When To Use

On longer web pages, it's helpful to stick component into the viewport. This is common for menus and actions.

Please note that Affix should not cover other content on the page, especially when the size of the viewport is small.

> *Notes for developers*
>
> *After version `5.10.0`, we rewrite Affix use FC, Some methods of obtaining `ref` and calling internal instance methods will invalid.*

Examples

Basic

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

const App: React.FC = () => {
  const [top, setTop] = React.useState<number>(100);
  const [bottom, setBottom] = React.useState<number>(100);
  return (
    <>
      <Affix offsetTop={top}>
        <Button type="primary" onClick={() => setTop(top + 10)}>
          Affix top
        </Button>
      </Affix>
      <br />
      <Affix offsetBottom={bottom}>
        <Button type="primary" onClick={() => setBottom(bottom + 10)}>
          Affix bottom
        </Button>
      </Affix>
    </>
  );
};

export default App;
```

Callback

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

```
const App: React.FC = () => (
  <Affix offsetTop={120} onChange={(affixed) => console.log(affixed)}>
    <Button>120px to affix top</Button>
  </Affix>
);

export default App;
```

Container to scroll.

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

const containerStyle: React.CSSProperties = {
  width: '100%',
  height: 100,
  overflow: 'auto',
  boxShadow: '0 0 0 1px #1677ff',
  scrollbarWidth: 'thin',
  scrollbarGutter: 'stable',
};

const style: React.CSSProperties = {
  width: '100%',
  height: 1000,
};

const App: React.FC = () => {
  const [container, setContainer] = React.useState<HTMLDivElement | null>
(null);
  return (
    <div style={containerStyle} ref={setContainer}>
      <div style={style}>
        <Affix target={() => container}>
          <Button type="primary">Fixed at the top of container</Button>
        </Affix>
      </div>
    </div>
  );
};

export default App;
```

debug

Debug

```tsx
import React, { useState } from 'react';
import { Affix, Button } from 'antd';

const App: React.FC = () => {
  const [top, setTop] = useState(10);

  return (
    <div style={{ height: 10000 }}>
      <div>Top</div>
      <Affix offsetTop={top}>
        <div style={{ background: 'red' }}>
          <Button type="primary" onClick={() => setTop(top + 10)}>
            Affix top
          </Button>
        </div>
      </Affix>
      <div>Bottom</div>
    </div>
  );
};

export default App;
```

API

Common props ref：Common props

Property	Description	Type	Default
offsetBottom	Offset from the bottom of the viewport (in pixels)	number	-
offsetTop	Offset from the top of the viewport (in pixels)	number	0
target	Specifies the scrollable area DOM node	() => HTMLElement	() => window
onChange	Callback for when Affix state is changed	(affixed?: boolean) => void	-

Note: Children of `Affix` must not have the property `position: absolute`, but you can set `position: absolute` on `Affix` itself:

```tsx
<Affix style={{ position: 'absolute', top: y, left: x }}>...</Affix>
```

FAQ

When binding container with `target` in Affix, elements sometimes move out of the container.

We only listen to container scroll events for performance consideration. You can add custom listeners if you still want to: https://codesandbox.io/s/stupefied-maxwell-ophqnm?file=/index.js

Related issues： #3938 #5642 #16120

When Affix is used in a horizontal scroll container, the position of the element `left` is incorrect.

Affix is generally only applicable to areas with one-way scrolling, and only supports usage in vertical scrolling containers. If you want to use it in a horizontal container, you can consider implementing with the native `position: sticky` property.

Related issues： #29108