Notification

Displays a global notification message at a corner of the page.

Basic usage

:::demo Element has registered the `$notify` method and it receives an object as its parameter. In the simplest case, you can set the `title` field and the`message` field for the title and body of the notification. By default, the notification automatically closes after 4500ms, but by setting `duration` you can control its duration. Specifically, if set to `0`, it will not close automatically. Note that `duration` receives a `Number` in milliseconds.

```
<template>
  <el-button
    plain
    @click="open1">
    Closes automatically
  </el-button>
  <el-button
    plain
    @click="open2">
    Won't close automatically
    </el-button>
</template>

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

        this.$notify({
          title: 'Title',
          message: h('i', { style: 'color: teal' }, 'This is a reminder')
        });
      },

      open2() {
        this.$notify({
          title: 'Prompt',
          message: 'This is a message that does not automatically close',
          duration: 0
        });
      }
    }
  }
```

```
</script>
```

:::

With types

We provide four types: success, warning, info and error.

:::demo Element provides four notification types: `success`, `warning`, `info` and `error`. They are set by the `type` field, and other values will be ignored. We also registered methods for these types that can be invoked directly like `open3` and `open4` without passing a `type` field.

```
<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.$notify({
          title: 'Success',
          message: 'This is a success message',
          type: 'success'
        });
      },
      open2() {
        this.$notify({
          title: 'Warning',
          message: 'This is a warning message',
          type: 'warning'
        });
      },

      open3() {
        this.$notify.info({
          title: 'Info',
          message: 'This is an info message'
        });
      },

      open4() {
        this.$notify.error({
          title: 'Error',
          message: 'This is an error message'
        });
      }
    }
  }
</script>
```

:::

Custom position

Notification can emerge from any corner you like.

:::demo The `position` attribute defines which corner Notification slides in. It can be `top-right`, `top-left`, `bottom-right` or `bottom-left`. Defaults to `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: 'I\'m at the top right corner'
        });
      },

      open2() {
        this.$notify({
          title: 'Custom Position',
          message: 'I\'m at the bottom right corner',
          position: 'bottom-right'
        });
      },

      open3() {
        this.$notify({
          title: 'Custom Position',
          message: 'I\'m at the bottom left corner',
          position: 'bottom-left'
        });
      },

      open4() {
        this.$notify({
          title: 'Custom Position',
          message: 'I\'m at the top left corner',
          position: 'top-left'
        });
      }
    }
  }
```

```
</script>
```

:::

With offset

Customize Notification's offset from the edge of the screen.

:::demo Set the `offset` attribute to customize Notification's offset from the edge of the screen. Note that every Notification instance of the same moment should have the same offset.

```
<template>
  <el-button
    plain
    @click="open">
    Notification with offset
  </el-button>
</template>

<script>
  export default {
    methods: {
      open() {
        this.$notify.success({
          title: 'Success',
          message: 'This is a success message',
          offset: 100
        });
      }
    }
  }
</script>
```

:::

Use HTML string

`message` supports HTML string.

:::demo Set `dangerouslyUseHTMLString` to true and `message` will be treated as an HTML string.

```
<template>
  <el-button
    plain
    @click="open">
    Use HTML String
  </el-button>
```

```
</template>

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

:::

Although `message` property supports HTML strings, dynamically rendering arbitrary HTML on your website can be very dangerous because it can easily lead to XSS attacks. So when `dangerouslyUseHTMLString` is on, please make sure the content of `message` is trusted, and **never** assign `message` to user-provided content.

Hide close button

It is possible to hide the close button

:::demo Set the `showClose` attribute to `false` so the notification cannot be closed by the user.

```
<template>
  <el-button
    plain
    @click="open">
    Hide close button
    </el-button>
</template>

<script>
  export default {
    methods: {
      open() {
        this.$notify.success({
          title: 'Info',
          message: 'This is a message without close button',
          showClose: false
        });
      }
    }
  }
</script>
```

:::

Global method

Element has added a global method `$notify` for Vue.prototype. So in a vue instance you can call `Notification` like what we did in this page.

Local import

Import `Notification`:

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

In this case you should call `Notification(options)`. We have also registered methods for different types, e.g. `Notification.success(options)`. You can call `Notification.closeAll()` to manually close all the instances.

Options

Attribute	Description	Type	Accepted Values	Default
title	title	string	—	—
message	description text	string/Vue.VNode	—	—
dangerouslyUseHTMLString	whether `message` is treated as HTML string	boolean	—	false
type	notification type	string	success/warning/info/error	—
iconClass	custom icon's class. It will be overridden by `type`	string	—	—
customClass	custom class name for Notification	string	—	—

Attribute	Description	Type	Accepted Values	Default
duration	duration before close. It will not automatically close if set 0	number	—	4500
position	custom position	string	top-right/top-left/bottom-right/bottom-left	top-right
showClose	whether to show a close button	boolean	—	true
onClose	callback function when closed	function	—	—
onClick	callback function when notification clicked	function	—	—
offset	offset from the top edge of the screen. Every Notification instance of the same moment should have the same offset	number	—	0

Methods

`Notification` and `this.$notify` returns the current Notification instance. To manually close the instance, you can call `close` on it. | Method | Description | | —- | —- | | close | close the Notification |