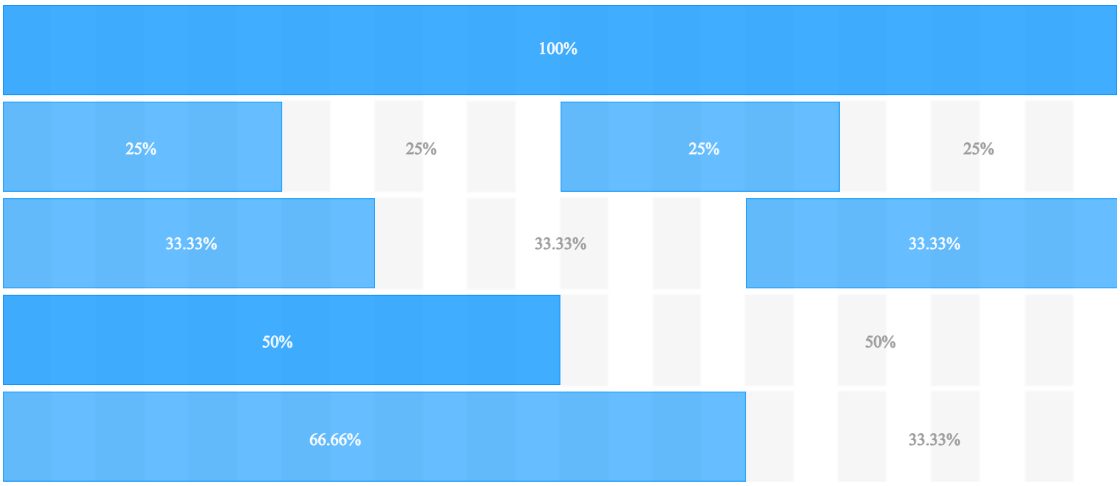
Grid

24 Grids System.

Design concept



In most business situations, Ant Design needs to solve a lot of information storage problems within the design area, so based on 12 Grids System, we divided the design area into 24 sections.

We name the divided area 'box'. We suggest four boxes for horizontal arrangement at most, one at least. Boxes are proportional to the entire screen as shown in the picture above. To ensure a high level of visual comfort, we customize the typography inside of the box based on the box unit.

Outline

In the grid system, we define the frame outside the information area based on `row` and `column`, to ensure that every area can have stable arrangement.

Following is a brief look at how it works:

- Establish a set of `column` in the horizontal space defined by `row` (abbreviated col).
- Your content elements should be placed directly in the `col`, and only `col` should be placed directly in `row`.
- The column grid system is a value of 1-24 to represent its range spans. For example, three columns of equal width can be created by `<Col span={8} />`.
- If the sum of `col` spans in a `row` are more than 24, then the overflowing `col` as a whole will start a new line arrangement.

Our grid systems base on Flex layout to allow the elements within the parent to be aligned horizontally - left, center, right, wide arrangement, and decentralized arrangement. The Grid system also supports vertical alignment - top aligned, vertically centered, bottom-aligned. You can also define the order of elements by using `order`.

Layout uses a 24 grid layout to define the width of each "box", but does not rigidly adhere to the grid layout.

Examples

Basic Grid

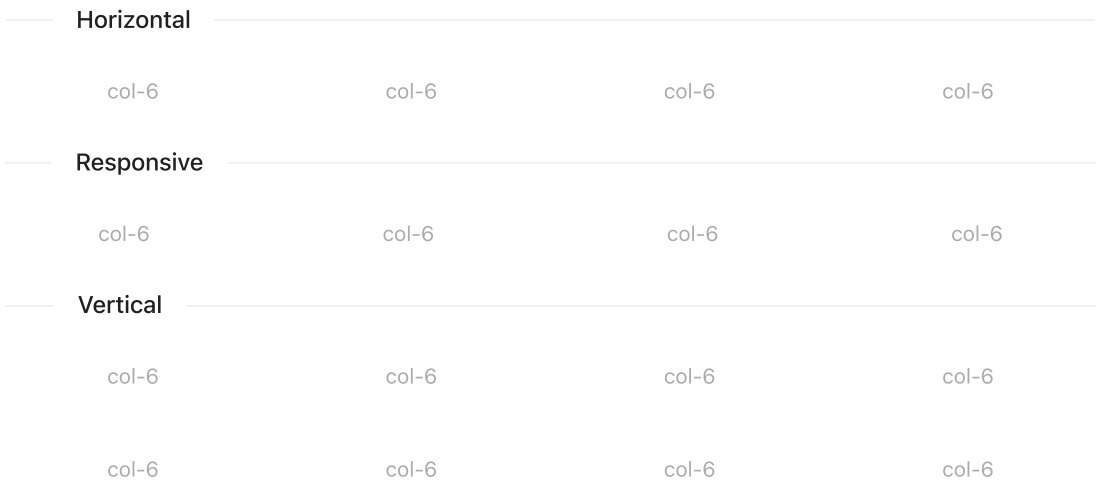
From the stack to the horizontal arrangement.

You can create a basic grid system by using a single set of `Row` and `Col` grid assembly, all of the columns (Col) must be placed in `Row`.

```
import React from 'react';
import { Col, Row } from 'antd';

const App: React.FC = () => (
  <>
    <Row>
      <Col span={24}>col</Col>
    </Row>
    <Row>
      <Col span={12}>col-12</Col>
      <Col span={12}>col-12</Col>
    </Row>
    <Row>
      <Col span={8}>col-8</Col>
      <Col span={8}>col-8</Col>
      <Col span={8}>col-8</Col>
    </Row>
    <Row>
      <Col span={6}>col-6</Col>
      <Col span={6}>col-6</Col>
      <Col span={6}>col-6</Col>
      <Col span={6}>col-6</Col>
    </Row>
  </>
);

export default App;
```



Grid Gutter

You can use the `gutter` property of `Row` as grid spacing, we recommend set it to `(16 + 8n) px` (`n` stands for natural number).

You can set it to a object like `{ xs: 8, sm: 16, md: 24, lg: 32 }` for responsive design.

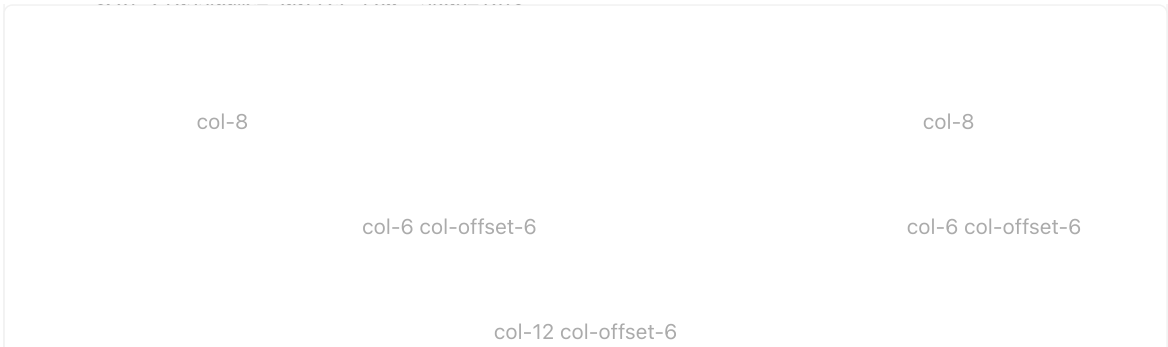
You can use an array to set vertical spacing, `[horizontal, vertical]` `[16, { xs: 8, sm: 16, md: 24, lg: 32 }]`.

> vertical gutter was supported after `3.24.0`.

```
import React from 'react';
import { Col, Divider, Row } from 'antd';

const style: React.CSSProperties = { background: '#0092ff', padding: '8px 0' };

const App: React.FC = () => (
  <>
    <Divider orientation="left">Horizontal</Divider>
    <Row gutter={16}>
      <Col className="gutter-row" span={6}>
        <div style={style}>col-6</div>
      </Col>
      <Col className="gutter-row" span={6}>
        <div style={style}>col-6</div>
      </Col>
      <Col className="gutter-row" span={6}>
        <div style={style}>col-6</div>
      </Col>
      <Col className="gutter-row" span={6}>
        <div style={style}>col-6</div>
      </Col>
    </Row>
    <Divider orientation="left">Responsive</Divider>
    <Row gutter={{ xs: 8, sm: 16, md: 24, lg: 32 }}>
      <Col className="gutter-row" span={6}>
        <div style={style}>col-6</div>
      </Col>
      <Col className="gutter-row" span={6}>
        <div style={style}>col-6</div>
      </Col>
      <Col className="gutter-row" span={6}>
        <div style={style}>col-6</div>
      </Col>
      <Col className="gutter-row" span={6}>
        <div style={style}>col-6</div>
      </Col>
    </Row>
    <Divider orientation="left">Vertical</Divider>
    <Row gutter={[16, 24]}>
      <Col className="gutter-row" span={6}>
```



Column offset

`offset` can set the column to the right side. For example, using `offset = {4}` can set the element shifted to the right four columns width.

```
import React from 'react';
import { Col, Row } from 'antd';

const App: React.FC = () => (
  <>
    <Row>
      <Col span={8}>col-8</Col>
      <Col span={8} offset={8}>
        col-8
      </Col>
    </Row>
    <Row>
      <Col span={6} offset={6}>
        col-6 col-offset-6
      </Col>
      <Col span={6} offset={6}>
        col-6 col-offset-6
      </Col>
    </Row>
    <Row>
      <Col span={12} offset={6}>
        col-12 col-offset-6
      </Col>
    </Row>
  </>
);

export default App;
```

Grid sort

By using `push` and `pull` class you can easily change column order.

```
import React from 'react';
import { Col, Row } from 'antd';

const App: React.FC = () => (
  <Row>
    <Col span={18} push={6}>
      col-18 col-push-6
    </Col>
    <Col span={6} pull={18}>
      col-6 col-pull-18
    </Col>
  </Row>
);

export default App;
```

Typesetting

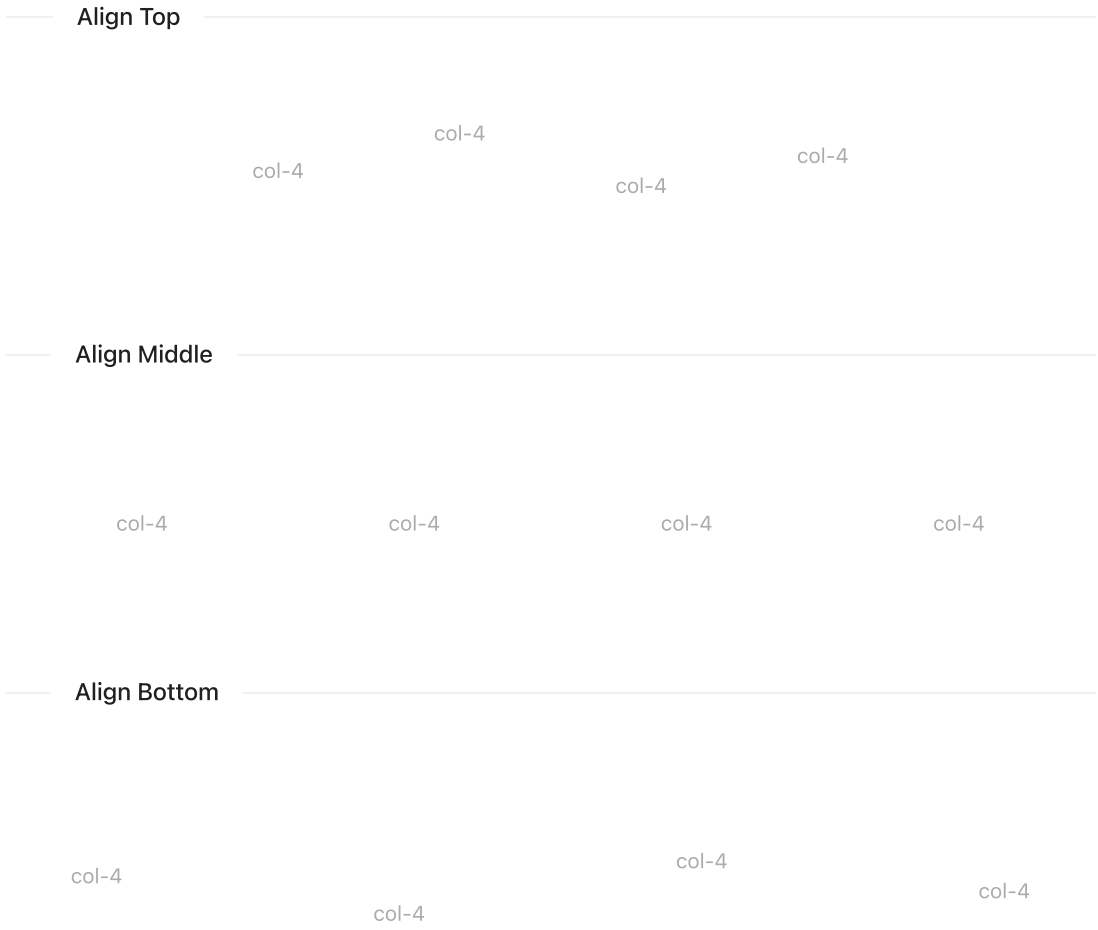
Child elements depending on the value of the `start`, `center`, `end`, `space-between`, `space-around` and `space-evenly`, which are defined in its parent node typesetting mode.

```
import React from 'react';
import { Col, Divider, Row } from 'antd';

const App: React.FC = () => (
  <>
    <Divider orientation="left">sub-element align left</Divider>
    <Row justify="start">
      <Col span={4}>col-4</Col>
      <Col span={4}>col-4</Col>
      <Col span={4}>col-4</Col>
      <Col span={4}>col-4</Col>
    </Row>

    <Divider orientation="left">sub-element align center</Divider>
    <Row justify="center">
      <Col span={4}>col-4</Col>
      <Col span={4}>col-4</Col>
      <Col span={4}>col-4</Col>
      <Col span={4}>col-4</Col>
    </Row>

    <Divider orientation="left">sub-element align right</Divider>
    <Row justify="end">
      <Col span={4}>col-4</Col>
      <Col span={4}>col-4</Col>
```



Alignment

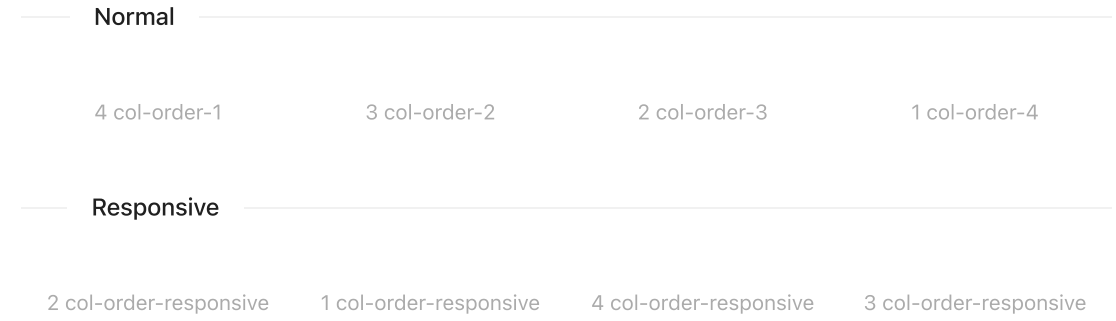
Child elements vertically aligned.

```
import React from 'react';
import { Col, Divider, Row } from 'antd';

const DemoBox: React.FC<{ children: React.ReactNode; value: number }> = (props) => (
  <p className={`height-${props.value}`}>{props.children}</p>
);

const App: React.FC = () => (
  <>
    <Divider orientation="left">Align Top</Divider>
    <Row justify="center" align="top">
      <Col span={4}>
        <DemoBox value={100}>col-4</DemoBox>
      </Col>
      <Col span={4}>
        <DemoBox value={50}>col-4</DemoBox>
      </Col>
      <Col span={4}>
        <DemoBox value={120}>col-4</DemoBox>
      </Col>
      <Col span={4}>
        <DemoBox value={80}>col-4</DemoBox>
      </Col>
    </Row>

    <Divider orientation="left">Align Middle</Divider>
    <Row justify="space-around" align="middle">
      <Col span={4}>
        <DemoBox value={100}>col-4</DemoBox>
      </Col>
      <Col span={4}>
```



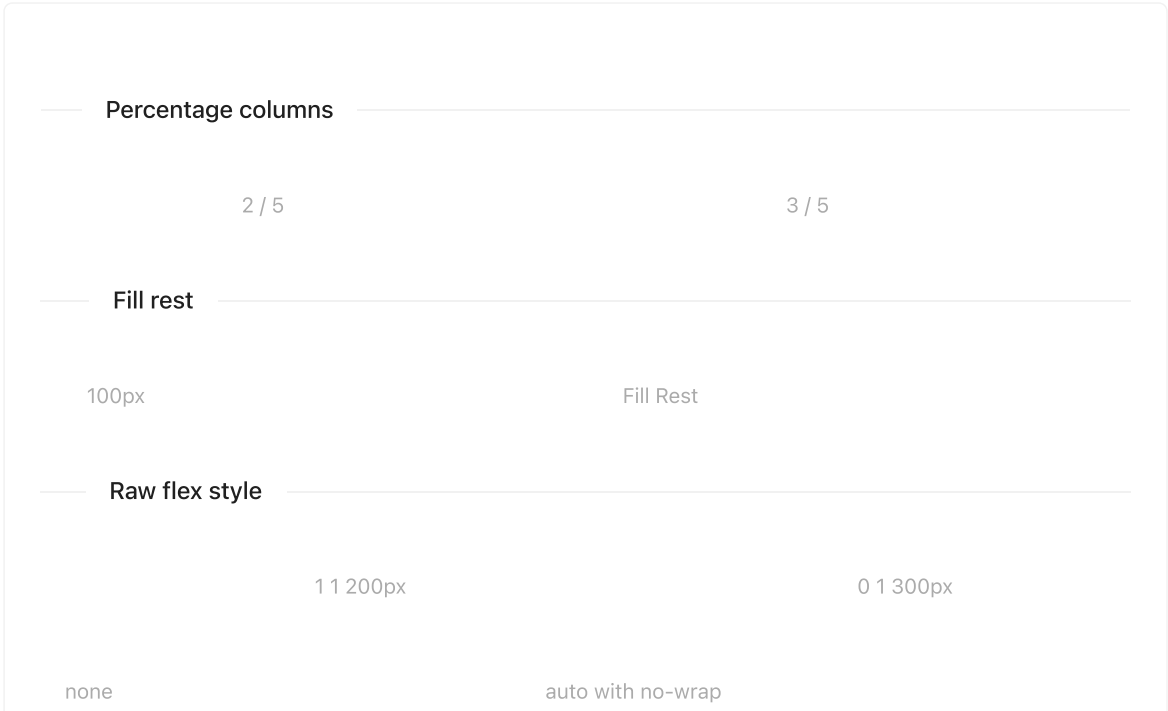
Order

To change the element sort by `order`.

```
import React from 'react';
import { Col, Divider, Row } from 'antd';

const App: React.FC = () => (
  <>
    <Divider orientation="left">Normal</Divider>
    <Row>
      <Col span={6} order={4}>
        1 col-order-4
      </Col>
      <Col span={6} order={3}>
        2 col-order-3
      </Col>
      <Col span={6} order={2}>
        3 col-order-2
      </Col>
      <Col span={6} order={1}>
        4 col-order-1
      </Col>
    </Row>
    <Divider orientation="left">Responsive</Divider>
    <Row>
      <Col span={6} xs={{ order: 1 }} sm={{ order: 2 }} md={{ order: 3 }} lg={{ order: 4 }}>
        1 col-order-responsive
      </Col>
      <Col span={6} xs={{ order: 2 }} sm={{ order: 1 }} md={{ order: 4 }} lg={{ order: 3 }}>
        2 col-order-responsive
      </Col>
      <Col span={6} xs={{ order: 3 }} sm={{ order: 4 }} md={{ order: 2 }} lg={{ order: 1 }}>
        3 col-order-responsive
      </Col>
      <Col span={6} xs={{ order: 4 }} sm={{ order: 3 }} md={{ order: 1 }} lg={{ order: 2 }}>
        4 col-order-responsive
      </Col>
    </Row>
  </>
);

export default App;
```



Flex Stretch

Col provides `flex` prop to support fill rest.

```
import React from 'react';
import { Col, Divider, Row } from 'antd';

const App: React.FC = () => (
  <>
    <Divider orientation="left">Percentage columns</Divider>
    <Row>
      <Col flex={2}>2 / 5</Col>
      <Col flex={3}>3 / 5</Col>
    </Row>
    <Divider orientation="left">Fill rest</Divider>
    <Row>
      <Col flex="100px">100px</Col>
      <Col flex="auto">Fill Rest</Col>
    </Row>
    <Divider orientation="left">Raw flex style</Divider>
    <Row>
      <Col flex="1 1 200px">1 1 200px</Col>
      <Col flex="0 1 300px">0 1 300px</Col>
    </Row>

    <Row wrap={false}>
      <Col flex="none">
        <div style={{ padding: '0 16px' }}>none</div>
      </Col>
      <Col flex="auto">auto with no-wrap</Col>
    </Row>
  </>
);

export default App;
```

Responsive

Referring to the Bootstrap responsive design, here preset six dimensions: `xs` `sm` `md` `lg` `xl` `xxl`.

```
import React from 'react';
import { Col, Row } from 'antd';

const App: React.FC = () => (
  <Row>
    <Col xs={2} sm={4} md={6} lg={8} xl={10}>
      Col
    </Col>
    <Col xs={20} sm={16} md={12} lg={8} xl={4}>
      Col
    </Col>
    <Col xs={2} sm={4} md={6} lg={8} xl={10}>
      Col
    </Col>
  </Row>
);

export default App;
```

More responsive

`span` `pull` `push` `offset` `order` property can be embedded into `xs` `sm` `md` `lg` `xl` `xxl` properties to use, where `xs={6}` is equivalent to `xs={{span: 6}}`.

```
import React from 'react';
import { Col, Row } from 'antd';

const App: React.FC = () => (
  <Row>
    <Col xs={{ span: 5, offset: 1 }} lg={{ span: 6, offset: 2 }}>
      Col
    </Col>
    <Col xs={{ span: 11, offset: 1 }} lg={{ span: 6, offset: 2 }}>
      Col
    </Col>
    <Col xs={{ span: 5, offset: 1 }} lg={{ span: 6, offset: 2 }}>
      Col
    </Col>
  </Row>
);

export default App;
```

Horizontal Gutter (px):

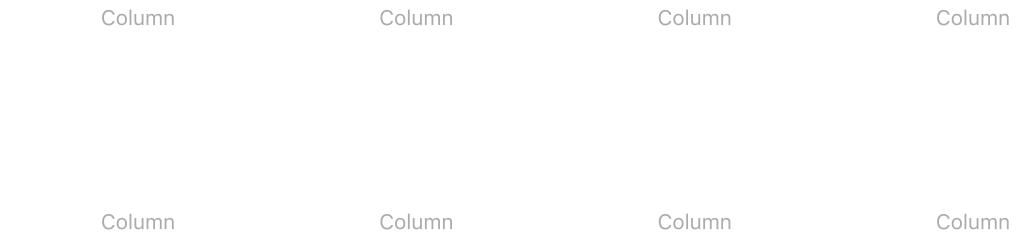
8 16 24 32 40 48

Vertical Gutter (px):

8 16 24 32 40 48

Column Count:

2 3 4 6 8 12



Another Row:


```
<Row gutter={[16, 16]}>
  <Col span={6} />
  <Col span={6} />
  <Col span={6} />
  <Col span={6} />

  <Col span={6} />
  <Col span={6} />
  <Col span={6} />
  <Col span={6} />
</Row>

<Row gutter={[16, 16]}>
  <Col span={6} />
  <Col span={6} />
  <Col span={6} />
  <Col span={6} />
</Row>
```

Playground

A simple playground for column count and gutter.

```
import React, { useState } from 'react';
import { Col, Row, Slider } from 'antd';

const gutters: Record<PropertyKey, number> = {};
const vgutters: Record<PropertyKey, number> = {};
const colCounts: Record<PropertyKey, number> = {};

[8, 16, 24, 32, 40, 48].forEach((value, i) => {
  gutters[i] = value;
});
[8, 16, 24, 32, 40, 48].forEach((value, i) => {
  vgutters[i] = value;
});
```

Current break point:

useBreakpoint Hook

Use `useBreakpoint` Hook provide personalized layout.

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

const { useBreakpoint } = Grid;

const App: React.FC = () => {
  const screens = useBreakpoint();

  return (
    <>
      Current break point:{' '}
      {Object.entries(screens)
        .filter((screen) => !!screen[1])
        .map((screen) => (
          <Tag color="blue" key={screen[0]}>
            {screen[0]}
          </Tag>
        ))}
    </>
  );
};

export default App;
```

```
        </div>
        <span>Vertical Gutter (px): </span>
        <div style={{ width: '50%' }}>
          <Slider
            min={0}
            max={Object.keys(vgutters).length - 1}
            value={vgutterKey}
            onChange={setVgutterKey}
            marks={vgutters}
            step={null}
            tooltip={{ formatter: (value: number) => vgutters[value] }}
          />
        </div>
        <span>Column Count:</span>
        <div style={{ width: '50%', marginBottom: 48 }}>
          <Slider
            min={0}
            max={Object.keys(colCounts).length - 1}
            value={colCountKey}
            onChange={setColCountKey}
            marks={colCounts}
            step={null}
            tooltip={{ formatter: (value: number) => colCounts[value] }}
          />
        </div>
        <Row gutter={[gutters[gutterKey], vgutters[vgutterKey]]}>
          {cols}
          {cols}
        </Row>
        Another Row:
        <Row gutter={[gutters[gutterKey], vgutters[vgutterKey]]}>{cols}</Row>
        <pre className="demo-code">{`<Row gutter={[${gutters[gutterKey]}, ${vgutters[vgutterKey]}]}
        <pre className="demo-code">{`<Row gutter={[${gutters[gutterKey]}, ${vgutters[vgutterKey]}]}
      </>
    );
};

export default App;
```

API

If the Ant Design grid layout component does not meet your needs, you can use the excellent layout components of the community:

- react-flexbox-grid
- react-blocks

Row

Property	Description	Type	Default	Version
align	Vertical alignment	`top` | `middle` | `bottom` | `stretch` | {[key in 'xs' | 'sm' | 'md' | 'lg' | 'xl' | 'xxl']: 'top' | 'middle' | 'bottom' | 'stretch'}	`top`	object: 4.24.0
gutter	Spacing between grids, could be a number or a object like { xs: 8, sm: 16, md: 24}. Or you can use array to make horizontal and vertical spacing work at the same time `[horizontal, vertical]`	number | object | array	0	
justify	Horizontal arrangement	`start` | `end` | `center` | `space-around` | `space-between` | `space-evenly` | {[key in 'xs' | 'sm' | 'md' | 'lg' | 'xl' | 'xxl']: 'start' | 'end' | 'center' | 'space-around' | 'space-between' | 'space-evenly'}	`start`	object: 4.24.0
wrap	Auto wrap line	boolean	true	4.8.0

Property	Description	Type	Default	Version

Col

Property	Description	Type	Default	Version
flex	Flex layout style	string | number	-	
offset	The number of cells to offset Col from the left	number	0	
order	Raster order	number	0	
pull	The number of cells that raster is moved to the left	number	0	
push	The number of cells that raster is moved to the right	number	0	
span	Raster number of cells to occupy, 0 corresponds to `display: none`	number	none	
xs	`screen < 576px` and also default setting, could be a `span` value or an object containing above props	number | object	-	
sm	`screen ≥ 576px`, could be a `span` value or an object containing above props	number | object	-	
md	`screen ≥ 768px`, could be a `span` value or an object containing above props	number | object	-	
lg	`screen ≥ 992px`, could be a `span` value or an object containing above props	number | object	-	
xl	`screen ≥ 1200px`, could be a `span` value or an object containing above props	number | object	-	
xxl	`screen ≥ 1600px`, could be a `span` value or an object containing above props	number | object	-	

You can modify the breakpoints values using by modifying `screen[XS|SM|MD|LG|XL|XXL]` with theme customization (since 5.1.0, sandbox demo).

The breakpoints of responsive grid follow BootStrap 4 media queries rules (not including `occasionally part`).

Design Token