Handling Responsieness

CSS

Topics are Covered :

- Images
- Aspects Ration’s
- Media Queries
- Flex Box

CSS images

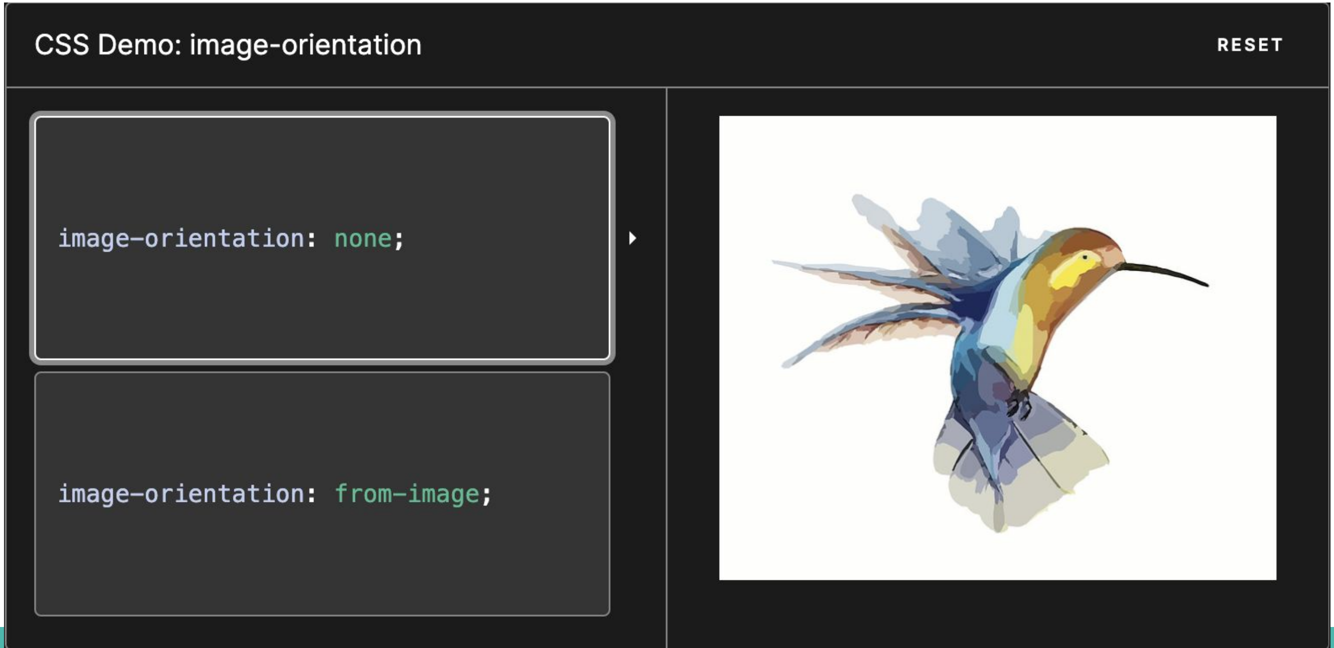
The CSS images module defines the types of images that can be used (the <image> type, containing URLs, gradients and other types of images), how to resize them and how they, and other replaced content, interact with the different layout models.

Properties :

- Image-orientation
- Image-rendering
- Image-resolution
- Object-fit
- object-position

Image-orientation:

The image-orientation CSS property specifies a layout-independent correction to the orientation of an image.

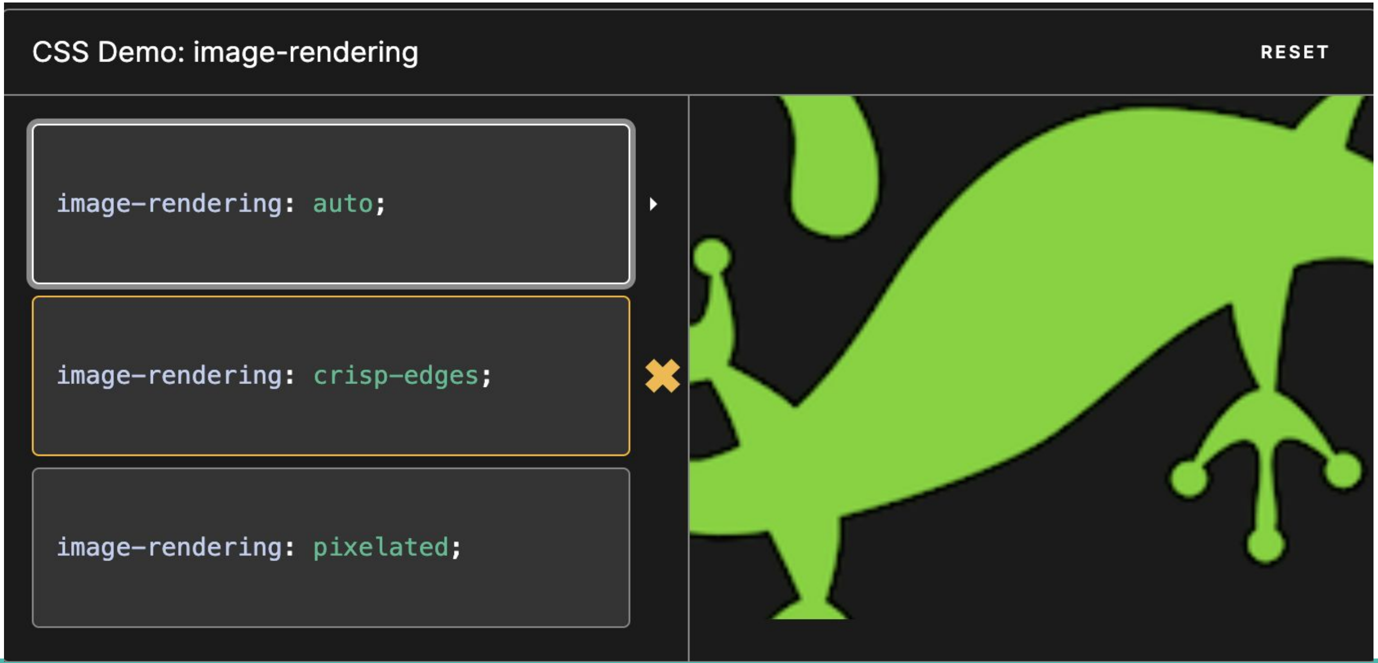

CSS Demo: image-orientation

```css
image-orientation: none;
```

```css
image-orientation: from-image;
```

Image-rendering:

The image-rendering CSS property sets an image scaling algorithm. The property applies to an element itself, to any images set in its other properties, and to its descendants.



```
image-rendering: auto;
```

```
image-rendering: crisp-edges;
```

```
image-rendering: pixelated;
```

Image-resolution:

The image-resolution CSS property specifies the intrinsic resolution of all raster images used in or on the element. It affects content images such as replaced elements and generated content, and decorative images such as background-image images.

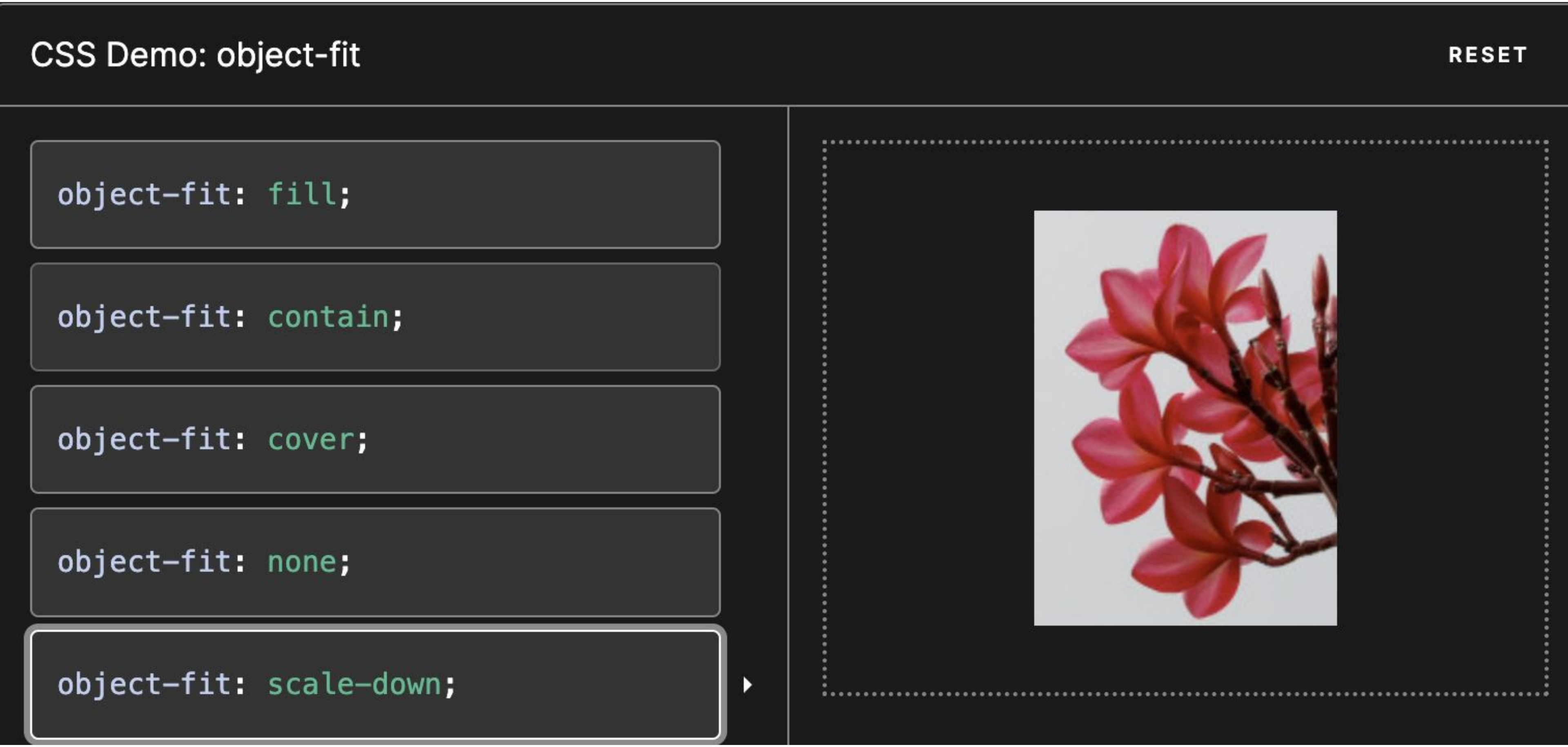
The image resolution is defined as the number of image pixels per unit length, e.g., pixels per inch. By default, CSS assumes a resolution of one image pixel per CSS px unit; however, the image-resolution property allows a different resolution to be specified.

```
image-resolution: from-image;

image-resolution: 300dpi;

image-resolution: from-image 300dpi;

image-resolution: 300dpi snap;


/* Global values */

image-resolution: inherit;

image-resolution: initial;

image-resolution: revert;

image-resolution: revert-layer;

image-resolution: unset;
```

Object-fit:

The object-fit CSS property sets how the content of a replaced element, such as an <img> or <video>, should be resized to fit its container.

You can alter the alignment of the replaced element's content object within the element's box using the object-position property.



CSS Demo: object-fit

```css
object-fit: fill;
```

```css
object-fit: contain;
```

```css
object-fit: cover;
```

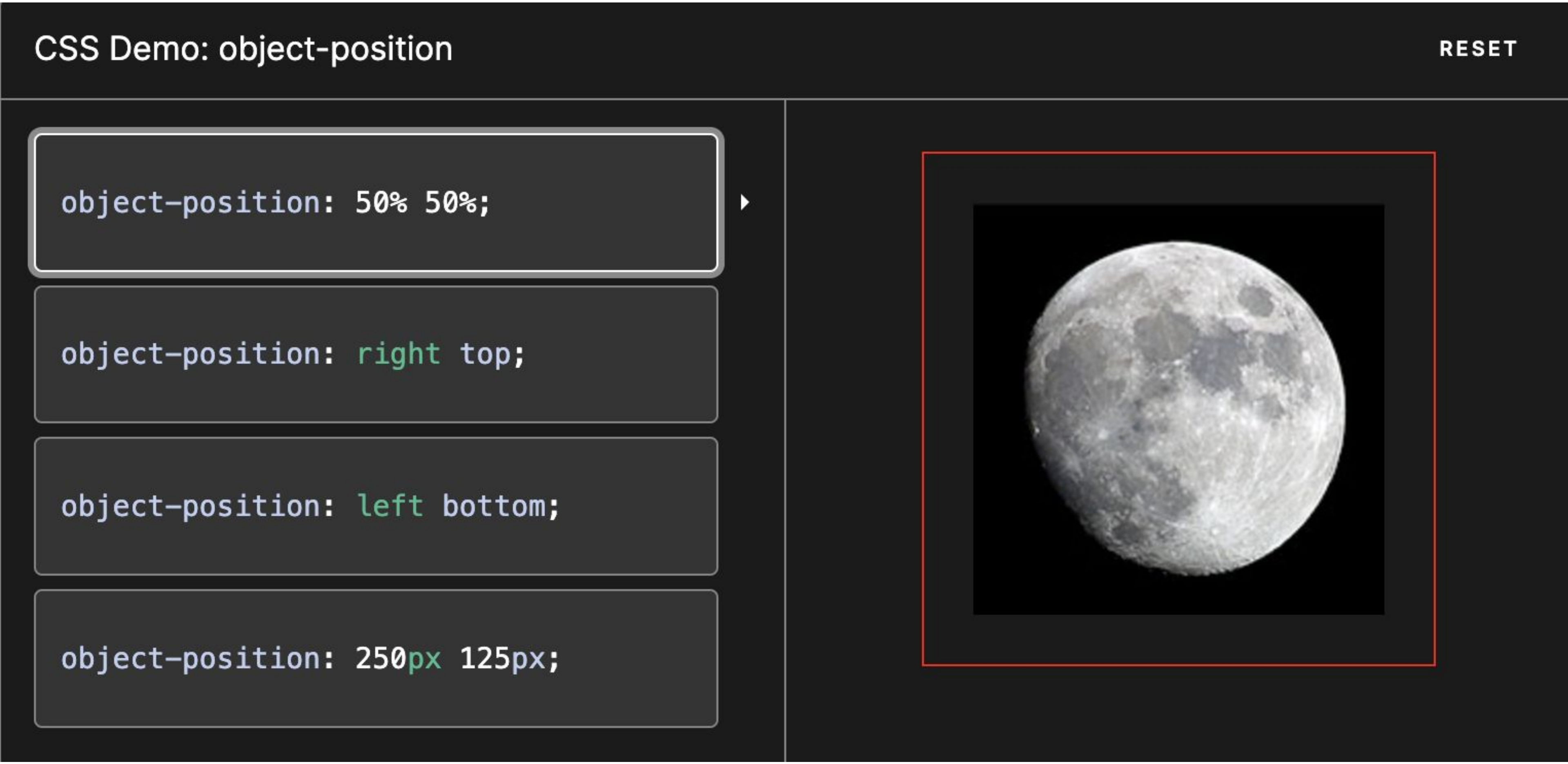
```css
object-fit: none;
```

```css
object-fit: scale-down;
```

Object-position:

The object-position CSS property specifies the alignment of the selected replaced element's contents within the element's box. Areas of the box which aren't covered by the replaced element's object will show the element's background.

You can adjust how the replaced element's object's intrinsic size (that is, its natural size) is adjusted to fit within the element's box using the object-fit property.



CSS Demo: object-position

```
object-position: 50% 50%;
```

```
object-position: right top;
```

```
object-position: left bottom;
```

```
object-position: 250px 125px;
```

Functions:

- linear-gradient()
- radial-gradient()
- repeating-linear-gradient()
- repeating-radial-gradient()
- conic-gradient()
- repeating-conic-gradient()
- url()

Data Types:

- **<gradient>**

The <gradient> CSS data type is a special type of <image> that consists of a progressive transition between two or more colors.

- **<image>**

The <image> data type can be represented with any of the following:

An image denoted by the url() data type

A <gradient> data type

A part of the webpage, defined by the element() function

An image, image fragment or solid patch of color, defined by the image() function

A blending of two or more images defined by the cross-fade() function.

A selection of images chosen based on resolution defined by the image-set() function.

Aspect ratio css :

Definition and Usage

The aspect-ratio property allows you to define the ratio between width and height of an element.

If aspect-ratio and width properties are set, the height will follow in the defined aspect ratio.

To better understand the aspect-ratio property, view a demo.

Tip: Use the aspect-ratio property in responsive layouts where elements often vary in size and you want to preserve the ratio between width and height.

At least one of the box's sizes needs to be automatic in order for aspect-ratio to have any effect. If neither the width nor height is an automatic size, then the provided aspect ratio has no effect on the box's preferred sizes.=

Syntax:

```
aspect-ratio: 1 / 1;

aspect-ratio: 1;

/* fallback to 'auto' for replaced elements */

aspect-ratio: auto 3/4;

aspect-ratio: 9/6 auto;

/* Global values */

aspect-ratio: inherit;

aspect-ratio: initial;

aspect-ratio: revert;

aspect-ratio: revert-layer;

aspect-ratio: unset;
```

Media queries in css:

The @media rule, introduced in CSS2, made it possible to define different style rules for different media types.

Media queries in CSS3 extended the CSS2 media types idea: Instead of looking for a type of device, they look at the capability of the device.

Media queries can be used to check many things, such as:

width and height of the viewport

orientation of the viewport (landscape or portrait)

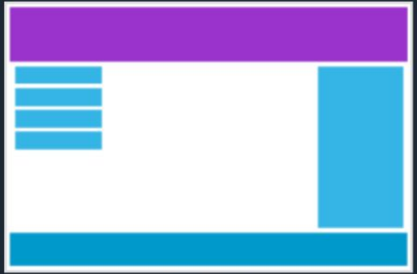
resolution

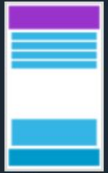
Using media queries are a popular technique for delivering a tailored style sheet to desktops, laptops, tablets, and mobile phones (such as iPhone and Android phones).

Add a Breakpoint

Earlier in this tutorial we made a web page with rows and columns, and it was responsive, but it did not look good on a small screen.

Media queries can help with that. We can add a breakpoint where certain parts of the design will behave differently on each side of the breakpoint.

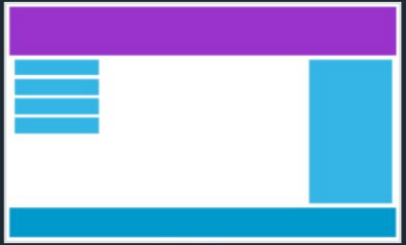

Desktop

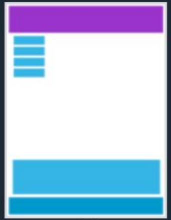

Phone

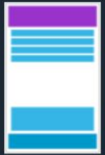
Another Breakpoint

You can add as many breakpoints as you like.

We will also insert a breakpoint between tablets and mobile phones.


Desktop


Tablet


Phone

We do this by adding one more media query (at 600px), and a set of new classes for devices larger than 600px (but smaller than 768px):

Syntax:

A media query is composed of an optional media type and any number of media feature expressions, which may optionally be combined in various ways using logical operators. Media queries are case-insensitive.

Media types define the broad category of device for which the media query applies: all, print, screen. The type is optional (assumed to be all) except when using the not or only logical operators.

Media features describe a specific characteristic of the user agent, output device, or environment:

- Any-hover
- Forced-colorsspect-ratio
- Color
- Color-gamut
- Color-index
- device-aspect-ratio Deprecated
- device-height Deprecated
- device-width Deprecated
- Display-mode
- Dynamic-range
- Forced-colors

- Hover
- Inverted-colors
- Monochrome
- Orientation
- Overflow-block
- Overflow-inline
- Pointer
- Prefers-color-scheme
- Prefers-contrast
- Prefers-reduced-motion
- prefers-reduced-transparency Experimental

FlexBox:

The flexible box layout module, usually referred to as flexbox, was designed as a one-dimensional layout model, and as a method that could offer space distribution between items in an interface and powerful alignment capabilities. This article gives an outline of the main features of flexbox, which we will be exploring in more detail in the rest of these guides.

When we describe flexbox as being one-dimensional we are describing the fact that flexbox deals with layout in one dimension at a time — either as a row or as a column. This can be contrasted with the two-dimensional model of CSS Grid Layout, which controls columns and rows together.

CSS Flexbox Layout Module:

Before the Flexbox Layout module, there were four layout modes:

- Block, for sections in a webpage
- Inline, for text
- Table, for two-dimensional table data
- Positioned, for explicit position of an element

The Flexible Box Layout Module, makes it easier to design flexible responsive layout structure without using float or positioning.

The two axes of flexbox:

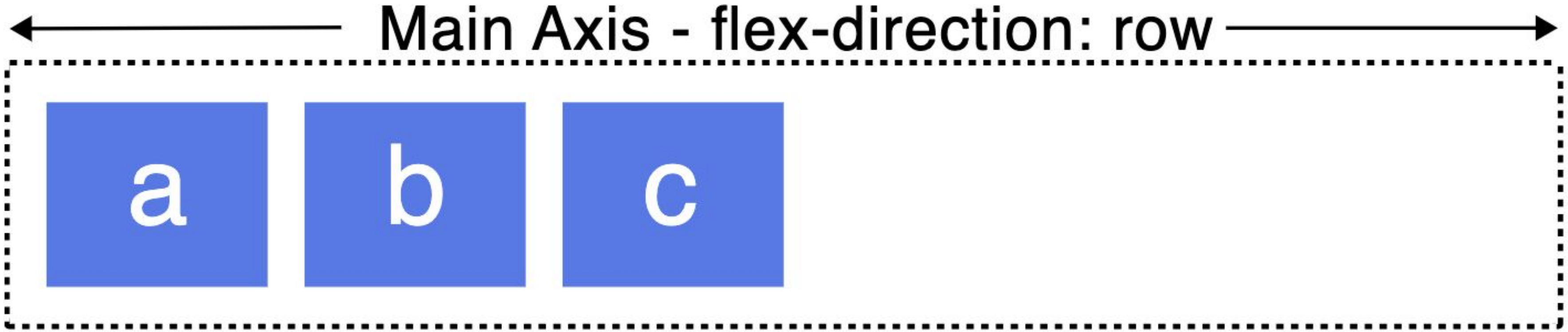
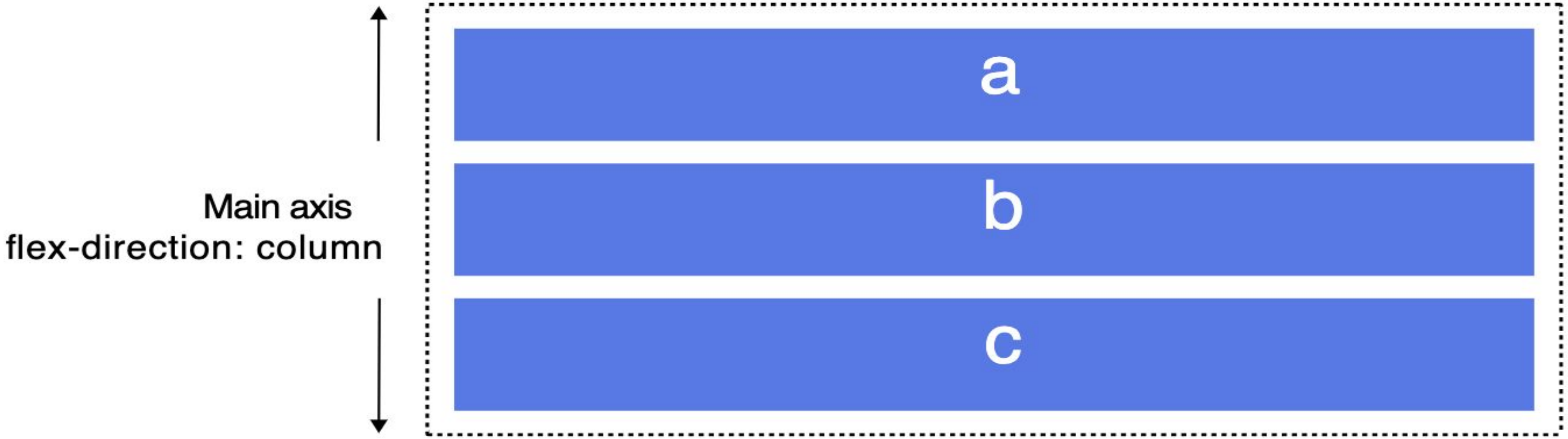
When working with flexbox you need to think in terms of two axes — the main axis and the cross axis. The main axis is defined by the flex-direction property, and the cross axis runs perpendicular to it. Everything we do with flexbox refers back to these axes, so it is worth understanding how they work from the outset.

The main axis:

The main axis is defined by flex-direction, which has four possible values:

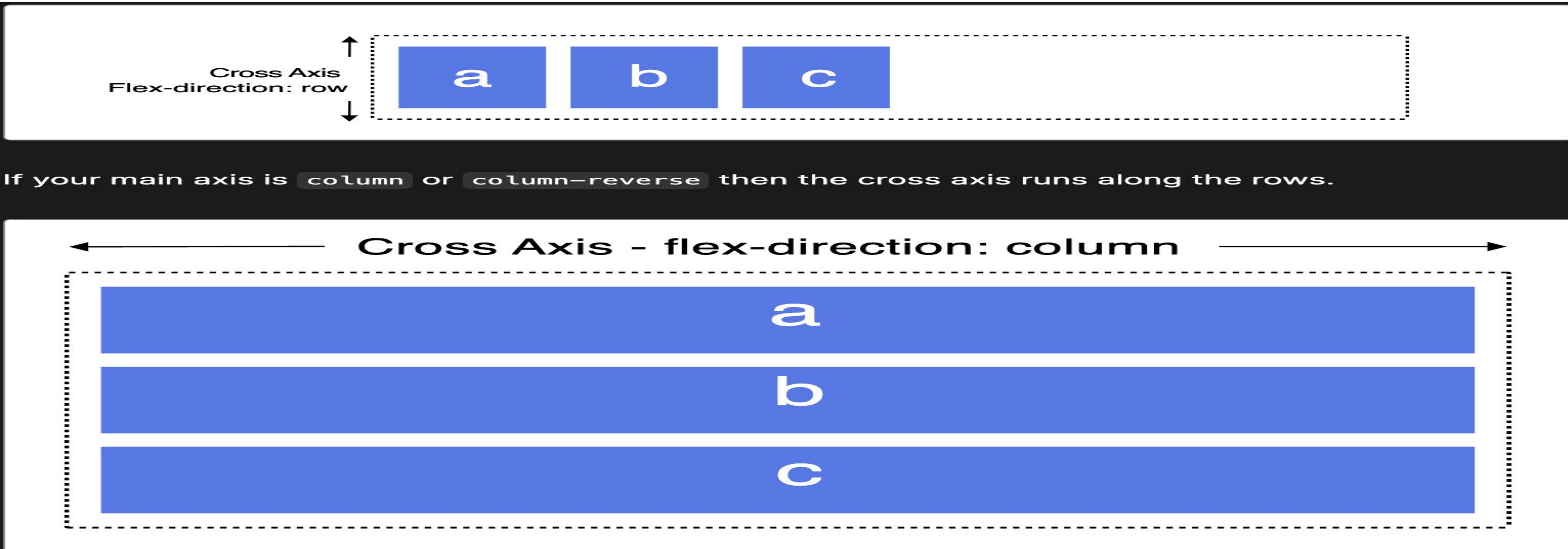
- Row
- Row-reverse
- Column
- column-reverse

Should you choose row or row-reverse, your main axis will run along the row in the inline direction.


Main Axis - flex-direction: row
a
b
c

Main axis
flex-direction: column
a
b
c

The cross axis:

The cross axis runs perpendicular to the main axis, therefore if your flex-direction (main axis) is set to row or row-reverse the cross axis runs down the columns.



If your main axis is `column` or `column-reverse` then the cross axis runs along the rows.

Thank You

By Abdul Haseeb