

Layout

Handling the overall layout of a page.

Specification

Size

The first level navigation is left aligned near a logo, and the secondary menu is right aligned.

- Top Navigation: the height of the first level navigation `64px` , the second level navigation `48px` .
- Top Navigation (for landing pages): the height of the first level navigation `80px` , the second level navigation `56px` .
- Calculation formula of a top navigation: `48+8n` .
- Calculation formula of an aside navigation: `200+8n` .

Interaction rules

- The first level navigation and the last level navigation should be distinguishable by visualization;
- The current item should have the highest priority of visualization;
- When the current navigation item is collapsed, the style of the current navigation item is applied to its parent level;
- The left side navigation bar has support for both the accordion and expanding styles; you can choose the one that fits your case the best.

Visualization rules

Style of a navigation should conform to its level.

- **Emphasis by colorblock**
When background color is a deep color, you can use this pattern for the parent level navigation item of the current page.
- **The highlight match stick**
When background color is a light color, you can use this pattern for the current page navigation item; we recommend using it for the last item of the navigation path.
- **Highlighted font**
From the visualization aspect, a highlighted font is stronger than colorblock; this pattern is often used for the parent level of the current item.
- **Enlarge the size of the font**
`12px` , `14px` is a standard font size of navigation's, `14px` is used for the first and the second level of the navigation. You can choose an appropriate font size regarding the level of your navigation.

Component Overview

- `Layout` : The layout wrapper, in which `Header` `Sider` `Content` `Footer` or `Layout` itself can be nested, and can be placed in any parent container.
- `Header` : The top layout with the default style, in which any element can be nested, and must be placed in `Layout` .
- `Sider` : The sidebar with default style and basic functions, in which any element can be nested, and must be placed in `Layout` .
- `Content` : The content layout with the default style, in which any element can be nested, and must be placed in `Layout` .

- `Footer` : The bottom layout with the default style, in which any element can be nested, and must be placed in `Layout` .

Based on `flex layout` , please pay attention to the [compatibility](#).

Examples

Header

Content

Footer

Header

Sider

Content

Footer

Header

Content

Sider

Footer

Sider

Header

[Home](#) / [List](#) / [App](#)

Content

Ant Design ©2023 Created by Ant UED

Header-Content-Footer

The most basic "header-content-footer" layout.

Generally, the mainnav is placed at the top of the page, and includes the logo, the first level navigation, and the secondary menu (users, settings, notifications) from left to right in it. We always put contents in a fixed size navigation (eg: `1200px`), the layout of the whole page is stable, it's not affected by viewing area.

Top-bottom structure is conform with the top-bottom viewing habit, it's a classical navigation pattern of websites. This pattern demonstrates efficiency in the main workarea, while using some vertical space. And because the horizontal space of the navigation is limited, this pattern is not suitable for cases when the first level navigation contains many elements or links.

```
import React from 'react';
import { Breadcrumb, Layout, Menu, theme } from 'antd';

const { Header, Content, Footer } = Layout;

const App: React.FC = () => {
  const {
    token: { colorBgContainer },
  } = theme.useToken();

  return (
    <Layout className="layout">
      <Header>
        <div className="logo" />
        <Menu
          theme="dark"
          mode="horizontal"
          defaultSelectedKeys={['2']}
          items={new Array(15).fill(null).map((_, index) => {
            const key = index + 1;
            return {
              key,
              label: `nav ${key}`,
            };
          })}
        />
      </Header>
      <Content style={{ padding: '0 50px' }}>
        <Breadcrumb style={{ margin: '16px 0' }}>
          <Breadcrumb.Item>Home</Breadcrumb.Item>
          <Breadcrumb.Item>List</Breadcrumb.Item>
          <Breadcrumb.Item>App</Breadcrumb.Item>
        </Breadcrumb>
        <div className="site-layout-content" style={{ background: colorBgContainer }}>
          Content
        </div>
      </Content>
    </Layout>
  );
};
```

🔍 subnav 1

option1

option2

option3

option4

📖 subnav 2

🔍 subnav 3

Home / List / App

Content

Header Sider 2 [🔗](#)

Both the top navigation and the sidebar, commonly used in application site.

```
import React from 'react';
import { LaptopOutlined, NotificationOutlined, UserOutlined } from '@ant-design/icons';
import type { MenuProps } from 'antd';
import { Breadcrumb, Layout, Menu, theme } from 'antd';

const { Header, Content, Sider } = Layout;


const items1: MenuProps['items'] = ['1', '2', '3'].map((key) => ({
  key,
  label: `nav ${key}`,
}));

const items2: MenuProps['items'] = [UserOutlined, LaptopOutlined, NotificationOutlined].map(
  (icon, index) => {
    const key = String(index + 1);

    return {
      key: `sub${key}`,
      icon: React.createElement(icon),
      label: `subnav ${key}`,

      children: new Array(4).fill(null).map((_, j) => {
        const subKey = index * 4 + j + 1;
        return {
          key: subKey,
          label: `option${subKey}`,
        };
      }),
    };
  },
);
```

Home / List / App

 subnav 1

option1

option2

option3

option4

 subnav 2 subnav 3

Content

Ant Design ©2023 Created by Ant UED

Header-Sider

Both the top navigation and the sidebar, commonly used in documentation site.

```
import React from 'react';
import { LaptopOutlined, NotificationOutlined, UserOutlined } from '@ant-design/icons';
import type { MenuProps } from 'antd';
import { Breadcrumb, Layout, Menu, theme } from 'antd';

const { Header, Content, Footer, Sider } = Layout;

const items1: MenuProps['items'] = ['1', '2', '3'].map((key) => ({
  key,
  label: `nav ${key}`,
}));

const items2: MenuProps['items'] = [UserOutlined, LaptopOutlined, NotificationOutlined].map(
  (icon, index) => {
    const key = String(index + 1);

    return {
      key: `sub${key}`,
      icon: React.createElement(icon),
      label: `subnav ${key}`,

      children: new Array(4).fill(null).map((_, j) => {
        const subKey = index * 4 + j + 1;
        return {
          key: subKey,
          label: `option${subKey}`,
        };
      });
    };
  }
);
```

Sider

Two-columns layout. The sider menu can be collapsed when horizontal space is limited.

Generally, the mainnav is placed on the left side of the page, and the secondary menu is placed on the top of the working area. Contents will adapt the layout to the viewing area to improve the horizontal space usage, while the layout of the whole page is not stable.

The level of the aside navigation is scalable. The first, second, and third level navigations could be present more fluently and relevantly, and aside navigation can be fixed, allowing the user to quickly switch and spot the current position, improving the user experience. However, this navigation occupies some horizontal space of the contents.

```
import React, { useState } from 'react';
import {
  DesktopOutlined,
  FileOutlined,
  PieChartOutlined,
  TeamOutlined,
  UserOutlined,
} from '@ant-design/icons';
import type { MenuProps } from 'antd';
import { Breadcrumb, Layout, Menu, theme } from 'antd';

const { Header, Content, Footer, Sider } = Layout;

type MenuItem = Required<MenuProps>['items'][number];

function getItem(
  label: React.ReactNode,
  key: React.Key,
  icon?: React.ReactNode,
  children?: MenuItem[],
): MenuItem {
  return {
    key,
    icon,
    children,
    label,
  } as MenuItem;
}

const items: MenuItem[] = [
  getItem('Option 1', '1', <PieChartOutlined />),
  getItem('Option 2', '2', <DesktopOutlined />),
  getItem('User', 'sub1', <UserOutlined />, [
    getItem('Tom', '3'),
    getItem('Bill', '4'),
    getItem('Alex', '5'),
  ]),
  getItem('Team', 'sub2', <TeamOutlined />, [getItem('Team 1', '6'), getItem('Team 2', '8')]),
  getItem('Files', '9', <FileOutlined />),
];
```


nav 1

nav 2

nav 3



Content

Custom trigger [↗](#)

If you want to use a customized trigger, you can hide the default one by setting `trigger={null}` .

```
import React, { useState } from 'react';
import {
  MenuFoldOutlined,
  MenuUnfoldOutlined,
  UploadOutlined,
  UserOutlined,
  VideoCameraOutlined,
} from '@ant-design/icons';
import { Layout, Menu, Button, theme } from 'antd';

const { Header, Sider, Content } = Layout;

const App: React.FC = () => {
  const [collapsed, setCollapsed] = useState(false);
  const {
    token: { colorBgContainer },
  } = theme.useToken();

  return (
    <Layout>
      <Sider trigger={null} collapsible collapsed={collapsed}>
        <div className="logo" />
        <Menu
          theme="dark"
          mode="inline"
          defaultSelectedKeys={['1']}
          items={[
            {
              key: '1',
              icon: <UserOutlined />,
              label: 'nav 1',
            },
            {
              key: '2',
              icon: <VideoCameraOutlined />,
              label: 'nav 2',
            },
            {
              key: '3',
              icon: <UploadOutlined />,
            },
          ]}
        />
      </Sider>
      <Content>
        <div>Content</div>
      </Content>
    </Layout>
  );
};
```

nav 1

nav 2

nav 3

nav 4

content

Ant Design ©2023 Created by Ant UED

Responsive

Layout.Sider supports responsive layout.

Note: You can get a responsive layout by setting `breakpoint`, the Sider will collapse to the width of `collapsedWidth` when window width is below the `breakpoint`. And a special trigger will appear if the `collapsedWidth` is set to 0.

```
import React from 'react';
import { UploadOutlined, UserOutlined, VideoCameraOutlined } from '@ant-design/icons';
import { Layout, Menu, theme } from 'antd';
```

```
const { Header, Content, Footer, Sider } = Layout;
```

```
const App: React.FC = () => {
  const {
    token: { colorBgContainer },
  } = theme.useToken();
```

```
  return (
    <Layout>
      <Sider
        breakpoint="lg"
        collapsedWidth="0"
        onBreakpoint={(broken) => {
          console.log(broken);
        }}
        onCollapse={(collapsed, type) => {
          console.log(collapsed, type);
        }}
      >
```

```
      <div className="logo" />
```

```
      <Menu
```

```
        theme="dark"
```

```
        mode="inline"
```

```
        defaultSelectedKeys={['1', '2']}
```

Fixed Header

Fixed Header is generally used to fix the top navigation to facilitate page switching.

```
import React from 'react';
import { Breadcrumb, Layout, Menu, theme } from 'antd';

const { Header, Content, Footer } = Layout;

const App: React.FC = () => {
  const {
    token: { colorBgContainer },
  } = theme.useToken();

  return (
    <Layout>
      <Header style={{ position: 'sticky', top: 0, zIndex: 1, width: '100%' }}>
        <div
          style={{
            float: 'left',
            width: 120,
            height: 31,
            margin: '16px 24px 16px 0',
            background: 'rgba(255, 255, 255, 0.2)',
          }}
        />
        <Menu
          theme="dark"
          mode="horizontal"
          defaultSelectedKeys={['2']}
          items={new Array(3).fill(null).map((_, index) => ({
            key: String(index + 1),
            label: `nav ${index + 1}`,
          }))}
        />
      </Header>
      <Content className="site-layout" style={{ padding: '0 50px' }}>
        <Breadcrumb style={{ margin: '16px 0' }}>
          <Breadcrumb.Item>Home</Breadcrumb.Item>
          <Breadcrumb.Item>List</Breadcrumb.Item>
          <Breadcrumb.Item>App</Breadcrumb.Item>
        </Breadcrumb>
        <div style={{ padding: 24, minHeight: 380, background: colorBgContainer }}>Content</div>
      </Content>
      <Footer style={{ textAlign: 'center' }}>Ant Design ©2023 Created by Ant UED</Footer>
    </Layout>
  );
};

export default App;
```

Fixed Sider

When dealing with long content, a fixed sider can provide a better user experience.

```
import React from 'react';
import {
  AppstoreOutlined,
  BarChartOutlined,
  CloudOutlined,
  ShopOutlined,
  TeamOutlined,
  UploadOutlined,
  UserOutlined,
  VideoCameraOutlined,
} from '@ant-design/icons';
import type { MenuProps } from 'antd';
import { Layout, Menu, theme } from 'antd';

const { Header, Content, Footer, Sider } = Layout;

const items: MenuProps['items'] = [
  UserOutlined,
  VideoCameraOutlined,
  UploadOutlined,
  BarChartOutlined,
  CloudOutlined,
  AppstoreOutlined,
  TeamOutlined,
  ShopOutlined,
].map((icon, index) => ({
  key: String(index + 1),
  icon: React.createElement(icon),
  label: `nav ${index + 1}`,
}));

const App: React.FC = () => {
  const {
    token: { colorBgContainer },
  } = theme.useToken();

  return (
    <Layout hasSider>
      <Sider
        style={{
          overflow: 'auto',
          height: '100vh',
          position: 'fixed',
          left: 0,
          top: 0,
          bottom: 0,
        }}
      >
        <div style={{ height: 32, margin: 16, background: 'rgba(255, 255, 255, 0.2)' }} />
        <Menu theme="dark" mode="inline" defaultSelectedKeys={['4']} items={items} />
      </Sider>
    </Layout>
  );
};
```

```

</style>
<Layout className="site-layout" style={{ marginLeft: 200 }}>
  <Header style={{ padding: 0, background: colorBgContainer }} />
  <Content style={{ margin: '24px 16px 0', overflow: 'initial' }}>
    <div style={{ padding: 24, textAlign: 'center', background: colorBgContainer }}>
      <p>long content</p>
      {
        // indicates very long content
        Array.from({ length: 100 }, (_, index) => (
          <React.Fragment key={index}>
            {index % 20 === 0 && index ? 'more' : '...'}
            <br />
          </React.Fragment>
        ))
      }
    </div>
  </Content>
  <Footer style={{ textAlign: 'center' }}>Ant Design ©2023 Created by Ant UED</Footer>
</Layout>
</Layout>
);
};

export default App;

```


API

```
<Layout>
  <Header>header</Header>
  <Layout>
    <Sider>left sidebar</Sider>
    <Content>main content</Content>
    <Sider>right sidebar</Sider>
  </Layout>
  <Footer>footer</Footer>
</Layout>
```

Layout

The wrapper.

Property	Description	Type	Default
className	Container className	string	-
hasSider	Whether contain Sider in children, don't have to assign it normally. Useful in SSR avoid style flickering	boolean	-
style	To customize the styles	CSSProperties	-

Layout.Sider

The sidebar.

Property	Description	Type	Default
breakpoint	Breakpoints of the responsive layout	<div>xs sm md lg xl xxl</div>	-
className	Container className	string	-
collapsed	To set the current status	boolean	-
collapsedWidth	Width of the collapsed sidebar, by setting to 0 a special trigger will appear	number	80
collapsible	Whether can be collapsed	boolean	false
defaultCollapsed	To set the initial status	boolean	false
reverseArrow	Reverse direction of arrow, for a sider that expands from the right	boolean	false
style	To customize the styles	CSSProperties	-
theme	Color theme of the sidebar	<div>light dark</div>	<div>dark</div>
trigger	Specify the customized trigger, set to null to hide the trigger	ReactNode	-
width	Width of the sidebar	number string	200
zeroWidthTriggerStyle	To customize the styles of the special trigger that appears when <code>collapsedWidth</code> is 0	object	-
onBreakpoint	The callback function, executed when breakpoints changed	(broken) => {}	-
onCollapse	The callback function, executed by clicking the trigger or activating the responsive layout	(collapsed, type) => {}	-

breakpoint width

```
{
  xs: '480px',
  sm: '576px',
  md: '768px',
  lg: '992px',
  xl: '1200px',
  xxl: '1600px',
}
```

Design Token

▼ Global Token

Token Name	Description	Type	Default Value
colorBgContainer	Container background color, e.g: default button, input box, etc. Be sure not to confuse this with `colorBgElevated`.	string	<input type="checkbox"/> #ffffff
colorBgLayout	This color is used for the background color of the overall layout of the page. This token will only be used when it is necessary to be at the B1 visual level in the page. Other usages are wrong.	string	<input type="checkbox"/> #f5f5f5
colorText	Default text color which comply with W3C standards, and this color is also the darkest neutral color.	string	<input type="checkbox"/> 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	<input type="checkbox"/> #fff
borderRadius	Border radius of base components	number	6
controlHeight	The height of the basic controls such as buttons and input boxes in Ant Design	number	32
controlHeightLG	LG component height	number	40
controlHeightSM	SM component height	number	24
fontSize	The most widely used font size in the design system, from which the text gradient will be derived.	number	14
fontSizeXL	Super large font size	number	20
marginXXS	Control the margin of an element, with the smallest size.	number	4
motionDurationMid	Motion speed, medium speed. Used for medium element animation interaction.	string	0.2s
motionDurationSlow	Motion speed, slow speed. Used for large element animation interaction.	string	0.3s

