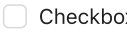
Checkbox

Checkbox component.

When To Use

- Used for selecting multiple values from several options.
- If you use only one checkbox, it is the same as using Switch to toggle between two states. The difference is that Switch will trigger the state change directly, but Checkbox just marks the state as changed and this needs to be submitted.

Examples



Basic

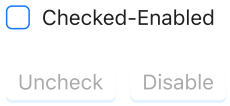
Basic usage of checkbox.

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

const onChange = (e: CheckboxChangeEvent) => {
  console.log(`checked = ${e.target.checked}`);
};

const App: React.FC = () => <Checkbox onChange

export default App;
```


Disabled

Disabled checkbox.

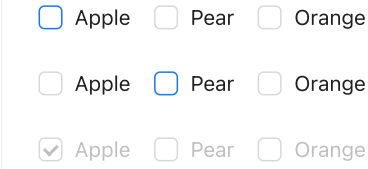
```
import React from 'react';
import { Checkbox } from 'antd';

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

export default App;
```



Controlled Checkbox

Communicated with other components.

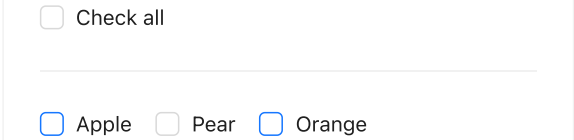
```
import React, { useState } from 'react';
import { Button, Checkbox } from 'antd';
import type { CheckboxChangeEvent } from 'antd

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

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

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

  const onChange = (e: CheckboxChangeEvent) =>
    console.log('checked = ', e.target.checked
    setChecked(e.target.checked);
  };

  const label = `${checked ? 'Checked' : 'Unch

  return (
    <>
      <p style={{ marginBottom: '20px' }}>
        <Checkbox checked={checked} disabled={
          {label}
        </Checkbox>
      </p>
      <p>
        <Button type="primary" size="small" or
          {!checked ? 'Check' : 'Uncheck'}
        </Button>
        <Button style={{ margin: '0 10px' }} t
          {!disabled ? 'Disable' : 'Enable'}
        </Button>
      </p>
    </>
  );
};

export default App;
```



Checkbox Group

Generate a group of checkboxes from an array.

```
import { Checkbox } from 'antd';
import type { CheckboxValueType } from 'antd/e
import React from 'react';

const onChange = (checkedValues: CheckboxValue
  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', disablec
];

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

export default App;
```



Check all

The `indeterminate` property can help you to achieve a 'check all' effect.

```
import React, { useState } from 'react';
import { Checkbox, Divider } from 'antd';
import type { CheckboxChangeEvent } from 'antd
import type { CheckboxValueType } from 'antd/e

const CheckboxGroup = Checkbox.Group;

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

const App: React.FC = () => {
  const [checkedList, setCheckedList] = useSta
  const [indeterminate, setIndeterminate] = us
  const [checkAll, setCheckAll] = useState(fal

  const onChange = (list: CheckboxValueType[])
    setCheckedList(list);
    setIndeterminate(!!list.length && list.ler
    setCheckAll(list.length === plainOptions.l
  };

  const onCheckAllChange = (e: CheckboxChangeE
    setCheckedList(e.target.checked ? plainOpt
    setIndeterminate(false);
    setCheckAll(e.target.checked);
  };

  return (
    <>
      <Checkbox indeterminate={indeterminate}
        Check all
      </Checkbox>
      <Divider />
      <CheckboxGroup options={plainOptions} va
    </>
  );
};

export default App;
```


Use with Grid

We can use Checkbox and Grid in Checkbox.Group, to implement complex layout.

```
import React from 'react';
import { Checkbox, Col, Row } from 'antd';
import type { CheckboxValueType } from 'antd/e

const onChange = (checkedValues: CheckboxValue
  console.log('checked = ', checkedValues);
};

const App: React.FC = () => (
  <Checkbox.Group style={{ width: '100%' }} or
    <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;
```

API

Checkbox

Property	Description	Type	Default	Version
autoFocus	If get focus when component mounted	boolean	false	
checked	Specifies whether the checkbox is selected	boolean	false	

Property	Description	Type	Default	Version
defaultChecked	Specifies the initial state: whether or not the checkbox is selected	boolean	false	
disabled	If disable checkbox	boolean	false	
indeterminate	The indeterminate checked state of checkbox	boolean	false	
onChange	The callback function that is triggered when the state changes	(e: CheckboxChangeEvent) => void	-	

Checkbox Group

Property	Description	Type	Default	Version
defaultValue	Default selected value	(string | number)[]	[]	
disabled	If disable all checkboxes	boolean	false	
name	The `name` property of all `input[type="checkbox"]` children	string	-	
options	Specifies options	string[] | number[] | Option[]	[]	
value	Used for setting the currently selected value	(string | number | boolean)[]	[]	
onChange	The callback function that is triggered when the state changes	(checkedValue: CheckboxValueType[]) => void	-	

Option

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

Methods

Checkbox

Name	Description	Version
`blur()`	Remove focus	
`focus()`	Get focus	

Design Token

▾ Global Token

Token Name	Description	Type	Default Value
colorBgContainer	Container background color, e.g: default button, input box, etc. Be sure not to confuse this with `colorBgElevated`.	`string`	`#ffffff`
colorBgContainerDisabled	Control the background color of container in disabled state.	`string`	`rgba(0, 0, 0, 0.04)`
colorBorder	Default border color, used to separate different elements, such as: form separator, card separator, etc.	`string`	`#d9d9d9`
colorPrimary	Brand color is one of the most direct visual elements to reflect the characteristics and communication of the product. After you have selected the brand color, we will automatically generate a complete color palette and assign it effective design semantics.	`string`	`#1677ff`
colorPrimaryBorder	The stroke color under the main color gradient, used on the stroke of components such as Slider.	`string`	`#91caff`
colorPrimaryHover	Hover state under the main color gradient.	`string`	`#4096ff`
colorText	Default text color which comply with W3C standards, and this color is also the darkest neutral color.	`string`	`rgba(0, 0, 0, 0.88)`
colorTextDisabled	Control the color of text in disabled state.	`string`	`rgba(0, 0, 0, 0.25)`
colorWhite	Pure white color don't changed by theme	`string`	`#fff`
borderRadiusSM	SM size border radius, used in small size components, such as Button, Input, Select and other input components in small size	`number`	4
controlInteractiveSize	Control the interactive size of control component.	`number`	16
fontFamily	The font family of Ant Design prioritizes the default interface font of the system, and provides a set of alternative font libraries that are suitable for screen display to maintain the readability and readability of the font under different	`string`	-apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto, 'Helvetica Neue', Arial, 'Noto Sans', sans-serif, 'Apple Color Emoji', 'Segoe UI Emoji',

Token Name	Description	Type	Default Value
	platforms and browsers, reflecting the friendly, stable and professional characteristics.		'Segoe UI Symbol', 'Noto Color Emoji'
fontSize	The most widely used font size in the design system, from which the text gradient will be derived.	`number`	14
fontSizeLG	Large font size	`number`	16
lineHeight	Line height of text.	`number`	1.5714285714285714
lineType	Border style of base components	`string`	solid
lineWidth	Border width of base components	`number`	1
lineWidthBold	The default line width of the outline class components, such as Button, Input, Select, etc.	`number`	2
lineWidthFocus	Control the width of the line when the component is in focus state.	`number`	4
marginXS	Control the margin of an element, with a small size.	`number`	8
motionDurationFast	Motion speed, fast speed. Used for small element animation interaction.	`string`	0.1s
motionDurationMid	Motion speed, medium speed. Used for medium element animation interaction.	`string`	0.2s
motionDurationSlow	Motion speed, slow speed. Used for large element animation interaction.	`string`	0.3s
motionEaseInBack	Preset motion curve.	`string`	cubic-bezier(0.71, -0.46, 0.88, 0.6)
motionEaseOutBack	Preset motion curve.	`string`	cubic-bezier(0.12, 0.4, 0.29, 1.46)
paddingXS	Control the extra small padding of the element.	`number`	8