When To Use

Used when the text needs to be converted into a QR Code.

Examples

base

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

const App: React.FC = () => {
  const [text, setText] = React.useState('https://ant.design/');

  return (
    <Space direction="vertical" align="center">
      <QRCode value={text || '-'} />
      <Input
        placeholder="-"
        maxLength={60}
        value={text}
        onChange={(e) => setText(e.target.value)}
      />
    </Space>
  );
};

export default App;
```

With Icon

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

const App: React.FC = () => (
  <QRCode
    errorLevel="H"
    value="https://ant.design/"

icon="https://gw.alipayobjects.com/zos/rmsportal/KDpgvguMpGfqaHPjicRK.svg"
  />
);

export default App;
```

other status

```
import React from 'react';
import { Flex, QRCode } from 'antd';

const value = 'https://ant.design';

const App: React.FC = () => (
  <Flex gap="middle" wrap>
    <QRCode value={value} status="loading" />
    <QRCode value={value} status="expired" onRefresh={() =>
console.log('refresh')} />
    <QRCode value={value} status="scanned" />
  </Flex>
);

export default App;
```

custom status render

v5.20.0

```
import React from 'react';
import { CheckCircleFilled, CloseCircleFilled, ReloadOutlined } from '@ant-
design/icons';
import type { QRCodeProps } from 'antd';
import { Button, Flex, QRCode, Space, Spin } from 'antd';

const value = 'https://ant.design';

const customStatusRender: QRCodeProps['statusRender'] = (info) => {
  switch (info.status) {
    case 'expired':
      return (
        <div>
          <CloseCircleFilled style={{ color: 'red' }} />
{info.locale?.expired}
          <p>
            <Button type="link" onClick={info.onRefresh}>
              <ReloadOutlined /> {info.locale?.refresh}
            </Button>
          </p>
        </div>
      );
    case 'loading':
      return (
        <Space direction="vertical">
          <Spin />
          <p>Loading...</p>
        </Space>
      );
    case 'scanned':
      return (
        <div>
          <CheckCircleFilled style={{ color: 'green' }} />
{info.locale?.scanned}
        </div>
      );
    default:
      return null;
  }
};

const App: React.FC = () => (
  <Flex gap="middle" wrap>
    <QRCode value={value} status="loading" statusRender=
{customStatusRender} />
    <QRCode
      value={value}
      status="expired"
      onRefresh={() => console.log('refresh')}
      statusRender={customStatusRender}
    />
    <QRCode value={value} status="scanned" statusRender=
{customStatusRender} />
  </Flex>
);

export default App;
```

Custom Render Type

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

const App: React.FC = () => (
  <Space>
    <QRCode type="canvas" value="https://ant.design/" />
    <QRCode type="svg" value="https://ant.design/" />
  </Space>
);
```

```
export default App;
```

Custom Size

```
import React, { useState } from 'react';
import { MinusOutlined, PlusOutlined } from '@ant-design/icons';
import { Button, QRCode, Space } from 'antd';

const MIN_SIZE = 48;
const MAX_SIZE = 300;

const App: React.FC = () => {
  const [size, setSize] = useState<number>(160);

  const increase = () => {
    setSize((prevSize) => {
      const newSize = prevSize + 10;
      if (newSize >= MAX_SIZE) {
        return MAX_SIZE;
      }
      return newSize;
    });
  };

  const decline = () => {
    setSize((prevSize) => {
      const newSize = prevSize - 10;
      if (newSize <= MIN_SIZE) {
        return MIN_SIZE;
      }
      return newSize;
    });
  };

  return (
    <>
      <Space.Compact style={{ marginBottom: 16 }}>
        <Button onClick={decline} disabled={size <= MIN_SIZE} icon=
{<MinusOutlined />}>
          Smaller
        </Button>
        <Button onClick={increase} disabled={size >= MAX_SIZE} icon=
{<PlusOutlined />}>
          Larger
        </Button>
      </Space.Compact>
      <QRCode
        errorLevel="H"
        size={size}
        iconSize={size / 4}
        value="https://ant.design/"

icon="https://gw.alipayobjects.com/zos/rmsportal/KDpgvguMpGfqaHPjicRK.svg"
      />
    </>
  );
};

export default App;
```

Custom Color

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

const { useToken } = theme;

const App: React.FC = () => {
  const { token } = useToken();
  return (
    <Space>
      <QRCode value="https://ant.design/" color={token.colorSuccessText} />
      <QRCode
        value="https://ant.design/"
        color={token.colorInfoText}
        bgColor={token.colorBgLayout}
      />
    </Space>
  );
};

export default App;
```

Download QRCode

```
import React from 'react';
import { Button, QRCode, Segmented, Space } from 'antd';
import type { QRCodeProps } from 'antd';

function doDownload(url: string, fileName: string) {
  const a = document.createElement('a');
  a.download = fileName;
  a.href = url;
  document.body.appendChild(a);
  a.click();
  document.body.removeChild(a);
}

const downloadCanvasQRCode = () => {
  const canvas =
document.getElementById('myqrcode')?.querySelector<HTMLCanvasElement>
('canvas');
  if (canvas) {
    const url = canvas.toDataURL();
    doDownload(url, 'QRCode.png');
  }
};

const downloadSvgQRCode = () => {
  const svg =
document.getElementById('myqrcode')?.querySelector<SVGElement>('svg');
  const svgData = new XMLSerializer().serializeToString(svg!);
  const blob = new Blob([svgData], { type: 'image/svg+xml;charset=utf-8'
});
  const url = URL.createObjectURL(blob);

  doDownload(url, 'QRCode.svg');
};

const App: React.FC = () => {
  const [renderType, setRenderType] = React.useState<QRCodeProps['type']>
('canvas');
  return (
    <Space id="myqrcode" direction="vertical">
      <Segmented options={['canvas', 'svg']} value={renderType} onChange=
{setRenderType} />
      <div>
        <QRCode
          type={renderType}
          value="https://ant.design/"
          bgColor="#fff"
          style={{ marginBottom: 16 }}

icon="https://gw.alipayobjects.com/zos/rmsportal/KDpgvguMpGfqaHPjicRK.svg"
        />
        <Button
```

```
          type="primary"
          onClick={renderType === 'canvas' ? downloadCanvasQRCode :
downloadSvgQRCode}
        >
          Download
        </Button>
      </div>
    </Space>
  );
};

export default App;
```

Error Level

```
import React, { useState } from 'react';
import type { QRCodeProps } from 'antd';
import { QRCode, Segmented } from 'antd';

const App: React.FC = () => {
  const [level, setLevel] = useState<QRCodeProps['errorLevel']>('L');
  return (
    <>
      <QRCode
        style={{ marginBottom: 16 }}
        errorLevel={level}

value="https://gw.alipayobjects.com/zos/rmsportal/KDpgvguMpGfqaHPjicRK.svg"
      />
      <Segmented options={['L', 'M', 'Q', 'H']} value={level} onChange=
{setLevel} />
    </>
  );
};

export default App;
```

Advanced Usage

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

const App: React.FC = () => (
  <Popover content={<QRCode value="https://ant.design" bordered={false}
/>}>
```

```
    <Button type="primary">Hover me</Button>
  </Popover>
);

export default App;
```

API

Common props ref: [Common props](#)

> *This component is available since `antd@5.1.0`*

Property	Description	Type	Default	Version
value	scanned text	string	-	
type	render type	`canvas | svg`	`canvas`	5.6.0
icon	include image url (only image link are supported)	string	-	
size	QRCode size	number	160	
iconSize	include image size	number | { width: number; height: number }	40	5.19.0
color	QRCode Color	string	`#000`	
bgColor	QRCode Background Color	string	`transparent`	5.5.0
bordered	Whether has border style	boolean	`true`	
errorLevel	Error Code Level	`'L' | 'M' | 'Q' | 'H'`	`M`	
status	QRCode status	`active | expired | loading | scanned`	`active`	scanned: 5.13.0
statusRender	custom status	`(info: \ [StatusRenderInfo](./en-`	5.20.0	

	render	US/qr-code-cn#statusrenderinfo.pdf)) => React.ReactNode		
onRefresh	callback	() => void	-	

StatusRenderInfo

```
type StatusRenderInfo = {
  status: QRStatus;
  locale: Locale['QRCode'];
  onRefresh?: () => void;
};
```

Design Token

FAQ

About QRCode ErrorLevel

The ErrorLevel means that the QR code can be scanned normally after being blocked, and the maximum area that can be blocked is the error correction rate.

Generally, the QR code is divided into 4 error correction levels: Level `L` can correct about `7%` errors, Level `M` can correct about `15%` errors, Level `Q` can correct about `25%` errors, and Level `H` can correct about `30%` errors. When the content encoding of the QR code carries less information, in other words, when the value link is short, set different error correction levels, and the generated image will not change.

> *For more information, see the: https://www.qrcode.com/en/about/error_correction*

⚠️ ⚠️ ⚠️ Cannot scan the QR code?

If the QR code cannot be scanned for identification, it may be because the link address is too long, which leads to too dense pixels.

You can configure the QR code to be larger through size, or shorten the link through short link services.