# Table of Contents

1. CSS Syntax
2. Basic styles
3. Margin, Border and Padding
4. **Layout using Grid**
5. Layout using Flexbox
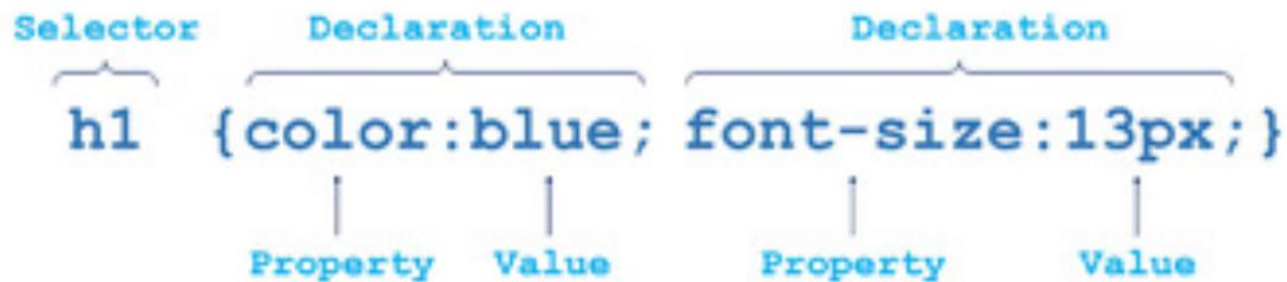6. Responsive Web Design (RWD)
7. Bootstrap

# CSS Syntax

# CSS – Cascading Style Sheets

- CSS is used to control the **style** and **layout** of web pages

- Allows **separating** web page content from its design and visual appearance

- Used in conjunction with HTML

  - **HTML** is used for describing the **content** of a web page, **CSS** is used for describing its **presentation**

# Style Sheets Syntax

- Stylesheets consist of **rules**. Each rule has **selectors** and declarations. A declaration specify a **property** and its **value**.

```
Selector      Declaration              Declaration

h1   {color:blue; font-size:13px;}

        Property  Value      Property      Value
```

- Selectors are separated by commas
- Declarations are separated by semicolons
- Properties and values are separated by colons

- Comment in CSS  **/* comment */**

# Ways to incorporate CSS in an HTML document

- **Inline** – style included as the attribute of an HTML tag:

```
<p style="color:sienna;margin-left:20px;">This is a paragraph.</p>
```

- **Internal** – CSS code is contained in the head section:

```
<head>
<style>
p {color:sienna; margin-left:20px;}
body {text-align:center;}
</style>
</head>
```

- **External** - separate .css file referenced in the HTML:

HTML source code:

```
<head>
<link rel="stylesheet" type="text/css"
href="main.css">
</head>
```

'main.css':

```
p {color:sienna;
margin-left:20px;}
body {text-align:center;}
```

√ Ensure **consistent look and feel**
√ Improve **reusability** and **maintainability**

# Selectors: used to <u>select</u> elements to style on an HTML page

- **Tag Selectors**
  - Apply page-wide

  e.g., `p { font-family: verdana; }` applies the style to all <p> tags

- **Class Selectors**
  - Defines a **named** style (prefix the name with dot (.) )
  - Can apply to any page element using the class attribute

  e.g., `.redBorder {border: 1px solid red}` defines a style named redBorder

  `<p class='redBorder'>Using the class attribute to apply the redBoder style to this paragrpah</p>`

- **ID Selectors**
  - Apply to one specific tag
  - Use hash (#) followed by the tag id to select the element to be styled
  - Good for linking to specific part of a page

  e.g., `#errorMsg { color: red; }` apply the style to the element with id **errorMsg**

# Combined Selectors

| element, element | div, p | Selects all <div> elements and all <p> elements |
|---|---|---|
| element element | div p | Selects all <p> elements inside <div> elements |
| element **>** element | div **>** p | Selects all <p> elements where the parent is a <div> element |

**e.g.,**

`li a {text-decoration: underline}`

  o This will match all <a> tags that are inside of <li>

https://www.w3schools.com/cssref/css_selectors.asp

# Attribute Selectors

| [attribute ^= value] | a[href ^= "https"] | Selects every <a> element whose href attribute value begins with "https" |
| --- | --- | --- |
| [attribute $= value] | a[href $= ".pdf"] | Selects every <a> element whose href attribute value ends with ".pdf" |
| [attribute *= value] | a[href *= "qu"] | Selects every <a> element whose href attribute value contains "qu" |

https://www.w3schools.com/cssref/css_selectors.asp

# Pseudo-classes

- Pseudo-classes to define **element state**

  o `:link`, `:hover`, `:visited`, `:active`

`a:hover { color: red; }` -> Style link on mouse over

`:link` pseudo-class selects anchors tags that were not visited

- Pseudo-elements to **insert content** around the selected element

`p::before { content: "«"; }`

Insert « before the content of each <p> element

`p::after { content: "»"; }`

Insert » after the content of each <p> element

# Structural Pseudo-classes

| | | |
|---|---|---|
| :first-child | tr:**first-child** | First row of an HTML table |
| :last-child | tr:**last-child** | Last row of an HTML table |
| :nth-child(*n*) | tr:**nth-child(2)** | Second row of an HTML table |
| :nth-last-child(*n*) | tr:**nth-last-child(2)** | Second row of an HTML table, counting from the last row |
| nth-child(odd) | tr:**nth-child(odd)** | Every odd row of an HTML table |
| nth-child(even) | tr:**nth-child(even)** | Every even row of an HTML table |

https://www.w3schools.com/cssref/trysel.asp

# Selectors Summary

- A style consists of a selector, followed by property/value pairs

- Selectors:
  - Tag Selectors
  - Class Selectors
  - ID Selectors
  - Combined Selectors
  - Attribute selectors
  - Pseudo-elements
  - Structural pseudo-classes

**Examples**

- 2.selectors
  - 1.Attribute Selectors.html
  - 2.Structural Selectors (empty).html
  - 3.Structural Selectors (first-of-type).html
  - 4.Structural Selectors (nth-child).html
  - 5.Pseudo-classes.html

# Basic styles

# Text-related CSS Properties

- `color` – specifies the color of the text

- `font-size` – size of font: `xx-small`, `x-small`, `small`, `medium`, `large`, `x-large`, `xx-large`… or numeric value

- `font-family` – comma separated font names

  - Example: `verdana`, `sans-serif`, …

  - The browser loads the first one that is available

- `font-weight` can be `normal`, `bold`….

# Text-related CSS Properties (2)

- `font-style` – styles the font
  - Values: `normal`, `italic`, `oblique`

- `text-decoration` – decorates the text
  - Values: `none`, `underline`, `line-trough...`

- `text-align` – defines the alignment
  - Values: `left`, `right`, `center`, `justify`

# Styles for Lists

- List properties are used to define the look and feel of the list items
  - Values for <ul>: circle, square,…
  - Values for <ol>: upper-roman, lower-alpha
  - Values for both: none

```
ul
{
    list-style-type:none;
}
```
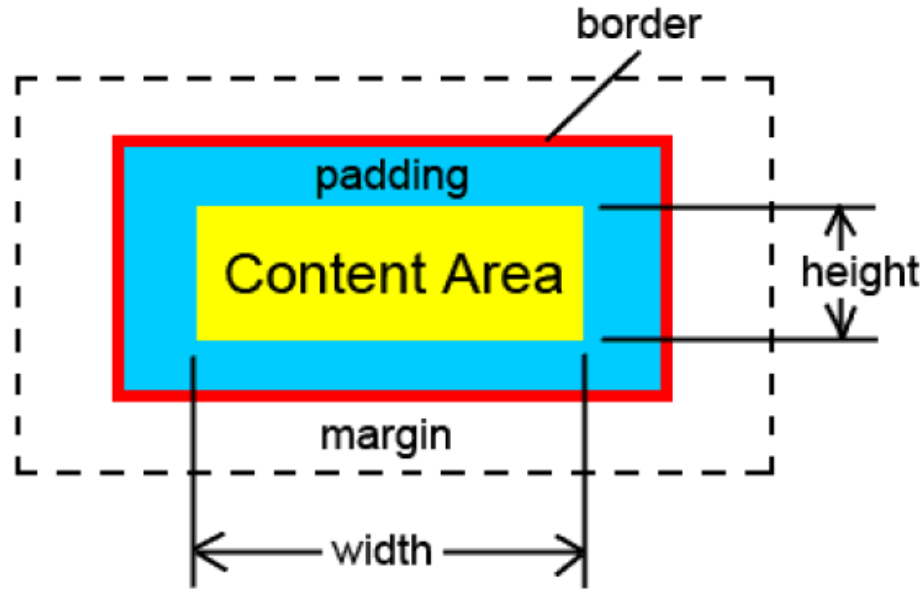
Item 1
Item 2
Item 3
Item 4
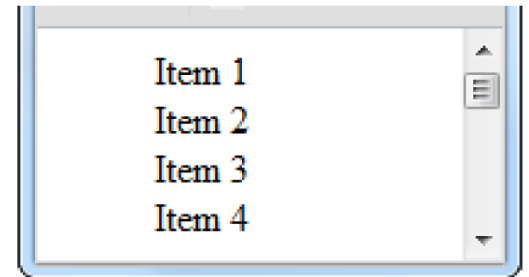
# **Practice …**

- Use the W3Schools try-it-yourself editor to try styling each of these properties

- Background
  [http://www.w3schools.com/css/css_background.asp](http://www.w3schools.com/css/css_background.asp)

- Text
  [http://www.w3schools.com/css/css_text.asp](http://www.w3schools.com/css/css_text.asp)

- Fonts
  [http://www.w3schools.com/css/css_font.asp](http://www.w3schools.com/css/css_font.asp)

- Lists
  [https://www.w3schools.com/css/css_list.asp](https://www.w3schools.com/css/css_list.asp)

# Margin, Border and Padding

# Box Model



- Each tag is a box and its properties can be styled:
    - **Margin** –the space that separates the boxes
    - **Border** –the line around each edge of the box
    - **Padding** –the space between the border and the contents

# Margin and Padding

- Margin and padding define the spacing around the element
  - Numerical value, e.g. 10px
  - Can be defined for each of the four sides separately: **margin-top**, **padding-left**, …  or using short rules:
- margin: 5px;
  - Sets all four sides to have margin of 5 px;
- margin: 10px 20px;
  - top and bottom to 10px, left and right to 20px;
- margin: 1px 3px 5px 7px;
  - top, right, bottom, left (clockwise from top)
- Same for padding

# Borders

- Border style:

```
border-width:1px;
border-color:red;
border-style:solid;
```

  - o **border-width**: thin, medium, thick or numerical value
  - o **border-color**: color alias or RGB value
  - o **border-style**: none, dotted, dashed, solid, double, …
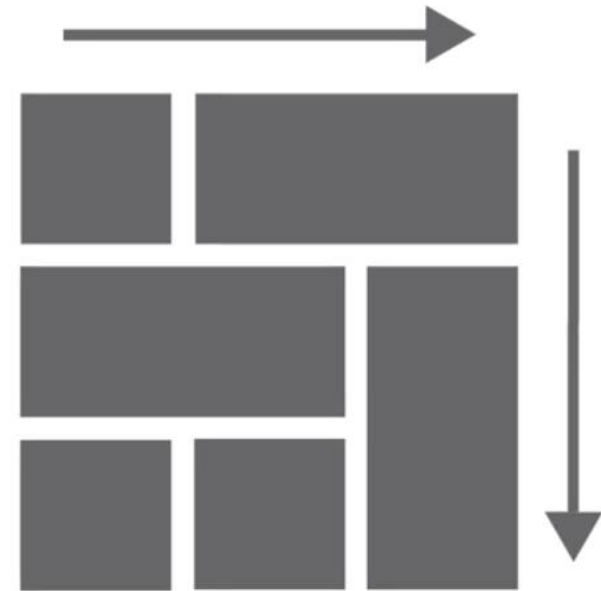
- Shorthand rule for setting border properties:

```
border: 1px solid red;
```

- Can specify different borders for the sides using:
  border-top, border-left, border-right, border-bottom

# 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

**CSS Grid**

**TWO DIMENSIONS**

**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;
}
```
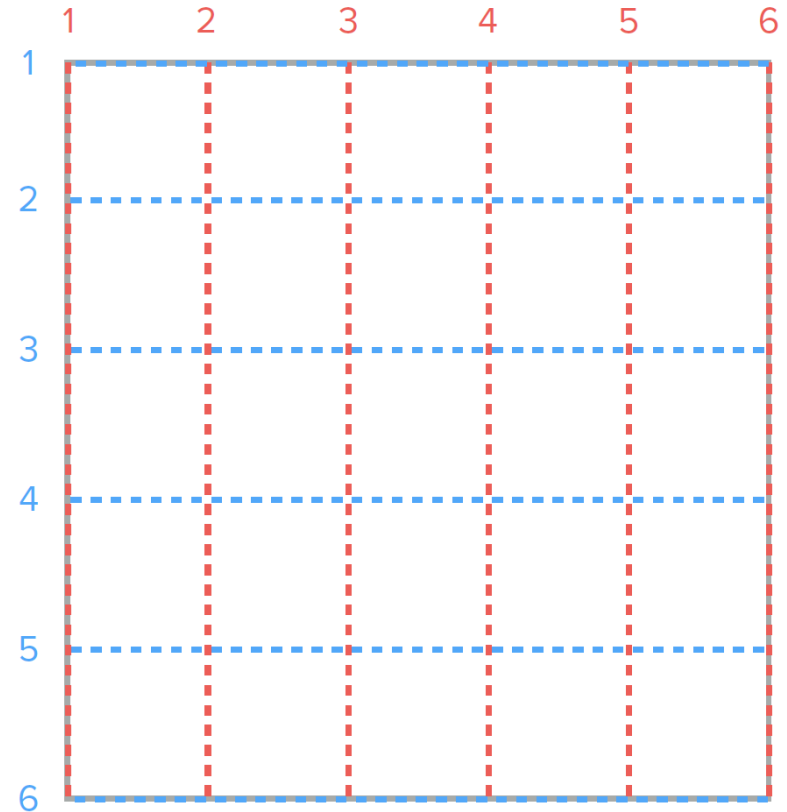
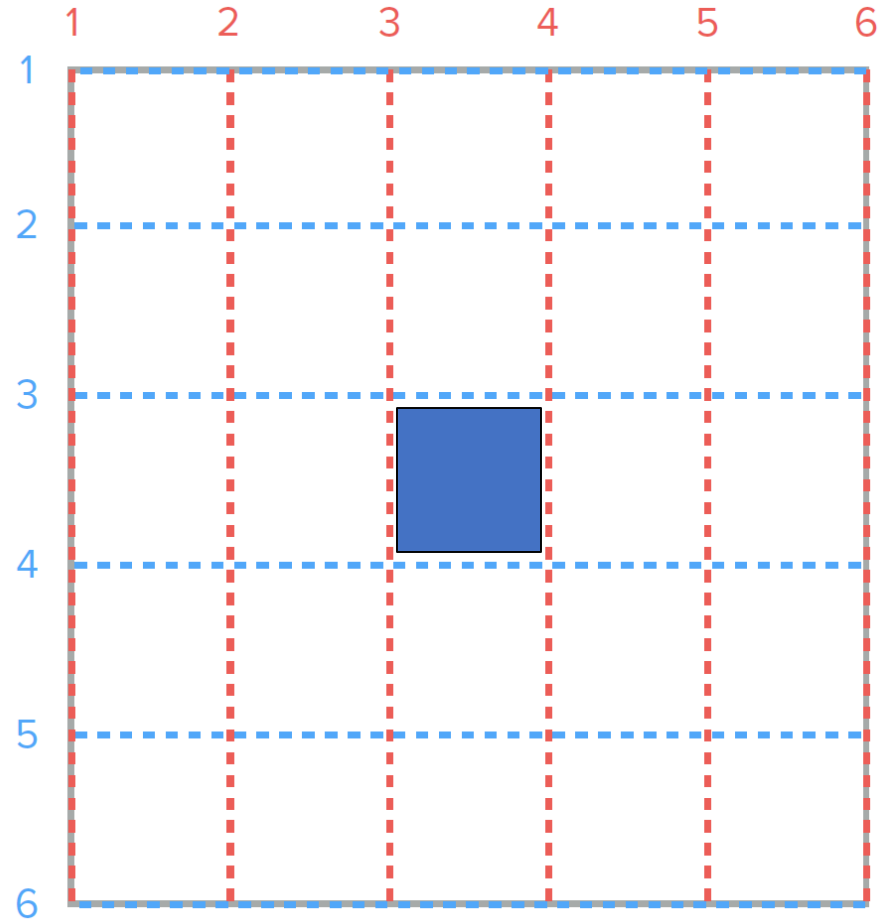**This creates a grid container**

```
.page
<div class="page">
    <header class="page-header">
    </header>

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

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

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

# Grid **item**

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

```
<div class="page">
    <header class="page-header">
    </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*
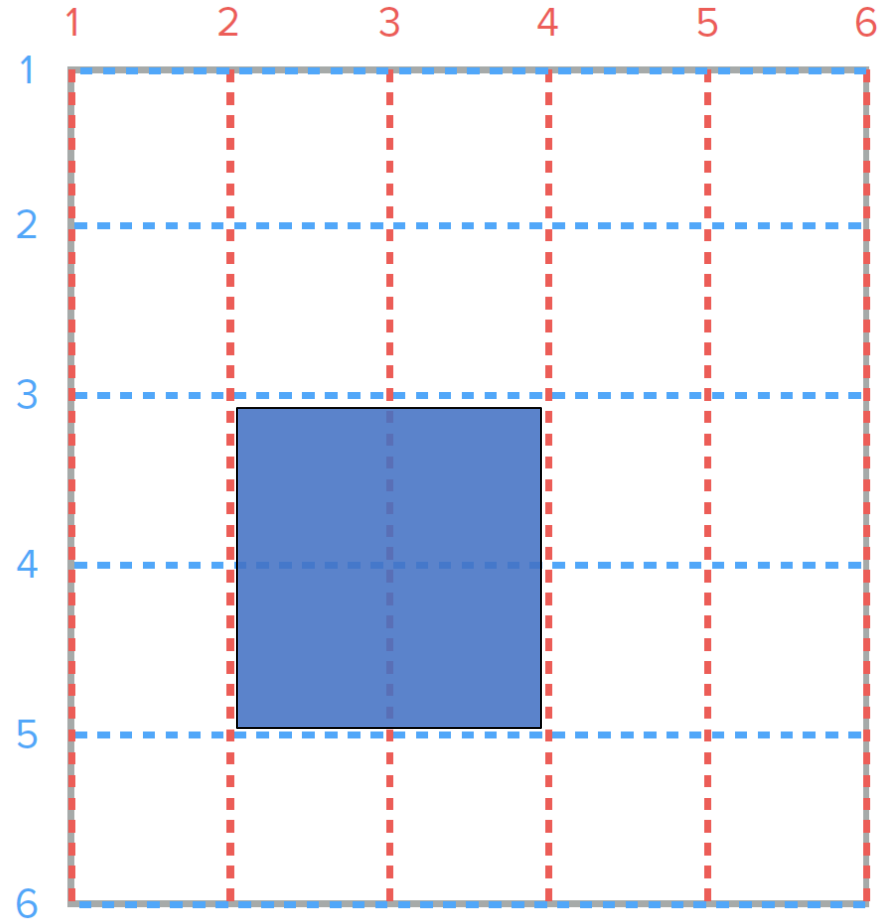
# Grid *cell*

- 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

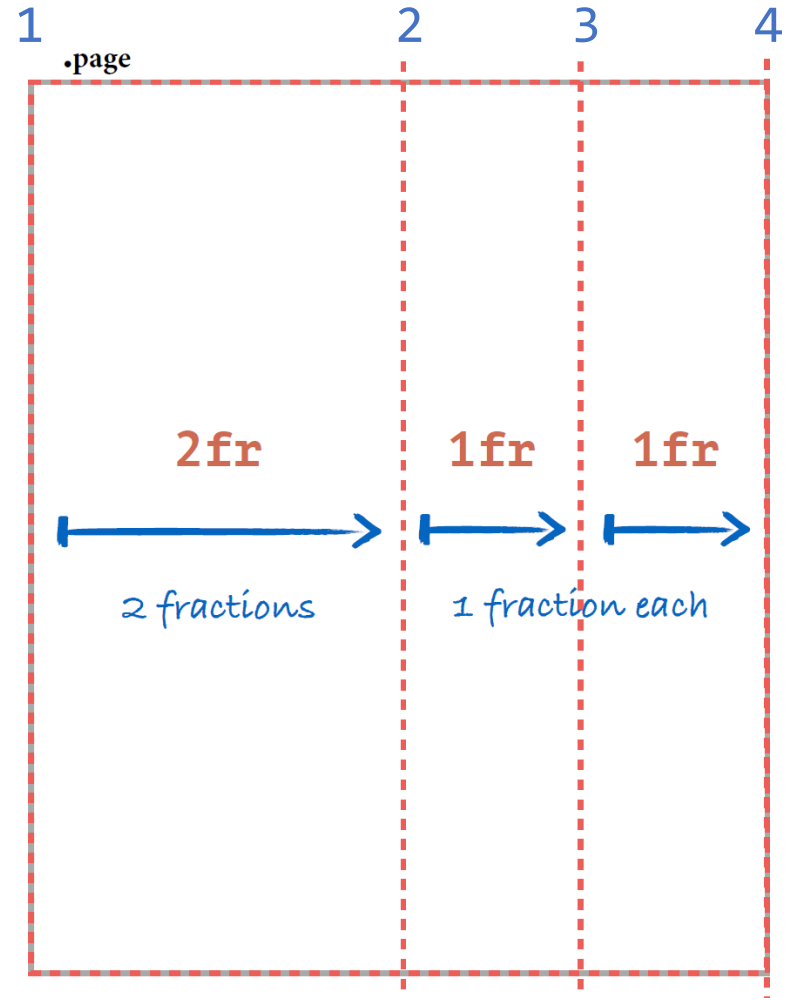- E.g., blue area between row lines 3 and 5 and column lines 2 and 4

# 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.
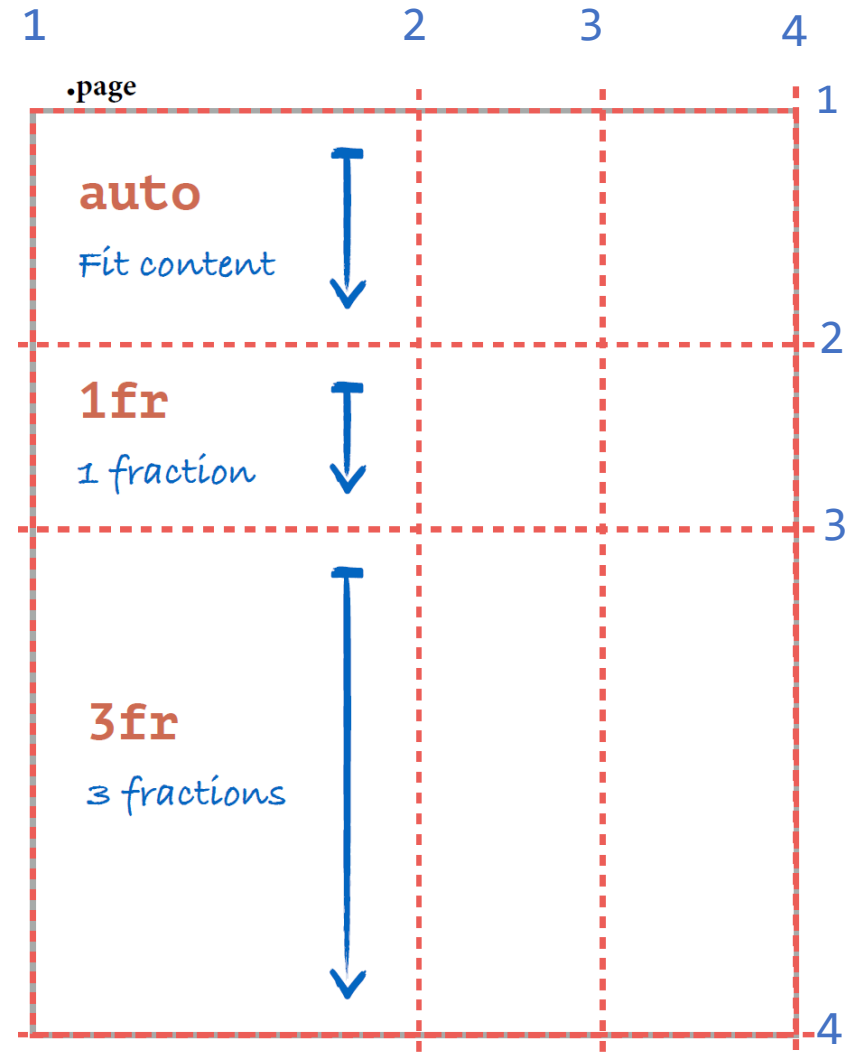
# Grid rows

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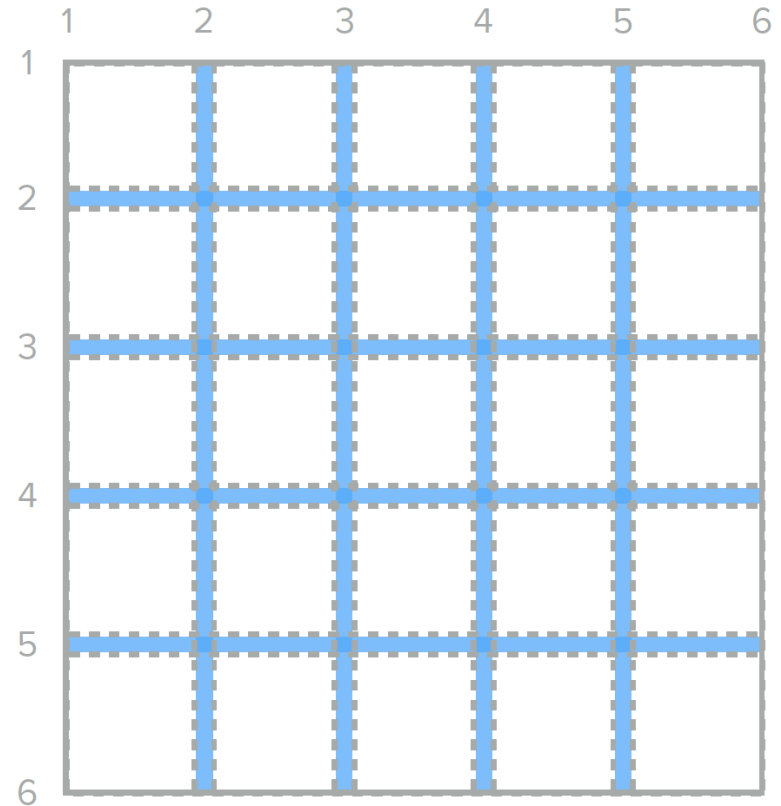`auto 1fr 3fr;`

———————

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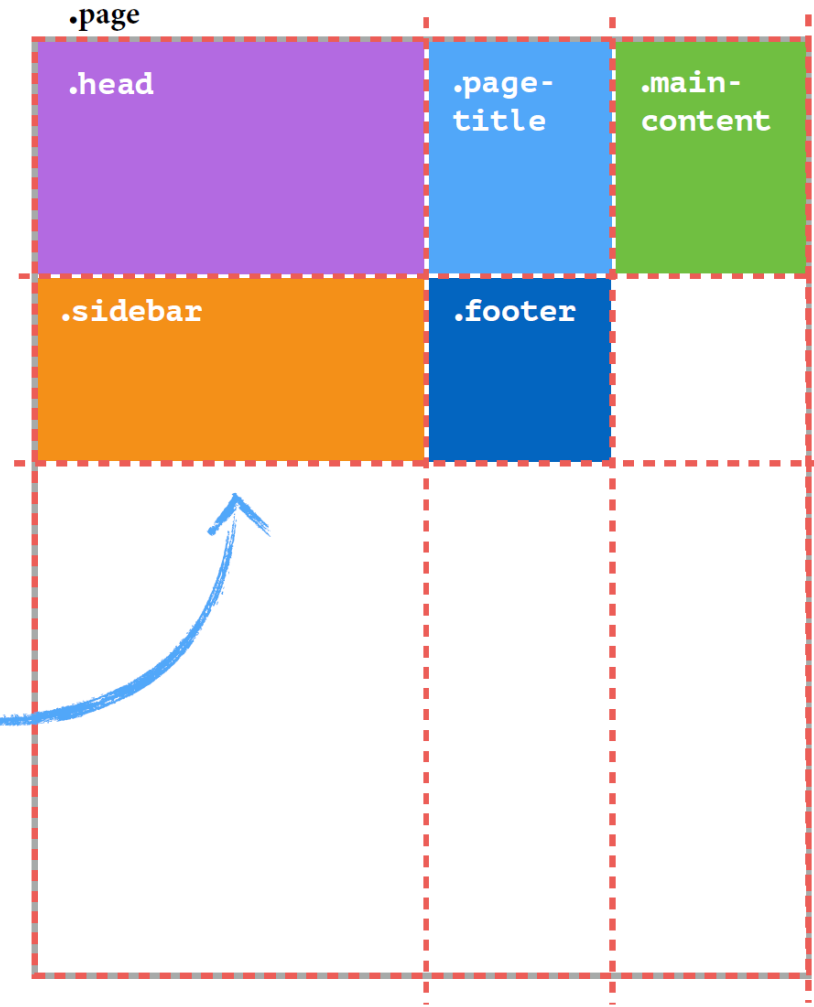
# Grid **gap**

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

- Commonly called *gutters*

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

.page

| .head | .page-title | .main-content |
|---|---|---|
| .sidebar | .footer | |

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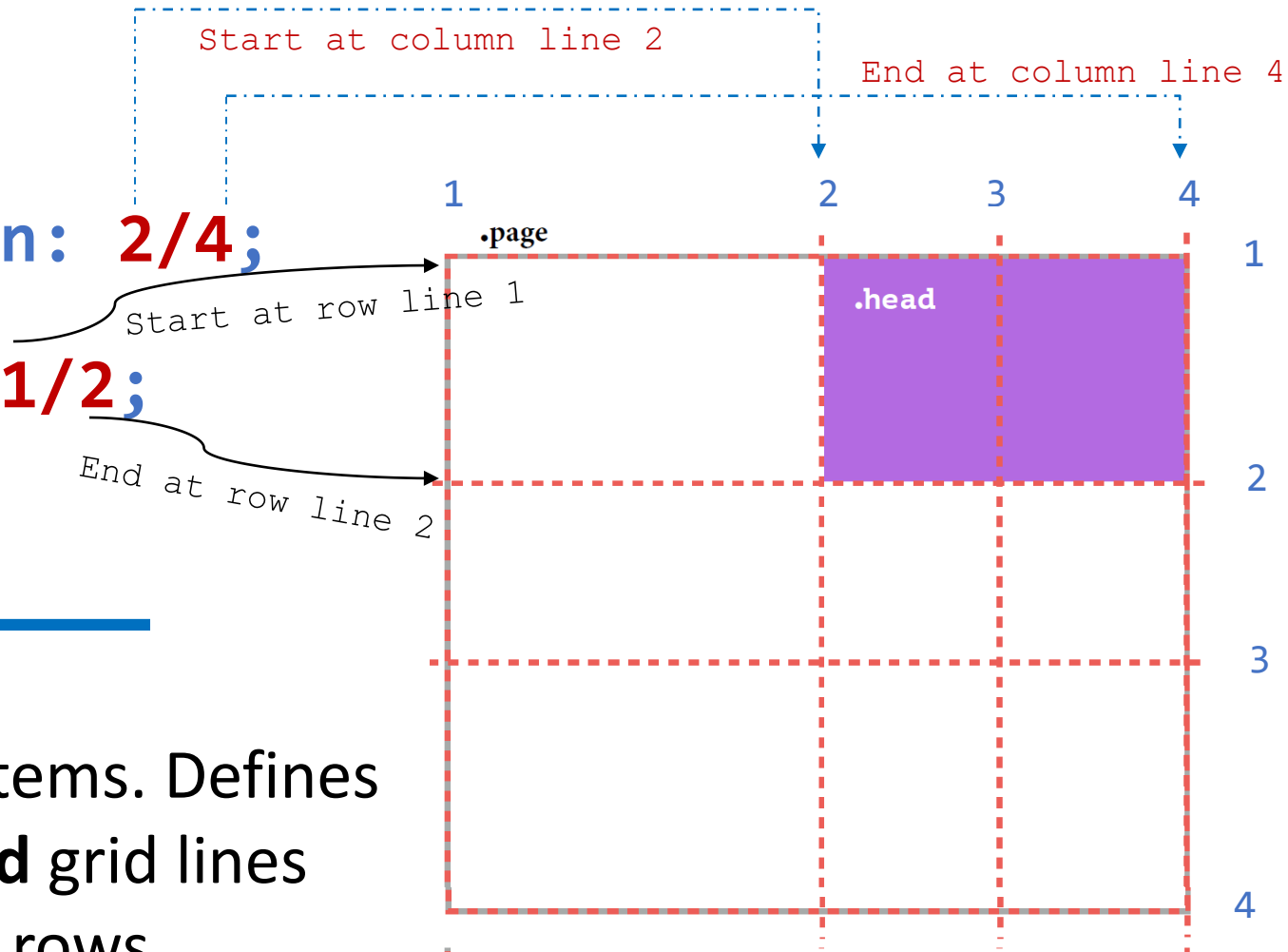
# Items placement in Grid

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

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

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

Start at column line 2

End at column line 4

Start at row line 1

End at row line 2

.page

.head

1    2    3    4

1

2

3

4

1    2    3    4

# Example

```css
.page {
    display: grid;
    grid-template-columns: 2fr 1fr 1fr;
    grid-template-rows: auto 1fr 3fr;
}
.header {
    grid-column: 2/4;
    grid-row: 2/3;
}
.page-title {
    grid-column: 1/4;
    grid-row: 1/2;
}
.main-content {
    grid-column: 1/2;
    grid-row: 2/4;
}
/* etc etc */
```
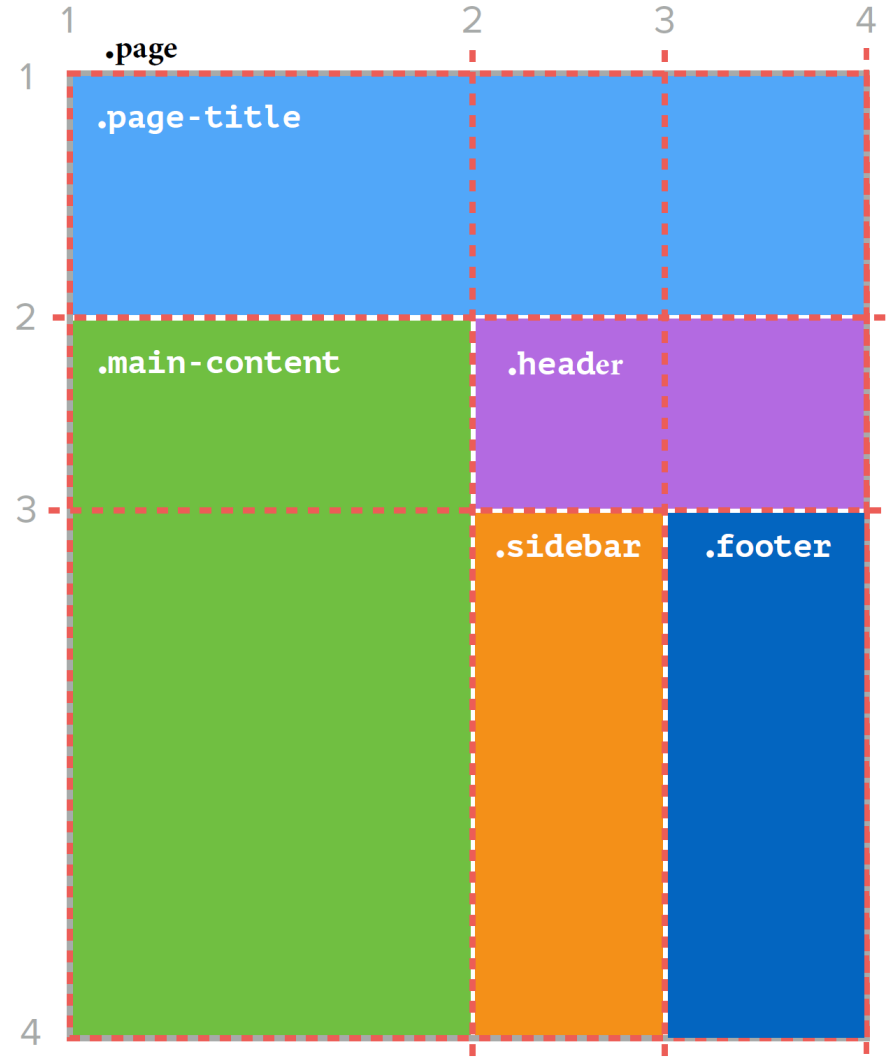
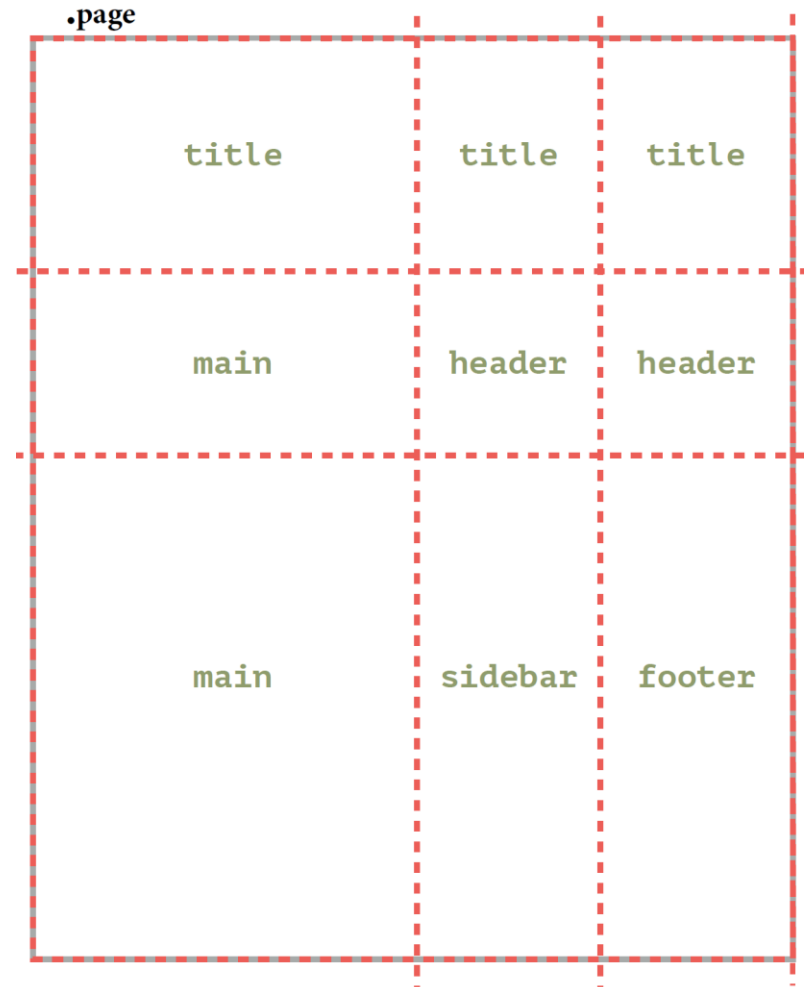Ok, but remembering what lines to target seems tricky… especially when the site is responsive

# 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

```css
.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 */
```
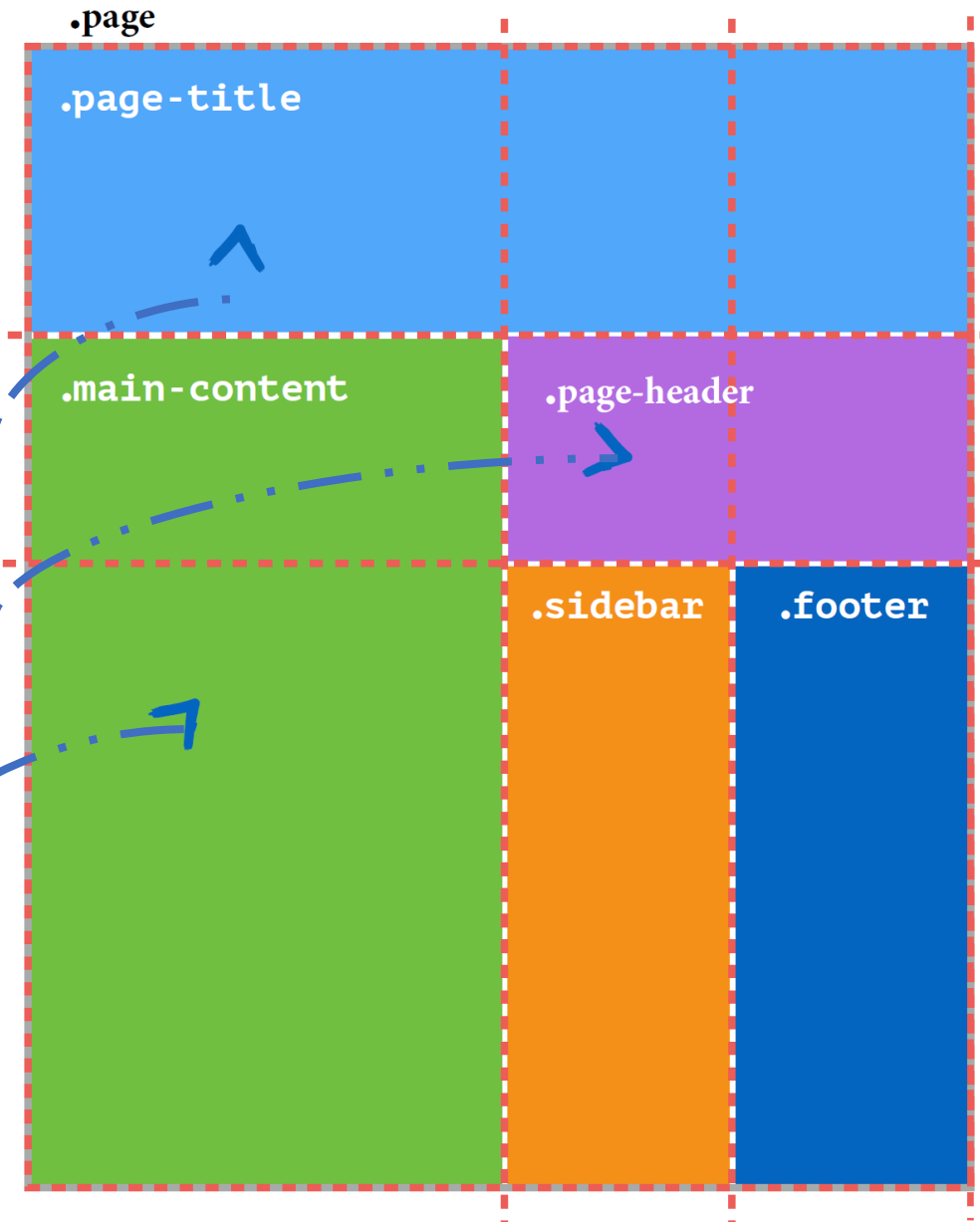
.page

.page-title

.main-content

.page-header

.sidebar

.footer

# Responsive Grid

```css
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 200px and a maximum of **sharing the available space equality among the columns**

See posted example

# Layout using Flexbox

# Flexbox

- The Flexbox provide a more efficient way to define **one-dimensional layout** and distribute space among items in a container while accommodating different screen sizes

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

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

**Container**  **Items**
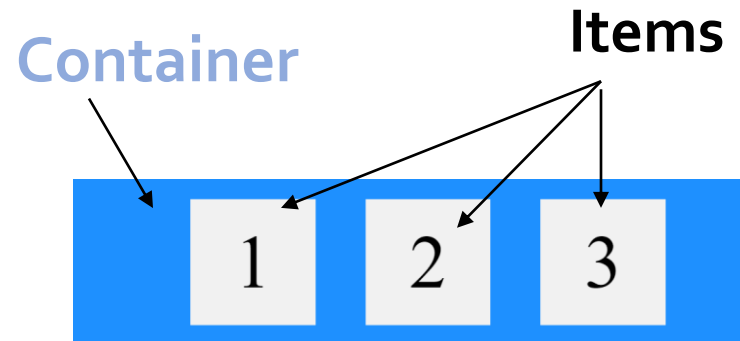


https://www.w3schools.com/css/css3_flexbox.asp

# Key Properties

```
.container{
    display: flex;
    flex-wrap: wrap;
    justify-content: space-around;
}
```
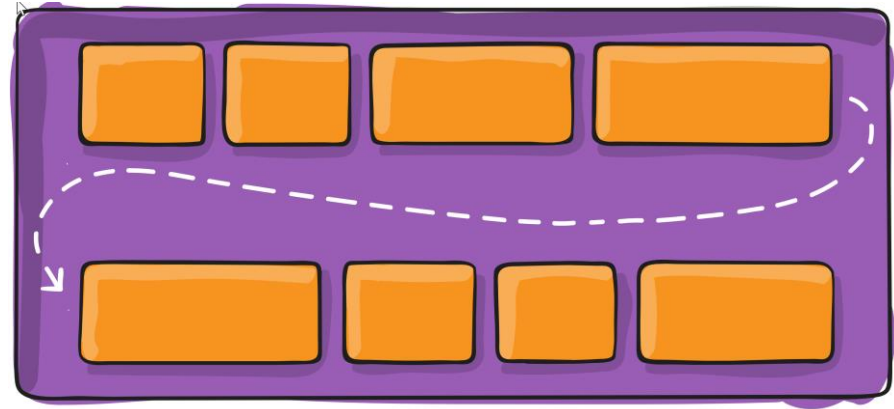


**justify-content**



More info @ https://css-tricks.com/snippets/css/a-guide-to-flexbox/

# CSS Grid vs Flexbox

- CSS 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).
  - If you only need to define a layout as a row **or** a column, then you probably need flexbox. If you want to define a grid and fit content into it in two dimensions — you need the grid.

- In practice you combine these layout models. Often you can use a Flexbox container inside a Grid container
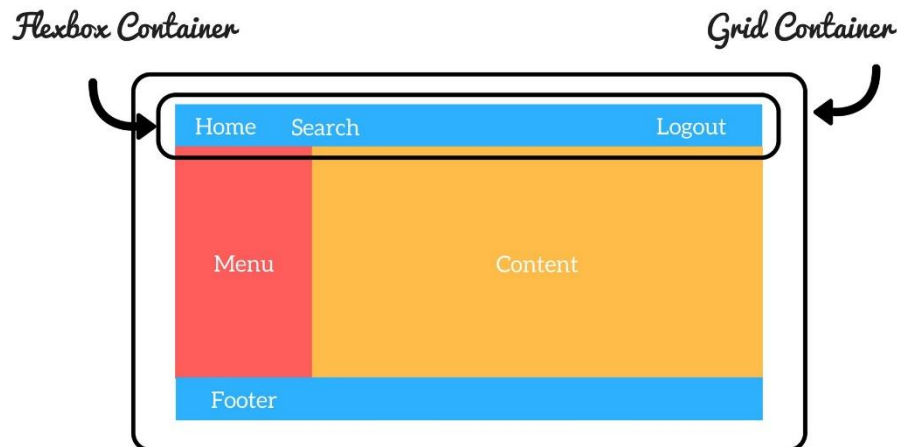
# Responsive Web Design (RWD)



- 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.

42

# Responsive page layout using grid

```css
@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
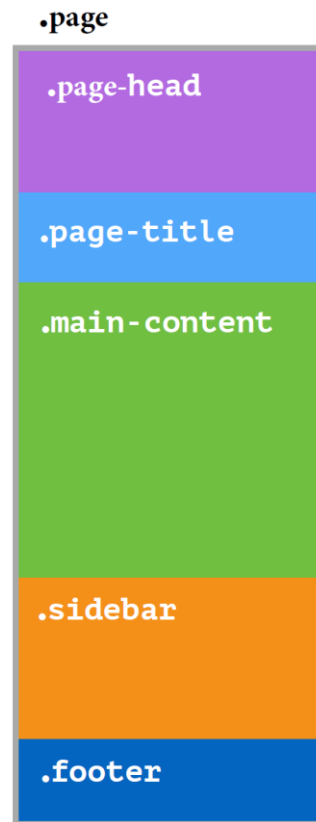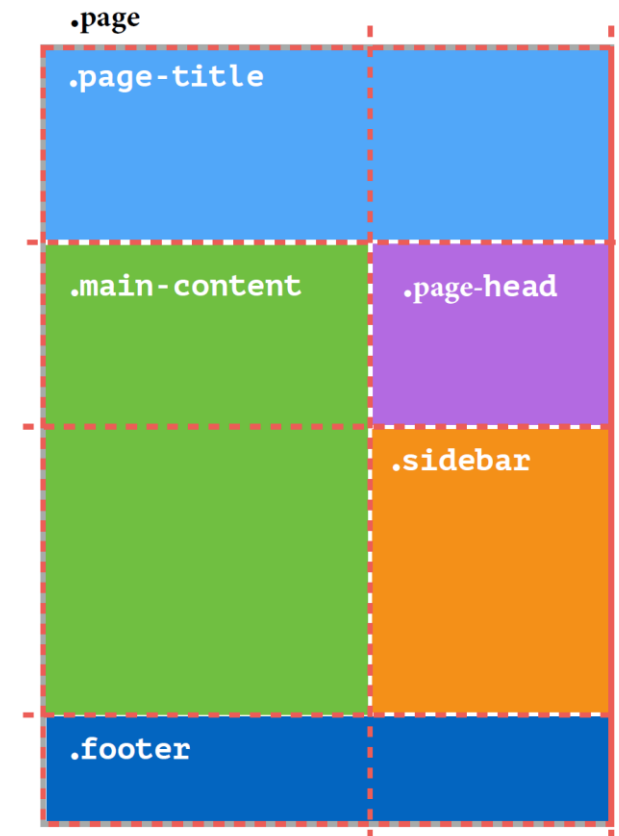
# Bootstrap

**CSS framework with built-in CSS classes**

**Do <mark>NOT</mark> use Bootstrap Grid**

# Bootstrap Navigation Bar

- Bootstrap navigation bar can extend or collapse, depending on the screen size.

Link 1   Link 2   Link 3

```html
<!-- A grey horizontal navbar that becomes vertical on extra small screens -->
<nav class="navbar navbar-expand-sm bg-light">
    <ul class="navbar-nav">
        <li class="nav-item">
            <a class="nav-link" href="#">Link 1</a>
        </li>
        <li class="nav-item">
            <a class="nav-link" href="#">Link 2</a>
        </li>
        <li class="nav-item">
            <a class="nav-link" href="#">Link 3</a>
        </li>
    </ul>
</nav>
```

https://www.w3schools.com/bootstrap4/bootstrap_navbar.asp

# Bootstrap Text Styling

- **Typography – text colors**
- .text-muted – grey
- .text-primary – light blue
- .text-success – green
- .text-info – dark blue
- .text-warning – yellow
- .text-danger – red

**Typography - alignment classes**

.text-left

.text-center

.text-right

https://www.w3schools.com/bootstrap4/bootstrap_typography.asp

# Bootstrap Table Styling

- Tables - on <table> elements
  - o .table – default table
  - o .table-striped – every second row is colored
  - o .table-bordered – adds border to a table
  - o .table-hover – adds hover element to a table
  - o .table-condensed – makes the table compact
- Tables – on <tr> and <td> elements
  - o .active, .success, .warning, .danger

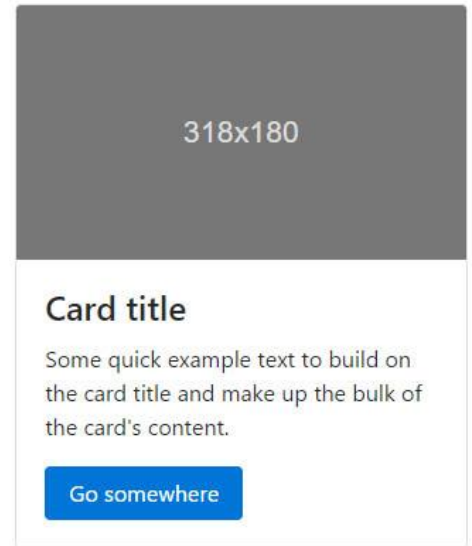https://www.w3schools.com/bootstrap4/bootstrap_tables.asp

# Bootstrap Form Styling

- Forms
  - Add .form to the <form> element
  - Add .form-control to all form elements
  - Group them with .form-group
- Buttons
  - .btn for buttons
  - .btn-primary, .btn-danger, … for colors
  - .btn-lg, btn-sm, btn-xs for sizes

https://www.w3schools.com/bootstrap4/bootstrap_forms.asp

# Cards

- **.card** to create a card

- Use **.card-title** for titles

- Can add image heading

```html
<div class="container">
    <div class="card" style="width:300px">
        <img class="card-img-top" src="images/tamim-al-majid.png">
        <div class="card-body">
            <h4 class="card-title">His Highness Sheikh Tamim bin Hamad Al Thani</h4>
            <p class="card-text">Emir of the State of Qatar</p>
            <a href="...." class="btn btn-primary">See Profile</a>
        </div>
    </div>
    <br>
</div>
```

https://www.w3schools.com/bootstrap4/bootstrap_cards.asp

49

# **Bootstrap Image Styling**

- Images

  o .img-rounded – for image with round edges

  o .img-circle – for image as circle

  o .img-fluid – auto resize to screen size


- Helper classes

  o Quick float - .float-left, .float-right


https://www.w3schools.com/bootstrap4/bootstrap_images.asp

# 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

# References

- CSS Tutorials [http://www.w3schools.com/css/](http://www.w3schools.com/css/)

- Cheat sheet [https://htmlcheatsheet.com/css/](https://htmlcheatsheet.com/css/)

- CSS developer guide

[https://developer.mozilla.org/en-US/docs/Web/Guide/CSS](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](http://code.tutsplus.com/tutorials/the-30-css-selectors-you-must-memorize--net-16048)

- CSS Grid

  o [https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Grid_Layout](https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Grid_Layout)

  o [https://gridbyexample.com/learn/](https://gridbyexample.com/learn/)

  o [https://css-tricks.com/snippets/css/complete-guide-grid/](https://css-tricks.com/snippets/css/complete-guide-grid/)

  o [https://mozilladevelopers.github.io/playground/css-grid/](https://mozilladevelopers.github.io/playground/css-grid/)