Développement front

MMI 2 – TP#1 S4

IUT CERGY-PONTOISE

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

James Ward
@_JamesWard

Follow

Discovered this sign at the W3C headquarters...

8:17 AM - 11 Feb 2015

1,992 Retweets **1,037** Likes

40 2.0K 1.0K

Eco-système javascript natif

- Très limité par défaut
- Expansible grâce à des bibliothèques tierces (appelée également librairies)
  - Ajoutent de nouvelles fonctionnalités
  - Facilitent le développement

Point technique – Frameworks et librairies

- **Ce ne sont pas des synonymes**
- **Framework :** Propose une boite à outils, des concepts à adapter à ses besoins
  - Exemples : Angular, Symfony (PHP)...
- **Library (ou bibliothèque) :** Permet de réaliser un ensemble fini de choses
  - Exemples : lodash, jQuery, reactjs...

Source(s) :

- https://www.les-metiers-du-web.com/quelle-est-la-difference-entre-un-framework-et-une-library/
- https://stackoverflow.com/questions/148747/what-is-the-difference-between-a-framework-and-a-library

Point technique – Frameworks et librairies

Frameworks (et certaines librairies) nécessitent une connaissance du langage (ici javascript), sous peine de gros problèmes pour en tirer quelque chose

Source(s) :

- https://www.les-metiers-du-web.com/quelle-est-la-difference-entre-un-framework-et-une-library/
- https://stackoverflow.com/questions/148747/what-is-the-difference-between-a-framework-and-a-library

React js

- **Librairie** javascript open source
- Développé et maintenu par facebook
- Très populaire
  - Utilisé par facebook, instagram, reddit…
- Sorti en 2013

Source(s) :

- https://fr.reactjs.org/

React js

- Aide à réaliser des interfaces utilisateurs
  - Simplifie énormément la manipulation du DOM (structure HTML)
- Utile pour développer des outils ou des sites entiers
- Documentation détaillée et **en français**
- **Très recherché dans le monde du travail**

Source(s) :

- https://fr.reactjs.org/

Pratiquons ! - Découvrons ReactJS (Partie 1)

Pré-requis :

- Avoir la ressource ressources/react

A télécharger ici :
https://download-directory.github.io?url=https://github.com/DanYellow/cours/tree/main/developpement-front-s4/travaux-pratiques/numero-1/ressources

Analysons tout ça ensemble

Notre première application

```
<div id="root"></div>

<script type="text/babel">
    function MyApp() {
        return <h1>Hello, world!</h1>;
    }

    const container = document.getElementById('root');
    const root = ReactDOM.createRoot(container);
    root.render(<MyApp />);
</script>
```

Notre première application

```
const container = document.getElementById('root');
const root = ReactDOM.createRoot(container);
root.render(<MyApp />);
```

- Permet d'initialiser l'application React
- Indique notre composant racine (ici <MyApp>)
  - Le composant qui va contenir notre application

Notre première application

```
<div id="root"></div>
```

- Conteneur de notre application React
- **Note :** une même page peut avoir plusieurs applications indépendantes
  - Il faudra penser à dupliquer également les lignes de la slide précédente

Notre première application

```
<script type="text/babel">
```

- Permet d'utiliser babel
  - Outil permettant d'utiliser des fonctions avancées de javascript sur des navigateurs non compatibles
- Permet d'utiliser le jsx (JavaScript XML)

Source(s) :

- https://en.wikipedia.org/wiki/JSX_(JavaScript)
- https://en.wikipedia.org/wiki/Babel_(transcompiler)

Notre première application

```
function MyApp() {
    return <h1>Hello, world!</h1>;
}
```

- **Composant** de notre application
  - Ici il s'appelle MyApp (la casse est importante)

Composant

- Cœur de ReactJS
- **Réutilisable**, Testable et Maintenable
- S'exprime sous forme de fonction javascript
- Un composant React peut contenir d'autres composants React, etc.
- Peut retourner du JSX

Composant

```
/*
    Syntaxe moderne pour écrire des fonctions
    et donc des composants React
*/
const MyComponent = () => {
    return (
        <a href="">
            <img src="mon-image.jpg" alt="" />
        </a>
    )
}

function OtherComponent () {
    return (<p>Bonjour MMI !</p>)
}
```

Composant – Imbrication de composants

```jsx
const FicheEtudiant = () => {
    return (
        <article className="fiche-etudiant">
            <figure>
                <img src="img-042.jpg" alt="" srcset="" />
            </figure>
            <p>Prénom : XXX - Nom : XXX</p>
        </article>
    )
}

const FicheEtudiantComplete = () => {
    return (
        <section>
            <FicheEtudiant />
            <p>[...]</p>
        </section>
    )
}
```

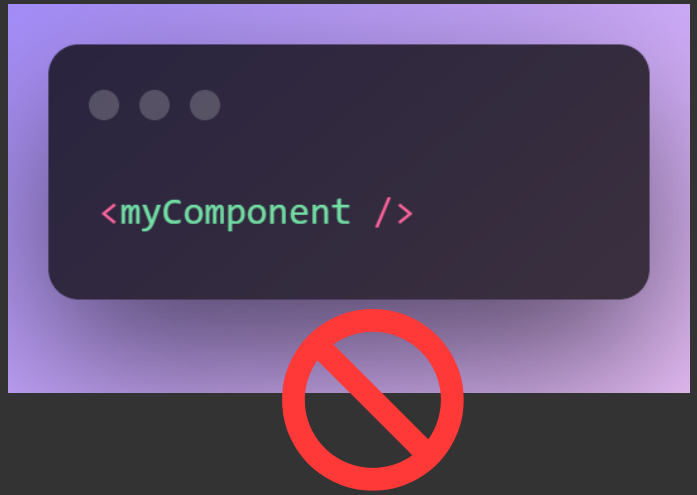
Dans le code ci-contre, le composant <FicheEtudiant /> est affiché par le composant <FicheEtudiantComplete />

Composant

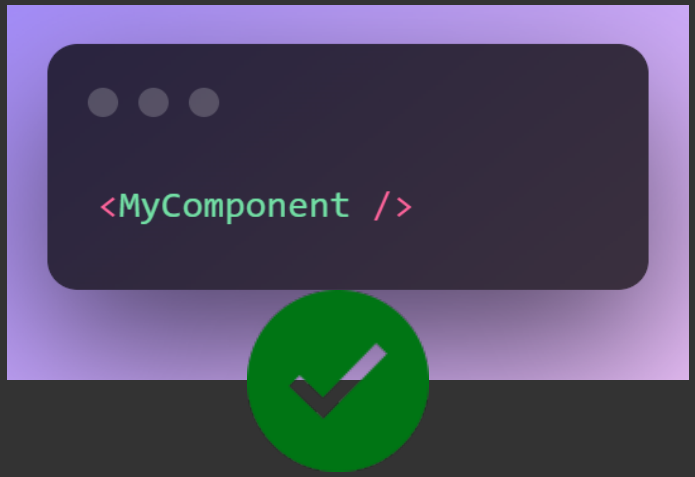
- **Doit impérativement retourner** quelque chose :
  - Chaîne de caractères / nombre
  - JSX
  - null
  - ...

Nommage composant – Bonnes pratiques

- Nommez-les en PascalCase



Interdit
Le composant n'est pas en PascalCase



Correct
Le composant est en PascalCase

Découpage composant – Bonnes pratiques

- Pour rendre vos composants réutilisables. Pensez à les découper.

Chaque élément coloré est un composant. On voit que certains sont réutilisés.

JSX

- Langage inspiré par HTML
  - Ce que vous pouvez faire en HTML, vous pouvez me faire en JSX (avec parfois des variantes)
- Syntaxe moderne pour écrire des composants React
  - Utilisation facultative
- A besoin de babelJS pour être interprété par les navigateurs

Source(s) :

- https://fr.reactjs.org/docs/introducing-jsx.html
- https://www.youtube.com/watch?v=SFFZ0hpIk5Q - moins de 15 minutes

JSX

```jsx
const myVar = "Bonjour";
// JSX
const el = <div className="conteneur">{myVar}</div>;

// Non-JSX
const el = React.createElement('div', { className: "conteneur" }, myVar);
```

La syntaxe JSX n'est-elle pas plus concise et claire ?

Source(s) :

- https://fr.reactjs.org/docs/introducing-jsx.html
- https://www.youtube.com/watch?v=SFFZ0hpIk5Q - moins de 15 minutes

JSX

Accéder à un outil montrant la conversion jsx → javascript

Source(s) :

- https://fr.reactjs.org/docs/introducing-jsx.html
- https://www.youtube.com/watch?v=SFFZ0hpIk5Q - moins de 15 minutes

JSX – Différences avec le HTML

- Certains attributs changent de noms ou syntaxes
  - Exemple : class → className
- La barre oblique finale est obligatoire si ce n'est pas une balise pairée (voir exemples suivants)

Source(s) :

- https://fr.reactjs.org/docs/introducing-jsx.html
- https://www.youtube.com/watch?v=SFFZ0hpIk5Q - moins de 15 minutes

JSX – Différences avec le HTML

- Le jsx permet d'afficher du javascript (voir exemples suivants)
- Un composant ne peut retourner qu'une seule balise parente à la fois (voir exemples suivants)

Source(s) :

- https://fr.reactjs.org/docs/introducing-jsx.html
- https://www.youtube.com/watch?v=SFFZ0hpIk5Q - moins de 15 minutes

JSX – Différences avec le HTML

```jsx
const MyBadComponent = () => {
    return (
        <a href="">
            <img src="mon-image.jpg" alt="" />
        </a>
        <p>Ce composant est mauvais</p>
    )
}
```

Interdit

Le composant retourne plusieurs balises en même temps

```jsx
const MyGoodComponent = () => {
    return (
        <div>
            <a href="">
                <img src="mon-image.jpg" alt="" />
            </a>
            <p>Ce composant est mauvais</p>
        </div>
    )
}
```

Correct

Le composant ne retourne qu'une seule balise à la fois

(Il est possible de remplacer par <></>)

Source(s) :

- https://fr.reactjs.org/docs/introducing-jsx.html
- https://www.youtube.com/watch?v=SFFZ0hpIk5Q - moins de 15 minutes

JSX – Différences avec le HTML

```jsx
const MonComposant = () => {
    const age = 25;
    return (
        <p>Bonjour, j'ai {age} ans.</p>
    )
}
```

En jsx, les variables sont affichées entre accolades ({})
(ça fonctionne également avec les attributs)

Source(s) :

- https://fr.reactjs.org/docs/introducing-jsx.html
- https://www.youtube.com/watch?v=SFFZ0hpIk5Q - moins de 15 minutes

JSX – Différences avec le HTML

```
function MyApp() {
    return <MyComponent>;
}
```

Interdit

Le composant n'a pas de barre oblique (/) finale

```
function MyApp() {
    return <MyComponent />;
}
```

Correct

Le composant a une barre oblique finale

Source(s) :

- https://fr.reactjs.org/docs/introducing-jsx.html
- https://www.youtube.com/watch?v=SFFZ0hpIk5Q - moins de 15 minutes

Notre première application

```
<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>
```

- Scripts nous permettant d'utiliser React
- Note : Cette structure est utilisée à titre d'exemple, personne ne fait ça dans le monde professionnel

Pratiquons ! - Découvrons ReactJS (Partie 2/3)

Pré-requis :

- Avoir la ressource ressources/react

A télécharger ici :
https://download-directory.github.io?url=https://github.com/DanYellow/cours/tree/main/developpement-front-s4/travaux-pratiques/numero-1/ressources

ReactJS

- Bibliothèque javascript très populaire
- Permet de manipuler plus facilement le DOM
- Basé autour de composants
  - Fonctions devant retourner quelque chose : élément React, null, chaîne de caractères...

ReactJS

- Les composants se veulent dynamiques et communicants
  - Grâce aux props et aux states

React n'est pas la solution à tout

- React est un outil formidable mais parfois il ne sera pas très utile pour vos projets
  - Il peut même causer plus de problèmes qu'il n'en résout

React n'est pas la solution à tout

- Quand utiliser react :
  - Mon UI possède beaucoup d'interactions
    - Ex : Je clique sur un élément, ça charge des données, change l'UI...
  - Mon site possède un widget lent qui doit être modernisé

React n'est pas la solution à tout

- Quand **ne pas** utiliser react :
  - Mon projet est trop simple
    - Ex : mon portfolio
  - Je ne connais pas react / javascript
    - Il est préférable d'avoir des connaissances plus solide
  - Je n'ai pas de serveur node pour gérer le référencement

React n'est pas seul

- Il existe des alternatives à React
  - Angular : développé par Google
  - Vuejs
  - Svelte
  - ...

Il n'y a pas un outil meilleur qu'un autre, son utilisation dépendra de vos besoins et affinités avec l'outil.
Les slides concernant l'utilisation ou non de React s'appliquent également ici.

En bref,
ce n'est pas parce qu'un outil est à la mode qu'il faut l'utiliser.

Questions ?