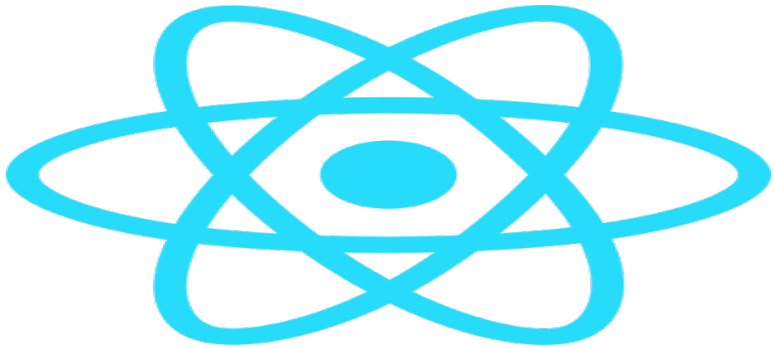
React : introduction

Achref El Mouelhi

Docteur de l'université d'Aix-Marseille
Chercheur en programmation par contrainte (IA)
Ingénieur en génie logiciel

elmouelhi.achref@gmail.com



Plan

React

Framework

- Plusieurs traductions
  - cadriciel
  - environnement de développement
  - cadre d'applications
  - ...
- Ensemble de composants logiciels
- Facilitant la réalisation d'une (partie de l') application
- Imposant une certaine structure, logique, syntaxe...

React

Plusieurs types de Framework

- Frameworks applicatifs pour le développement d'applications web :
  - **Angular**, **React**, **Vue.js** pour **JavaScript**,
  - **Spring** pour **Java**,
  - **Symfony**, **Laravel** pour **PHP**
  - ...
- Frameworks de présentation de contenu web :
  - **Bootstrap**, **Tailwind** pour **CSS**
  - ...
- Frameworks de persistance
- Frameworks de logging
- ...

React

React, **React.js** ou **ReactJS**

- Bibliothèque (Framework) **JavaScript**
- Open source
- Créé par la communauté **Facebook** en 2011
- Permettant de créer des applications web et mobiles (avec **React Native**)
  - Front-End
  - Single page
- Utilisant
  - les composants web
  - le DOM Virtuel

React

React : l'un des frameworks frontends les plus populaires

- https://2023.stateofjs.com/fr-FR/libraries/front-end-frameworks/
- https://npmtrends.com/angular-vs-react-vs-vue

React

DOM : Document Object Model

- Structure en arbre représentant la page web
- Généré par le navigateur depuis le **HTML** pour afficher la page
- Manipulable via l'onglet **Elements** dans le **DevTools** du navigateur.

React

DOM : **D**ocument **O**bject **M**odel

- Structure en arbre représentant la page web
- Généré par le navigateur depuis le **HTML** pour afficher la page
- Manipulable via l'onglet **Elements** dans le **DevTools** du navigateur.

Termes couramment utilisés pour parler du **DOM**

- **DOM** réel : le **DOM** généré et manipulé par le navigateur
- **DOM** physique : terme moins courant, souvent utilisé comme synonyme de **DOM** réel

React

DOM Virtuel ?

- introduit, initialement, par **React**.
- une représentation en mémoire du **DOM** réel.
- permettant des mises à jour plus rapides et efficaces en ne modifiant que les parties nécessaires de l'interface utilisateur avant de synchroniser ces changements avec le **DOM** réel.

React

DOM Virtuel ?

- introduit, initialement, par **React**.
- une représentation en mémoire du **DOM** réel.
- permettant des mises à jour plus rapides et efficaces en ne modifiant que les parties nécessaires de l'interface utilisateur avant de synchroniser ces changements avec le **DOM** réel.

Remarque

- Les modifications apportées au **DOM** physique entraînent généralement des opérations coûteuses en termes de temps de traitement.
- Chaque modification peut déclencher des réorganisations et des recalculs complexes de la mise en page.

React

Quelques outils utilisés par **React**

- **npm** (**n**ode **p**ackage **m**anager) : le gestionnaire de paquets par défaut pour les applications **JavaScript**
- **Webpack** : bundler **JavaScript**
  - utilisé par défaut par le système **CRA** : **C**reate **R**eact **A**pp,
  - construit le graphe de dépendances,
  - permet de regrouper les fichiers **JavaScript**, de gérer les `assets` comme les images et les fichiers `CSS`, et d'appliquer des transformations avec des plugins (par exemple, **Babel** pour la transpilation du code **ES6+**).
- **Rollup** : bundler **JavaScript**
  - utilisé pour les bibliothèques **React**,
  - réduit la taille du code final en éliminant les modules inutilisés.

React

CRA : **C**reate **R**eact **A**pp (**CLI** de **React**) qui offre

- **Configuration zéro** : il propose une configuration prédéfinie pour (**Webpack**, **Babel**, **ESLint**...).
- **Commandes de base** : il fournit des commandes pour
  - le développement (`npm start`),
  - la construction (`npm run build`),
  - et les tests (`npm test`).
- **Support du HMR** (**H**ot **M**odule **R**eplacement) : Pendant le développement, **CRA** permet de mettre à jour les modules sans recharger toute l'application.

React

Quels langages utilise **React** ?

- **HTML**
- **CSS**
- **JSX** (**J**ava**S**cript **X**ML)

React

Les différentes versions de **React**

- React 0.1.0 : Mars 2013
- React 0.2.0 : Mars 2013
- React 0.3.0 : Juin 2013
- React 0.4.0 : Juillet 2013
- React 0.5.0 : Octobre 2013
- React 0.6.0 : Janvier 2014
- React 0.7.0 : Mars 2014
- React 0.8.0 : Mai 2014
- React 0.9.0 : Juin 2014
- React 0.10.0 : Juillet 2014
- React 0.11.0 : Août 2014
- React 0.12.0 : Octobre 2014
- React 0.13.0 : Mars 2015
- React 0.14.0 : Octobre 2015
- React 15.0.0 : Avril 2016
- React 16.0.0 : Septembre 2017
- React 17.0.0 : Octobre 2020
- React 18.0.0 : Mars 2022
- React 18.1.0 : Avril 2022
- React 18.2.0 : Juin 2022
- React 18.3.0 : Avril 2024
- React 19.0.0 : Octobre 2024

React

Quelques sites Web réalisés avec **React** ?

- Netflix
- Airbnb
- Sony
- Yahoo
- ...

React

Documentation officielle (en français)

`https://fr.react.dev/`

React

Quel IDE (Environnement de développement intégré) ?

- **Visual Studio Code** (À ne pas confondre avec Visual Studio)
- WebStorm
- ...

React

Quel IDE (Environnement de développement intégré) ?

- **Visual Studio Code** (À ne pas confondre avec Visual Studio)
- WebStorm
- ...

Visual Studio Code (ou VSC), pourquoi ?

- Gratuit
- Multi-OS et multi-langages
- Extensible
- Léger
- Intégration **Git** et **Emmet**

React

VSC : téléchargement

`code.visualstudio.com/download`

React

VSC : téléchargement

`code.visualstudio.com/download`

Extensions VSC pour **React**

- `ES7 React/Redux/React-Native/JS snippets` et
- `JS JSX Snippets`

React

Quelques raccourcis **VSC**

- Pour activer la sauvegarde automatique : aller dans `File > AutoSave`
- Pour indenter son code : `Alt` `Shift` `f`
- Pour commenter/décommenter : `Ctrl` `:`
- Pour sélectionner toutes les occurrences : `Ctrl` `f2`
- Pour sélectionner l'occurrence suivante : `Ctrl` `d`
- Pour placer le curseur dans plusieurs endroits différents : `Alt`

React

React Devtools

- Extension multi-navigateur pour le débogage des applications **React**.
- Installation : `https://react.dev/learn/react-developer-tools`

React

Solution avec **CDN** : démarche

- Depuis **VSC**, allez dans `File > Open Folder...`
- Cliquez sur `Nouveau dossier` et saisissez `react-cdn`
- Cliquez sur le dossier `react-cdn` puis sur le dossier `Sélectionner un dossier`
- Dans `react-cdn`, créez deux fichiers `index.html` et `script.js`
- Dans `index.html`, saisissez `html:5` ou `!` puis cliquez sur `Entree`

React

Code généré

```
<!DOCTYPE html>
<html lang="fr">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>React + CDN</title>
</head>

<body>

</body>

</html>
```

React

Référençons `script.js` **dans** `index.html`

```
<!DOCTYPE html>
<html lang="fr">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>React + CDN</title>
</head>

<body>

    <script src="./script.js"></script>
</body>

</html>
```

React

Ajoutons les CDN (React)

```
<!DOCTYPE html>
<html lang="fr">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>React + CDN</title>
</head>

<body>

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

</html>
```

React

Spécifions la zone de la page où l'application React sera chargée

```
<!DOCTYPE html>
<html lang="fr">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>React + CDN</title>
</head>

<body>
    <div id='root'>
    </div>
    <script crossorigin src="https://unpkg.com/react@18/umd/react.development.js">
    </script>
    <script crossorigin src="https://unpkg.com/react-dom@18/umd/react-dom.development.js">
    </script>
    <script src="./script.js"></script>
</body>

</html>
```

React

Ajoutons le contenu suivant dans `script.js`

```
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render("Hello world");
```

React

Ajoutons le contenu suivant dans `script.js`

```
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render("Hello world");
```

Explication

- `ReactDOM` : objet prédéfini chargé depuis la **CDN**
- `createRoot()` : méthode permettant de créer une application **React**
- `root.render()` : méthode permettant de charger le paramètre dans la zone `root`

React

Pour tester, utilisons l'extension **Live Server**

- Installez l'extension **Live Server**
- Faites un clic doit sur `index.html`
- Cliquez sur `Open with Live Server`
- Vérifiez l'affichage de `Hello world`

React

Entourons `Hello world` **par la balise** `h1`

```
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render("<h1>Hello world</h1>");
```

React

Entourons `Hello world` **par la balise** `h1`

```
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render("<h1>Hello world</h1>");
```


Constat

Vérifiez que la balise n'a pas été interprétée.

React

Entourons `Hello world` par la balise `h1`

```
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render("<h1>Hello world</h1>");
```


Constat

Vérifiez que la balise n'a pas été interprétée.

Remarque

Pour que la balise soit interprétée, on peut utiliser **JSX**.

React

JSX

- **J**avaScript **S**yntax E**x**tension, appelé aussi **J**ava**S**cript **X**ML
- Permettant de mélanger du code **JavaScript** avec des balises **HTML**
- Permettant également d'incorporer des expressions **JavaScript** en les enveloppant dans des accolades { }.

React

Commençons par ajouter une CDN pour Babel qui permettra d'interpréter le code JSX

```
<!DOCTYPE html>
<html lang="fr">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>React + CDN</title>
</head>

<body>
    <div id='root'>
    </div>
    <script crossorigin src="https://unpkg.com/react@18/umd/react.development.js">
    </script>
    <script crossorigin src="https://unpkg.com/react-dom@18/umd/react-dom.development.js">
    </script>
    <script src="https://unpkg.com/babel-standalone@6/babel.min.js"></script>
    <script src="./script.js"></script>
</body>

</html>
```

React

Pour utiliser JSX, ajoutons l'attribut `type="text/babel"` à la balise `script`

```
<!DOCTYPE html>
<html lang="fr">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>React + CDN</title>
</head>

<body>
    <div id='root'>
    </div>
    <script crossorigin src="https://unpkg.com/react@18/umd/react.development.js">
    </script>
    <script crossorigin src="https://unpkg.com/react-dom@18/umd/react-dom.development.js">
    </script>
    <script src="https://unpkg.com/babel-standalone@6/babel.min.js"></script>
    <script type="text/babel" src="./script.js">
    </script>
</body>

</html>
```

React

Nous pouvons désormais utiliser JSX (JSX est une expression et non pas une `string`, donc pas besoin de quote)

```
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<h1>Hello world</h1>);
```

React

Nous pouvons désormais utiliser JSX (JSX est une expression et non pas une `string`, donc pas besoin de quote)

```
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<h1>Hello world</h1>);
```


Constat

Vérifiez que le contenu s'affiche et que la balise `h1` a bien été interprétée.

React

React avec **CDN** : avantages

- Simplicité et rapidité d'intégration
- Idéal pour les petites applications
- Diminution du poids du projet
- Pas de transpilation requise

React avec **CDN** : inconvénients

- Gestion difficile des dépendances
- Moins adapté aux grandes applications
- Performance de chargement
- Dépendance au réseau

React

Remarque

Pour installer **React**, il faut télécharger et installer **Node.js**

React

Remarque

Pour installer **React**, il faut télécharger et installer **Node.js**

Pour **Node.js**, il faut

- aller sur `https://nodejs.org/en/`
- choisir une version récente, télécharger et installer

React

Remarque

Pour installer **React**, il faut télécharger et installer **Node.js**

Pour **Node.js**, il faut

- aller sur `https://nodejs.org/en/`
- choisir une version récente, télécharger et installer

Pour vérifier l'installation depuis une console (invite de commandes), exécutez

```
node -v
```

React

Pour créer le squelette d'une application React

```
npx create-react-app react-npm
```

React

Pour créer le squelette d'une application React

```
npx create-react-app react-npm
```

Pour lancer le projet

```
npm start
```

React

Arborescence d'un projet **React**

- `node_modules` : contenant les modules **Node.js** nécessaires pour **React**
- `public` : contenant les fichiers accessibles de l'extérieur
- `src` : contenant les fichiers sources de l'application (exigé par **Webpack**)
- `package.json` : contenant l'ensemble de dépendance de l'application

React

Que contient `public` ?

- `index.html` : unique fichier **HTML** d'une application **React**
- `favicon.ico`, `logo192.png` et `logo512.png` : les logo de **React**
- `manifest.json` : fichier **JSON** permettant de décrire l'application (nom, auteur...) pour configurer l'apparence et le comportement de l'application lorsqu'elle est installée sur un appareil mobile ou utilisée comme une application web autonome.
- `Robots.txt` : fichier texte consulté par les moteurs de recherche pour savoir si l'affichage de contenu de l'application est autorisé dans les résultats de recherche.

React

Contenu d'`index.html` (sans les commentaires générés)

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta name="description" content="Web site created using create-react-app" />

    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />

    <title>React App</title>
</head>

<body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
</body>

</html>
```

React

Question 1

Pourquoi `index.js` n'est pas référencé par `index.html` ?

React

Question 1

Pourquoi `index.js` n'est pas référencé par `index.html` ?

Réponse

- **En développement**, **CRA** utilise un serveur de développement (comme **Webpack Dev Server**) qui injecte dynamiquement le script dans la page.
- **En production (build)**, `index.js` est intégré par **Webpack** pendant le processus de `build`. Le bundler génère un fichier **JavaScript** optimisé.

React

Que contient `src` ?

- `index.js` : le point d'entrée de l'application
- `index.css` : la feuille de style associée au point d'entrée
- `App.js` : le premier composant
- `App.css` : la feuille de style associée au premier composant
- `App.test.js` : le fichier de test du premier composant
- `reportWebVitals.js` : permet la configuration de mesure et de suivi des performances des applications **React** à l'aide de l'outil **web-vitals** (package **npm**).
- `setupTests.js` : le fichier de configuration globale de test

Contenu d'`index.js` (point d'entrée)

```
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')
);

reportWebVitals();
```

Contenu d'`index.js` (point d'entrée)

```
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')
);

reportWebVitals();
```

Explication

- `React` : **API** permettant de gérer les composants
- `ReactDOM` : **API** permettant d'attacher les composants au **DOM**

Contenu d'`App.js` (premier composant)

```
import logo from './logo.svg';
import './App.css';

function App() {
  return (
    <div className="App">
      <header className="App-header">
        <img src={logo} className="App-logo" alt="logo" />
        <p>
          Edit <code>src/App.js</code> and save to reload.
        </p>
        <a
          className="App-link"
          href="https://reactjs.org"
          target="_blank"
          rel="noopener noreferrer"
        >
          Learn React
        </a>
      </header>
    </div>
  );
}

export default App;
```

Contenu d'`App.js` (premier composant)

```
import logo from './logo.svg';
import './App.css';

function App() {
  return (
    <div className="App">
      <header className="App-header">
        <img src={logo} className="App-logo" alt="logo" />
        <p>
          Edit <code>src/App.js</code> and save to reload.
        </p>
        <a
          className="App-link"
          href="https://reactjs.org"
          target="_blank"
          rel="noopener noreferrer"
        >
          Learn React
        </a>
      </header>
    </div>
  );
}

export default App;
```

Explication

`App-header` et `App-logo` : classes **CSS** définies dans `App.css`