Introduction

React, c’est quoi ?

Introduction

Qu'est-ce que React ?

- Une bibliothèque JavaScript qui a été initialement créée par Facebook en 2011.
  - Open-Source depuis Mai 2013.
  - Développé par : Facebook, Instagram et la communauté.
- Permet de construire des interfaces pour des utilisateurs.
- Une structure basé sur des composants déclaratifs
  - Le rendu d'un composant est généré sur base des valeurs de l'état local.
  - Lorsqu'une valeur est modifiée, le rendu est automatiquement actualisé.
- L'interaction avec le DOM est extrêmement rapide via un DOM Virtuel.
- Intégration possible dans une application Web existante *(ASP.Net, PHP, Java, ...)*

Une bibliothèque ? Pas un Framework ?

React a été conçu comme étant une bibliothèque et non un framework, contrairement à ses concurrents (Angular, VueJS, ...).

Le fonctionnement de React est uniquement dédié à la gestion de l'interface de l'application. Il peut être considéré comme la vue dans le modèle MVC.

Par ailleurs, la bibliothèque de React n'intègre pas de mécanisme de routage, de gestionnaire de service, de générateur de code (CLI).

Flexibilité dans le développement

A part pour la génération des vues, React n'impose pas de technologie ou de pattern.
Le développeur a donc le contrôle total de son code et de ses dépendances.

En fonction des besoins, il est possible d'ajouter des bibliothèques tierces pour enrichir les fonctionnalités de l'application lors de son développement.

Cette philosophie permet de bénéficier de plus de liberté qu'avec un framework.

Par exemple :
« react-router » ajoute un mécanisme de routage dans l'application.
« redux & react-redux » permet la mise en place d'un gestionnaire de données.

Langages de programmation utilisés

Pour développer des applications React, nous allons principalement utiliser les langages de programmation suivants :

- Le JavaScript (ECMAScript 2015+)
  Langage qui nous permettra de coder la logique de nos composants.
- Le JSX
  Une extension du Javascript qui permet réaliser le rendu des composants.

Il est possible d'utiliser le langage TypeScript à la place du JavaScript.
Cheatsheet TS pour React : https://react-typescript-cheatsheet.netlify.app

JavaScript & ECMAScript

Introduction

ECMAScript

L'ECMAScript (ECMA-262) est la standardisation du Javascript (depuis 1997).

La spécification du langage a stagné quelques années en version 5 (publier en 2009). En 2015, la nouvelle norme (ES6 / ES2015) a été publiée, celle-ci apportant énormément de nouveautés. Depuis celle-ci, la norme est mise à jour chaque année.

L'implémentation de l'ECMAScript dans le Javascript est mise en œuvre par la fondation Mozilla .

Plus d'info:
developer.mozilla.org/fr/docs/Web/JavaScript/JavaScript_technologies_overview

ECMAScript - Les fonctions fléchées

L'écriture fléchée (=>) offre une syntaxe raccourcie pour créer des fonctions.

En déclarant la fonction sous forme de constante, son unicité est assurée.

Les fonctions fléchées permettent également de créer des expressions lambda.

```
// ES5
var myFn = function(x) {
    return x + 1;
};

// ES2015+
const myFn = x => {
    return x + 1;
};

// Lambda expression
const myFn = (x) => x + 1;
```

ECMAScript - Le destructuring

Permet d'assigner des variables depuis les valeurs d'un objet ou d'un tableau.

- Objet

Le nom des variables à assigner doit être identique aux noms des propriétés.

- Tableau

L'assignation des variables est basé sur la position des éléments dans le tableau.

```
// Destructuring d'objet
const myObject = {
    number: 42,
    message: "Hello world!"
};

const {number, message} = myObject;

// Destructuring du tableau
const tab = ['A', 'B', 'C', 'D']

const [valA, valB] = tab;
```

ECMAScript - Le paramètre « Rest »

Permet de créer un tableau avec un nombre indéterminé d'argument.
Pour l'utiliser, il faut appliquer le préfixe « ... » sur le dernier argument.

```
const combine = function(separator, ...tabs) {
    return tabs.join(separator);
}

const combinedString = combine("-", "Riri", "Fifi", "Loulou");
```

ECMAScript - Le paramètre « Rest »

★ Exemple d'utilisation du paramètre « Rest » avec du destructuring

```
//Destructurer un tableau
const myTeam = ["Donald", "Daisy", "Riri", "Fifi", "Loulou"];

// ES5
const first = myTeam[0];       //"Donald"
const second = myTeam[1];      //"Daisy"
const rest = myTeam.slice(2); //["Riri", "Fifi", "Loulou"]

// ES2015
const [first, second, ...rest] = myTeam; //Même résulat
```

ECMAScript - L'opérateur « Spread »

Également appelé *"Opérateur de décomposition"*. Il permet de décomposer un objet itérable. Pour l'utiliser, il faut appliquer le préfixe « ... » sur l'objet à décomposer.

```
// Exemple de décomposition
const myString = "Riri";

const myArray = [...myString];
// myArray => ["R", "i", "r", "i"]
```

ECMAScript - L'opérateur « Spread »

★ Exemple de l'utilisation de l'opérateur « Spread »

```
// Concaténer plusieurs éléments itérables
const neveux = ["Riri", "Fifi", "Loulou"];

// ES5
const team = ["Donald", "Daisy"].concat(neveux);

// ES2015+
const team = ["Donald", "Daisy", ...neveux];

// Dans les 2 cas le resultat sera :
// ["Donald", "Daisy", "Riri", "Fifi", "Loulou"]
```

Manipulation de tableau de données

Le langage JavaScript définit des méthodes qui permettent de manipuler les tableaux de manière simple.

Parmi celle-ci, nous allons souvent utiliser les méthodes suivantes en React :

- Array.prototype.filter()
- Array.prototype.map()

Array - Méthode « filter »

La méthode « filter » permet d'obtenir un nouveau tableau contenant tous les éléments du tableau remplissant la condition déterminée par le « callback ».

```
// Liste de données
const myTeam = ['Donald', 'Daisy', 'Riri', 'Fifi', 'Loulou'];

// Filtre via une fonction
const myFilter = function(world) {
    return world.includes("o");
}
const r1 = myTeam.filter(myFilter);      // ["Donald", "Loulou"]

// Filtre via une fonction Lambda
const r2 = myTeam.filter(p => p.length < 5); // ["Riri", "Fifi"]
```

Array - Méthode « map »

La méthode « map » permet d'obtenir un nouveau tableau contenant tous les éléments modifiés par l'exécution de la méthode « callback ».

```
// Liste de nombres
const myValues = [1, 2, 3, 4, 5];

// Utilisation à l'aide d'une fonction Lambda
const myDoubles = myValues.map(n => n * 2);
// myDoubles => [2, 4, 6, 8, 10]
```

Exemple d'utilisation avec React : Elle permet de transformer facilement un tableau d'objet (en JS) en tableau de Composant (en JSX), pour la création d'une liste.

Le JSX

Introduction

Le JSX

Le JSX est une syntaxe d'extension du Javascript permettant de faciliter l'écriture du rendu des composants React. Cette syntaxe est très similaire à la syntaxe du HTML.

```
const element = (
    <div>
        <h1 className="classHtml">Mon Titre</h1>
        <img src={user.avatarUrl} />
    </div>
);
```

L'écriture JSX utilise le *"camelCase"* pour les attributs des éléments.
L'attribut HTML « **class** » devient « **className** », car le mot "class" est réservé.

Exemple précédent en Javascript

Brève explication des paramètres de la méthode « createElement » :

- Premier paramètre (string) :
  Le type de la balise
- Deuxième paramètre (objet) :
  Les attributs de la balise
- Les paramètres suivants (...objet) :
  Les éléments à placer dans la balise

Note: Depuis la version 17 de React, une nouvelle méthode transformation est utilisée.

```
const element = React.createElement(
    "div",
    null,
    React.createElement(
        "h1",
        { className: "classHtml" },
        "Mon Titre"
    ),
    React.createElement(
        "img",
        { src: user.avatarUrl }
    )
);
```

Utilisation de JSX avec le DOM

- Affichage d'un élément JSX à l'aide de ReactDOM

page.html

```html
<html>
  <body>
    <div id="container"></div>
  </body>
  <!--Resource React nécessaire-->
  <script src="script.js"></script>
</html>
```

script.js

```
const element = <div>Hello World</div>;

ReactDOM.render(
  element,
  document.getElementById('container')
);
```

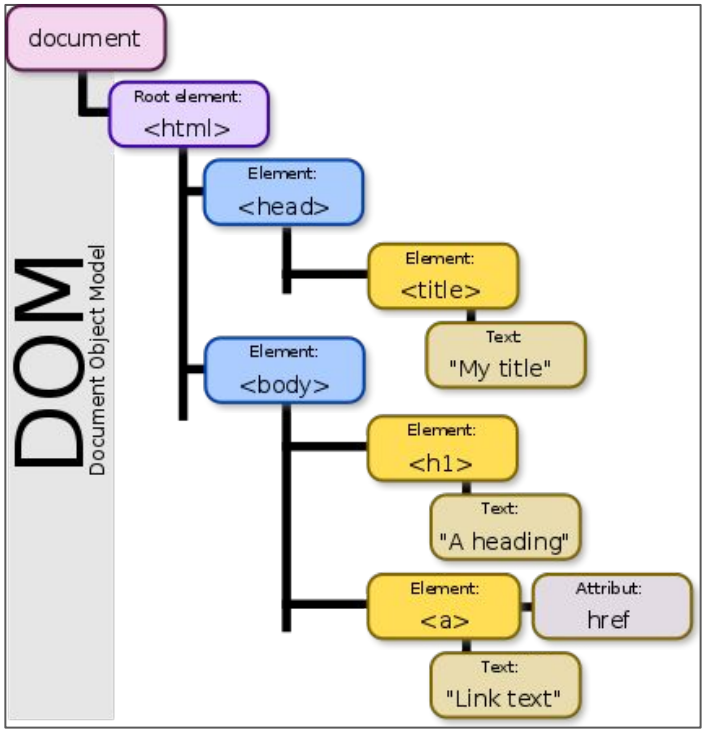
Le DOM Virtuel

Introduction

Document Object Model

Le DOM est une API qui permet d'interagir en JavaScript avec les documents HTML ou XML.

La représentation du document est un arbre nodal. Chaque nœud représente une partie du document.



Le DOM virtuel

Interagir avec le DOM devient vite lourd et complexe. Pour éviter cela, React utilise un DOM virtuel qui est une représentation du DOM sous forme d'objet JavaScript.

Le DOM virtuel est créer à partir des méthodes de rendu des composants
et celui-ci est régénéré après chaque changement d'état de l'application.

Les avantages sont :

- La génération du DOM virtuel est extrêmement rapide.
- Analyse l'évolution du DOM virtuel pour définir les modifications à réaliser.
- React réalise le minimum de modifications sur le DOM à l'aide d'opérations simples.

L'environnement de développement

Introduction

L'environnement de développement

Pour développer en React, nous allons utiliser Node JS

- C'est une plateforme de développement JavaScript (en dehors des navigateurs).
- Permet de configurer les dépendances du projet à l'aide de « npm ».

Pour écrire notre code, nous pouvons utiliser par exemple :

- Visual Studio Code
- Webstorm (Payant)
- Sublime Text 3

L'environnement de développement

Le plugin “React Developer Tools” est un outils qui permet :

➔ Inspecter l’arborescence des composants React

➔ Voir l’état de « Props » et du « State » d’un composant.

Le plugins s'intègre dans les outils de développement du navigateur → Onglet React

“React Developer Tools” est disponible pour Firefox et Chrome