React JSX

JSX est une syntaxe facilitant l'écriture du code des composants, il sécurise également l'affichage des valeurs en échappant tous les caractères spéciaux (attaque XSS).

On utilise la casse camleCase en JSX.

```
const el = <div className="main">Main</div>;
```

Exemple de JSX

En JSX on écrit :

```
const el = (
  <h1 className="main">
    Hello, world!
  </h1>
);
```

En JS on écrirait (...) :

```
const el = React.createElement(
  'h1',
  {className: 'main'},
  'Hello, world!'
);
```

Tous les développements par la suite se feront en JSX dans ce cours.

La syntaxe JSX nécessite un compilateur (babel) que nous importons directement dans le fichier index.html pour compiler le code JSX en JS compréhensible par votre navigateur.

Avec les navigateurs modernes le code JSX sera théoriquement correctement compilé.

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8" />
    <title>Hello React</title>
    <script src="https://unpkg.com/react@16/umd/react.development.js"></script>
    <script src="https://unpkg.com/react-dom@16/umd/react-dom.development.js"></script>

    <!-- babel => compilation du JSX -->
    <script src="https://unpkg.com/babel-standalone@6.15.0/babel.min.js"></script>
    <style>
```

```
        .heading{
            color: purple;
        }
    </style>
</head>
<body>
    <div id="root"></div>
    <script type="text/babel">

        // Component
        const Hello = (props) => {
            return (
                <div className="heading" >
                    <h1>{props.message}</h1>
                    <p>{props.subtitle}</p>
                </div>
            )
        }

        // Rendu dans le DOM
        ReactDOM.render(
            <Hello message = "Hello React" subtitle="Enjoy ! "/>,
            document.getElementById('root')
        );
    </script>
</body>
</html>
```

Attributs

Ils s'écrivent soit dans des cotes simples ou doubles lorsqu'on a des chaîne de caractères comme valeur ou bien des accolades lorsqu'on a une expression en JS :

```
const el1 = <div className="main">Main</div>;

const el2 = <div style={{ backgroundColor : 'red' }}>Main</div>;
```

Les balises enfants

Attention vous devez toujours mettre un div wrapper pour déclarer des éléments enfants, il faut se rappeler que la syntaxe JSX est du JS. Nous verrons ultérieurement que ces div parfois inutiles seront remplacées par du code React que l'on appelle des fragments.

```
const el3 = (
  <div>
    <h1>Bonjour !</h1>
    <h2>Le Monde.</h2>
  </div>
);
```

Attatque XSS

La syntaxe JSX permet de se protéger des injections XSS, les caractères spéciaux seront échapés. Voyez l'exemple ci-dessous :

```
const content = <script>alert('xss')</script>;
// Ceci est sans risque :
const el4 = <h1>{content}</h1>;
```

Parcourir un tableau JS

Pour parcourir un tableau JS dans le rendu de votre composant vous ne pourrez utiliser que la méthode map sur celui-ci. De plus chaque élément a besoin d'une clé unique pour être identifié par React dans le DOM. La clé key est donc un mot réservé dans la librairie React, elle ne peut être récupérée :

```
 // Component
const Hello = (props) => {
    const elems = ['a', 'b', 'c', 'd' ];
    return (
        <div className="heading" >
        {elems.map((elem, i) => {

            return (
                <p key={i}>{elem}</p>
            )
        })}
        </div>
    )
}
```

React dépend de deux structures de données importantes

React dépend des collections **Map** et **Set** en JS. Ces collections sont basées sur un système de clé/valeur, elles sont utilisées pour des questions d'accès à des valeurs et d'optimisation. Un Set n'a pas de doublon possible par exemple. Et un map est une structure de données construite avec un système de clé/valeur.