Message

Used to show feedback after an activity. The difference with Notification is that the latter is often used to show a system level passive notification.

Basic usage

Displays at the top, and disappears after 3 seconds.

:::demo The setup of Message is very similar to notification, so parts of the options won't be explained in detail here. You can check the options table below combined with notification doc to understand it. Element has registered a `$message` method for invoking. Message can take a string or a VNode as parameter, and it will be shown as the main body.

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

<script>
  export default {
    methods: {
      open() {
        this.$message('This is a message.');
      },

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

:::

Types

Used to show the feedback of Success, Warning, Message and Error activities.

:::demo When you need more customizations, Message component can also take an object as parameter. For example, setting value of `type` can define different types, and its default is `info`. In such cases the main body is passed in as the

value of `message`. Also, we have registered methods for different types, so you can directly call it without passing a type like `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.$message('This is a message.');
      },
      open2() {
        this.$message({
          message: 'Congrats, this is a success message.',
          type: 'success'
        });
      },

      open3() {
        this.$message({
          message: 'Warning, this is a warning message.',
          type: 'warning'
        });
      },

      open4() {
        this.$message.error('Oops, this is a error message.');
      }
    }
  }
</script>
```

:::

Closable

A close button can be added.

:::demo A default Message cannot be closed manually. If you need a closable message, you can set `showClose` field. Besides, same as notification, message has a controllable `duration`. Default duration is 3000 ms, and it won't disappear when set to `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: 'This is a message.'
        });
      },

      open2() {
        this.$message({
          showClose: true,
          message: 'Congrats, this is a success message.',
          type: 'success'
        });
      },

      open3() {
        this.$message({
          showClose: true,
          message: 'Warning, this is a warning message.',
          type: 'warning'
        });
      },

      open4() {
        this.$message({
          showClose: true,
          message: 'Oops, this is a error message.',
          type: 'error'
        });
      }
    }
  }
</script>
```

:::

Centered text

Use the `center` attribute to center the text.

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

<script>
  export default {
    methods: {
      openCenter() {
        this.$message({
          message: 'Centered text',
          center: true
        });
      }
    }
  }
</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="true" @click="openHTML">Use HTML String</el-button>
</template>

<script>
  export default {
    methods: {
      openHTML() {
        this.$message({
          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.

Global method

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

Local import

Import `Message`:

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

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

Options

Attribute	Description	Type	Accepted Values	Default
message	message text	string / VNode	—	—
type	message type	string	success/warning/info/error	info
iconClass	custom icon's class, overrides `type`	string	—	—
dangerouslyUseHTMLString	whether `message` is treated as HTML string	boolean	—	false
customClass	custom class name for Message	string	—	—
duration	display duration, millisecond. If set to 0, it will not turn off automatically	number	—	3000

Attribute	Description	Type	Accepted Values	Default
showClose	whether to show a close button	boolean	—	false
center	whether to center the text	boolean	—	false
onClose	callback function when closed with the message instance as the parameter	function	—	—
offset	set the distance to the top of viewport	number	—	20

Methods

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