Layout using Flexbox

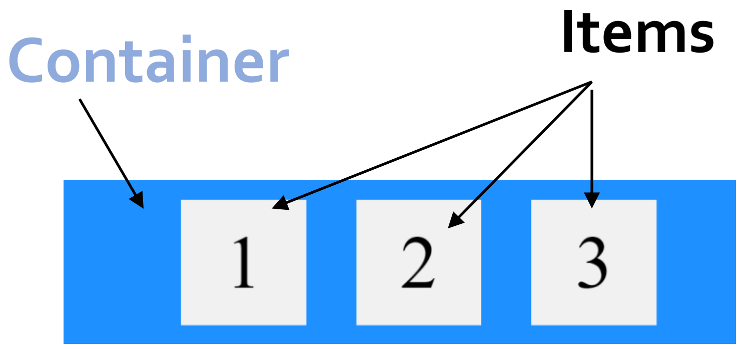
https://css-tricks.com/snippets/css/a-guide-to-flexbox/
https://marina-ferreira.github.io/tutorials/css/flexbox/

< Back

Flexbox

- The Flexbox provide an efficient way to define **one-dimensional layout** that allows easy control of **space distribution** and **alignment** of items in a container
    - Enable **responsive design** to accommodate different screen sizes

```
 .flex-container {
     display: flex;
     justify-content: center;
 }

<div class="flex-container">
    <div>1</div>
    <div>2</div>
    <div>3</div>
</div>
```



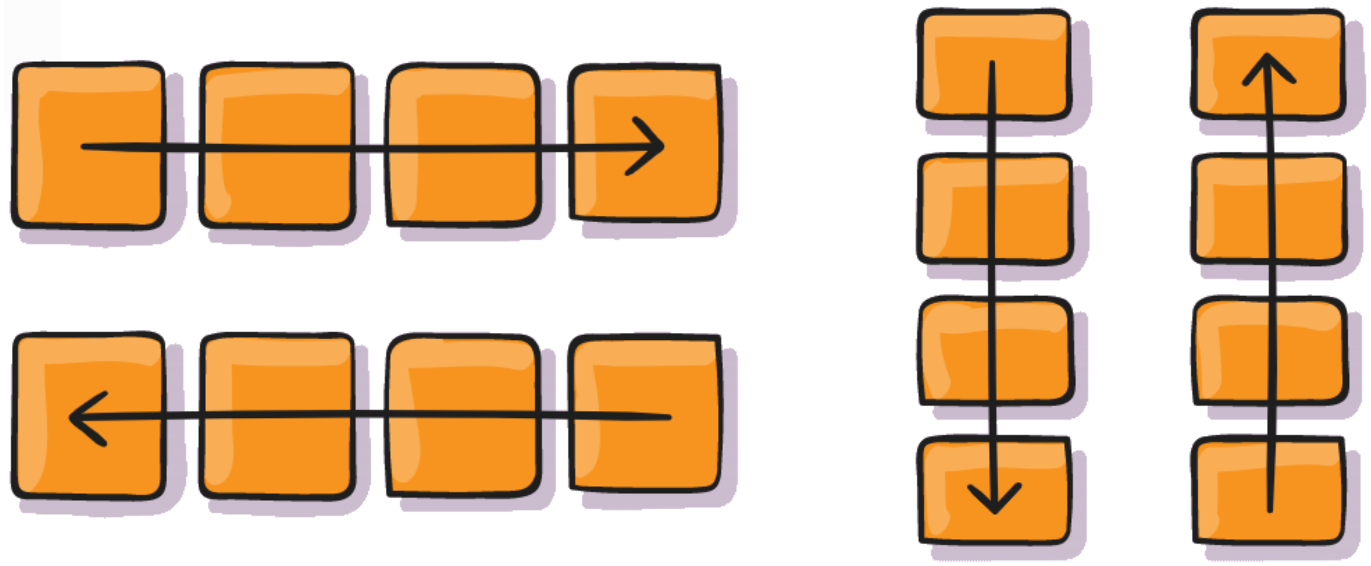
https://www.w3schools.com/css/css3_flexbox.asp

Flex Container Properties

- flex-direction
- flex-wrap
- justify-content
- align-items
- align-content

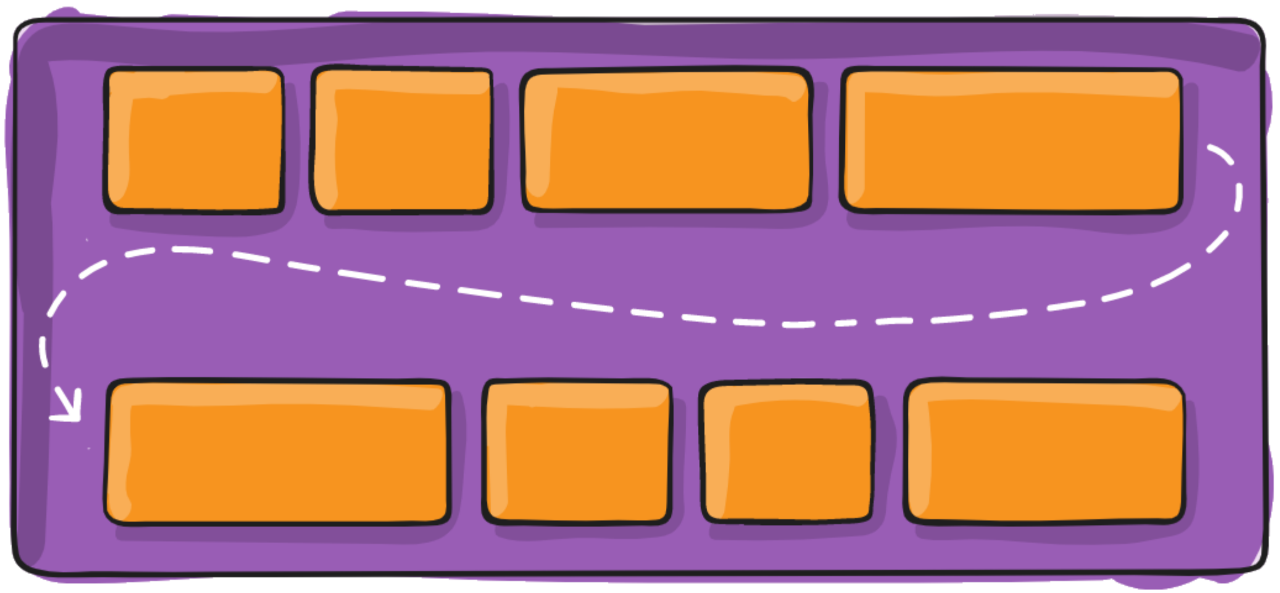
flex-direction

- Layout Flex items either in **horizontal rows** or **vertical columns:**
  - **row** (default): left to right
  - **row-reverse**: right to left
  - **column**: top to bottom
  - **column-reverse**: bottom to top

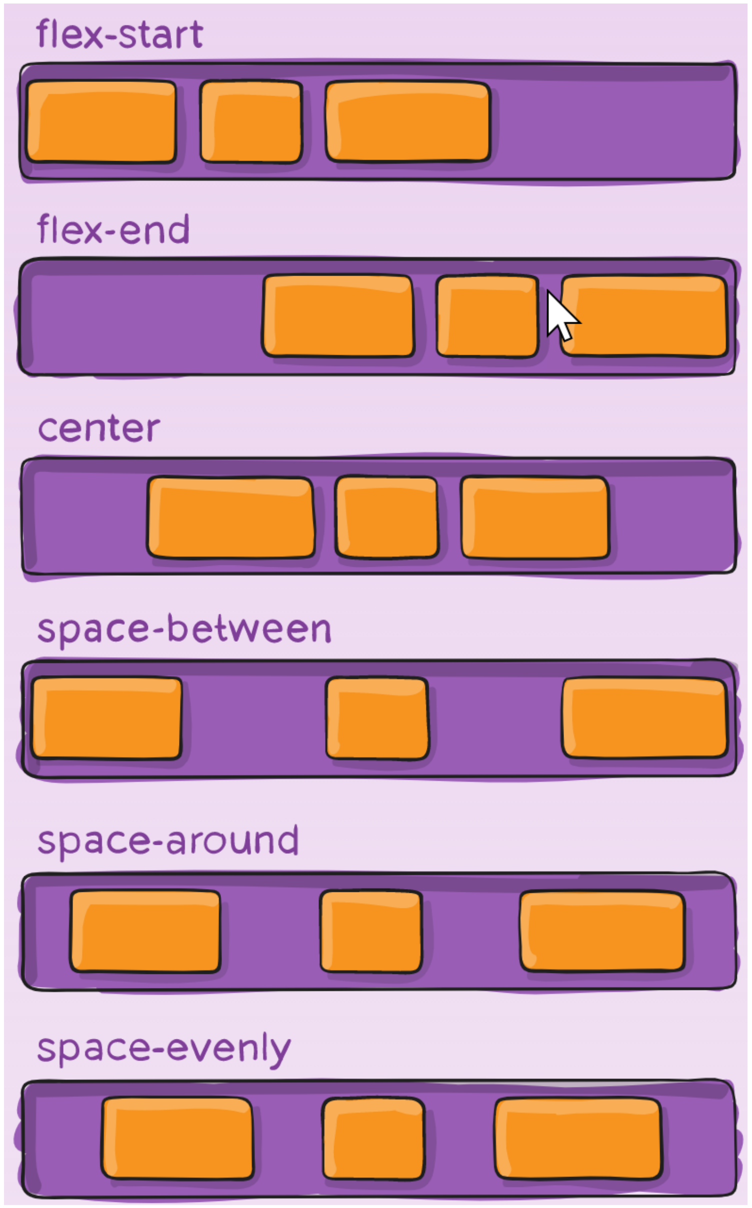


flex-wrap

- **nowrap** (default): all flex items will be on one line
- **wrap**: flex items will wrap onto multiple lines, from top to bottom
- wrap-reverse: flex items will wrap onto multiple lines from bottom to top

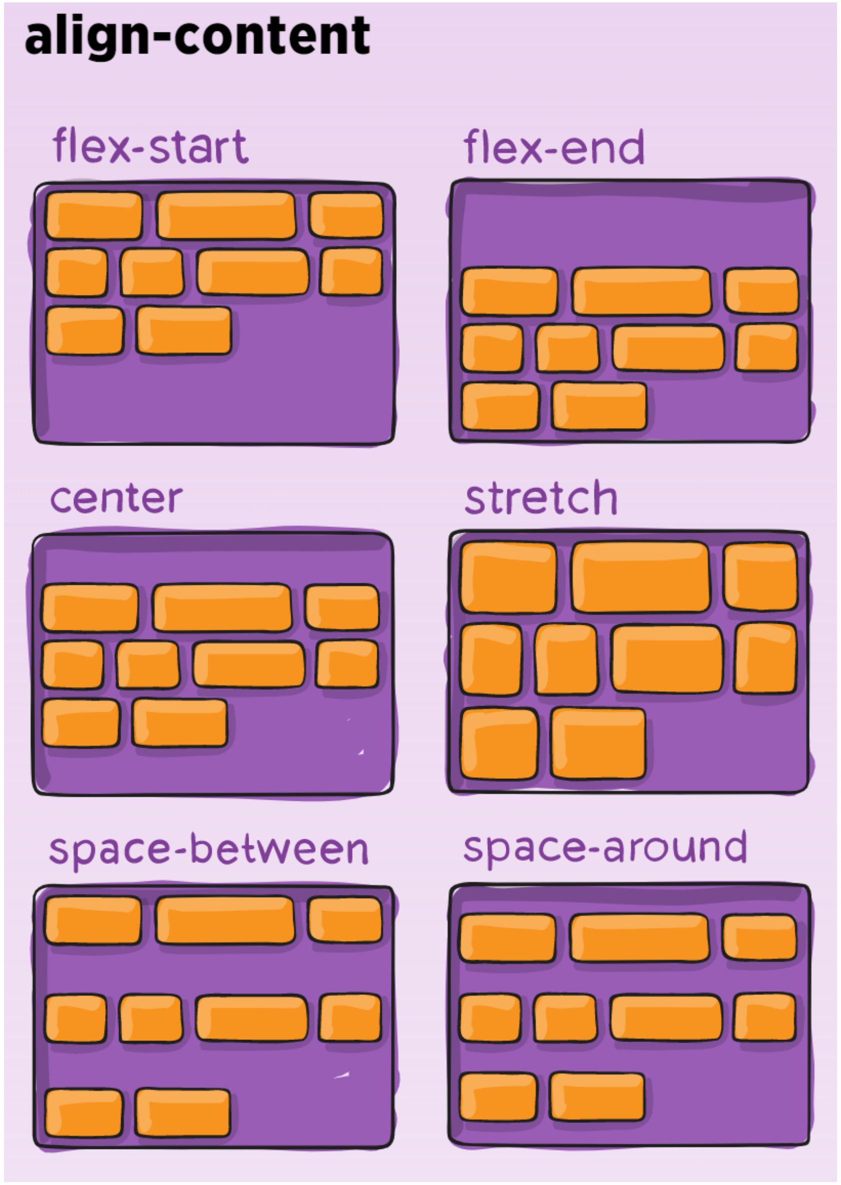


justify-content



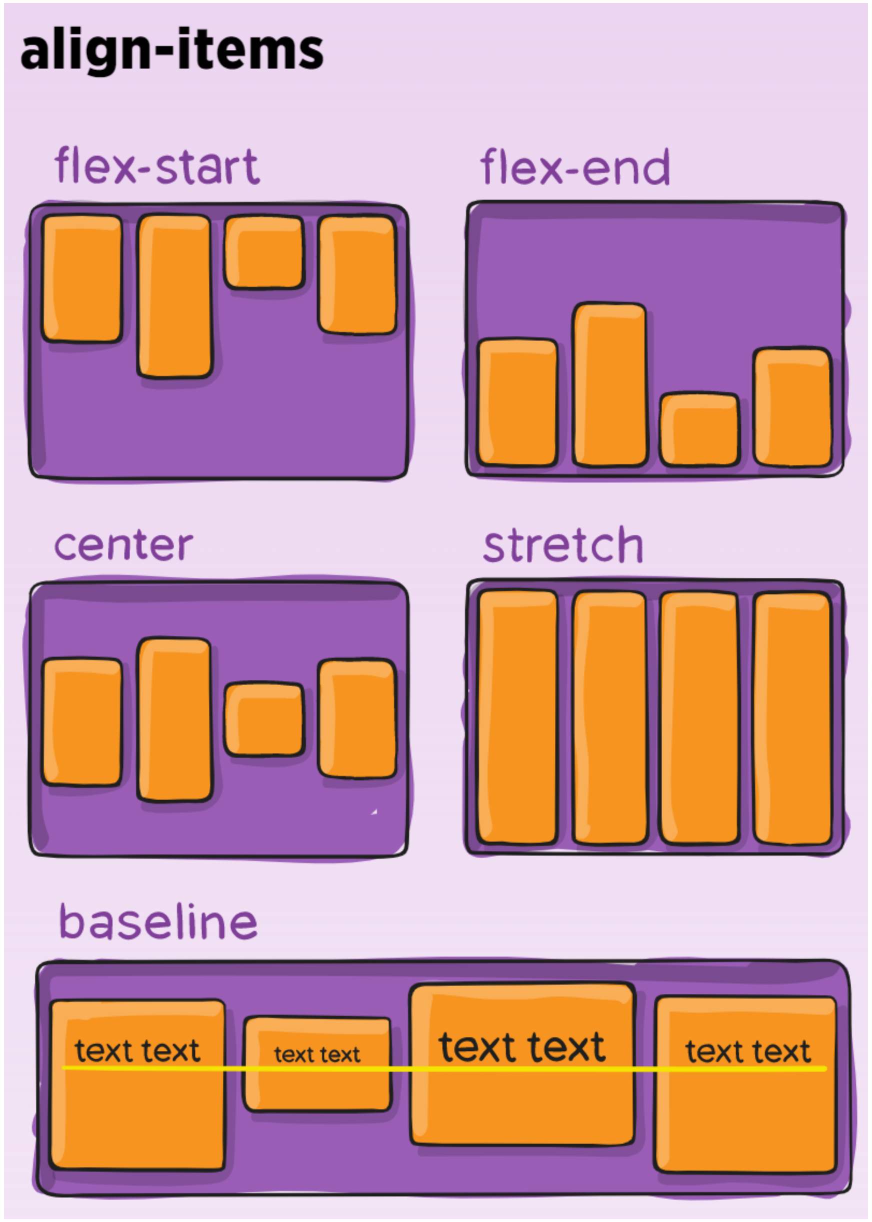
- Distribute extra leftover **free space** along the **main axis**
- **flex-start** is the default: items are packed toward the start

align-content

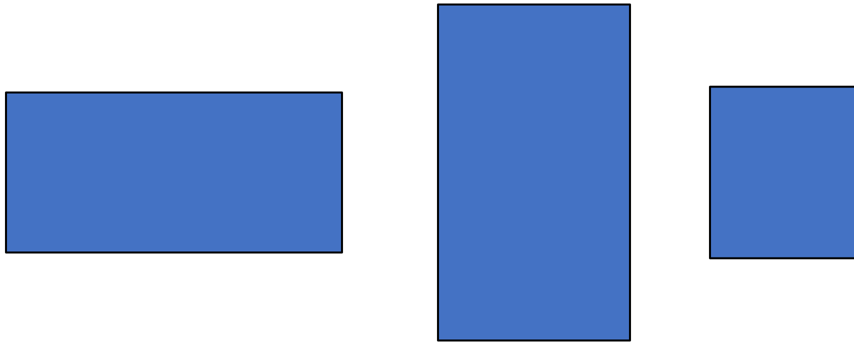


- Distribute extra leftover free space along the **cross axis**
- stretch is the default: lines stretch to take up the remaining space

align-items



- Defines how flex items are laid out along the cross axis
- Stretch is the default: flex items stretch to fill the container



Properties for flex items

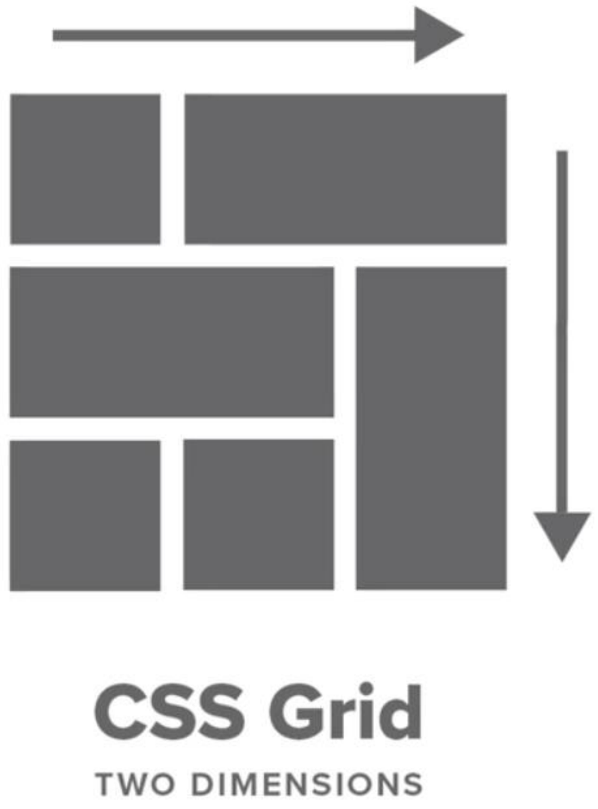
- order
- flex-grow
- flex-shrink
- flex-basis
- align-self

https://www.w3schools.com/css/css3_flexbox_items.asp

Layout using Grid

CSS Grid

- CSS Grid is a **two-dimensional layout** system to design the page layout
- Two Steps to use CSS Grid:
  1. Define a grid
  2. Place items within the grid



Watch and practice @
https://mozilladevelopers.github.io/playground/css-grid

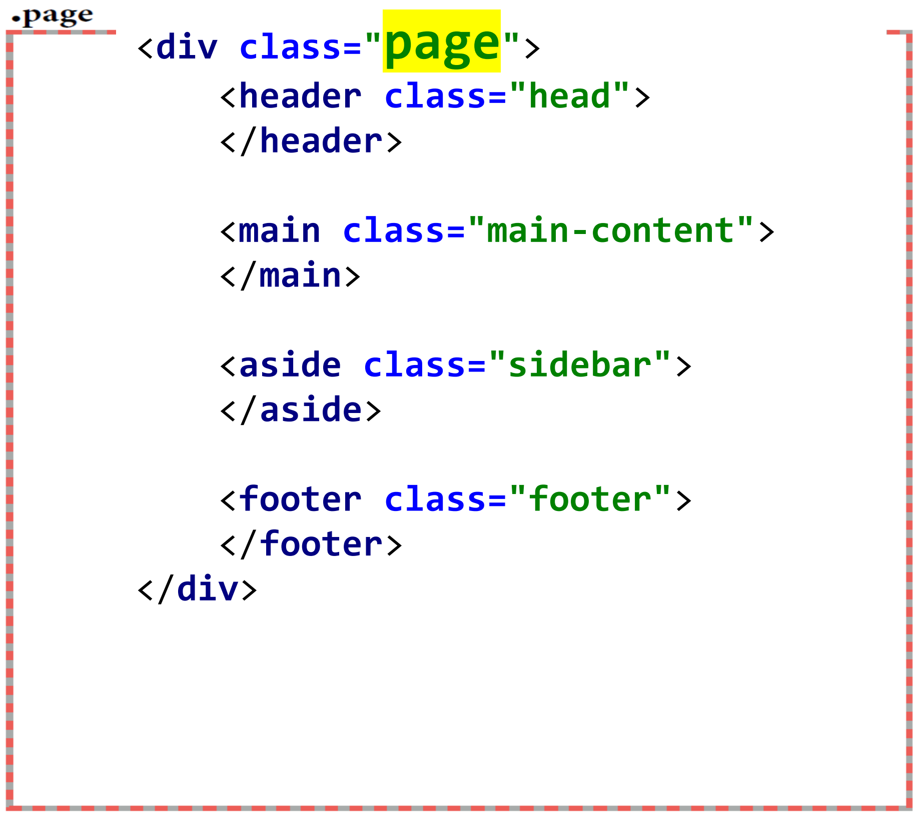
Grid **container**

- Grid **container** is defined by setting the ***display*** property of the container element to ***grid***

- css:

```
.page {
    display: grid;
}
```

This creates a grid container



```
<div class="page">
    <header class="head">
    </header>

    <main class="main-content">
    </main>

    <aside class="sidebar">
    </aside>

    <footer class="footer">
    </footer>
</div>
```

Grid **item**

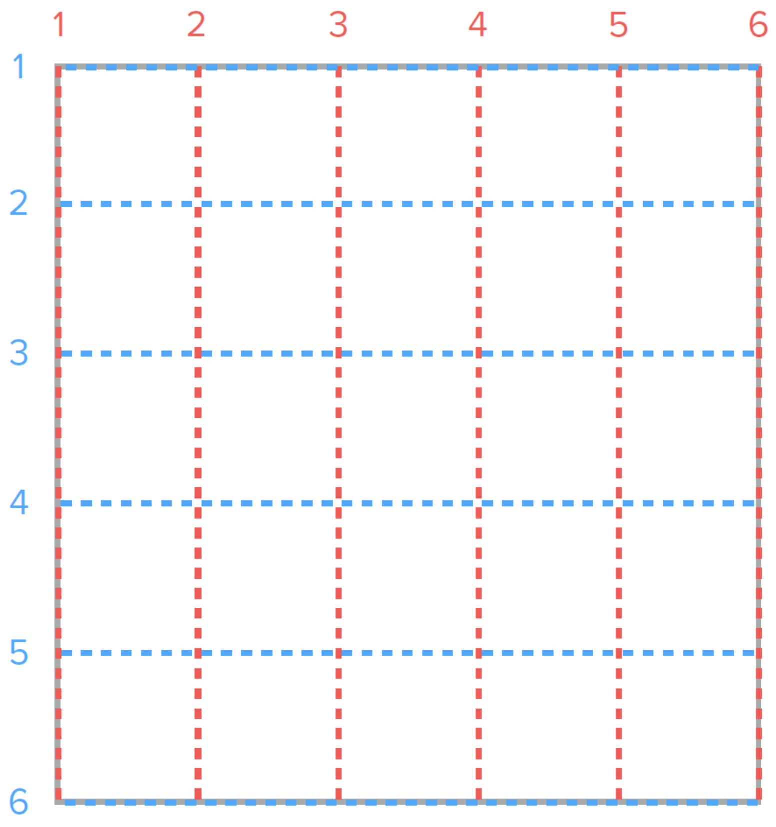
- Grid item = *Element that is a direct descendant of the grid* ***container***

```html
<div class="page">
    <header class="head">
    </header>

    <main class="main-content">
    </main>

    <aside class="sidebar">
    </aside>

    <footer class="footer">
    </footer>
</div>
```

Grid **line**

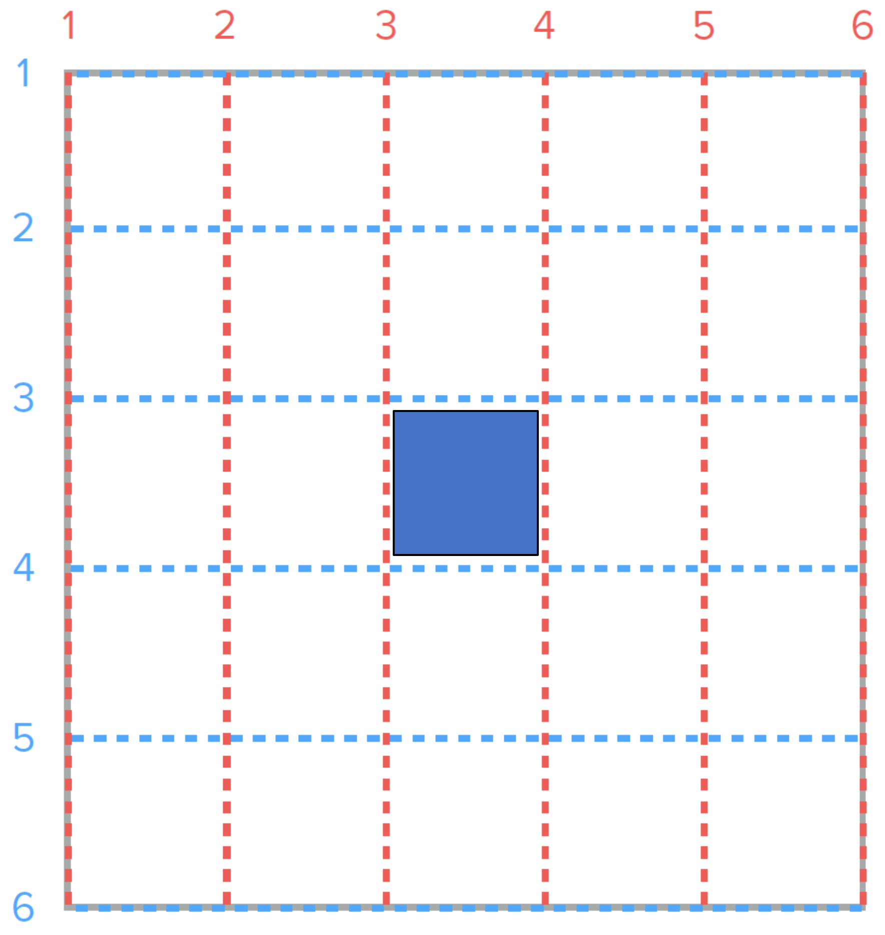
- *Horizontal (**row**) or vertical (**column**) line separating the grid into sections*

- *Grid lines are referenced by numbers, starting and ending with the outer borders of the grid*



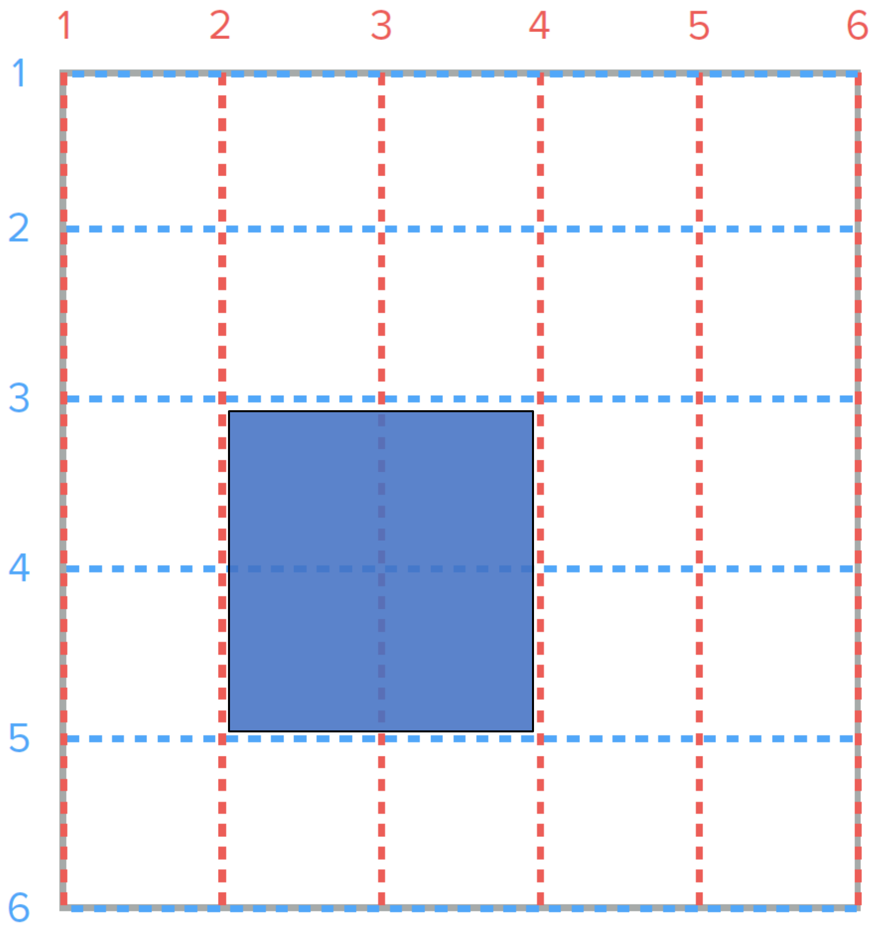
Grid ***cell***

- The *intersection* between a grid row and a grid column



Grid ***area***

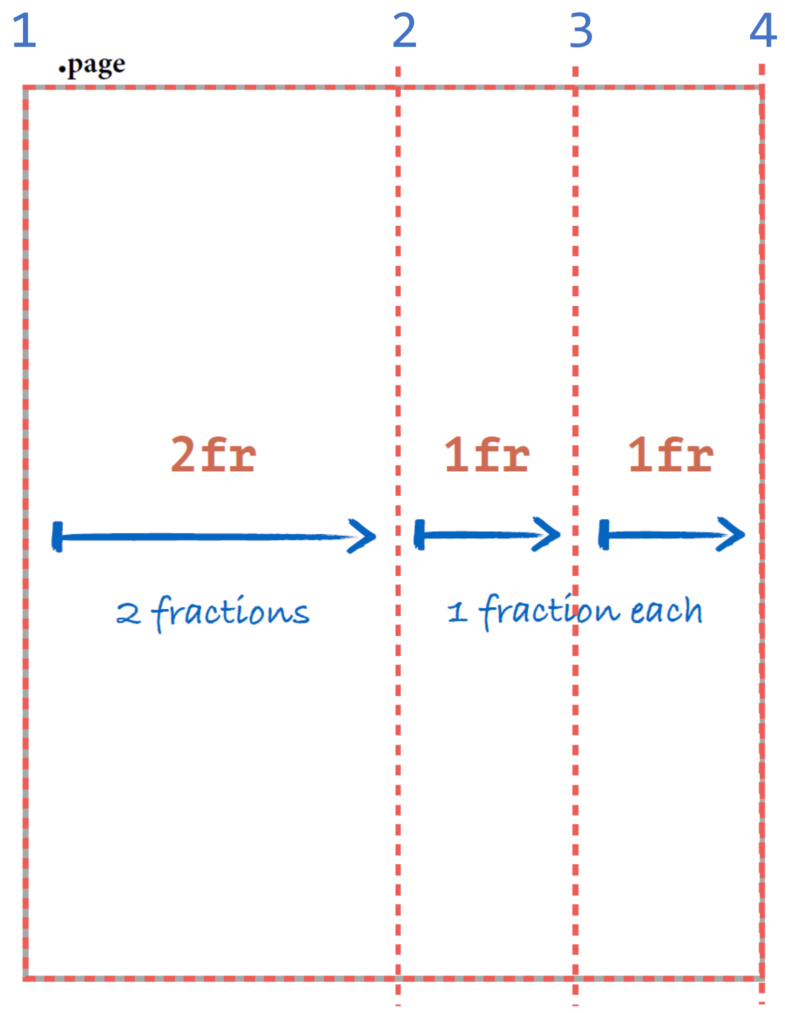
- *Rectangular area* between four specified grid lines
- Grid areas can cover one or more cells
- E.g., blue area between row lines 3 and 5 and column lines 2 and 4



Grid columns

```
grid-template-columns:
    2fr 1fr 1fr;
```

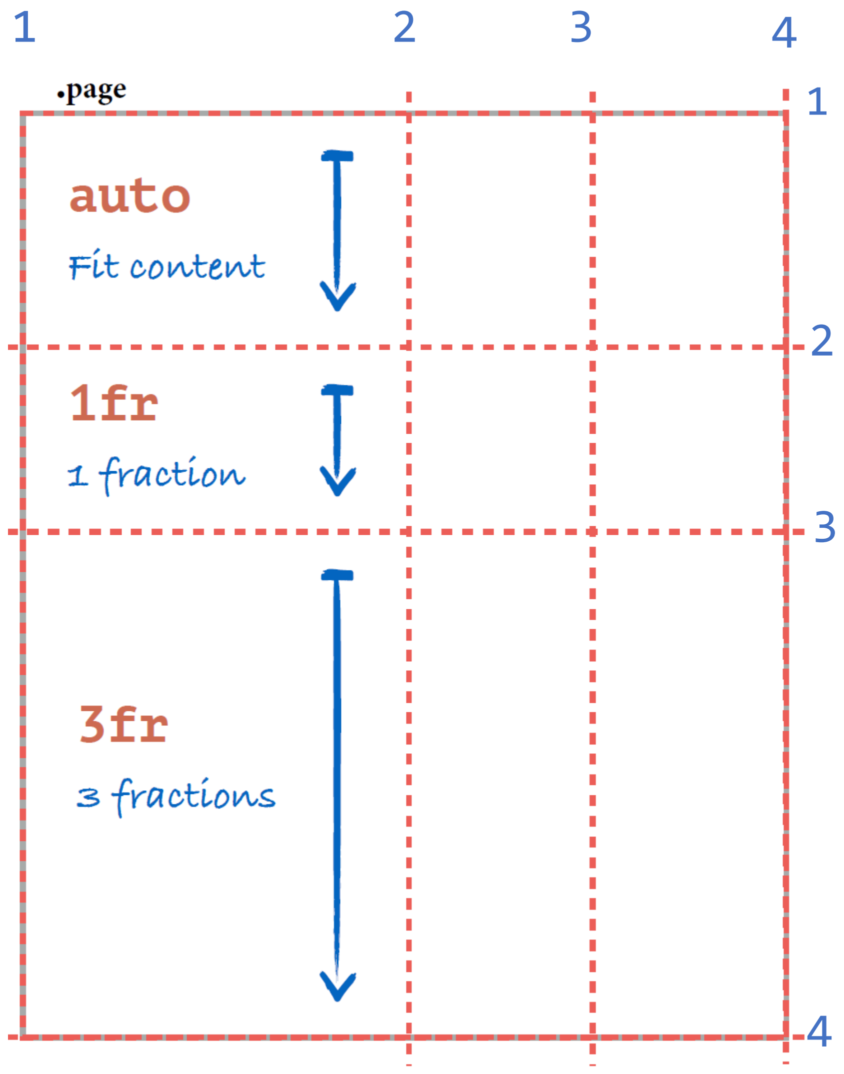
Draws grid lines. Takes list of length values (em, px, %, **fr**, etc.) denoting the distance between each line



Grid rows

```
grid-template-rows:
auto 1fr 3fr;
```

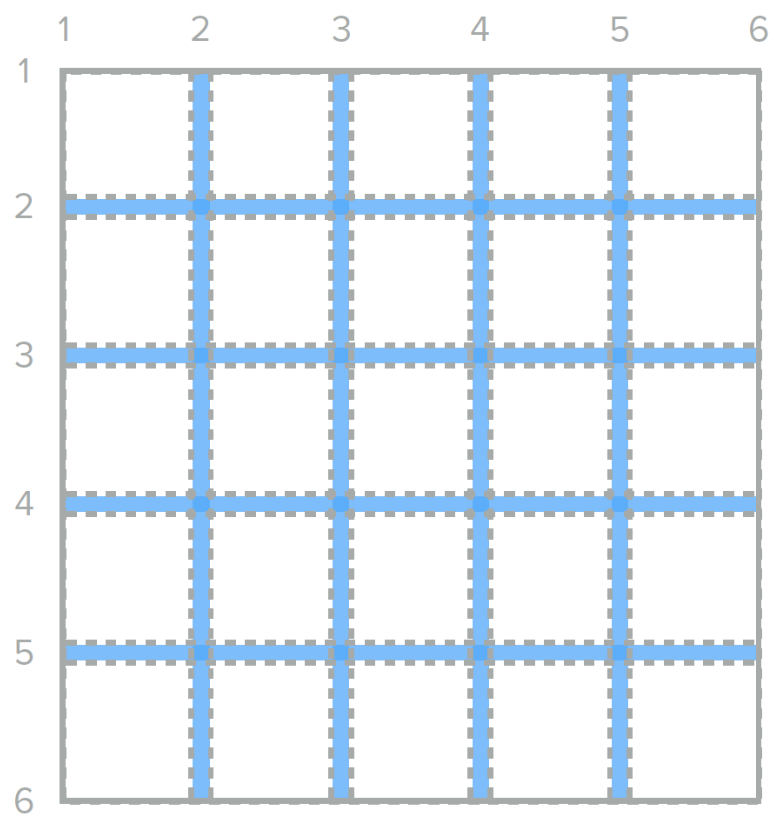
Draws grid lines. Takes list of length values (em, px, %, **fr**, etc.) denoting the distance between each line

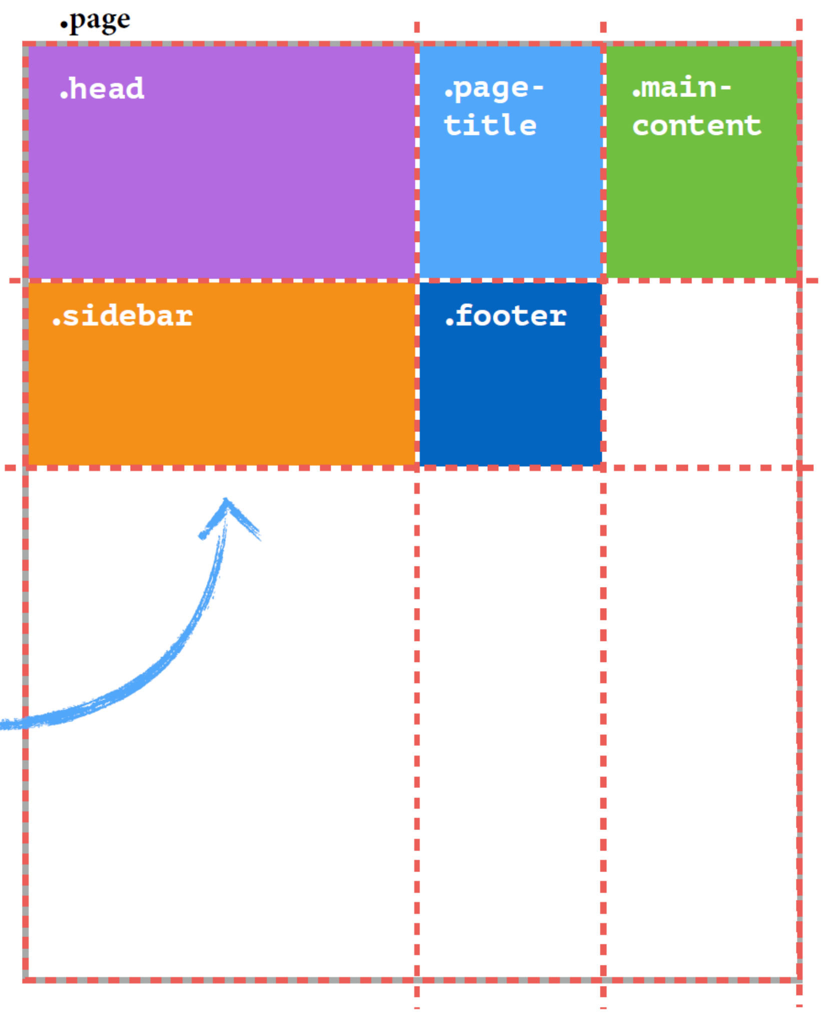


Grid **gap**

- Empty space between grid tracks (shown in **blue**)
- Commonly called ***gutters***

```
.page {
      display: grid;
      grid-gap: 10px;
}
```

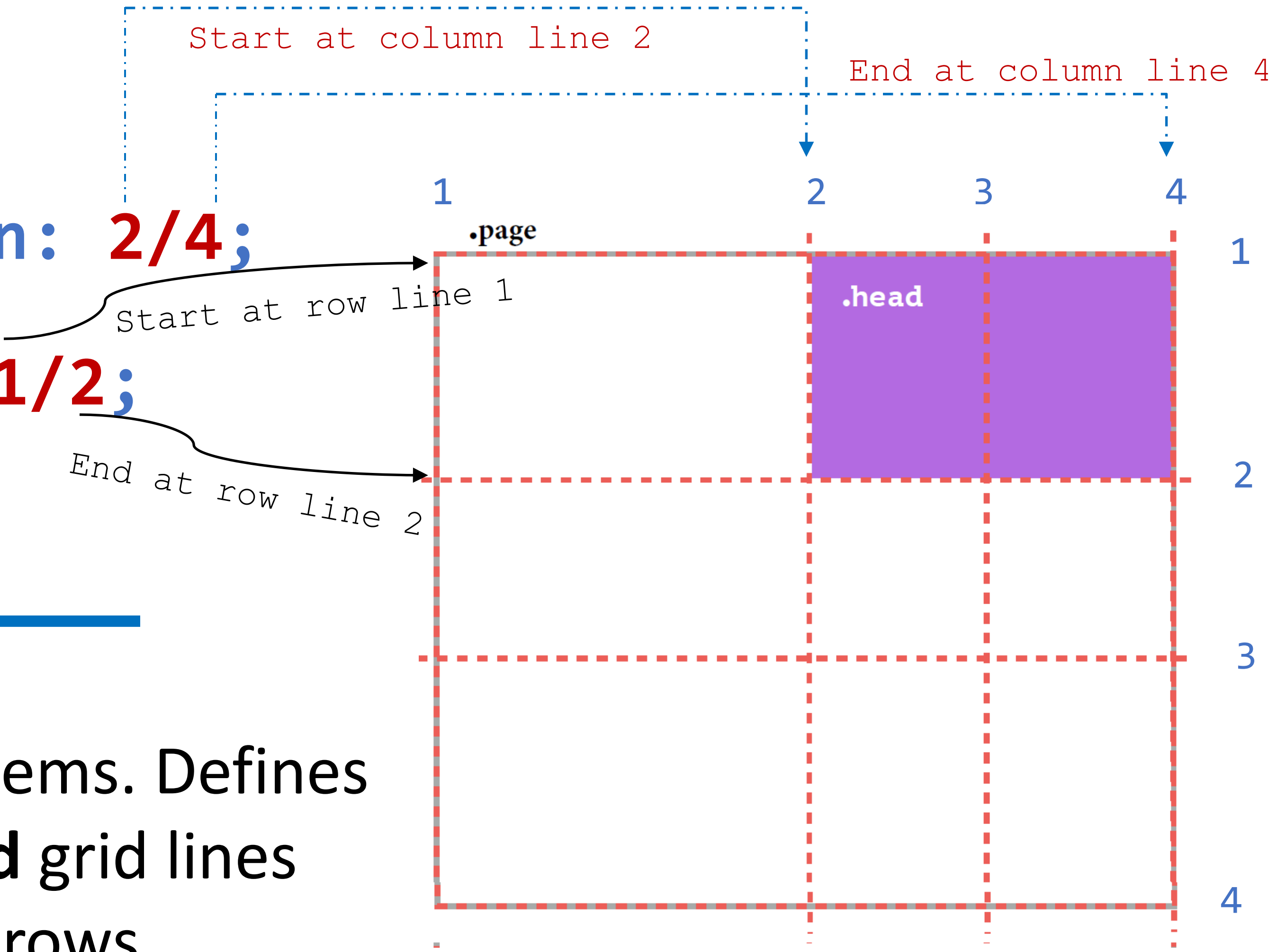




Grid items automatically populate grid from top left to bottom right based on HTML source order.

Items placement in Grid

```
.head {
  grid-column: 2/4;

  grid-row: 1/2;
}
```

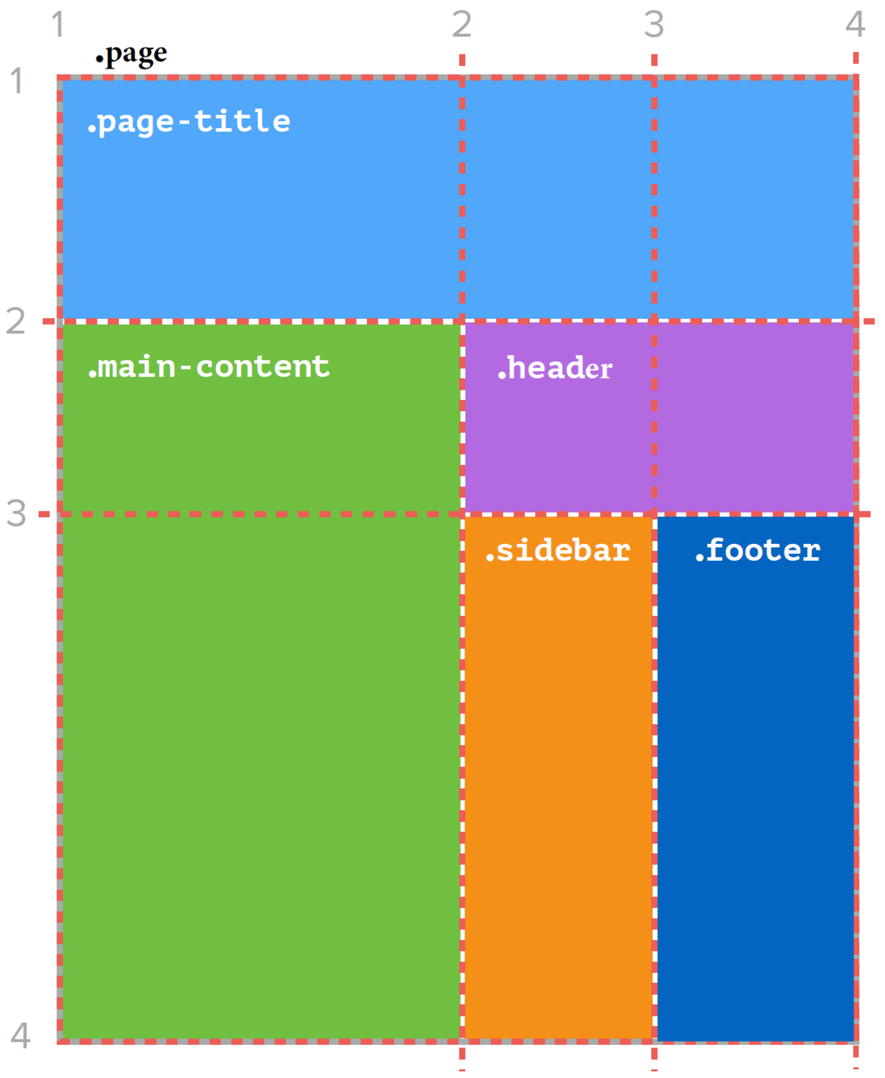


Applied to grid items. Defines the **start** and **end** grid lines for columns and rows

Example

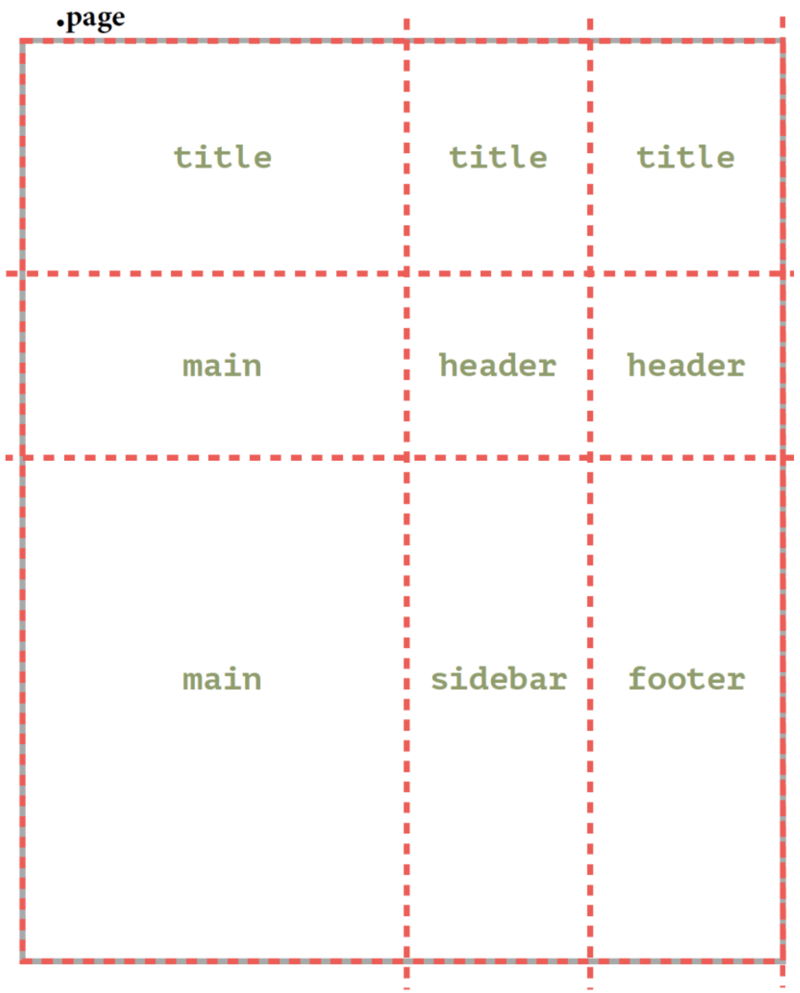
```
.page {
    display: grid;
    grid-template-columns: 2fr 1fr 1fr;
    grid-template-rows: auto 1fr 3fr;
}

.page-title {
    grid-column: 1/4;
    grid-row: 1/2;
}
.header {
    grid-column: 2/4;
    grid-row: 2/3;
}

.main-content {
    grid-column: 1/2;
    grid-row: 2/4;
}
/* etc etc */
```

Ok, but remembering what lines to target seems tricky… especially when the site is responsive



Define grid areas

```
.page {
    display: grid;
    grid-template-columns: 2fr 1fr 1fr;
    grid-template-rows: auto 1fr 3fr;
    grid-template-areas:
            "title title title"
            "main header header"
            "main sidebar footer";
}
```

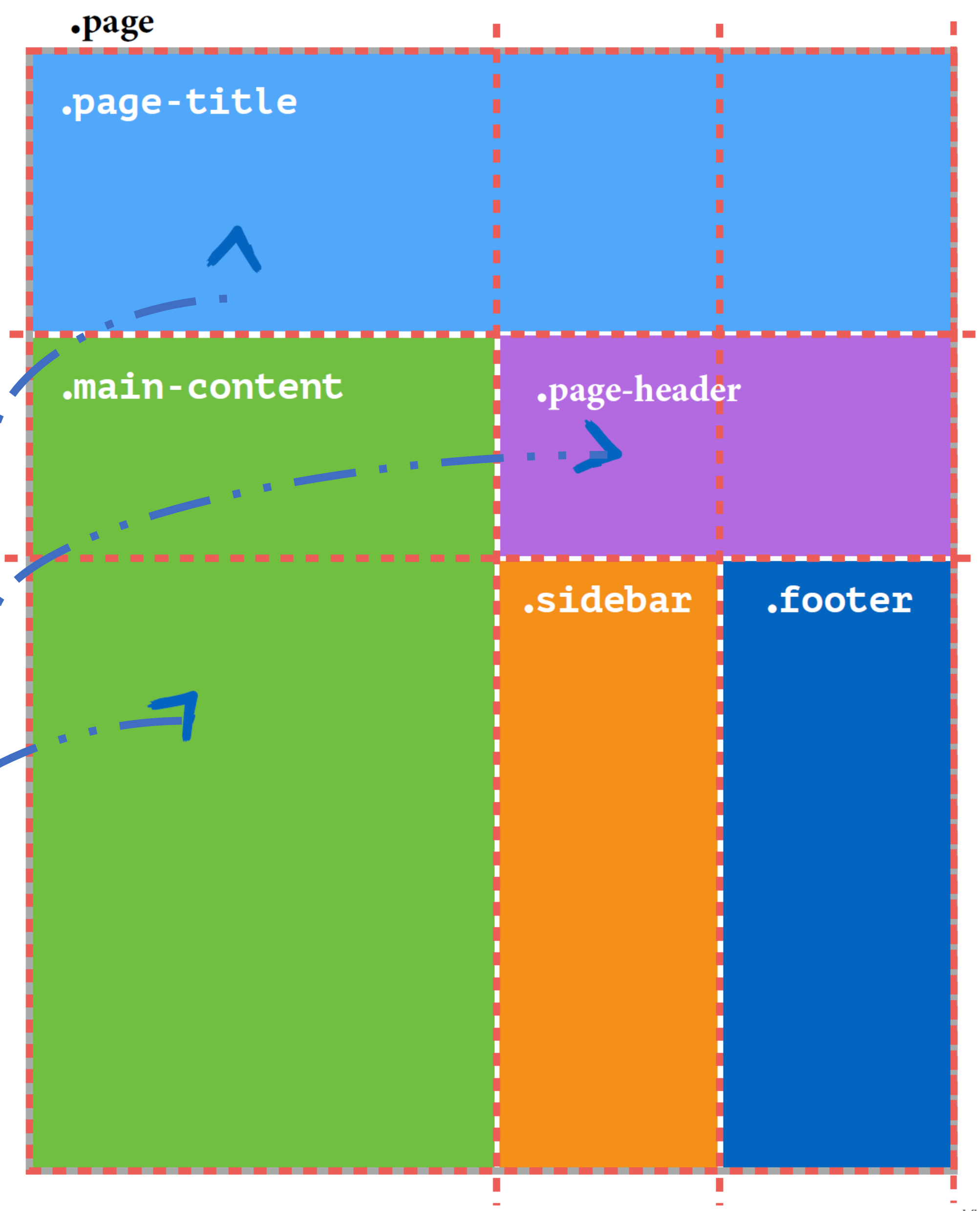


grid-template-areas

is used to **define named grid areas**

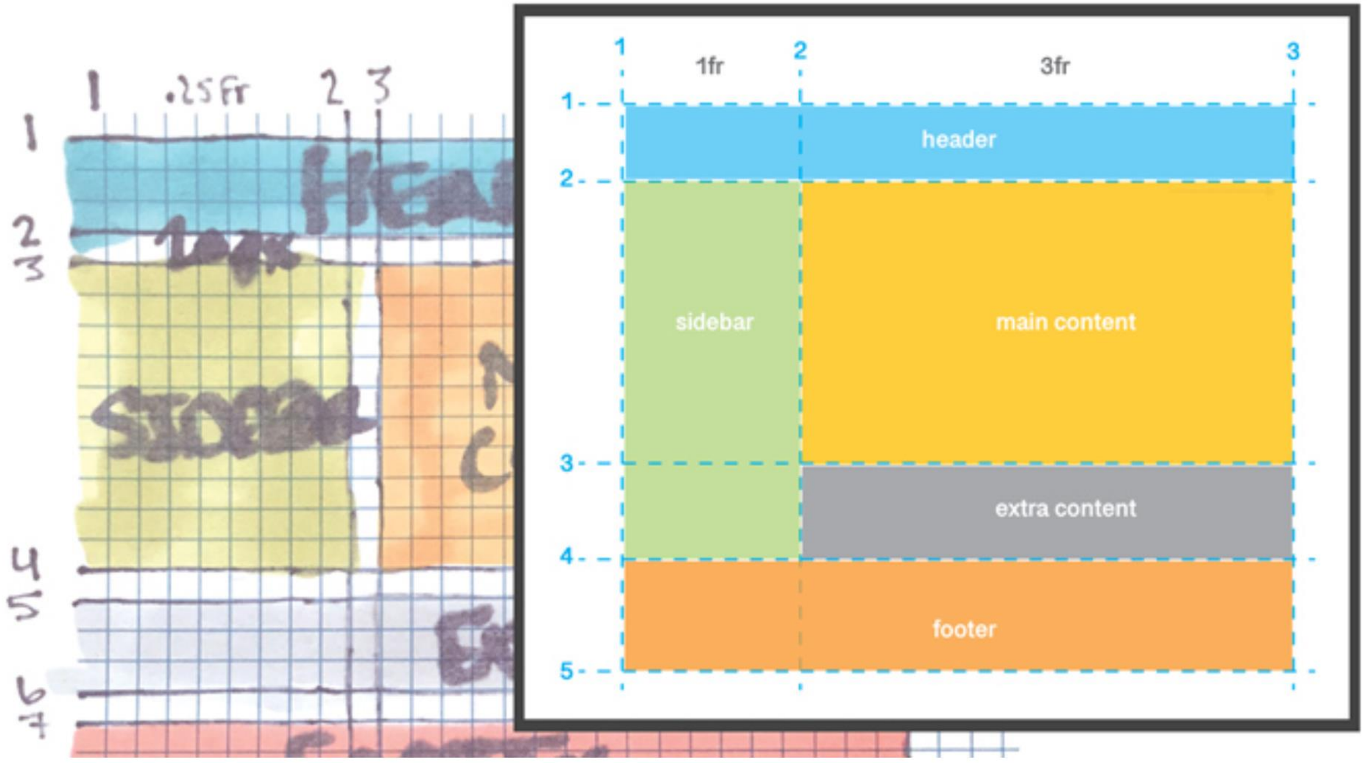
Placing items in the grid areas

```
.page {
    display: grid;
    grid-template-columns: 2fr 1fr 1fr;
    grid-template-rows: auto 1fr 3fr;
    grid-template-areas:
            "title title title"
            "main header header"
            "main sidebar footer";
}

/* Placing items in the grid areas: */
.page-title {
    grid-area: title;
}

.page-header {
    grid-area: header;
}


.main-content {
    grid-area: main;
}

/* etc etc */
```



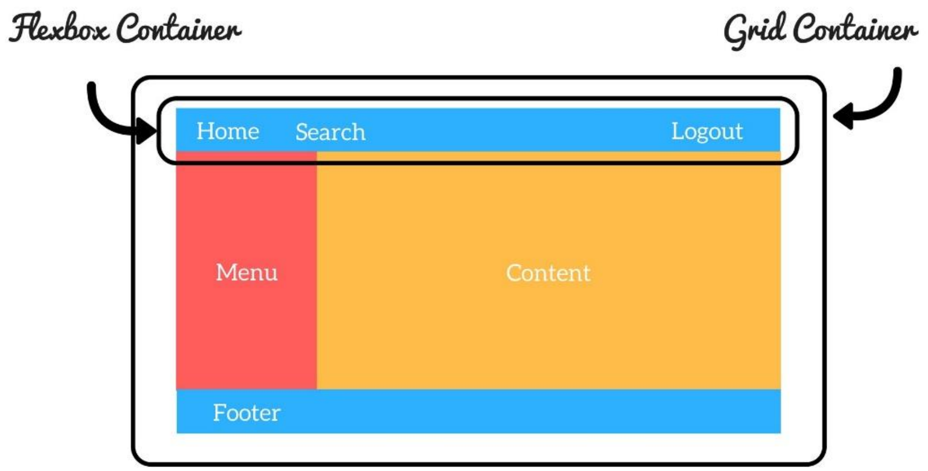
Grid areas

- Defining grid areas and using them to place elements is **best way** to design the page layout as it allows direct translation of the paper-based design to a CSS grid



Grid vs Flexbox

- Grid Layout is a **two-dimensional** system with columns and rows, unlike flexbox which is a **one-dimensional system** (either in a column or a row).
- In practice you combine these layout models. Often you can use a Flexbox container inside a Grid container
  - Grid is often used for the overall page layout of the homepage (i.e., **larger scale** layout) while the **flexbox is used for small-scale** one-dimensional layouts (e.g., menu or card layout)



Responsive Web Design (RWD)

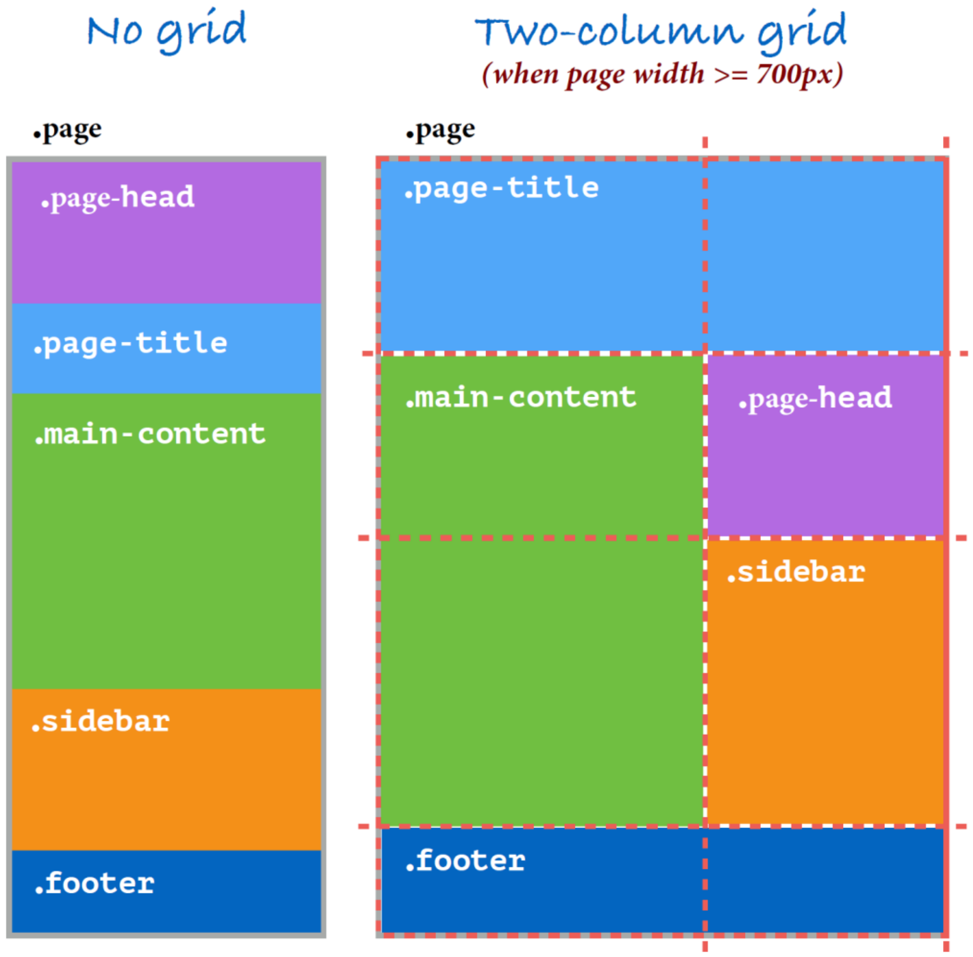
- RWD is an approach to **serve different layouts for different screen sizes**
  - **Optimize the viewing experience on range of devices**: mobile, desktop, tablet, TV…
  - Can be accomplished using CSS **media queries** and grid/flexbox
  - Mobile-first layouts work well on all screen widths

27

Responsive page layout using grid

```
@media screen and (min-width: 700px) {
    .page {
        display: grid;
        grid-template-columns: 2fr 1fr 1fr;
        grid-template-rows: auto 1fr 3fr;
        grid-template-areas: "title title"
                             "main header"
                             "main sidebar"
                             "footer footer";

    }
}
```

- Responsive page layout using media queries and grid
- Media queries allows applying styles based on the browser screen size



Responsive Grid

```
main {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(280px, 1fr));
}
```

Browser!

- I want you to **auto-create the grid columns** you decide how many you can fit using the auto-placement algorithm
- I want the columns to be minimum 280px and a maximum of **sharing the available space equality among the columns**

See posted example

Summary

- Use Grid any time you work with ***two-dimensional*** layouts to divide the page into several sections having different size and position
- Use Flexbox for ***one-dimensional*** layout that offers space allocation between items + the ability to alter its items' width/height (and order) to best fill the available space
- Use Media Queries and Grid layout for responsive design
- .. mastering CSS needs hands-on practice …

References

- CSS Tutorials http://www.w3schools.com/css/
- Cheat sheet https://htmlcheatsheet.com/css/
- CSS developer guide

https://developer.mozilla.org/en-US/docs/Web/Guide/CSS

- Selectors http://code.tutsplus.com/tutorials/the-30-css-selectors-you-must-memorize--net-16048
- CSS Grid
  - https://1linelayouts.glitch.me/
  - https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Grid_Layout
  - https://gridbyexample.com/learn/
  - https://css-tricks.com/snippets/css/complete-guide-grid/
  - https://mozilladevelopers.github.io/playground/css-grid/