

何时使用 {#when-to-use}

当用户需要自定义颜色选择的时候使用。

代码演示

基本使用

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

const Demo = () => <ColorPicker defaultValue="#1677ff" />;

export default Demo;
```

触发器尺寸大小

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

const Demo = () => (
  <Space>
    <Space direction="vertical">
      <ColorPicker defaultValue="#1677ff" size="small" />
      <ColorPicker defaultValue="#1677ff" />
      <ColorPicker defaultValue="#1677ff" size="large" />
    </Space>
    <Space direction="vertical">
      <ColorPicker defaultValue="#1677ff" size="small" showText />
      <ColorPicker defaultValue="#1677ff" showText />
      <ColorPicker defaultValue="#1677ff" size="large" showText />
    </Space>
  </Space>
);

export default Demo;
```

受控模式

```
import React, { useState } from 'react';
import { ColorPicker, Space } from 'antd';
import type { ColorPickerProps, GetProp } from 'antd';

type Color = GetProp<ColorPickerProps, 'value'>;
```

```

const Demo: React.FC = () => {
  const [color, setColor] = useState<Color>('#1677ff');

  return (
    <Space>
      <ColorPicker value={color} onChange={setColor} />
      <ColorPicker value={color} onChangeComplete={setColor} />
    </Space>
  );
};

export default Demo;

```

渐变色

v5.20.0

```

import React from 'react';
import { ColorPicker, Space } from 'antd';

const DEFAULT_COLOR = [
  {
    color: 'rgb(16, 142, 233)',
    percent: 0,
  },
  {
    color: 'rgb(135, 208, 104)',
    percent: 100,
  },
];

const Demo = () => (
  <Space direction="vertical">
    <ColorPicker
      defaultValue={DEFAULT_COLOR}
      allowClear
      showText
      mode={['single', 'gradient']}
      onChangeComplete={(color) => {
        console.log(color.toCssString());
      }}
    />
    <ColorPicker
      defaultValue={DEFAULT_COLOR}
      allowClear
      showText

```

```

        mode="gradient"
        onChangeComplete={(color) => {
            console.log(color.toCssString());
        }}
    />
</Space>
);

export default Demo;

```

渲染触发器文本

```

import React, { useState } from 'react';
import { DownOutlined } from '@ant-design/icons';
import { ColorPicker, Space } from 'antd';

const Demo = () => {
    const [open, setOpen] = useState(false);
    return (
        <Space direction="vertical">
            <ColorPicker defaultValue="#1677ff" showText allowClear />
            <ColorPicker
                defaultValue="#1677ff"
                showText={(color) => <span>Custom Text ({color.toHexString()})
            </span>}
            />
            <ColorPicker
                defaultValue="#1677ff"
                open={open}
                onOpenChange={setOpen}
                showText={() => (
                    <DownOutlined
                        rotate={open ? 180 : 0}
                        style={{
                            color: 'rgba(0, 0, 0, 0.25)',
                        }}
                    />
                )}
            />
        </Space>
    );
};

export default Demo;

```

禁用

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

export default () => <ColorPicker defaultValue="#1677ff" showText disabled />;
```

禁用透明度

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

const Demo = () => <ColorPicker defaultValue="#1677ff" disabledAlpha />;

export default Demo;
```

清除颜色

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

export default () => {
  const [color, setColor] = React.useState<string>('#1677ff');

  return (
    <ColorPicker
      value={color}
      allowClear
      onChange={(c) => {
        setColor(c.toHexString());
      }}
    />
  );
};
```

自定义触发器

```
import React, { useMemo, useState } from 'react';
import { Button, ColorPicker } from 'antd';
import type { ColorPickerProps, GetProp } from 'antd';

type Color = Extract<GetProp<ColorPickerProps, 'value'>, string> | {
```

```

cleared: any }>;

const Demo: React.FC = () => {
  const [color, setColor] = useState<Color>('#1677ff');

  const bgColor = useMemo<string>(
    () => (typeof color === 'string' ? color : color!.toHexString()),
    [color],
  );

  const btnStyle: React.CSSProperties = {
    backgroundColor: bgColor,
  };

  return (
    <ColorPicker value={color} onChange={setColor}>
      <Button type="primary" style={btnStyle}>
        open
      </Button>
    </ColorPicker>
  );
};

export default Demo;

```

自定义触发事件

```

import React from 'react';
import { ColorPicker } from 'antd';

const Demo = () => <ColorPicker defaultValue="#1677ff" trigger="hover" />;

export default Demo;

```

颜色编码

```

import React, { useState } from 'react';
import { ColorPicker, Space } from 'antd';
import type { ColorPickerProps, GetProp } from 'antd';

type Color = Extract<GetProp<ColorPickerProps, 'value'>, string | {
  cleared: any }>;
type Format = GetProp<ColorPickerProps, 'format'>;

const HexCase: React.FC = () => {

```

```

const [colorHex, setColorHex] = useState<Color>('#1677ff');
const [formatHex, setFormatHex] = useState<Format | undefined>('hex');

const hexString = React.useMemo<string>(
  () => (typeof colorHex === 'string' ? colorHex :
colorHex?.toHexString()),
  [colorHex],
);

return (
  <Space>
    <ColorPicker
      format={formatHex}
      value={colorHex}
      onChange={setColorHex}
      onFormatChange={setFormatHex}
    />
    <span>HEX: {hexString}</span>
  </Space>
);
};

const HsbCase: React.FC = () => {
  const [colorHsb, setColorHsb] = useState<Color>('hsb(215, 91%, 100%)');
  const [formatHsb, setFormatHsb] = useState<ColorPickerProps['format']>
('hsb');

  const hsbString = React.useMemo(
    () => (typeof colorHsb === 'string' ? colorHsb :
colorHsb?.toHsbString()),
    [colorHsb],
  );

  return (
    <Space>
      <ColorPicker
        format={formatHsb}
        value={colorHsb}
        onChange={setColorHsb}
        onFormatChange={setFormatHsb}
      />
      <span>HSB: {hsbString}</span>
    </Space>
  );
};

```

```

const RgbCase: React.FC = () => {
  const [colorRgb, setColorRgb] = useState<Color>('rgb(22, 119, 255)');
  const [formatRgb, setFormatRgb] = useState<ColorPickerProps['format']>
('rgb');

  const rgbString = React.useMemo(
    () => (typeof colorRgb === 'string' ? colorRgb :
colorRgb?.toRgbString()),
    [colorRgb],
  );

  return (
    <Space>
      <ColorPicker
        format={formatRgb}
        value={colorRgb}
        onChange={setColorRgb}
        onFormatChange={setFormatRgb}
      />
      <span>RGB: {rgbString}</span>
    </Space>
  );
};

const Demo: React.FC = () => (
  <Space direction="vertical" size="middle" style={{ display: 'flex' }}>
    <HexCase />
    <HsbCase />
    <RgbCase />
  </Space>
);

export default Demo;

```

预设颜色

```

import React from 'react';
import { generate, green, presetPalettes, red } from '@ant-design/colors';
import { ColorPicker, theme } from 'antd';
import type { ColorPickerProps } from 'antd';

type Presets = Required<ColorPickerProps>['presets'][number];

function genPresets(presets = presetPalettes) {
  return Object.entries(presets).map<Presets>(([label, colors]) => ({
    label, colors, key: label }));
}

```

```

}

const Demo: React.FC = () => {
  const { token } = theme.useToken();
  const presets = genPresets({ primary: generate(token.colorPrimary), red,
green });
  return <ColorPicker presets={presets} defaultValue="#1677ff" />;
};

export default Demo;

```

自定义面板

```

import React from 'react';
import { cyan, generate, green, presetPalettes, red } from '@ant-
design/colors';
import { Col, ColorPicker, Divider, Row, Space, theme } from 'antd';
import type { ColorPickerProps } from 'antd';

type Presets = Required<ColorPickerProps>['presets'][number];

function genPresets(presets = presetPalettes) {
  return Object.entries(presets).map<Presets>(([label, colors]) => ({
label, colors, key: label }));
}

const HorizontalLayoutDemo = () => {
  const { token } = theme.useToken();

  const presets = genPresets({
    primary: generate(token.colorPrimary),
    red,
    green,
    cyan,
  });

  const customPanelRender: ColorPickerProps['panelRender'] = (
    _ ,
    { components: { Picker, Presets } },
  ) => (
    <Row justify="space-between" wrap={false}>
      <Col span={12}>
        <Presets />
      </Col>
      <Divider type="vertical" style={{ height: 'auto' }} />
      <Col flex="auto">

```



```

        <Picker />
      </Col>
    </Row>
  );

  return (
    <ColorPicker
      defaultValue={token.colorPrimary}
      styles={{ popupOverlayInner: { width: 480 } }}
      presets={presets}
      panelRender={customPanelRender}
    />
  );
};

```

```

const BasicDemo = () => (
  <ColorPicker
    defaultValue="#1677ff"
    panelRender={(panel) => (
      <div className="custom-panel">
        <div
          style={{
            fontSize: 12,
            color: 'rgba(0, 0, 0, 0.88)',
            lineHeight: '20px',
            marginBottom: 8,
          }}
        >
          Color Picker
        </div>
        {panel}
      </div>
    )}
  />
);

```

```

export default () => (
  <Space direction="vertical">
    <Space>
      <span>Add title:</span>
      <BasicDemo />
    </Space>
    <Space>
      <span>Horizontal layout:</span>
      <HorizontalLayoutDemo />
    </Space>
  </Space>
);

```

```
    </Space>
  );
```

Pure Render

Debug

```
import React, { useState } from 'react';
import { ColorPicker } from 'antd';
import type { ColorPickerProps, GetProp } from 'antd';

const { _InternalPanelDoNotUseOrYouWillBeFired: PureRenderColorPicker } =
ColorPicker;

type Color = GetProp<ColorPickerProps, 'value'>;

const Demo: React.FC = () => {
  const [color, setColor] = useState<Color>('#1677ff');
  return (
    <div style={{ paddingInlineStart: 100 }}>
      <PureRenderColorPicker value={color} onChange={setColor} />
    </div>
  );
};

export default Demo;
```

API

通用属性参考：[通用属性](#)

自 `antd@5.5.0` 版本开始提供该组件。

参数	说明	类型	默认值	版本
<code>allowClear</code>	允许清除选择的颜色	<code>boolean</code>	<code>false</code>	
<code>arrow</code>	配置弹出的箭头	<code>boolean</code> { <code>pointAtCenter</code> : <code>boolean</code> }	<code>true</code>	
<code>children</code>	颜色选择器的触发器	<code>React.ReactNode</code>	-	
<code>defaultValue</code>	颜色默认的值	<code>string</code> <code>Color</code>	-	
<code>defaultFormat</code>	颜色格式默认的值	<code>rgb</code> <code>hex</code> <code>hsb</code>	<code>hex</code>	5.9.0
<code>disabled</code>	禁用颜色选择器	<code>boolean</code>	-	

disabledAlpha	禁用透明度	boolean	-	5.8.0
disabledFormat	禁用选择颜色格式	boolean	-	
destroyTooltipOnHide	关闭后是否销毁弹窗	boolean	false	5.7.0
format	颜色格式	rgb hex hsb	-	
mode	选择器模式，用于配置单色与渐变	'single' 'gradient' ('single' 'gradient')[]	single	5.20.0
open	是否显示弹出窗口	boolean	-	
presets	预设的颜色	{ label: ReactNode, colors: Array<string Color>, defaultOpen?: boolean, key?: React.Key }[]	-	default 5.11.0 5.23.0
placement	弹出窗口的位置	同 Tooltips 组件的 placement 参数设计	bottomLeft	
panelRender	自定义渲染面板	(panel: React.ReactNode, extra: { components: { Picker: FC; Presets: FC } }) => React.ReactNode	-	5.7.0
showText	显示颜色文本	boolean (color: Color) => React.ReactNode	-	5.7.0
size	设置触发器大小	large middle small	middle	5.7.0
trigger	颜色选择器的触发模式	hover click	click	
value	颜色的值	string Color	-	

onChange	颜色变化的回调	(value: Color, css: string) => void	-	
onChangeComplete	颜色选择完成的回调，通过 onChangeComplete 对 value 受控时拖拽不会改变展示颜色	(value: Color) => void	-	5.7.0
onFormatChange	颜色格式变化的回调	(format: 'hex' 'rgb' 'hsb') => void	-	
onOpenChange	当 open 被改变时的回调	(open: boolean) => void	-	
onClear	清除的回调	() => void	-	5.6.0

Color

参数	说明	类型	版本
toCssString	转换成 CSS 支持的格式	() => string	5.20.0
toHex	转换成 hex 格式字符，返回格式如：1677ff	() => string	-
toHexString	转换成 hex 格式颜色字符串，返回格式如：#1677ff	() => string	-
toHsb	转换成 hsb 对象	() => ({ h: number, s: number, b: number, a: number })	-
toHsbString	转换成 hsb 格式颜色字符串，返回格式如：hsb(215, 91%, 100%)	() => string	-
toRgb	转换成 rgb 对象	() => ({ r: number, g: number, b: number, a: number })	-
toRgbString	转换成 rgb 格式颜色字符串，返回格式如：rgb(22, 119, 255)	() => string	-

FAQ

关于颜色赋值的问题

颜色选择器的值同时支持字符串色值和选择器生成的 `Color` 对象，但由于不同格式的颜色字符串互相转换会有精度误差问题，所以受控场景推荐使用选择器生成的 `Color` 对象来进行赋值操作，这样可以避免精度问题，保证取值是精准的，选择器也可以按照预期工作。