

React Introduction

Cedric Dumoulin

Vue rapide de React

Qu'est-ce que React ?

- bibliothèque JavaScript déclarative,
 - efficace et flexible
 - pour construire des interfaces utilisateurs (UI)
- permet de construire des UI complexes
 - Par composition
 - ➔ notion de composants

Historique

- Sources : <https://fr.wikipedia.org/wiki/React>
- Aussi appelé **React.js** ou **ReactJS**
- maintenue par
 - Meta (anciennement Facebook)
 - + communauté de développeurs
- créé par Jordan Walke,
 - Ingénieur chez Meta
- 1ere publication :
 - Mai 2013

Principe de base

- React se charge uniquement du rendu (de l'affichage)
 - Coté client
 - Ou coté serveur
- Notion de composants
 - Chaque composant permet de faire du rendu
 - On assemble les composants
 - Un composants peut contenir d'autres composants
- Bibliothèque Javascript
 - Un composant = code Javascript

Principe de base

- SPA (Single Page Application)
 - Application React = une seule page !
 - Chargé une seule fois (charge la bibliothèque JS)
 - Constituée de composants
- Changement dynamique des composants
 - Permet d'afficher des contenus différents
- Un composant peut demander des données au serveur
 - Ex: Composant affichage de la liste des étudiants
 - Demande la liste de données au serveur (réponse en Json)
 - Se charge du rendu

Principe de base

- Changement dynamique des composants
 - Peut se faire avec un composant 'Router'
 - Affiche un composant différent en fonction d'une URL (locale)

Deux façon d'écrire des composants

- Classe étendant `React.Component`
 - A éviter
- ou **Fonction Composant** + Hooks
 - Préconiser cette approche

Classe Composant

```
class ShoppingList extends React.Component {
  render() {
    return (
      <div className="shopping-list">
        <h1>Liste de courses pour {this.props.name}</h1>
        <ul>
          <li>Instagram</li>
          <li>WhatsApp</li>
          <li>Oculus</li>
        </ul>
      </div>
    );
  }
}
```

// Exemple d'utilisation : <ShoppingList name="Marc" />

Fonction Composant

```
function Example() {
  // Déclaration d'une nouvelle variable d'état, que
  const [count, setCount] = useState(0);

  return (
    <div>
      <p>Vous avez cliqué {count} fois</p>
      <button onClick={() => setCount(count + 1)}>
        Cliquez ici
      </button>
    </div>
  );
}
```


Composants et JSX

- Un composant React retourne du JSX
 - JSX : mélange de pseudo-html et d'appel de Composant
- Pseudo-html
 - Commence par une minuscule
 - Reprend le nom des balises html
- Insertion de composant
 - Le nom commence par une majuscule
 - Utilise le nom du composant

```
function MyButton() {  
  return (  
    <button>I'm a button</button>  
  );  
}
```

```
export default function MyApp() {  
  return (  
    <div>  
      <h1>Welcome to my app</h1>  
      <MyButton />  
    </div>  
  );  
}
```

Langage JSX

- La syntaxe **mélange** du Javascript et du Html
 - C'est du **JSX**
 - On retrouve la syntaxe Javascript
 - On peut mettre du code qui ressemble à des balises HTML
 - (en fait du JSX)
 - On peut mettre nos propres balises
 - balises définies par les composants
 - Nom de la balise commence par une Majuscule

```
class ShoppingList extends React.Component {  
  render() {  
    return (  
      <div className="shopping-list">  
        <h1>Liste de courses pour {this.props.name}</h1>  
        <ul>  
          <li>Instagram</li>  
          <li>WhatsApp</li>  
          <li>Oculus</li>  
        </ul>  
      </div>  
    );  
  }  
}  
  
// Exemple d'utilisation : <ShoppingList name="Marc" />
```

Composant et paramètres

- On peut passer des arguments à un composant :

- Dans l'exemple : 2 arguments :
 - 'person' (prend un objet)
 - et size (prend un int)

```
export default function Profile() {  
  return (  
    <Avatar  
      person={{ name: 'Lin Lanying', imageId: '1bX5QH6' }}  
      size={100}  
    />  
  );  
}
```

- Le composant ne déclare qu'un seul paramètre : props

- props est un objet qui contient les arguments
 - props.person
 - props.size
- On utilise la destructuration
 - Notez les {}: {person, size}

```
function Avatar( props ) {  
  person = props.person;  
  size = props.size;  
}
```

- Plus d'info :

- <https://react.dev/learn/passing-props-to-a-component>

```
function Avatar({ person, size }) {  
  // person and size are available here  
}
```

Composant et état

- Chaque composant est associé à un ou plusieurs 'états' (state)
 - état = ensemble de valeurs utilisées (ex: affichées) par le composant
- Quand l'état change
 - React rafraichit le composant
 - ➔ une partie de la page est redessinée

```
class Square extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      value: null,
    };
  }

  render() {
    return (
      <button className="square" onClick={() => alert('clik')}>
        {this.props.value}
      </button>
    );
  }
}
```

```
function Example() {
  // Déclaration d'une nouvelle variable d'état, que
  const [count, setCount] = useState(0);

  return (
    <div>
      <p>Vous avez cliqué {count} fois</p>
      <button onClick={() => setCount(count + 1)}>
        Cliquez ici
      </button>
    </div>
  );
}
```

Quel langage ?

Javascript ou Typescript ?

- React supporte les deux
- Vous utilisez le langage que vous préférerez
- La plupart des exemples et des tutoriaux du net sont en Javascript
- Plus facile d'apprendre avec Javascript
 - ➔ Exemple du cours en JS
- Large applications,
 - ➔ Typescript

SPA : Single Page Application

- React permet d'écrire une application dans « **une seule page** » !!
 - Le contenu de la page change en fonction des interactions de l'utilisateur
- La page est construite par assemblage de composants

React et Arbre HTML

- Le code React permet de réécrire l'arbre html (DOM) interprété par le navigateur
- Réécriture du DOM ➔ changement de l'affichage dans le navigateur
- Le navigateur ne fait que afficher l'arbre html
- React construit et modifie l'arbre HTML

Mettre en place l'environnement

npx et npm

- Vous avez besoin des outils **npx** et **npm**
- Peuvent être installé séparément
- Ou installé avec NodeJS
 - ➔ Installer NodeJS
 - <https://nodejs.org/fr>

Installer NodeJS sur sa machine

- Installer Node.js
 - <https://nodejs.org/fr>

Installer NodeJS sur les machines du M5

- Voir <https://intranet.fil.univ-lille.fr/2020/04/09/nodejs-et-npm/>



Créer un nouveau projet

Environnement de développement

Visual Studio Code

- Installez visual Studio Code
 - https://code.visualstudio.com/?wt.mc_id=vscom_downloads
- Ajouter :
 - **Git Graph**
- Créer un répertoire de travail
- Ouvrir le terminal dans VS
- Créer un projet React
 - `npx create-react-app my-app`
- Tester !
 - `cd my-app`
 - `npm start`

Créer un projet

- Utiliser un outil de création :
 - CRA (Create React App)
 - Vite
- CRA (Create React App)

CRA (Create React App)

- <https://create-react-app.dev/>
- Créer un répertoire de travail
- Ouvrir le terminal dans VS
- Créer un projet React
 - `npx create-react-app my-app`
- Tester !
 - `cd my-app`
 - `npm start`
- N'est plus maintenu (a priori)

npm vs npx

NPM is a package manager used to install, delete, and update Javascript packages on your machine. NPX is a package executor, and it is used to execute javascript packages directly, without installing them.

<https://www.naukri.com/code360/library/difference-between-npm-and-npx>

Vite

- <https://vitejs.dev/guide/>
- build tool that aims to provide a faster and leaner development experience for modern web projects
- Two parts
 - A dev server
 - A build command that bundles your code with [Rollup](#)
- Créer un projet React
 - `npm create vite@latest`
 - Demande à installer les libs si nécessaire
 - Demande le nom du projet : `vite-project`
 - `cd vite-project`
 - `npm install`
 - `npm run dev`

Première application

Demo !

Premier projet React (CRA)

- Ouvrir Visual Studio Code
- Créer un répertoire de travail
- Ouvrir le terminal dans VS
- Créer un projet React
 - `npx create-react-app my-app`
- Tester !
 - `cd my-app`
 - `npm start`

Explications projet CRA

- public/index.html
- src/index.js
 - Cherche le node 'id'
 - Le remplace par le composant 'App.js'
- src/App.js
 - Return
 - ()
 - JSX

Demo !

Premier projet React (Vite)

- Ouvrir Visual Studio Code
- Créer un répertoire de travail
- Ouvrir le terminal dans VS
- Créer un projet React
 - `npm create vite@latest`
 - Demande à installer les libs si nécessaire
 - Demande le nom du projet : `vite-project`
 - `cd vite-project`
 - `npm install`
- Tester !
 - `cd vite-project`
 - `npm run dev`

Explications projet VITE

- `./index.html`
 - Lance le script `/src/main.jsx`
- `/src/main.jsx`
 - Cherche le node 'root'
 - Le remplace par le composant 'App.js'
- `src/App.js`
 - Return
 - `()`
 - JSX
 - Un état ...

Tutorial Tic Tac Toe

Atelier

- Tutorial Tic Tac Toe
 - <https://react.dev/learn/tutorial-tic-tac-toe>
 - **Faites le tutorial dans VSCode** (et non dans le navigateur)
 - Créer un projet avec CRA ou Vite

Penser en React

Penser en React

- <https://react.dev/learn/thinking-in-react>
- When you build a user interface with React,
 - you will first **break it apart into pieces called *components***.
 - Then, you will **describe the different visual states** for each of your components.
 - Finally, you will **connect your components together** so that the data flows through them.

Atelier

- Thinking in React
 - <https://react.dev/learn/thinking-in-react>
- Faite le tutoriel
 - Vous apprendrez
 - À découper votre application en composants
 - A réaliser des 'prototypes' de vos composants
- Vous pouvez déjà réfléchir aux composants de votre application

Application gestion de stage

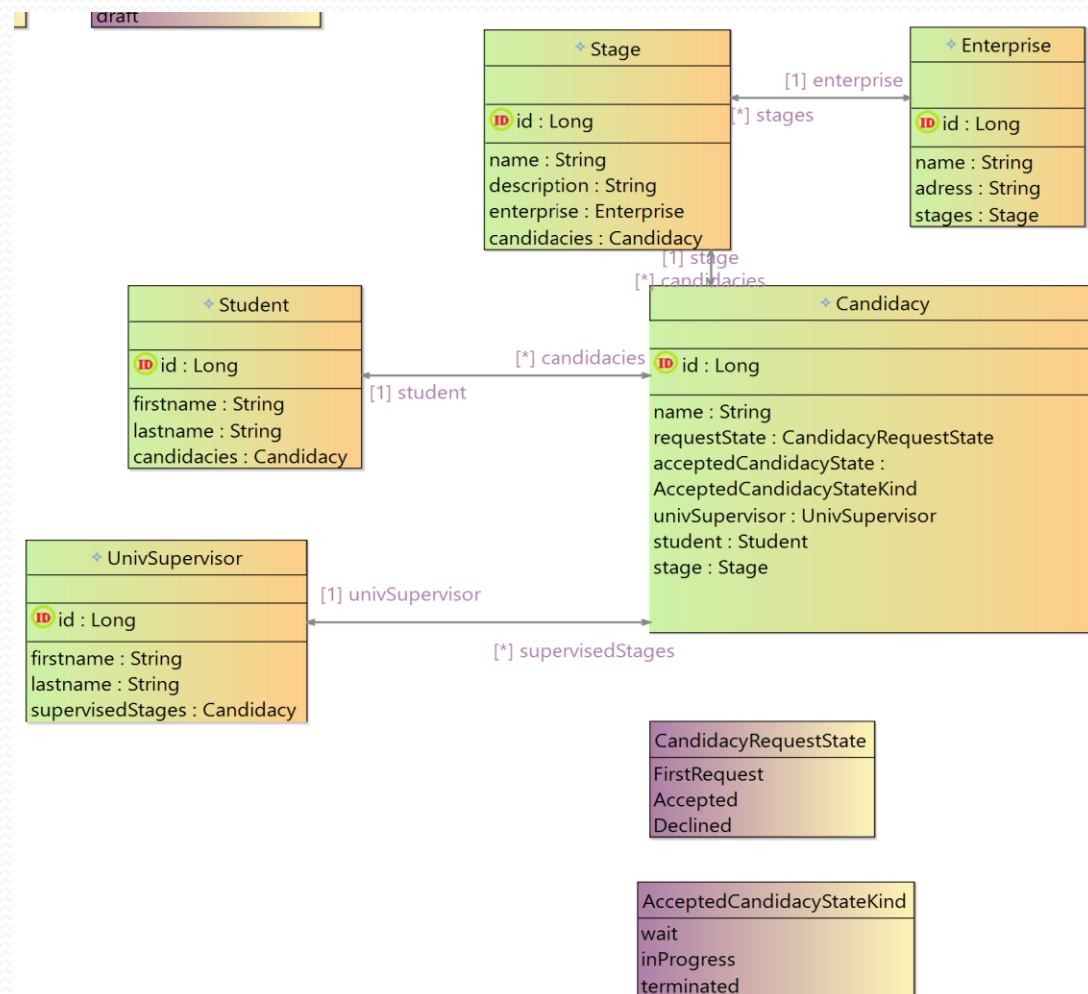
Identification des composants
Et réalisation d'une maquette

Application gestion des stages

- Description plus complète :
 - 11.DescriptionGestionStages.pdf
- Permet aux entreprises de proposer des stages, et aux étudiants de consulter les stages disponibles.
- Un étudiant peut candidater à un stage
- Un stage doit avoir un encadrant universitaire
- L'appli permet d'affecter un encadrant
- Il y a d'autres fonctionnalités ...
 - Voir le doc de présentation de l'appli

Les principales entités

- Student
- UnivSupervisor
- Stage
- Enterprise
- Candidacy



Atelier

Identification des composants et maquette

- Concevoir et réaliser la maquette des premiers composants de votre application
 - En suivant la méthodologie vue précédemment
 - Sur quelques pages de l'application
- Décrivez graphiquement sur papier (ou outils de dessin) les pages de votre applications
- Identifiez les composants
- Concevez les composants en React
 - Uniquement du rendu
 - Ils ne sont pas fonctionnels
- Tips :
 - Pour chaque entity, on retrouve des pages similaires (CRUD)
- Ex:
 - `showAllStudent()`, `showOneStudent()`
`updateStudent()` ...

