KodeGo

Introduction to Web Development

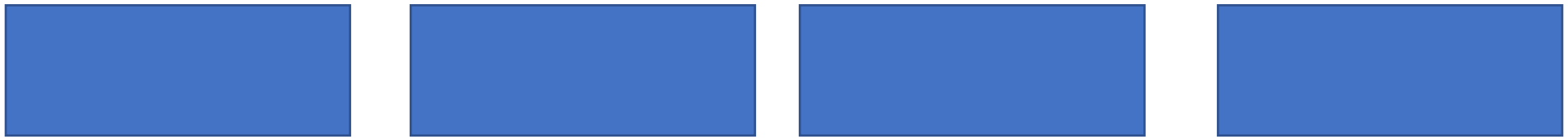
Overview

1. What is Web Development?
2. HTML Basics
3. Activity

HTML Block and Inline Elements

Inline Elements

- An inline element does not start on a new line.
- An inline element only takes up as much width as necessary.



 div 1

Block-level Elements

- A block-level element always **starts on a new line**, and the browsers automatically add some space (a margin) before and after the element.
- A block-level element always takes up the full width available (stretches out to the left and right as far as it can).
- Two commonly used block elements are: `<p>` and `<div>`.


HTML

CSS SELECTORS

Used to "find" or select the HTML elements you want to style.

BASICS Element Selector

id Selector - #

Class Selector- .

Universal Selector - *

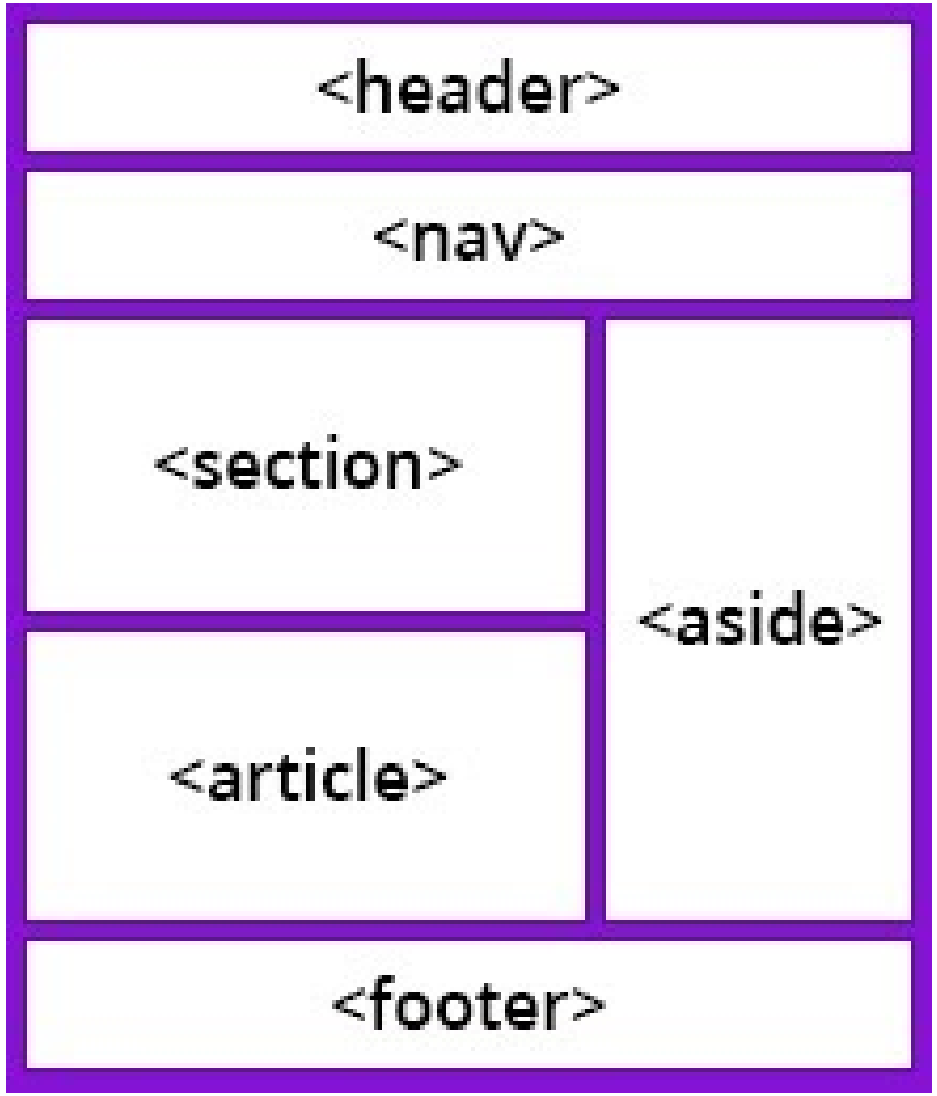
Grouping Selector - selects all the HTML elements with the same style definitions.

SEMANTIC TAGS

Semantic Tags

are used to convey the structure of the document in a clear manner. They define the purpose of the element.

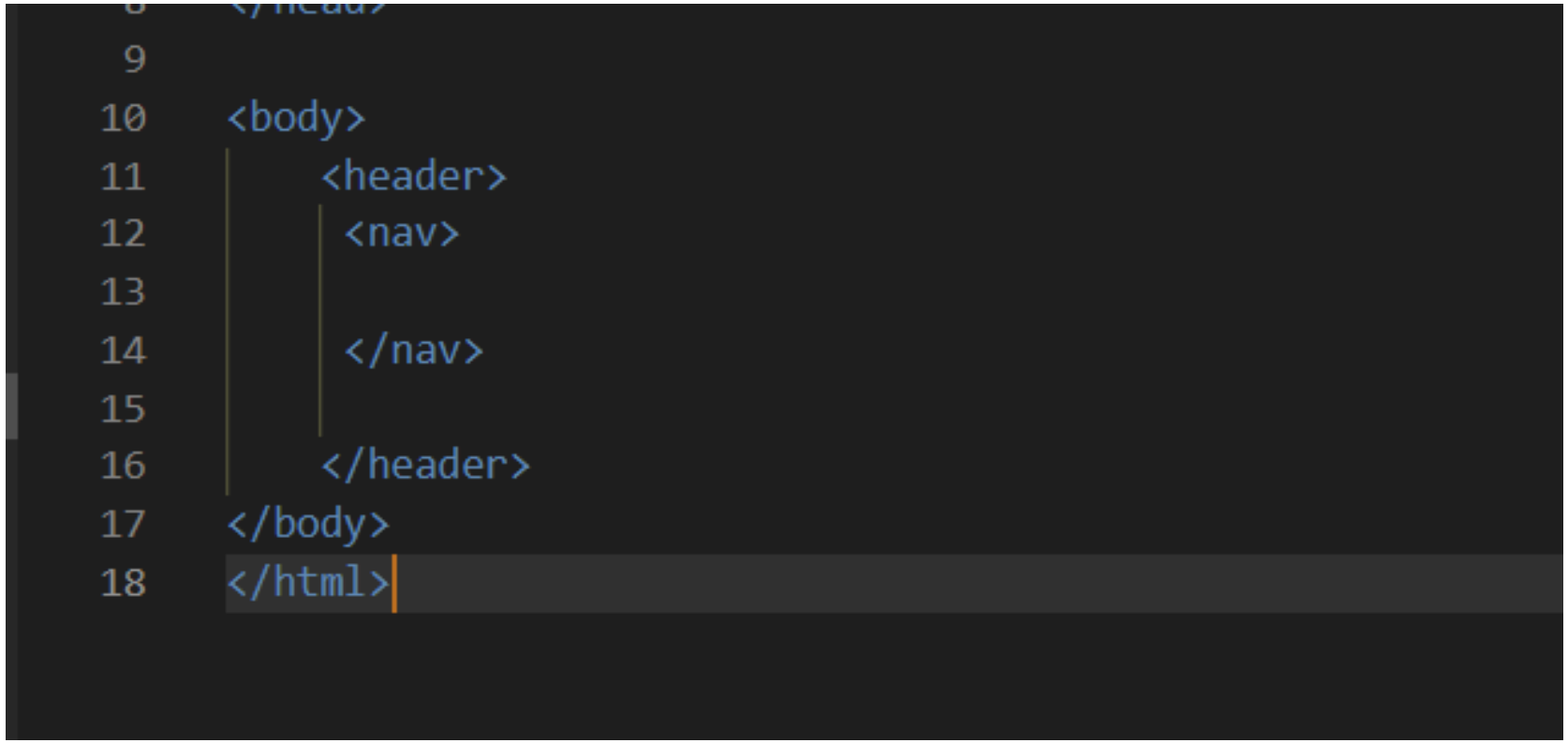
Semantic Tags



KodeGo

Header

```
 9
10  <body>
11      <header>
12
13
14      </header>
15  </body>
16  </html>
```


9
10 <body>
11 <header>
12 <nav>
13
14 </nav>
15
16 </header>
17 </body>
18 </html>

```
10  <body>
11      <header>
12       <nav>
13
14       </nav>
15      </header>
16      <section>
17          <article>
18
19          </article>
20
21      </section>
22
23  </body>
24  </html>
```

```
9
10  <body>
11      <header>
12       <nav>
13
14       </nav>
15      </header>
16      <section>
17          <article>
18
19          </article>
20
21      </section>
22      <footer>
23
24      </footer>
25  </body>
26  </html>
```

Semantic Tags

<header> - defines a header of your document. It is always visible for the users at the top part of the page.

<nav> - defines the space for the navigations links.

<section> - defines a separate section within a webpage. Has its own content; Considered as the "*Parent of articles*" .

<article> - defines the article content. The sub-section.

Semantic Tags

<aside> - defines the content which will be set *to the side*. It is usually used for creating sidebars and side contents.

<footer> - defines the footnote of the web page or content. Can also be used to show the address and give out the reference links

CSS LAYOUT PROPERTY POSITION

- static
- relative
- fixed
- absolute
- sticky

position: static;

it is always positioned according to the normal flow of the page

position: relative;

Setting the top, right, bottom, and left properties of an element with position: relative; property will cause it to adjust from its normal position.

position: fixed;

An element with position: fixed; is positioned relative to the viewport.

position: absolute;

is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed).

position: sticky;

is positioned based on the user's scroll position.
A sticky element toggles between relative and fixed, depending on the scroll position.

Display

`display: block;`- displays the element as a ***block*** element.
Block level elements do not sit inline instead they will create a new **line break.** By default (without setting a width) they take up as much horizontal space as they can.

`display: inline;`- displays the element as an ***inline*** element.
An inline element will accept margin and padding, but the element still sits inline as you might expect.

Margin and padding will only push other elements horizontally away, not vertically.

Display

display: inline-block;- the element will have the characteristic of a *block* element but sits on a *line*. You are now able to set the *width* and *height*, which will be respected.

display: none;- hides the element.

KodeGo

Intro to Web Development

Header

Article heading

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Donec a diam lectus. Set sit amet ipsum mauris. Maecenas congue ligula as quam viverra nec consectetur ant hendrerit. Donec et mollis dolor. Praesent et diam eget libero egestas mattis sit amet vitae augue. Nam tincidunt congue enim, ut porta lorem lacinia consectetur.

subsection

Donec ut librero sed accu vehicula ultricies a non tortor. Lorem ipsum dolor sit amet, consectetur adipisicing elit. Aenean ut gravida lorem. Ut turpis felis, pulvinar a semper sed, adipiscing id dolor.

Pelientesque auctor nisi id magna consequat sagittis. Curabitur dapibus, enim sit amet elit pharetra tincidunt feugiat nist imperdiet. Ut convallis libero in urna ultrices accumsan. Donec sed odio eros.

Another subsection

Donec viverra mi quis quam pulvinar at malesuada arcu rhoncus. Cum soclis natoque penatibus et manis dis parturient montes, nascetur ridiculus mus. In rutrum accumsan ultricies. Mauris vitae nisi at sem facilisis semper ac in est.

Vivamus fermentum semper porta. Nunc diam velit, adipscing ut tristique vitae sagittis vel odio. Maecenas convallis ullamcorper ultricied. Curabitur ornare, ligula semper consectetur sagittis, nisi diam iaculis velit, is fringille sem nunc vet mi.

Related

- Oh I do like to be beside the seaside
- Oh I do like to be beside the sea
- Although in the North of England
- It never stops raining
- Oh well...

KodeGo

LEARNING OUTCOMES

1. Apply the best practices
   -> CSS selectors
   -> CSS positioning.
1. Apply CSS Flexbox.

CSS SELECTORS

KodeGo

Used to ”find” or select the HTML elements you want to style.

Element Selector - elements based on the element name.

id Selector - uses the id attribute of an HTML element to select a specific element. (unique)

Class Selector- elements with a specific class attribute.

Universal Selector - (*) selects all HTML elements on the page.

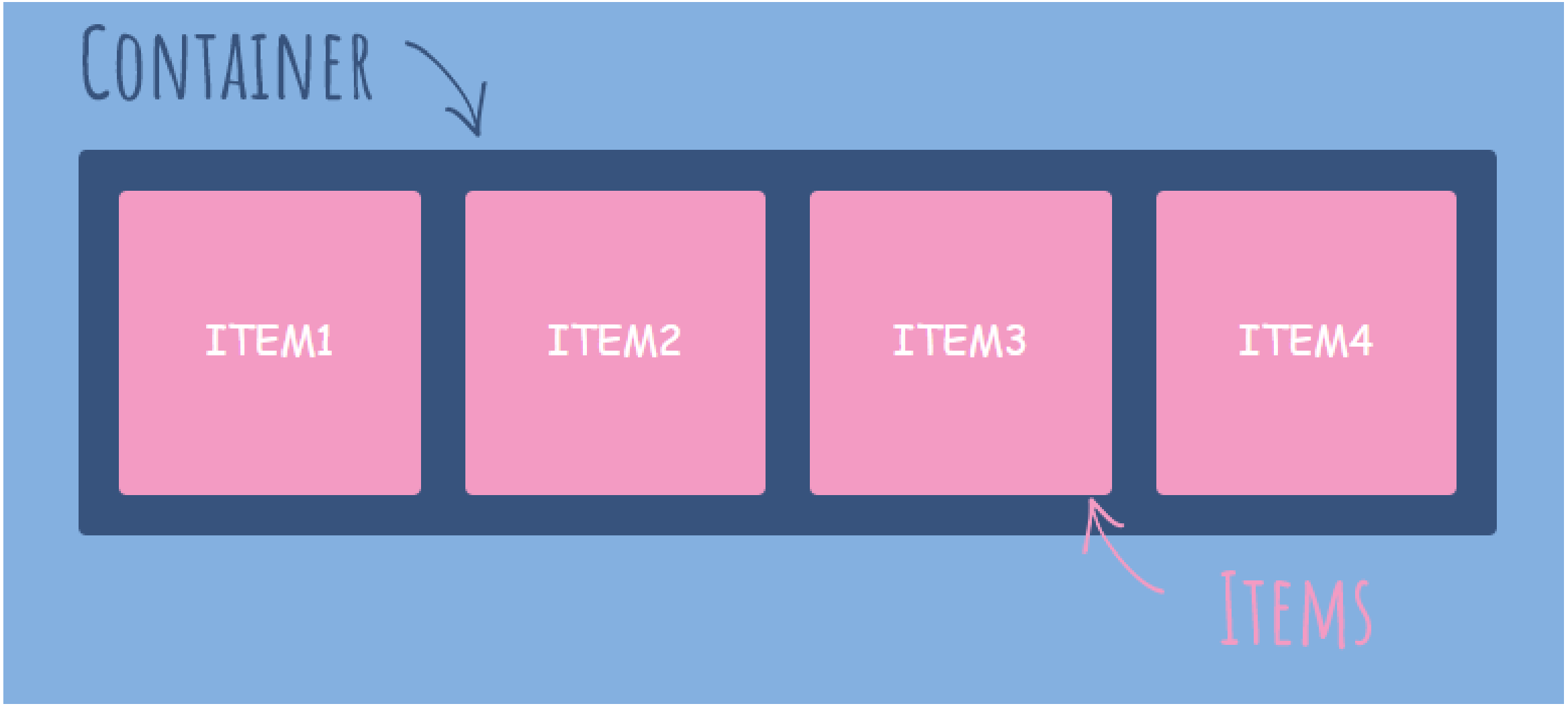
Grouping Selector - selects all the HTML elements with the same style definitions.

FLEXBOX

CSS flexible box layout, commonly known as Flexbox, is a CSS3 web layout model

The flex layout allows responsive elements within a container to be automatically arranged depending upon screen size.

FLEXBOX



EXAMPLE

KodeGo

```
<div class="container">
    <div class="item">
        <p>Item1</p>
    </div>
    <div class="item">
        <p>Item2</p>
    </div>
    <div class="item">
        <p>Item3</p>
    </div>
    <div class="item">
        <p>Item4</p>
    </div>
</div>
```

ADVANTAGES

- Automatically scale elements (alter height or width)
- Prevent overflow
- Change the order of the items.
- Solve the problem of horizontal and vertical centering
- Create columns of the same height
- Create a footer sticking to the bottom of the page
- Design navigation panels
- And more

How to Create the Flex Container

The display: flex; property applies to the container, makes the container a block element, and enables the flex layout for all its direct children.

```
.container {
  display: flex;
}
```

How to Create the Flex Container

The display: inline-flex; works in the same way. Only it creates a container as an inline element.

```
.container {
  display: inline-flex;
}
```

How to Create the Flex Container

The display: flex; property applies to the container, makes the container a block element, and enables the flex layout for all its direct children.

```
.container {
  display: flex;
}
```

How to Create a Row or Column

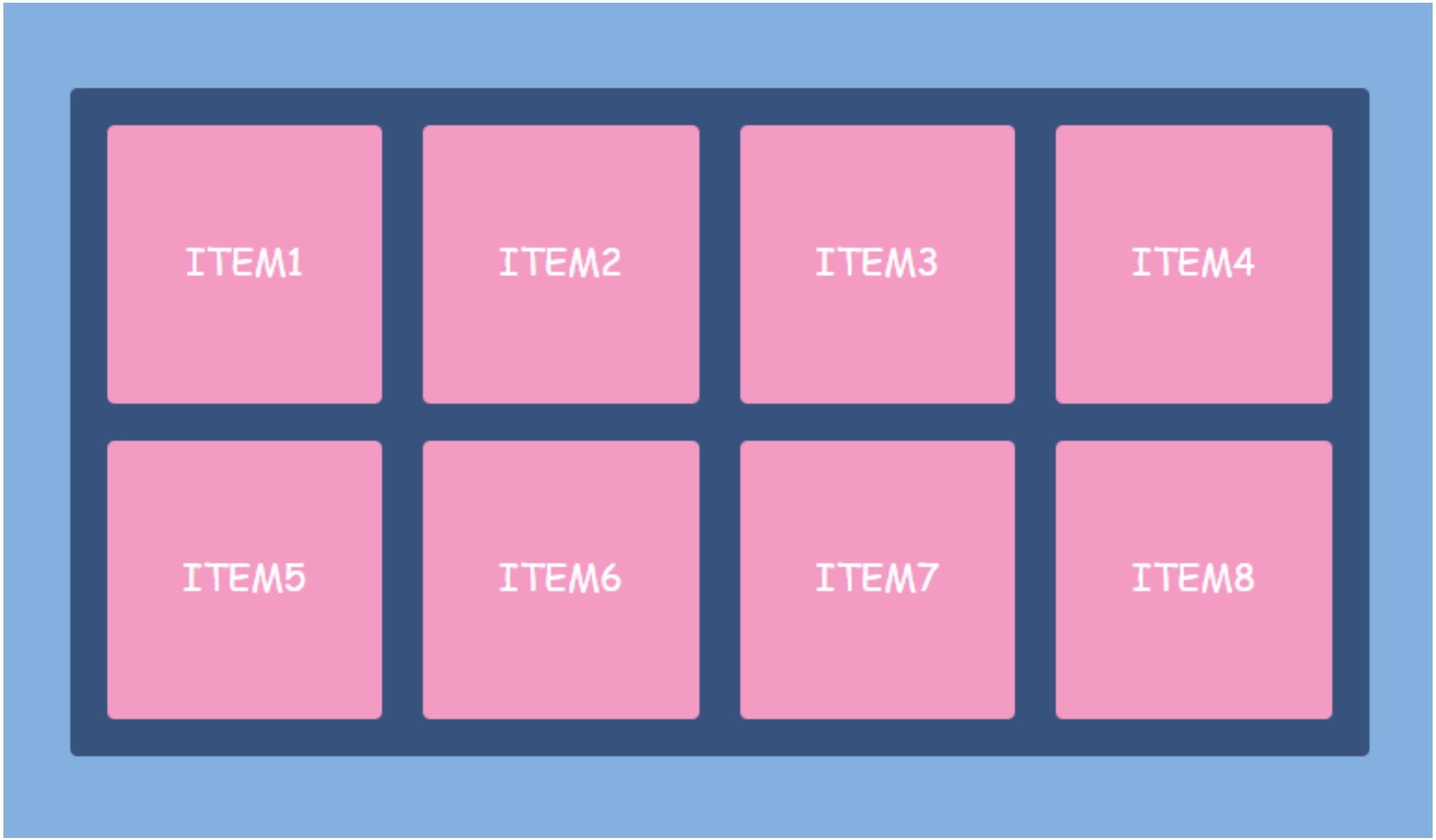
The `flex-direction: row;` property will make a horizontal row. The row is the default value.

```
.container {
  flex-direction: row;
}
```


ITEM1
ITEM2
ITEM3
ITEM4

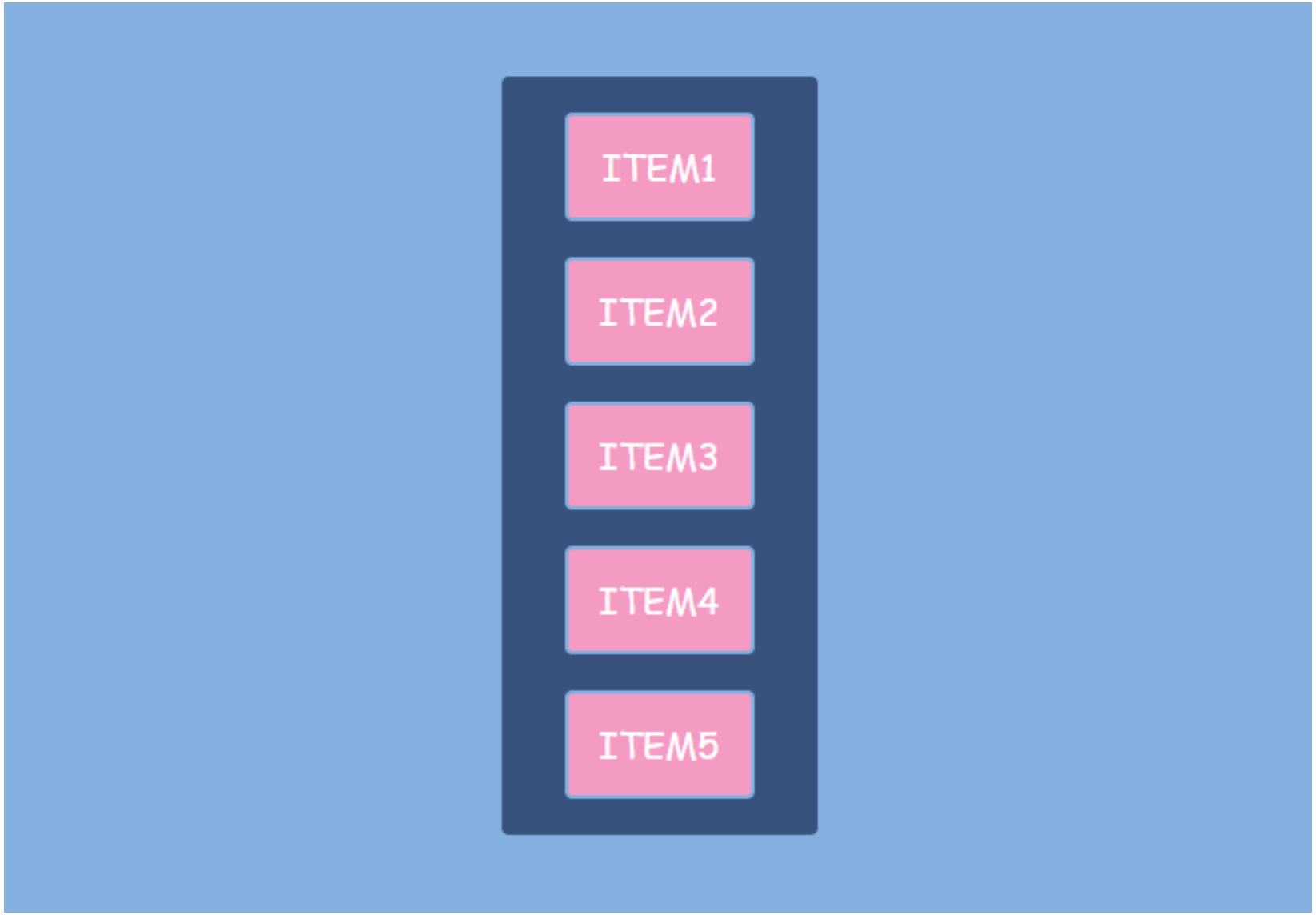
If you have more items that can fit in one row and you still want a horizontal layout, the flex-wrap: wrap; property will come in handy.

```
.container {
  display: flex;
  flex-direction: row;
  flex-wrap: wrap;
}
```


ITEM1
ITEM2
ITEM3
ITEM4
ITEM5
ITEM6
ITEM7
ITEM8

To make a vertical column, apply the flex-direction: column; to the container.

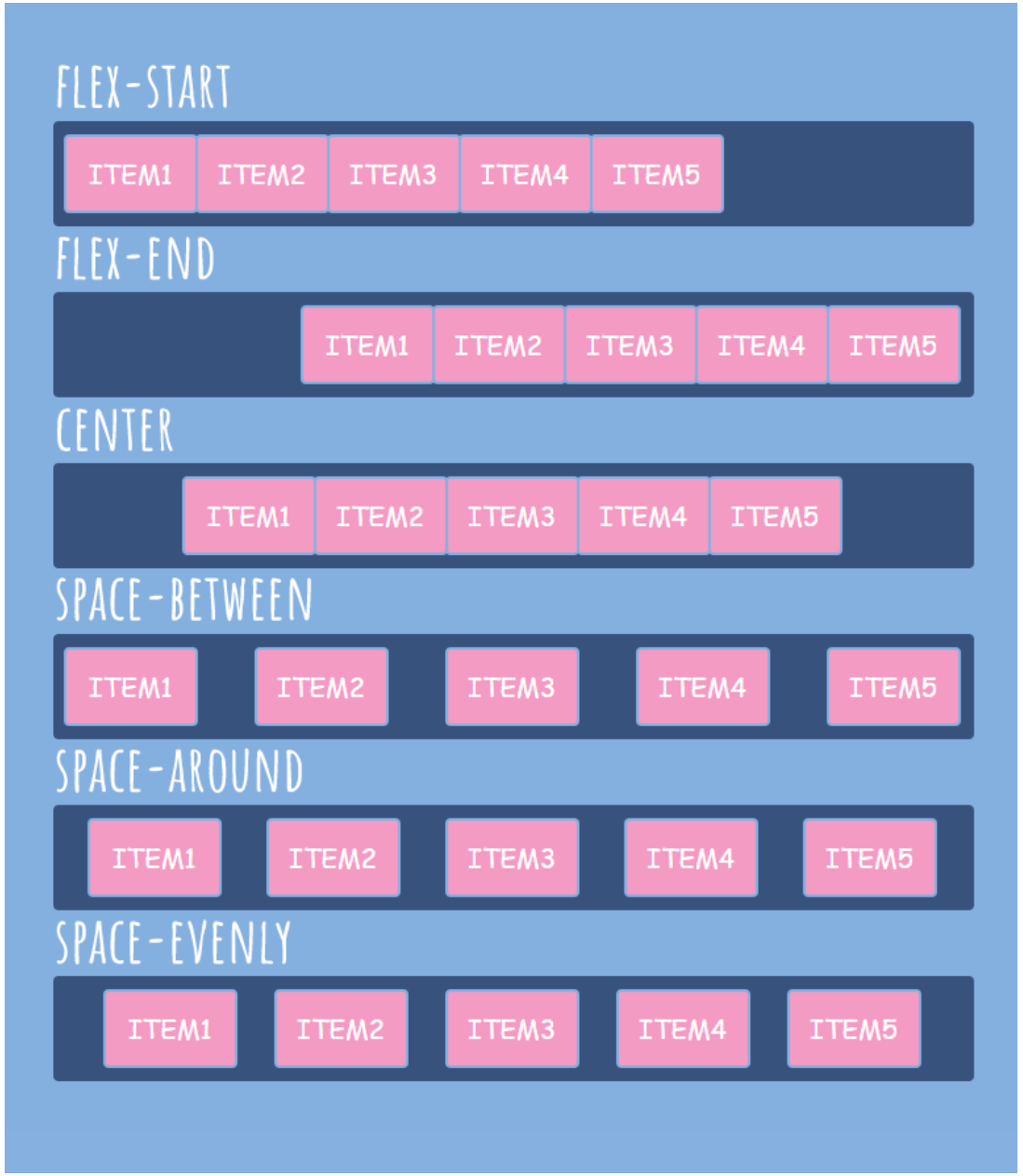
```
.container {
  flex-direction: column;
}
```


ITEM1
ITEM2
ITEM3
ITEM4
ITEM5

Align Elements Horizontally

```
.container {
  justify-content: flex-start | flex-end | center | space-between |
}
```

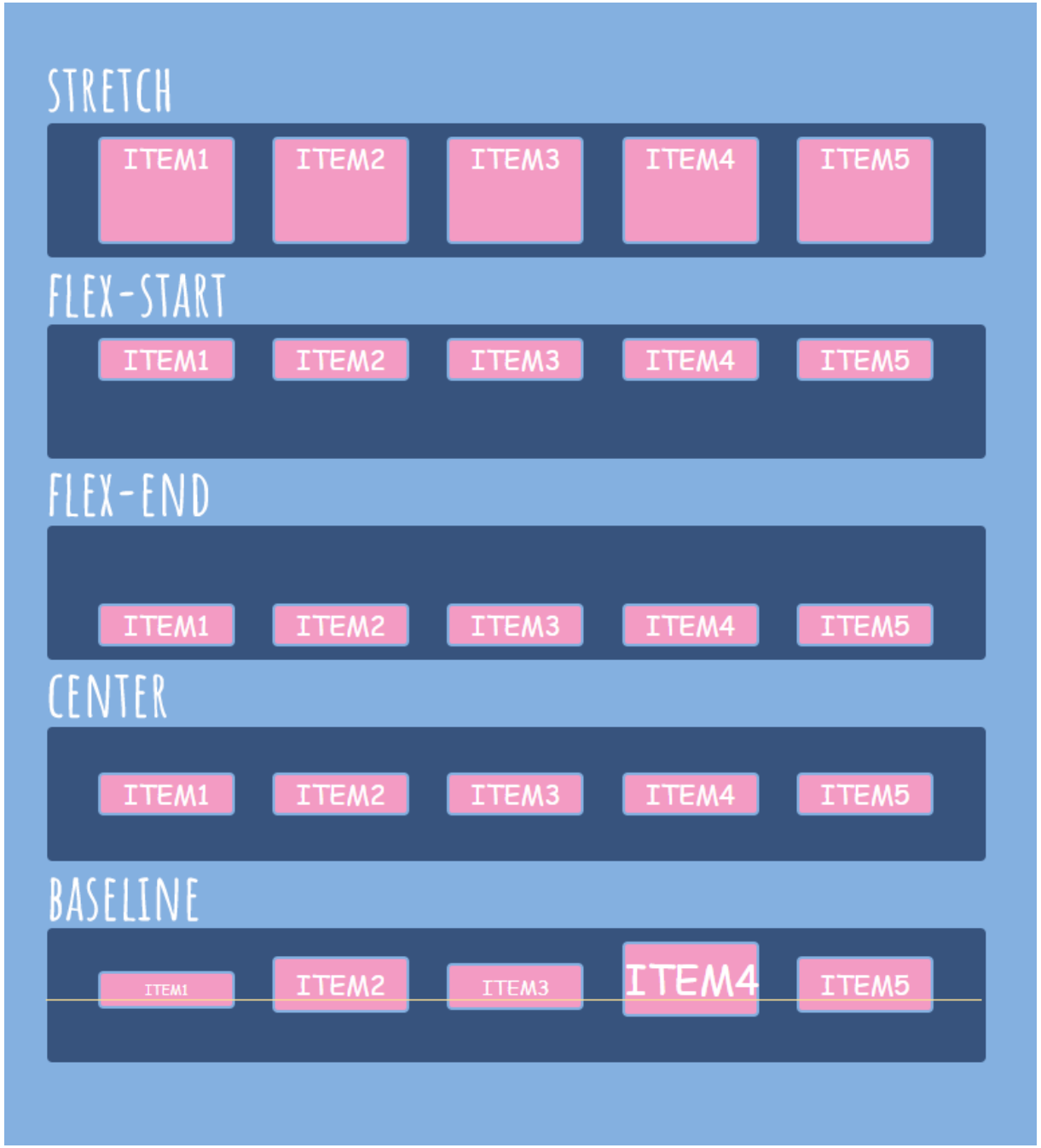
To define the horizontal alignment of items, use the justify-content property.


FLEX-START
ITEM1
ITEM2
ITEM3
ITEM4
ITEM5
FLEX-END
ITEM1
ITEM2
ITEM3
ITEM4
ITEM5
CENTER
ITEM1
ITEM2
ITEM3
ITEM4
ITEM5
SPACE-BETWEEN
ITEM1
ITEM2
ITEM3
ITEM4
ITEM5
SPACE-AROUND
ITEM1
ITEM2
ITEM3
ITEM4
ITEM5
SPACE-EVENLY
ITEM1
ITEM2
ITEM3
ITEM4
ITEM5

How to Align Elements Vertically

To align flex items vertically, use the `align-items`, `align-content` or `align-self` properties.

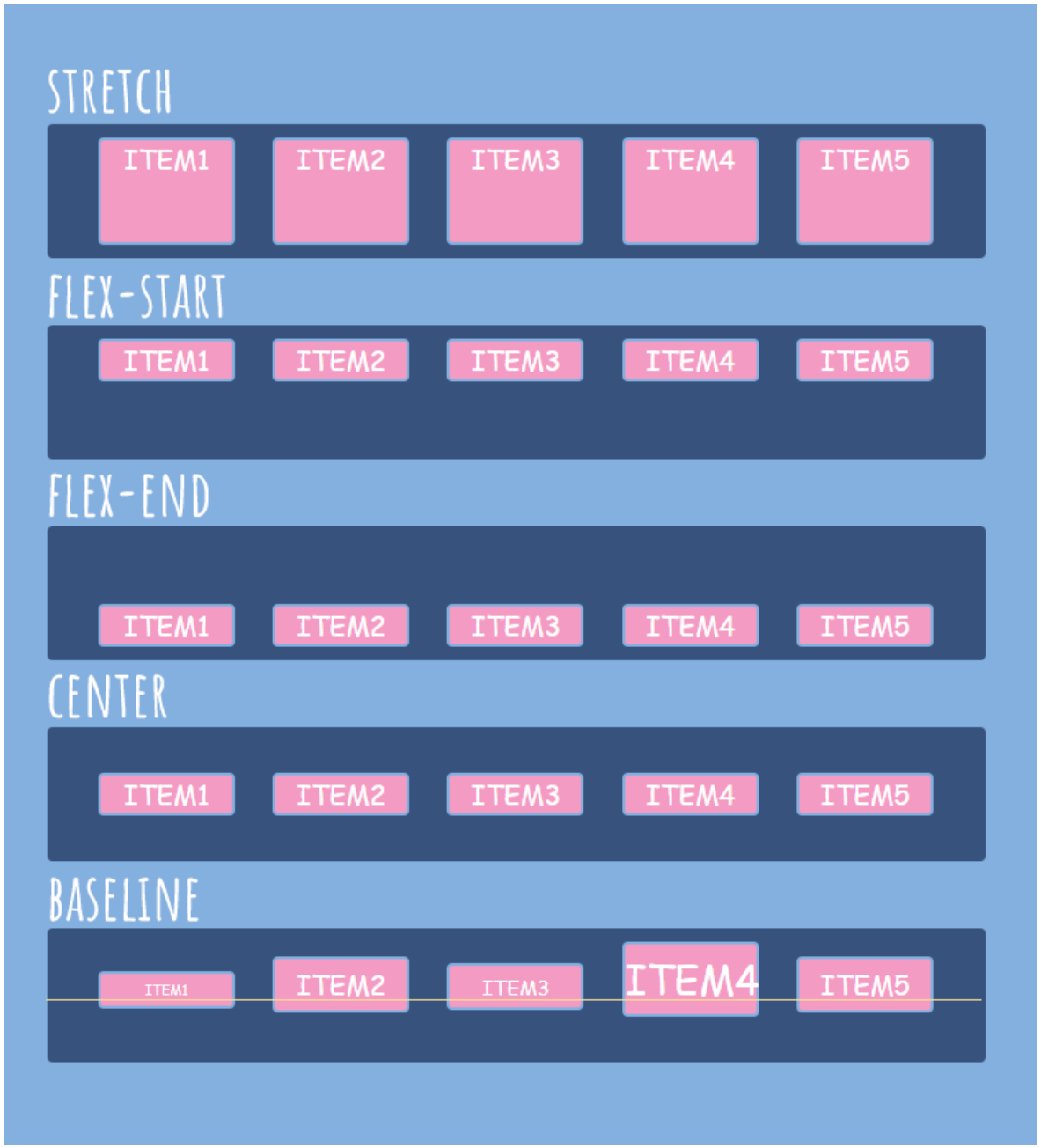
```
.container {
  align-items: stretch | flex-start | flex-end | center | baseline;
}
```


STRETCH
ITEM1
ITEM2
ITEM3
ITEM4
ITEM5
FLEX-START
ITEM1
ITEM2
ITEM3
ITEM4
ITEM5
FLEX-END
ITEM1
ITEM2
ITEM3
ITEM4
ITEM5
CENTER
ITEM1
ITEM2
ITEM3
ITEM4
ITEM5
BASELINE
ITEM1
ITEM2
ITEM3
ITEM4
ITEM5

How to Align Elements Vertically

```
.container {
  justify-content: flex-start | flex-end | center | space-between |
}
```

To align flex items vertically, use the `align-items`, `align-content` or `align-self` properties.


STRETCH
ITEM1
ITEM2
ITEM3
ITEM4
ITEM5
FLEX-START
ITEM1
ITEM2
ITEM3
ITEM4
ITEM5
FLEX-END
ITEM1
ITEM2
ITEM3
ITEM4
ITEM5
CENTER
ITEM1
ITEM2
ITEM3
ITEM4
ITEM5
BASELINE
ITEM1
ITEM2
ITEM3
ITEM4
ITEM5

You should consider using flexbox when:

- You have a small design to implement.
- You need to align elements.
- You need a content-first design.

Thank you!