何时使用 {#when-to-use}

- 在一组可选项中进行多项选择时；
- 单独使用可以表示两种状态之间的切换，和 `switch` 类似。区别在于切换 `switch` 会直接触发状态改变，而 `checkbox` 一般用于状态标记，需要和提交操作配合。

代码演示

基本用法

```
import React from 'react';
import { Checkbox } from 'antd';
import type { CheckboxProps } from 'antd';

const onChange: CheckboxProps['onChange'] = (e) => {
  console.log(`checked = ${e.target.checked}`);
};

const App: React.FC = () => <Checkbox onChange=
{onChange}>Checkbox</Checkbox>;

export default App;
```

不可用

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

const App: React.FC = () => (
  <>
    <Checkbox defaultChecked={false} disabled />
    <br />
    <Checkbox indeterminate disabled />
    <br />
    <Checkbox defaultChecked disabled />
  </>
);

export default App;
```

受控的 Checkbox

```
import React, { useState } from 'react';
import { Button, Checkbox } from 'antd';
import type { CheckboxProps } from 'antd';
```

```
const App: React.FC = () => {
  const [checked, setChecked] = useState(true);
  const [disabled, setDisabled] = useState(false);

  const toggleChecked = () => {
    setChecked(!checked);
  };

  const toggleDisable = () => {
    setDisabled(!disabled);
  };

  const onChange: CheckboxProps['onChange'] = (e) => {
    console.log('checked = ', e.target.checked);
    setChecked(e.target.checked);
  };

  const label = `${checked ? 'Checked' : 'Unchecked'}-${disabled ?
'Disabled' : 'Enabled'}`;

  return (
    <>
      <p style={{ marginBottom: '20px' }}>
        <Checkbox checked={checked} disabled={disabled} onChange=
{onChange}>
          {label}
        </Checkbox>
      </p>
      <p>
        <Button type="primary" size="small" onClick={toggleChecked}>
          {!checked ? 'Check' : 'Uncheck'}
        </Button>
        <Button style={{ margin: '0 10px' }} type="primary" size="small"
onClick={toggleDisable}>
          {!disabled ? 'Disable' : 'Enable'}
        </Button>
      </p>
    </>
  );
};

export default App;
```

Checkbox 组

```tsx
import React from 'react';
import { Checkbox } from 'antd';
import type { GetProp } from 'antd';

const onChange: GetProp<typeof Checkbox.Group, 'onChange'> =
(checkedValues) => {
  console.log('checked = ', checkedValues);
};

const plainOptions = ['Apple', 'Pear', 'Orange'];

const options = [
  { label: 'Apple', value: 'Apple' },
  { label: 'Pear', value: 'Pear' },
  { label: 'Orange', value: 'Orange' },
];

const optionsWithDisabled = [
  { label: 'Apple', value: 'Apple' },
  { label: 'Pear', value: 'Pear' },
  { label: 'Orange', value: 'Orange', disabled: false },
];

const App: React.FC = () => (
  <>
    <Checkbox.Group options={plainOptions} defaultValue={['Apple']}
onChange={onChange} />
    <br />
    <br />
    <Checkbox.Group options={options} defaultValue={['Pear']} onChange=
{onChange} />
    <br />
    <br />
    <Checkbox.Group
      options={optionsWithDisabled}
      disabled
      defaultValue={['Apple']}
      onChange={onChange}
    />
  </>
);

export default App;
```

全选

```tsx
import React, { useState } from 'react';
import { Checkbox, Divider } from 'antd';
import type { CheckboxProps } from 'antd';

const CheckboxGroup = Checkbox.Group;

const plainOptions = ['Apple', 'Pear', 'Orange'];
const defaultCheckedList = ['Apple', 'Orange'];

const App: React.FC = () => {
  const [checkedList, setCheckedList] = useState<string[]>
(defaultCheckedList);

  const checkAll = plainOptions.length === checkedList.length;
  const indeterminate = checkedList.length > 0 && checkedList.length <
plainOptions.length;

  const onChange = (list: string[]) => {
    setCheckedList(list);
  };

  const onCheckAllChange: CheckboxProps['onChange'] = (e) => {
    setCheckedList(e.target.checked ? plainOptions : []);
  };

  return (
    <>
      <Checkbox indeterminate={indeterminate} onChange={onCheckAllChange}
checked={checkAll}>
        Check all
      </Checkbox>
      <Divider />
      <CheckboxGroup options={plainOptions} value={checkedList} onChange=
{onChange} />
    </>
  );
};

export default App;
```

布局

```tsx
import React from 'react';
import { Checkbox, Col, Row } from 'antd';
import type { GetProp } from 'antd';
```

```tsx
const onChange: GetProp<typeof Checkbox.Group, 'onChange'> =
(checkedValues) => {
  console.log('checked = ', checkedValues);
};

const App: React.FC = () => (
  <Checkbox.Group style={{ width: '100%' }} onChange={onChange}>
    <Row>
      <Col span={8}>
        <Checkbox value="A">A</Checkbox>
      </Col>
      <Col span={8}>
        <Checkbox value="B">B</Checkbox>
      </Col>
      <Col span={8}>
        <Checkbox value="C">C</Checkbox>
      </Col>
      <Col span={8}>
        <Checkbox value="D">D</Checkbox>
      </Col>
      <Col span={8}>
        <Checkbox value="E">E</Checkbox>
      </Col>
    </Row>
  </Checkbox.Group>
);

export default App;
```

同行布局

Debug

```tsx
import React from 'react';
import { Checkbox, ConfigProvider, Radio, Space } from 'antd';

const sharedStyle: React.CSSProperties = {
  border: '1px solid red',
  marginBottom: 16,
};

const App: React.FC = () => (
  <div>
    <Space style={sharedStyle} align="center">
      <Checkbox value="light" />
```

```
  <div>Bamboo</div>
  <Checkbox value="little">Little</Checkbox>
</Space>

<Space style={sharedStyle} align="center">
  <Radio value="light" />
  <div>Bamboo</div>
  <Radio value="little">Little</Radio>
</Space>

<div
  style={{
    ...sharedStyle,
    display: 'flex',
    alignItems: 'center',
  }}
>
  <Checkbox value="light" />
  <div>Bamboo</div>
  <Checkbox value="little">Little</Checkbox>
</div>

<div
  style={{
    ...sharedStyle,
    display: 'flex',
    alignItems: 'center',
  }}
>
  <Radio value="light" />
  <div>Bamboo</div>
  <Radio value="little">Little</Radio>
</div>

<div>
  <ConfigProvider
    theme={{
      token: {
        controlHeight: 48,
      },
    }}
  >
    <Checkbox>Aligned</Checkbox>
  </ConfigProvider>
</div>
```

```
    <div>
      <Checkbox>
        <span style={{ fontSize: 32 }}>Aligned</span>
      </Checkbox>
    </div>
  </div>
);

export default App;
```

禁用下的 Tooltip

Debug

```
import React from 'react';
import { Checkbox, Popover } from 'antd';

const App: React.FC = () => (
  <div style={{ padding: 56 }}>
    <Popover content="xxxx" trigger="hover">
      <Checkbox disabled checked />
    </Popover>
  </div>
);

export default App;
```

自定义 lineWidth

Debug

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

const App: React.FC = () => (
  <>
    <ConfigProvider
      theme={{
        components: {
          Checkbox: {
            lineWidth: 6,
          },
        },
      }}
    >
      <Checkbox checked />
```

```
      <Checkbox />
    </ConfigProvider>
    <Checkbox checked />
    <Checkbox />
  </>
);

export default App;
```

API

通用属性参考：通用属性

Checkbox

参数	说明	类型	默认值	版本
autoFocus	自动获取焦点	boolean	false	
checked	指定当前是否选中	boolean	false	
defaultChecked	初始是否选中	boolean	false	
disabled	失效状态	boolean	false	
indeterminate	设置 indeterminate 状态，只负责样式控制	boolean	false	
onChange	变化时的回调函数	(e: CheckboxChangeEvent) => void	-	
onBlur	失去焦点时的回调	function()	-	
onFocus	获得焦点时的回调	function()	-	

Checkbox Group

参数	说明	类型	默认值	版本
defaultValue	默认选中的选项	(string | number)[]	[]	
disabled	整组失效	boolean	false	
name	CheckboxGroup 下所有 `input[type="checkbox"]` 的 name 属性	string	-	
options	指定可选项	string[] | number[] | Option[]	[]	

value	指定选中的选项	(string | number | boolean)[]	[]	
onChange	变化时的回调函数	(checkedValue: T[]) => void	-	

Option

```
interface Option {
  label: string;
  value: string;
  disabled?: boolean;
}
```

方法

Checkbox

名称	描述	版本
blur()	移除焦点	
focus()	获取焦点	
nativeElement	返回 Checkbox 的 DOM 节点	5.17.3

主题变量（Design Token）

FAQ

为什么在 Form.Item 下不能绑定数据?

Form.Item 默认绑定值属性到 `value` 上，而 Checkbox 的值属性为 `checked` 。你可以通过 `valuePropName` 来修改绑定的值属性。

```
<Form.Item name="fieldA" valuePropName="checked">
  <Checkbox />
</Form.Item>
```