

# List

Simple List.

## When To Use

A list can be used to display content related to a single subject. The content can consist of multiple elements of varying type and size.

## Examples



### Default Size

|  |
|--|
| Header   |
| [ITEM] Racing car sprays burning fuel into crowd.  |
| [ITEM] Japanese princess to wed commoner.          |
| [ITEM] Australian walks 100km after outback crash. |
| [ITEM] Man charged over missing wedding girl.      |
| [ITEM] Los Angeles battles huge wildfires.         |
| Footer   |

### Small Size

|   |
|---|
| Header                                      |
| Racing car sprays burning fuel into crowd.  |
| Japanese princess to wed commoner.          |
| Australian walks 100km after outback crash. |
| Man charged over missing wedding girl.      |
| Los Angeles battles huge wildfires.         |
| Footer                                      |

### Large Size

|   |
|---|
| Header                                      |
| Racing car sprays burning fuel into crowd.  |
| Japanese princess to wed commoner.          |
| Australian walks 100km after outback crash. |
| Man charged over missing wedding girl.      |
| Los Angeles battles huge wildfires.         |
| Footer                                      |

### Simple list [✎](#)

Ant Design supports a default list size as well as a large and small size.

If a large or small list is desired, set the size property to either large or small respectively. Omit the size property for a list with the default size.

Customizing the header and footer of list by setting `header` and `footer` property.

```
import React from 'react';
import { Divider, List, Typography } from 'antd';
```



### Ant Design Title 1

Ant Design, a design language for background applications, is refined by Ant UED Team



### Ant Design Title 2

Ant Design, a design language for background applications, is refined by Ant UED Team



### Ant Design Title 3

Ant Design, a design language for background applications, is refined by Ant UED Team



### Ant Design Title 4

Ant Design, a design language for background applications, is refined by Ant UED Team

## Basic list


Basic list.

```
import { Avatar, List } from 'antd';
import React from 'react';

const data = [
  {
    title: 'Ant Design Title 1',
  },
  {
    title: 'Ant Design Title 2',
  },
  {
    title: 'Ant Design Title 3',
  },
  {
    title: 'Ant Design Title 4',
  },
];

const App: React.FC = () => (
  <List
    itemLayout="horizontal"
    dataSource={data}
    renderItem={(item, index) => (
      <List.Item>
        <List.Item.Meta
          avatar={<Avatar src={`https://xsgames.co/randomusers/avatar.php?g=pixel&key=${index}`} />
          title={<a href="https://ant.design">{item.title}</a>}
          description="Ant Design, a design language for background applications, is refined by Ant UED Team" />
        </List.Item.Meta>
      </List.Item>
    )}
  />
);

export default App;
```

[Load more](#) 

Load more list with `loadMore` property.

```
import React, { useEffect, useState } from 'react';
import { Avatar, Button, List, Skeleton } from 'antd';

interface DataType {
  gender?: string;
  name: {
    title?: string;
    first?: string;
    last?: string;
  };
  email?: string;
  picture: {
    large?: string;
    medium?: string;
    thumbnail?: string;
  };
  nat?: string;
  loading: boolean;
}

const count = 3;
const fakeDataUrl = `https://randomuser.me/api/?results=${count}&inc=name,gender,email,nat,pictu

const App: React.FC = () => {
  const [initLoading, setInitLoading] = useState(true);
  const [loading, setLoading] = useState(false);
  const [data, setData] = useState<DataType[]>([]);
  const [list, setList] = useState<DataType[]>([]);

  useEffect(() => {
    fetch(fakeDataUrl)
      .then((res) => res.json())
      .then((res) => {
        setInitLoading(false);
        setData(res.results);
        setList(res.results);
      });
  }, []);

  const onLoadMore = () => {
    setLoading(true);
    setList(
      data.concat([...new Array(count)].map(() => ({ loading: true, name: {}, picture: {} }))),
    );
    fetch(fakeDataUrl)
      .then((res) => res.json())
      .then((res) => {
        const newData = data.concat(res.results);
```

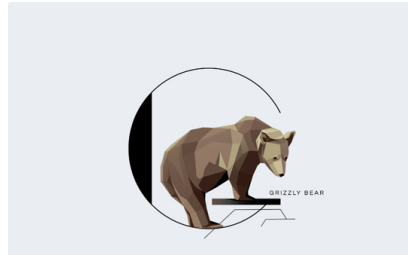


## ant design part 0

Ant Design, a design language for background applications, is refined by Ant UED Team.

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.

☆ 156    👍 156    💬 2

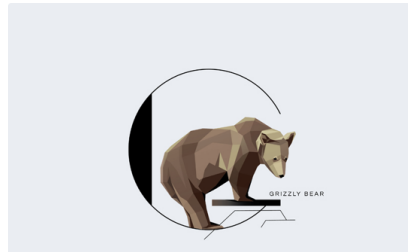


## ant design part 1

Ant Design, a design language for background applications, is refined by Ant UED Team.

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.

☆ 156    👍 156    💬 2

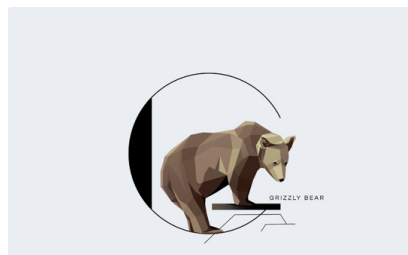


## ant design part 2

Ant Design, a design language for background applications, is refined by Ant UED Team.

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.

☆ 156    👍 156    💬 2



### ant design footer part

< 1 2 3 4 5 ... 8 >

### Vertical

Set the `itemLayout` property to `vertical` to create a vertical list.

```
import { LikeOutlined, MessageOutlined, StarOutlined } from '@ant-design/icons';
import { Avatar, List, Space } from 'antd';
import React from 'react';

const data = Array.from({ length: 23 }).map((_, i) => ({
  href: 'https://ant.design',
  title: `ant design part ${i}`,
  avatar: `https://xsgames.co/randomusers/avatar.php?g=pixel&key=${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'
}));

const IconText = ({ icon, text }: { icon: React.FC; text: string }) => (
  <Space>
    {React.createElement(icon)}
    {text}
  </Space>
);
```

Pagination Position: ☐ top ☒ bottom ☐ both

Pagination Align: ☐ start ☒ center ☐ end



#### Ant Design Title 1

Ant Design, a design language for background applications, is refined by Ant UED Team



#### Ant Design Title 2

Ant Design, a design language for background applications, is refined by Ant UED Team



#### Ant Design Title 3

Ant Design, a design language for background applications, is refined by Ant UED Team



#### Ant Design Title 4

Ant Design, a design language for background applications, is refined by Ant UED Team

< 1 >

### Pagination Settings [✎](#)

List pagination can be used and set through the `pagination` property.

```
import { Avatar, List, Radio, Space } from 'antd';
import React, { useState } from 'react';

type PaginationPosition = 'top' | 'bottom' | 'both';

type PaginationAlign = 'start' | 'center' | 'end';

const data = [
  {
    title: 'Ant Design Title 1',
  },
  {
    title: 'Ant Design Title 2',
  },
  {
    title: 'Ant Design Title 3',
  },
  {
    title: 'Ant Design Title 4',
  },
];

const positionOptions = ['top', 'bottom', 'both'];

const alignOptions = ['start', 'center', 'end'];

const App: React.FC = () => {
  const [position, setPosition] = useState<PaginationPosition>('bottom');
  const [align, setAlign] = useState<PaginationAlign>('center');

  return (
    <>
      <Space direction="vertical" style={{ marginBottom: '20px' }} size="middle">
        <Space>
          <span>Pagination Position:</span>
          <Radio.Group
            optionType="button"
            value={position}
            onChange={(e) => {
              setPosition(e.target.value);
            }}
          >
            {positionOptions.map((item) => (
```

**Title 1**

Card content

**Title 2**

Card content

**Title 3**

Card content

**Title 4**

Card content

## Grid

Create a grid layout by setting the `grid` property of List.

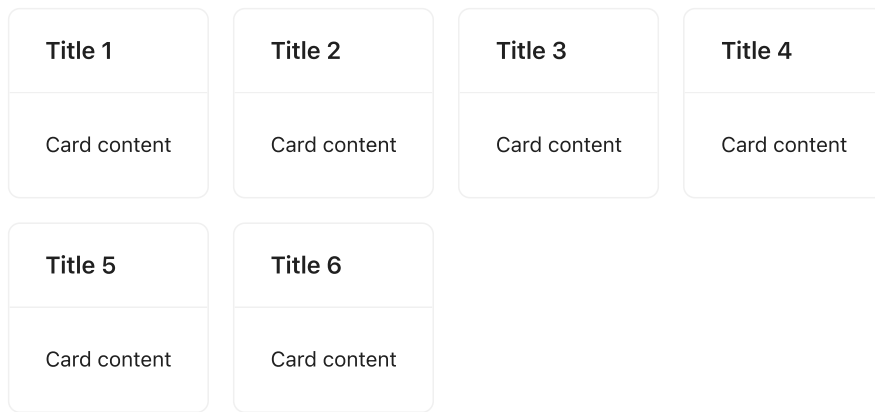
```
import React from 'react';
import { Card, List } from 'antd';

const data = [
  {
    title: 'Title 1',
  },
  {
    title: 'Title 2',
  },
  {
    title: 'Title 3',
  },
  {
    title: 'Title 4',
  },
];

const App: React.FC = () => (
  <List
    grid={{ gutter: 16, column: 4 }}
    dataSource={data}
    renderItem={(item) => (
      <List.Item>
        <Card title={item.title}>Card content</Card>
      </List.Item>
    )}
  />
);

export default App;
```





### Responsive grid list

Responsive grid list. The size property the is as same as [Layout Grid](#).

```
import React from 'react';
import { Card, List } from 'antd';

const data = [
  {
    title: 'Title 1',
  },
  {
    title: 'Title 2',
  },
  {
    title: 'Title 3',
  },
  {
    title: 'Title 4',
  },
  {
    title: 'Title 5',
  },
  {
    title: 'Title 6',
  },
];

const App: React.FC = () => (
  <List
    grid={{
      gutter: 16,
      xs: 1,
      sm: 2,
      md: 4,
      lg: 4,
      xl: 6,
      xxl: 3,
    }}
    dataSource={data}
    renderItem={(item) => (
      <List.Item>
        <Card title={item.title}>Card content</Card>
      </List.Item>
    )}
  />
);

export default App;
```



No data

### Scrolling loaded

The example of infinite load with [react-infinite-scroll-component](#).

```
import React, { useEffect, useState } from 'react';
import { Avatar, Divider, List, Skeleton } from 'antd';
import InfiniteScroll from 'react-infinite-scroll-component';

interface DataType {
  gender: string;
  name: {
    title: string;
    first: string;
    last: string;
  };
  email: string;
  picture: {
    large: string;
    medium: string;
    thumbnail: string;
  };
  nat: string;
}

const App: React.FC = () => {
  const [loading, setLoading] = useState(false);
  const [data, setData] = useState<DataType[]>([]);

  const loadMoreData = () => {
    if (loading) {
      return;
    }
    setLoading(true);
    fetch('https://randomuser.me/api/?results=10&inc=name,gender,email,nat,picture&noinfo')
      .then((res) => res.json())
      .then((body) => {
        setData([...data, ...body.results]);
        setLoading(false);
      })
      .catch(() => {
        setLoading(false);
      });
  };

  useEffect(() => {
    loadMoreData();
  }, []);

  return (
```

## virtual list

An example of infinite & virtualized list via using [rc-virtual-list](#).

```
import React, { useEffect, useState } from 'react';
import { Avatar, List, message } from 'antd';
import Virtuallist from 'rc-virtual-list';

interface UserItem {
  email: string;
  gender: string;
  name: {
    first: string;
    last: string;
    title: string;
  };
  nat: string;
  picture: {
    large: string;
    medium: string;
    thumbnail: string;
  };
};

const fakeDataUrl =
  'https://randomuser.me/api/?results=20&inc=name,gender,email,nat,picture&noinfo';
const ContainerHeight = 400;

const App: React.FC = () => {
  const [data, setData] = useState<UserItem[]>([]);

  const appendData = () => {
    fetch(fakeDataUrl)
      .then((res) => res.json())
      .then((body) => {
        setData(data.concat(body.results));
        message.success(`${body.results.length} more items loaded!`);
      });
  };

  useEffect(() => {
    appendData();
  }, []);

  const onScroll = (e: React.UIEvent<HTMLDivElement, UIEvent>) => {
    if (e.currentTarget.scrollHeight - e.currentTarget.scrollTop === ContainerHeight) {
      appendData();
    }
  };
};
```

```
return (  
  <List>  
    <VirtualList  
      data={data}  
      height={ContainerHeight}  
      itemHeight={47}  
      itemKey="email"  
      onScroll={onScroll}  
    >  
      {(item: UserItem) => (  
        <List.Item key={item.email}>  
          <List.Item.Meta  
            avatar={<Avatar src={item.picture.large} />}  
            title={<a href="https://ant.design">{item.name.last}</a>}  
            description={item.email}  
          />  
          <div>Content</div>  
        </List.Item>  
      )}  
    </VirtualList>  
  </List>  
)  
};  
  
export default App;
```

# API

## List

| Property   | Description   | Type   | Default               | Version |
|------------|---|--|-----------------------|---------|
| bordered   | Toggles rendering of the border around the list                                   | boolean  | false                 |         |
| dataSource | DataSource array for list   | any[]  | -                     |         |
| footer     | List footer renderer  | ReactNode  | -                     |         |
| grid       | The grid type of list. You can set grid to something like {gutter: 16, column: 4} | <a href="#">object</a>                                       | -                     |         |
| header     | List header renderer  | ReactNode  | -                     |         |
| itemLayout | The layout of list  | <div>horizontal</div> <div> </div> <div>vertical</div>       | <div>horizontal</div> |         |
| loading    | Shows a loading indicator while the contents of the list are being fetched        | boolean   <a href="#">SpinProps</a> ( <a href="#">more</a> ) | false                 |         |
| loadMore   | Shows a load more content   | ReactNode  | -                     |         |
| locale     | The i18n text including   | object   | {emptyText:           |         |

| Property   | Description   | Type   | Default                | Version |
|------------|---|--|------------------------|---------|
|            | empty text  |  | <code>No Data</code> } |         |
| pagination | Pagination <a href="#">config</a> , hide it by setting it to false  | boolean   object   | false                  |         |
| renderItem | Customize list item when using <code>dataSource</code>  | (item) => ReactNode  | -                      |         |
| rowKey     | Item's unique value, could be an Item's key which holds a unique value of type <code>React.Key</code> or function that receives Item and returns a <code>React.Key</code> | <code>keyof T   (item: T) =&gt; React.Key</code>               | "key"                  |         |
| size       | Size of list  | <code>default</code>   <code>large</code>   <code>small</code> | default                |         |
| split      | Toggles rendering of the split under the list item  | boolean  | true                   |         |

pagination

Properties for pagination.

| Property | Description  | Type  | Default |
|----------|--|---|---------|
| position | The specify the position of <code>Pagination</code>  | <code>top</code>   <code>bottom</code>   <code>both</code>  | bottom  |
| align    | The specify the alignment of <code>Pagination</code> | <code>start</code>   <code>center</code>   <code>end</code> | end     |

More about pagination, please check [Pagination](#) .

List grid props

| Property | Description                           | Type   | Default | Version |
|----------|---------------------------------------|--------|---------|---------|
| column   | The column of grid                    | number | -       |         |
| gutter   | The spacing between grid              | number | 0       |         |
| xs       | <code>&lt;576px</code> column of grid | number | -       |         |
| sm       | <code>≥576px</code> column of grid    | number | -       |         |
| md       | <code>≥768px</code> column of grid    | number | -       |         |
| lg       | <code>≥992px</code> column of grid    | number | -       |         |
| xl       | <code>≥1200px</code> column of grid   | number | -       |         |

| Property | Description                         | Type   | Default | Version |
|----------|-------------------------------------|--------|---------|---------|
| xxl      | <code>≥1600px</code> column of grid | number | -       |         |

List.Item

| Property | Description   | Type                                | Default | Version |
|----------|---|-------------------------------------|---------|---------|
| actions  | The actions content of list item. If <code>itemLayout</code> is <code>vertical</code> , shows the content on bottom, otherwise shows content on the far right | <code>Array&lt;ReactNode&gt;</code> | -       |         |
| extra    | The extra content of list item. If <code>itemLayout</code> is <code>vertical</code> , shows the content on right, otherwise shows content on the far right    | <code>ReactNode</code>              | -       |         |

List.Item.Meta

| Property    | Description                  | Type                   | Default | Version |
|-------------|------------------------------|------------------------|---------|---------|
| avatar      | The avatar of list item      | <code>ReactNode</code> | -       |         |
| description | The description of list item | <code>ReactNode</code> | -       |         |
| title       | The title of list item       | <code>ReactNode</code> | -       |         |

Design Token

▼ Global Token

| Token Name   | Description   | Type                | Default Value                    |
|--------------|---|---------------------|----------------------------------|
| colorBorder  | Default border color, used to separate different elements, such as: form separator, card separator, etc.  | <code>string</code> | <code>#d9d9d9</code>             |
| 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. | <code>string</code> | <code>#1677ff</code>             |
| colorSplit   | Used as the color of separator, this color is the same as <code>colorBorderSecondary</code> but with transparency.  | <code>string</code> | <code>rgba(5, 5, 5, 0.06)</code> |
| colorText    | Default text color which comply with W3C standards, and this color is also the darkest neutral color.   | <code>string</code> | <code>rgba(0, 0, 0, 0.88)</code> |

| Token Name           | Description   | Type   | Default Value   |
|----------------------|---|--------|---|
| colorTextDescription | Control the font color of text description.   | string | <code>rgba(0, 0, 0, 0.45)</code>  |
| colorTextDisabled    | Control the color of text in disabled state.  | string | <code>rgba(0, 0, 0, 0.25)</code>  |
| borderRadiusLG       | LG size border radius, used in some large border radius components, such as Card, Modal and other components.   | number | 8   |
| controlHeight        | The height of the basic controls such as buttons and input boxes in Ant Design  | number | 32  |
| controlHeightLG      | LG component height   | number | 40  |
| 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 platforms and browsers, reflecting the friendly, stable and professional characteristics. | string | -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto, 'Helvetica Neue', Arial, 'Noto Sans', sans-serif, 'Apple Color Emoji', 'Segoe UI Emoji', '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  |
| fontSizeSM           | Small font size   | number | 12  |
| lineHeight           | Line height of text.  | number | 1.5714285714285714  |
| lineHeightLG         | Line height of large text.  | number | 1.5   |
| lineType             | Border style of base components   | string | solid   |
| lineWidth            | Border width of base components   | number | 1   |
| margin               | Control the margin of an element, with a medium size.   | number | 16  |
| marginLG             | Control the margin of an element, with a large size.  | number | 24  |
| marginSM             | Control the margin of an element, with a medium-small size.   | number | 12  |
| marginXXL            | Control the margin of an element, with the largest size.  | number | 48  |
| marginXXS            | Control the margin of an element, with the smallest size.   | number | 4   |
| motionDurationSlow   | Motion speed, slow speed. Used for large element animation interaction.   | string | 0.3s  |



| Token Name                 | Description   | Type   | Default Value |
|----------------------------|---|--------|---------------|
| padding                    | Control the padding of the element.   | number | 16            |
| paddingContentHorizontal   | Control the horizontal padding of content element.                                    | number | 16            |
| paddingContentHorizontalLG | Control the horizontal padding of content element, suitable for large screen devices. | number | 24            |
| paddingContentVertical     | Control the vertical padding of content element.                                      | number | 12            |
| paddingContentVerticalLG   | Control the vertical padding of content element, suitable for large screen devices.   | number | 16            |
| paddingContentVerticalSM   | Control the vertical padding of content element, suitable for small screen devices.   | number | 8             |
| paddingLG                  | Control the large padding of the element.   | number | 24            |
| paddingSM                  | Control the small padding of the element.   | number | 12            |
| paddingXS                  | Control the extra small padding of the element.                                       | number | 8             |
| screenMD                   | Control the screen width of medium screens.   | number | 768           |
| screenSM                   | Control the screen width of small screens.  | number | 576           |