

# Page layout

Web development I: Front-end engineering

**Fixed layouts** stay at a specific pixel width regardless of the size of the browser window or text size

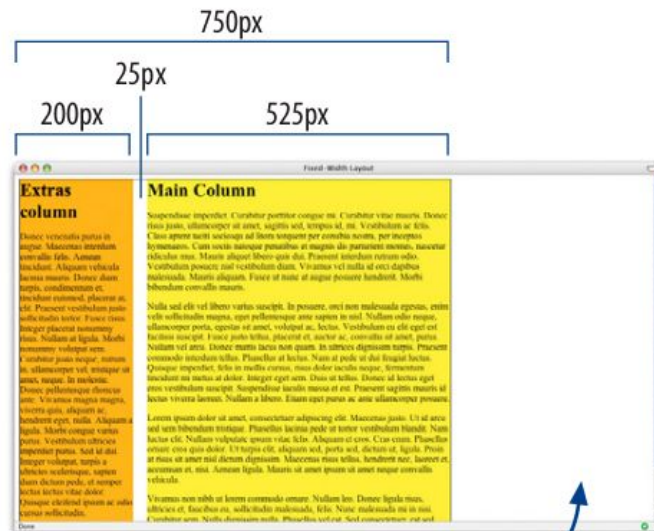
**Fluid (or liquid) layouts** resize proportionally when the browser window resizes

**Elastic layouts** resize proportionally based on the size of the text

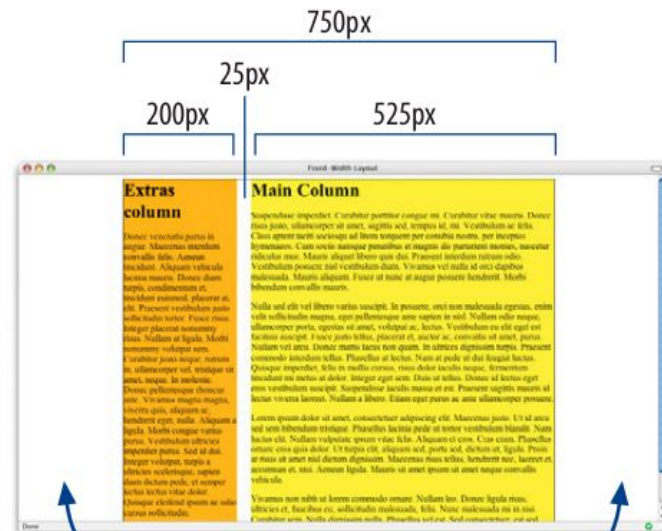
**Hybrid layouts** combine fixed and scalable areas

# Fixed layouts

```
#wrapper {width: 750px;  
position: absolute;  
margin-left: auto;  
margin-right: auto;  
border: 1px solid black;  
padding: 0px;}  
  
#extras {position: absolute;  
top: 0px;  
left: 0px;  
width: 200px;  
background: orange;}  
  
#main {margin-left: 225px;  
background-color: yellow;}
```



Extra space on right



Extra space split on left and right sides

## **Advantages:**

Predictable layout

Better control over line length

Easier to design and produce

## **Disadvantages:**

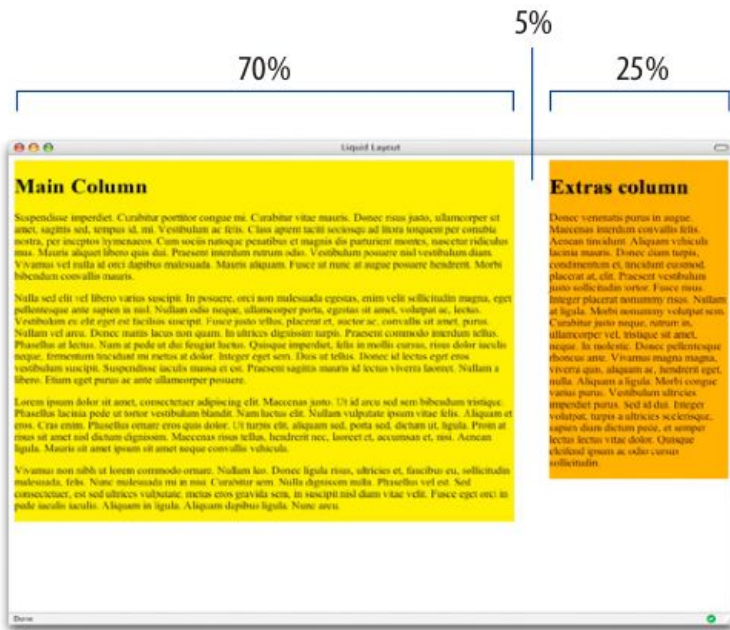
Horizontal scrolling on small screens

Lots of whitespace on large screens

Short line lengths at large text sizes

Not user-controllable

# Fluid layouts



```
div#main {  
  width: 70%;  
  margin-right: 5%;  
  float: left;  
  background: yellow;  
}
```

```
div#extras {  
  width: 25%;  
  float: left;  
  background: orange;  
}
```

## Advantages:

Follow the nature of the Web

Avoid potentially empty spaces.

Users can control the width of content

No horizontal scrollbars

## Disadvantages:

On large screens, line lengths can get very long

Less predictable: elements may be too spread out or too cramped

More difficult to achieve whitespace

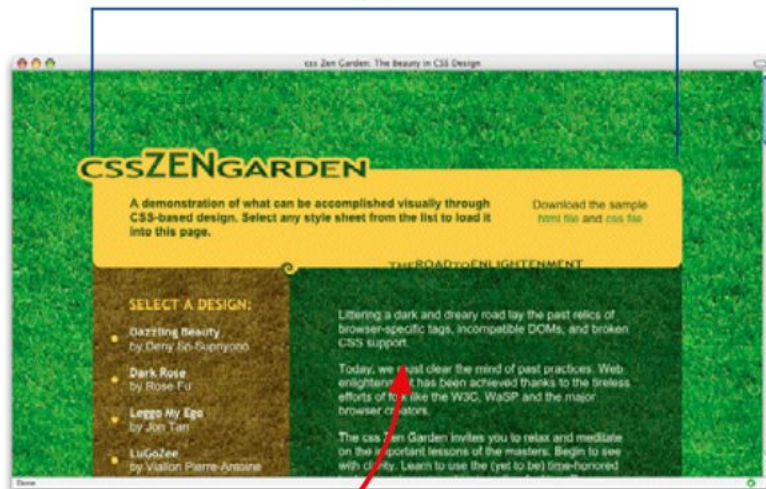
More complex calculation of measurements

# Elastic layouts

48 em



48 em



Line length and line breaks stay the same

## Advantages:

Consistent layout while flexible text sizes

Better control over line lengths

## Disadvantages:

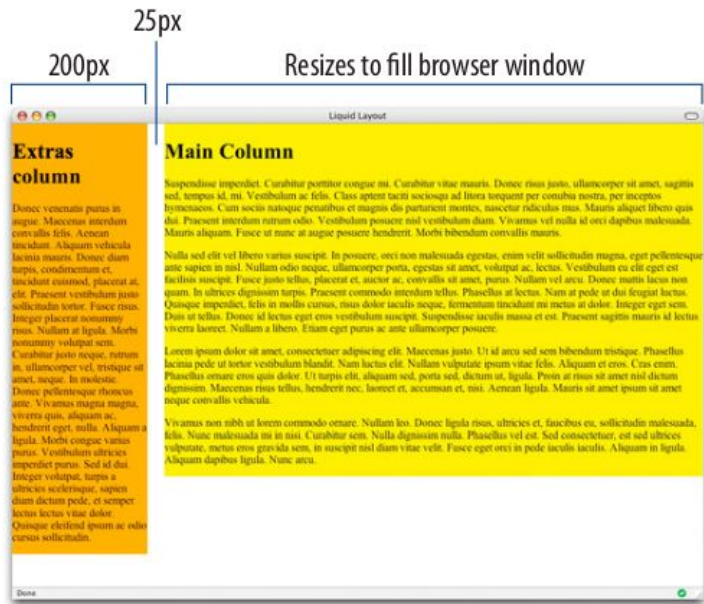
Rescaling of media is more challenging

On large screens, window width might exceed device width

More complicated to create than fixed-width layouts



# Hybrid layouts

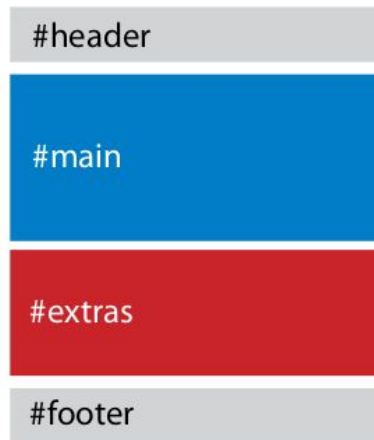


```
div#main {  
  width: auto;  
  position: absolute;  
  top: 0;  
  left: 225px;  
  background: yellow; }  
  
div#extras {  
  width: 200px;  
  position: absolute;  
  top: 0;  
  left: 0;  
  background: orange; }
```

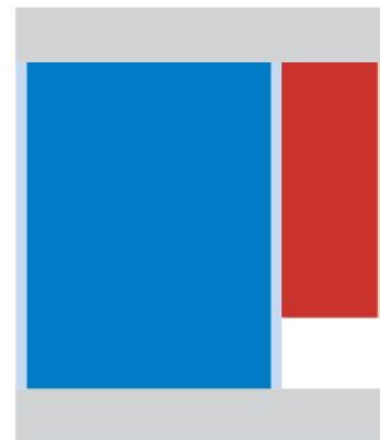
# Page layout techniques

## Two columns, fluid layout

```
#main { float: left; width: 60%; }  
#extras { float: left; width: 25%; }  
#footer { clear: left; }
```



Source order

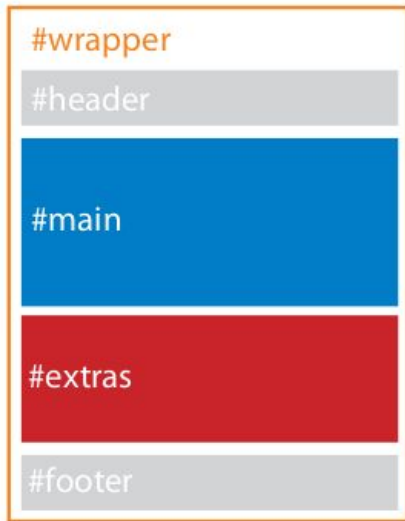


Layout

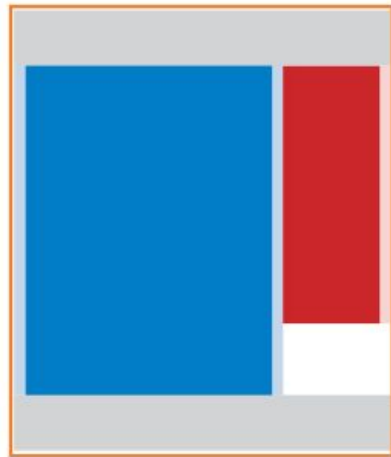
# Page layout techniques

## Two columns, fixed-width layout

```
#wrapper { width: 960px; }  
  
#main { float: left; width: 650px; }  
  
#extras { float: left; width: 250px; }  
  
#footer { clear: left; }
```



Source order



Layout

# Page layout techniques

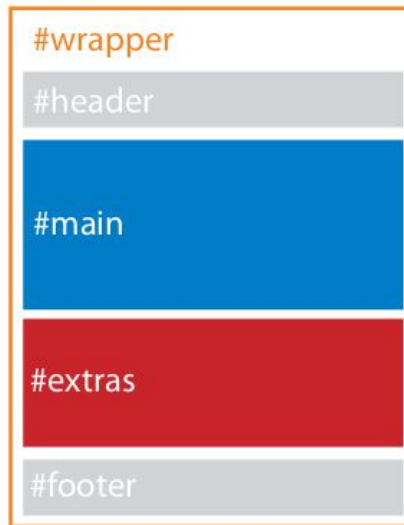
## Two columns, fixed width, centered

```
#wrapper { width: 960px; margin: 0 auto; }
```

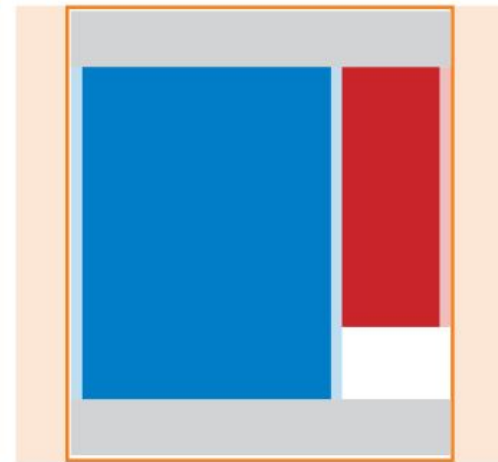
```
#main { float: left; width: 650px; }
```

```
#extras { float: left; width: 250px; }
```

```
#footer { clear: left; }
```



Source order



Layout

# Page layout techniques

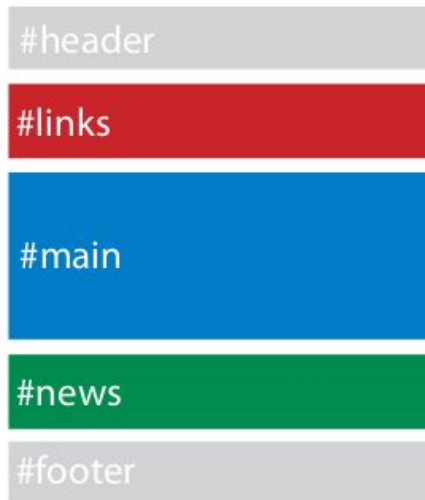
## Three columns, fluid layout

```
#links { float: left; width: 22.5%; }
```

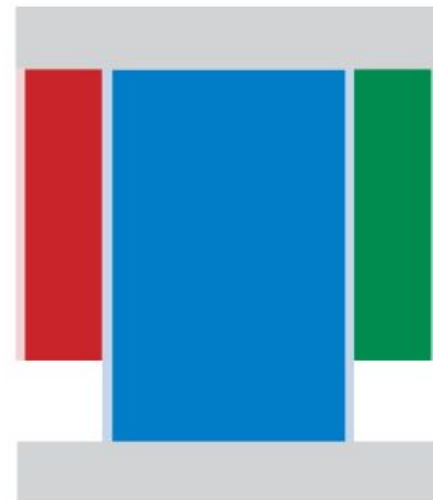
```
#main { float: left; width: 45%; }
```

```
#news { float: left; width: 22.5%; }
```

```
#footer { clear: left; }
```



Source order



Layout

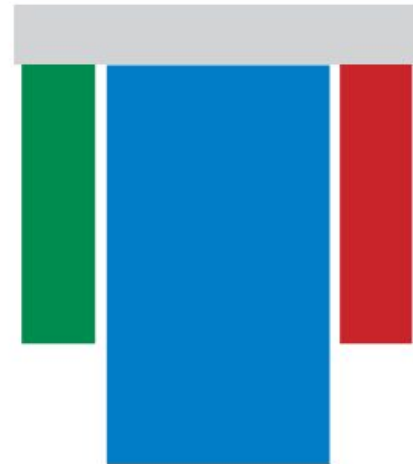
# Page layout techniques

## Three columns, positioned, fluid layout

```
#content { position: relative; }  
  
#main { width: 50%; position: absolute;  
top: 0; left: 25%; }  
  
#news { width: 20%; position: absolute;  
top: 0; left: 2.5%; }  
  
#links { width: 20%; position: absolute;  
top: 0; right: 2.5%; }
```



Source order



Layout

# Page layout techniques

## Three columns, positioned, fixed

```
#wrapper { width: 960px; margin: 0 auto; }
```

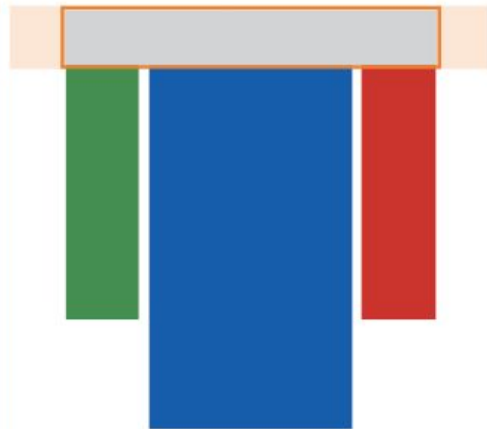
```
#main { width: 520px; position: absolute;  
top: 0; left: 220px; }
```

```
#news { width: 200px; position: absolute;  
top: 0; left: 0; }
```

```
#links { width: 200px; position: absolute;  
top: 0; right: 0; }
```



Source order

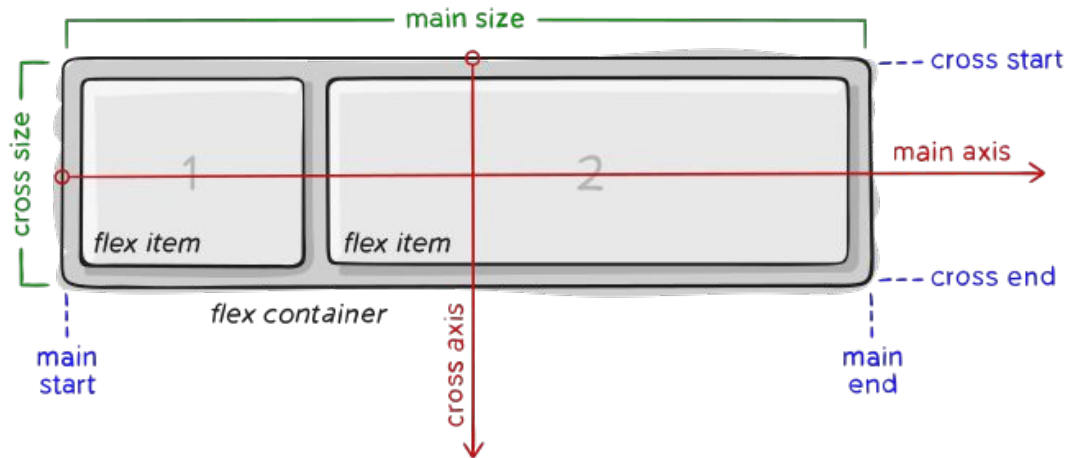


Layout

Regular layouts are based on block and inline flow directions

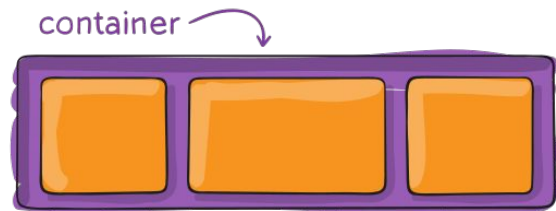
The flex layout is based on single-axis **directions**

Full spec: <https://www.w3.org/TR/css-flexbox-1/>





# Flexbox



display

flex-direction

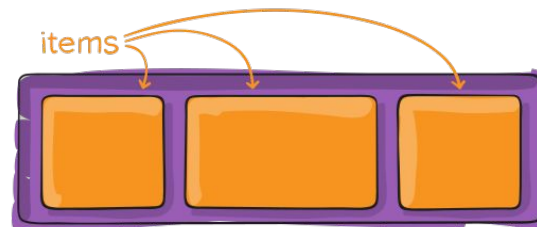
flex-wrap

flex-flow

justify-content

align-items

align-content



order

flex-grow

align-self

flex-shrink

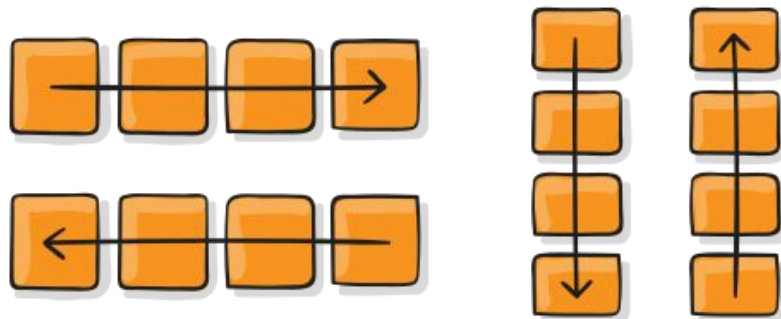
flex-basis

flex

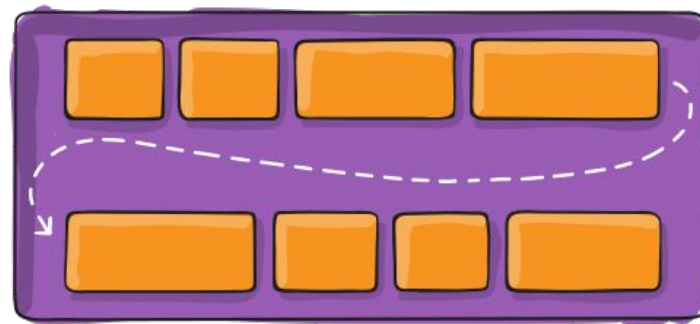
# Flexbox: basic markup

```
.container {  
    display: flex;  
}  
  
.items {  
    flex: 1;  
}
```

# Flexbox: container



**flex-direction:** row | row-reverse | column  
| column-reverse



**flex-wrap:** nowrap | wrap | wrap-reverse;

# Flexbox: container

**flex-flow:** shorthand for `flex-direction` and `flex-wrap`

**justify-content:** `flex-start` | `flex-end` | `center` | `space-between` | `space-around` | `space-evenly` | `start` | `end` | `left` | `right` ... + `safe` | `unsafe`

flex-start



flex-end



center



space-between



space-around



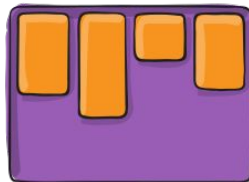
space-evenly



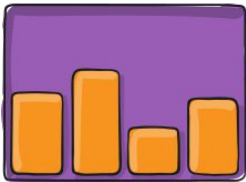
# Flexbox: container

**align-items:** stretch | flex-start | flex-end | center | baseline  
| first baseline | last baseline | start | end | self-start |  
self-end + ... safe | unsafe

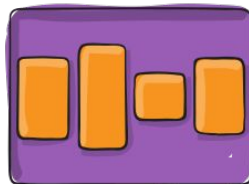
flex-start



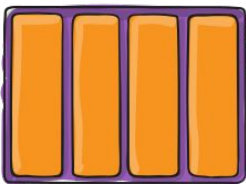
flex-end



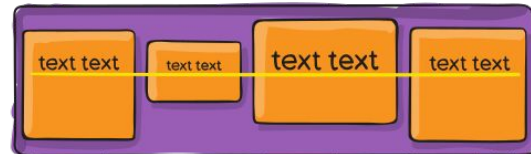
center



stretch



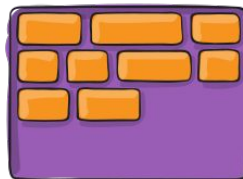
baseline



# Flexbox: container

**align-content:** flex-start | flex-end | center | space-between  
| space-around | space-evenly | stretch | start | end |  
baseline | first baseline | last baseline + ... safe | unsafe

flex-start



flex-end



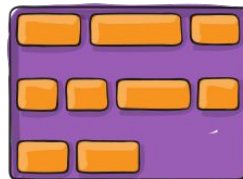
center



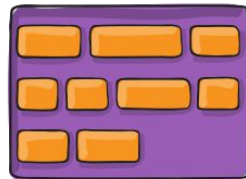
stretch



space-between

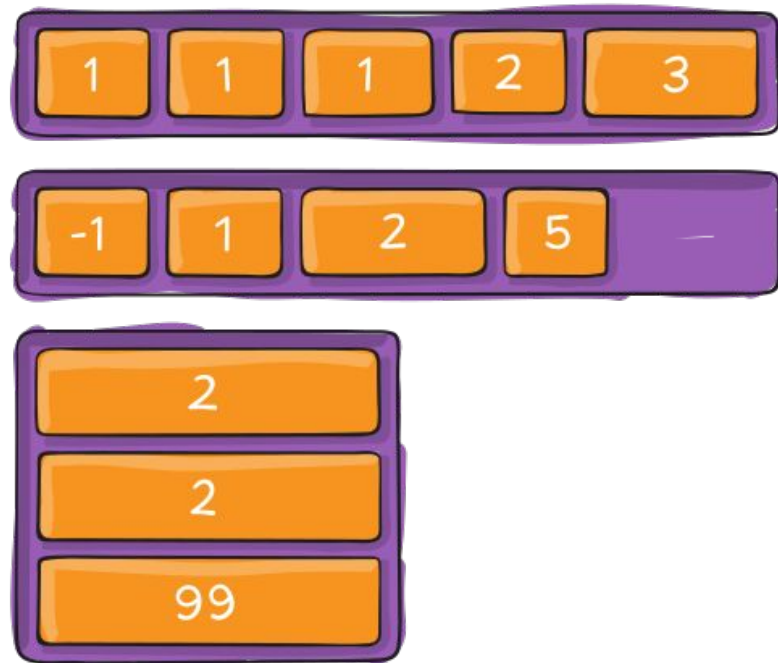


space-around



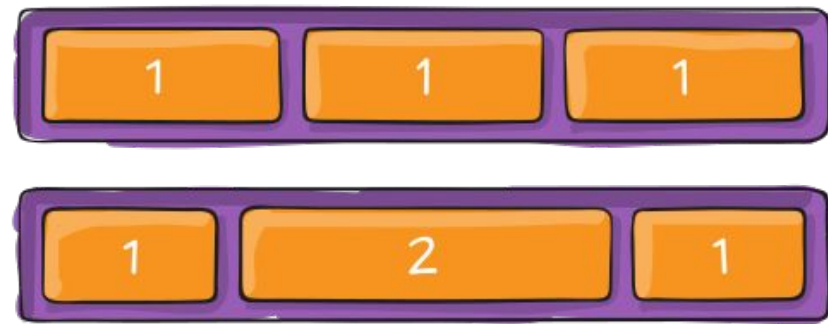
# Flexbox: items

**order:** number (default: 0)



# Flexbox: items

**flex-grow:** number (default: 0)





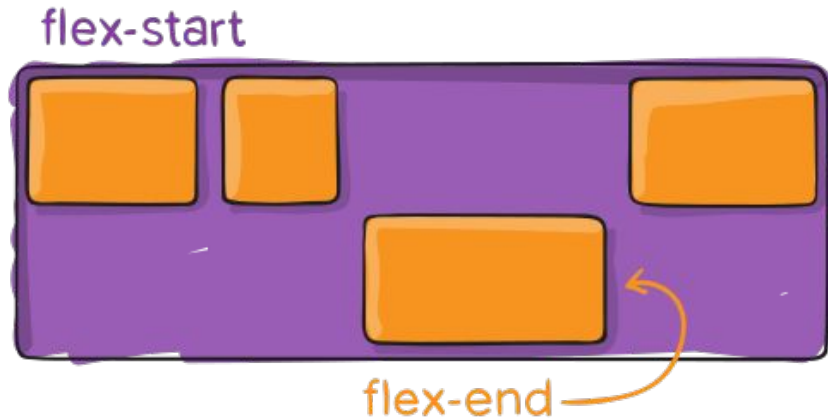
# Flexbox: items

**align-self:** auto | flex-start | flex-end | center | baseline | stretch

**flex-shrink:** number (default: 1)

**flex-basis:** number | auto

**flex:** none | flex-grow flex-shrink | flex-basis



# Grid

Fairly advanced layout system

Sometimes [more convenient than flexbox](#)

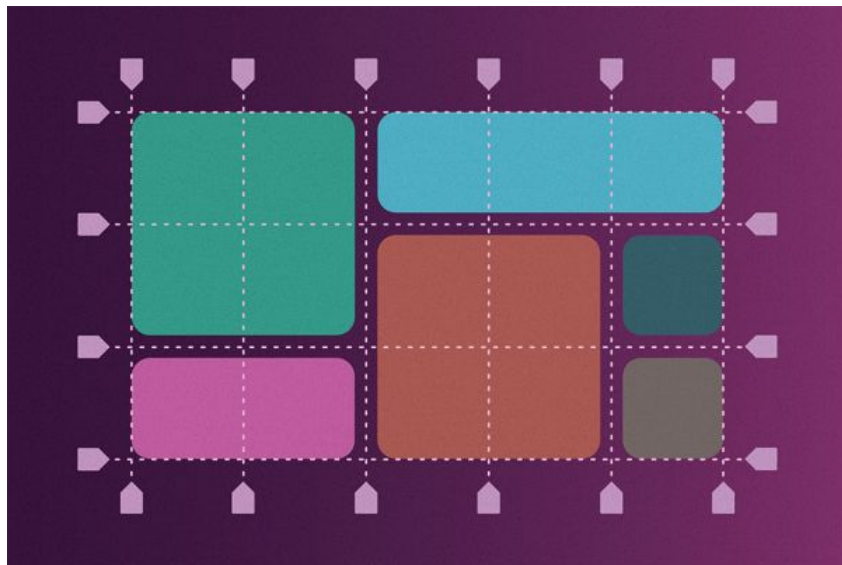
The grid layout is based on dual-axis  
**alignments**

Guide:

<https://css-tricks.com/snippets/css/complete-guide-grid/>

Full spec:

<https://www.w3.org/TR/css-grid-1/>



# Grid: basic markup

```
.container {  
  display: grid;  
}  
  
.item1 {  
  grid-column: 1; grid-row: 1;  
}  
  
.item2 {  
  grid-column: 1; grid-row: 3;  
}  
...
```

