Dialog

Informs users while preserving the current page state.

Basic usage

Dialog pops up a dialog box, and it's quite customizable.

:::demo Set the `visible` attribute with a `Boolean`, and Dialog shows when it is `true`. The Dialog has two parts: `body` and `footer`, and the latter requires a `slot` named `footer`. The optional `title` attribute (empty by default) is for defining a title. Finally, this example demonstrates how `before-close` is used.

```
<el-button type="text" @click="dialogVisible = true">click to open the Dialog</el-button>

<el-dialog
  title="Tips"
  :visible.sync="dialogVisible"
  width="30%"
  :before-close="handleClose">
  <span>This is a message</span>
  <span slot="footer" class="dialog-footer">
    <el-button @click="dialogVisible = false">Cancel</el-button>
    <el-button type="primary" @click="dialogVisible = false">Confirm</el-button>
  </span>
</el-dialog>

<script>
  export default {
    data() {
      return {
        dialogVisible: false
      };
    },
    methods: {
      handleClose(done) {
        this.$confirm('Are you sure to close this dialog?')
          .then(_ => {
            done();
          })
          .catch(_ => {});
      }
    }
  };
</script>
```

:::

:::tip `before-close` only works when user clicks the close icon or the backdrop. If you have buttons that close the Dialog in the `footer` named slot, you can add what you would do with `before-close` in the buttons' click event handler. :::

Customizations

The content of Dialog can be anything, even a table or a form. This example shows how to use Element Table and Form with Dialog。

:::demo

```
<!-- Table -->
<el-button type="text" @click="dialogTableVisible = true">open a Table nested
Dialog</el-button>

<el-dialog title="Shipping address" :visible.sync="dialogTableVisible">
  <el-table :data="gridData">
    <el-table-column property="date" label="Date" width="150"></el-table-column>
    <el-table-column property="name" label="Name" width="200"></el-table-column>
    <el-table-column property="address" label="Address"></el-table-column>
  </el-table>
</el-dialog>

<!-- Form -->
<el-button type="text" @click="dialogFormVisible = true">open a Form nested
Dialog</el-button>

<el-dialog title="Shipping address" :visible.sync="dialogFormVisible">
  <el-form :model="form">
    <el-form-item label="Promotion name" :label-width="formLabelWidth">
      <el-input v-model="form.name" autocomplete="off"></el-input>
    </el-form-item>
    <el-form-item label="Zones" :label-width="formLabelWidth">
      <el-select v-model="form.region" placeholder="Please select a zone">
        <el-option label="Zone No.1" value="shanghai"></el-option>
        <el-option label="Zone No.2" value="beijing"></el-option>
      </el-select>
    </el-form-item>
  </el-form>
  <span slot="footer" class="dialog-footer">
    <el-button @click="dialogFormVisible = false">Cancel</el-button>
    <el-button type="primary" @click="dialogFormVisible = false">Confirm</el-button>
  </span>
</el-dialog>

<script>
  export default {
    data() {
      return {
        gridData: [{
          date: '2016-05-02',
          name: 'John Smith',
          address: 'No.1518,  Jinshajiang Road, Putuo District'
        }, {
          date: '2016-05-04',
          name: 'John Smith',
```

```
          address: 'No.1518,  Jinshajiang Road, Putuo District'
        }, {
          date: '2016-05-01',
          name: 'John Smith',
          address: 'No.1518,  Jinshajiang Road, Putuo District'
        }, {
          date: '2016-05-03',
          name: 'John Smith',
          address: 'No.1518,  Jinshajiang Road, Putuo District'
        }],
        dialogTableVisible: false,
        dialogFormVisible: false,
        form: {
          name: '',
          region: '',
          date1: '',
          date2: '',
          delivery: false,
          type: [],
          resource: '',
          desc: ''
        },
        formLabelWidth: '120px'
      };
    }
  };
</script>
```

:::

Nested Dialog

If a Dialog is nested in another Dialog, `append-to-body` is required. :::demo Normally we do not recommend using nested Dialog. If you need multiple Dialogs rendered on the page, you can simply flat them so that they're siblings to each other. If you must nest a Dialog inside another Dialog, set `append-to-body` of the nested Dialog to true, and it will append to body instead of its parent node, so both Dialogs can be correctly rendered.

```
<template>
  <el-button type="text" @click="outerVisible = true">open the outer Dialog</el-
button>

  <el-dialog title="Outer Dialog" :visible.sync="outerVisible">
    <el-dialog
        width="30%"
        title="Inner Dialog"
        :visible.sync="innerVisible"
        append-to-body>
    </el-dialog>
    <div slot="footer" class="dialog-footer">
      <el-button @click="outerVisible = false">Cancel</el-button>
      <el-button type="primary" @click="innerVisible = true">open the inner
```

```
Dialog</el-button>
    </div>
  </el-dialog>
</template>

<script>
  export default {
    data() {
      return {
        outerVisible: false,
        innerVisible: false
      };
    }
  }
</script>
```

:::

Centered content

Dialog's content can be centered.

:::demo Setting `center` to `true` will center dialog's header and footer horizontally. `center` only affects Dialog's header and footer. The body of Dialog can be anything, so sometimes it may not look good when centered. You need to write some CSS if you wish to center the body as well.

```
<el-button type="text" @click="centerDialogVisible = true">Click to open the
Dialog</el-button>

<el-dialog
  title="Warning"
  :visible.sync="centerDialogVisible"
  width="30%"
  center>
  <span>It should be noted that the content will not be aligned in center by
default</span>
  <span slot="footer" class="dialog-footer">
    <el-button @click="centerDialogVisible = false">Cancel</el-button>
    <el-button type="primary" @click="centerDialogVisible = false">Confirm</el-
button>
  </span>
</el-dialog>

<script>
  export default {
    data() {
      return {
        centerDialogVisible: false
      };
    }
  };
</script>
```

:::

:::tip The content of Dialog is lazily rendered, which means the default slot is not rendered onto the DOM until it is firstly opened. Therefore, if you need to perform a DOM manipulation or access a component using `ref`, do it in the `open` event callback. :::

:::tip If the variable bound to `visible` is managed in Vuex store, the `.sync` can not work properly. In this case, please remove the `.sync` modifier, listen to `open` and `close` events of Dialog, and commit Vuex mutations to update the value of that variable in the event handlers. :::

Attributes

Attribute	Description	Type	Accepted Values	Default
visible	visibility of Dialog, supports the .sync modifier	boolean	—	false
title	title of Dialog. Can also be passed with a named slot (see the following table)	string	—	—
width	width of Dialog	string	—	50%
fullscreen	whether the Dialog takes up full screen	boolean	—	false
top	value for `margin-top` of Dialog CSS	string	—	15vh
modal	whether a mask is displayed	boolean	—	true
modal-append-to-body	whether to append modal to body element. If false, the modal will be appended to Dialog's parent element	boolean	—	true
append-to-body	whether to append Dialog itself to body. A nested Dialog should have this attribute set to `true`	boolean	—	false
lock-scroll	whether scroll of body is disabled while Dialog is displayed	boolean	—	true
custom-class	custom class names for Dialog	string	—	—
close-on-click-modal	whether the Dialog can be closed by clicking the mask	boolean	—	true
close-on-press-escape	whether the Dialog can be closed by pressing ESC	boolean	—	true
show-close	whether to show a close button	boolean	—	true
before-	callback before Dialog closes, and it will	function(done),	—	—

close	prevent Dialog from closing	done is used to close the Dialog		
center	whether to align the header and footer in center	boolean	—	false
destroy-on-close	Destroy elements in Dialog when closed	boolean	—	false

Slot

Name	Description
—	content of Dialog
title	content of the Dialog title
footer	content of the Dialog footer

Events

Event Name	Description	Parameters
open	triggers when the Dialog opens	—
opened	triggers when the Dialog opening animation ends	—
close	triggers when the Dialog closes	—
closed	triggers when the Dialog closing animation ends	—