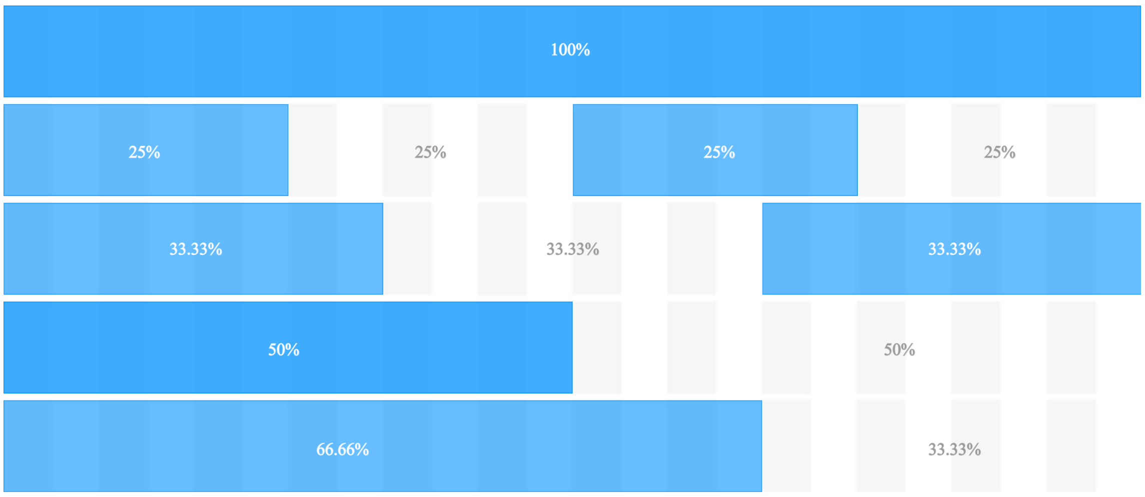
24 Grids System.

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.

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`	`top`	
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`	`start`	
wrap	Auto wrap line	boolean	true	4.8.0

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	-	

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