Tabs

Divide data collections which are related yet belong to different types.

Basic usage

Basic and concise tabs.

:::demo Tabs provide a selective card functionality. By default the first tab is selected as active, and you can activate any tab by setting the `value` attribute.

```
<template>
  <el-tabs v-model="activeName" @tab-click="handleClick">
    <el-tab-pane label="User" name="first">User</el-tab-pane>
    <el-tab-pane label="Config" name="second">Config</el-tab-pane>
    <el-tab-pane label="Role" name="third">Role</el-tab-pane>
    <el-tab-pane label="Task" name="fourth">Task</el-tab-pane>
  </el-tabs>
</template>
<script>
  export default {
    data() {
      return {
        activeName: 'first'
      };
    },
    methods: {
      handleClick(tab, event) {
        console.log(tab, event);
      }
    }
  };
</script>
```

:::

Card Style

Tabs styled as cards.

:::demo Set `type` to `card` can get a card-styled tab.

```
<template>
  <el-tabs type="card" @tab-click="handleClick">
    <el-tab-pane label="User">User</el-tab-pane>
    <el-tab-pane label="Config">Config</el-tab-pane>
    <el-tab-pane label="Role">Role</el-tab-pane>
    <el-tab-pane label="Task">Task</el-tab-pane>
  </el-tabs>
</template>
<script>
  export default {
```

```
    data() {
      return {
        activeName: 'first'
      };
    },
    methods: {
      handleClick(tab, event) {
        console.log(tab, event);
      }
    }
  };
</script>
```

:::

Border card

Border card tabs.

:::demo Set `type` to `border-card`.

```
<el-tabs type="border-card">
  <el-tab-pane label="User">User</el-tab-pane>
  <el-tab-pane label="Config">Config</el-tab-pane>
  <el-tab-pane label="Role">Role</el-tab-pane>
  <el-tab-pane label="Task">Task</el-tab-pane>
</el-tabs>
```

:::

Tab position

You can use `tab-position` attribute to set the tab's position.

:::demo You can choose from four directions: `tabPosition="left|right|top|bottom"`

```
<template>
  <el-radio-group v-model="tabPosition" style="margin-bottom: 30px;">
    <el-radio-button label="top">top</el-radio-button>
    <el-radio-button label="right">right</el-radio-button>
    <el-radio-button label="bottom">bottom</el-radio-button>
    <el-radio-button label="left">left</el-radio-button>
  </el-radio-group>

  <el-tabs :tab-position="tabPosition" style="height: 200px;">
    <el-tab-pane label="User">User</el-tab-pane>
    <el-tab-pane label="Config">Config</el-tab-pane>
    <el-tab-pane label="Role">Role</el-tab-pane>
    <el-tab-pane label="Task">Task</el-tab-pane>
  </el-tabs>
</template>
<script>
```

```
  export default {
    data() {
      return {
        tabPosition: 'left'
      };
    }
  };
</script>
```

:::

Custom Tab

You can use named slot to customize the tab label content.

:::demo

```
<el-tabs type="border-card">
  <el-tab-pane>
    <span slot="label"><i class="el-icon-date"></i> Route</span>
    Route
  </el-tab-pane>
  <el-tab-pane label="Config">Config</el-tab-pane>
  <el-tab-pane label="Role">Role</el-tab-pane>
  <el-tab-pane label="Task">Task</el-tab-pane>
</el-tabs>
```

:::

Add & close tab

Only card type Tabs support addable & closeable.

:::demo

```
<el-tabs v-model="editableTabsValue" type="card" editable @edit="handleTabsEdit">
  <el-tab-pane
    v-for="(item, index) in editableTabs"
    :key="item.name"
    :label="item.title"
    :name="item.name"
  >
    {{item.content}}
  </el-tab-pane>
</el-tabs>
<script>
  export default {
    data() {
      return {
        editableTabsValue: '2',
        editableTabs: [{
          title: 'Tab 1',
```

```
          name: '1',
          content: 'Tab 1 content'
        }, {
          title: 'Tab 2',
          name: '2',
          content: 'Tab 2 content'
        }],
        tabIndex: 2
      }
    },
    methods: {
      handleTabsEdit(targetName, action) {
        if (action === 'add') {
          let newTabName = ++this.tabIndex + '';
          this.editableTabs.push({
            title: 'New Tab',
            name: newTabName,
            content: 'New Tab content'
          });
          this.editableTabsValue = newTabName;
        }
        if (action === 'remove') {
          let tabs = this.editableTabs;
          let activeName = this.editableTabsValue;
          if (activeName === targetName) {
            tabs.forEach((tab, index) => {
              if (tab.name === targetName) {
                let nextTab = tabs[index + 1] || tabs[index - 1];
                if (nextTab) {
                  activeName = nextTab.name;
                }
              }
            });
          }

          this.editableTabsValue = activeName;
          this.editableTabs = tabs.filter(tab => tab.name !== targetName);
        }
      }
    }
  }
</script>
```

:::

Customized trigger button of new tab

:::demo

```
<div style="margin-bottom: 20px;">
  <el-button
    size="small"
```

```
    @click="addTab(editableTabsValue)"
  >
    add tab
  </el-button>
</div>
<el-tabs v-model="editableTabsValue" type="card" closable @tab-remove="removeTab">
  <el-tab-pane
    v-for="(item, index) in editableTabs"
    :key="item.name"
    :label="item.title"
    :name="item.name"
  >
    {{item.content}}
  </el-tab-pane>
</el-tabs>
<script>
  export default {
    data() {
      return {
        editableTabsValue: '2',
        editableTabs: [{
          title: 'Tab 1',
          name: '1',
          content: 'Tab 1 content'
        }, {
          title: 'Tab 2',
          name: '2',
          content: 'Tab 2 content'
        }],
        tabIndex: 2
      }
    },
    methods: {
      addTab(targetName) {
        let newTabName = ++this.tabIndex + '';
        this.editableTabs.push({
          title: 'New Tab',
          name: newTabName,
          content: 'New Tab content'
        });
        this.editableTabsValue = newTabName;
      },
      removeTab(targetName) {
        let tabs = this.editableTabs;
        let activeName = this.editableTabsValue;
        if (activeName === targetName) {
          tabs.forEach((tab, index) => {
            if (tab.name === targetName) {
              let nextTab = tabs[index + 1] || tabs[index - 1];
              if (nextTab) {
                activeName = nextTab.name;
              }
```

```
            }
          });
        }

        this.editableTabsValue = activeName;
        this.editableTabs = tabs.filter(tab => tab.name !== targetName);
      }
    }
  }
</script>
```

:::

Tabs Attributes

Attribute	Description	Type	Accepted Values	Default
value / v-model	binding value, name of the selected tab	string	—	name of first tab
type	type of Tab	string	card/border-card	—
closable	whether Tab is closable	boolean	—	false
addable	whether Tab is addable	boolean	—	false
editable	whether Tab is addable and closable	boolean	—	false
tab-position	position of tabs	string	top/right/bottom/left	top
stretch	whether width of tab automatically fits its container	boolean	-	false
before-leave	hook function before switching tab. If `false` is returned or a `Promise` is returned and then is rejected, switching will be prevented	Function(activeName, oldActiveName)	—	—

Tabs Events

Event Name	Description	Parameters
tab-click	triggers when a tab is clicked	clicked tab
tab-remove	triggers when tab-remove button is clicked	name of the removed tab
tab-add	triggers when tab-add button is clicked	—
edit	triggers when tab-add button or tab-remove is clicked	(targetName, action)

Tab-pane Attributes

Attribute	Description	Type	Accepted Values	Default
label	title of the tab	string	—	—
disabled	whether Tab is disabled	boolean	—	false
name	identifier corresponding to the name of Tabs, representing the alias of the tab-pane	string	—	ordinal number of the tab-pane in the sequence, e.g. the first tab-pane is '1'
closable	whether Tab is closable	boolean	—	false
lazy	whether Tab is lazily rendered	boolean	—	false