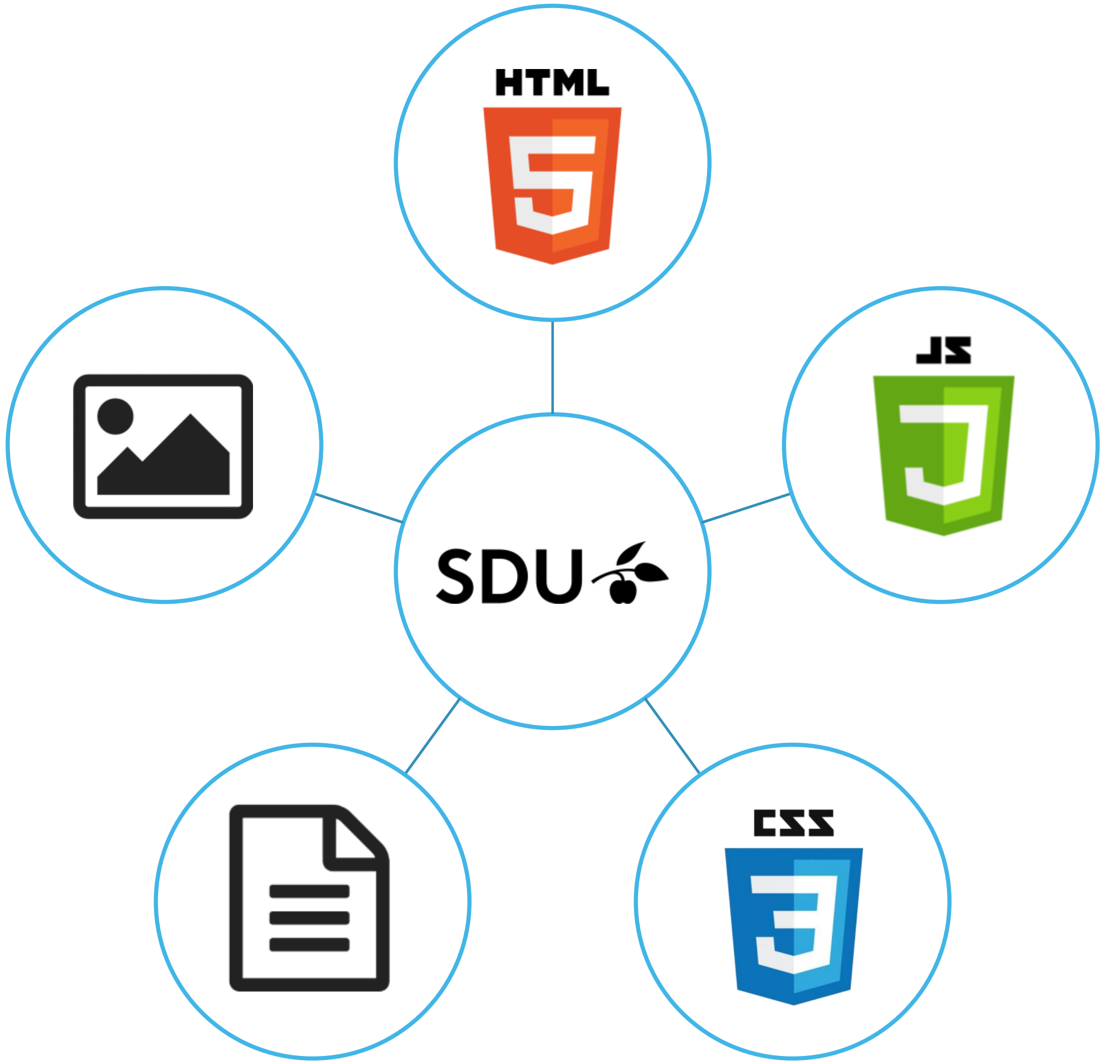
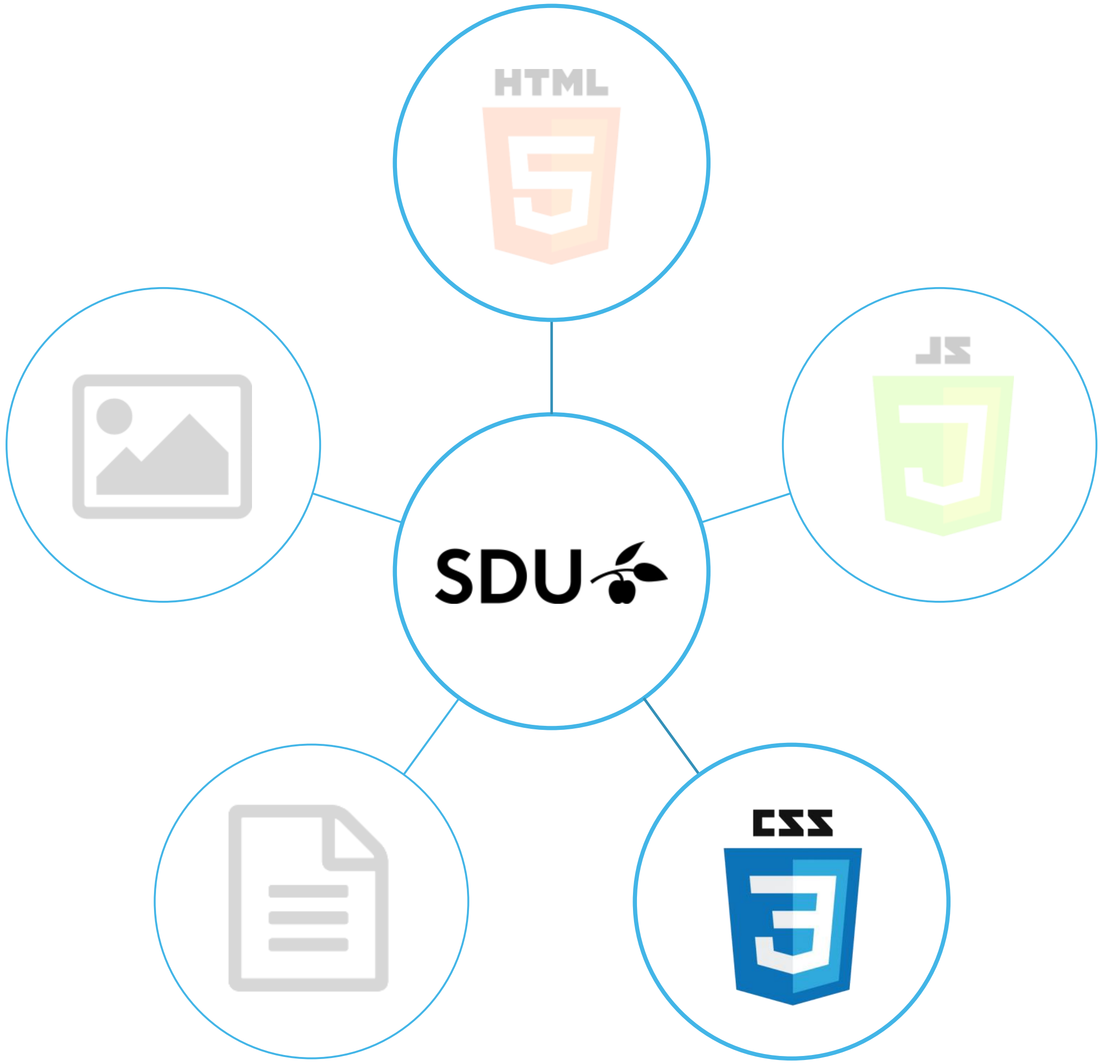
Introduction – CSS

27. februar 2018

What is a website?



What is a website?



HTML – HyperText Markup Language

Defines the structure of a webpage

CSS – Cascading Style Sheet

Defines the visual presentation of HTML elements

JS – Javascript

Handles user interaction and dynamic content

Cascading Style Sheets

```
12 text-center wow fadeInDown"
elay="200ms">

ata-to="565">565</span>

fa-3x"></i>

12 text-center wow fadeInDown"
elay="400ms">

ata-to="95">95</span>

3x"></i>

12 text-center wow fadeInDown"
elay="600ms">
rgin-bottom">

ata-to="2500">2500</span>
```

```
@media (m
    .navbar-inverse .n
        padding: 30px
    }
    .navbar-inverse.sm
        padding: 20px
    }
    .navbar-inverse .n
        padding-right:
    }
    #slitSlider .carou
        font-size: 50p
    }
#nav-dots {
    display: block;
}
#nav-arrows {
    display: none;
}
.icon-box {
  margin: 0 auto;
  width: 60%;
}
.about-content {
  margin: 0 auto 40px;
  padding: 0;
  width: 60%;
}
.why-item {
  margin: 0 auto 35px;
  text-align: center;
  width: 50%;
}
```

The Stylesheet

A stylesheet is a set of rules defining how an HTML element will be presented in a browser.

These rules are targeted to specific elements in an HTML document

The Cascade

The *cascade* part of CSS is a set of rules for resolving conflicts.

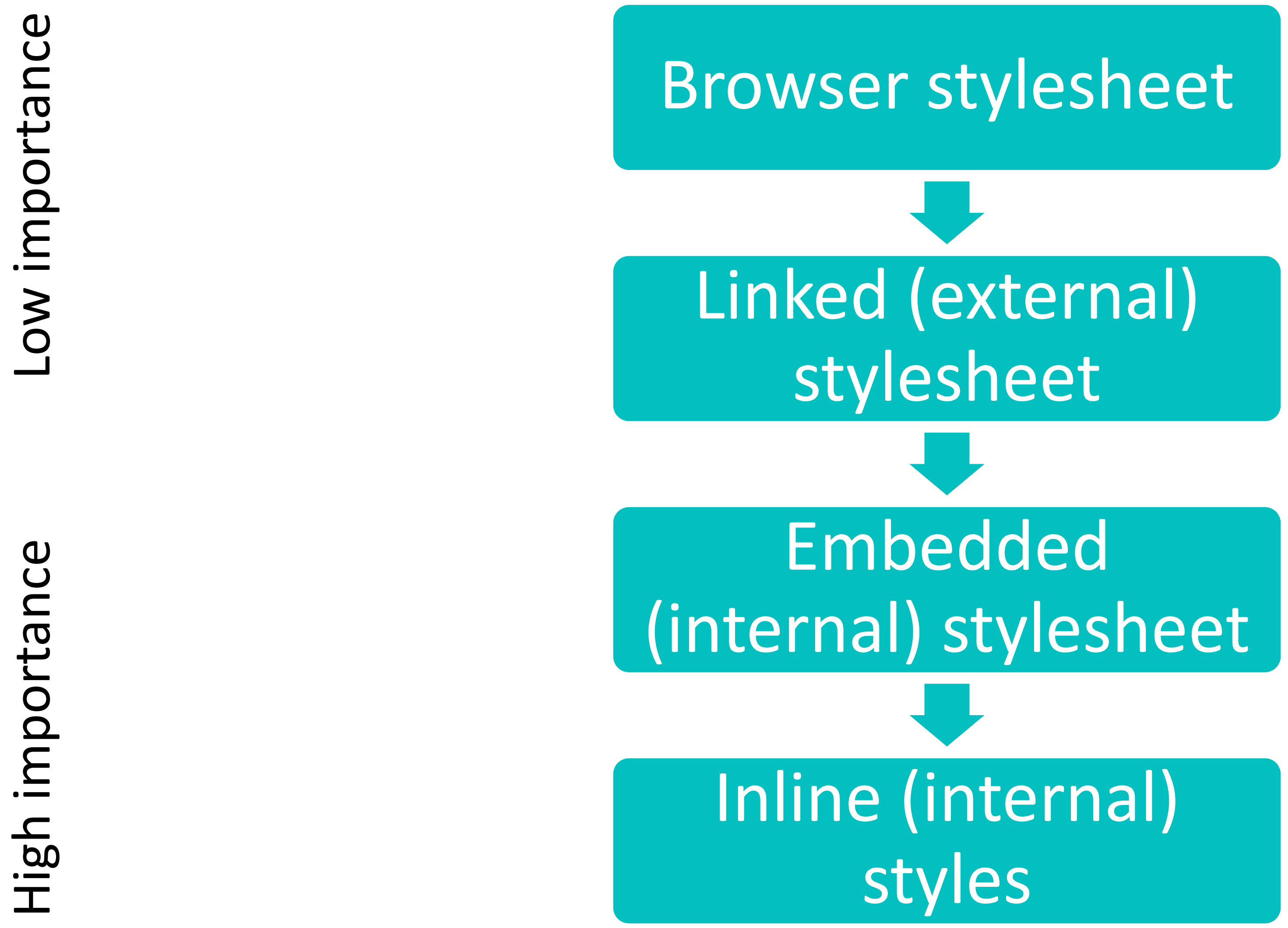
Conflicts arise when multiple CSS rules are defined for a given element.

Example:

If there are two rules defining the color of your <h1> elements,

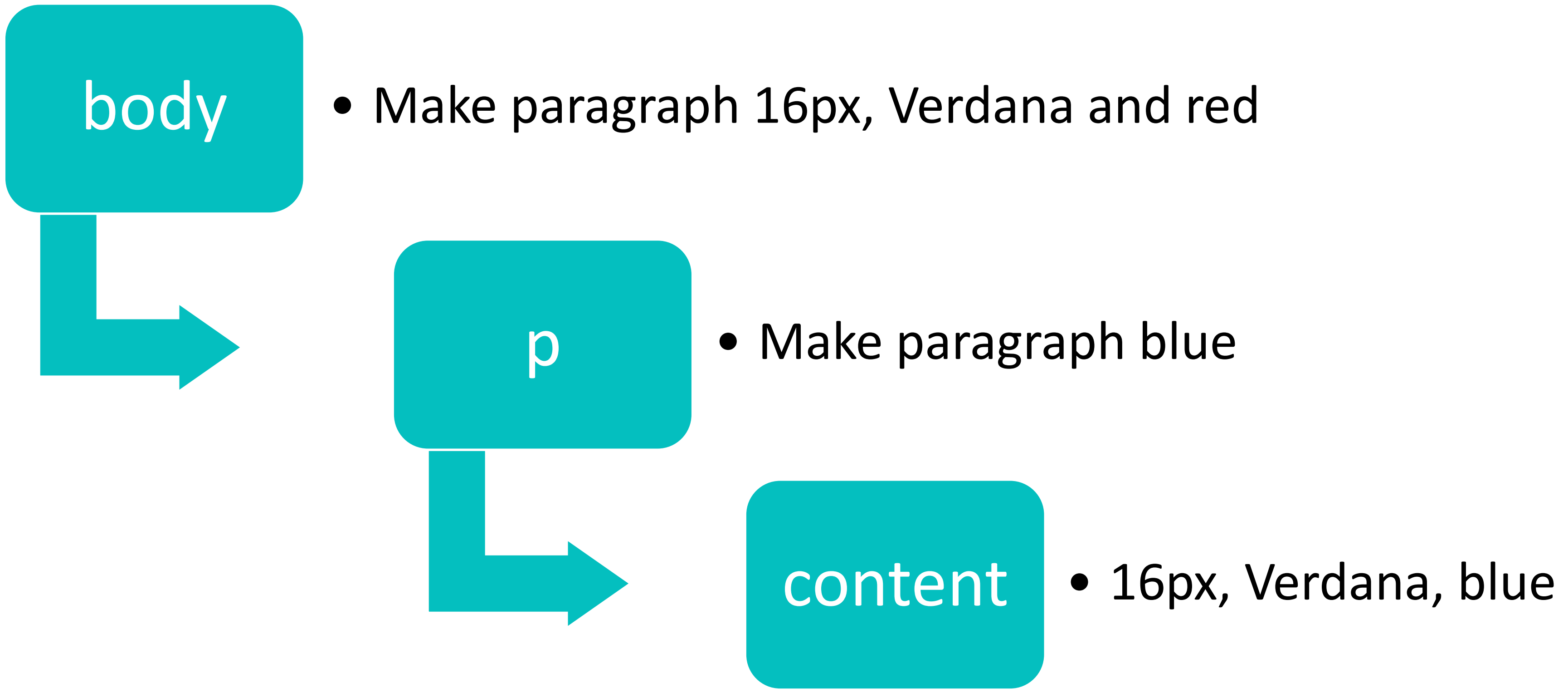
the rule that comes last in the cascade will trump all other rules.

Order of importance



Inheritance

Most elements will inherit some style properties from their parent elements by default



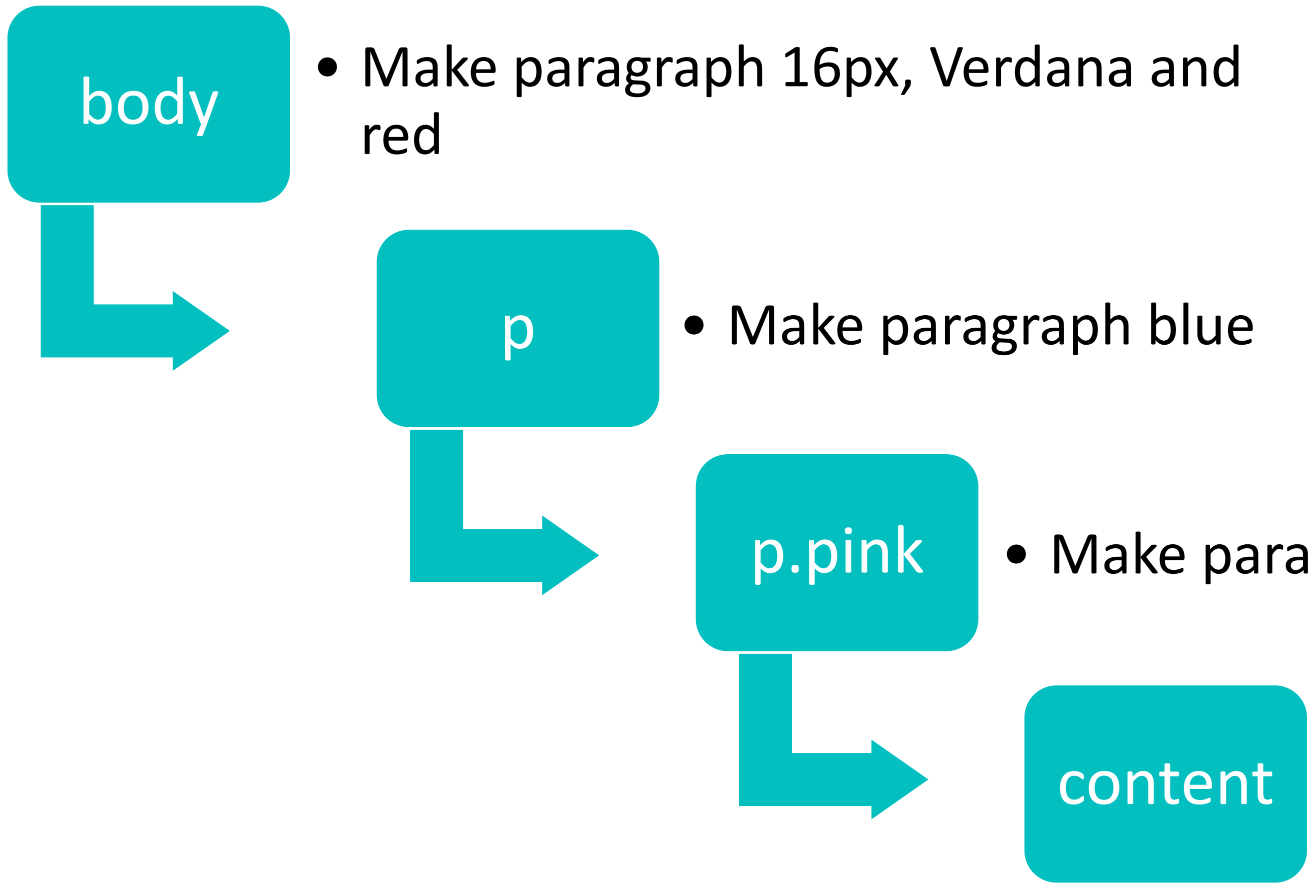
Specificity

Shortly after styling your first HTML elements, you will find yourself wanting more control over where your styles are applied

This is where ***specificity*** comes into play

Specificity refers to how specific your selector is in naming an element

Specificity (cont.)



- Make paragraph 16px, Verdana and red
- Make paragraph blue
- Make paragraph pink
- 16px, Verdana, pink

Syntax

CSS Rule

CSS Selector

CSS Declaration

CSS Rule

```
selector {
    property: value;
}
```

Every style is defined by a **selector** and a **declaration**.

The declaration contains at least one **property/value** pair.

Together they form a **CSS Rule.**

CSS Selector

The CSS Selector associates css rules with HTML elements

The selector is typed in front of the declaration

Example:

```
p {
    text-color: red
}
```

Typically extra space indentations and line feed are added for readability

CSS Declaration

The declaration is always defined as a property/value pair.

The two are separated by a colon and with a semi-colon.

How you define these pair, determines how your HTML elements are displayed.

Selectors

p	Type (element)
#	ID reference
.	Class reference

Type (element) selectors

The simplest selector is the type selector.

This selector targets html element by name.

```
body { declaration }
p { declaration }
h1, h2 { declaration }
ul { declaration }
```

ID Selectors

An ID is an html attribute.

It can be added to all HTML elements.

You then reference that ID in your CSS with a #

```
#logo { declaration }

<img id=”logo” src=”…” alt=”…” />
```

Class selectors

A class is an HTML attribute

It can be added to all HTML elements

You reference the class with a period.

```
.small-image { declaration }

<img id=”logo” class=”small-image” src=”…” alt=”…” />
```

Examples

```
body {
    background-color: #ff00ff;
}

h1.header, h2 {
    text-align: center;
    font-size: 32px;
    font-weight: bold;
}

#logo {
    text-color: #6F6F6C
}
```

IDs vs. Classes

The most important difference is that IDs are unique on a per page basis.
Whereas classes can be defined multiple times.

Classes can be used to style multiple elements in a common way

Ids can be used to style a specific unique element.

ID is more specific than a class

An element can have one ID and multiple classes

IDs vs. Classes (cont.)

ID: #344-34-4344

Class: Male

Class: Employee

ID: #123-54-9877

Class: Female

Class: Employee

Descendant selectors

CSS

```
#sidebar h1 .author { declaration }
```

HTML

```
<div id=”sidebar”>
    <h1>
        <span class=”author”>…</span>
    </h1>
</div>
```

A space between selectors indicates a descendant selector.

The above style targets the span with the author class.

Descendant selectors (cont.)

CSS

```
#sidebar h1 .author.important { declaration }
```

HTML

```
<div id=”sidebar”>
    <h1>
        <span class=”author important”>…</span>
    </h1>
</div>
```

Elements can have multiple classes, giving more granular control.

They are written in CSS in the exact order in which they appear in the HTML

Adding styling

Inside <style> tags

```
<style>
    div, p {
        text-color: #6F6F6C
    }
</style>
```

On an element

```
<p style=”color: #6F6F6C;” >
```

Referencing a css file

Or reference an external CSS file…

```
<link href=”/path/to/file/file.css” rel=”stylesheet” type=”text/css”/>
```

Place it in the <head>

This is the right way!

CSS Box Model

Content

- The content of the box, where text and images appear

Padding

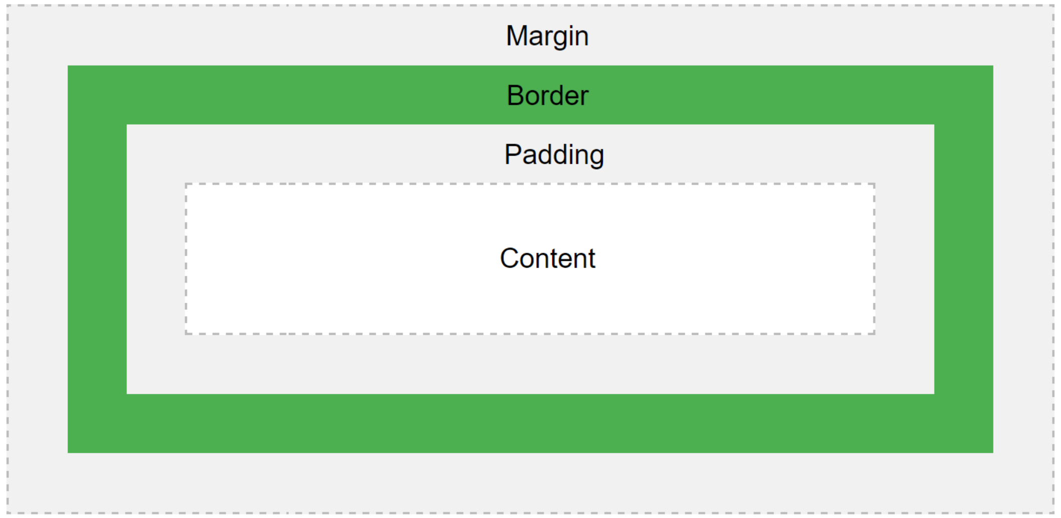
- Clears an area around the content. The padding is transparent

Border

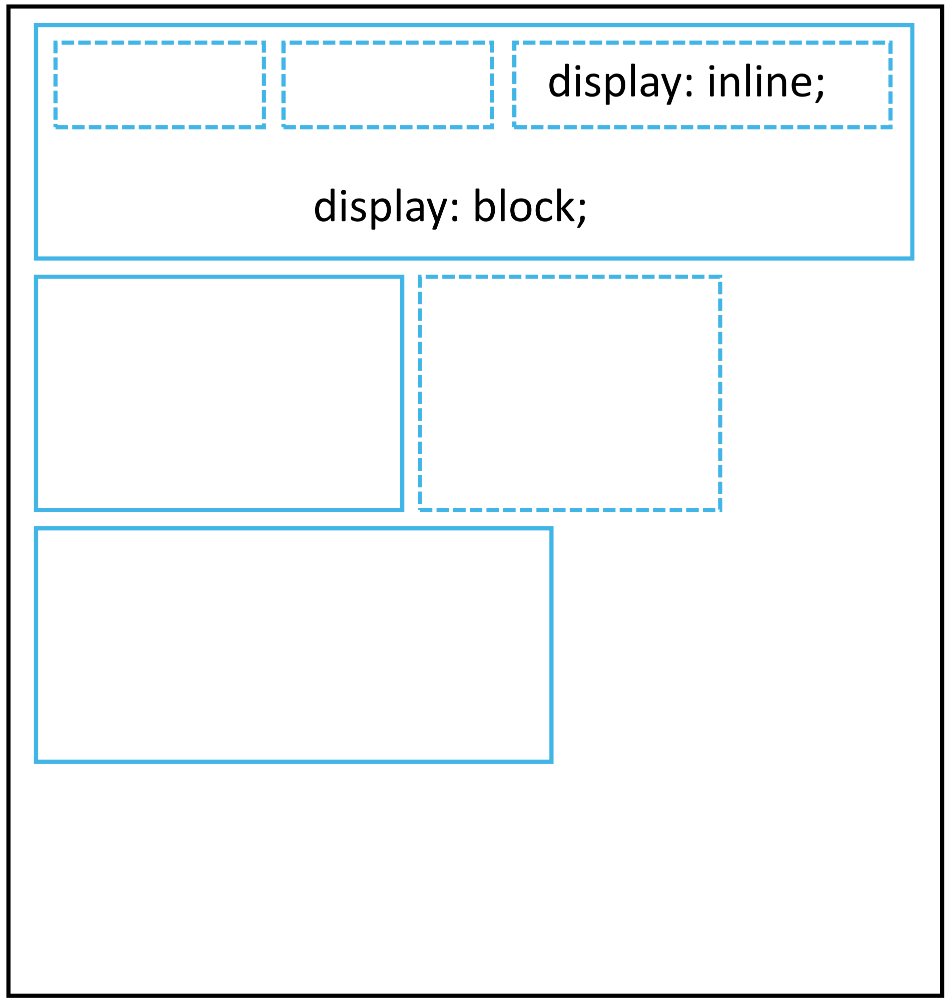
- A border that goes around the padding and content

Margin

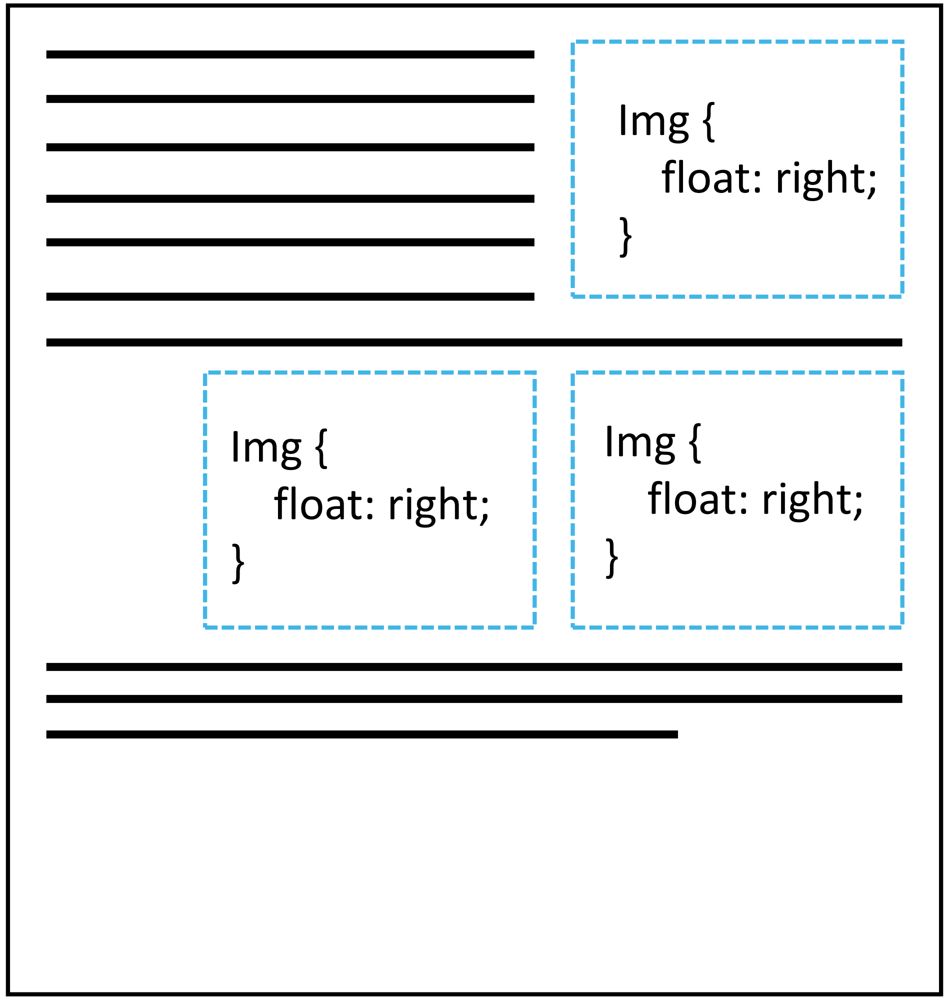
- Clears an area outside the border. The margin is transparent



Placement



Placement (cont.)

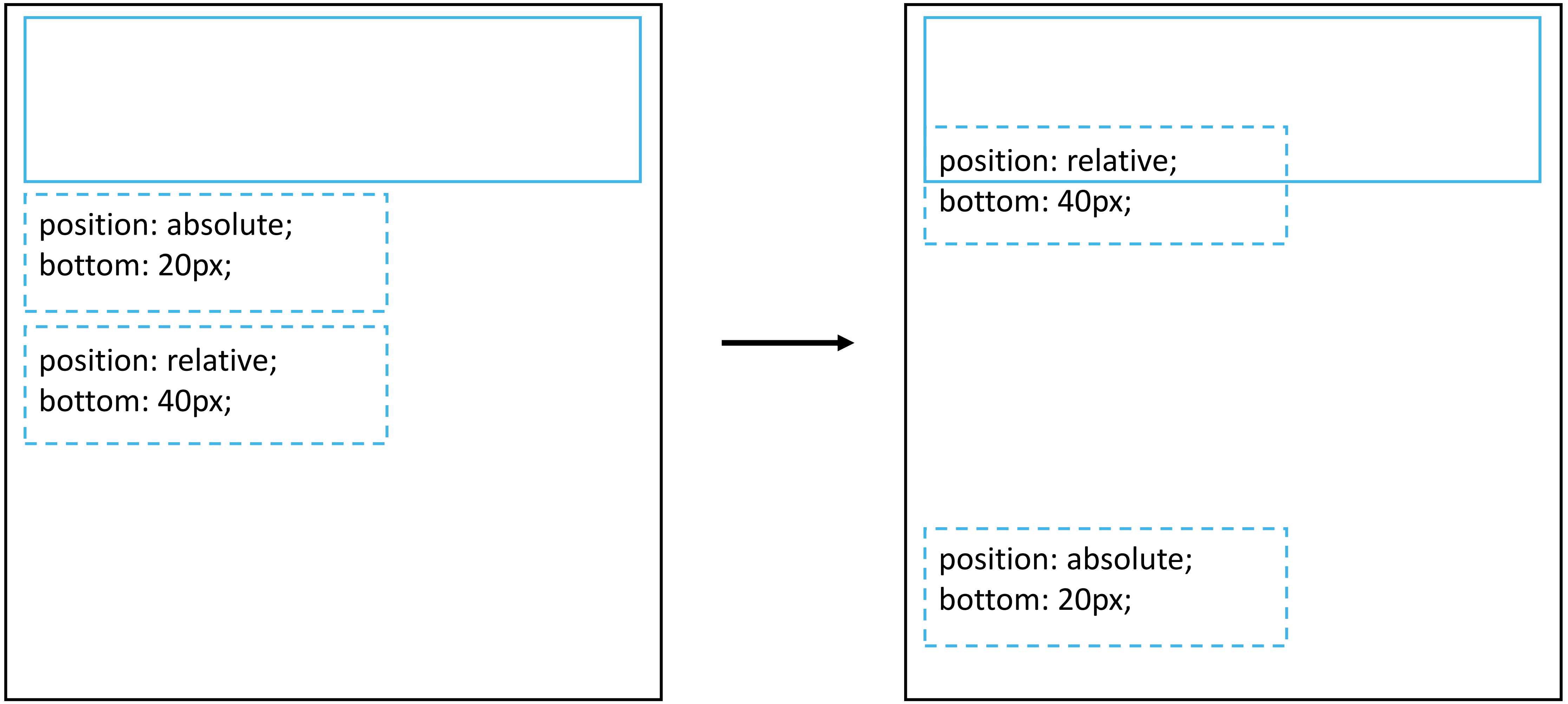


```
Img {
    float: right;
}
```

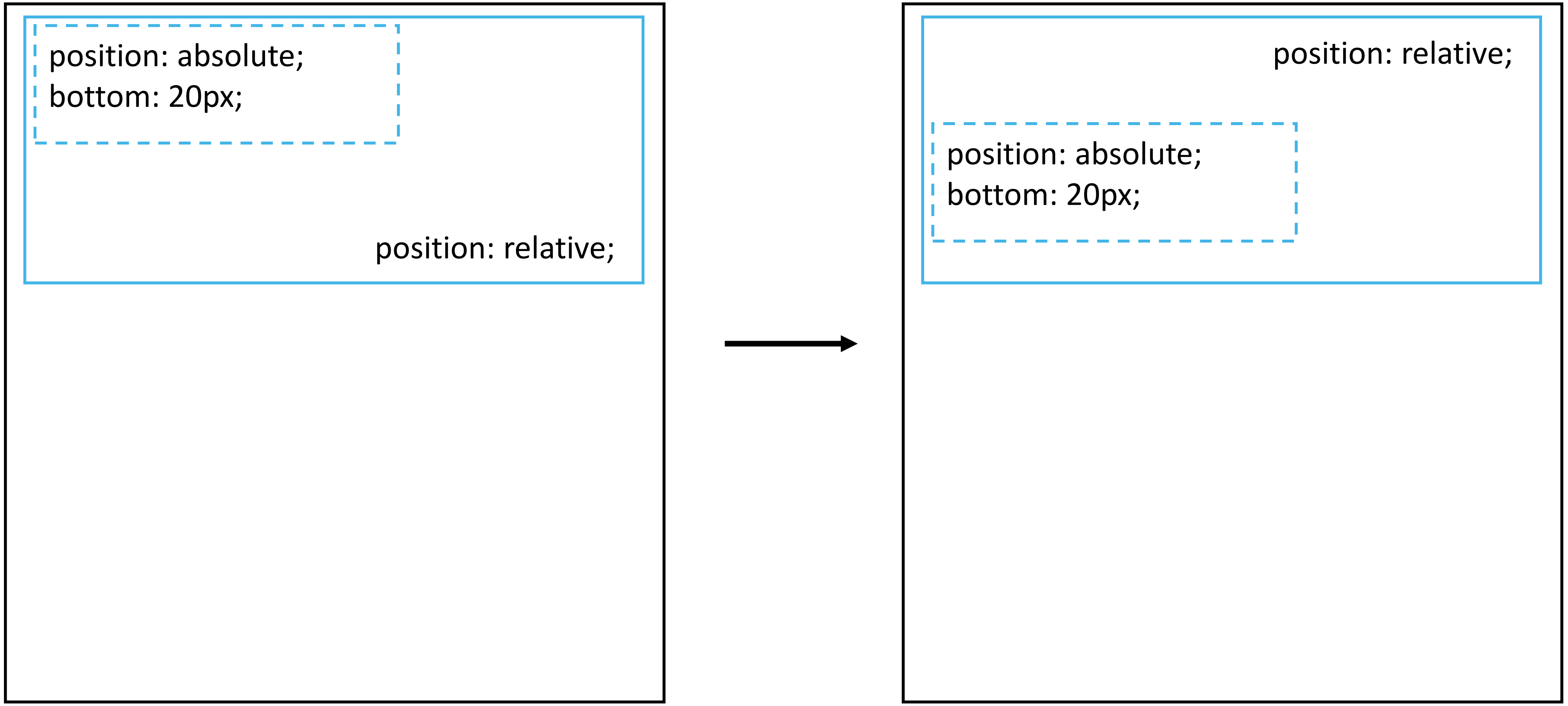
```
Img {
    float: right;
}
```

```
Img {
    float: right;
}
```

Positioning



Positioning (cont.)



Exercise
→ Style your Google web page to look more like the real thing

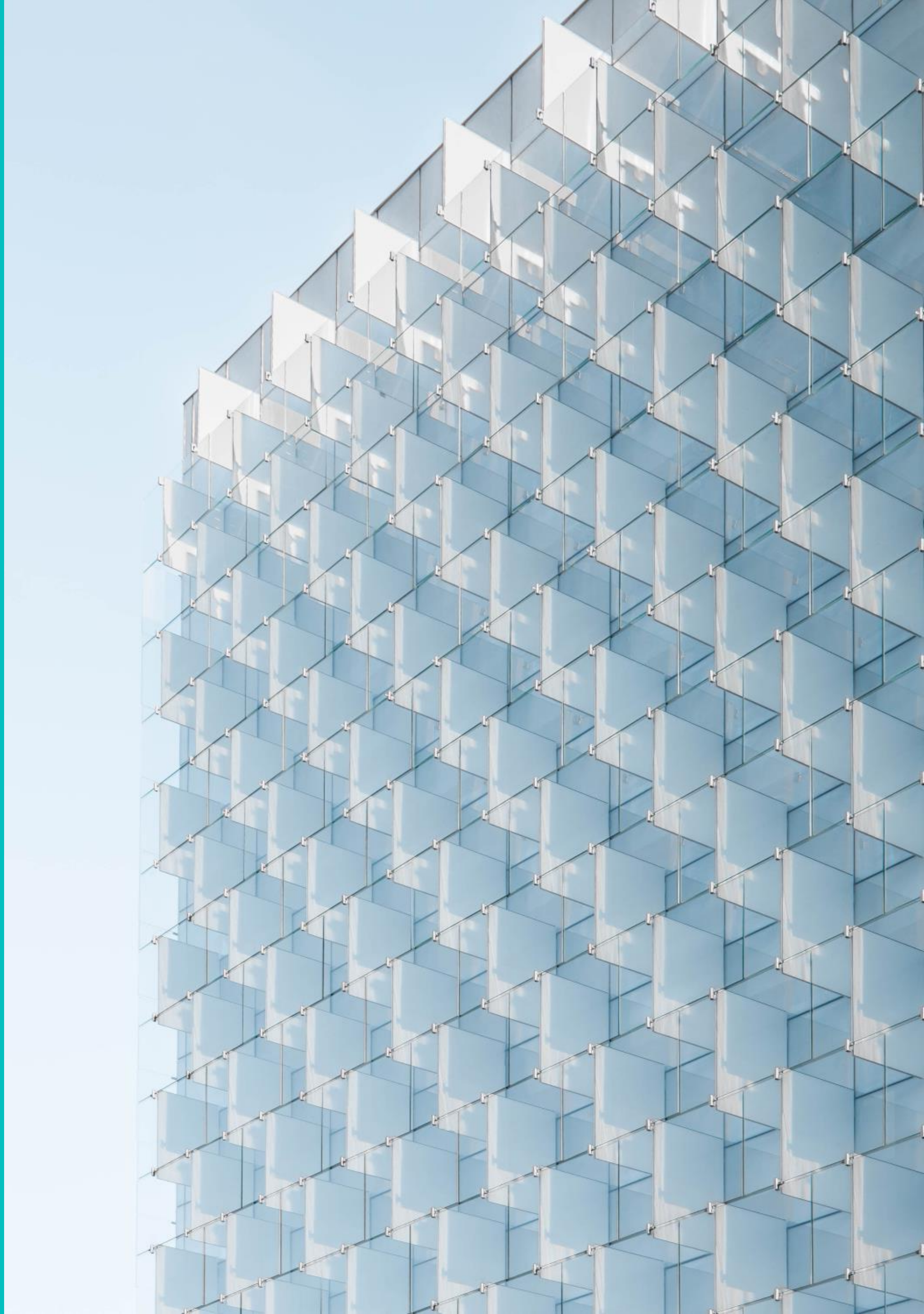
codecademy

Steps

1. Create an account on codecademy.com (You should have done that)
2. Start the course ”Learn CSS”
3. Complete the following:
    1. CSS Fundementals

This course should teach everyone without any prior knowledge, the fundementals of HTML and CSS.

CSS Grid



Preparation

Download Mozilla Firefox Developer Edition (also known as Firefox Quantum)

It has an awesome grid inspector!

Pretext

Web layouts are broken

We’ve basically just refined how we break them

Flex box fixed a lot of issues

Lots of markup consisted of wrappers within wrappers

Problem and solution

Problem:

Current tools for web layout is *content-out* and *one-dimensional*

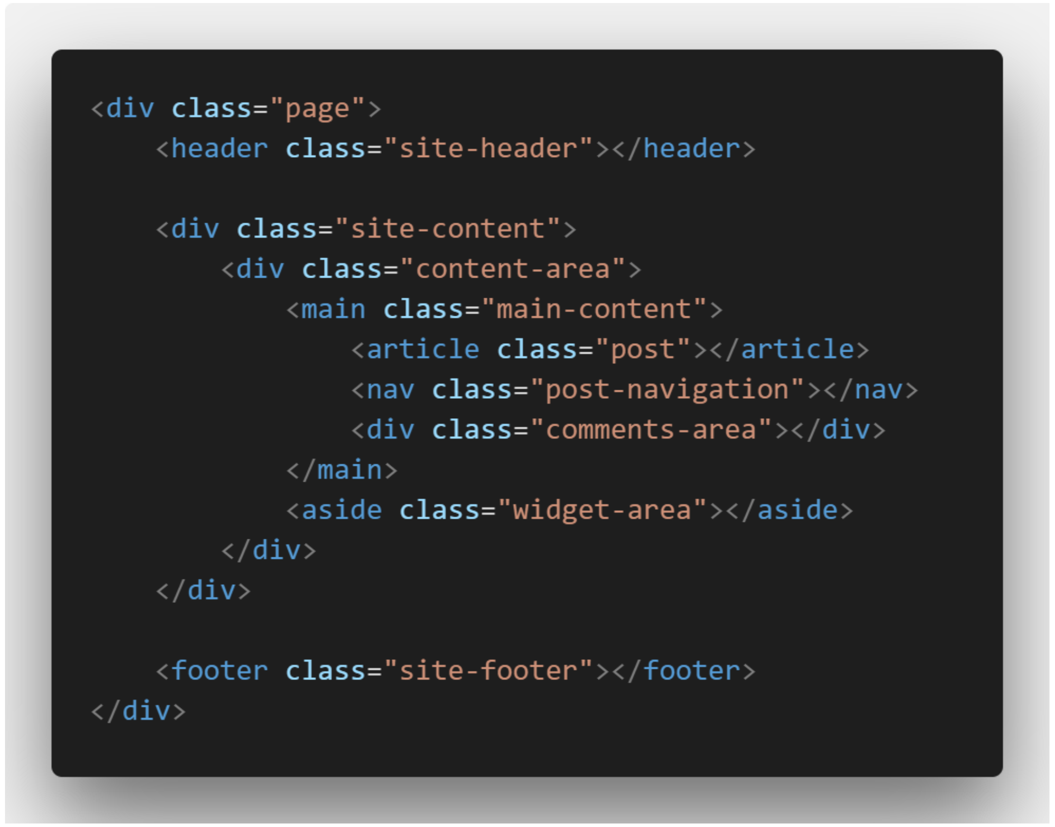
Solution:

Two-dimensional layout-in tool to separate content from presentation

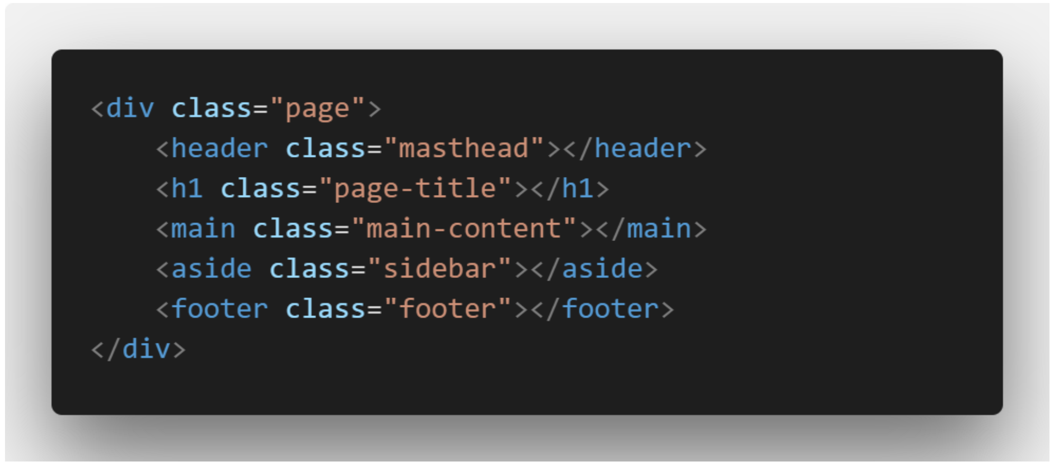
A griddy approach

Instead of relying on multiple wrappers,

we instead rely of defining the space each element will occupy.

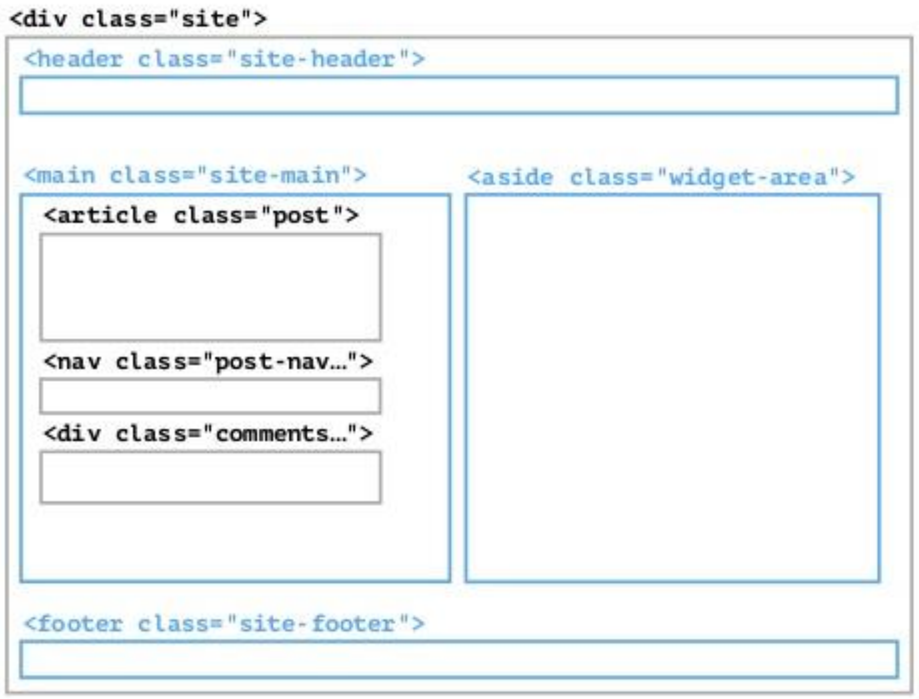


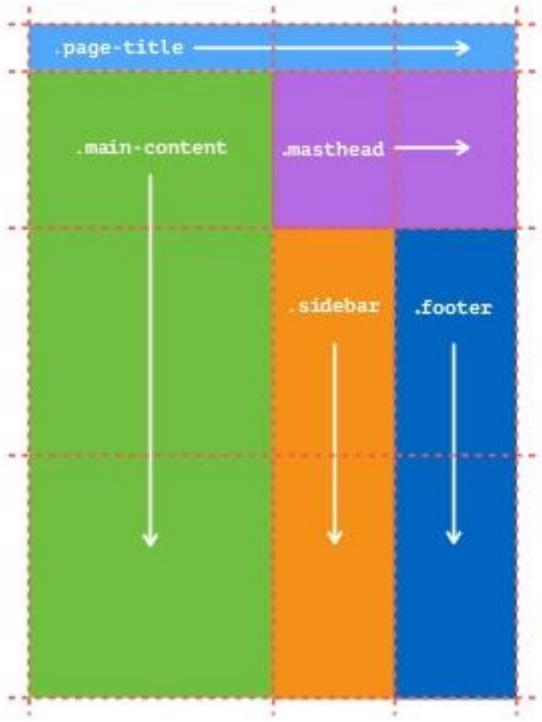
```html
<div class="page">
    <header class="site-header"></header>

    <div class="site-content">
        <div class="content-area">
            <main class="main-content">
                <article class="post"></article>
                <nav class="post-navigation"></nav>
                <div class="comments-area"></div>
            </main>
            <aside class="widget-area"></aside>
        </div>
    </div>

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



```html
<div class="page">
    <header class="masthead"></header>
    <h1 class="page-title"></h1>
    <main class="main-content"></main>
    <aside class="sidebar"></aside>
    <footer class="footer"></footer>
</div>
```

A griddy approah (cont.)





Grid terminology

Grid ***container***

Grid ***item***

Grid ***line***

Grid ***cell***

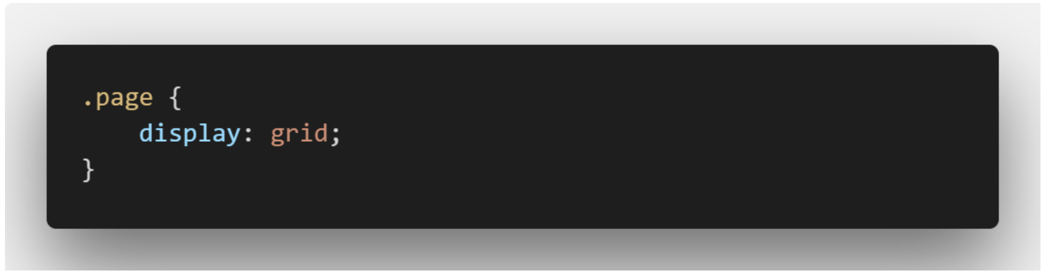
Grid ***track***

Grid ***area***

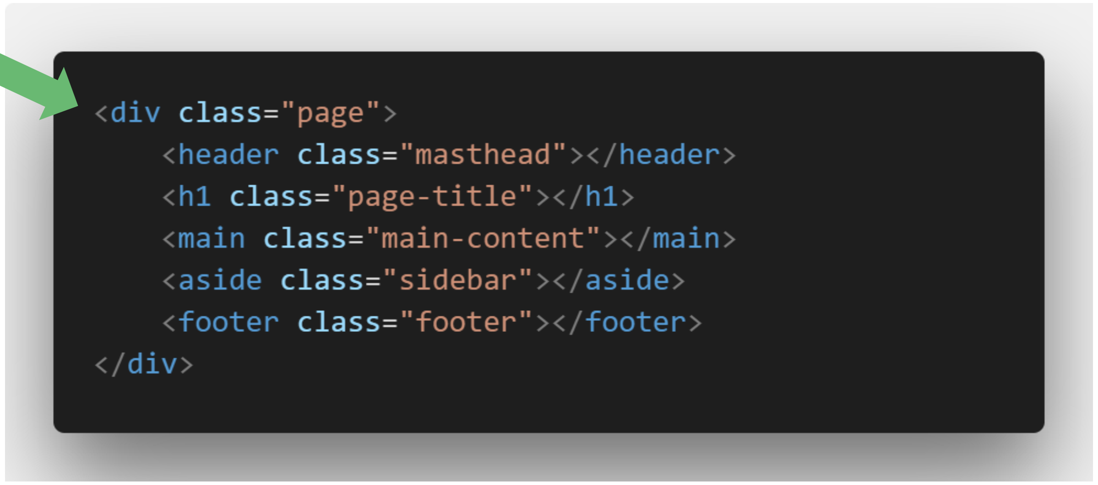
Grid ***gap***

Grid container

An element containing a grid is denoted by
display: grid;



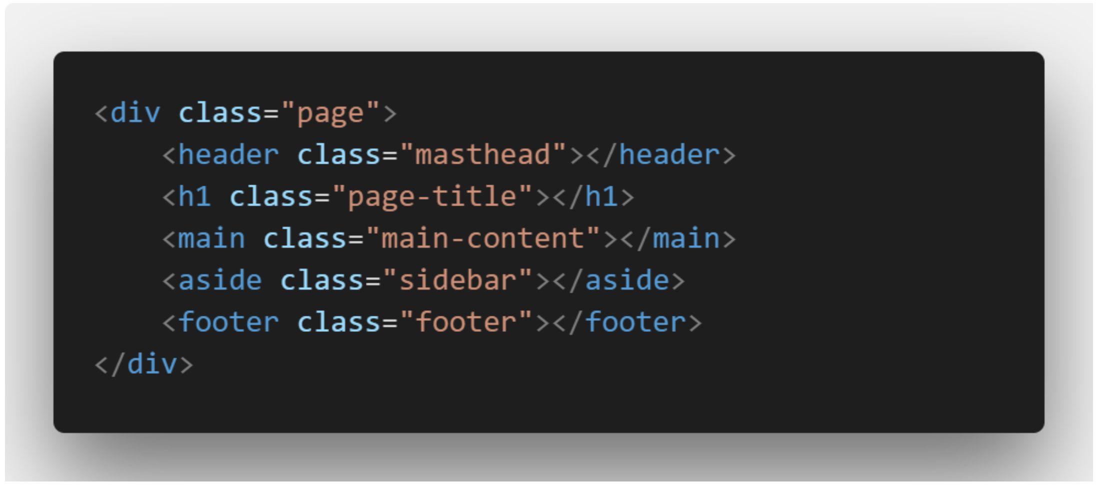
```css
.page {
    display: grid;
}
```



```html
<div class="page">
    <header class="masthead"></header>
    <h1 class="page-title"></h1>
    <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



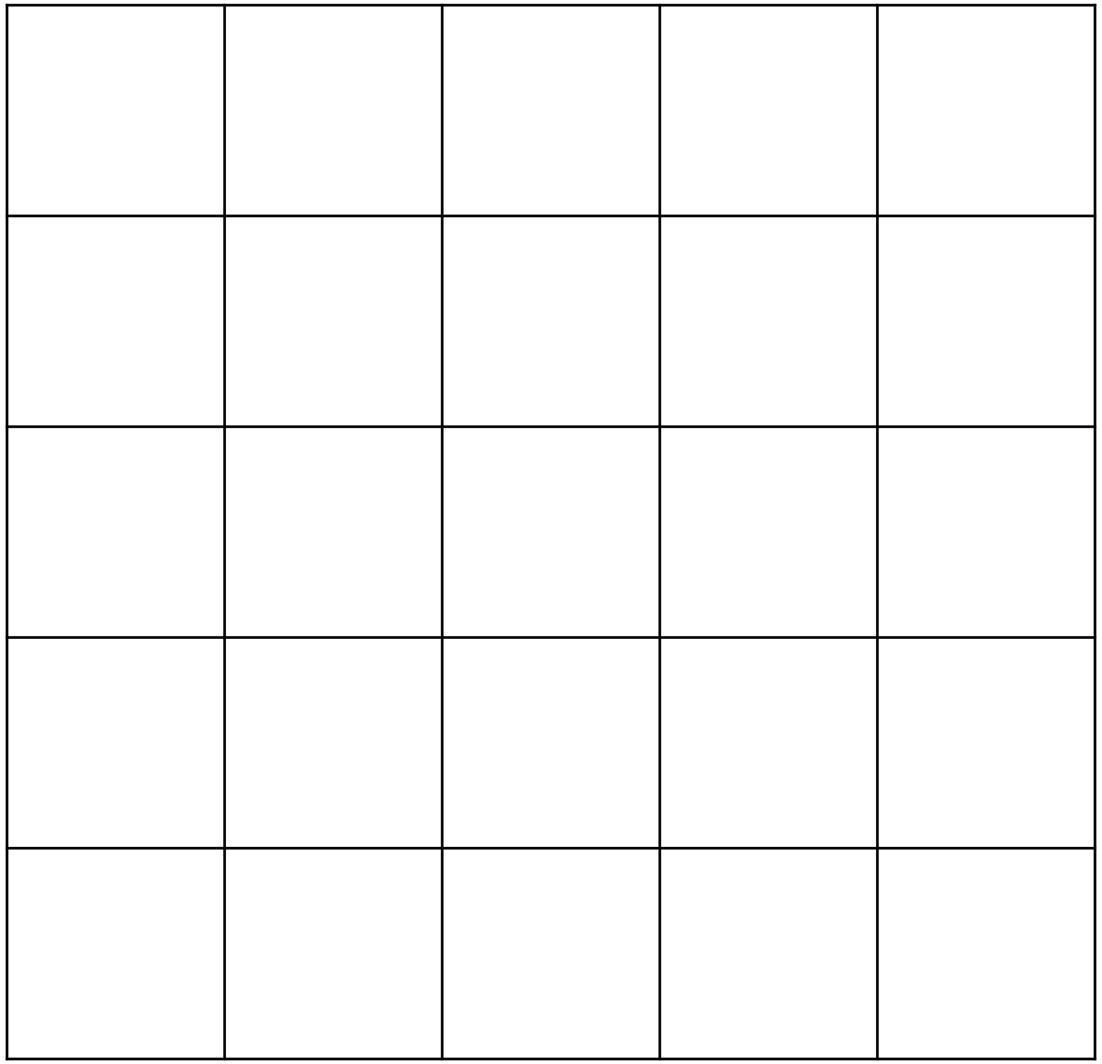
```
<div class="page">
    <header class="masthead"></header>
    <h1 class="page-title"></h1>
    <main class="main-content"></main>
    <aside class="sidebar"></aside>
    <footer class="footer"></footer>
</div>
```

Grid line

A grid consists of Rows and Columns

Rows are horizontal

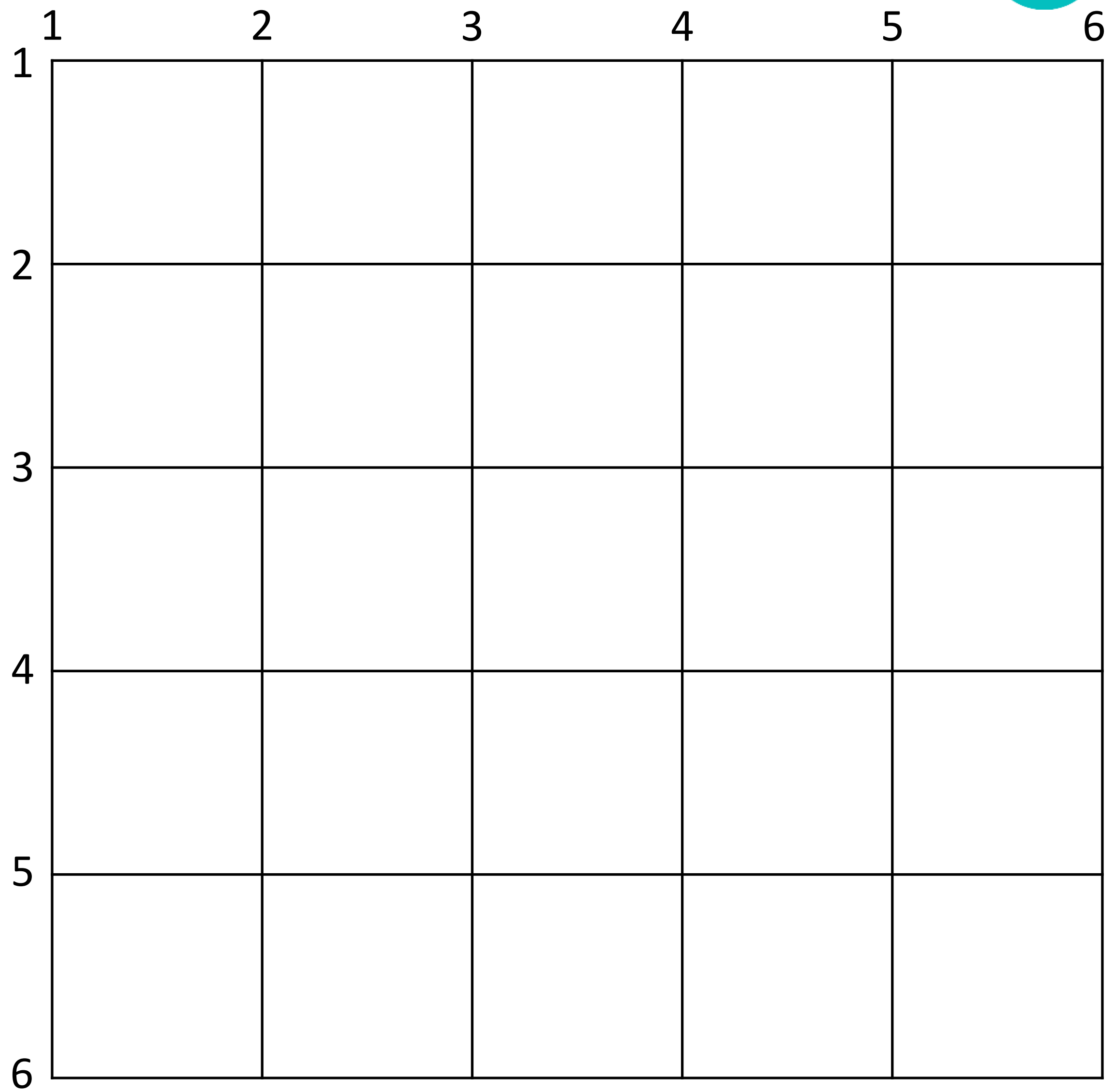
Columns are vertical



Grid line

Grid lines are referenced by number

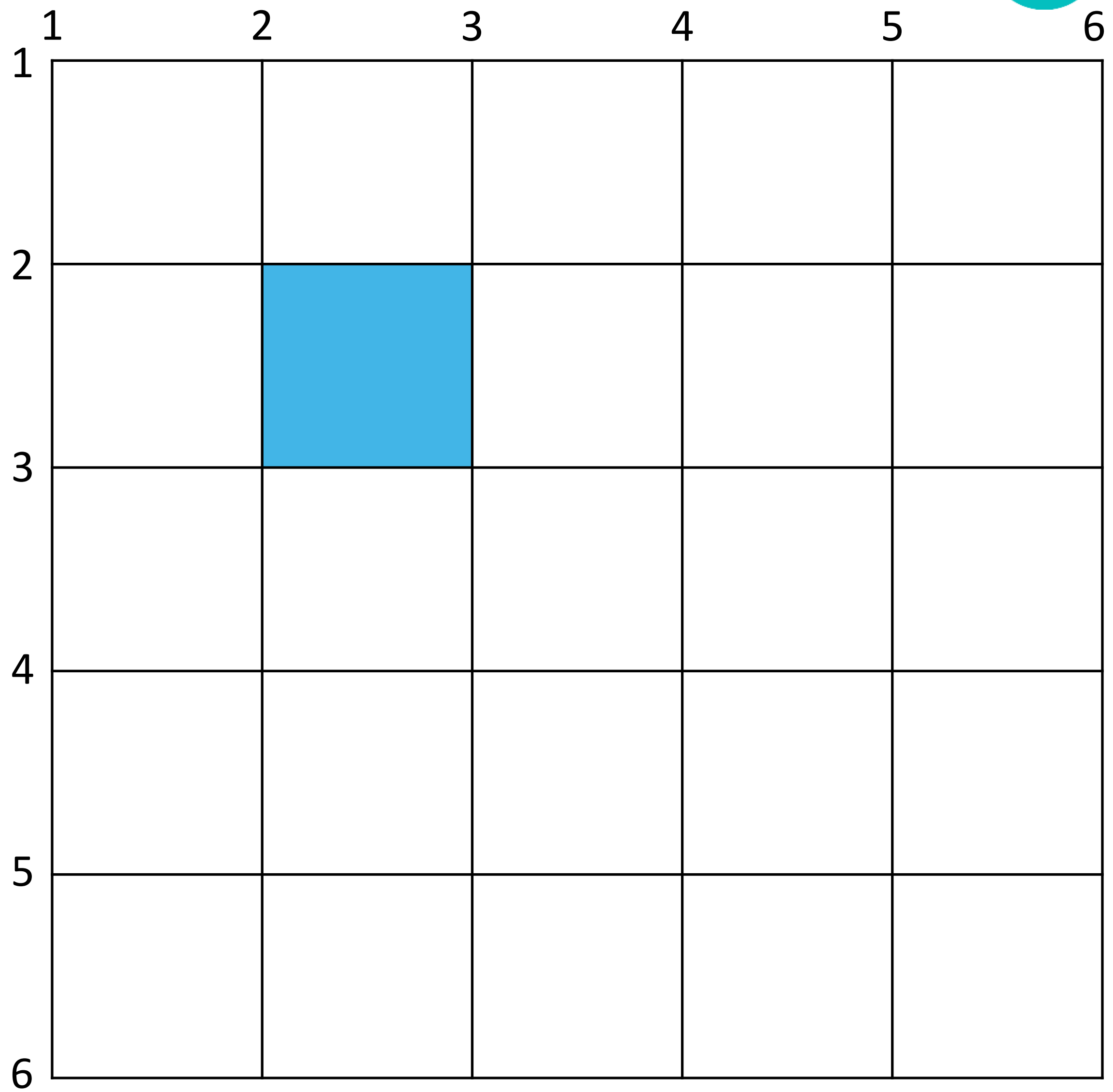
Starting and ending with outer borders



Grid cell

The intersection between a grid row and a grid column.

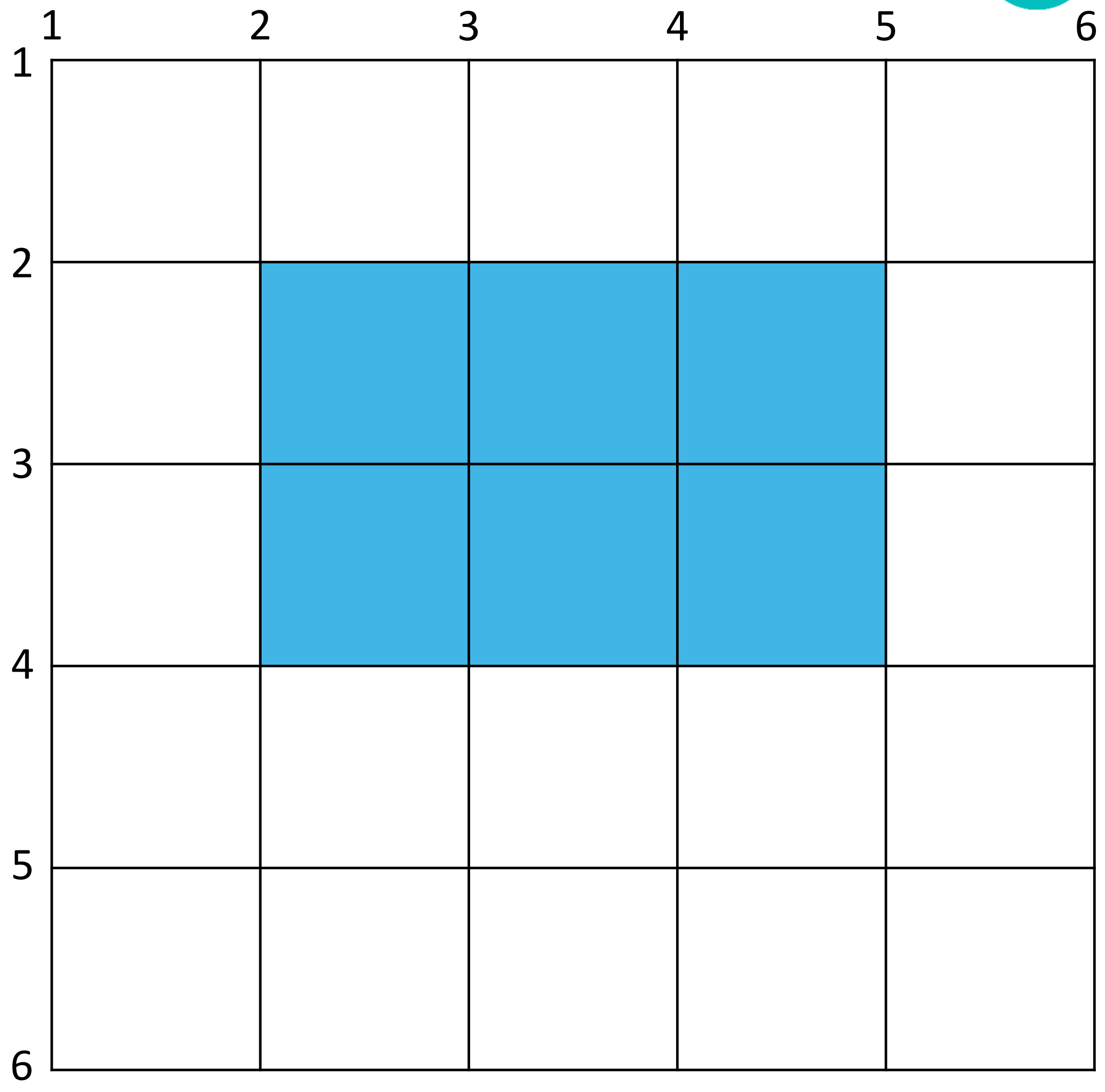
Effectively the same as a table cell



Grid area

Rectangular area between 4 specified grid lines.

Grid areas can cover one or more cells.

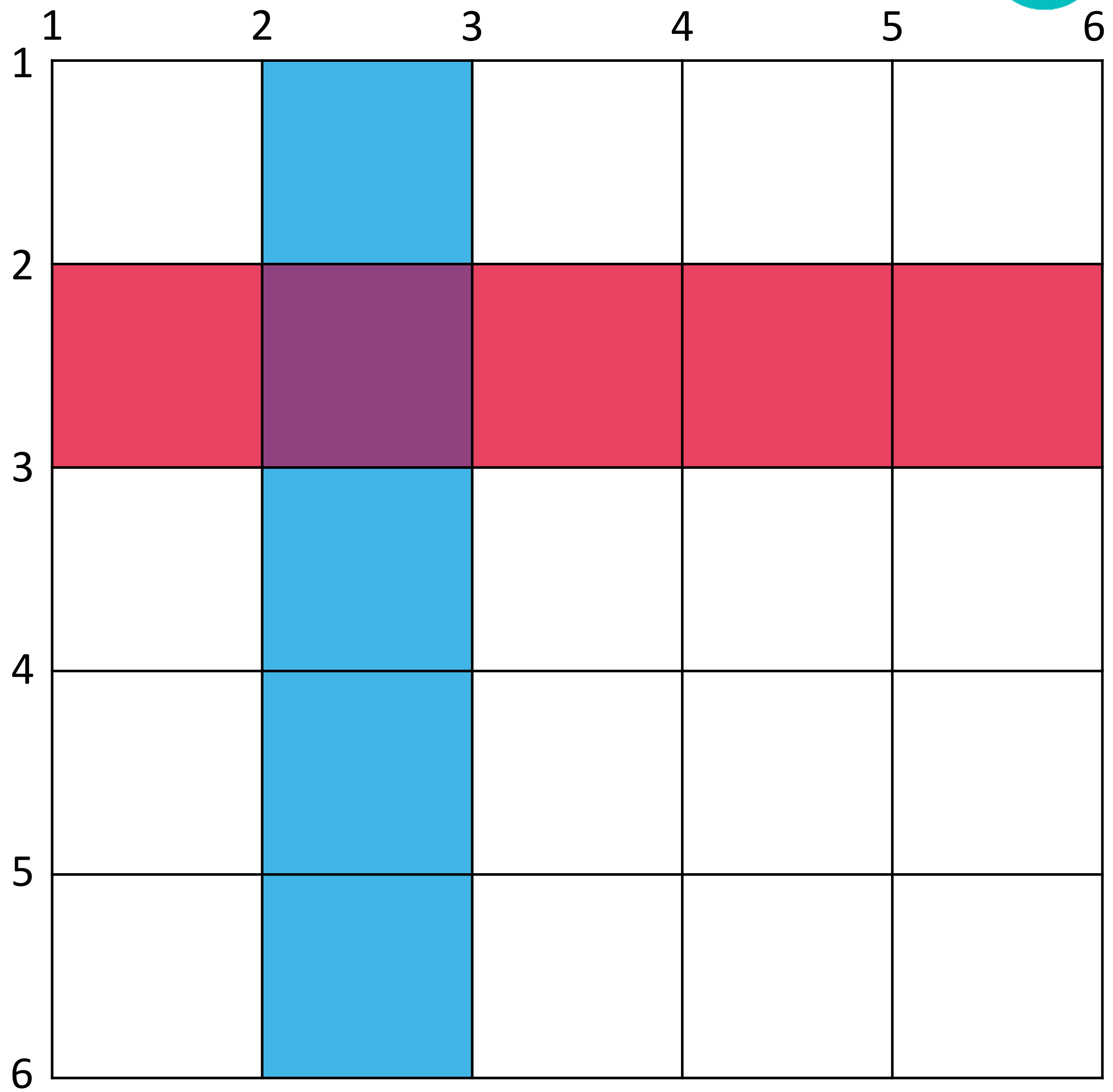


Grid track

The space between two or more adjacent grid lines.

Row tracks are horizontal

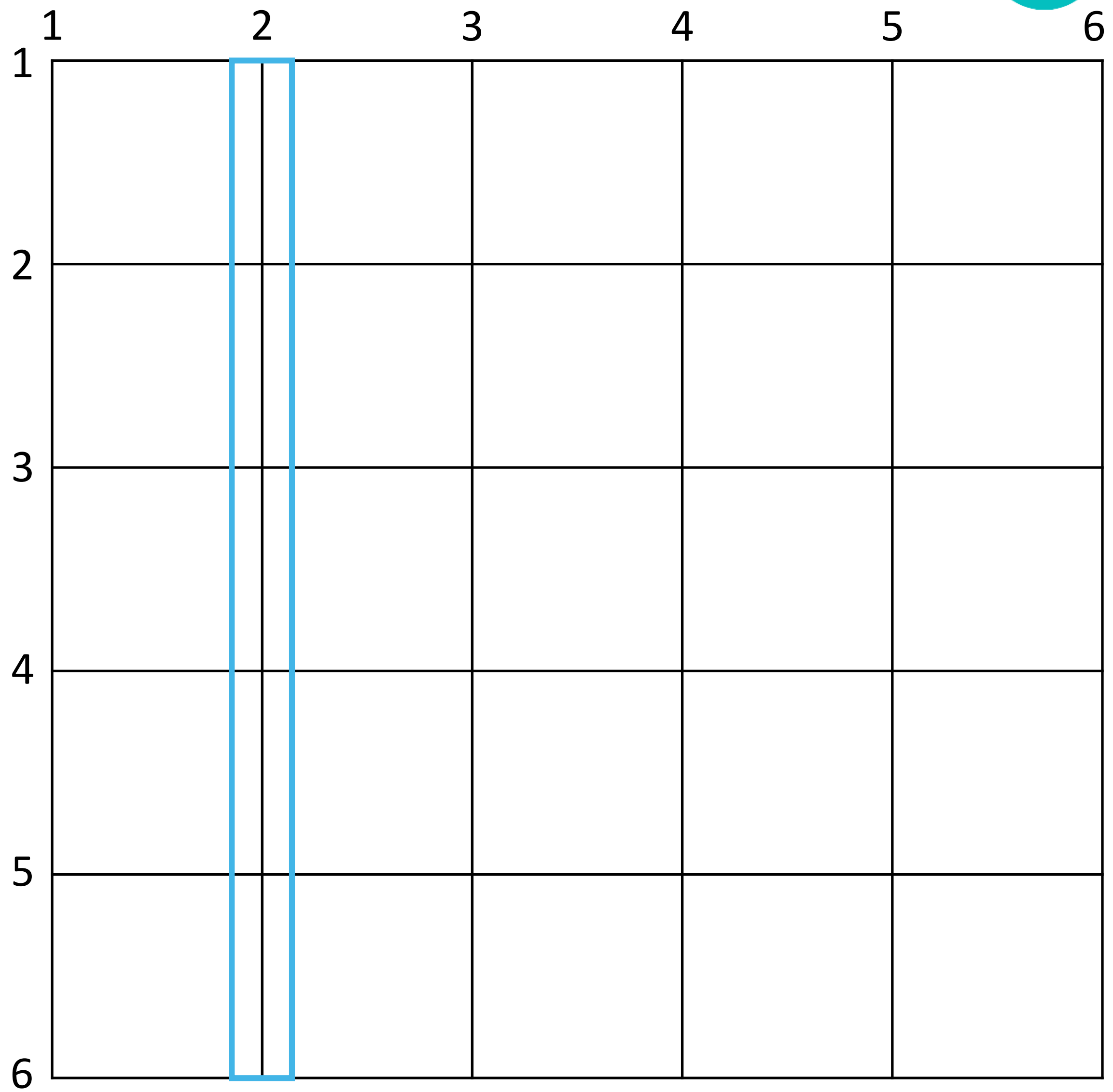
Column tracks are vertical



Grid gap

Empty space between grid tracks

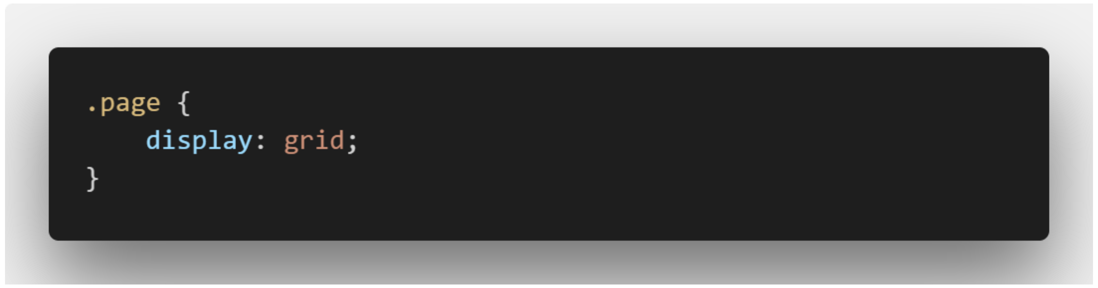
Commonly known as gutters



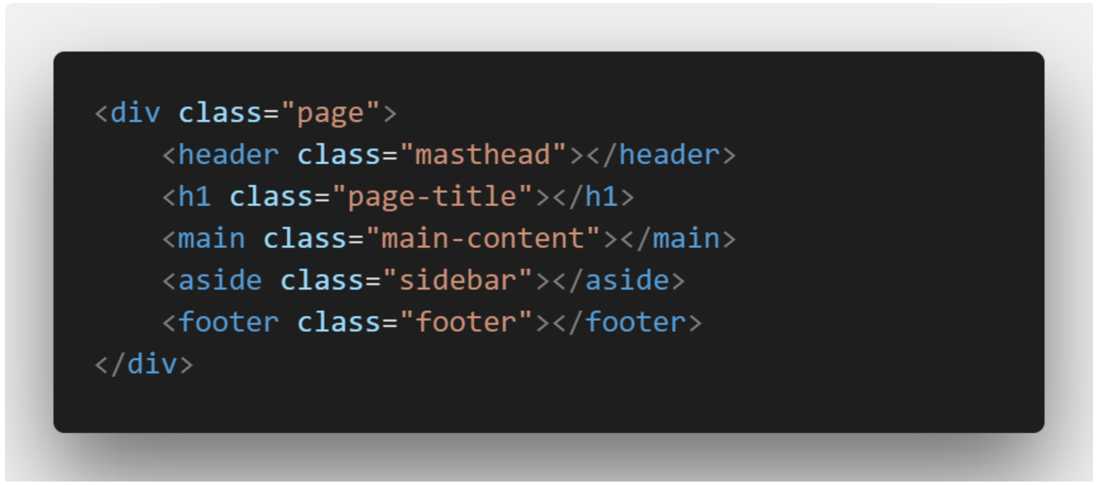
CSS Grid in a nutshell

1. Define a grid
2. Place items in that grid
3. Make world peace!

Defining a grid

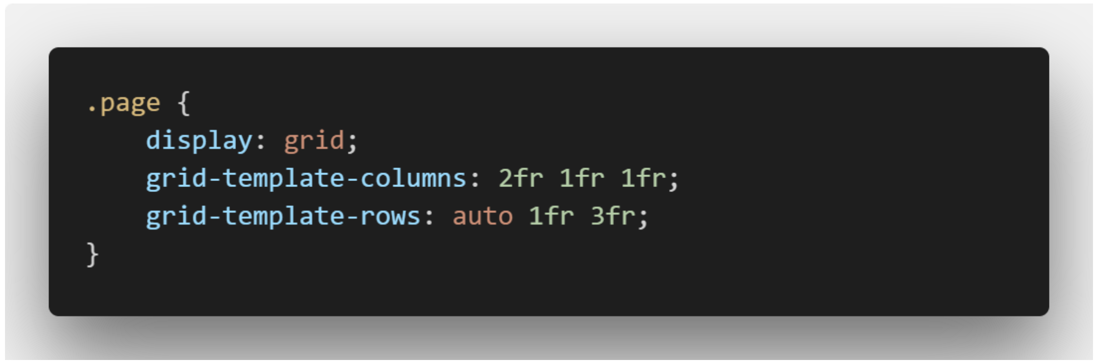


```css
.page {
    display: grid;
}
```

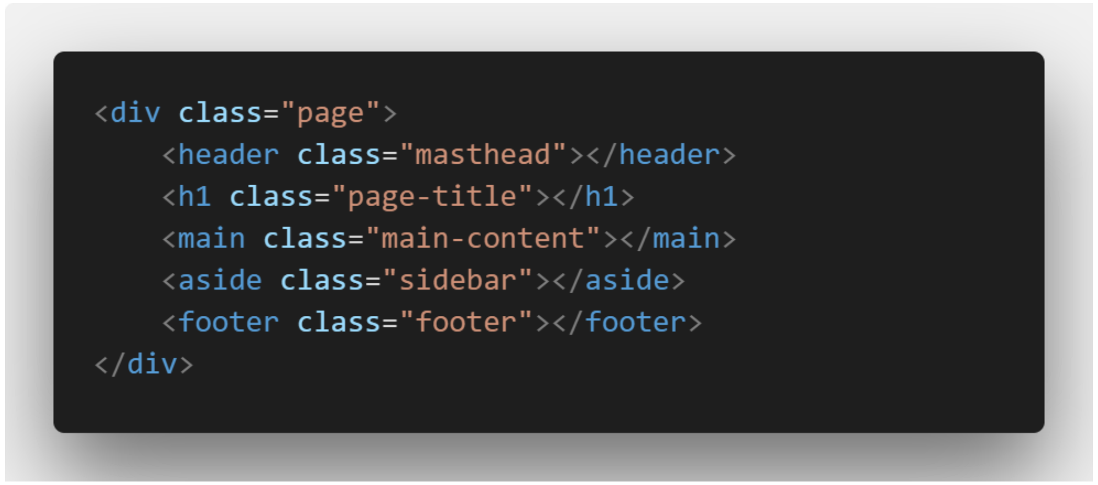


```html
<div class="page">
    <header class="masthead"></header>
    <h1 class="page-title"></h1>
    <main class="main-content"></main>
    <aside class="sidebar"></aside>
    <footer class="footer"></footer>
</div>
```

Defining a grid (cont.)

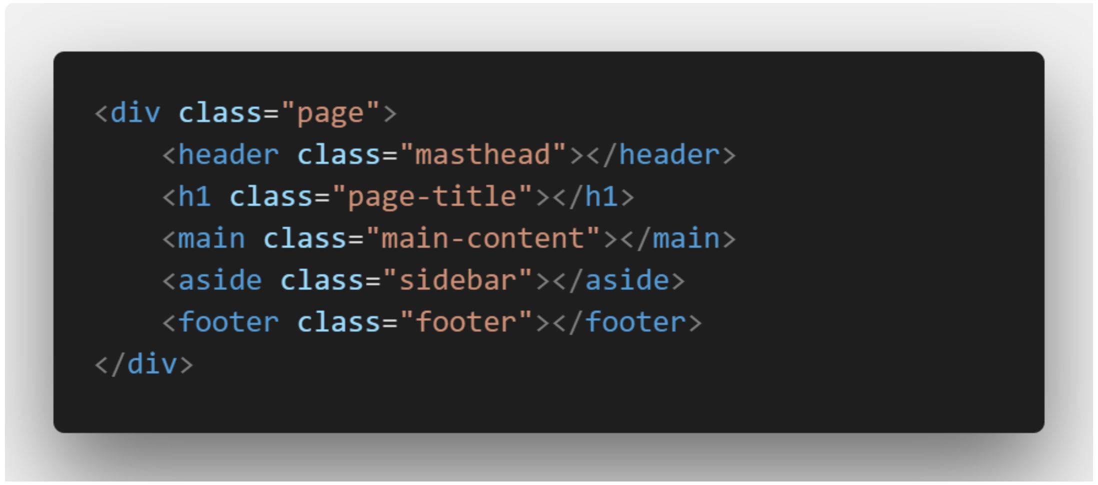


```css
.page {
    display: grid;
    grid-template-columns: 2fr 1fr 1fr;
    grid-template-rows: auto 1fr 3fr;
}
```



```html
<div class="page">
    <header class="masthead"></header>
    <h1 class="page-title"></h1>
    <main class="main-content"></main>
    <aside class="sidebar"></aside>
    <footer class="footer"></footer>
</div>
```

Defining a grid (cont.)



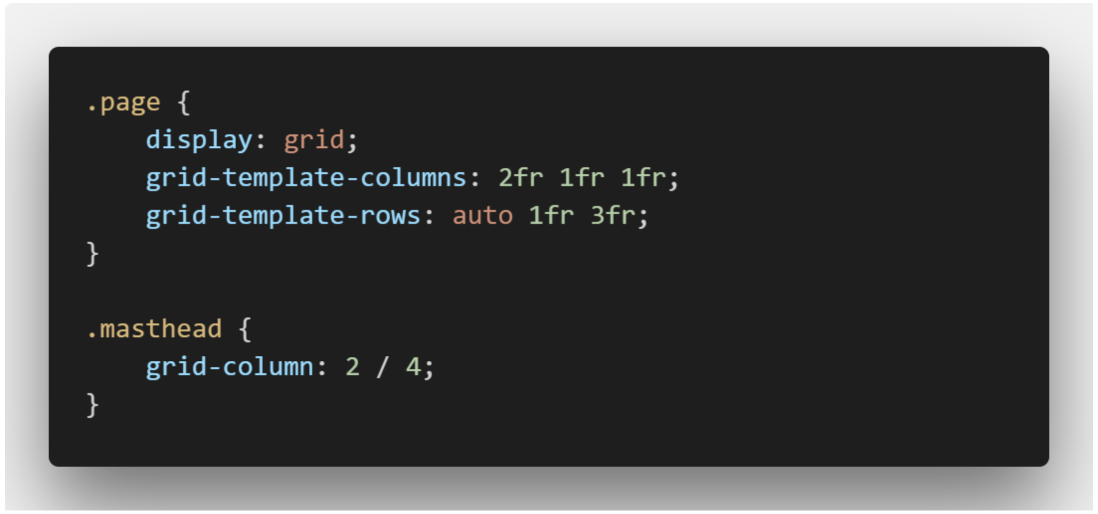
```html
<div class="page">
    <header class="masthead"></header>
    <h1 class="page-title"></h1>
    <main class="main-content"></main>
    <aside class="sidebar"></aside>
    <footer class="footer"></footer>
</div>
```

Grid items are automatically placed in the grid

It is populated from the top-left towards bottom-right

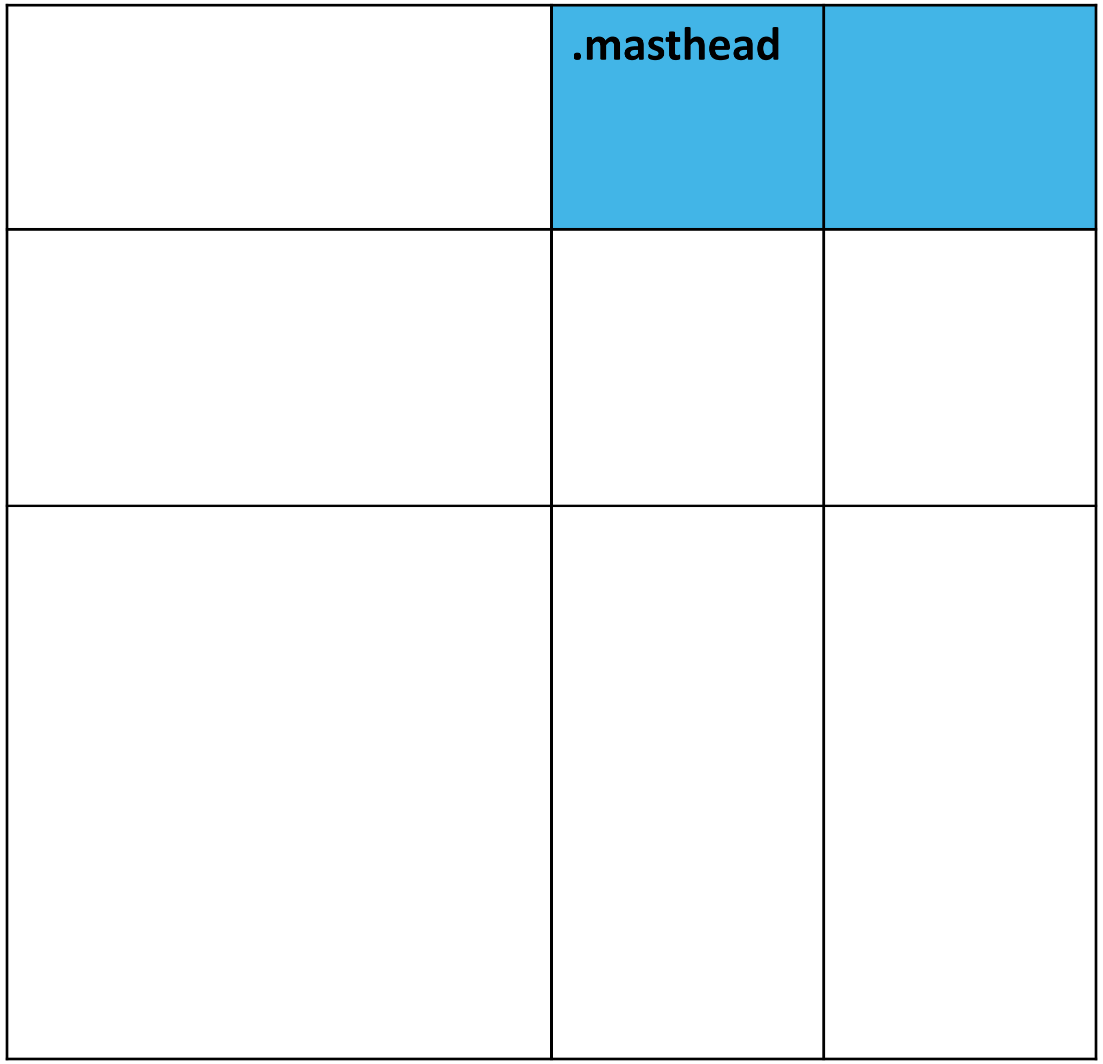
.masthead	**.page-title**	**.main-content**
.sidebar	**.footer**	

Defining a grid (cont.)

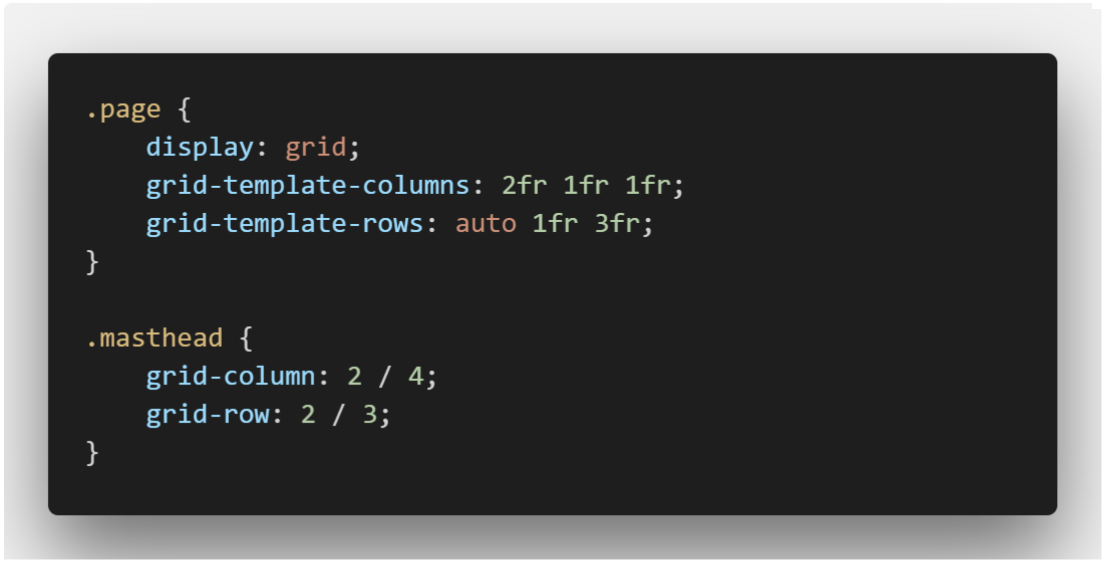


```
.page {
    display: grid;
    grid-template-columns: 2fr 1fr 1fr;
    grid-template-rows: auto 1fr 3fr;
}

.masthead {
    grid-column: 2 / 4;
}
```

Start at column line 2
End at column line 4

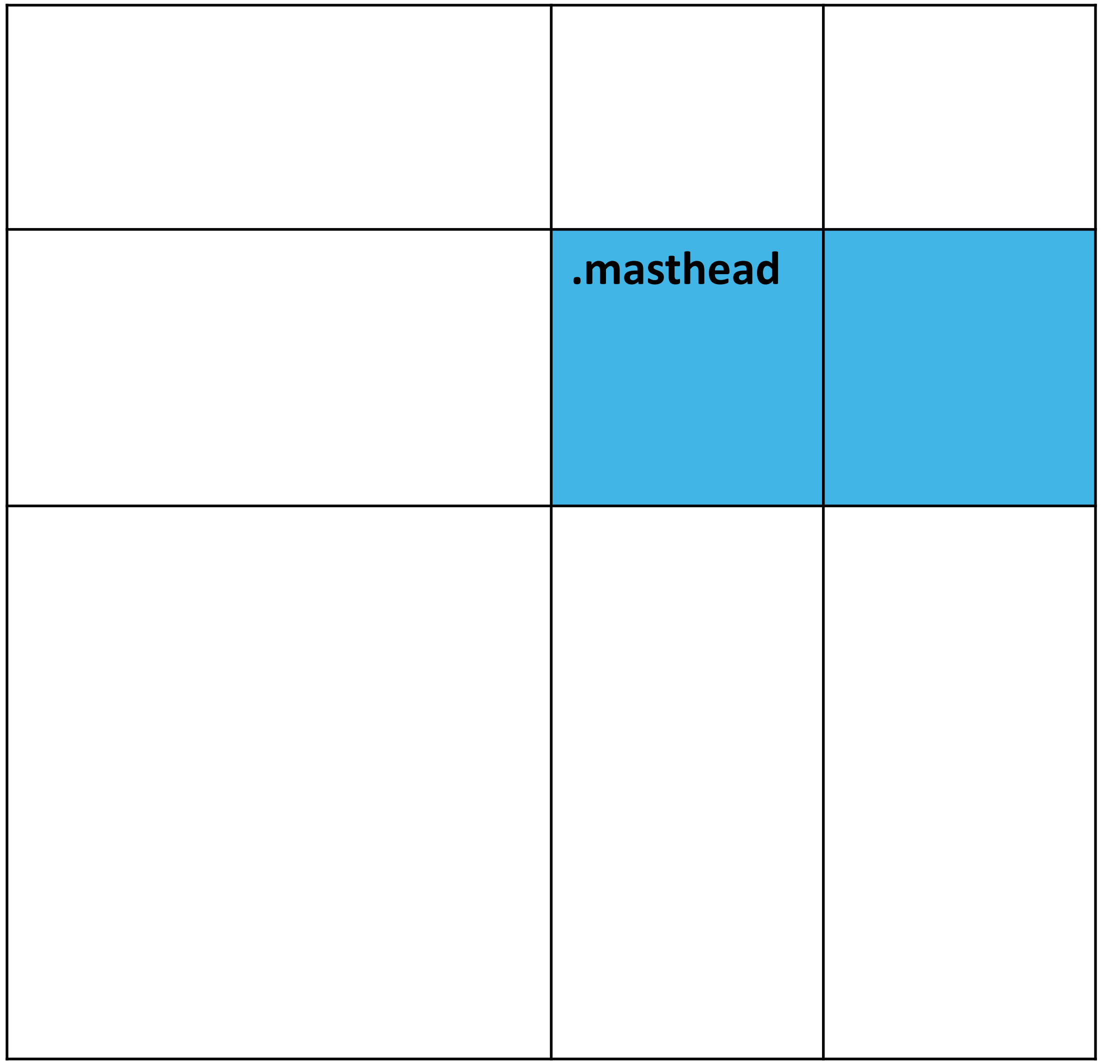


Defining a grid (cont.)



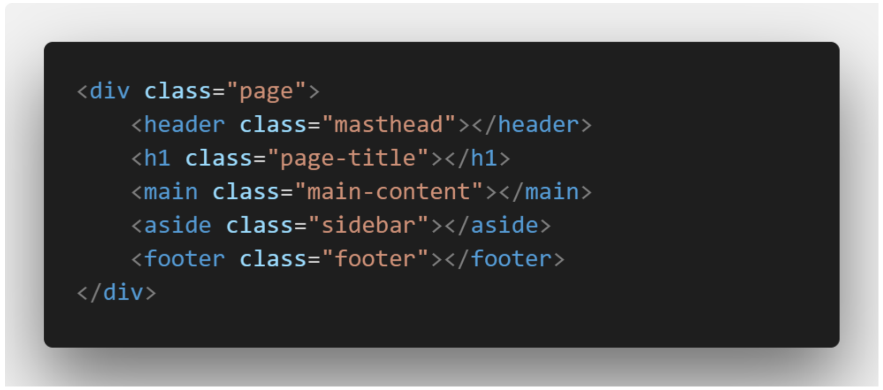
```
.page {
    display: grid;
    grid-template-columns: 2fr 1fr 1fr;
    grid-template-rows: auto 1fr 3fr;
}

.masthead {
    grid-column: 2 / 4;
    grid-row: 2 / 3;
}
```

Start at row line 2
End at row line 3



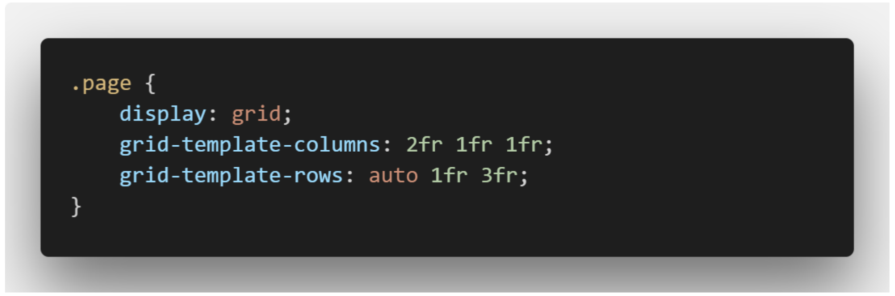
Exercise

Copy html:



```html
<div class="page">
    <header class="masthead"></header>
    <h1 class="page-title"></h1>
    <main class="main-content"></main>
    <aside class="sidebar"></aside>
    <footer class="footer"></footer>
</div>
```

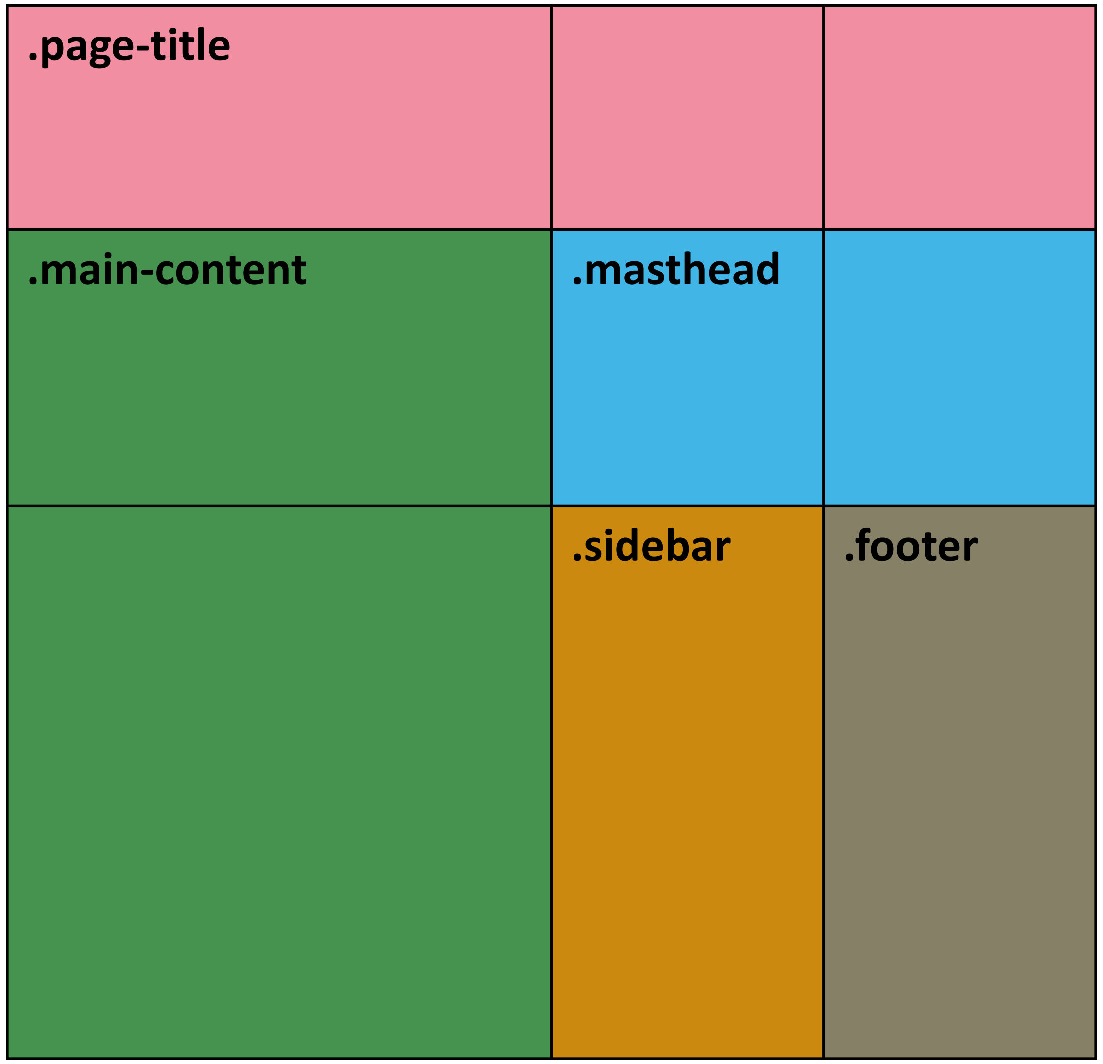
Copy Css:



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

Place elements according to the grid.

Put some content inside each element to verify the placement



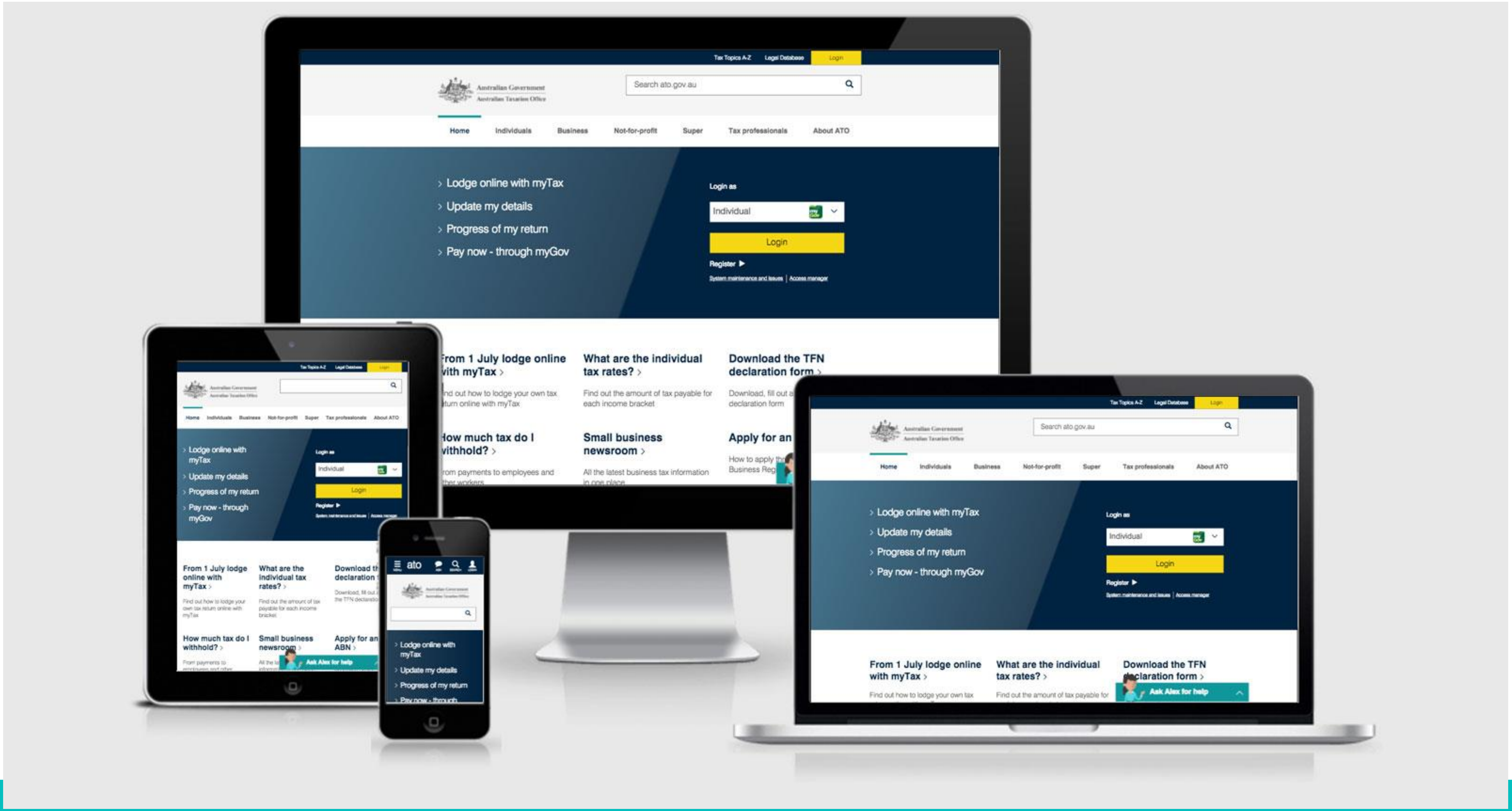
Looks promising!

BUT

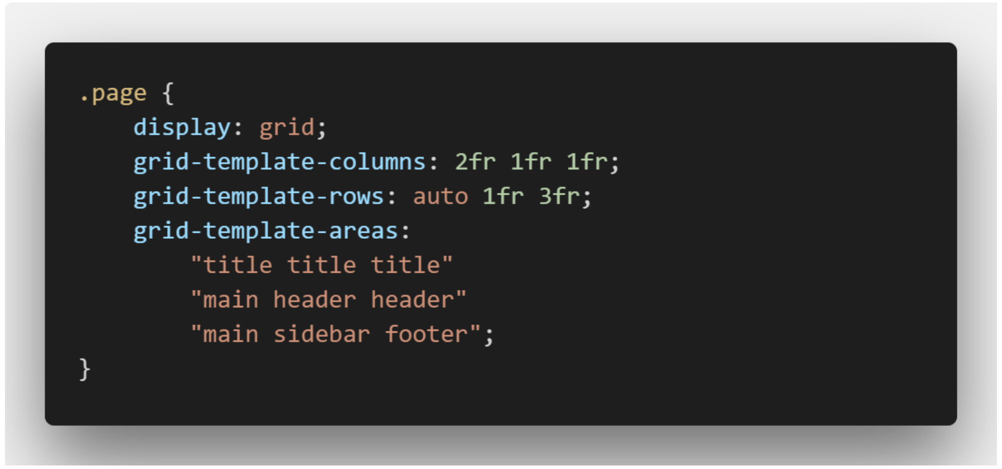
Remembering different lines is difficult!

Especially if the site is responsive.

What is responsive?



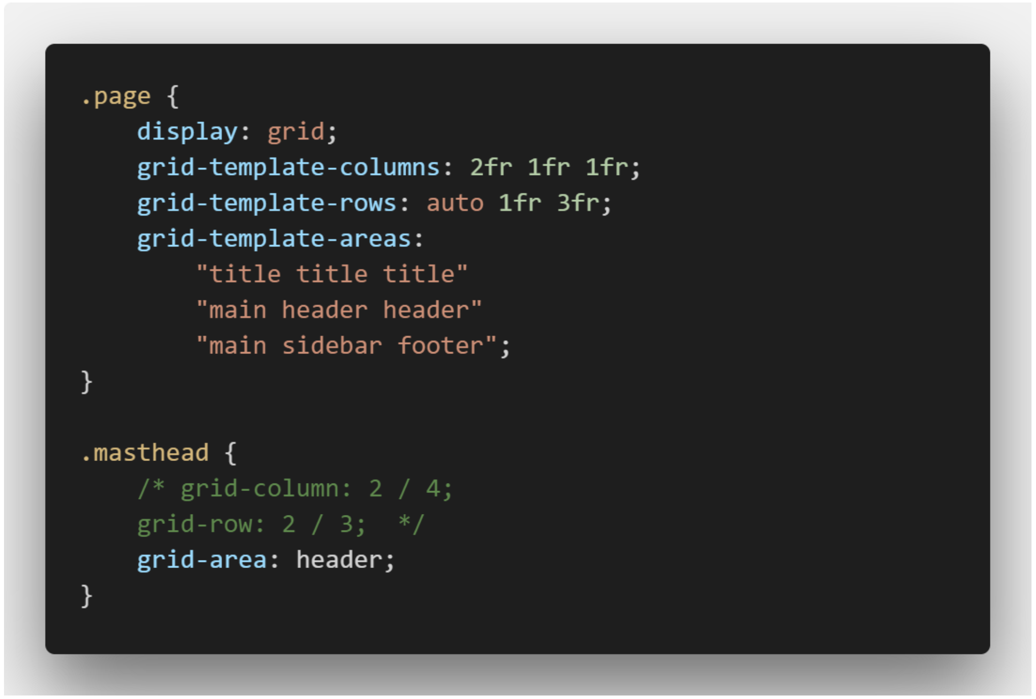
Grid Template areas



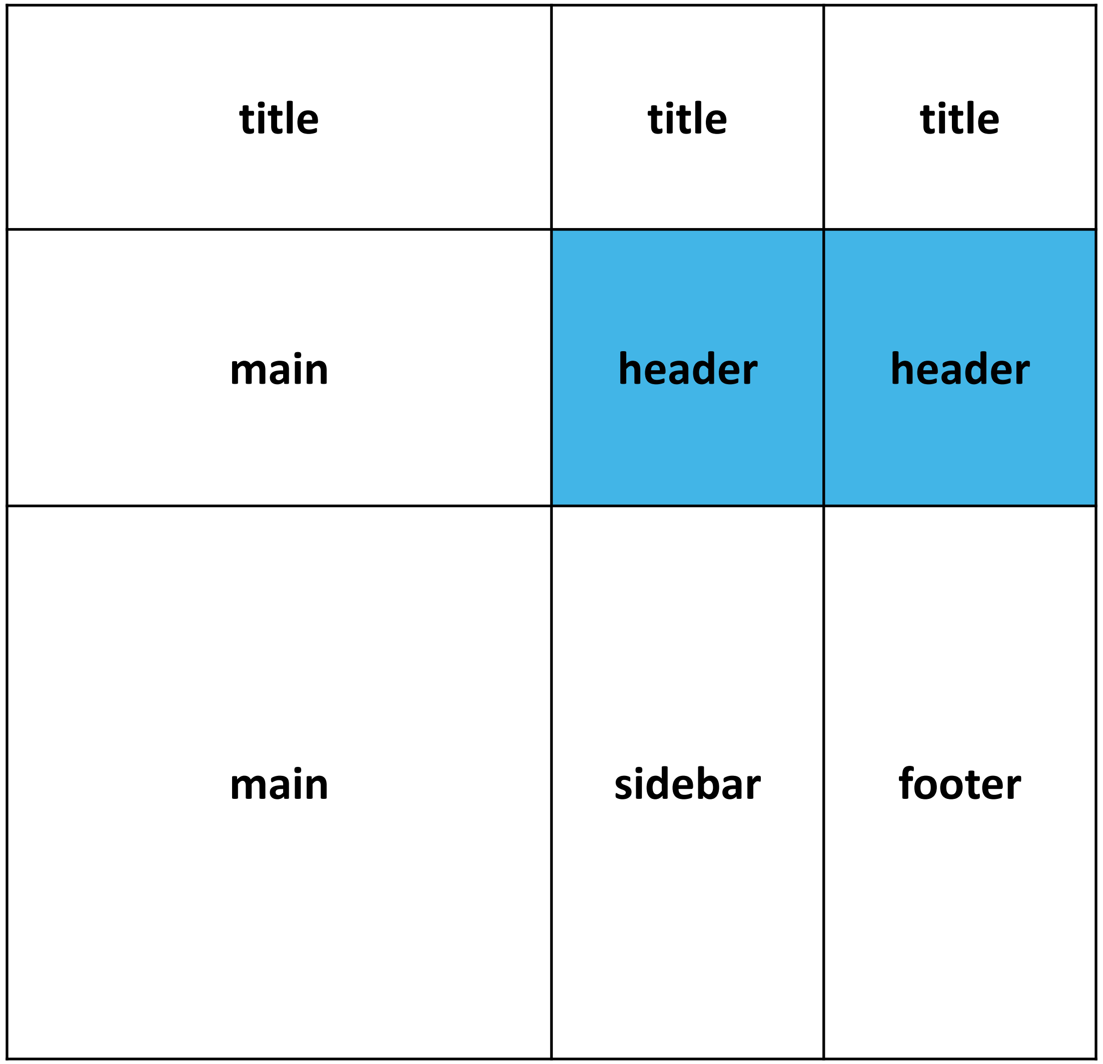
```
.page {
    display: grid;
    grid-template-columns: 2fr 1fr 1fr;
    grid-template-rows: auto 1fr 3fr;
    grid-template-areas:
        "title title title"
        "main header header"
        "main sidebar footer";
}
```

title	title	title
main	header	header
main	sidebar	footer

Grid Template areas

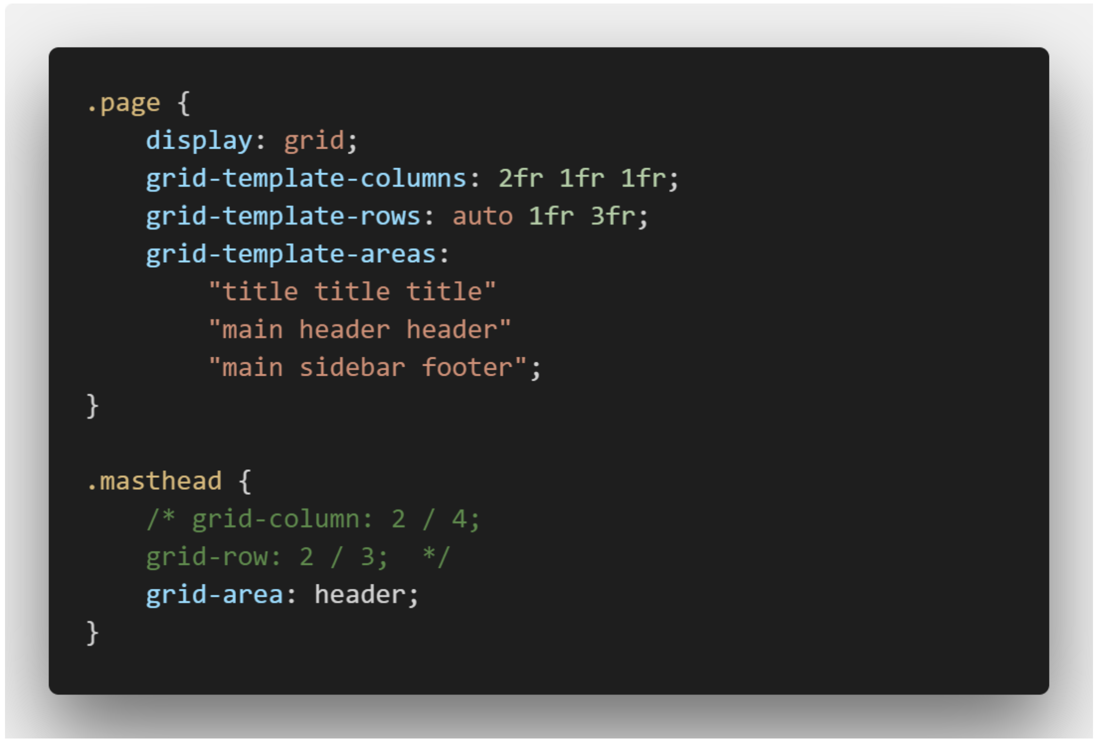


```
.page {
    display: grid;
    grid-template-columns: 2fr 1fr 1fr;
    grid-template-rows: auto 1fr 3fr;
    grid-template-areas:
        "title title title"
        "main header header"
        "main sidebar footer";
}

.masthead {
    /* grid-column: 2 / 4;
    grid-row: 2 / 3;  */
    grid-area: header;
}
```



Exercise

Rewrite the previous grid to use Grid Template areas instead of Grid row and column line definitions.

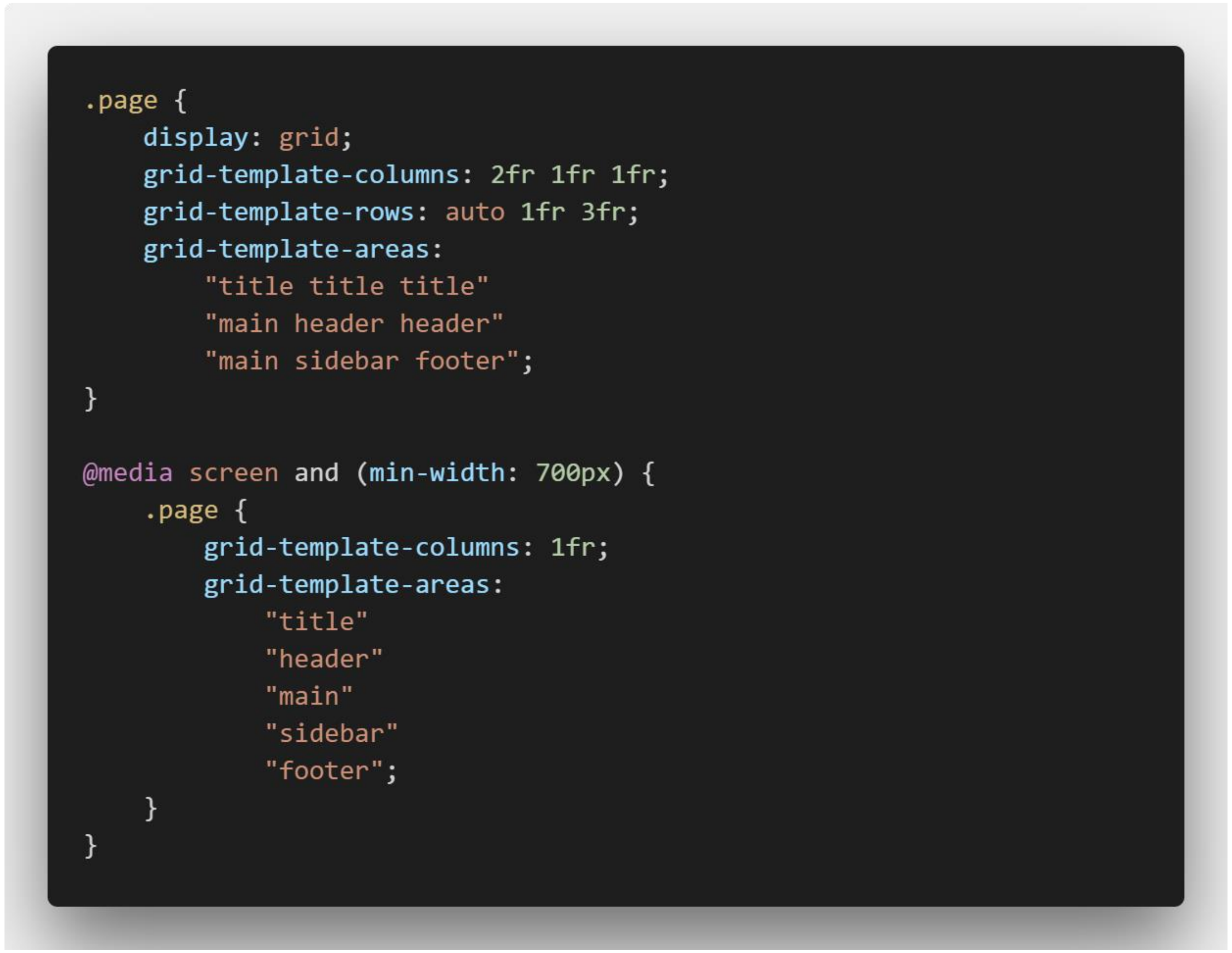


```css
.page {
    display: grid;
    grid-template-columns: 2fr 1fr 1fr;
    grid-template-rows: auto 1fr 3fr;
    grid-template-areas:
        "title title title"
        "main header header"
        "main sidebar footer";
}

.masthead {
    /* grid-column: 2 / 4;
    grid-row: 2 / 3;  */
    grid-area: header;
}
```

BUSINESS
Issue 764
Economy of the European Union
World Bank's stock at all-time high
US rate rise chances

Responsive web design using Grid

Media queries



```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";
}

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

Nesting

Nested Grids

Grids are not inherited

Therefore we create grids within grids

This is called **nested grids**

Questions?