmm.jpeg';

import Home from "./components/Home";

import MangaDetail from "./components/MangaDetail";

import MangaPage from "./components/MangaPage";

import MangaData from "./data/MangaData";

import "./App.css";
```

- `React` : Importe le module React pour utiliser des composants React.
- `useState` : Importe la fonction `useState` de React pour gérer l'état local.
- `BrowserRouter as Router` : Importe l'objet `BrowserRouter` et le renomme en `Router` pour gérer les routes de l'application.
- `Switch`, `Route` : Importe les composants `Switch` et `Route` de `react-router-dom` pour définir les itinéraires et le rendu conditionnel des composants.
- `Link` : Importe le composant `Link` de `react-router-dom` pour créer des liens entre les différentes pages de l'application.
- `logo` : Importe le chemin d'accès à l'image du logo de votre application.
- `Home`, `MangaDetail`, `MangaPage` : Importe les composants React correspondant à la page d'accueil, au détail du manga et à la page du manga.
- `MangaData` : Importe les données des mangas à partir d'un fichier séparé.
- `"./App.css"` : Importe le fichier CSS pour le style du composant `App`.
  Composant `App` :

```
const App = () => {
```

```
  const [selectedManga, setSelectedManga] = useState(null);


  const handleMangaClick = (mangaId) => {
    const selected = MangaData.find((manga) => manga.id === mangaId);
    setSelectedManga(selected);
    console.log("Selected Manga:", selected);
  };


  console.log("Selected Manga State:", selectedManga);


  return (
    <Router>
      <div className="manga">
        <Link to="/">
          <img src={logo} alt="logo-la-maison-manga" className="lmm-logo" />
        </Link>
        <h1 className="bg-blue-500 text-white p-4">La Maison Manga</h1>
        <Switch>
          <Route exact path="/">
            <Home mangaList={MangaData} onMangaClick={handleMangaClick} />
          </Route>
          <Route path="/manga/:id">
            <MangaPage manga={selectedManga} />
          </Route>
```

```
          <Route path="/detail/:id">
            <MangaDetail manga={selectedManga} />
          </Route>
        </Switch>
      </div>
    </Router>
  );
};
```

- `useState(null)` : Déclare une variable d'état `selectedManga` avec une valeur initiale de `null`. Cette variable d'état sera utilisée pour stocker le manga sélectionné.
- `handleMangaClick` : Une fonction qui est appelée lorsque vous cliquez sur un manga. Elle recherche le manga correspondant dans les données des mangas et met à jour la variable d'état `selectedManga` avec le manga sélectionné.
- `Router` : Définit le conteneur pour gérer les routes de l'application.
- `Link to="/"`: Entoure l'image du logo avec le composant `Link` et définit l'attribut `to` sur `"/"`, ce qui redirigera le clic sur le logo vers la page d'accueil (`App`).
- `img` : Affiche l'image du logo en utilisant le chemin d'accès importé.
- `h1` : Affiche le titre de votre application.
- `Switch` : Encapsule les composants `Route` pour gérer le rendu conditionnel des composants en fonction de l'URL actuelle.
- `Route exact path="/"`: Définit la route pour la page d'accueil. Lorsque l'URL correspond exactement à `"/"`, le composant `Home` sera rendu et les données des mangas et la fonction de gestion du clic sur le manga (`handleMangaClick`) seront passées en tant que props.
- `Route path="/manga/:id"` : Définit la route pour la page de détail du manga. Lorsque l'URL correspond à `"/manga/:id"`, le composant `MangaPage` sera rendu et le manga sélectionné sera passé en tant que prop.
- `Route path="/detail/:id"` : Définit la route pour le détail du manga. Lorsque l'URL correspond à `"/detail/:id"`, le composant `MangaDetail` sera rendu et le manga sélectionné sera passé en tant que prop.

C'est ainsi que le composant `App` est configuré pour gérer la navigation entre la page d'accueil, la page de détail du manga et la page du manga. Lorsque vous cliquez sur le logo, il vous redirigera vers la page d'accueil (`App`).