何时使用 {#when-to-use}

提供平级的区域将大块内容进行收纳和展现，保持界面整洁。

Ant Design 依次提供了三级选项卡，分别用于不同的场景。

- 卡片式的页签，提供可关闭的样式，常用于容器顶部。
- 既可用于容器顶部，也可用于容器内部，是最通用的 Tabs。
- Radio.Button 可作为更次级的页签来使用。

代码演示

基本

```
import React from 'react';
import { Tabs } from 'antd';
import type { TabsProps } from 'antd';

const onChange = (key: string) => {
  console.log(key);
};

const items: TabsProps['items'] = [
  {
    key: '1',
    label: 'Tab 1',
    children: 'Content of Tab Pane 1',
  },
  {
    key: '2',
    label: 'Tab 2',
    children: 'Content of Tab Pane 2',
  },
  {
    key: '3',
    label: 'Tab 3',
    children: 'Content of Tab Pane 3',
  },
];

const App: React.FC = () => <Tabs defaultActiveKey="1" items={items}
onChange={onChange} />;

export default App;
```

禁用

```tsx
import React from 'react';
import { Tabs } from 'antd';

const App: React.FC = () => (
  <Tabs
    defaultActiveKey="1"
    items={[
      {
        label: 'Tab 1',
        key: '1',
        children: 'Tab 1',
      },
      {
        label: 'Tab 2',
        key: '2',
        children: 'Tab 2',
        disabled: true,
      },
      {
        label: 'Tab 3',
        key: '3',
        children: 'Tab 3',
      },
    ]}
  />
);

export default App;
```

居中

```tsx
import React from 'react';
import { Tabs } from 'antd';

const App: React.FC = () => (
  <Tabs
    defaultActiveKey="1"
    centered
    items={Array.from({ length: 3 }).map((_, i) => {
      const id = String(i + 1);
      return {
        label: `Tab ${id}`,
        key: id,
        children: `Content of Tab Pane ${id}`,
      };
```

```
    })}
  />
);

export default App;
```

图标

```
import React from 'react';
import { AndroidOutlined, AppleOutlined } from '@ant-design/icons';
import { Tabs } from 'antd';

const App: React.FC = () => (
  <Tabs
    defaultActiveKey="2"
    items={[AppleOutlined, AndroidOutlined].map((Icon, i) => {
      const id = String(i + 1);
      return {
        key: id,
        label: `Tab ${id}`,
        children: `Tab ${id}`,
        icon: <Icon />,
      };
    })}
  />
);

export default App;
```

指示条

```
import React from 'react';
import { Segmented, Tabs } from 'antd';
import type { TabsProps } from 'antd';

const onChange = (key: string) => {
  console.log(key);
};

const items: TabsProps['items'] = [
  { key: '1', label: 'Tab 1', children: 'Content of Tab Pane 1' },
  { key: '2', label: 'Tab 2', children: 'Content of Tab Pane 2' },
  { key: '3', label: 'Tab 3', children: 'Content of Tab Pane 3' },
];
```

```tsx
type Align = 'start' | 'center' | 'end';

const App: React.FC = () => {
  const [alignValue, setAlignValue] = React.useState<Align>('center');
  return (
    <>
      <Segmented
        value={alignValue}
        style={{ marginBottom: 8 }}
        onChange={setAlignValue}
        options={['start', 'center', 'end']}
      />
      <Tabs
        defaultActiveKey="1"
        items={items}
        onChange={onChange}
        indicator={{ size: (origin) => origin - 20, align: alignValue }}
      />
    </>
  );
};

export default App;
```

滑动

```tsx
import React, { useState } from 'react';
import type { RadioChangeEvent } from 'antd';
import { Radio, Tabs } from 'antd';

type TabPosition = 'left' | 'right' | 'top' | 'bottom';

const App: React.FC = () => {
  const [mode, setMode] = useState<TabPosition>('top');

  const handleModeChange = (e: RadioChangeEvent) => {
    setMode(e.target.value);
  };

  return (
    <div>
      <Radio.Group onChange={handleModeChange} value={mode} style={{
marginBottom: 8 }}>
        <Radio.Button value="top">Horizontal</Radio.Button>
        <Radio.Button value="left">Vertical</Radio.Button>
      </Radio.Group>
```

```
      <Tabs
        defaultActiveKey="1"
        tabPosition={mode}
        style={{ height: 220 }}
        items={Array.from({ length: 30 }, (_, i) => {
          const id = String(i);
          return {
            label: `Tab-${id}`,
            key: id,
            disabled: i === 28,
            children: `Content of tab ${id}`,
          };
        })}
      />
    </div>
  );
};

export default App;
```

附加内容

```
import React, { useMemo, useState } from 'react';
import { Button, Checkbox, Divider, Tabs } from 'antd';

const CheckboxGroup = Checkbox.Group;

const operations = <Button>Extra Action</Button>;

const OperationsSlot: Record<PositionType, React.ReactNode> = {
  left: <Button className="tabs-extra-demo-button">Left Extra
Action</Button>,
  right: <Button>Right Extra Action</Button>,
};

const options = ['left', 'right'];

type PositionType = 'left' | 'right';

const items = Array.from({ length: 3 }).map((_, i) => {
  const id = String(i + 1);
  return {
    label: `Tab ${id}`,
    key: id,
    children: `Content of tab ${id}`,
  };
```

```
});

const App: React.FC = () => {
  const [position, setPosition] = useState<PositionType[]>(['left',
'right']);

  const slot = useMemo(() => {
    if (position.length === 0) {
      return null;
    }
    return position.reduce(
      (acc, direction) => ({ ...acc, [direction]: OperationsSlot[direction]
}),
      {},
    );
  }, [position]);

  return (
    <>
      <Tabs tabBarExtraContent={operations} items={items} />
      <br />
      <br />
      <br />
      <div>You can also specify its direction or both side</div>
      <Divider />
      <CheckboxGroup
        options={options}
        value={position}
        onChange={(value) => {
          setPosition(value as PositionType[]);
        }}
      />
      <br />
      <br />
      <Tabs tabBarExtraContent={slot} items={items} />
    </>
  );
};

export default App;
```

大小

```
import React, { useState } from 'react';
import type { RadioChangeEvent, TabsProps } from 'antd';
import { Radio, Tabs } from 'antd';
```

```tsx
type TargetKey = React.MouseEvent | React.KeyboardEvent | string;

const App: React.FC = () => {
  const [size, setSize] = useState<'small' | 'middle' | 'large'>('small');
  const [activeKey, setActiveKey] = useState('1');
  const [items, setItems] = useState<TabsProps['items']>([
    {
      label: 'Tab 1',
      key: '1',
      children: 'Content of editable tab 1',
    },
    {
      label: 'Tab 2',
      key: '2',
      children: 'Content of editable tab 2',
    },
    {
      label: 'Tab 3',
      key: '3',
      children: 'Content of editable tab 3',
    },
  ]);

  const add = () => {
    const newKey = String((items || []).length + 1);
    setItems([
      ...(items || []),
      {
        label: `Tab ${newKey}`,
        key: newKey,
        children: `Content of editable tab ${newKey}`,
      },
    ]);
    setActiveKey(newKey);
  };

  const remove = (targetKey: TargetKey) => {
    if (!items) return;
    const targetIndex = items.findIndex((item) => item.key === targetKey);
    const newItems = items.filter((item) => item.key !== targetKey);

    if (newItems.length && targetKey === activeKey) {
      const newActiveKey =
        newItems[targetIndex === newItems.length ? targetIndex - 1 :
targetIndex].key;
```

```
      setActiveKey(newActiveKey);
    }

    setItems(newItems);
  };

  const onEdit = (targetKey: TargetKey, action: 'add' | 'remove') => {
    if (action === 'add') {
      add();
    } else {
      remove(targetKey);
    }
  };

  const onChange = (e: RadioChangeEvent) => {
    setSize(e.target.value);
  };

  return (
    <div>
      <Radio.Group value={size} onChange={onChange} style={{ marginBottom:
16 }}>
        <Radio.Button value="small">Small</Radio.Button>
        <Radio.Button value="middle">Middle</Radio.Button>
        <Radio.Button value="large">Large</Radio.Button>
      </Radio.Group>
      <Tabs
        defaultActiveKey="1"
        size={size}
        style={{ marginBottom: 32 }}
        items={Array.from({ length: 3 }).map((_, i) => {
          const id = String(i + 1);
          return {
            label: `Tab ${id}`,
            key: id,
            children: `Content of tab ${id}`,
          };
        })}
      />
      <Tabs
        defaultActiveKey="1"
        type="card"
        size={size}
        style={{ marginBottom: 32 }}
        items={Array.from({ length: 3 }).map((_, i) => {
          const id = String(i + 1);
```

```
          return {
            label: `Card Tab ${id}`,
            key: id,
            children: `Content of card tab ${id}`,
          };
        })}
      />
      <Tabs
        type="editable-card"
        size={size}
        activeKey={activeKey}
        onChange={setActiveKey}
        onEdit={onEdit}
        items={items}
      />
    </div>
  );
};

export default App;
```

位置

```
import React, { useState } from 'react';
import type { RadioChangeEvent } from 'antd';
import { Radio, Space, Tabs } from 'antd';

type TabPosition = 'left' | 'right' | 'top' | 'bottom';

const App: React.FC = () => {
  const [tabPosition, setTabPosition] = useState<TabPosition>('left');

  const changeTabPosition = (e: RadioChangeEvent) => {
    setTabPosition(e.target.value);
  };

  return (
    <>
      <Space style={{ marginBottom: 24 }}>
        Tab position:
        <Radio.Group value={tabPosition} onChange={changeTabPosition}>
          <Radio.Button value="top">top</Radio.Button>
          <Radio.Button value="bottom">bottom</Radio.Button>
          <Radio.Button value="left">left</Radio.Button>
          <Radio.Button value="right">right</Radio.Button>
        </Radio.Group>
```

```
      </Space>
      <Tabs
        tabPosition={tabPosition}
        items={Array.from({ length: 3 }).map((_, i) => {
          const id = String(i + 1);
          return {
            label: `Tab ${id}`,
            key: id,
            children: `Content of Tab ${id}`,
          };
        })}
      />
    </>
  );
};

export default App;
```

卡片式页签

```
import React from 'react';
import { Tabs } from 'antd';

const onChange = (key: string) => {
  console.log(key);
};

const App: React.FC = () => (
  <Tabs
    onChange={onChange}
    type="card"
    items={Array.from({ length: 3 }).map((_, i) => {
      const id = String(i + 1);
      return {
        label: `Tab ${id}`,
        key: id,
        children: `Content of Tab Pane ${id}`,
      };
    })}
  />
);

export default App;
```

新增和关闭页签

```tsx
import React, { useRef, useState } from 'react';
import { Tabs } from 'antd';

type TargetKey = React.MouseEvent | React.KeyboardEvent | string;

const initialItems = [
  { label: 'Tab 1', children: 'Content of Tab 1', key: '1' },
  { label: 'Tab 2', children: 'Content of Tab 2', key: '2' },
  {
    label: 'Tab 3',
    children: 'Content of Tab 3',
    key: '3',
    closable: false,
  },
];

const App: React.FC = () => {
  const [activeKey, setActiveKey] = useState(initialItems[0].key);
  const [items, setItems] = useState(initialItems);
  const newTabIndex = useRef(0);

  const onChange = (newActiveKey: string) => {
    setActiveKey(newActiveKey);
  };

  const add = () => {
    const newActiveKey = `newTab${newTabIndex.current++}`;
    const newPanes = [...items];
    newPanes.push({ label: 'New Tab', children: 'Content of new Tab', key:
newActiveKey });
    setItems(newPanes);
    setActiveKey(newActiveKey);
  };

  const remove = (targetKey: TargetKey) => {
    let newActiveKey = activeKey;
    let lastIndex = -1;
    items.forEach((item, i) => {
      if (item.key === targetKey) {
        lastIndex = i - 1;
      }
    });
    const newPanes = items.filter((item) => item.key !== targetKey);
    if (newPanes.length && newActiveKey === targetKey) {
      if (lastIndex >= 0) {
        newActiveKey = newPanes[lastIndex].key;
```

```
      } else {
        newActiveKey = newPanes[0].key;
      }
    }
    setItems(newPanes);
    setActiveKey(newActiveKey);
  };

  const onEdit = (
    targetKey: React.MouseEvent | React.KeyboardEvent | string,
    action: 'add' | 'remove',
  ) => {
    if (action === 'add') {
      add();
    } else {
      remove(targetKey);
    }
  };

  return (
    <Tabs
      type="editable-card"
      onChange={onChange}
      activeKey={activeKey}
      onEdit={onEdit}
      items={items}
    />
  );
};

export default App;
```

卡片式页签容器

Debug

```
import React from 'react';
import { Tabs } from 'antd';
import { createStyles } from 'antd-style';

const useStyle = createStyles(({ token, css }) => {
  const antdTabsCls = '.ant-tabs';

  return css`
    ${antdTabsCls}${antdTabsCls}-card {
      ${antdTabsCls}-content {
```

```
        padding: ${token.padding}px;
        background: ${token.colorBgContainer};
      }

      ${antdTabsCls}-nav {
        margin: 0;

        ${antdTabsCls}-nav-wrap > ${antdTabsCls}-nav-list > ${antdTabsCls}-
tab {
          background: transparent;
          border-color: transparent;

          &-active {
            border-color: ${token.colorBorderBg};
            background: ${token.colorBgContainer};
          }
        }

        &::before {
          display: none;
        }
      }
    }
  `;
});

const items = Array.from({ length: 3 }).map((_, i) => {
  const id = String(i + 1);
  return {
    label: `Tab Title ${id}`,
    key: id,
    children: (
      <>
        <p>Content of Tab Pane {id}</p>
        <p>Content of Tab Pane {id}</p>
        <p>Content of Tab Pane {id}</p>
      </>
    ),
  };
});

const App = () => {
  const { styles } = useStyle();

  return (
    <div className={styles}>
```

```
      <Tabs type="card" items={items} />
    </div>
  );
};

export default App;
```

自定义新增页签触发器

```
import React, { useRef, useState } from 'react';
import { Button, Tabs } from 'antd';

type TargetKey = React.MouseEvent | React.KeyboardEvent | string;

const defaultPanes = Array.from({ length: 2 }).map((_, index) => {
  const id = String(index + 1);
  return { label: `Tab ${id}`, children: `Content of Tab Pane ${index +
1}`, key: id };
});

const App: React.FC = () => {
  const [activeKey, setActiveKey] = useState(defaultPanes[0].key);
  const [items, setItems] = useState(defaultPanes);
  const newTabIndex = useRef(0);

  const onChange = (key: string) => {
    setActiveKey(key);
  };

  const add = () => {
    const newActiveKey = `newTab${newTabIndex.current++}`;
    setItems([...items, { label: 'New Tab', children: 'New Tab Pane', key:
newActiveKey }]);
    setActiveKey(newActiveKey);
  };

  const remove = (targetKey: TargetKey) => {
    const targetIndex = items.findIndex((pane) => pane.key === targetKey);
    const newPanes = items.filter((pane) => pane.key !== targetKey);
    if (newPanes.length && targetKey === activeKey) {
      const { key } = newPanes[targetIndex === newPanes.length ?
targetIndex - 1 : targetIndex];
      setActiveKey(key);
    }
    setItems(newPanes);
  };
```

```
  const onEdit = (targetKey: TargetKey, action: 'add' | 'remove') => {
    if (action === 'add') {
      add();
    } else {
      remove(targetKey);
    }
  };

  return (
    <div>
      <div style={{ marginBottom: 16 }}>
        <Button onClick={add}>ADD</Button>
      </div>
      <Tabs
        hideAdd
        onChange={onChange}
        activeKey={activeKey}
        type="editable-card"
        onEdit={onEdit}
        items={items}
      />
    </div>
  );
};

export default App;
```

自定义页签头

```
import React from 'react';
import type { TabsProps } from 'antd';
import { Tabs, theme } from 'antd';
import StickyBox from 'react-sticky-box';

const items = Array.from({ length: 3 }).map((_, i) => {
  const id = String(i + 1);
  return {
    label: `Tab ${id}`,
    key: id,
    children: `Content of Tab Pane ${id}`,
    style: i === 0 ? { height: 200 } : undefined,
  };
});

const App: React.FC = () => {
```

```
  const {
    token: { colorBgContainer },
  } = theme.useToken();
  const renderTabBar: TabsProps['renderTabBar'] = (props, DefaultTabBar) =>
(
    <StickyBox offsetTop={64} offsetBottom={20} style={{ zIndex: 1 }}>
      <DefaultTabBar {...props} style={{ background: colorBgContainer }} />
    </StickyBox>
  );
  return <Tabs defaultActiveKey="1" renderTabBar={renderTabBar} items=
{items} />;
};

export default App;
```

可拖拽标签

```
import React, { useState } from 'react';
import type { DragEndEvent } from '@dnd-kit/core';
import { closestCenter, DndContext, PointerSensor, useSensor } from '@dnd-
kit/core';
import {
  arrayMove,
  horizontalListSortingStrategy,
  SortableContext,
  useSortable,
} from '@dnd-kit/sortable';
import { CSS } from '@dnd-kit/utilities';
import { Tabs } from 'antd';
import type { TabsProps } from 'antd';

interface DraggableTabPaneProps extends
React.HTMLAttributes<HTMLDivElement> {
  'data-node-key': string;
}

const DraggableTabNode: React.FC<Readonly<DraggableTabPaneProps>> = ({
className, ...props }) => {
  const { attributes, listeners, setNodeRef, transform, transition } =
useSortable({
    id: props['data-node-key'],
  });

  const style: React.CSSProperties = {
    ...props.style,
    transform: CSS.Translate.toString(transform),
```

```
    transition,
    cursor: 'move',
  };

  return React.cloneElement(props.children as React.ReactElement<any>, {
    ref: setNodeRef,
    style,
    ...attributes,
    ...listeners,
  });
};

const App: React.FC = () => {
  const [items, setItems] = useState<NonNullable<TabsProps['items']>>([
    { key: '1', label: 'Tab 1', children: 'Content of Tab Pane 1' },
    { key: '2', label: 'Tab 2', children: 'Content of Tab Pane 2' },
    { key: '3', label: 'Tab 3', children: 'Content of Tab Pane 3' },
  ]);

  const sensor = useSensor(PointerSensor, { activationConstraint: {
distance: 10 } });

  const onDragEnd = ({ active, over }: DragEndEvent) => {
    if (active.id !== over?.id) {
      setItems((prev) => {
        const activeIndex = prev.findIndex((i) => i.key === active.id);
        const overIndex = prev.findIndex((i) => i.key === over?.id);
        return arrayMove(prev, activeIndex, overIndex);
      });
    }
  };

  return (
    <Tabs
      items={items}
      renderTabBar={(tabBarProps, DefaultTabBar) => (
        <DndContext sensors={[sensor]} onDragEnd={onDragEnd}
collisionDetection={closestCenter}>
          <SortableContext items={items.map((i) => i.key)} strategy=
{horizontalListSortingStrategy}>
            <DefaultTabBar {...tabBarProps}>
              {(node) => (
                <DraggableTabNode
                  {...(node as
React.ReactElement<DraggableTabPaneProps>).props}
                  key={node.key}
```

```
                >
                  {node}
                </DraggableTabNode>
              )}
            </DefaultTabBar>
          </SortableContext>
        </DndContext>
      )}
    />
  );
};

export default App;
```

动画

Debug

```
import React from 'react';
import { Space, Switch, Tabs } from 'antd';

const App: React.FC = () => {
  const [inkBar, setInkBar] = React.useState(true);
  const [tabPane, setTabPane] = React.useState(true);

  return (
    <>
      <Space>
        <Switch
          checkedChildren="inkBar"
          unCheckedChildren="inkBar"
          checked={inkBar}
          onChange={() => setInkBar(!inkBar)}
        />
        <Switch
          checkedChildren="tabPane"
          unCheckedChildren="tabPane"
          checked={tabPane}
          onChange={() => setTabPane(!tabPane)}
        />
      </Space>

      <Tabs
        animated={{ inkBar, tabPane }}
        items={[
```

```
          {
            label: 'Bamboo',
            key: '1',
            children: 'Hello Bamboo!',
            style: {
              height: 200,
              boxShadow: '0 0 3px rgba(255, 0, 0, 0.5)',
            },
          },
          {
            label: 'Little',
            key: '2',
            children: 'Hi Little!',
            style: {
              height: 300,
              boxShadow: '0 0 3px rgba(0, 255, 0, 0.5)',
            },
          },
          {
            label: 'Light',
            key: '3',
            children: 'Welcome Light!',
            style: {
              height: 100,
              boxShadow: '0 0 3px rgba(0, 0, 255, 0.5)',
            },
          },
        ]}
      />
    </>
  );
};

export default App;
```

嵌套

Debug

```
import React, { useState } from 'react';
import { Select, Tabs } from 'antd';

const { Option } = Select;

const positionList = ['left', 'right', 'top', 'bottom'];
```

```tsx
const App: React.FC = () => {
  const [parentPos, setParentPos] = useState(undefined);
  const [childPos, setChildPos] = useState(undefined);
  const [parentType, setParentType] = useState(undefined);
  const [childType, setChildType] = useState(undefined);

  return (
    <div>
      <Select
        style={{ width: 200 }}
        onChange={(val) => {
          setParentPos(val);
        }}
      >
        {positionList.map((pos) => (
          <Option key={pos} value={pos}>
            Parent - {pos}
          </Option>
        ))}
      </Select>

      <Select
        style={{ width: 200 }}
        onChange={(val) => {
          setChildPos(val);
        }}
      >
        {positionList.map((pos) => (
          <Option key={pos} value={pos}>
            Child - {pos}
          </Option>
        ))}
      </Select>

      <Select
        style={{ width: 200 }}
        onChange={(val) => {
          setParentType(val);
        }}
      >
        <Option value="line">Parent - line</Option>
        <Option value="card">Parent - card</Option>
        <Option value="editable-card">Parent - card edit</Option>
      </Select>

      <Select
```

```
        style={{ width: 200 }}
        onChange={(val) => {
          setChildType(val);
        }}
      >
        <Option value="line">Child - line</Option>
        <Option value="card">Child - card</Option>
        <Option value="editable-card">Parent - card edit</Option>
      </Select>
      <Tabs
        defaultActiveKey="1"
        tabPosition={parentPos}
        type={parentType}
        items={[
          {
            label: 'Tab 1',
            key: '1',
            children: (
              <Tabs
                defaultActiveKey="1"
                tabPosition={childPos}
                type={childType}
                style={{ height: 300 }}
                items={Array.from({ length: 20 }).map((_, index) => {
                  const key = String(index);
                  return {
                    label: `Tab ${key}`,
                    key,
                    children: `TTTT ${key}`,
                  };
                })}
              />
            ),
          },
          {
            label: 'Tab 2',
            key: '2',
            children: 'Content of Tab Pane 2',
          },
        ]}
      />
    </div>
  );
};

export default App;
```

组件 Token

Debug

```
import React from 'react';
import { Button, ConfigProvider, Tabs } from 'antd';

const App: React.FC = () => (
  <ConfigProvider
    theme={{
      components: {
        Tabs: {
          cardBg: '#f6ffed',
          cardHeight: 60,
          cardPadding: `20px`,
          cardPaddingSM: `20px`,
          cardPaddingLG: `20px`,
          titleFontSize: 20,
          titleFontSizeLG: 20,
          titleFontSizeSM: 20,
          inkBarColor: '#52C41A',
          horizontalMargin: `0 0 12px 0`,
          horizontalItemGutter: 12, // Fixed Value
          horizontalItemPadding: `20px`,
          horizontalItemPaddingSM: `20px`,
          horizontalItemPaddingLG: `20px`,
          verticalItemPadding: `8px`,
          verticalItemMargin: `4px 0 0 0`,
          itemColor: 'rgba(0,0,0,0.85)',
          itemSelectedColor: '#389e0d',
          itemHoverColor: '#d9f7be',
          itemActiveColor: '#b7eb8f',
          cardGutter: 12,
        },
      },
    }}
  >
    <div>
      <Tabs
        defaultActiveKey="1"
        tabBarExtraContent={<Button>Extra Action</Button>}
        style={{ marginBottom: 32 }}
        items={Array.from({ length: 3 }).map((_, i) => {
          const id = String(i + 1);
          return {
            label: `Tab ${id}`,
```

```
      key: id,
      children: `Content of tab ${id}`,
    };
  })}
/>
<Tabs
  tabPosition="left"
  defaultActiveKey="1"
  tabBarExtraContent={<Button>Extra Action</Button>}
  style={{ marginBottom: 32 }}
  items={Array.from({ length: 3 }).map((_, i) => {
    const id = String(i + 1);
    return {
      label: `Tab ${id}`,
      key: id,
      children: `Content of tab ${id}`,
    };
  })}
/>
<Tabs
  size="small"
  defaultActiveKey="1"
  tabBarExtraContent={<Button>Extra Action</Button>}
  style={{ marginBottom: 32 }}
  items={Array.from({ length: 3 }).map((_, i) => {
    const id = String(i + 1);
    return {
      label: `Tab ${id}`,
      key: id,
      children: `Content of tab ${id}`,
    };
  })}
/>
<Tabs
  size="large"
  defaultActiveKey="1"
  tabBarExtraContent={<Button>Extra Action</Button>}
  style={{ marginBottom: 32 }}
  items={Array.from({ length: 3 }).map((_, i) => {
    const id = String(i + 1);
    return {
      label: `Tab ${id}`,
      key: id,
      children: `Content of tab ${id}`,
    };
  })}
```

```
    />
    <Tabs
      defaultActiveKey="1"
      centered
      type="card"
      items={Array.from({ length: 3 }).map((_, i) => {
        const id = String(i + 1);
        return {
          disabled: i === 2,
          label: `Tab ${id}`,
          key: id,
          children: `Content of Tab Pane ${id}`,
        };
      })}
    />
    <Tabs
      size="small"
      defaultActiveKey="1"
      centered
      type="card"
      items={Array.from({ length: 3 }).map((_, i) => {
        const id = String(i + 1);
        return {
          disabled: i === 2,
          label: `Tab ${id}`,
          key: id,
          children: `Content of Tab Pane ${id}`,
        };
      })}
    />
    <Tabs
      size="large"
      defaultActiveKey="1"
      centered
      type="card"
      items={Array.from({ length: 3 }).map((_, i) => {
        const id = String(i + 1);
        return {
          disabled: i === 2,
          label: `Tab ${id}`,
          key: id,
          children: `Content of Tab Pane ${id}`,
        };
      })}
    />
  </div>
```

```
  </ConfigProvider>
);

export default App;
```

API

通用属性参考：通用属性

Tabs

参数	说明	类型	默认值
activeKey	当前激活 tab 面板的 key	string	-
addIcon	自定义添加按钮，设置 type="editable-card" 时有效	ReactNode	<PlusOutlined /
animated	是否使用动画切换 Tabs	boolean| { inkBar: boolean, tabPane: boolean }	{ inkBar: true, tabPane: false }
centered	标签居中展示	boolean	false
defaultActiveKey	初始化选中面板的 key，如果没有设置 activeKey	string	第一个面板的 key
hideAdd	是否隐藏加号图标，在 type="editable-card" 时有效	boolean	false
indicator	自定义指示条的长度和对齐方式	{ size?: number | (origin: number) => number; align: start | center | end; }	-
items	配置选项卡内容	TabItemType	[]
more	自定义折叠菜单属性	MoreProps	{ icon: <EllipsisOutlin /> , trigger: 'hover
removeIcon	自定义删除按钮，设置	ReactNode	<CloseOutlined

	type="editable-card" 时有效		
popupClassName	更多菜单的 className	string	-
renderTabBar	替换 TabBar，用于二次封装标签头	(props: DefaultTabBarProps, DefaultTabBar: React.ComponentClass) => React.ReactElement	-
size	大小，提供 large middle 和 small 三种大小	string	middle
tabBarExtraContent	tab bar 上额外的元素	ReactNode | {left?: ReactNode, right?: ReactNode}	-
tabBarGutter	tabs 之间的间隙	number	-
tabBarStyle	tab bar 的样式对象	CSSProperties	-
tabPosition	页签位置，可选值有 top right bottom left	string	top
destroyInactiveTabPane	被隐藏时是否销毁 DOM 结构	boolean	false
type	页签的基本样式，可选 line、card editable-card 类型	string	line
onChange	切换面板的回调	(activeKey: string) => void	-
onEdit	新增和删除页签的回调，在 type="editable-card" 时有效	(action === 'add' ? event : targetKey, action) => void	-
onTabClick	tab 被点击的回调	(key: string, event: MouseEvent) => void	-
onTabScroll	tab 滚动时触发	({ direction: left | right | top | bottom }) => void	-

> *更多属性查看 rc-tabs tabs*

TabItemType

参数	说明	类型	默认值	版本
closeIcon	自定义关闭图标，在 `type="editable-card"` 时有效。5.7.0：设置为 `null` 或 `false` 时隐藏关闭按钮	ReactNode	-	
destroyInactiveTabPane	被隐藏时是否销毁 DOM 结构	boolean	false	5.11.0
disabled	禁用某一项	boolean	false	
forceRender	被隐藏时是否渲染 DOM 结构	boolean	false	
key	对应 activeKey	string	-	
label	选项卡头显示文字	ReactNode	-	
icon	选项卡头显示图标	ReactNode	-	5.12.0
children	选项卡头显示内容	ReactNode	-	
closable	是否显示选项卡的关闭按钮，在 `type="editable-card"` 时有效	boolean	true	

MoreProps

参数	说明	类型	默认值	版本
icon	自定义折叠图标	ReactNode	-	
DropdownProps				

主题变量（Design Token）