When To Use

- Use when the page needs to be watermarked to identify the copyright.
- Suitable for preventing information theft.

Examples

Basic

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

const App: React.FC = () => (
  <Watermark content="Ant Design">
    <div style={{ height: 500 }} />
  </Watermark>
);

export default App;
```

Multi-line watermark

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

const App: React.FC = () => (
  <Watermark content={['Ant Design', 'Happy Working']}>
    <div style={{ height: 500 }} />
  </Watermark>
);

export default App;
```

Image watermark

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

const App: React.FC = () => (
  <Watermark
    height={30}
    width={130}

image="https://mdn.alipayobjects.com/huamei_7uahnr/afts/img/A*lkAoRbywo0oAAAA
  >
```

```
    <div style={{ height: 500 }} />
  </Watermark>
);

export default App;
```

Custom configuration

```
import React, { useState } from 'react';
import { ColorPicker, Flex, Form, Input, InputNumber, Slider, Typography,
Watermark } from 'antd';
import type { ColorPickerProps, GetProp, WatermarkProps } from 'antd';

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

const { Paragraph } = Typography;

interface WatermarkConfig {
  content: string;
  color: string | Color;
  fontSize: number;
  zIndex: number;
  rotate: number;
  gap: [number, number];
  offset?: [number, number];
}

const App: React.FC = () => {
  const [form] = Form.useForm();
  const [config, setConfig] = useState<WatermarkConfig>({
    content: 'Ant Design',
    color: 'rgba(0, 0, 0, 0.15)',
    fontSize: 16,
    zIndex: 11,
    rotate: -22,
    gap: [100, 100],
    offset: undefined,
  });
  const { content, color, fontSize, zIndex, rotate, gap, offset } = config;

  const watermarkProps: WatermarkProps = {
    content,
    zIndex,
    rotate,
    gap,
    offset,
    font: { color: typeof color === 'string' ? color : color.toRgbString(),
fontSize },
  };

  return (
    <Flex gap="middle">
      <Watermark {...watermarkProps}>
        <Typography>
          <Paragraph>
            The light-speed iteration of the digital world makes products
more complex. However,
            human consciousness and attention resources are limited. Facing
this design
            contradiction, the pursuit of natural interaction will be the
consistent direction of
            Ant Design.
          </Paragraph>
          <Paragraph>
            Natural user cognition: According to cognitive psychology,
about 80% of external
            information is obtained through visual channels. The most
important visual elements in
            the interface design, including layout, colors, illustrations,
icons, etc., should fully
            absorb the laws of nature, thereby reducing the user&apos;s
cognitive cost and bringing
            authentic and smooth feelings. In some scenarios, opportunely
adding other sensory
            channels such as hearing, touch can create a richer and more
natural product experience.
          </Paragraph>
          <Paragraph>
            Natural user behavior: In the interaction with the system, the
designer should fully
            understand the relationship between users, system roles, and
task objectives, and also
            contextually organize system functions and services. At the
same time, a series of
            methods such as behavior analysis, artificial intelligence and
sensors could be applied
            to assist users to make effective decisions and reduce extra
operations of users, to
            save users&apos; mental and physical resources and make human-
computer interaction more
            natural.
          </Paragraph>
        </Typography>
        <img
          style={{ zIndex: 10, width: '100%', maxWidth: 800, position:
'relative' }}

src="https://gw.alipayobjects.com/mdn/rms_08e378/afts/img/A*zx7LTI_ECSAAAAAAA
          alt="img"
        />
      </Watermark>
      <Form
        style={{ width: 280, flexShrink: 0, borderLeft: '1px solid #eee',
paddingInlineStart: 16 }}
        form={form}
        layout="vertical"
        initialValues={config}
        onValuesChange={(_, values) => {
          setConfig(values);
        }}
      >
        <Form.Item name="content" label="Content">
          <Input placeholder="请输入" />
        </Form.Item>
        <Form.Item name="color" label="Color">
          <ColorPicker />
        </Form.Item>
        <Form.Item name="fontSize" label="FontSize">
          <Slider step={1} min={1} max={100} />
        </Form.Item>
        <Form.Item name="zIndex" label="zIndex">
          <Slider step={1} min={0} max={100} />
        </Form.Item>
        <Form.Item name="rotate" label="Rotate">
          <Slider step={1} min={-180} max={180} />
        </Form.Item>
        <Form.Item label="Gap" style={{ marginBottom: 0 }}>
          <Flex gap="small">
            <Form.Item name={['gap', 0]}>
              <InputNumber placeholder="gapX" style={{ width: '100%' }} />
            </Form.Item>
            <Form.Item name={['gap', 1]}>
              <InputNumber placeholder="gapY" style={{ width: '100%' }} />
            </Form.Item>
          </Flex>
        </Form.Item>
        <Form.Item label="Offset" style={{ marginBottom: 0 }}>
```

```
          <Flex gap="small">
            <Form.Item name={['offset', 0]}>
              <InputNumber placeholder="offsetLeft" style={{ width: '100%'
}} />
            </Form.Item>
            <Form.Item name={['offset', 1]}>
              <InputNumber placeholder="offsetTop" style={{ width: '100%'
}} />
            </Form.Item>
          </Flex>
        </Form.Item>
      </Form>
    </Flex>
  );
};

export default App;
```

Modal or Drawer

```
import React from 'react';
import { Button, Drawer, Flex, Modal, Watermark } from 'antd';

const style: React.CSSProperties = {
  height: 300,
  display: 'flex',
  justifyContent: 'center',
  alignItems: 'center',
  backgroundColor: 'rgba(150, 150, 150, 0.2)',
};

const placeholder = <div style={style}>A mock height</div>;

const App: React.FC = () => {
  const [showModal, setShowModal] = React.useState(false);
  const [showDrawer, setShowDrawer] = React.useState(false);
  const [showDrawer2, setShowDrawer2] = React.useState(false);

  const closeModal = () => setShowModal(false);
  const closeDrawer = () => setShowDrawer(false);
  const closeDrawer2 = () => setShowDrawer2(false);

  return (
    <>
      <Flex gap="middle">
        <Button type="primary" onClick={() => setShowModal(true)}>
```

```
          Show in Modal
        </Button>
        <Button type="primary" onClick={() => setShowDrawer(true)}>
          Show in Drawer
        </Button>
        <Button type="primary" onClick={() => setShowDrawer2(true)}>
          Not Show in Drawer
        </Button>
      </Flex>
      <Watermark content="Ant Design">
        <Modal
          destroyOnClose
          open={showModal}
          title="Modal"
          onCancel={closeModal}
          onOk={closeModal}
        >
          {placeholder}
        </Modal>
        <Drawer destroyOnClose open={showDrawer} title="Drawer" onClose=
{closeDrawer}>
          {placeholder}
        </Drawer>
      </Watermark>
      <Watermark content="Ant Design" inherit={false}>
        <Drawer destroyOnClose open={showDrawer2} title="Drawer" onClose=
{closeDrawer2}>
          {placeholder}
        </Drawer>
      </Watermark>
    </>
  );
};

export default App;
```

API

Common props ref: Common props

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

Watermark

Property	Description	Type	Default	Version

width	The width of the watermark, the default value of `content` is its own width	number	120	
height	The height of the watermark, the default value of `content` is its own height	number	64	
inherit	Pass the watermark to the pop-up component such as Modal, Drawer	boolean	true	5.11.0
rotate	When the watermark is drawn, the rotation Angle, unit °	number	-22	
zIndex	The z-index of the appended watermark element	number	9	
image	Image source, it is recommended to export 2x or 3x image, high priority (support base64 format)	string	-	
content	Watermark text content	string | string[]	-	
font	Text style	Font	Font	
gap	The spacing between watermarks	[number, number]	[100, 100]	
offset	The offset of the watermark from the upper left corner of the container. The default is `gap/2`	[number, number]	[gap[0]/2, gap[1]/2]	

Font

Property	Description	Type	Default	Version
color	font color	CanvasFillStrokeStyles.fillStyle	rgba(0,0,0,.15)	
fontSize	font size	number	16	
fontWeight	font weight	`normal` | `light` | `weight` | number	normal	
fontFamily	font family	string	sans-serif	
fontStyle	font style	`none` | `normal` | `italic` | `oblique`	normal	
textAlign	specify the text alignment	CanvasTextAlign	`center`	5.10.0

	direction			

Design Token

FAQ

Handle abnormal image watermarks

When using an image watermark and the image loads abnormally, you can add `content` at the same time to prevent the watermark from becoming invalid (since 5.2.3).

```
<Watermark
  height={30}
  width={130}
  content="Ant Design"

image="https://mdn.alipayobjects.com/huamei_7uahnr/afts/img/A*lkAoRbywo0oAAAA
>
  <div style={{ height: 500 }} />
</Watermark>
```

Why `overflow: hidden` style is added since version 5.18.0?

User can hide the watermark by setting the container height to 0 through the developer tool in the previous version. To avoid this situation, we added the `overflow: hidden` style to the container. When the container height changes, the content is also hidden. You can override the style to modify this behavior:

```
<Watermark style={{ overflow: 'visible' }} />
```