When To Use

Ant Design has 3 types of Tabs for different situations.

- Card Tabs: for managing too many closeable views.
- Normal Tabs: for functional aspects of a page.
- Radio.Button: for secondary tabs.

Examples

Basic

```
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;
```

Disabled

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

```tsx
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;
```

Centered

```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;
```

Icon

```
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;
```

Indicator

```
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' },
];

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;
```

Slide

```
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;
```

Extra content

```
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;
```

Size

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

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

```tsx
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;
```

Position

```
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;
```

Card type tab

```
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;
```

Add & close tab

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

```tsx
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;
```

Container of card type Tab

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;
```

Customized trigger of new tab

```
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;
```

Customized bar of tab

```
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;
```

Draggable Tabs

```
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;
```

Animated

Debug

```tsx
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;
```

Nest

Debug

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

const { Option } = Select;

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

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;
```

component 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

Common props ref：Common props

Tabs

Property	Description	Type	Default
activeKey	Current TabPane's key	string	-
addIcon	Customize add icon, only works with `type="editable-card"`	ReactNode	`<PlusOutlined /`
animated	Whether to change tabs with animation.	boolean | { inkBar: boolean, tabPane: boolean }	{ inkBar: true, tabPane: false }
centered	Centers tabs	boolean	false
defaultActiveKey	Initial active TabPane's key, if `activeKey` is not set	string	`The key of firs tab`
hideAdd	Hide plus icon or not. Only works while `type="editable-card"`	boolean	false
indicator	Customize `size` and `align` of indicator	{ size?: number | (origin: number) => number; align: `start` | `center` | end; }	-
items	Configure tab content	TabItemType	[]
more	Customize the collapse menu	MoreProps	{ icon: `<EllipsisOutlin />` , trigger: 'hover

removeIcon	The custom icon of remove, only works with `type="editable-card"`	ReactNode	`<CloseOutlined`
popupClassName	`className` for more dropdown.	string	-
renderTabBar	Replace the TabBar	(props: DefaultTabBarProps, DefaultTabBar: React.ComponentClass) => React.ReactElement	-
size	Preset tab bar size	`large` | `middle` | `small`	`middle`
tabBarExtraContent	Extra content in tab bar	ReactNode | {left?: ReactNode, right?: ReactNode}	-
tabBarGutter	The gap between tabs	number	-
tabBarStyle	Tab bar style object	CSSProperties	-
tabPosition	Position of tabs	`top` | `right` | `bottom` | `left`	`top`
destroyInactiveTabPane	Whether destroy inactive TabPane when change tab	boolean	false
type	Basic style of tabs	`line` | `card` | `editable-card`	`line`
onChange	Callback executed when active tab is changed	(activeKey: string) => void	-
onEdit	Callback executed when tab is added or removed. Only works while `type="editable-card"`	(action === 'add' ? event : targetKey, action) => void	-
onTabClick	Callback executed when tab is	(key: string, event: MouseEvent) => void	-

	clicked		
onTabScroll	Trigger when tab scroll	({ direction: `left` | `right` | `top` | `bottom` }) => void	-

More option at [rc-tabs tabs]

TabItemType

Property	Description	Type	Default	Version
closeIcon	Customize close icon in TabPane's head. Only works while `type="editable-card"`. 5.7.0: close button will be hidden when setting to `null` or `false`	ReactNode	-	
destroyInactiveTabPane	Whether destroy inactive TabPane when change tab	boolean	false	5.11.0
disabled	Set TabPane disabled	boolean	false	
forceRender	Forced render of content in tabs, not lazy render after clicking on tabs	boolean	false	
key	TabPane's key	string	-	
label	TabPane's head display text	ReactNode	-	
icon	TabPane's head display icon	ReactNode	-	5.12.0
children	TabPane's head display content	ReactNode	-	
closable	Whether a close (x) button is visible, Only works while `type="editable-card"`	boolean	true	

MoreProps

Property	Description	Type	Default	Version
icon	The custom icon	ReactNode	-	
DropdownProps				

Design Token