Notification

Affiche une notification globale dans un coin de la page.

Usage

:::demo Element enregistre la méthode `$notify` qui reçoit un objet en paramètre. Dans le cas le plus simple, vous pouvez simplement configurer les champs `title` et`message`. Par défaut, la notification se ferme automatiquement après 4500ms, mais vous pouvez régler une autre durée avec `duration`. Si la durée est mise à `0`, la notification ne se fermera pas. `duration` prends donc un `Number` en millisecondes.

```
<template>
  <el-button
    plain
    @click="open1">
    Se ferme automatiquement
  </el-button>
  <el-button
    plain
    @click="open2">
    Ne se ferme pas automatiquement
    </el-button>
</template>

<script>
  export default {
    methods: {
      open1() {
        const h = this.$createElement;

        this.$notify({
          title: 'Titre',
          message: h('i', { style: 'color: teal' }, 'Ceci est un rappel')
        });
      },

      open2() {
        this.$notify({
          title: 'Prompt',
          message: 'Ceci est un message qui ne se ferme pas',
          duration: 0
        });
      }
    }
  }
```

```
</script>
```

:::

Types

Nous fournissons quatre types: succès, avertissement, information et erreur.

:::demo Element fournit quatre types de notifications: `success`, `warning`, `info` et `error`. Il sont choisis grâce au champs `type`, et n'importe quelle autre valeur sera ignorée. Il existe des méthodes enregistrées pour chaque type, comme dans `open3` et `open4`, qui ne nécessitent donc pas le champs `type`.

```
<template>
  <el-button
    plain
    @click="open1">
    Success
  </el-button>
  <el-button
    plain
    @click="open2">
    Warning
  </el-button>
  <el-button
    plain
    @click="open3">
    Info
  </el-button>
  <el-button
    plain
    @click="open4">
    Error
  </el-button>
</template>

<script>
  export default {
    methods: {
      open1() {
        this.$notify({
          title: 'Success',
          message: 'Ceci est un message de succès',
          type: 'success'
        });
      },
```

```
      open2() {
        this.$notify({
          title: 'Warning',
          message: 'Ceci est un avertissement',
          type: 'warning'
        });
      },

      open3() {
        this.$notify.info({
          title: 'Info',
          message: 'Ceci est une information'
        });
      },

      open4() {
        this.$notify.error({
          title: 'Error',
          message: 'Ceci est une erreur'
        });
      }
    }
  }
</script>
```

:::

Position personnalisée

La notification peut apparaître dans le coin de votre choix.

:::demo L'attribut `position` définit le coin d'apparition de la notification. Cela peut être `top-right`, `top-left`, `bottom-right` ou `bottom-left`. Le défaut est `top-right`.

```
<template>
  <el-button
    plain
    @click="open1">
    Top Right
  </el-button>
  <el-button
    plain
    @click="open2">
    Bottom Right
  </el-button>
  <el-button
```

```
    plain
    @click="open3">
    Bottom Left
  </el-button>
  <el-button
    plain
    @click="open4">
    Top Left
  </el-button>
</template>

<script>
  export default {
    methods: {
      open1() {
        this.$notify({
          title: 'Custom Position',
          message: 'Je suis dans le coin supérieur droit'
        });
      },

      open2() {
        this.$notify({
          title: 'Custom Position',
          message: 'Je suis dans le coin inférieur droit',
          position: 'bottom-right'
        });
      },

      open3() {
        this.$notify({
          title: 'Custom Position',
          message: 'Je suis dans le coin inférieur gauche',
          position: 'bottom-left'
        });
      },

      open4() {
        this.$notify({
          title: 'Custom Position',
          message: 'Je suis dans le coin supérieur gauche',
          position: 'top-left'
        });
      }
    }
  }
```

```
</script>
```

:::

Avec décalage

Vous pouvez décaler l'emplacement de la notification par rapport au bord de la page.

:::demo Réglez `offset` pour choisir le décalage de la notification. Notez que chaque notification apparaissant au même moment devrait avoir le même décalage.

```
<template>
  <el-button
    plain
    @click="open">
    Notification avec décalage
  </el-button>
</template>

<script>
  export default {
    methods: {
      open() {
        this.$notify.success({
          title: 'Success',
          message: 'Ceci est un message de succès',
          offset: 100
        });
      }
    }
  }
</script>
```

:::

Utiliser du HTML

L'attribut `message` supporte le HTML.

:::demo Mettez `dangerouslyUseHTMLString` à true et `message` sera traité comme du HTML.

```
<template>
  <el-button
    plain
    @click="open">
    Utiliser du HTML
```

```
  </el-button>
</template>

<script>
  export default {
    methods: {
      open() {
        this.$notify({
          title: 'HTML String',
          dangerouslyUseHTMLString: true,
          message: '<strong>Ceci est du <i>HTML</i></strong>'
        });
      }
    }
  }
</script>
```

:::

Bien que la propriété `message` supporte le HTML, générer du contenu HTML dynamiquement peut être très dangereux, car cela permet des attaques XSS. Donc lorsque `dangerouslyUseHTMLString` est présent, soyez certain de sécuriser le contenu de `message`, et n'assignez **jamais** à `message` du contenu fournit par l'utilisateur.

Cacher le bouton de fermeture

Il est possible de cacher le bouton de fermeture.

:::demo Mettez `showClose` à `false` Pour que la notification ne puisse pas être fermée par l'utilisateur.

```
<template>
  <el-button
    plain
    @click="open">
    Bouton de fermeture caché
    </el-button>
</template>

<script>
  export default {
    methods: {
      open() {
        this.$notify.success({
          title: 'Info',
          message: 'Ceci est un message sans bouton de fermeture',
          showClose: false
```

```
        });
      }
    }
  }
</script>
```

:::

Méthode globale

Element ajoute la méthode `$notify` à Vue.prototype. Vous pouvez donc appeler `Notification` dans l'instance de Vue comme nous avons fait dans cette page.

Import à la demande

Importez `Notification`:

```
import { Notification } from 'element-ui';
```

Dans ce cas vous devrez appeler `Notification(options)`. Il existe aussi des méthodes pour chaque type, e.g. `Notification.success(options)`. Vous pouvez appeler `Notification.closeAll()` pour fermer manuellement toutes les instances.

Options

Attribut	Description	Type	Valeurs acceptées	Défaut
title	Titre de la notification.	string	—	—
message	Message de la notification.	string/Vue.VNode	—	—
dangerouslyUseHTMLString	Si `message` doit être traité comme du HTML.	boolean	—	false
type	Type de notification.	string	success/warning/info/error	—
iconClass	Classe d'icône. Écrasé par `type`.	string	—	—
customClass	Nom de classe pour la notification.	string	—	—

Attribut	Description	Type	Valeurs acceptées	Défaut
duration	Durée avant fermeture. Infinie si mise à 0.	number	—	4500
position	Position de la notification.	string	top-right/top-left/bottom-right/bottom-left	top-right
showClose	Si le bouton de fermeture doit être affiché.	boolean	—	true
onClose	Callback de fermeture.	function	—	—
onClick	Callback quand la notification est cliquée.	function	—	—
offset	Décalage par rapport au bord de la page. Toutes les notifications arrivant au même moment devraient avoir le même décalage.	number	—	0

Méthodes

`Notification` et `this.$notify` retourne l'instance actuelle de Notification. Pour fermer chaque instance manuellement, appelez la méthode `close`.

Méthode	Description
close	Ferme la notification.