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

Hooks usage (recommended)

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

type NotificationPlacement = NotificationArgsProps['placement'];

const Context = React.createContext({ name: 'Default' });

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

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

  const contextValue = useMemo(() => ({ name: 'Ant Design' }), []);

  return (
    <Context.Provider value={contextValue}>
      {contextHolder}
      <Space>
        <Button
```

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

export default App;
```

Duration after which the notification box is closed

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

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

  const openNotification = () => {
```

```
    api.open({
      message: 'Notification Title',
      description:
        'I will never close automatically. This is a purposely very very
long description that has many many characters and words.',
      duration: 0,
    });
  };

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

export default App;
```

Notification with icon

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

type NotificationType = 'success' | 'info' | 'warning' | 'error';

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

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

  return (
    <>
      {contextHolder}
      <Space>
        <Button onClick={() =>
openNotificationWithIcon('success')}>Success</Button>
```

```
        <Button onClick={() =>
openNotificationWithIcon('info')}>Info</Button>
        <Button onClick={() =>
openNotificationWithIcon('warning')}>Warning</Button>
        <Button onClick={() =>
openNotificationWithIcon('error')}>Error</Button>
      </Space>
    </>
  );
};

export default App;
```

Custom close button

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

const close = () => {
  console.log(
    'Notification was closed. Either the close button was clicked or
duration time elapsed.',
  );
};

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

  const openNotification = () => {
    const key = `open${Date.now()}`;
    const btn = (
      <Space>
        <Button type="link" size="small" onClick={() => api.destroy()}>
          Destroy All
        </Button>
        <Button type="primary" size="small" onClick={() =>
api.destroy(key)}>
          Confirm
        </Button>
      </Space>
    );
    api.open({
      message: 'Notification Title',
      description:
        'A function will be be called after the notification is closed
(automatically after the "duration" time of manually).',
```

```
      btn,
      key,
      onClose: close,
    });
  };

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

export default App;
```

Customized Icon

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

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

  const openNotification = () => {
    api.open({
      message: 'Notification Title',
      description:
        'This is the content of the notification. This is the content of
the notification. This is the content of the notification.',
      icon: <SmileOutlined style={{ color: '#108ee9' }} />,
    });
  };

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

```
export default App;
```

Placement

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

type NotificationPlacement = NotificationArgsProps['placement'];

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

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

  return (
    <>
      {contextHolder}
      <Space>
        <Button type="primary" onClick={() => openNotification('top')}
icon={<BorderTopOutlined />}>
          top
        </Button>
        <Button
          type="primary"
          onClick={() => openNotification('bottom')}
          icon={<BorderBottomOutlined />}
        >
          bottom
```

```
        </Button>
      </Space>
      <Divider />
      <Space>
        <Button
          type="primary"
          onClick={() => openNotification('topLeft')}
          icon={<RadiusUpleftOutlined />}
        >
          topLeft
        </Button>
        <Button
          type="primary"
          onClick={() => openNotification('topRight')}
          icon={<RadiusUprightOutlined />}
        >
          topRight
        </Button>
      </Space>
      <Divider />
      <Space>
        <Button
          type="primary"
          onClick={() => openNotification('bottomLeft')}
          icon={<RadiusBottomleftOutlined />}
        >
          bottomLeft
        </Button>
        <Button
          type="primary"
          onClick={() => openNotification('bottomRight')}
          icon={<RadiusBottomrightOutlined />}
        >
          bottomRight
        </Button>
      </Space>
    </>
  );
};

export default App;
```

Customized style

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

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

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

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

Update Message Content

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

const key = 'updatable';

const App: React.FC = () => {
  const [api, contextHolder] = notification.useNotification();
  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={openNotification}>
        Open the notification box
      </Button>
    </>
  );
};

export default App;
```

Stack

v5.10.0

```
import React, { useMemo } from 'react';
import { Button, Divider, InputNumber, notification, Space, Switch } from
'antd';

const Context = React.createContext({ name: 'Default' });

const App: React.FC = () => {
  const [enabled, setEnabled] = React.useState(true);
  const [threshold, setThreshold] = React.useState(3);
  const [api, contextHolder] = notification.useNotification({
    stack: enabled
      ? {
          threshold,
        }
      : false,
  });

  const openNotification = () => {
    api.open({
      message: 'Notification Title',
      description: `${Array.from(
        { length: Math.round(Math.random() * 5) + 1 },
        () => 'This is the content of the notification.',
```

```
      ).join('\n')}`,
      duration: null,
    });
  };

  const contextValue = useMemo(() => ({ name: 'Ant Design' }), []);

  return (
    <Context.Provider value={contextValue}>
      {contextHolder}
      <div>
        <Space size="large">
          <Space style={{ width: '100%' }}>
            <span>Enabled: </span>
            <Switch checked={enabled} onChange={(v) => setEnabled(v)} />
          </Space>
          <Space style={{ width: '100%' }}>
            <span>Threshold: </span>
            <InputNumber
              disabled={!enabled}
              value={threshold}
              step={1}
              min={1}
              max={10}
              onChange={(v) => setThreshold(v || 0)}
            />
          </Space>
        </Space>
        <Divider />
        <Button type="primary" onClick={openNotification}>
          Open the notification box
        </Button>
      </div>
    </Context.Provider>
  );
};

export default App;
```

Show with progress

v5.18.0

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

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

  const openNotification = (pauseOnHover: boolean) => () => {
    api.open({
      message: 'Notification Title',
      description:
        'This is the content of the notification. This is the content of
the notification. This is the content of the notification.',
      showProgress: true,
      pauseOnHover,
    });
  };

  return (
    <>
      {contextHolder}
      <Space>
        <Button type="primary" onClick={openNotification(true)}>
          Pause on hover
        </Button>
        <Button type="primary" onClick={openNotification(false)}>
          Don&apos;t pause on hover
        </Button>
      </Space>
    </>
  );
};

export default App;
```

Static Method (deprecated)

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

const openNotification = () => {
  notification.open({
    message: 'Notification Title',
    description:
      'This is the content of the notification. This is the content of the
notification. This is the content of the notification.',
    onClick: () => {
      console.log('Notification Clicked!');
    },
  });
```

```
};
const App: React.FC = () => (
  <Button type="primary" onClick={openNotification}>
    Open the notification box
  </Button>
);

export default App;
```

_InternalPanelDoNotUseOrYouWillBeFired

Debug

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

/** Test usage. Do not use in your production. */
const { _InternalPanelDoNotUseOrYouWillBeFired: InternalPanel } =
notification;

export default () => (
  <InternalPanel
    message="Hello World!"
    description="Hello World?"
    type="success"
    actions={
      <Button type="primary" size="small">
        My Button
      </Button>
    }
  />
);
```

API

Common props ref：Common props

- `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	
actions	Customized button group	ReactNode	-
~~btn~~	Customized close button group, please use `actions` instead	ReactNode	-
className	Customized CSS class	string	-
closeIcon	Custom close icon	ReactNode	tr
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	4
showProgress	Show progress bar for auto-closing notification	boolean	
pauseOnHover	keep the timer running or not on hover	boolean	tr
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 `top` | `topLeft` | `topRight` | `bottom` | `bottomLeft` | `bottomRight`	string	t
style	Customized inline style	CSSProperties	-
role	The semantics of notification content recognized by screen readers. The default value is `alert`. When set as the default value, the screen reader will promptly interrupt any ongoing content reading and prioritize the notification content for immediate attention.	`alert` | `status`	a
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	Object	-

	only allows `data-testid` instead of `data-*` in TypeScript. See https://github.com/microsoft/TypeScript/issues/28960.		

- `notification.useNotification(config)`

The properties of config are as follows:

Property	Description	Type	Default	Version
bottom	Distance from the bottom of the viewport, when `placement` is `bottom` `bottomRight` or `bottomLeft` (unit: pixels)	number	24	
closeIcon	Custom close icon	ReactNode	true	5.7.0: close button will be hidden when setting to null or false
getContainer	Return the mount node for Notification	() => HTMLNode	() => document.body	
placement	Position of Notification, can be one of `top` | `topLeft` | `topRight` | `bottom` | `bottomLeft` | `bottomRight`	string	`topRight`	
showProgress	Show progress bar for auto-closing notification	boolean		5.18.0
pauseOnHover	keep the timer running or not on hover	boolean	true	5.18.0
rtl	Whether to enable RTL mode	boolean	false	

stack	Notifications will be stacked when amount is over threshold	boolean | { threshold: number }	{ threshold: 3 }	5.10.0
top	Distance from the top of the viewport, when `placement` is `top` `topRight` or `topLeft` (unit: pixels)	number	24	
maxCount	Max Notification show, drop oldest if exceed limit	number	-	4.17.0

`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 `bottom` `bottomRight` or `bottomLeft` (unit: pixels)	number	24	

closeIcon	Custom close icon	ReactNode	true	5.7.0: close button will be hidden when setting to null or false
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, but still display at fullScreen	() => HTMLNode	() => document.body	
placement	Position of Notification, can be one of `top` `topLeft` `topRight` `bottom` `bottomLeft` `bottomRight`	string	`topRight`	
showProgress	Show progress bar for auto-closing notification	boolean		5.18.0
pauseOnHover	keep the timer running or not on hover	boolean	true	5.18.0
rtl	Whether to enable RTL mode	boolean	false	
top	Distance from the top of the viewport, when `placement` is `top` `topRight` or `topLeft` (unit: pixels)	number	24	
maxCount	Max Notification show, drop oldest if exceed limit	number	-	4.17.0

Design Token

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`