何时使用 {#when-to-use}

- 网络较慢，需要长时间等待加载处理的情况下。
- 图文信息内容较多的列表/卡片中。
- 只在第一次加载数据的时候使用。
- 可以被 Spin 完全代替，但是在可用的场景下可以比 Spin 提供更好的视觉效果和用户体验。

代码演示

基本

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

const App: React.FC = () => <Skeleton />;

export default App;
```

复杂的组合

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

const App: React.FC = () => <Skeleton avatar paragraph={{ rows: 4 }} />;

export default App;
```

动画效果

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

const App: React.FC = () => <Skeleton active />;

export default App;
```

按钮/头像/输入框/图像/自定义节点

```
import React, { useState } from 'react';
import { DotChartOutlined } from '@ant-design/icons';
import type { RadioChangeEvent } from 'antd';
import { Flex, Divider, Form, Radio, Skeleton, Space, Switch } from 'antd';

type SizeType = 'default' | 'small' | 'large';
type ButtonShapeType = 'circle' | 'square' | 'round' | 'default';
```

```tsx
type AvatarShapeType = 'circle' | 'square';

const App: React.FC = () => {
  const [active, setActive] = useState(false);
  const [block, setBlock] = useState(false);
  const [size, setSize] = useState<SizeType>('default');
  const [buttonShape, setButtonShape] = useState<ButtonShapeType>
('default');
  const [avatarShape, setAvatarShape] = useState<AvatarShapeType>
('circle');

  const handleActiveChange = (checked: boolean) => {
    setActive(checked);
  };

  const handleBlockChange = (checked: boolean) => {
    setBlock(checked);
  };

  const handleSizeChange = (e: RadioChangeEvent) => {
    setSize(e.target.value);
  };

  const handleShapeButton = (e: RadioChangeEvent) => {
    setButtonShape(e.target.value);
  };

  const handleAvatarShape = (e: RadioChangeEvent) => {
    setAvatarShape(e.target.value);
  };

  return (
    <Flex gap="middle" vertical>
      <Space>
        <Skeleton.Button active={active} size={size} shape={buttonShape}
block={block} />
        <Skeleton.Avatar active={active} size={size} shape={avatarShape} />
        <Skeleton.Input active={active} size={size} />
      </Space>
      <Skeleton.Button active={active} size={size} shape={buttonShape}
block={block} />
      <Skeleton.Input active={active} size={size} block={block} />
      <Space>
        <Skeleton.Image active={active} />
        <Skeleton.Node active={active} style={{ width: 160 }} />
        <Skeleton.Node active={active}>
```

```
          <DotChartOutlined style={{ fontSize: 40, color: '#bfbfbf' }} />
        </Skeleton.Node>
      </Space>
      <Divider />
      <Form layout="inline" style={{ margin: '16px 0' }}>
        <Space size={16} wrap>
          <Form.Item label="Active">
            <Switch checked={active} onChange={handleActiveChange} />
          </Form.Item>
          <Form.Item label="Button and Input Block">
            <Switch checked={block} onChange={handleBlockChange} />
          </Form.Item>
          <Form.Item label="Size">
            <Radio.Group value={size} onChange={handleSizeChange}>
              <Radio.Button value="default">Default</Radio.Button>
              <Radio.Button value="large">Large</Radio.Button>
              <Radio.Button value="small">Small</Radio.Button>
            </Radio.Group>
          </Form.Item>
          <Form.Item label="Button Shape">
            <Radio.Group value={buttonShape} onChange={handleShapeButton}>
              <Radio.Button value="default">Default</Radio.Button>
              <Radio.Button value="square">Square</Radio.Button>
              <Radio.Button value="round">Round</Radio.Button>
              <Radio.Button value="circle">Circle</Radio.Button>
            </Radio.Group>
          </Form.Item>
          <Form.Item label="Avatar Shape">
            <Radio.Group value={avatarShape} onChange={handleAvatarShape}>
              <Radio.Button value="square">Square</Radio.Button>
              <Radio.Button value="circle">Circle</Radio.Button>
            </Radio.Group>
          </Form.Item>
        </Space>
      </Form>
    </Flex>
  );
};

export default App;
```

包含子组件

```
import React, { useState } from 'react';
import { Button, Skeleton, Space } from 'antd';
```

```tsx
const App: React.FC = () => {
  const [loading, setLoading] = useState<boolean>(false);

  const showSkeleton = () => {
    setLoading(true);
    setTimeout(() => {
      setLoading(false);
    }, 3000);
  };

  return (
    <Space direction="vertical" style={{ width: '100%' }} size={16}>
      <Skeleton loading={loading}>
        <h4 style={{ marginBottom: 16 }}>Ant Design, a design language</h4>
        <p>
          We supply a series of design principles, practical patterns and
high quality design
          resources (Sketch and Axure), to help people create their product
prototypes beautifully
          and efficiently.
        </p>
      </Skeleton>
      <Button onClick={showSkeleton} disabled={loading}>
        Show Skeleton
      </Button>
    </Space>
  );
};

export default App;
```

列表

```tsx
import React, { useState } from 'react';
import type Icon from '@ant-design/icons';
import { LikeOutlined, MessageOutlined, StarOutlined } from '@ant-
design/icons';
import { Avatar, List, Skeleton, Switch } from 'antd';

interface IconTextProps {
  icon: typeof Icon;
  text: React.ReactNode;
}

const listData = Array.from({ length: 3 }).map((_, i) => ({
  href: 'https://ant.design',
```

```
  title: `ant design part ${i + 1}`,
  avatar: `https://api.dicebear.com/7.x/miniavs/svg?seed=${i}`,
  description:
    'Ant Design, a design language for background applications, is refined
by Ant UED Team.',
  content:
    'We supply a series of design principles, practical patterns and high
quality design resources (Sketch and Axure), to help people create their
product prototypes beautifully and efficiently.',
}));

const IconText: React.FC<IconTextProps> = ({ icon, text }) => (
  <>
    {React.createElement(icon, { style: { marginInlineEnd: 8 } })}
    {text}
  </>
);

const App: React.FC = () => {
  const [loading, setLoading] = useState(true);

  const onChange = (checked: boolean) => {
    setLoading(!checked);
  };

  return (
    <>
      <Switch checked={!loading} onChange={onChange} style={{ marginBottom:
16 }} />
      <List
        itemLayout="vertical"
        size="large"
        dataSource={listData}
        renderItem={(item) => (
          <List.Item
            key={item.title}
            actions={
              !loading
                ? [
                    <IconText icon={StarOutlined} text="156" key="list-
vertical-star-o" />,
                    <IconText icon={LikeOutlined} text="156" key="list-
vertical-like-o" />,
                    <IconText icon={MessageOutlined} text="2" key="list-
vertical-message" />,
                  ]
```

```
                : undefined
            }
            extra={
              !loading && (
                <img
                  width={272}
                  alt="logo"
src="https://gw.alipayobjects.com/zos/rmsportal/mqaQswcyDLcXyDKnZfES.png"
                />
              )
            }
          >
            <Skeleton loading={loading} active avatar>
              <List.Item.Meta
                avatar={<Avatar src={item.avatar} />}
                title={<a href={item.href}>{item.title}</a>}
                description={item.description}
              />
              {item.content}
            </Skeleton>
          </List.Item>
        )}
      />
    </>
  );
};

export default App;
```

自定义组件 Token

Debug

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

const App: React.FC = () => (
  <ConfigProvider
    theme={{
      components: {
        Skeleton: {
          blockRadius: 30,
          titleHeight: 50,
          gradientFromColor: '#222',
          gradientToColor: '#444',
```

```
          paragraphMarginTop: 30,
          paragraphLiHeight: 30,
        },
      },
    }}
  >
    <Skeleton loading active />
  </ConfigProvider>
);

export default App;
```

API

通用属性参考：通用属性

Skeleton

属性	说明	类型	默认值
active	是否展示动画效果	boolean	false
avatar	是否显示头像占位图	boolean | SkeletonAvatarProps	false
loading	为 true 时，显示占位图。反之则直接展示子组件	boolean	-
paragraph	是否显示段落占位图	boolean | SkeletonParagraphProps	true
round	为 true 时，段落和标题显示圆角	boolean	false
title	是否显示标题占位图	boolean | SkeletonTitleProps	true

SkeletonAvatarProps

属性	说明	类型	默认值
active	是否展示动画效果，仅在单独使用头像骨架时生效	boolean	false
shape	指定头像的形状	`circle` | `square`	-
size	设置头像占位图的大小	number | `large` | `small` | `default`	-

SkeletonTitleProps

属性	说明	类型	默认值
width	设置标题占位图的宽度	number | string	-

SkeletonParagraphProps

属性	说明	类型	默认值
rows	设置段落占位图的行数	number	-
width	设置段落占位图的宽度，若为数组时则为对应的每行宽度，反之则是最后一行的宽度	number | string | Array<number | string>	-

SkeletonButtonProps

属性	说明	类型	默认值	版本
active	是否展示动画效果	boolean	false	
block	将按钮宽度调整为其父宽度的选项	boolean	false	4.17.0
shape	指定按钮的形状	`circle` | `round` | `square` | `default`	-	
size	设置按钮的大小	`large` | `small` | `default`	-	

SkeletonInputProps

属性	说明	类型	默认值
active	是否展示动画效果	boolean	false
size	设置输入框的大小	`large` | `small` | `default`	-

主题变量（Design Token）