

Développement front

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

MMI 2 – TP#6 S4

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

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

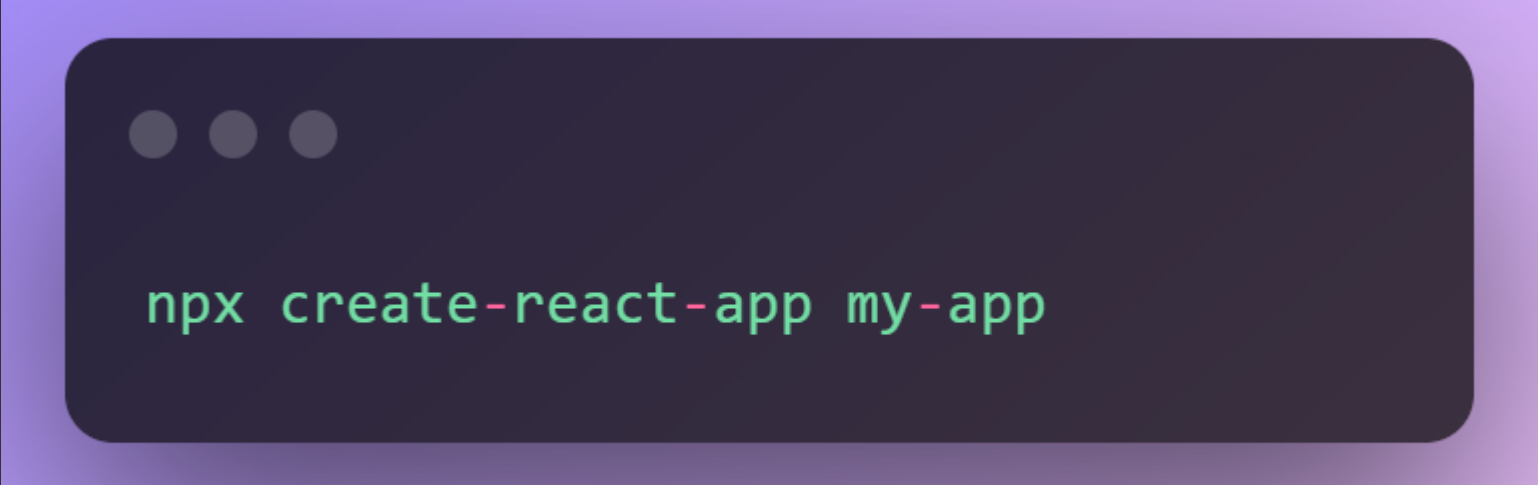
CRA - create-react-app

- Outil (basé sur Node) permettant d'avoir un environnement de travail pour React
- Permet aux développeurs de travailler sur une application react et non développer des outils pour développer une app react

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 plu qu'à coder.

Source(s) :

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

npx – Node Package eXecute

- Commande liée à node
- 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 fichier de 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'app
 - Un linter pour vérifier les erreurs de code
 - Un env pour tests unitaires
 - Et plein d'autres

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.

Source(s) :

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

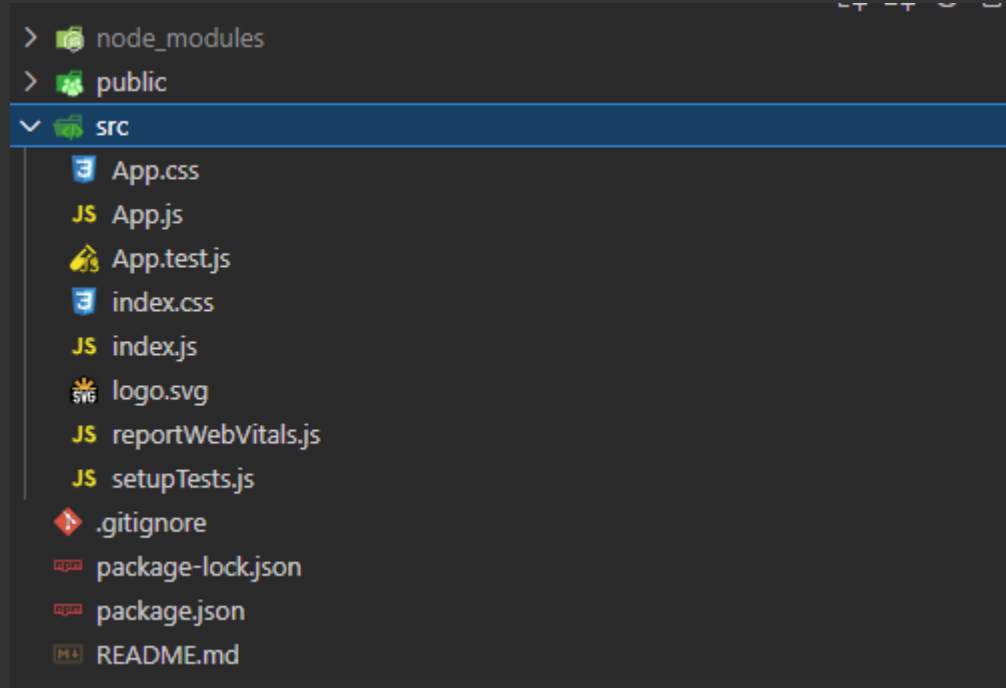
CRA - create-react-app

- 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

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

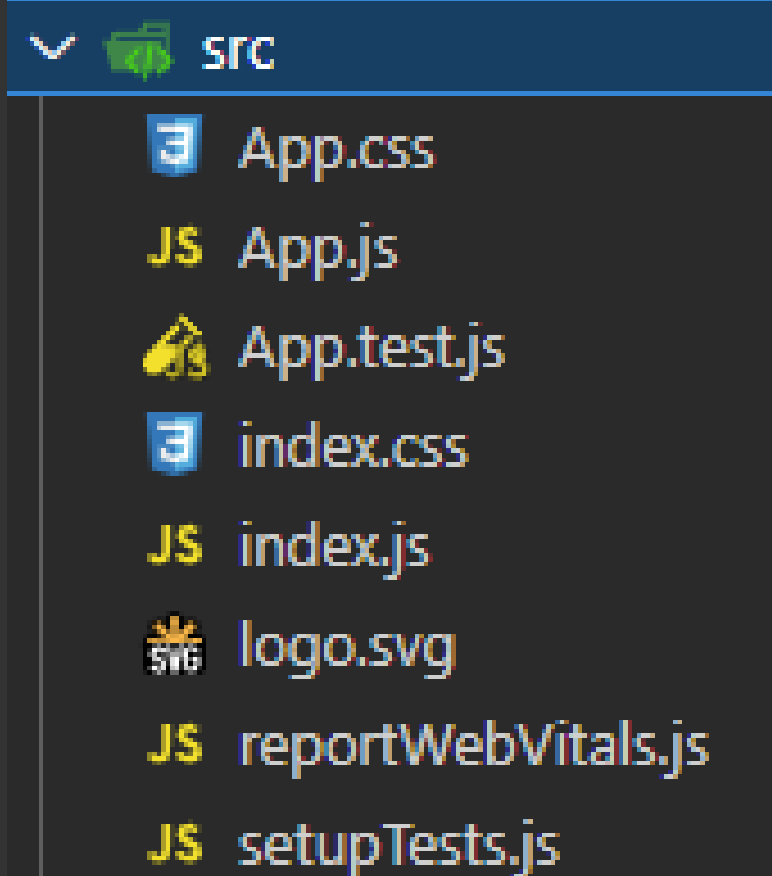
>  public

- 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



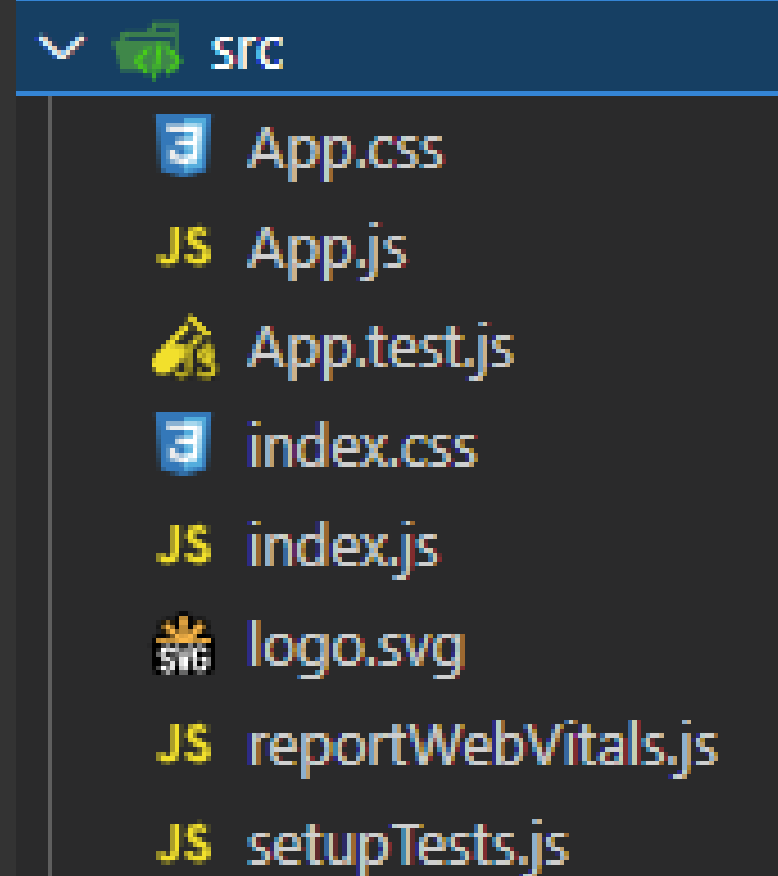
- C'est dans ce dossier que vous allez travailler principalement
- index.js est le point d'entrée de l'application
- logo.svg est une image qu'on peut supprimer

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



Source(s) :

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

CRA – Gestion du CSS

- CRA permet l'import de fichiers CSS
 - Comme l'import de fichiers javascript
- Avantages : à la compilation CRA à 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>

Pratiquons ! - react dans un env sain (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-2/ressources>

CRA – Gestion du CSS

- Il existe d'autres façon de gérer le CSS dans un env node
 - CSS Modules (déjà géré par cra)
 - Styled-components
 - Sass (gérable par cra)
 - ...

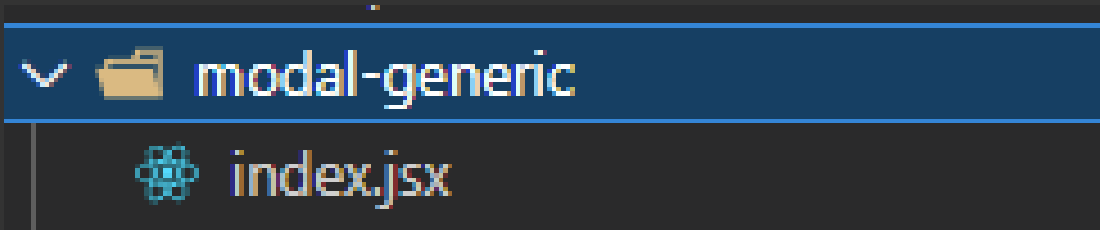
Source(s) :

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

Nommage des fichiers – Bonnes pratiques

- Par convention on met 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

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

On l'importe dans un autre fichier

Nommage des fichiers – Bonnes pratiques



```
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"

Questions ?

