Structure et syntaxe JSX

En React, le contenu des pages web est construit à partir de composantes. Chaque composante est un élément HTML (et ses éléments enfants s'il en a) retourné par une fonction javascript.

Le code suivant dans le fichier **App.js** affichera un message simple:

```
function App() {
  return "Allo le monde";
}
```

Le langage utilisé par React n'est pas du javascript pur cependant: on le nomme JSX et c'est un langage plus approprié pour générer du HTML à partir de fonctions. Un de ses particularités est qu'on peut éliminer les guillemets lorsque qu'on retourne des éléments HTML:

```
function App() {
  return <h1>Allo le monde</h1>;
}
```

À des fins de lisibilité, il est souvent préférable d'utiliser l'indentation dans des structures HTML plus complexes. Pour faciliter la lecture, on met donc généralement le code retourné par la fonction entre parenthèses, sur plusieurs lignes, comme suit:

```
return (
    <a href="https://www.google.com">
        <p>Un lien vers Google</p>
    </a>
);
```

Attention au point-virgule à la fermeture de la parenthèse.

Une composante retournée par une fonction React doit être entièrement contenue dans un seul élément HTML. Par exemple le code suivant est illégal:

```
return (
    <a href="https://reactjs.org/">
        <p>React</p>
    </a>
    <a href="https://nodejs.org/">
        <p>Node.js</p>
    </a>
);
```

Pour corriger ce problème, il faut les inclure dans le même élément, par exemple un **div**:

```
return (
    <div>
      <a href="https://reactjs.org/">
          <p>React</p>
      </a>
      <a href="https://nodejs.org/">
          <p>Node.js</p>
      </a>
    </div>
);
```

Il est aussi possible de stocker le code HTML dans des variables:

```
function App() {
  const composante = (
    <div>
      <p>un DIV dans une variable</p>
    </div>
  );
  return composante;
}
```

Attributs

On peut utiliser les attributs habituels pour définir des styles. Par exemple avec la feuille de style suivante:

```
#s1 {
  color: #ff0000;
}

#s2 {
  color: #008000;
}

.citation {
  color: #494949;
  font-style: italic;
}

.titre {
  font-weight: bold;
  font-family:'Times New Roman', Times, serif;
}
```

On peut utiliser les identifiants comme d'habitude:

```
return (
    <div>
        <p id="s1">Texte en rouge</p>
        <p id="s2">Texte en vert</p>
    </div>
);
```

Cependant les classes ne peuvent pas utiliser le terme class **car ce mot est reservé en javascript**. Il faut donc utiliser `className`, comme suit:

```
return (
    <div>
        <p className="titre">Ceci est un titre</p>
        <p className="citation">Ceci est une citation</p>
    </div>
);
```

Enfin, pour des raisons de lisibilité encore une fois, il est possible de répartir sur plusieurs lignes les attributs d'un élément HTML, comme suit:

```
return (
    <div>
        <a
          href="http://www.react.org"
          id="s1"
          className="titre"
        >
          Un lien
        </a>
    </div>
);
```

Agrégation

En JSX, les fonctions ont le nom des composantes qu'elles retournent. Par exemple dans le fichier index.js on a le code suivant:

```
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);
```

Ici l'élément `<App />` correspond à un appel de la fonction `App()`. Ainsi, il nous est possible de définir nos propres fonctions où chacune correspond à une composante de la page, et d'appeler chacune dans la fonction principale:

```
function CompA() {
  return (
    <div>Composante A</div>
  );
}

function CompB() {
  return (
    <div>Composante B</div>
  );
}

function App() {
  return (
    <div>
        <CompA />
        <CompB />
    </div>
  );
}
```

Injection

Il est très facile de passer des expressions javascript à l'intérieur de JSX: il suffit de les mettre entre accolades:

```
function App() {
  const deuxPlusDeux = 2+2;
  return (
    <div>
        <p>1 + 1 = {1+1}</p>
        <p>2 + 2 = {deuxPlusDeux}</p>
    </div>
  );
}
```

Ces expressions peuvent être des opérations, des variables et même des fonctions:

```
function clic(e) {
    alert("Clic!");
}

function App() {
  return (
    <div>
```

```
        <button onClick={clic}>Bouton</button>
      </div>
    );
  }
```