Drawer

A veces, `Dialog` no siempre satisface nuestros requisitos, digamos que tiene un formulario masivo, o necesita espacio para mostrar algo como `terminos & condiciones`, `Drawer` tiene una API casi idéntica a `Dialog`, pero introduce una experiencia de usuario diferente.

Uso básico

Llamada de un drawer temporal, desde varias direcciones

:::demo Debe establecer `visible` para `Drawer` como lo hace `Dialog` para controlar la visibilidad. `visible` es del tipo `boolean`. `Drawer` tiene partes: `title` & `body`, el `title` es un slot con nombre, también puede establecer el título a través de un atributo llamado `title`, por defecto a una cadena vacía, la parte `body` es el área principal de `Drawer`, que contiene contenido definido por el usuario. Al abrir, `Drawer` se expande desde la **esquina derecha a la izquierda** cuyo tamaño es **30%** de la ventana del navegador por defecto. Puede cambiar ese comportamiento predeterminado estableciendo los atributos `direction` y `size`. Este caso de demostración también muestra cómo utilizar la API `before-close`, consulte la sección Atributos para obtener más detalles.

```
<el-radio-group v-model="direction">
  <el-radio label="ltr">left to right</el-radio>
  <el-radio label="rtl">right to left</el-radio>
  <el-radio label="ttb">top to bottom</el-radio>
  <el-radio label="btt">bottom to top</el-radio>
</el-radio-group>

<el-button @click="drawer = true" type="primary" style="margin-left: 16px;">
  open
</el-button>

<el-drawer
  title="I am the title"
  :visible.sync="drawer"
  :direction="direction"
  :before-close="handleClose">
  <span>Hi, there!</span>
</el-drawer>

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

```
          })
          .catch(_ => {});
      }
    }
  };
</script>
```

:::

Sin titulo

Si no necesitas el titulo lo puedes eliminar del drawer.

:::demo Asigne **false** al atributo `withHeader`, se puede eliminar el atributo title del drawer, de esa manera el drawer tendrá mas espacio para el contenido. Por razones de accesibilidad se recomienda asignar siempre un contenido valido al atributo `title`.

```
<el-button @click="drawer = true" type="primary" style="margin-left: 16px;">
  open
</el-button>

<el-drawer
  title="I am the title"
  :visible.sync="drawer"
  :with-header="false">
  <span>Hi there!</span>
</el-drawer>

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

:::

Personalizar el contenido

Al igual que `Dialog`, `Drawer` puede hacer muchas interacciones diversas.

:::demo

```
<el-button type="text" @click="table = true">Open Drawer with nested table</el-
button>
<el-button type="text" @click="dialog = true">Open Drawer with nested form</el-
button>
<el-drawer
  title="I have a nested table inside!"
```

```
  :visible.sync="table"
  direction="rtl"
  size="50%">
   <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-drawer>

<el-drawer
  title="I have a nested form inside!"
  :before-close="handleClose"
  :visible.sync="dialog"
  direction="ltr"
  custom-class="demo-drawer"
  ref="drawer"
  >
  <div class="demo-drawer__content">
    <el-form :model="form">
      <el-form-item label="Name" :label-width="formLabelWidth">
        <el-input v-model="form.name" autocomplete="off"></el-input>
      </el-form-item>
      <el-form-item label="Area" :label-width="formLabelWidth">
        <el-select v-model="form.region" placeholder="Please select activity area">
          <el-option label="Area1" value="shanghai"></el-option>
          <el-option label="Area2" value="beijing"></el-option>
        </el-select>
      </el-form-item>
    </el-form>
    <div class="demo-drawer__footer">
      <el-button @click="cancelForm">Cancel</el-button>
      <el-button type="primary" @click="$refs.drawer.closeDrawer()"
:loading="loading">{{ loading ? 'Submitting ...' : 'Submit' }}</el-button>
    </div>
  </div>
</el-drawer>

<script>
export default {
  data() {
    return {
      table: false,
      dialog: false,
      loading: false,
      gridData: [{
        date: '2016-05-02',
        name: 'Peter Parker',
        address: 'Queens, New York City'
      }, {
        date: '2016-05-04',
        name: 'Peter Parker',
```

```
        address: 'Queens, New York City'
      }, {
        date: '2016-05-01',
        name: 'Peter Parker',
        address: 'Queens, New York City'
      }, {
        date: '2016-05-03',
        name: 'Peter Parker',
        address: 'Queens, New York City'
      }],
      form: {
        name: '',
        region: '',
        date1: '',
        date2: '',
        delivery: false,
        type: [],
        resource: '',
        desc: ''
      },
      formLabelWidth: '80px',
      timer: null,
    };
  },
  methods: {
    handleClose(done) {
      if (this.loading) {
        return;
      }
      this.$confirm('Do you want to submit?')
        .then(_ => {
          this.loading = true;
          this.timer = setTimeout(() => {
            done();
            // animation takes time
            setTimeout(() => {
              this.loading = false;
            }, 400);
          }, 2000);
        })
        .catch(_ => {});
    },
    cancelForm() {
      this.loading = false;
      this.dialog = false;
      clearTimeout(this.timer);
    }
  }
}
</script>
```

:::

Drawer anidados

También puede tener varias capas de `Drawer` al igual que con `Dialog`. :::demo Si necesita varios drawer en diferentes capas, debe establecer el atributo `append-to-body` en **true**

```
<el-button @click="drawer = true" type="primary" style="margin-left: 16px;">
  open
</el-button>

<el-drawer
  title="I'm outer Drawer"
  :visible.sync="drawer"
  size="50%">
  <div>
   <el-button @click="innerDrawer = true">Click me!</el-button>
   <el-drawer
     title="I'm inner Drawer"
     :append-to-body="true"
     :before-close="handleClose"
     :visible.sync="innerDrawer">
     <p>_(:з」∠)_</p>
   </el-drawer>
  </div>
</el-drawer>

<script>
  export default {
    data() {
      return {
        drawer: false,
        innerDrawer: false,
      };
    },
    methods: {
      handleClose(done) {
        this.$confirm('You still have unsaved data, proceed?')
          .then(_ => {
            done();
          })
          .catch(_ => {});
      }
    }
  };
</script>
```

:::

:::tip

El contenido dentro del Drawer debe ser renderizado de forma perezosa, lo que significa que el contenido dentro del Drawer no afectará al rendimiento inicial del renderizado, por lo que cualquier operación DOM debe realizarse a

través de `ref' o después de que se emita el evento` open'.

:::

:::tip

El Drawer proporciona una API llamada "destroyOnClose", que es una variable de bandera que indica que debe destruir el contenido hijo dentro del Drawer después de que se haya cerrado. Puede utilizar esta API cuando necesite que su ciclo de vida "mounted" sea llamado cada vez que se abra el Cajón.

:::

:::tip

Si la variable `visible` se gestiona en el almacén de Vuex, el `.sync` no puede funcionar correctamente. En este caso, elimine el modificador `.sync`, escuche los eventos `open` y `close` de Drawer, y envíe mutaciones Vuex para actualizar el valor de esa variable en los manejadores de eventos.

:::

Atributos de Drawer

Parámetros	Descripción	Tipo	Valores aceptados	Por defecto
append-to-body	Los controles deberían insertar Drawer en el elemento DocumentBody, los Drawer anidados deben asignar este parámetro a **true**	boolean	—	false
before-close	Si está configurado, el procedimiento de cierre se detendrá.	function(done), done es un tipo de función que acepta un booleano como parámetro, una llamada hecha con true o sin parámetro abortará el procedimiento de cierre.	—	—
close-on-press-escape	Indica si el Drawer puede cerrarse pulsando ESC	boolean	—	true
custom-class	Nombre extra de clase para Drawer	string	—	—
destroy-on-close	Indica si los children deben ser destruidos después de cerrar el Drawer.	boolean	-	false
modal	Mostrará una capa de sombra	boolean	—	true
modal-append-to-body	Indica si se debe insertar una capa de sombreado en el elemento DocumentBody	boolean	—	true

direction	Dirección de apertura del Drawer	Direction	rtl / ltr / ttb / btt	rtl
show-close	Se mostrará el botón de cerrar en la parte superior derecha del Drawer	boolean	—	true
size	Tamaño del Drawer. Si el Drawer está en modo horizontal, afecta a la propiedad width, de lo contrario afecta a la propiedad height, cuando el tamaño es tipo `number`, describe el tamaño por unidad de píxeles; cuando el tamaño es tipo `string`, se debe usar con notación `x%`, de lo contrario se interpretará como unidad de píxeles.	number / string	-	'30%'
title	El título del Drawer, también se puede establecer por slot con nombre, las descripciones detalladas se pueden encontrar en el formulario de slot.	string	—	—
visible	Si se muestra el Drawer, también soporta la notación `.sync`	boolean	—	false
wrapperClosable	Indica si el usuario puede cerrar el Drawer haciendo clic en la capa de sombreado.	boolean	-	true
withHeader	Indica si la sección header existirá, por defecto es true, cuando es false no tienen efecto, ambos, `title attribute` y `title slot`	boolean	-	true

Drawer Slot's

Nombre	Descripción
—	El contenido del Drawer
title	El titulo de la sección del Drawer

Métodos Drawer

Nombre	Descripción
closeDrawer	Para cerrar el Drawer, este método llamará `before-close`.

Eventos Drawer

Nombre	Descripción	Parámetros

open	Se activa antes de que comience la animación de apertura del Drawer.	—
opened	Se activa cuando finaliza la animación de apertura del Drawer.	—
close	Se activa antes de que comience la animación de cierre del Drawer.	—
closed	Se activa después de que finaliza la animación de cierre del Drawer.	—