When To Use

- It can be used to tag by dimension or property.
- When categorizing.

Examples

Basic

```tsx
import React from 'react';
import { CloseCircleOutlined } from '@ant-design/icons';
import { Tag } from 'antd';

const preventDefault = (e: React.MouseEvent<HTMLElement>) => {
  e.preventDefault();
  console.log('Clicked! But prevent default.');
};

const App: React.FC = () => (
  <>
    <Tag>Tag 1</Tag>
    <Tag>
      <a
        href="https://github.com/ant-design/ant-design/issues/1862"
        target="_blank"
        rel="noopener noreferrer"
      >
        Link
      </a>
    </Tag>
    <Tag closeIcon onClose={preventDefault}>
      Prevent Default
    </Tag>
    <Tag closeIcon={<CloseCircleOutlined />} onClose={console.log}>
      Tag 2
    </Tag>
  </>
);

export default App;
```

Colorful Tag

```tsx
import React from 'react';
import { Divider, Flex, Tag } from 'antd';
```

```tsx
const App: React.FC = () => (
  <>
    <Divider orientation="left">Presets</Divider>
    <Flex gap="4px 0" wrap>
      <Tag color="magenta">magenta</Tag>
      <Tag color="red">red</Tag>
      <Tag color="volcano">volcano</Tag>
      <Tag color="orange">orange</Tag>
      <Tag color="gold">gold</Tag>
      <Tag color="lime">lime</Tag>
      <Tag color="green">green</Tag>
      <Tag color="cyan">cyan</Tag>
      <Tag color="blue">blue</Tag>
      <Tag color="geekblue">geekblue</Tag>
      <Tag color="purple">purple</Tag>
    </Flex>
    <Divider orientation="left">Custom</Divider>
    <Flex gap="4px 0" wrap>
      <Tag color="#f50">#f50</Tag>
      <Tag color="#2db7f5">#2db7f5</Tag>
      <Tag color="#87d068">#87d068</Tag>
      <Tag color="#108ee9">#108ee9</Tag>
    </Flex>
  </>
);

export default App;
```

Inverse Colorful Tag

Debug

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

const App: React.FC = () => (
  <Flex gap="4px 0" wrap>
    <Tag color="magenta-inverse">magenta</Tag>
    <Tag color="red-inverse">red</Tag>
    <Tag color="volcano-inverse">volcano</Tag>
    <Tag color="orange-inverse">orange</Tag>
    <Tag color="gold-inverse">gold</Tag>
    <Tag color="lime-inverse">lime</Tag>
    <Tag color="green-inverse">green</Tag>
    <Tag color="cyan-inverse">cyan</Tag>
```

```
    <Tag color="blue-inverse">blue</Tag>
    <Tag color="geekblue-inverse">geekblue</Tag>
    <Tag color="purple-inverse">purple</Tag>
  </Flex>
);

export default App;
```

Add & Remove Dynamically

```
import React, { useEffect, useRef, useState } from 'react';
import { PlusOutlined } from '@ant-design/icons';
import type { InputRef } from 'antd';
import { Flex, Input, Tag, theme, Tooltip } from 'antd';

const tagInputStyle: React.CSSProperties = {
  width: 64,
  height: 22,
  marginInlineEnd: 8,
  verticalAlign: 'top',
};

const App: React.FC = () => {
  const { token } = theme.useToken();
  const [tags, setTags] = useState<string[]>(['Unremovable', 'Tag 2', 'Tag
3']);
  const [inputVisible, setInputVisible] = useState(false);
  const [inputValue, setInputValue] = useState('');
  const [editInputIndex, setEditInputIndex] = useState(-1);
  const [editInputValue, setEditInputValue] = useState('');
  const inputRef = useRef<InputRef>(null);
  const editInputRef = useRef<InputRef>(null);

  useEffect(() => {
    if (inputVisible) {
      inputRef.current?.focus();
    }
  }, [inputVisible]);

  useEffect(() => {
    editInputRef.current?.focus();
  }, [editInputValue]);

  const handleClose = (removedTag: string) => {
    const newTags = tags.filter((tag) => tag !== removedTag);
    console.log(newTags);
```

```
    setTags(newTags);
  };

  const showInput = () => {
    setInputVisible(true);
  };

  const handleInputChange = (e: React.ChangeEvent<HTMLInputElement>) => {
    setInputValue(e.target.value);
  };

  const handleInputConfirm = () => {
    if (inputValue && !tags.includes(inputValue)) {
      setTags([...tags, inputValue]);
    }
    setInputVisible(false);
    setInputValue('');
  };

  const handleEditInputChange = (e: React.ChangeEvent<HTMLInputElement>) =>
{
    setEditInputValue(e.target.value);
  };

  const handleEditInputConfirm = () => {
    const newTags = [...tags];
    newTags[editInputIndex] = editInputValue;
    setTags(newTags);
    setEditInputIndex(-1);
    setEditInputValue('');
  };

  const tagPlusStyle: React.CSSProperties = {
    height: 22,
    background: token.colorBgContainer,
    borderStyle: 'dashed',
  };

  return (
    <Flex gap="4px 0" wrap>
      {tags.map<React.ReactNode>((tag, index) => {
        if (editInputIndex === index) {
          return (
            <Input
              ref={editInputRef}
              key={tag}
```

```
          size="small"
          style={tagInputStyle}
          value={editInputValue}
          onChange={handleEditInputChange}
          onBlur={handleEditInputConfirm}
          onPressEnter={handleEditInputConfirm}
        />
      );
    }
    const isLongTag = tag.length > 20;
    const tagElem = (
      <Tag
        key={tag}
        closable={index !== 0}
        style={{ userSelect: 'none' }}
        onClose={() => handleClose(tag)}
      >
        <span
          onDoubleClick={(e) => {
            if (index !== 0) {
              setEditInputIndex(index);
              setEditInputValue(tag);
              e.preventDefault();
            }
          }}
        >
          {isLongTag ? `${tag.slice(0, 20)}...` : tag}
        </span>
      </Tag>
    );
    return isLongTag ? (
      <Tooltip title={tag} key={tag}>
        {tagElem}
      </Tooltip>
    ) : (
      tagElem
    );
  })}
  {inputVisible ? (
    <Input
      ref={inputRef}
      type="text"
      size="small"
      style={tagInputStyle}
      value={inputValue}
      onChange={handleInputChange}
```

```
          onBlur={handleInputConfirm}
          onPressEnter={handleInputConfirm}
        />
      ) : (
        <Tag style={tagPlusStyle} icon={<PlusOutlined />} onClick=
{showInput}>
          New Tag
        </Tag>
      )}
    </Flex>
  );
};

export default App;
```

Checkable

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

const tagsData = ['Movies', 'Books', 'Music', 'Sports'];

const App: React.FC = () => {
  const [selectedTags, setSelectedTags] = React.useState<string[]>
(['Movies']);
  const handleChange = (tag: string, checked: boolean) => {
    const nextSelectedTags = checked
      ? [...selectedTags, tag]
      : selectedTags.filter((t) => t !== tag);
    console.log('You are interested in: ', nextSelectedTags);
    setSelectedTags(nextSelectedTags);
  };

  return (
    <Flex gap={4} wrap align="center">
      <span>Categories:</span>
      {tagsData.map<React.ReactNode>((tag) => (
        <Tag.CheckableTag
          key={tag}
          checked={selectedTags.includes(tag)}
          onChange={(checked) => handleChange(tag, checked)}
        >
          {tag}
        </Tag.CheckableTag>
      ))}
```

```
    </Flex>
  );
};

export default App;
```

Animate

```
import React, { useEffect, useRef, useState } from 'react';
import { PlusOutlined } from '@ant-design/icons';
import type { InputRef } from 'antd';
import { Input, Tag, theme } from 'antd';
import { TweenOneGroup } from 'rc-tween-one';

const App: React.FC = () => {
  const { token } = theme.useToken();
  const [tags, setTags] = useState(['Tag 1', 'Tag 2', 'Tag 3']);
  const [inputVisible, setInputVisible] = useState(false);
  const [inputValue, setInputValue] = useState('');
  const inputRef = useRef<InputRef>(null);

  useEffect(() => {
    if (inputVisible) {
      inputRef.current?.focus();
    }
  }, [inputVisible]);

  const handleClose = (removedTag: string) => {
    const newTags = tags.filter((tag) => tag !== removedTag);
    console.log(newTags);
    setTags(newTags);
  };

  const showInput = () => {
    setInputVisible(true);
  };

  const handleInputChange = (e: React.ChangeEvent<HTMLInputElement>) => {
    setInputValue(e.target.value);
  };

  const handleInputConfirm = () => {
    if (inputValue && tags.indexOf(inputValue) === -1) {
      setTags([...tags, inputValue]);
    }
```

```
  setInputVisible(false);
  setInputValue('');
};

const forMap = (tag: string) => (
  <span key={tag} style={{ display: 'inline-block' }}>
    <Tag
      closable
      onClose={(e) => {
        e.preventDefault();
        handleClose(tag);
      }}
    >
      {tag}
    </Tag>
  </span>
);

const tagChild = tags.map(forMap);

const tagPlusStyle: React.CSSProperties = {
  background: token.colorBgContainer,
  borderStyle: 'dashed',
};

return (
  <>
    <div style={{ marginBottom: 16 }}>
      <TweenOneGroup
        appear={false}
        enter={{ scale: 0.8, opacity: 0, type: 'from', duration: 100 }}
        leave={{ opacity: 0, width: 0, scale: 0, duration: 200 }}
        onEnd={(e) => {
          if (e.type === 'appear' || e.type === 'enter') {
            (e.target as any).style = 'display: inline-block';
          }
        }}
      >
        {tagChild}
      </TweenOneGroup>
    </div>
    {inputVisible ? (
      <Input
        ref={inputRef}
        type="text"
        size="small"
```

```
          style={{ width: 78 }}
          value={inputValue}
          onChange={handleInputChange}
          onBlur={handleInputConfirm}
          onPressEnter={handleInputConfirm}
        />
      ) : (
        <Tag onClick={showInput} style={tagPlusStyle}>
          <PlusOutlined /> New Tag
        </Tag>
      )}
    </>
  );
};

export default App;
```

Icon

```
import React from 'react';
import {
  FacebookOutlined,
  LinkedinOutlined,
  TwitterOutlined,
  YoutubeOutlined,
} from '@ant-design/icons';
import { Flex, Tag } from 'antd';

const App: React.FC = () => (
  <Flex gap="4px 0" wrap>
    <Tag icon={<TwitterOutlined />} color="#55acee">
      Twitter
    </Tag>
    <Tag icon={<YoutubeOutlined />} color="#cd201f">
      Youtube
    </Tag>
    <Tag icon={<FacebookOutlined />} color="#3b5999">
      Facebook
    </Tag>
    <Tag icon={<LinkedinOutlined />} color="#55acee">
      LinkedIn
    </Tag>
  </Flex>
);
```

```
export default App;
```

Status Tag

```
import React from 'react';
import {
  CheckCircleOutlined,
  ClockCircleOutlined,
  CloseCircleOutlined,
  ExclamationCircleOutlined,
  MinusCircleOutlined,
  SyncOutlined,
} from '@ant-design/icons';
import { Divider, Flex, Tag } from 'antd';

const App: React.FC = () => (
  <>
    <Divider orientation="left">Without icon</Divider>
    <Flex gap="4px 0" wrap>
      <Tag color="success">success</Tag>
      <Tag color="processing">processing</Tag>
      <Tag color="error">error</Tag>
      <Tag color="warning">warning</Tag>
      <Tag color="default">default</Tag>
    </Flex>
    <Divider orientation="left">With icon</Divider>
    <Flex gap="4px 0" wrap>
      <Tag icon={<CheckCircleOutlined />} color="success">
        success
      </Tag>
      <Tag icon={<SyncOutlined spin />} color="processing">
        processing
      </Tag>
      <Tag icon={<CloseCircleOutlined />} color="error">
        error
      </Tag>
      <Tag icon={<ExclamationCircleOutlined />} color="warning">
        warning
      </Tag>
      <Tag icon={<ClockCircleOutlined />} color="default">
        waiting
      </Tag>
      <Tag icon={<MinusCircleOutlined />} color="default">
        stop
      </Tag>
```

```
    </Flex>
  </>
);

export default App;
```

borderless

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

const App: React.FC = () => (
  <>
    <Flex gap="4px 0" wrap>
      <Tag bordered={false}>Tag 1</Tag>
      <Tag bordered={false}>Tag 2</Tag>
      <Tag bordered={false} closable>
        Tag 3
      </Tag>
      <Tag bordered={false} closable>
        Tag 4
      </Tag>
    </Flex>
    <Divider />
    <Flex gap="4px 0" wrap>
      <Tag bordered={false} color="processing">
        processing
      </Tag>
      <Tag bordered={false} color="success">
        success
      </Tag>
      <Tag bordered={false} color="error">
        error
      </Tag>
      <Tag bordered={false} color="warning">
        warning
      </Tag>
      <Tag bordered={false} color="magenta">
        magenta
      </Tag>
      <Tag bordered={false} color="red">
        red
      </Tag>
      <Tag bordered={false} color="volcano">
        volcano
      </Tag>
```

```
      <Tag bordered={false} color="orange">
        orange
      </Tag>
      <Tag bordered={false} color="gold">
        gold
      </Tag>
      <Tag bordered={false} color="lime">
        lime
      </Tag>
      <Tag bordered={false} color="green">
        green
      </Tag>
      <Tag bordered={false} color="cyan">
        cyan
      </Tag>
      <Tag bordered={false} color="blue">
        blue
      </Tag>
      <Tag bordered={false} color="geekblue">
        geekblue
      </Tag>
      <Tag bordered={false} color="purple">
        purple
      </Tag>
    </Flex>
  </>
);

export default App;
```

borderless in layout

Debug

```
import React from 'react';
import { Divider, Flex, Tag, theme } from 'antd';

const App: React.FC = () => {
  const { token } = theme.useToken();
  return (
    <div style={{ backgroundColor: token.colorBgLayout, padding:
token.padding }}>
      <Flex gap="4px 0" wrap>
        <Tag bordered={false}>Tag 1</Tag>
        <Tag bordered={false}>Tag 2</Tag>
        <Tag bordered={false} closable>
```

```
          Tag 3
        </Tag>
        <Tag bordered={false} closable>
          Tag 4
        </Tag>
      </Flex>
      <Divider />
      <Flex gap="4px 0" wrap>
        <Tag bordered={false} color="magenta">
          magenta
        </Tag>
        <Tag bordered={false} color="red">
          red
        </Tag>
        <Tag bordered={false} color="volcano">
          volcano
        </Tag>
        <Tag bordered={false} color="orange">
          orange
        </Tag>
        <Tag bordered={false} color="gold">
          gold
        </Tag>
        <Tag bordered={false} color="lime">
          lime
        </Tag>
        <Tag bordered={false} color="green">
          green
        </Tag>
        <Tag bordered={false} color="cyan">
          cyan
        </Tag>
        <Tag bordered={false} color="blue">
          blue
        </Tag>
        <Tag bordered={false} color="geekblue">
          geekblue
        </Tag>
        <Tag bordered={false} color="purple">
          purple
        </Tag>
      </Flex>
    </div>
  );
};
```

```tsx
export default App;
```

Customize close

Debug

```tsx
import React from 'react';
import { CloseCircleOutlined } from '@ant-design/icons';
import { Flex, Tag } from 'antd';

const App: React.FC = () => (
  <Flex gap="4px 0" wrap>
    <Tag closable closeIcon="关 闭">
      Tag1
    </Tag>
    <Tag closable closeIcon={<CloseCircleOutlined />}>
      Tag2
    </Tag>
  </Flex>
);

export default App;
```

Draggable Tag

```tsx
import React, { useState } from 'react';
import { closestCenter, DndContext, PointerSensor, useSensor, useSensors }
from '@dnd-kit/core';
import type { DragEndEvent } from '@dnd-kit/core';
import {
  arrayMove,
  horizontalListSortingStrategy,
  SortableContext,
  useSortable,
} from '@dnd-kit/sortable';
import { Flex, Tag } from 'antd';

interface Item {
  id: number;
  text: string;
}

interface DraggableTagProps {
  tag: Item;
}
```

```
const commonStyle: React.CSSProperties = {
  cursor: 'move',
  transition: 'unset', // Prevent element from shaking after drag
};

const DraggableTag: React.FC<DraggableTagProps> = (props) => {
  const { tag } = props;
  const { listeners, transform, transition, isDragging, setNodeRef } =
useSortable({ id: tag.id });

  const style = transform
    ? {
        ...commonStyle,
        transform: `translate3d(${transform.x}px, ${transform.y}px, 0)`,
        transition: isDragging ? 'unset' : transition, // Improve
performance/visual effect when dragging
      }
    : commonStyle;

  return (
    <Tag style={style} ref={setNodeRef} {...listeners}>
      {tag.text}
    </Tag>
  );
};

const App: React.FC = () => {
  const [items, setItems] = useState<Item[]>([
    { id: 1, text: 'Tag 1' },
    { id: 2, text: 'Tag 2' },
    { id: 3, text: 'Tag 3' },
  ]);

  const sensors = useSensors(useSensor(PointerSensor));

  const handleDragEnd = (event: DragEndEvent) => {
    const { active, over } = event;
    if (!over) {
      return;
    }
    if (active.id !== over.id) {
      setItems((data) => {
        const oldIndex = data.findIndex((item) => item.id === active.id);
        const newIndex = data.findIndex((item) => item.id === over.id);
        return arrayMove(data, oldIndex, newIndex);
```

```
      });
    }
  };

  return (
    <DndContext sensors={sensors} onDragEnd={handleDragEnd}
collisionDetection={closestCenter}>
      <SortableContext items={items} strategy=
{horizontalListSortingStrategy}>
        <Flex gap="4px 0" wrap>
          {items.map<React.ReactNode>((item) => (
            <DraggableTag tag={item} key={item.id} />
          ))}
        </Flex>
      </SortableContext>
    </DndContext>
  );
};

export default App;
```

Component Token

Debug

```
import React from 'react';
import { CloseCircleOutlined, SyncOutlined } from '@ant-design/icons';
import { ConfigProvider, Flex, Tag } from 'antd';

const App: React.FC = () => (
  <ConfigProvider
    theme={{ components: { Tag: { defaultBg: '#f9f0ff', defaultColor:
'#4b34d3' } } }}
  >
    <Flex gap="4px 0" wrap>
      <Tag>
        <a
          href="https://github.com/ant-design/ant-design/issues/1862"
          target="_blank"
          rel="noreferrer"
        >
          Link
        </a>
      </Tag>
      <Tag bordered={false}>
```

```
        <a
          href="https://github.com/ant-design/ant-design/issues/1862"
          target="_blank"
          rel="noreferrer"
        >
          Link
        </a>
      </Tag>
      <Tag closable color="magenta">
        Tag 2
      </Tag>
      <Tag icon={<CloseCircleOutlined />} color="error">
        error
      </Tag>
      <Tag color="red-inverse">red</Tag>
      <Tag bordered={false} color="magenta">
        magenta
      </Tag>
      <Tag icon={<SyncOutlined spin />} color="processing">
        processing
      </Tag>
    </Flex>
  </ConfigProvider>
);

export default App;
```

API

Common props ref：Common props

Tag

Property	Description	Type	Default	Version
closeIcon	Custom close icon. 5.7.0: close button will be hidden when setting to `null` or `false`	ReactNode	false	4.4.0
color	Color of the Tag	string	-	
icon	Set the icon of tag	ReactNode	-	

bordered	Whether has border style	boolean	true	5.4.0
onClose	Callback executed when tag is closed	(e: React.MouseEvent<HTMLElement, MouseEvent>) => void	-	

Tag.CheckableTag

Property	Description	Type	Default
checked	Checked status of Tag	boolean	false
onChange	Callback executed when Tag is checked/unchecked	(checked) => void	-

Design Token