Message

Utilisé pour avoir un retour après une action particulière. La différence avec Notification est que ce dernier est surtout utilisé pour afficher des notifications système passives.

Usage

S'affiche en haut de la page et disparaît après trois secondes.

:::demo L'utilisation de Message est très similaire à Notification, la plupart des options ne sont donc pas expliquées ici. Référez-vous à la table en fin de page et celle de Notification pour en savoir plus. Element affecte la méthode `$message` pour appeler Message. Il peut prendre en paramètre un string ou un VNode, qui sera affiché en tant que body principal.

```
<template>
  <el-button :plain="true" @click="open">Afficher le message</el-button>
  <el-button :plain="true" @click="openVn">VNode</el-button>
</template>

<script>
  export default {
    methods: {
      open() {
        this.$message('Ceci est un message.');
      },

      openVn() {
        const h = this.$createElement;
        this.$message({
          message: h('p', null, [
            h('span', null, 'Message peut être '),
            h('i', { style: 'color: teal' }, 'VNode')
          ])
        });
      }
    }
  }
</script>
```

:::

Types

Utilisé pour montrer un retour d'activités Success, Warning, Message ou Error.

:::demo Lorsque vous avez besoin de plus de personnalisation, Message peut aussi accepter un objet en paramètre. Par exemple, le paramètre `type` définit différents types, son défaut étant `info`. Dans ce cas le body est passé comme valeur de `message`. De plus, il existe des méthodes pour chaque type, afin que vous puissiez vous passer de la propriété `type` comme dans `open4`.

```
<template>
  <el-button :plain="true" @click="open2">success</el-button>
```

```
  <el-button :plain="true" @click="open3">warning</el-button>
  <el-button :plain="true" @click="open1">message</el-button>
  <el-button :plain="true" @click="open4">error</el-button>
</template>

<script>
  export default {
    methods: {
      open1() {
        this.$message('Ceci est un message.');
      },
      open2() {
        this.$message({
          message: 'Félicitations, ceci est un message de succès.',
          type: 'success'
        });
      },

      open3() {
        this.$message({
          message: 'Attention, ceci est un avertissement.',
          type: 'warning'
        });
      },

      open4() {
        this.$message.error('Ouups, ceci est une erreur.');
      }
    }
  }
</script>
```

:::

Fermeture

Un bouton de fermeture peut être ajouté.

:::demo Un Message ne peut être fermé par défaut. Utiliséez `showClose` si vous avez besoin de pouvoir le fermer. De plus, tout comme Notification, Message possède une `duration` réglable. La durée par défaut est de 3000 ms, et infinie si à `0` .

```
<template>
  <el-button :plain="true" @click="open1">message</el-button>
  <el-button :plain="true" @click="open2">success</el-button>
  <el-button :plain="true" @click="open3">warning</el-button>
  <el-button :plain="true" @click="open4">error</el-button>
</template>

<script>
  export default {
    methods: {
```

```
      open1() {
        this.$message({
          showClose: true,
          message: 'Ceci est un message.'
        });
      },

      open2() {
        this.$message({
          showClose: true,
          message: 'Félicitations, ceci est un message de succès.',
          type: 'success'
        });
      },

      open3() {
        this.$message({
          showClose: true,
          message: 'Attention, ceci est un avertissement.',
          type: 'warning'
        });
      },

      open4() {
        this.$message({
          showClose: true,
          message: 'Ouups, ceci est une erreur.',
          type: 'error'
        });
      }
    }
  }
</script>
```

:::

Texte centré

Utilisez l'attribut `center` pour centrer le texte.

:::demo

```
<template>
  <el-button :plain="true" @click="openCenter">Texte centré</el-button>
</template>

<script>
  export default {
    methods: {
      openCenter() {
        this.$message({
          message: 'Texte centré',
```

```
          center: true
        });
      }
    }
  }
</script>
```

:::

Utiliser du HTML

`message` supporte le HTML.

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

```
<template>
  <el-button :plain="true" @click="openHTML">Utiliser du HTML</el-button>
</template>

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

:::

:::warning 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.
:::

Méthode globale

Element ajoute une méthode `$message` à Vue.prototype. Vous pouvez donc appeler `Message` dans l'instance de Vue comme dans cette page.

Import à la demande

Importez `Message` :

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

Dans ce cas il faudra appeler `Message(options)`. Les méthodes des différents types sont aussi ajoutées, e.g. `Message.success(options)`. Vous pouvez appeler `Message.closeAll()` pour fermer manuellement toutes les instances.

Options

Attribut	Description	Type	Valeurs acceptées	Défaut
message	Texte du message.	string / VNode	—	—
type	Type du message.	string	success/warning/info/error	info
iconClass	Classe de l'icône, écrase `type`.	string	—	—
dangerouslyUseHTMLString	Si `message` doit être traité comme du HTML.	boolean	—	false
customClass	Nom de classe pour Message.	string	—	—
duration	La durée d'affichage, en millisecondes. Si 0, la durée est infinie.	number	—	3000
showClose	Si un bouton de fermeture doit être affiché.	boolean	—	false
center	Si le texte doit être centré.	boolean	—	false
onClose	Callback de fermeture avec en paramètre l'instance de Message.	function	—	—
offset	set the distance to the top of viewport	number	—	20

Méthodes

`Message` et `this.$message` retourne l'instance actuelle. Pour fermer manuellement cette instance, appelez sa méthode `close`.

Méthode	Description
close	Ferme l'instance de Message.