Le style et les classes

Ibrahim ALAME

14/02/2023

1 Introduction et portée des styles

1.1 Comportement par défaut

Par défaut, tout le CSS est global car il est réuni dans le bundle par Vite. Si vous définissez des classes à l'intérieur des balises `<style>`, ces classes s'appliqueront donc à tous vos composants.

1.2 L'attribut scoped

L'attribut scoped permet d'encapsuler le CSS d'un composant monofichier (SFC) en ajoutant, grâce à PostCSS un attribut unique (par exemple data-v-29e5g83) à l'élément en compilant par exempl .ma-classe en .ma-classe[data-v-29e5g83].

Cette encapsulation est effectuée automatiquement par Vite. Vous n'avez rien à faire d'autre que d'ajouter l'attribut scoped. Pour utiliser scoped, il suffit de l'ajouter sur la balise style de votre composant monofichier :

```
<style scoped>
  .ma-classe {
  }
</style>
```

2 La liaison de classes

2.1 La liaison de classes

Nous avons vu qu'il était possible de lier une ou plusieurs classes à l'attribut class en utilisant la directive v-bind. Pour permettre une utilisation très flexible, Vue.js permet plus de syntaxes lorsque nous utilisons v-bind avec les classes ou les styles.

Bien sûr la liaison n'a de sens que si les classes sont réactives suivant la valeur de propriétés. Sinon il suffit d'écrire les classes en dur côté template. Nous allons récapituler toutes les syntaxes possibles.

2.1.1 Liaison avec des objets

La première syntaxe permet d'utiliser un objet :

```
<div :class="{ actif: estActif }"></div>
```

La classe actif sera ajoutée si la propriété `estActif` vaut `true` :

```
const estActif = ref(true)
```

Si la propriété réactive passe à `false`, la classe sera automatiquement retirée.

2.1.2 Combinaison de l'attribut class et d'une liaison :class

Vous pouvez combiner sans problème l'attribut class avec une liaison avec la directive v-bind en utilisant :class :

```
<div
  class="classe1"
  :class="{ active: isActive, error: hasError }"
></div>
```

Ici nous aurons quoiqu'il arrive la `classe1`.

- La classe active sera ajoutée si la propriété isActive vaut `true`.
- La classe error sera ajoutée si la propriété hasError vaut `true`.

Dans tous les cas, Vue.js créera la div sur le DOM avec un seul attribut classe, par exemple :

```
<div class="classe1 isActive hasError"></div>
```

2.1.3 Utilisation d'un objet réactif

Vous pouvez utiliser un objet réactif côté script pour la liaison avec l'attribut class :

```
const mesClasses = reactive({
  active: true,
  error: false,
  classeX: false
});
```

Il suffit ensuite de le passer à v-bind :

```
<div :class="mesClasses"></div>
```

2.1.4 Utilisation d'une propriété calculée

Vous pouvez même utiliser une propriété calculée :

```
const mesClasses = computed(() => ({
  active: isActive.value && !error.value,
  error: error.value && error.value.type === 'fatal'
}))
```

2.1.5 Utilisation d'un tableau de classes

Vous pouvez utiliser un tableau de classes :

```
<div :class="[classe1, classe2]"></div>
```

3 La liaison de styles

3.1 Liaison de styles

Passons maintenant aux différentes syntaxes possibles pour les liaisons de style.

3.1.1 Liaison avec des objets

La première syntaxe permet d'utiliser un objet :

```
<div :style="{ color: activeColor, fontSize: fontSize + 'px' }"></div>
```

Avec côté script :

```
const activeColor = ref('red')
const fontSize = ref(30)
```

3.1.2 Utilisation des propriétés CSS natives

Vous pouvez également utiliser directement les noms des propriétés CSS en utilisant des guillemets simples :

```
<div :style="{ 'font-size': fontSize + 'px' }"></div>
```

3.1.3 Utilisation d'un objet réactif

Vous pouvez utiliser un objet réactif côté script pour la liaison avec l'attribut style :

```
const mesStyles = reactive({
  color: 'red',
  fontSize: '13px'
})
```

Et côté template :

```
<div :style="mesStyles"></div>
```

3.1.4 Utilisation d'un tableau

Vous pouvez enfin utiliser un tableau :

```
<div :style="[stylesDeBase, stylesParticuliers]"></div>
```

Ici les styles contenus dans l'objet stylesDeBase seront appliqués. Ensuite les styles contenus dans l'objet stylesParticuliers seront fusionnés. Si certaines propriétés sont dans les deux objets, les valeurs du dernier objet écraseront les valeurs précédentes.

Autrement dit, si par exemple il y a `color:'red'` dans stylesDeBase et `color:'blue'` dans stylesParticuliers. La valeur de la propriété `color` sera `blue`.

4 Utilisation de Sass

L'utilisation de Sass avec Vite et Vue.js est extrêmement simple. Il suffit d'installer sass :

```
npm add -D sass
```

Et dans les composants SFC de préciser, comme pour TypeScript côté template, que les styles sont écrits en Sass :

```
<template>
  <h1><span>Bonjour</span> le monde !</h1>
</template>

<script setup lang="ts"></script>

<style scoped lang="scss">
h1 {
  color: red;
  span {
    color: blue;
  }
```

```
}
</style>
```