Cadriciel Web côté client

Cours 4 - Introduction à React

Table des matières

Définition du DOM

Le terme DOM veut dire : Document Object Model.

Celui-ci représente la structure d'une page HTML (interface Web) de manière hiéarchique et permet à des languages de programmation ou des Cadriciels de le manipuler. Javascript et JQuery sont des choix populaires, et relativement datés mais toujours utilisés en entreprise.

Respectivement, changer le DOM change l'aspect visuel et/ou des propriétés de l'interface Web.

La différence entre le DOM normal et le DOM Virtuel de React

Le DOM Virtuel permet d'optimiser la responsabilité de générer le HTML final à partir de quelle instance (l'utilisateur final ou le serveur).

Celui-ci accomplit toutes les manipulations au côté serveur et envoie le HTML final à l'utilisateur final pour qu'il ait une expérience plus conviviale (rapidité accrue).

La différence entre le DOM normal et le DOM Virtuel de React

Le DOM Virtuel à un désavantage, mais qui est vis-à-vis à l'apprivoisement de React (ou peu importe le Cadriciel de même type) et qui se règle dès la comprhéension faite.

Au niveau du paradigme, il faut éviter d'utiliser de la manipulation directe du DOM (Javascript ou JQuery).

Pourquoi ? Parce que les variables (Hooks) sont **réactives et changent le DOM automatiquement**.

La différence entre le DOM normal et le DOM Virtuel de React

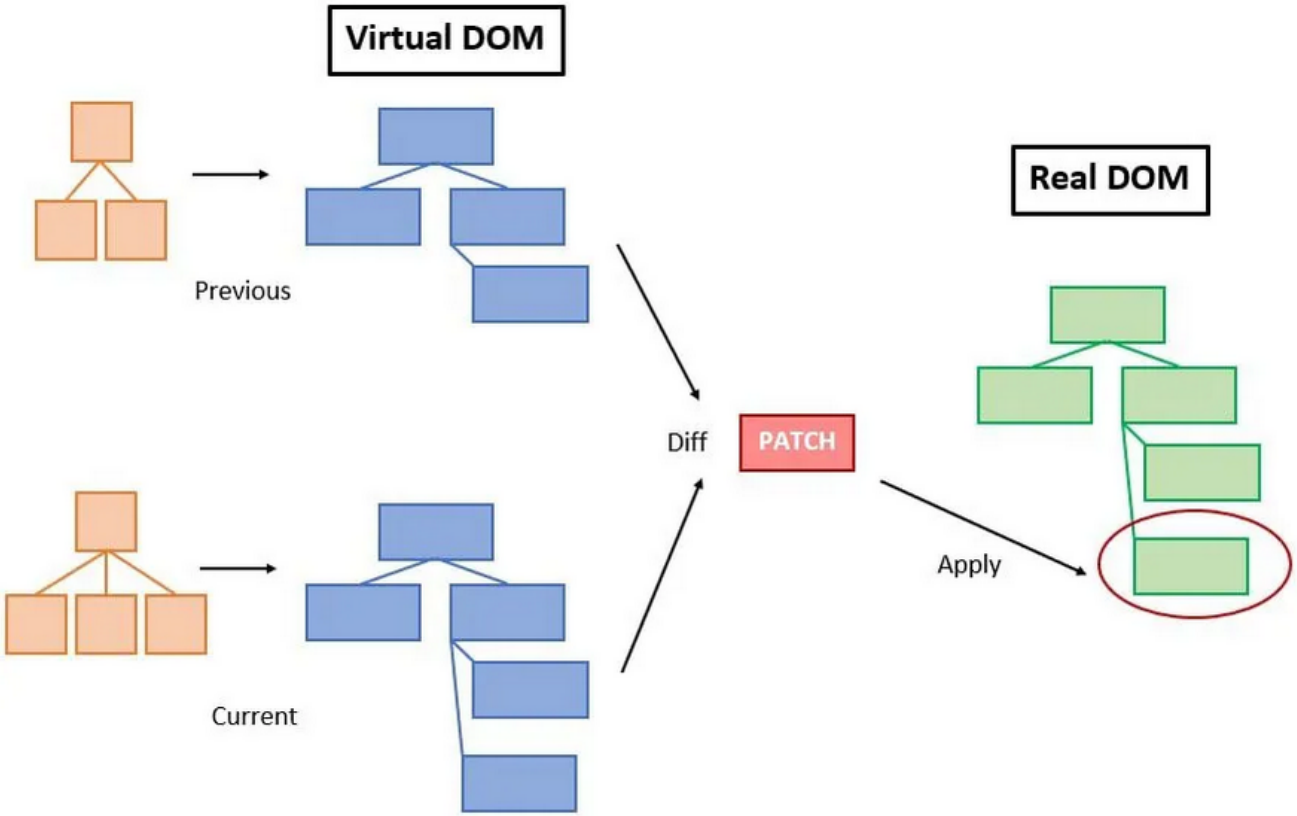


Figure: https://medium.com/cstech/demystifying-javascript-virtual-dom-a-guide-for-web-developers-fae7dd9d0cd0

Structure d'un fichier TSX (React Typescript)

La structure principale d'un fichier pour un Component est comme suit (dans cet ordre) :

- L'annotation client (lorsqu'on utilise des Hooks ou autre variables React
- Les imports
- La fonction (exportée par défaut normalement)
- Les variables Javascript et les Hooks
- Le code Javascript/Typescript
- useEffect() (Le rendu au début lors du premier chargement de page)
- return() (Rendu du HTML après l'exécution de la logique JS/TS

Structure d'un fichier TSX (React Typescript)

```tsx
"use client"; // Necessary to declare a client side Component.
import { useState, useEffect } from "react"; // Module importation.

export default function Structure() { // An auto exported Component.
  const [count, setCount] = useState(0); // A Hook React variable.

  // All this space for Javascript/Typescript normal programming.

  // The useEffect renders only at first mount (load).
  useEffect(() => { setCount(42); }, []);

  return ( // Also known as the Render() in traditional React. Returns HTML.
    <> {/* You NEED everything as one node, thus this special empty node */}
      <div className="grid place-items-center h-screen">
        <p>My number: {count}</p> {/* You need variables in brackets */}
      </div>
    </>
  );
}
```

Les variables React de base

Voici les deux variables React fondamentales abordées dans cette séance :

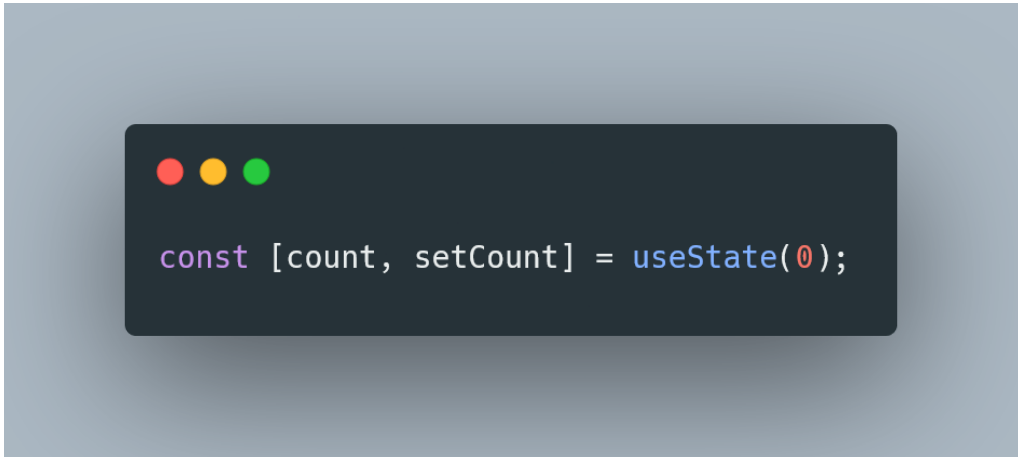
- useState() : les hooks, les variables reactive qui changent automatiquement dans le DOM
- useEffect() : le code enclanché au début lors du premier démarrage de la page Web

Les Hooks (variables "Réactive") : useState()

count : variable réactive déclarée pour son utilisation

setCount : un mutateur (setter) pour modifier la variable réactive "count"

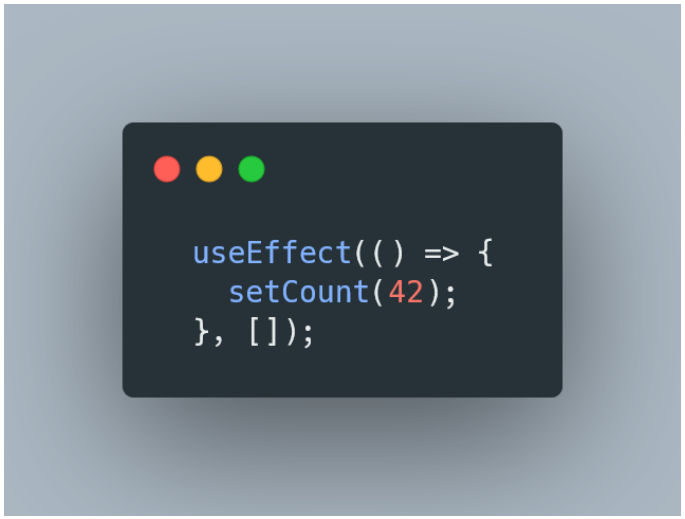
useState(0) : déclaration obligatoire, le zéro (0) est la valeur par défaut



Exécution de départ au démarrage de la page : useEffect()

useEffect contient un corps pour y mettre notre code que l'on veut exécuter seulement une fois au début lors du démarrage de l'application Web.

Le tableau à la fin ("[]") est pour mettre des variables qui doivent être incluses dans le scope du useEffect. Attention, desfois inclure une mauvaise variable peut faire répéter l'application un nombre infini de fois et créer un boucle infinie. Normalement, on inclue des variables props (abordé plus tard dans le cours).



Bibliographie

- `https://react.dev/`