Tag

Tag for categorizing or markup.

When To Use

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

Examples

Tag 1 Link Tag 2 ✕ Prevent Default ✕

Basic

Usage of basic Tag, and it could be closable by set `closable` property. Closable Tag supports `onClose` events.

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

const log = (e: React.MouseEvent<HTMLElement>)
  console.log(e);
};

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

const App: React.FC = () => (
  <Space size={[0, 8]} wrap>
    <Tag>Tag 1</Tag>
    <Tag>
      <a href="https://github.com/ant-design/
    </Tag>
    <Tag closable onClose={log}>
      Tag 2
    </Tag>
    <Tag closable onClose={preventDefault}>
      Prevent Default
    </Tag>
  </Space>
);

export default App;
```

Presets

magenta red volcano orange gold lime green cyan blue geekblue purple

Custom

#f50 #2db7f5 #87d068 #108ee9

Colorful Tag

We preset a series of colorful tag styles for use in different situations. You can also set it to a hex color string for custom color.

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

const App: React.FC = () => (
  <>
    <Divider orientation="left">Presets</Divi
    <Space size={[0, 8]} 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>
    </Space>
    <Divider orientation="left">Custom</Divide
    <Space size={[0, 8]} wrap>
      <Tag color="#f50">#f50</Tag>
      <Tag color="#2db7f5">#2db7f5</Tag>
      <Tag color="#87d068">#87d068</Tag>
      <Tag color="#108ee9">#108ee9</Tag>
    </Space>
  </>
);

export default App;
```

Unremovable   Tag 2 ×   Tag 3 ×   + New Tag

Add & Remove Dynamically

Generating a set of Tags by array, you can add and remove dynamically.

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

const App: React.FC = () => {
  const { token } = theme.useToken();
  const [tags, setTags] = useState(['Unremovab
  const [inputVisible, setInputVisible] = useS
  const [inputValue, setInputValue] = useState
  const [editInputIndex, setEditInputIndex] =
  const [editInputValue, setEditInputValue] =
  const inputRef = useRef<InputRef>(null);
  const editInputRef = useRef<InputRef>(null);

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

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

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

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

  const handleInputChange = (e: React.ChangeE
    setInputValue(e.target.value);
  };

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

  const handleEditInputChange = (e: React.Chan
    setEditInputValue(e.target.value);
  };

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

  const tagInputStyle: React.CSSProperties = {
    width: 78,
    verticalAlign: 'top',
  };
```

Categories: Movies   Books   Music   Sports

Checkable

`CheckableTag` works like Checkbox, click it to toggle checked state.

> it is an absolute controlled component and has no uncontrolled mode.

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

const { CheckableTag } = Tag;

const tagsData = ['Movies', 'Books', 'Music',

const App: React.FC = () => {
  const [selectedTags, setSelectedTags] = useS

  const handleChange = (tag: string, checked:
    const nextSelectedTags = checked
      ? [...selectedTags, tag]
      : selectedTags.filter((t) => t !== tag);
    console.log('You are interested in: ', nex
    setSelectedTags(nextSelectedTags);
  };

  return (
    <>
      <span style={{ marginRight: 8 }}>Categor
      <Space size={[0, 8]} wrap>
        {tagsData.map((tag) => (
          <CheckableTag
            key={tag}
            checked={selectedTags.includes(tag
            onChange={(checked) => handleChang
          >
            {tag}
          </CheckableTag>
        ))}
      </Space>
    </>
  );
};

export default App;
```

Tag 1 × Tag 2 × Tag 3 ×

+ New Tag

Animate

Animating the Tag by using rc-tween-one.

```
import React, { useEffect, useRef, useState }
import { PlusOutlined } from '@ant-design/ico
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', '
  const [inputVisible, setInputVisible] = use
  const [inputValue, setInputValue] = useStat
  const inputRef = useRef<InputRef>(null);

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

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

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

  const handleInputChange = (e: React.ChangeE
    setInputValue(e.target.value);
  };

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

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

  const tagChild = tags.map(forMap);

  const tagPlusStyle = {
```

Twitter Youtube Facebook

LinkedIn

Icon

`Tag` components can contain an `Icon`. This is done by setting the `icon` property or placing an `Icon` component within the `Tag`.

If you want specific control over the positioning and placement of the `Icon`, then that should be done by placing the `Icon` component within the `Tag` rather than using the `icon` property.

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

const App: React.FC = () => (
  <Space size={[0, 8]} wrap>
    <Tag icon={<TwitterOutlined />} color="#55
      Twitter
    </Tag>
    <Tag icon={<YoutubeOutlined />} color="#c
      Youtube
    </Tag>
    <Tag icon={<FacebookOutlined />} color="#3
      Facebook
    </Tag>
    <Tag icon={<LinkedinOutlined />} color="#5
      LinkedIn
    </Tag>
  </Space>
);

export default App;
```

Without icon

success processing error warning
default

With icon

success processing error
warning waiting stop

Status Tag

We preset five different colors, you can set color property such as `success`, `processing`, `error`, `default` and `warning` to indicate specific status.

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

const App: React.FC = () => (
  <>
    <Divider orientation="left">Without icon</
    <Space size={[0, 8]} 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>
    </Space>
    <Divider orientation="left">With icon</Di
    <Space size={[0, 8]} wrap>
      <Tag icon={<CheckCircleOutlined />} colo
        success
      </Tag>
      <Tag icon={<SyncOutlined spin />} color=
        processing
      </Tag>
      <Tag icon={<CloseCircleOutlined />} colo
        error
      </Tag>
      <Tag icon={<ExclamationCircleOutlined />
        warning
      </Tag>
      <Tag icon={<ClockCircleOutlined />} colo
        waiting
      </Tag>
      <Tag icon={<MinusCircleOutlined />} colo
        stop
      </Tag>
    </Space>
  </>
);

export default App;
```

Tag 1 Tag 2 Tag 3 × Tag 4 ×

magenta red volcano orange gold
lime green cyan blue geekblue
purple

borderless

borderless.

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

const App: React.FC = () => (
  <>
    <Space size={[0, 'small']} 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>
    </Space>
    <Divider />
    <Space size={[0, 'small']} 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>
    </Space>
  </>
);

export default App;
```

API

Tag

Property	Description	Type	Default	Version
closable	Whether the Tag can be closed	boolean	false	
closeIcon	Custom close icon	ReactNode	-	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) => 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

▾ Global Token

Token Name	Description	Type	Default Value
colorBorder	Default border color, used to separate different elements, such as: form separator, card separator, etc.	string	□ #d9d9d9
colorError	Used to represent the visual elements of the operation failure, such as the error Button, error Result component, etc.	string	□ #ff4d4f

Token Name	Description	Type	Default Value
colorErrorBg	The background color of the error state.	`string`	`#fff2f0`
colorErrorBorder	The border color of the error state.	`string`	`#ffccc7`
colorFillQuaternary	The weakest level of fill color is suitable for color blocks that are not easy to attract attention, such as zebra stripes, color blocks that distinguish boundaries, etc.	`string`	`rgba(0, 0, 0, 0.02)`
colorFillSecondary	The second level of fill color can outline the shape of the element more clearly, such as Rate, Skeleton, etc. It can also be used as the Hover state of the third level of fill color, such as Table, etc.	`string`	`rgba(0, 0, 0, 0.06)`
colorFillTertiary	The third level of fill color is used to outline the shape of the element, such as Slider, Segmented, etc. If there is no emphasis requirement, it is recommended to use the third level of fill color as the default fill color.	`string`	`rgba(0, 0, 0, 0.04)`
colorInfo	Used to represent the operation information of the Token sequence, such as Alert, Tag, Progress, and other components use these map tokens.	`string`	`#1677ff`
colorInfoBg	Light background color of information color.	`string`	`#e6f4ff`
colorInfoBorder	Border color of information color.	`string`	`#91caff`
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.	`string`	`#1677ff`
colorPrimaryActive	Dark active state under the main color gradient.	`string`	`#0958d9`
colorPrimaryHover	Hover state under the main color gradient.	`string`	`#4096ff`
colorSuccess	Used to represent the token sequence of operation success, such as Result, Progress and other components will use these map tokens.	`string`	`#52c41a`
colorSuccessBg	Light background color of success color, used for Tag and Alert success state background color	`string`	`#f6ffed`
colorSuccessBorder	Border color of success color, used for Tag and Alert success state border color	`string`	`#b7eb8f`
colorText	Default text color which comply with W3C standards, and this color is also the darkest neutral color.	`string`	`rgba(0, 0, 0, 0.88)`

Token Name	Description	Type	Default Value
colorTextDescription	Control the font color of text description.	`string`	□ `rgba(0, 0, 0, 0.45)`
colorTextHeading	Control the font color of heading.	`string`	□ `rgba(0, 0, 0, 0.88)`
colorTextLightSolid	Control the highlight color of text with background color, such as the text in Primary Button components.	`string`	□ `#fff`
colorWarning	Used to represent the warning map token, such as Notification, Alert, etc. Alert or Control component(like Input) will use these map tokens.	`string`	□ `#faad14`
colorWarningBg	The background color of the warning state.	`string`	□ `#fffbe6`
colorWarningBorder	The border color of the warning state.	`string`	□ `#ffe58f`
borderRadiusSM	SM size border radius, used in small size components, such as Button, Input, Select and other input components in small size	`number`	4
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
fontSizeIcon	Control the font size of operation icon in Select, Cascader, etc. Normally same as fontSizeSM.	`number`	12
fontSizeSM	Small font size	`number`	12
lineHeight	Line height of text.	`number`	1.5714285714285714
lineType	Border style of base components	`string`	solid
lineWidth	Border width of base components	`number`	1
marginXS	Control the margin of an element, with a small size.	`number`	8
motionDurationMid	Motion speed, medium speed. Used for medium element animation interaction.	`string`	0.2s
paddingXXS	Control the extra extra small padding of the element.	`number`	4