

Développement front

MMI 2 – TP#6 S4

Danielo **JEAN-LOUIS**
Michele **LINARDI**

React jusqu'à présent (ce que nous avons vu)

- Découverte du système de composants
- Découverte des props et state
- Utilisation de hooks
- Utilisation peu conventionnel de l'outil
 - Via la balise `<script>`

React jusqu'à présent (ce que nous avons vu)

```
<script src="https://unpkg.com/react@18/umd/react.development.js" defer></script>
<script src="https://unpkg.com/react-dom@18/umd/react-dom.development.js" defer></script>

<!-- NE JAMAIS UTILISER CECI EN PRODUCTION. JAMAIS ! -->
<script src="https://unpkg.com/@babel/standalone/babel.min.js" defer></script>
```

- Importation orthodoxe de React
 - **Personne ne fait ceci dans monde du travail**

C'est là qu'entre en jeu CRA

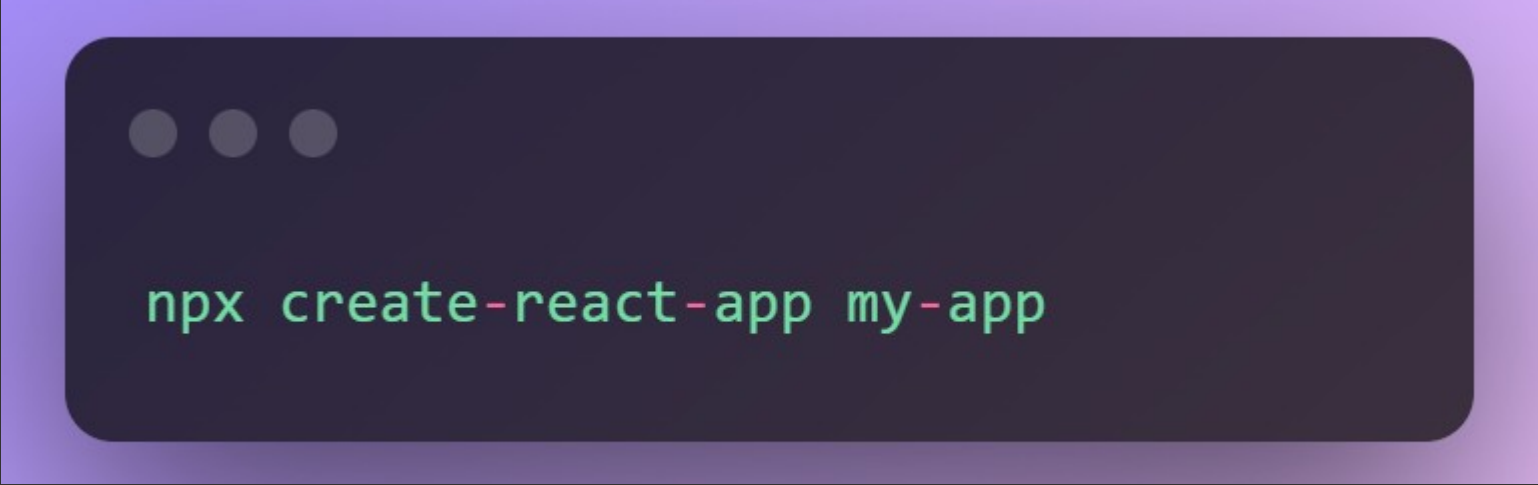
CRA - create-react-app

- Outil (basé sur Nodejs) permettant d'avoir un environnement de travail clé en main pour React
- Gratuit et open source
 - Développé par Facebook

Source(s) :

- <https://create-react-app.dev/docs/getting-started>

CRA - create-react-app

A stylized illustration of a terminal window with a purple border and three small circles in the top-left corner representing window controls. Inside the terminal, the command 'npx create-react-app my-app' is written in a green monospace font.

```
npx create-react-app my-app
```

La commande ci-dessus va créer un dossier nommé "my-app" avec un environnement de travail pour react, vous n'avez plus qu'à développer votre prochaine application React.

Source(s) :

- <https://create-react-app.dev/docs/getting-started>

npx – Node Package eXecute

- Commande liée à npm
- Permet d'exécuter à distance un `node_modules`
 - Pas besoin de mettre à jour le module en local. Pratique !

Source(s) :

- <https://nodejs.org/en/>

CRA - create-react-app

- Fournit un environnement de travail du développement à la production
 - **npm run start** : lancer le serveur de travail
 - **npm run build** : produire un projet pour la production
 - Vous n'avez plus à mettre le tout en ligne

Source(s) :

- <https://create-react-app.dev/docs/getting-started>

CRA - create-react-app

- Fournit également :
 - Un serveur pour faire tourner l'application
 - Un linter pour vérifier les erreurs de code
 - Et plein d'autres choses

Source(s) :

- <https://create-react-app.dev/docs/getting-started>

CRA - create-react-app

- Outil fermé mais pas trop
 - Possibilité d'ajouter de nouveaux `node_modules` au besoin
- Très utile quand on débute
 - CRA peut ne pas être adapté pour vos projets. Nécessité de faire sa propre architecture via des outils comme webpack

Source(s) :

- <https://create-react-app.dev/docs/getting-started>

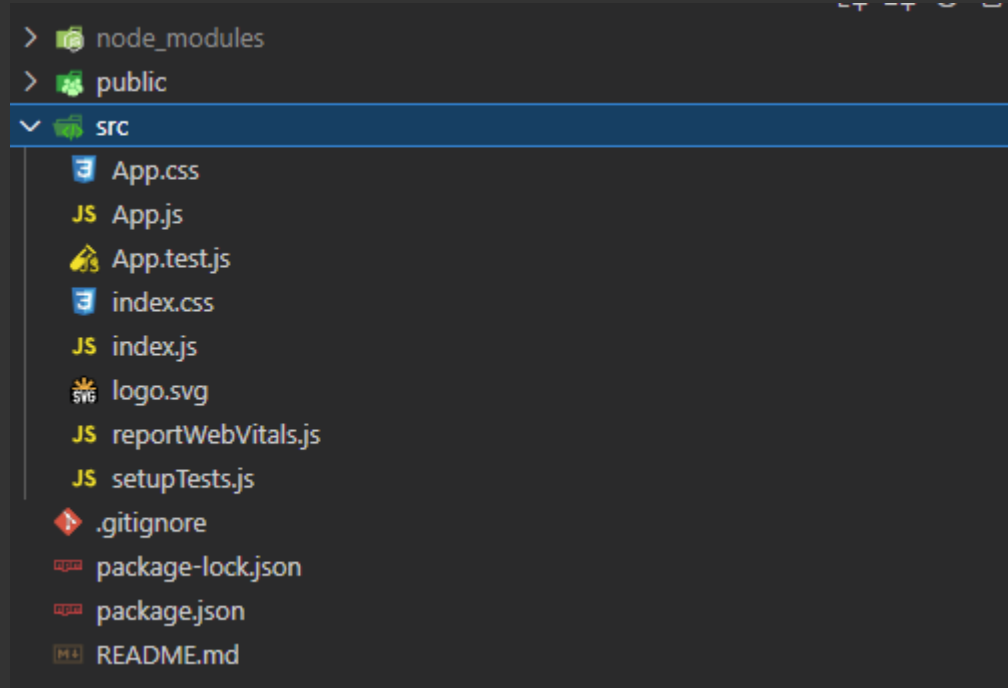
CRA - create-react-app

- Plus besoin de recharger la page
 - Chaque modification de code met à jour l'application

Source(s) :

- <https://create-react-app.dev/docs/getting-started>

CRA - create-react-app




Voilà la base d'un projet CRA
(obtenu après l'exécution de la commande « `npx create-react-app nom-dossier` »)

Source(s) :

- <https://create-react-app.dev/docs/getting-started>

CRA - create-react-app


```
>  node_modules
```

- Les dépendances de notre projet

Source(s) :

- <https://create-react-app.dev/docs/getting-started>

CRA - create-react-app

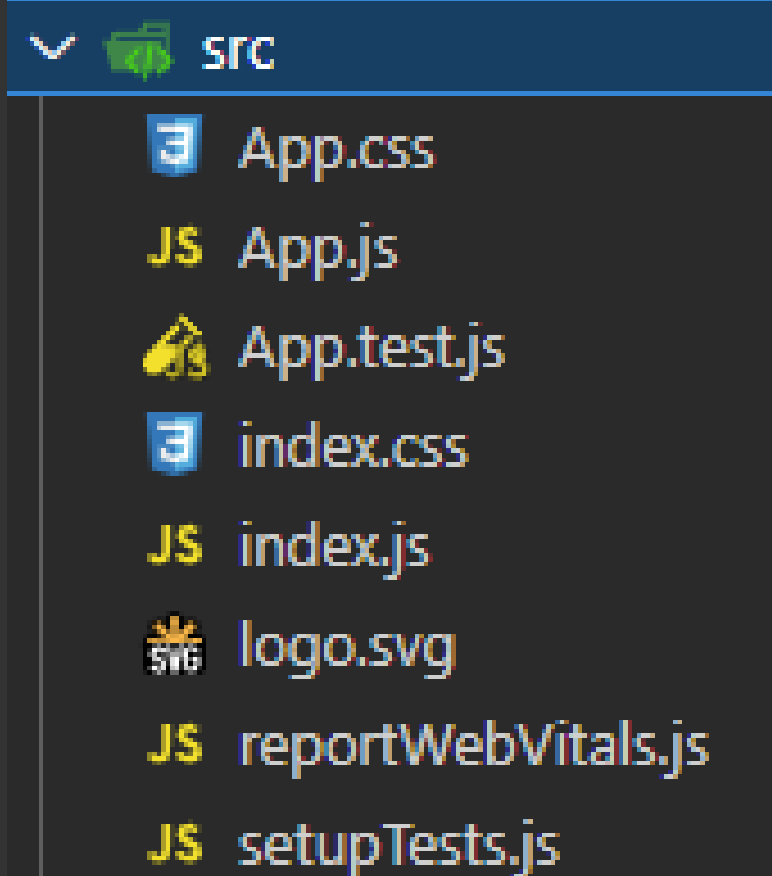
>  public

- Contient tous les fichiers statiques
 - Fichiers qui ne seront pas modifiés par javascript
 - Exemple : html de base, favicon ou texte statique

Source(s) :

- <https://create-react-app.dev/docs/getting-started>

CRA - create-react-app



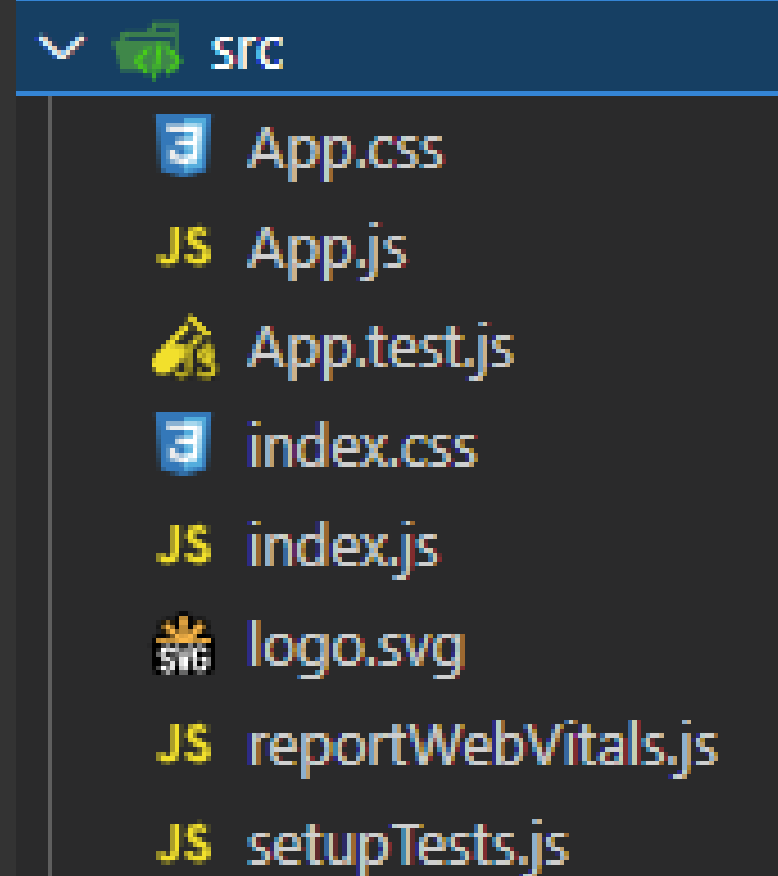
- C'est dans ce dossier que vous allez travailler principalement
- index.js est le point d'entrée de l'application
- logo.svg est une image qu'on peut supprimer

Source(s) :

- <https://create-react-app.dev/docs/getting-started>

CRA - create-react-app

- Les fichiers en ".test.js" sont des fichiers pour tests unitaires
 - Nous ne ferons pas de tests unitaires dans ce cours



Source(s) :

- <https://create-react-app.dev/docs/getting-started>

CRA – Gestion du CSS

- CRA permet l'import de fichiers CSS dans le javascript
 - Comme l'import de fichiers javascript
- Avantages : à la compilation CRA va créer un fichier CSS avec uniquement les classes utilisées et créer des classes uniques

Source(s) :

- <https://create-react-app.dev/docs/adding-a-stylesheet>

CRA – Gestion du CSS

```
/* Button.css */
```

```
.Button {  
  padding: 20px;  
}
```

```
/* Button.js */
```

```
import React, { Component } from 'react';  
import './Button.css'; // Import du CSS
```

```
const Button = (props) => {  
  // Utilisation de la classe  
  return <div className="Button" />;  
}
```

A gauche, un fichier CSS classique. A droite son utilisation directement dans notre fichier javascript.

Source(s) :

- <https://create-react-app.dev/docs/adding-a-stylesheet>

fetch

- API native de javascript
 - Ne nécessite pas de dépendance externe
- Communique avec un serveur de façon asynchrone
- **Ne fonctionne pas sans serveur pour un fichier local**
 - Nécessite un serveur php, nodejs...

Source(s) :

- https://developer.mozilla.org/fr/docs/Web/API/Fetch_API/Using_Fetch

fetch avec ReactJS

Le hook « `React.useEffect(() => {}, [])` » permet de produire des effets. Autrement un ensemble d'instructions qui se produisent d'un un vase clos ici l'appel vers une API et le remplissage de notre state (`setData`).

Le tableau vide à la fin indique de notre `React.useEffect` ne sera appelé qu'une seule fois.

Source(s) :

- https://developer.mozilla.org/fr/docs/Web/API/Fetch_API/Using_Fetch
- <https://fr.reactjs.org/docs/faq-ajax.html>
- <https://fr.reactjs.org/docs/hooks-reference.html#useeffect>

```
const MyComponent = () => {
  const [data, setData] = React.useState([]);

  React.useEffect(() => {
    fetch("LIEN-API")
      .then(res => res.json())
      .then((result) => {
        setData(result);
      })
  }, [])

  if (data.length === 0) {
    return <div>Chargement...</div>;
  }

  return (
    <ul>
      {items.map(item => (
        <li key={item.name}>
          {item.name} {item.price}
        </li>
      ))}
    </ul>
  );
}
```

fetch

- Un peu verbeux
 - Existe des alternatives comme axios

Source(s) :

- https://developer.mozilla.org/fr/docs/Web/API/Fetch_API/Using_Fetch
- <https://axios-http.com/>

React – Dynamique grâce à AJAX

- AJAX : Asynchronous Javascript And XML
 - Permet de faire des requêtes asynchrones sur le web
 - Communiquer avec le serveur sans recharger la page
- Utilisation de l'API "fetch"

Pratiquons ! - react dans un environnement nodejs (Partie 1/2)

Pré-requis :

- Avoir la ressource ressources/cra

A télécharger ici :

<https://download-directory.github.io?url=https://github.com/DanYellow/cours/tree/main/developpement-front-s4/travaux-pratiques/numero-6/ressources>

CRA – Gestion du CSS

- Il existe d'autres façon de gérer le CSS dans un environnement nodejs
 - CSS Modules (déjà géré par cra)
 - Styled-components
 - Sass (gérable par cra avec quelques ajouts)
 - ...

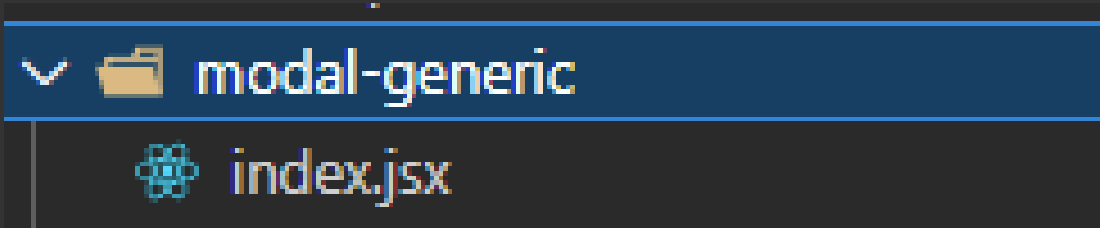
Source(s) :

- <https://create-react-app.dev/docs/adding-a-stylesheet>

Nommage des fichiers – Bonnes pratiques

- Par convention, on mettra l'extension ".jsx" aux fichiers jsx
 - Ex : message.jsx
- Ou faire un dossier par composant
 - Ex : messages/index.jsx

Nommage des fichiers – Bonnes pratiques




On définit un dossier avec notre composant



On l'importe dans un autre fichier

Nommage des fichiers – Bonnes pratiques

A code editor window with a purple border and a dark purple background. It features three light gray window control buttons (minimize, maximize, close) in the top-left corner. The text inside is a single line of code: `import modal from './modal-generic'`, where 'import' is pink, 'modal' is green, 'from' is pink, and the string is yellow.

```
import modal from './modal-generic'
```

Nous n'avons pas besoin de préciser le fichier car par défaut, node cherche un fichier qui s'appelle « index.js » ou « index.jsx »

Questions ?

