Qu'est-ce que React.js ?

Sur le site web officiel de React, on le décrit comme une "bibliothèque pour créer des interfaces utilisateur pour le Web et les applications mobiles." En effet, React.js est une bibliothèque JavaScript qui permet de développer des interfaces utilisateur interactives et dynamiques.

Un exemple frappant de l'utilisation de React est celui de Netflix, où les transitions entre les pages sont extrêmement fluides, presque instantanées, et où il n'y a pas de chargement de page perceptible entre les actions de l'utilisateur. Cette expérience utilisateur se rapproche de celle que l'on retrouve dans les applications mobiles, grâce à des réactions immédiates aux interactions de l'utilisateur. Cela a établi de nouvelles normes dans le développement web, et des bibliothèques comme React ont grandement contribué à faciliter la création de telles applications.

JavaScript est capable de modifier une page Web sans avoir à la recharger, ce qui lui permet d'apporter des modifications dynamiques à l'interface utilisateur en fonction des actions de l'utilisateur, telles que le clic sur un bouton ou le changement d'onglet.

Mais alors, pourquoi avons-nous besoin de React.js ?

Bien que JavaScript puisse accomplir cette magie de manipulation du DOM (Document Object Model), il peut devenir difficile à gérer et à maintenir dans des applications complexes. L'utilisation directe de JavaScript pour la manipulation du DOM peut également entraîner un risque accru d'erreurs et de comportements imprévisibles.

Quelle est la différence avec JavaScript ? Pourquoi utiliser React.js ?

La principale différence réside dans l'approche de développement. En JavaScript traditionnel, on adopte une approche impérative, où l'on décrit les étapes nécessaires pour atteindre un état donné de l'interface utilisateur. En revanche, avec React.js, on adopte une approche déclarative, où l'on définit l'état désiré de l'interface utilisateur, et React se charge de mettre à jour l'interface en conséquence.

Cette approche déclarative rend le code plus lisible, plus prévisible et plus facile à maintenir, car elle permet aux développeurs de se concentrer sur ce que l'interface utilisateur devrait être à un moment donné, plutôt que sur les étapes spécifiques pour y parvenir.

exemples pris sur Internet pour illustrer la différence:

- exemple JS : https://codesandbox.io/p/sandbox/vanilla-js-demo-6049kj?file=/index.html
- exemple react : https://codesandbox.io/p/sandbox/react-vs-vanilla-demo-uc08fv?file=/src/index.js

installation

github

création d'un repo sur github idéalement on l'appellera hetic_react mais vous pouvez évidemment choisir un autre nom.

création d'un projet en local à l'aide de vite

```
npm create vite@latest
cd hetic_react
npm install
npm run dev
```

deploiement sur GitHub-pages

```
npm install --save-dev gh-pages
```

Ajouter dans la partie "scripts" du fichier "package.json" les 2 lignes suivantes:

```
"predeploy": "npm run build",
"deploy": "gh-pages -d dist"
```

Modifier le fichier "vite.config.js" pour rajouter une ligne de configuration.

```
export default defineConfig({
  base: "/hetic_react/",
  plugins: [react()],
})
```

associez votre repo git comme remote:

```
git init
git remote add origin git@github.com:< user >/< project >.git
```

Vous pouvez maintenant deployer votre application React sur votre site github pages

```
npm run deploy
```

Votre site est maintenant disponible pour le monde entier!!! https://< user >.github.io/< project >/