Notification

Display a notification message globally.

When To Use

To display a notification message at any of the four corners of the viewport. Typically it can be used in the following cases:

- A notification with complex content.
- A notification providing a feedback based on the user interaction. Or it may show some details about upcoming steps the user may have to follow.
- A notification that is pushed by the application.

Examples

Open the notification box

Basic

The simplest usage that close the notification box after 4.5s.

```
import React from 'react';
import { Button, notification } from 'antd';

const openNotification = () => {
  notification.open({
    message: 'Notification Title',
    description:
      'This is the content of the notificatio
    onClick: () => {
      console.log('Notification Clicked!');
    },
  });
};
const App: React.FC = () => (
  <Button type="primary" onClick={openNotifica
    Open the notification box
  </Button>
);

export default App;
```

topLeft topRight

bottomLeft bottomRight

Hooks usage (recommended)

Use `notification.useNotification` to get `contextHolder` with context accessible issue. Please note that, we recommend to use top level registration instead of `notification` static method, because static method cannot consume context, and ConfigProvider data will not work.

```
import React, { useMemo } from 'react';
import {
  RadiusBottomleftOutlined,
  RadiusBottomrightOutlined,
  RadiusUpleftOutlined,
  RadiusUprightOutlined,
} from '@ant-design/icons';
import { Button, Divider, notification, Space
import type { NotificationPlacement } from 'ar

const Context = React.createContext({ name: 'D

const App: React.FC = () => {
  const [api, contextHolder] = notification.us

  const openNotification = (placement: Notific
    api.info({
      message: `Notification ${placement}`,
      description: <Context.Consumer>{({ name
      placement,
    });
  };

  const contextValue = useMemo(() => ({ name:

  return (
    <Context.Provider value={contextValue}>
      {contextHolder}
      <Space>
        <Button type="primary" onClick={() =>
          <RadiusUpleftOutlined />
          topLeft
        </Button>
        <Button type="primary" onClick={() =>
          <RadiusUprightOutlined />
          topRight
        </Button>
      </Space>
      <Divider />
      <Space>
        <Button type="primary" onClick={() =>
          <RadiusBottomleftOutlined />
          bottomLeft
        </Button>
        <Button type="primary" onClick={() =>
          <RadiusBottomrightOutlined />
          bottomRight
        </Button>
      </Space>
    </Context.Provider>
  );
};
```

Success Info Warning Error

Notification with icon

A notification box with a icon at the left side.

```
import React from 'react';
import { Button, notification, Space } from 'c

type NotificationType = 'success' | 'info' | '

const App: React.FC = () => {
  const [api, contextHolder] = notification.us

  const openNotificationWithIcon = (type: Noti
    api[type]({
      message: 'Notification Title',
      description:
        'This is the content of the notificati
    });
  };

  return (
    <>
      {contextHolder}
      <Space>
        <Button onClick={() => openNotificatio
        <Button onClick={() => openNotificatio
        <Button onClick={() => openNotificatio
        <Button onClick={() => openNotificatio
      </Space>
    </>
  );
};

export default App;
```

Open the notification box

Duration after which the notification box is closed

notification stays open. After the duration time elapses, the notification closes automatically. If not specified, default value is 4.5 seconds. If you set the value to 0, the notification box will never close automatically.

```
import React from 'react';
import { Button, notification } from 'antd';

const App: React.FC = () => {
  const [api, contextHolder] = notification.us

  const openNotification = () => {
    api.open({
      message: 'Notification Title',
      description:
        'I will never close automatically. Thi
      duration: 0,
    });
  };

  return (
    <>
      {contextHolder}
      <Button type="primary" onClick={openNoti
        Open the notification box
      </Button>
    </>
  );
};

export default App;
```

Open the notification box

Customized Icon

The icon can be customized to any react node.

```
import React from 'react';
import { SmileOutlined } from '@ant-design/icc
import { Button, notification } from 'antd';

const App: React.FC = () => {
  const [api, contextHolder] = notification.us

  const openNotification = () => {
    api.open({
      message: 'Notification Title',
      description:
        'This is the content of the notificati
      icon: <SmileOutlined style={{ color: '#1
    });
  };

  return (
    <>
      {contextHolder}
      <Button type="primary" onClick={openNoti
        Open the notification box
      </Button>
    </>
  );
};

export default App;
```

Open the notification box

Custom close button

To customize the style or font of the close button.

```
import React from 'react';
import { Button, notification, Space } from '

const close = () => {
  console.log(
    'Notification was closed. Either the close
  );
};

const App: React.FC = () => {
  const [api, contextHolder] = notification.us

  const openNotification = () => {
    const key = `open${Date.now()}`;
    const btn = (
      <Space>
        <Button type="link" size="small" onCli
          Destroy All
        </Button>
        <Button type="primary" size="small" or
          Confirm
        </Button>
      </Space>
    );
    api.open({
      message: 'Notification Title',
      description:
        'A function will be be called after th
      btn,
      key,
      onClose: close,
    });
  };

  return (
    <>
      {contextHolder}
      <Button type="primary" onClick={openNoti
        Open the notification box
      </Button>
    </>
  );
};

export default App;
```

Open the notification box

Customized style

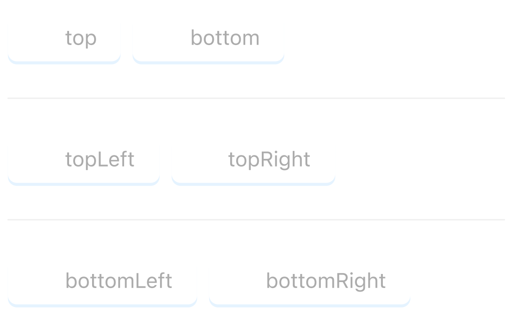
The style and className are available to customize Notification.

```
import React from 'react';
import { Button, notification } from 'antd';

const App: React.FC = () => {
  const [api, contextHolder] = notification.us

  const openNotification = () => {
    api.open({
      message: 'Notification Title',
      description:
        'This is the content of the notificati
      className: 'custom-class',
      style: {
        width: 600,
      },
    });
  };

  return (
    <>
      {contextHolder}
      <Button type="primary" onClick={openNoti
        Open the notification box
      </Button>
    </>
  );
};
export default App;
```



Placement

A notification box can appear from the `topRight`, `bottomRight`, `bottomLeft` or `topLeft` of the viewport via `placement`.

```
import React from 'react';
import {
  BorderBottomOutlined,
  BorderTopOutlined,
  RadiusBottomleftOutlined,
  RadiusBottomrightOutlined,
  RadiusUpleftOutlined,
  RadiusUprightOutlined,
} from '@ant-design/icons';
import { Button, Divider, notification, Space
import type { NotificationPlacement } from 'ar

const App: React.FC = () => {
  const [api, contextHolder] = notification.us

  const openNotification = (placement: Notific
    api.info({
      message: `Notification ${placement}`,
      description:
        'This is the content of the notificati
      placement,
    });
  };

  return (
    <>
      {contextHolder}
      <Space>
        <Button type="primary" onClick={() =>
          top
        </Button>
        <Button
          type="primary"
          onClick={() => openNotification('bot
          icon={<BorderBottomOutlined />}
        >
          bottom
        </Button>
      </Space>
      <Divider />
      <Space>
        <Button
          type="primary"
          onClick={() => openNotification('top
          icon={<RadiusUpleftOutlined />}
        >
          topLeft
        </Button>
        <Button
          type="primary"
          onClick={() => openNotification('top
          icon={<RadiusUprightOutlined />}
        >
```

Open the notification box

Update Message Content

Update content with unique key.

```
import React from 'react';
import { Button, notification } from 'antd';

const key = 'updatable';

const App: React.FC = () => {
  const [api, contextHolder] = notification.us
  const openNotification = () => {
    api.open({
      key,
      message: 'Notification Title',
      description: 'description.',
    });

    setTimeout(() => {
      api.open({
        key,
        message: 'New Title',
        description: 'New description.',
      });
    }, 1000);
  };

  return (
    <>
      {contextHolder}
      <Button type="primary" onClick={openNoti
        Open the notification box
      </Button>
    </>
  );
};

export default App;
```

API

- `notification.success(config)`
- `notification.error(config)`
- `notification.info(config)`
- `notification.warning(config)`
- `notification.open(config)`
- `notification.destroy(key?: String)`

The properties of config are as follows:

Property	Description	Type
bottom	Distance from the bottom of the viewport, when `placement` is `bottomRight` or `bottomLeft` (unit: pixels)	number
btn	Customized close button	ReactNode
className	Customized CSS class	string

Property	Description	Type
closeIcon	Custom close icon	ReactNode
description	The content of notification box (required)	ReactNode
duration	Time in seconds before Notification is closed. When set to 0 or null, it will never be closed automatically	number
getContainer	Return the mount node for Notification	() => HTMLNode
icon	Customized icon	ReactNode
key	The unique identifier of the Notification	string
message	The title of notification box (required)	ReactNode
placement	Position of Notification, can be one of `topLeft` `topRight` `bottomLeft` `bottomRight`	string
style	Customized inline style	[CSSProperties]()
top	Distance from the top of the viewport, when `placement` is `topRight` or `topLeft` (unit: pixels)	number
onClick	Specify a function that will be called when the notification is clicked	function
onClose	Trigger when notification closed	function
props	An object that can contain `data-*`, `aria-*`, or `role` props, to be put on the notification `div`. This currently only allows `data-testid` instead of `data-*` in TypeScript. See https://github.com/microsoft/TypeScript/issues/28960.	Object

`notification` also provides a global `config()` method that can be used for specifying the default options. Once this method is used, all the notification boxes will take into account these globally defined options when displaying.

Global configuration

`notification.config(options)`

> When you use `ConfigProvider` for global configuration, the system will automatically start RTL mode by default.(4.3.0+)
>
> When you want to use it alone, you can start the RTL mode through the following settings.

notification.config

```
notification.config({
  placement: 'bottomRight',
  bottom: 50,
  duration: 3,
  rtl: true,
});
```

Property	Description	Type	Default	Version
bottom	Distance from the bottom of the viewport, when `placement` is `bottomRight` or `bottomLeft` (unit: pixels)	number	24	
closeIcon	Custom close icon	ReactNode	-	
duration	Time in seconds before Notification is closed. When set to 0 or null, it will never be closed automatically	number	4.5	
getContainer	Return the mount node for Notification	() => HTMLNode	() => document.body	
placement	Position of Notification, can be one of `topLeft` `topRight` `bottomLeft` `bottomRight`	string	`topRight`	
rtl	Whether to enable RTL mode	boolean	false	
top	Distance from the top of the viewport, when `placement` is `topRight` or `topLeft` (unit: pixels)	number	24	
maxCount	Max Notification show, drop oldest if exceed limit	number	-	4.17.0

Design Token

▾ Global Token

Token Name	Description	Type	Default Value
colorBgElevated	Container background color of the popup layer, in dark mode the color value of this token will be a little brighter than `colorBgContainer`. E.g: modal, pop-up, menu, etc.	`string`	`#ffffff`
colorError	Used to represent the visual elements of the operation failure, such as the error Button, error Result component, etc.	`string`	`#ff4d4f`
colorIcon	Weak action. Such as `allowClear` or Alert close button	`string`	`rgba(0, 0, 0, 0.45)`
colorIconHover	Weak action hover color. Such as `allowClear` or Alert close button	`string`	`rgba(0, 0, 0, 0.88)`
colorInfo	Used to represent the operation information of the Token sequence, such as Alert, Tag, Progress, and	`string`	`#1677ff`

Token Name	Description	Type	Default Value
	other components use these map tokens.		
colorSuccess	Used to represent the token sequence of operation success, such as Result, Progress and other components will use these map tokens.	`string`	□ `#52c41a`
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)`
colorTextHeading	Control the font color of heading.	`string`	□ `rgba(0, 0, 0, 0.88)`
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`
borderRadiusLG	LG size border radius, used in some large border radius components, such as Card, Modal and other components.	`number`	8
borderRadiusSM	SM size border radius, used in small size components, such as Button, Input, Select and other input components in small size	`number`	4
boxShadow	Control the box shadow style of an element.	`string`	0 6px 16px 0 rgba(0, 0, 0, 0.08), 0 3px 6px -4px rgba(0, 0, 0, 0.12), 0 9px 28px 8px rgba(0, 0, 0, 0.05)
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
lineHeight	Line height of text.	`number`	1.5714285714285714
lineHeightLG	Line height of large text.	`number`	1.5
margin	Control the margin of an element, with a medium size.	`number`	16

Token Name	Description	Type	Default Value
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
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
motionEaseInOut	Preset motion curve.	`string`	cubic-bezier(0.645, 0.045, 0.355, 1)
paddingContentHorizontalLG	Control the horizontal padding of content element, suitable for large screen devices.	`number`	24
paddingLG	Control the large padding of the element.	`number`	24
paddingMD	Control the medium padding of the element.	`number`	20
wireframe	Used to change the visual effect of the component to wireframe, if you need to use the V4 effect, you need to enable the configuration item	`boolean`	false
zIndexPopupBase	Base zIndex of component like FloatButton, Affix which can be cover by large popup	`number`	1000

FAQ

Why I can not access context, redux, ConfigProvider `locale/prefixCls/theme` in notification?

antd will dynamic create React instance by `ReactDOM.render` when call notification methods. Whose context is different with origin code located context.

When you need context info (like ConfigProvider context), you can use `notification.useNotification` to get `api` instance and `contextHolder` node. And put it in your children:

```
const [api, contextHolder] = notification.useNotification();

return (
  <Context1.Provider value="Ant">
    {/* contextHolder is inside Context1 which means api will get value of Context1 */}
    {contextHolder}
    <Context2.Provider value="Design">
      {/* contextHolder is outside Context2 which means api will **not** get value of Context2 */
    </Context2.Provider>
  </Context1.Provider>
);
```

Note: You must insert `contextHolder` into your children with hooks. You can use origin method if you do not need context connection.

> App Package Component can be used to simplify the problem of `useNotification` and other methods that need to manually implant contextHolder.

How to set static methods prefixCls ?

You can config with `ConfigProvider.config`