MessageBox 弹框

模拟系统的消息提示框而实现的一套模态对话框组件，用于消息提示、确认消息和提交内容。

:::tip 从场景上说，MessageBox 的作用是美化系统自带的 `alert`、`confirm` 和 `prompt`，因此适合展示较为简单的内容。如果需要弹出较为复杂的内容，请使用 Dialog。 :::

消息提示

当用户进行操作时会被触发，该对话框中断用户操作，直到用户确认知晓后才可关闭。

:::demo 调用 `$alert` 方法即可打开消息提示，它模拟了系统的 `alert`，无法通过按下 ESC 或点击框外关闭。此例中接收了两个参数，`message` 和 `title`。值得一提的是，窗口被关闭后，它默认会返回一个 `Promise` 对象便于进行后续操作的处理。若不确定浏览器是否支持 `Promise`，可自行引入第三方 polyfill 或像本例一样使用回调进行后续处理。

```
<template>
  <el-button type="text" @click="open">点击打开 Message Box</el-button>
</template>

<script>
  export default {
    methods: {
      open() {
        this.$alert('这是一段内容', '标题名称', {
          confirmButtonText: '确定',
          callback: action => {
            this.$message({
              type: 'info',
              message: `action: ${ action }`
            });
          }
        });
      }
    }
  }
</script>
```

:::

确认消息

提示用户确认其已经触发的动作，并询问是否进行此操作时会用到此对话框。

:::demo 调用 `$confirm` 方法即可打开消息提示，它模拟了系统的 `confirm`。Message Box 组件也拥有极高的定制性，我们可以传入 `options` 作为第三个参数，它是一个字面量对象。`type` 字段表明消息类型，可以为 `success`，`error`，`info` 和 `warning`，无效的设置将会被忽略。注意，第二个参数 `title` 必须定义为 `String` 类型，如果是 `Object`，会被理解为 `options`。在这里我们用了 Promise 来处理后续响应。

```
<template>
  <el-button type="text" @click="open">点击打开 Message Box</el-button>
</template>
```

```
<script>
  export default {
    methods: {
      open() {
        this.$confirm('此操作将永久删除该文件，是否继续?', '提示', {
          confirmButtonText: '确定',
          cancelButtonText: '取消',
          type: 'warning'
        }).then(() => {
          this.$message({
            type: 'success',
            message: '删除成功!'
          });
        }).catch(() => {
          this.$message({
            type: 'info',
            message: '已取消删除'
          });
        });
      }
    }
  }
</script>
```

:::

提交内容

当用户进行操作时会被触发，中断用户操作，提示用户进行输入的对话框。

:::demo 调用 `$prompt` 方法即可打开消息提示，它模拟了系统的 `prompt` 。可以用 `inputPattern` 字段自己规定匹配模式，或者用 `inputValidator` 规定校验函数，可以返回 `Boolean` 或 `String` ，返回 `false` 或字符串时均表示校验未通过，同时返回的字符串相当于定义了 `inputErrorMessage` 字段。此外，可以用 `inputPlaceholder` 字段来定义输入框的占位符。

```
<template>
  <el-button type="text" @click="open">点击打开 Message Box</el-button>
</template>

<script>
  export default {
    methods: {
      open() {
        this.$prompt('请输入邮箱', '提示', {
          confirmButtonText: '确定',
          cancelButtonText: '取消',
          inputPattern: /[\w!#$%&'*+/=?^_`{|}~-]+(?:\.[\w!#$%&'*+/=?^_`{|}~-]+)*@(?:
[\w](?:[\w-]*[\w])?\.)+[\w](?:[\w-]*[\w])?/,
          inputErrorMessage: '邮箱格式不正确'
        }).then(({ value }) => {
          this.$message({
```

```
            type: 'success',
            message: '你的邮箱是: ' + value
          });
        }).catch(() => {
          this.$message({
            type: 'info',
            message: '取消输入'
          });
        });
      }
    }
  }
</script>
```

:::

自定义

可自定义配置不同内容。

:::demo 以上三个方法都是对 `$msgbox` 方法的再包装。本例直接调用 `$msgbox` 方法，使用了 `showCancelButton` 字段，用于显示取消按钮。另外可使用 `cancelButtonClass` 为其添加自定义样式，使用 `cancelButtonText` 来自定义按钮文本（Confirm 按钮也具有相同的字段，在文末的字段说明中有完整的字段列表）。此例还使用了 `beforeClose` 属性，它的值是一个方法，会在 MessageBox 的实例关闭前被调用，同时暂停实例的关闭。它有三个参数：`action`、实例本身和 `done` 方法。使用它能够在关闭前对实例进行一些操作，比如为确定按钮添加 `loading` 状态等；此时若需要关闭实例，可以调用 `done` 方法（若在 `beforeClose` 中没有调用 `done`，则实例不会关闭）。

```
<template>
  <el-button type="text" @click="open">点击打开 Message Box</el-button>
</template>

<script>
  export default {
    methods: {
      open() {
        const h = this.$createElement;
        this.$msgbox({
          title: '消息',
          message: h('p', null, [
            h('span', null, '内容可以是 '),
            h('i', { style: 'color: teal' }, 'VNode')
          ]),
          showCancelButton: true,
          confirmButtonText: '确定',
          cancelButtonText: '取消',
          beforeClose: (action, instance, done) => {
            if (action === 'confirm') {
              instance.confirmButtonLoading = true;
              instance.confirmButtonText = '执行中...';
              setTimeout(() => {
```

```
                done();
                setTimeout(() => {
                  instance.confirmButtonLoading = false;
                }, 300);
              }, 3000);
            } else {
              done();
            }
          }
        }).then(action => {
          this.$message({
            type: 'info',
            message: 'action: ' + action
          });
        });
      }
    }
  }
</script>
```

:::

:::tip 弹出层的内容可以是 `VNode`，所以我们能把一些自定义组件传入其中。每次弹出层打开后，Vue 会对新老 `VNode` 节点进行比对，然后将根据比较结果进行最小单位地修改视图。这也许会造成弹出层内容区域的组件没有重新渲染，例如 #8931。当这类问题出现时，解决方案是给 `VNode` 加上一个不相同的 key，参考这里。 :::

使用 HTML 片段

`message` 属性支持传入 HTML 片段。

:::demo 将 `dangerouslyUseHTMLString` 属性设置为 true，`message` 就会被当作 HTML 片段处理。

```
<template>
  <el-button type="text" @click="open">点击打开 Message Box</el-button>
</template>

<script>
  export default {
    methods: {
      open() {
        this.$alert('<strong>这是 <i>HTML</i> 片段</strong>', 'HTML 片段', {
          dangerouslyUseHTMLString: true
        });
      }
    }
  }
</script>
```

:::

:::warning `message` 属性虽然支持传入 HTML 片段，但是在网站上动态渲染任意 HTML 是非常危险的，因为容易导致 XSS 攻击。因此在 `dangerouslyUseHTMLString` 打开的情况下，请确保 `message` 的内容是可信的，**永远不**

要将用户提交的内容赋值给 `message` 属性。 :::

区分取消与关闭

有些场景下，点击取消按钮与点击关闭按钮有着不同的含义。

:::demo 默认情况下，当用户触发取消（点击取消按钮）和触发关闭（点击关闭按钮或遮罩层、按下 ESC 键）时，Promise 的 reject 回调和 `callback` 回调的参数均为 'cancel'。如果将 `distinguishCancelAndClose` 属性设置为 true，则上述两种行为的参数分别为 'cancel' 和 'close'。

```
<template>
  <el-button type="text" @click="open">点击打开 Message Box</el-button>
</template>

<script>
  export default {
    methods: {
      open() {
        this.$confirm('检测到未保存的内容，是否在离开页面前保存修改？', '确认信息', {
          distinguishCancelAndClose: true,
          confirmButtonText: '保存',
          cancelButtonText: '放弃修改'
        })
          .then(() => {
            this.$message({
              type: 'info',
              message: '保存修改'
            });
          })
          .catch(action => {
            this.$message({
              type: 'info',
              message: action === 'cancel'
                ? '放弃保存并离开页面'
                : '停留在当前页面'
            })
          });
      }
    }
  }
</script>
```

:::

居中布局

内容支持居中布局

:::demo 将 `center` 设置为 `true` 即可开启居中布局

```
<template>
  <el-button type="text" @click="open">点击打开 Message Box</el-button>
```

```
</template>

<script>
  export default {
    methods: {
      open() {
        this.$confirm('此操作将永久删除该文件，是否继续?', '提示', {
          confirmButtonText: '确定',
          cancelButtonText: '取消',
          type: 'warning',
          center: true
        }).then(() => {
          this.$message({
            type: 'success',
            message: '删除成功!'
          });
        }).catch(() => {
          this.$message({
            type: 'info',
            message: '已取消删除'
          });
        });
      }
    }
  }
</script>
```

:::

全局方法

如果你完整引入了 Element，它会为 Vue.prototype 添加如下全局方法：$msgbox, $alert, $confirm 和 $prompt。因此在 Vue instance 中可以采用本页面中的方式调用 `MessageBox`。调用参数为：

- `$msgbox(options)`
- `$alert(message, title, options)` 或 `$alert(message, options)`
- `$confirm(message, title, options)` 或 `$confirm(message, options)`
- `$prompt(message, title, options)` 或 `$prompt(message, options)`

单独引用

如果单独引入 `MessageBox`：

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

那么对应于上述四个全局方法的调用方法依次为：MessageBox, MessageBox.alert, MessageBox.confirm 和 MessageBox.prompt，调用参数与全局方法相同。

Options

参数	说明	类型	可选值	默认值

title	MessageBox 标题	string	—	—
message	MessageBox 消息正文内容	string / VNode	—	—
dangerouslyUseHTMLString	是否将 message 属性作为 HTML 片段处理	boolean	—	false
type	消息类型，用于显示图标	string	success / info / warning / error	—
iconClass	自定义图标的类名，会覆盖 `type`	string	—	—
customClass	MessageBox 的自定义类名	string	—	—
callback	若不使用 Promise，可以使用此参数指定 MessageBox 关闭后的回调	function(action, instance)，action 的值为'confirm', 'cancel'或'close', instance 为 MessageBox 实例，可以通过它访问实例上的属性和方法	—	—
showClose	MessageBox 是否显示右上角关闭按钮	boolean	—	true
beforeClose	MessageBox 关闭前的回调，会暂停实例的关闭	function(action, instance, done)，action 的值为'confirm', 'cancel'或'close'；instance 为 MessageBox 实例，可以通过它访问实例上的属性和方法；done 用于关闭 MessageBox 实例	—	—
distinguishCancelAndClose	是否将取消（点击取消按钮）与关闭（点击关闭按钮或遮罩层、按下 ESC 键）进行区分	boolean	—	false
lockScroll	是否在 MessageBox 出现时将 body 滚动锁定	boolean	—	true
showCancelButton	是否显示取消按钮	boolean	—	false（以 confirm 和 prompt

				方式调用时为 true)
showConfirmButton	是否显示确定按钮	boolean	—	true
cancelButtonText	取消按钮的文本内容	string	—	取消
confirmButtonText	确定按钮的文本内容	string	—	确定
cancelButtonClass	取消按钮的自定义类名	string	—	—
confirmButtonClass	确定按钮的自定义类名	string	—	—
closeOnClickModal	是否可通过点击遮罩关闭 MessageBox	boolean	—	true（以 alert 方式调用时为 false)
closeOnPressEscape	是否可通过按下 ESC 键关闭 MessageBox	boolean	—	true（以 alert 方式调用时为 false)
closeOnHashChange	是否在 hashchange 时关闭 MessageBox	boolean	—	true
showInput	是否显示输入框	boolean	—	false（以 prompt 方式调用时为 true)
inputPlaceholder	输入框的占位符	string	—	—
inputType	输入框的类型	string	—	text
inputValue	输入框的初始文本	string	—	—
inputPattern	输入框的校验表达式	regexp	—	—
inputValidator	输入框的校验函数。可以返回布尔值或字符串，若返回一个字符串, 则返	function	—	—

	回结果会被赋值给 inputErrorMessage			
inputErrorMessage	校验未通过时的提示文本	string	—	输入的数据不合法!
center	是否居中布局	boolean	—	false
roundButton	是否使用圆角按钮	boolean	—	false