Vue.js : Directives

Achref El Mouelhi

Docteur de l'université d'Aix-Marseille
Chercheur en programmation par contrainte (IA)
Ingénieur en génie logiciel

elmouelhi.achref@gmail.com

Plan

Plan

Plan

Vue.js

Deux types de directive **Vue.js**

- Directives structurelles : ayant comme but de modifier le **DOM** (`v-if`, `v-for`...)
- Directives attributes : ayant comme but de modifier de manière dynamique la valeur d'un élément (`v-bind`, `v-model`...)

Vue.js

Deux types de directive **Vue.js**

- Directives structurelles : ayant comme but de modifier le **DOM** (`v-if`, `v-for`...)
- Directives attributes : ayant comme but de modifier de manière dynamique la valeur d'un élément (`v-bind`, `v-model`...)

Remarque

Il est possible de créer sa propre directive.

Vue.js

Un composant peut être déclaré

- `local` : relatif à un composant, utilisable uniquement dans ce composant
- `global` : utilisable dans tous les composants de l'application

Vue.js

Démarche

- Créer un objet **JavaScript**
- Utiliser les fonctions **hooks** pour définir le comportement de la nouvelle directive
- Chaque fonction **hooks** un objet **JavaScript** de type **HTMLElement**

Vue.js

Exemple

- Créer une directive `v-mouvement` qui change la couleur d'un élément **HTML** survolé
- Remettre la couleur initiale lorsque l'élément n'est plus survolé

Vue.js

Dans `script` de `Produit.vue`, commençons par ajouter une partie `directive`

```
<script>

import PrixComponent from './Prix.vue'

export default {
    name: 'ProduitComponent',
    props: ['produit'],
    components: {
        PrixComponent
    },
    directives: {
    }
}

</script>
```

Vue.js

Déclarons ensuite notre directive `mouvement`

```
<script>

import PrixComponent from './Prix.vue'

export default {
    name: 'ProduitComponent',
    props: ['produit'],
    components: {
        PrixComponent
    },
    directives: {
        mouvement:  {
        }
    }
}

</script>
```

Définissons le comportement de cette directive à son chargement : afficher une couleur de fond rouge

```
<script>

import PrixComponent from './Prix.vue'

export default {
    name: 'ProduitComponent',
    props: ['produit'],
    components: {
        PrixComponent
    },
    directives: {
        mouvement:  {
            mounted: (el) => el.style.backgroundColor = 'red'
        }
    }
}

</script>
```

Vue.js

Pour utiliser cette directive dans `template`, on la préfixe par `v-`

```
<template>
    <ul>
        <li v-mouvement> {{ produit.nom }} </li>
        <li>Quantité en stock : {{ produit.quantite }} </li>
        <li>
            <PrixComponent :prix="produit.prix" />
        </li>
    </ul>
</template>
```

Le plus souvent, nous souhaitons que la directive concernerait les hooks `mounted` et `updated`, dans ce cas le code pourrait être simplifié

```
<script>

import PrixComponent from './Prix.vue'

export default {
    name: 'ProduitComponent',
    props: ['produit'],
    components: {
        PrixComponent
    },
    directives: {
        mouvement: (el) => el.style.backgroundColor = 'red'
        }
    }
}

</script>
```

Vue.js

Et si on voulait personnaliser la couleur à chaque utilisation

```
<template>
    <ul>
        <li v-mouvement.color="'skyblue'"> {{ produit.nom }} </
           li>
        <li v-mouvement="'pink'">Quantité en stock : {{ produit
           .quantite }} </li>
        <li>
            <PrixComponent :prix="produit.prix" />
        </li>
    </ul>
</template>
```

Vue.js

Il faudra utiliser un deuxième paramètre `binding`

```
<script>

import PrixComponent from './Prix.vue'

export default {
    name: 'ProduitComponent',
    props: ['produit'],
    components: {
        PrixComponent
    },
    directives: {
        mouvement: (el, binding) => el.style.backgroundColor = binding.value
    }
}

</script>
```

Vue.js

Il est aussi possible d'envoyer un objet avec plusieurs valeurs

```
<template>
    <ul>
        <li v-mouvement.color="{color: 'white', bgColor: 'skyblue'}">
            {{ produit.nom }}
        </li>
        <li v-mouvement="{color: 'white', bgColor: 'pink'}">
            Quantité en stock : {{ produit.quantite }}
        </li>
        <li>
            <PrixComponent :prix="produit.prix" />
        </li>
    </ul>
</template>
```

Vue.js

Pour les récupérer

```
<script>

import PrixComponent from './Prix.vue'

export default {
    name: 'ProduitComponent',
    props: ['produit'],
    components: {
        PrixComponent
    },
    directives: {
        mouvement: (el, binding) => {
            el.style.backgroundColor = binding.value.bgColor,
            el.style.color = binding.value.color
        }
    }
}

</script>
```

Vue.js

Hypothèse

- Si on voulait modifier la couleur de fond d'un élément s'il est survolé
- Et remettre sa couleur à la sortie du curseur
- Il faut attacher des évènements aux changements de couleur

Solution

```
<script>

import PrixComponent from './Prix.vue'

export default {
    name: 'ProduitComponent',
    props: ['produit'],
    components: {
        PrixComponent
    },
    directives: {
        mouvement: (el, binding) => {
            el.addEventListener('mouseenter', () => {
                el.style.backgroundColor = binding.value.bgColor
                el.style.color = binding.value.color
            })
            el.addEventListener('mouseleave', () => {
                el.removeAttribute('style')
            })
        }
    }
}
</script>
```

Même exemple avec Composition API : le nom de la directive doit commencer par la lettre `v` et doit être écrit en `Camel case`

```
<script>
import PrixComponent from './Prix.vue'
export default { name: 'ProduitComponent' }
</script>
<script setup>
import { defineProps } from 'vue';

defineProps({
    produit: Object
})

const vMouvement = (el, binding) => {
    el.addEventListener('mouseenter', () => {
        el.style.backgroundColor = binding.value.bgColor
        el.style.color = binding.value.color
    })
    el.addEventListener('mouseleave', () => {
        el.removeAttribute('style')
    })
}
</script>
```

Vue.js

Remarque

La directive `mouvement` est utilisable seulement dans le composant où elle a été déclarée.

Vue.js

Remarque

La directive `mouvement` est utilisable seulement dans le composant où elle a été déclarée.

Question

Comment l'utiliser dans plusieurs composants ?

Vue.js

Remarque

La directive `mouvement` est utilisable seulement dans le composant où elle a été déclarée.

Question

Comment l'utiliser dans plusieurs composants ?

Solution

Déclarer la directive dans `main.js`

Vue.js

Déplaçons le code de la directive `mouvement` **dans** `main.js` **et utilisons** `app.use()`

```
import { createApp } from 'vue'
import App from './App.vue'
import router from './router'
import axios from 'axios'
import VueAxios from 'vue-axios'

const app = createApp(App);
app.config.globalProperties.baseUrl = 'http://localhost:5555';
app
    .use(router)
    .use(VueAxios, axios)

app.provide('axios', app.config.globalProperties.axios)
app.provide('baseUrl', app.config.globalProperties.baseUrl)

app.directive('mouvement',  (el, binding) => {
    el.addEventListener('mouseenter', () => {
        el.style.backgroundColor = binding.value.bgColor
        el.style.color = binding.value.color
    })
    el.addEventListener('mouseleave', () => {
        el.removeAttribute('style')
    })
})
app.mount('#app')
```

Vue.js

Supprimons sa déclaration dans `Produit.vue`

```
<script>
import PrixComponent from './Prix.vue'
export default { name: 'ProduitComponent' }
</script>
<script setup>
import { defineProps } from 'vue';

defineProps({
    produit: Object
})
</script>

<template>
    <ul>
        <li v-mouvement.color="{color: 'white', bgColor: 'skyblue'}"> {{ produit.nom }}
            </li>
        <li v-mouvement="{color: 'white', bgColor: 'pink'}">Quantité en stock : {{
          produit.quantite }} </li>
        <li>
            <PrixComponent :prix="produit.prix" />
        </li>
    </ul>
</template>
```

Vue.js

Supprimons sa déclaration dans `Produit.vue`

```
<script>
import PrixComponent from './Prix.vue'
export default { name: 'ProduitComponent' }
</script>
<script setup>
import { defineProps } from 'vue';

defineProps({
    produit: Object
})
</script>

<template>
    <ul>
        <li v-mouvement.color="{color: 'white', bgColor: 'skyblue'}"> {{ produit.nom }}
            </li>
        <li v-mouvement="{color: 'white', bgColor: 'pink'}">Quantité en stock : {{
          produit.quantite }} </li>
        <li>
            <PrixComponent :prix="produit.prix" />
        </li>
    </ul>
</template>
```

Testez et vérifiez que ça marche.

Vue.js

Exercice

- Créez un composant `PeintreView.vue`
- Dans `script`, déclarez un attribut `couleurs` avec les valeurs données ci-dessous.
- Créez un composant fils nommé `Peinture.vue`.
- Le composant `peinture` a un attribut `value` recevant sa valeur du tableau `couleurs` défini dans `peintre.component.ts`.
- Le nombre de composants `peinture` = taille du tableau `couleurs`.
- Chaque composant `peinture` affiche le nom de la couleur qu'il a reçu de son parent.
- Créez une directive `pinceau` qui modifie la couleur de fond des composants `peinture` survolés. La couleur de fond est la valeur de l'attribut `value` du composant `peinture` survolé.

```
couleurs = ['red', 'skyblue', 'gray', 'green', 'yellow', 'teal']
```