Styliser une application React

CSS classique

CSS et applications React

Ajouter du style à une application React est faisable de différentes manières :

- Style inline
- Fichiers CSS "classiques"
- Utilisation d’un préprocesseur CSS (SASS par exemple)
- Utilisation de librairies conçues spécifiquement pour React

Style inline

Il est possible, mais non recommandé, d'ajouter du style inline aux éléments HTML.

- Le style est défini par un objet JavaScript,
- Les propriétés sont au format camel case (et non snake case)
- Les valeurs sont à définir via des strings

```
<button style={{
    border: '2px solid red',
    fontSize: '20px',
    backgroundColor: 'blue',
    color: '#FFF'
}}>
  A button
</button>
```

Fichiers CSS

Il est également possible de créer des fichiers CSS "classiques" pouvant être ensuite importés dans les composants React.

Il peut alors être nécessaire d'utiliser les propriétés `id` et `className` (équivalent de la propriété `class` en HTML).

```tsx
// components/Button/index.tsx
import React from 'react';
import './style.css';

const Button: React.FC = () =>
  <button className="my-button">
    A button
  </button>;
```

```css
/* components/Button/style.css */
.my-button {
    border: 2px solid red;
    font-size: 20px;
    background-color: blue;
    color: #FFF;
}
```

Fichiers CSS

Problème :

- Le style est global
- Attention à bien utiliser les classes et ID et à ne pas réutiliser les mêmes ailleurs dans l'application (sauf si intentionnel)
- Le style n'est pas "dynamique"
    - Impossible de passer des variables JavaScript au CSS
    - … enfin si, mais pas facilement
      (voir : https://www.w3schools.com/css/css3_variables_javascript.asp)

Styled Components

Présentation

Styled Components (https://styled-components.com) :

- Est une librairie permettant de styliser facilement des composants React
- Permet de définir le CSS en JavaScript
- Permet donc de rendre le style dynamique très facilement !
- Réutilise le principe de composants React
- Assigne à chaque composant stylisé une classe unique
    - Pas de risque de partage accidentel de classes CSS !

Installation

Installation des dépendances :

```
$ npm install styled-components

$ npm install --save-dev @types/styled-components
```

Pour Intellij Idea (WebStorm, PHPStorm) installez le plugin "**Styled Components & Styled JSX**" : https://plugins.jetbrains.com/plugin/9997-styled-components--styled-jsx

Pour VS Code, installez le plugin "**vscode-styled-components**" : https://marketplace.visualstudio.com/items?itemName=styled-components.vscode-styled-components

Exemple basique 1/2

Les styles doivent être définis dans des fichiers `*.tsx` :

```
// components/Button/styles.tsx
import styled from 'styled-components';

export const MyButton = styled.button`
  border: 2px solid red;
  font-size: 20px;
  background-color: blue;
  color: #FFF
`;
```

Exemple basique 2/2

Utilisation du style :

```
// components/Button/index.tsx
import React from 'react';
import { MyButton } from './styles';

const Button: React.FC = () =>
  <MyButton>
    A button
  </MyButton>;
```

Passage de propriétés 1/2

Tout composant Styled Components peut accepter des props pour y réagir dynamiquement :

```
interface MyButtonProps {
  active: boolean
}

export const MyButton = styled.button<MyButtonProps>`
  border: 2px solid red;
  font-size: 20px;
  color: #FFF

  background-color: ${(props) =>
    props.active ?
      'blue' :
      'gray'
  };
`;
```

Passage de propriétés 2/2

Passage de props à un composant Styled Components :

```
const Button: React.FC = () =>
  <MyButton active={false}>
    A button
  </MyButton>;
```

Styliser des composants complexes 1/2

Il est possible de styliser n'importe quel composant complexe, à condition que celui-ci assigne le **`className`** qui lui est passé par Styled Components à un élément HTML :

```
interface Props {
  className: string
}

const Button: React.FC<Props> = ({ className }) =>
  <button className={className}>
    A button
  </button>;
```

Styliser des composants complexes 2/2

Styliser un composant existant :

```
import styled from 'styled-components';
import Button from './index';

interface MyButtonProps {
  active: boolean
}

export const MyButton = styled(Button)<MyButtonProps>`
  border: 2px solid red;
  font-size: 20px;
  color: #FFF

  background-color: ${(props) =>
    props.active ?
      'blue' :
      'gray'
  };
`;
```

Style global

```
import { createGlobalStyle } from 'styled-components'

const GlobalStyle = createGlobalStyle`
  body {
    color: ${props => (props.whiteColor ? 'white' : 'black')};
  }
`;

// A la racine du projet (par exemple) :
<GlobalStyle whiteColor />
```

Exercice

Exercice

Donnez du style à votre application Game of Thrones Characters !