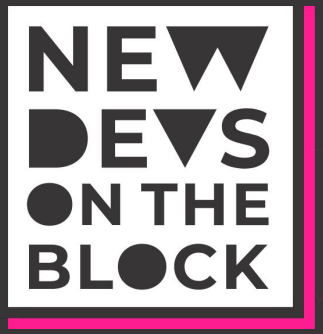
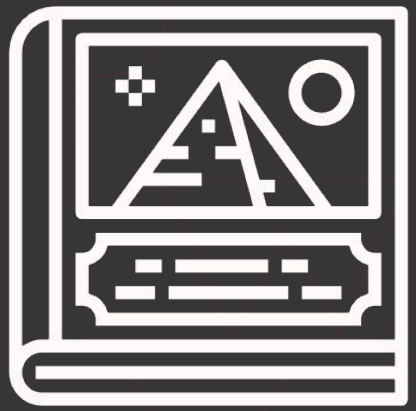


New Devs On The Block

Day 1: Build your first website

Structure

- This workshop is structured into
  - chunks of theory
  - and sprints
- Don’t hesitate to ask questions any time
  - Yes anytime
  - All questions are valid



Short history of the web

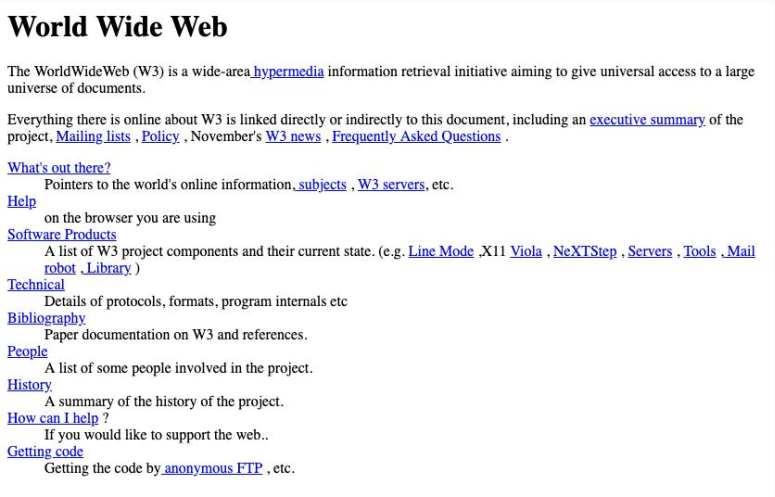
The birth of HTML and the Web

Tim Berners-Lee was working at CERN and developed the initial version of the web.

- First prototype was named ENQUIRE, a system share documents within CERN
- First proposal of HTML in 1989
- First implementation of HTML, browser and server software in 1990

The first website ever

Browser version of the first website



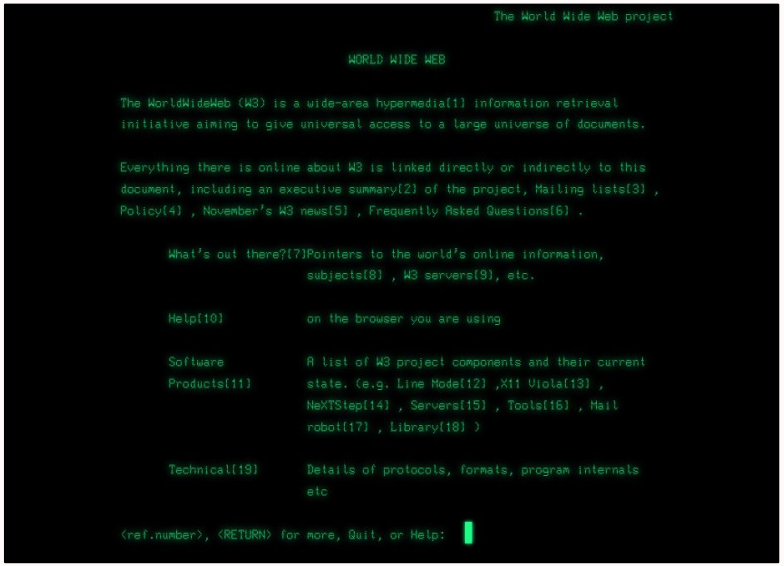
World Wide Web

The WorldWideWeb (W3) is a wide-area hypermedia information retrieval initiative aiming to give universal access to a large universe of documents.

Everything there is online about W3 is linked directly or indirectly to this document, including an executive summary of the project, Mailing lists , Policy , November's W3 news , Frequently Asked Questions .

What's out there?
Pointers to the world's online information, subjects , W3 servers, etc.
Help
on the browser you are using
Software Products
A list of W3 project components and their current state. (e.g. Line Mode ,X11 Viola , NeXTStep , Servers , Tools , Mail robot , Library)
Technical
Details of protocols, formats, program internals etc
Bibliography
Paper documentation on W3 and references.
People
A list of some people involved in the project.
History
A summary of the history of the project.
How can I help ?
If you would like to support the web..
Getting code
Getting the code by anonymous FTP , etc.

Line mode version of the first website



The Internet

HTML Versions

- November 03, 1992: HTML (without version number)
- November 24, 1995: HTML 2.0
- January 14, 1997: HTML 3.2
- December 18, 1997: HTML 4.0
- December 24, 1999: HTML 4.01
- October 28, 2014: HTML5
- November 1, 2016: HTML 5.1
- December 14, 2017: HTML 5.2

Cool new things in HTML5

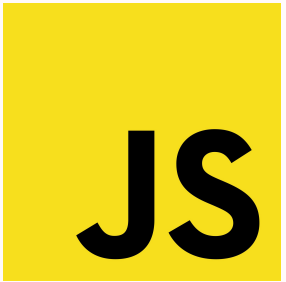
- Audio
- Video
- Canvas
- Semantic tags
- Form validations

The front-end languages

Languages of the web

To build a website, you can combine three languages

- HTML (Hyper Text Markup Language)
- CSS (Cascading Style Sheets)
- JavaScript
- All of them are important (don't let anyone tell you something else)



HTML

HTML defines the structure and the content of a website.

- Basic layer of a website
- Content is relevant for SEO
- Content (in a structured way) is needed for screen readers
- Remember: the original idea was to share information, not cat gifs 😺

CSS

CSS is the like the topping on a cake, it makes it pretty.

- Adds styling and prettiness to a website
- It allows animations
- Better content readability

JavaScript

JavaScript adds interactivity to a website.

- Dynamic content
- Interaction
- Interaction between the browser and the user
- JavaScript turns static websites into apps

The front-end cake

HTML fundamentals

HTML tags

- HTML is organized with tags
- HTML tags are also called *HTML elements*
- Each tag holds content
- Content can be text, images and other tags

<code>

```
<h1>My Website</h1>

<p>I like cats 😸</p>

<footer>Made with frontpage</footer>
```

HTML comment

- You can add comments to your HTML
- A comment won't be displayed in any means, it's just a developer thing

<code>

```
<!-- Don't mind me, I am a comment -->
```

HTML syntax

- Elements have an opening and a closing tag
- Some elements only have an opening tag
  - break tag (`<br>`)
  - image tag (`<img>`)
- Elements can have attributes with values
  - id attribute
  - href attribute

<code>

```
<!-- Opening and closing tag -->
<header>...</header>


<!-- Elements can have attributes -->
<button id="btn">...</button>

<!-- This will create a line break -->
<br>

<!-- Anchor tag with href attribute -->
<a href="https://newdevs.org>...</a>
```

HTML skeleton

- DOCTYPE defines the version of HTML
- <html> is the top-level element of a HTML document
    - The only elements allowed are the head and body element
- <head> holds general information about the website
- <body> represents the actual content of the website

<code>

```
<!DOCTYPE html> <!-- HTML5 -->
<html lang="en">
    <head>
        <!-- Metadata, e.g.
             - Author and description
             - Title
             - Encoding
        -->
    </head>
    <body>
        <!-- Website content -->
    </body>
</html>
```

HTML head

<code>

```
<head>
    <!-- Title of the website (displayed in the browsers tab) -->
    <title>caaaaaats!</title>

    <!-- Site description (will be displayed in google search results) -->
    <meta name="description" content="the best cats on the internet">
</head>
```

HTML head

- <title> holds the websites title
- <meta> tags have a name and content attribute
  - predefined names (e.g. description or viewport)
  - can hold arbitrary information (e.g. google-site-verification)
- <link> tag can associate a stylesheet file
- <style> tag can contain inline styles
- <script> tag links a JavaScript file

Semantic and generic tags

- A tag should describe its content
- Important for accessibility
- Semantic tags on MDN
- There are generic tags for general use
- HTML semantics cheat sheet

Headings

- Headings are represented with 6 hierarchical elements
  - `<h1>`
  - `<h2>`
  - `<h3>`
  - `<h4>`
  - `<h5>`
  - `<h6>`
- Avoid skipping heading levels
- Use `<h1>` only once per page
- MDN article on headings

<code>

```
<h1>Main title</h1>

<h2>Subtitle</h2>

<h3>Section title</h3>

<h4>Section subtitle</h4>
```

Document structure

- <header> usually contains a navigation, page title and a logo
- <nav> groups navigation links
- <aside> represents secondary content
- <main> holds the primary content
- <footer> usually holds copyright, contact and imprint information

<code>

```
<body>
    <header>
        <nav></nav>
    </header>

    <aside></aside>

    <main></main>

    <footer></footer>
</body>
```

Document structure

- <article> holds independent content
  - makes sense on its own if moved somewhere else
  - should have its own heading
- <section> represents a group of related content
  - structures a blogpost into different sections with its own headings

<code>

```
<main>
    <article>
        <h1>Cats</h1>
        <section>
            <h2>History of cats</h2>
        </section>
        <section>
            <h2>Cat Names</h2>
        </section>
    <article>

    <article>
        <h1>Dogs</h1>
        <section>
            <h2>Dogs on TV</h2>
        </section>
    <article>
</main>
```

Text

- <p> represents a generic paragraph of text
- marks text as highly emphasized and important
- marks text as emphasized and slightly more important
- <i> defines a technical term, a thought or phrase
    - can be combined with a lang attribute

<code>

```
<p>

    <em>Cats</em> are the
    <strong>best</strong> animals in
    the world.

    <i lang="el">Gata</i> (greek for
    cat) were already popular in greek
    mythology.

<p>
```

Text Example

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-wi
</head>
<body>
<p>
    <em>Cats</em> are the <strong>best</strong>
    animals in the world.

    <i lang="el">Gata</i> (greek for cat)
    were already popular in greek mythology.
<p>
</body>
</html>
```

Cats are the **best** animals in the world. *Gata* (greek for cat) were already popular in greek mythology.

Lists

- Lists are represented with the tags `<ul>` and `<ol>`
- `<ul>` stands for unordered list
- `<ol>` stands for ordered list
- Both lists contain `<li>` tags, which stands for list item
- Example


```
<ul>
    <li>Banana</li>
    <li>Apple</li>
    <li>Avocado</li>
</ul>

<ol>
    <li>Learn HTML</li>
    <li>Learn CSS</li>
    <li>Profit! 💰</li>
</ol>
```

Anchor tag

- The anchor tag `<a>` creates hyperlinks to
  - other web pages
  - files
  - email addresses
  - locations within the same page by referencing the elements id
- The href attribute sets destination that the hyperlink points to

<code>

```
<a href="newdevs.org">newdevs.org</a>

<a href="mailto:santa@clause.com">Mail
Santa</a>

<a href="#footer">Go to footer</a>

<!-- more content ... -->

<footer id="footer">...</footer>
```


- The image tag <img> embeds an image into the document
- Use the alt tag to add an alternative text description (in case the image does not load and for screen readers)
- Images will initially displayed with its original size

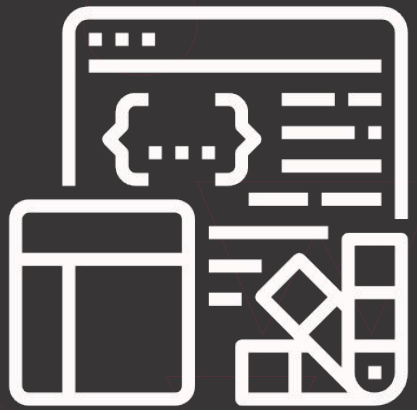
<code>

```
<img src="https://www.fillmurray.com/400/400"
alt="A picture showing actor Bill Murray">
```

Sprint 1 - Setup and structure

Sprint 1: Setup and structure

- Go to CodeSandbox
  - Login into CodeSandbox using your GitHub account
  - Setup your workspace
- Play around with the tags and create the basic structure and content for your website
  - Header with a site title and links to your content
  - Main content with articles, lists and images
  - Footer with some social links
- Add information to the header of your website
  - title and description



CSS fundamentals

Default styling

- The browser will apply a default styling to a website
- This default styling is called *Default Stylesheet*
- This default styling may vary a little between different browsers

HTML

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width">
</head>
<body>
<h1>My Website</h1>
<p>I like cats 🐱</p>
<footer>Made with frontpage</footer>
</body>
</html>
```

Output

My Website

I like cats 🐱

Made with frontpage

Why CSS?

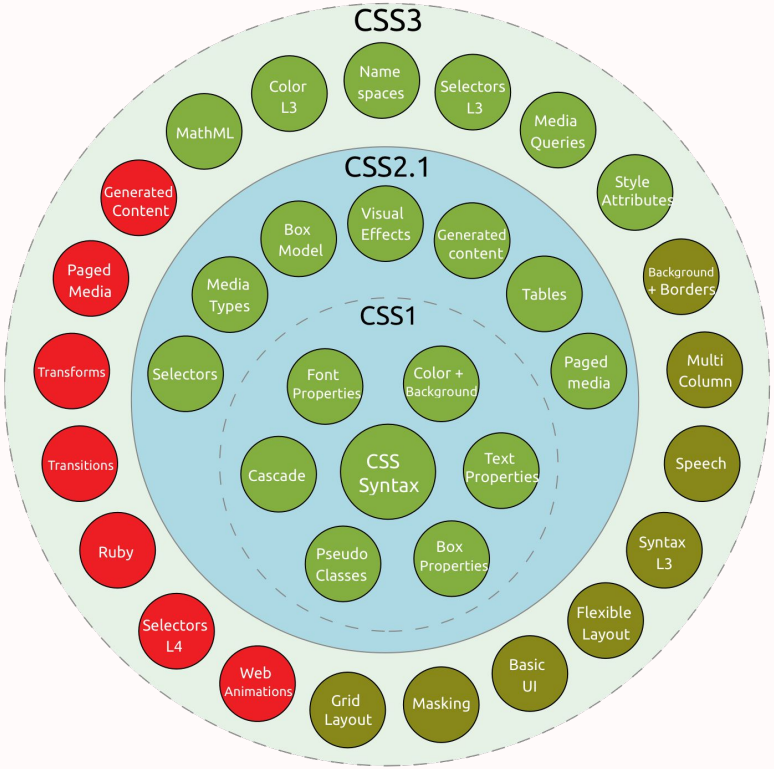
- HTML only defines structure and content
- Default browser styling is quite pure 🧐
- Separation of concerns
  - HTML for content
  - CSS for styling and visuals

CSS Version History

- December 1996: CSS Level 1
- May 1998: CSS Level 2
- June 1998: First drafts for CSS Level 3 modules
- There will be no CSS version 4, but single CSS modules with that version number

CSS Modules

- Examples for CSS Modules
  - Visual effects
  - Color + Background
  - Selectors
  - Media Queries
  - Grid Layout



Add CSS to your HTML

- CSS can be loaded using the `<link>` tag
- CSS can be written directly inside a `<style>` tag inside the `<head>` element
- CSS can be added using the style attribute on each HTML tag

<code>

```
<head>

  <!-- CSS as external resource -->
  <link rel="stylesheet" href="style.css">

  <!-- CSS in style tag -->
  <style>
    body { background-color: hotpink; }
  </style>

</head>
<body style="background-color: khaki;">
</body>
```

CSS ruleset

- CSS is defined by rulesets
- Each ruleset consists of
  - a selector
  - style declaration
- Each style declaration is made of
  - a css property
  - a value

<code>

```
/* selector */
body {

    /* ---- declaration --- */
    background-color: hotpink;
    /*      property: value */
}
```

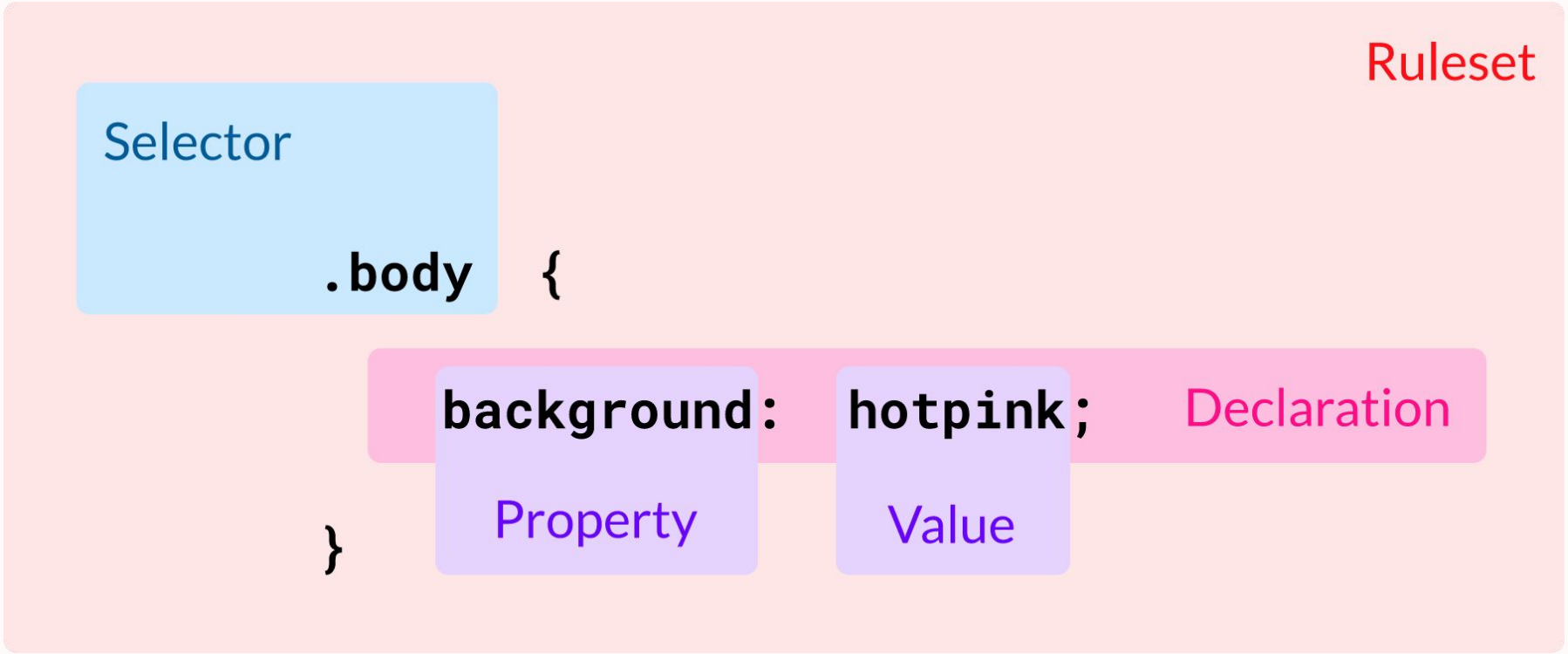
CSS ruleset

- selector → who?
- property → what?
- value → how?

<code>

```
/* Technical terms */
selector {
    property: value;
}

/* Easy to remember */
who {
    what: how;
}
```

CSS ruleset


Ruleset
Selector
.body {
background:
hotpink;
Declaration
Property
Value
}

CSS Selectors

Tag Selector

- Styles all elements with a specific tag name

<code>

```
/* CSS */
p {
    background-color: hotpink;
}

<!-- HTML -->
<body>
    <header>...</header>
    <h1>...</h1>
    <p>...</p>
    <div>...</div>
</body>
```

ID Selector

- Styles the element with a specific id
- Starting with a hash (#) followed by the name of the id
- ID is unique (or at least should be)

<code>

```
/* CSS */
#foo {
    background-color: hotpink;
}

<!-- HTML -->
<body>
    <header>...</header>
    <h1>...</h1>
    <p>...</p>
    <div id="foo">...</div>
</body>
```

Class Selector

- Styles all elements that have a specific class
- Starting with a dot (.) followed by the class name

<code>

```
/* CSS */
.pinky {
    background-color: hotpink;
}

<!-- HTML -->
<body>
    <header class="pinky">...</header>
    <h1>...</h1>
    <p class="example">...</p>
    <div class="pinky cats">...</div>
</body>
```

Class Selector

- CSS class names can be totally arbitrary
- All alphanumeric characters allowed
- Must not start with a number
- Can contain hyphens and dashes
- See W3C Spec for further details

<code>

```
/* cool */
.header { ... }

/* works, but please don't */
.muahaha--___111 { ... }

/* Still works, but spec says it shouldn't */
.---- { ... }

/* Wrong! Completely different semantic */
.header bla { ... }
```

Attribute Selector

- Styles all elements with a specific attribute
- Can be combined with a
  - value
  - value pattern
    - starts with
    - ends with
    - contains
- Name of attribute is enclosed with square brackets

<code>

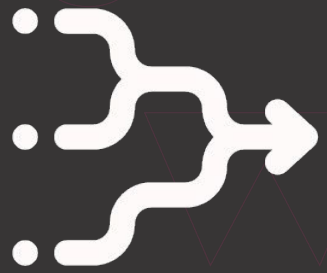
```
/* CSS */
[title] {
    background-color: hotpink;
}

<!-- HTML -->
<body>
    <header>...</header>
    <h1>...</h1>
    <p>...</p>
    <div title="Hey!">...</div>
</body>
```

Universal Selector

- Styles all elements in the document
- Useful for resets

<code>

```
/* CSS */
* {
    background-color: hotpink;
}

<!-- HTML -->
<body>
    <header>...</header>
    <h1>...</h1>
    <p>...</p>
    <div>...</div>
</body>
```



CSS Combinators & Selector Lists

Selector Lists

- One ruleset can have multiple selectors
- Multiple selectors are separated with commas


```
main,
article,
.fancy {
    background-color: hotpink;
}

p, div, [title] {
    background-color: salmon;
}
```

Descendant Combinator

- Syntax: selector1 selector2
- Two selectors separated by a space
- Matches all elements for selector2 that are descendants (child, grandchild, ...) of selector1

<code>

```
/* CSS */
main div {
    background-color: hotpink;
}

<!-- HTML -->
<main>
    <article>
        <div>...</div>
    </article>
    <div>...</div>
</main>
<div>...</div>
```

Child Combinator

- Syntax: selector1 > selector2
- Matches all elements for selector2 that are children of selector1

<code>

```
/* CSS */
main > div {
    background-color: hotpink;
}

<!-- HTML -->
<main>
    <article>
        <div>...</div>
    </article>
    <div>...</div>
</main>
```

Adjacent Sibling Combinator

- Syntax: selector1 + selector2
- Matches all elements for selector2 that is the next sibling of selector1

<code>

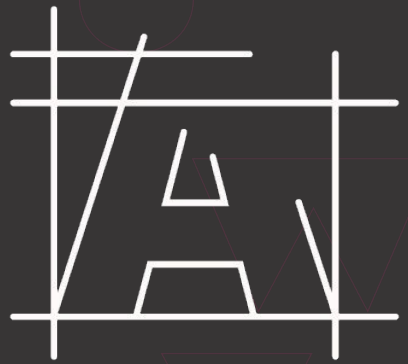
```
/* CSS */
article + div {
    background-color: hotpink;
}

<!-- HTML -->
<main>
    <article>
        <div>...</div>
    </article>
    <div>...</div>
    <div>...</div>
</main>
```

General Sibling Combinator

- Syntax: selector1 ~ selector2
- Matches all elements for selector2 that are siblings of selector1

<code>

```
/* CSS */
article ~ div {
    background-color: hotpink;
}

<!-- HTML -->
<article>
    <div>...</div>
</article>
<div>...</div>
<p>...</p>
<div>...</div>
```



Fonts

Setting the font family

- The font can be set with the font-family property
- font-family takes a specific font name and/or a generic family name
- If you set the font-family on the body, it will be inherited on all elements inside the body
    - unless you set a font-family directly with another selector

<code>

```
/* Use Arial, unless it's not available */
body {
    font-family: 'Arial', sans-serif;
}

/* Use any serif font */
p {
    font-family: serif;
}
```

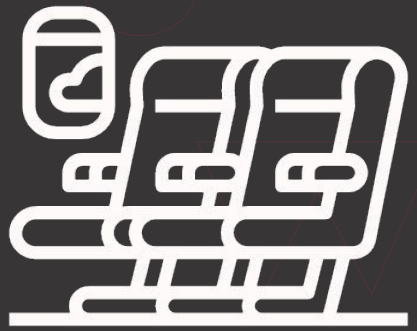
Google fonts

- Google Fonts offers a lot of fancy fonts, that can be linked directly in your HTML
- We will use Google Fonts in this workshop

Sprint 2: Fonts

Sprint 2: Fonts

- Go to Google Fonts and pick fonts for your website
- Follow the instructions on Google Fonts on how to add them to your website
- Pick at least two different fonts
  - One for the headings
  - One for regular text



Pseudo classes

Pseudo Classes & Elements

- A pseudo class or element is a keyword added to a selector
- Pseudo classes describe a **special state** of an element
- Pseudo elements describe a **special part** of an element

<code>

```
selector:pseudo-class {
    ...
}

selector::pseudo-element {
    ...
}
```

:link, :visited, :hover and :active

- :link → for links a user did not visit yet
- :visited → for links a user already visited
- :hover → you put the mouse over it
- :active → element is activated by the user
- Example


```
a:link {
    background-color: gold;
}

a:visited {
    background-color: lightblue;
}

a:hover {
    background-color: hotpink;
}

a:active {
    background-color: firebrick;
}
```

Other useful pseudo classes and elements

- :empty → no content
- :not(selector)
- :only-child → Einzelkind-Selector 🤷‍♀️
- ::selection → Text selection
- Complete list of pseudo classes on MDN
- Complete list of pseudo elements on MDN

Colors

Color names and values

- Colors are represented with
  - color names
  - hex color values
  - rgb values
  - hsl values
- Color picker for hex, rgb, hsl
- List of available color names
  - Best colors
    - hotpink
    - salmon
    - papayawhip
    - rebeccapurple

<code>

```
.make-it-pretty {

    /* color name */
    color: hotpink;

    /* hex color value */
    color: #FF0000;

    /* rgb */
    color: rgb(0,255,0);

    /* hsl */
    color: hsl(30, 100%, 50%);

}
```

(text) color and background-color

- the color property sets the text color of an element
- the background-color property sets the background of an element

<code>

```
header {
    background-color: salmon;
    color: gray;
}

footer {
    background-color: #1E90FF;
    color: #333333;
}
```

Colorful links

- Use the pseudo class :hover for colorful links when you hover over it
- Example


```
a {
    background-color: orange;
    color: black;
}

a:hover {
    background-color: #333;
    color: #fff;
}
```

Fancy text selection

- Use the ::selection pseudo element to style selected text
- Example

<code>

```
p::selection {
    background-color: hotpink;
    color: black;
}
```

Colorful borders

- Use the border property shorthand to create colorful element decoration
- Use the different border styles
  - solid
  - dotted
  - dashed
- Example

<code>

```
<!-- HTML -->
<h1 class="solid">OMG solid!</h1>
<h1 class="dotted">OMG dotted!</h1>
<h1 class="dashed">OMG dashed!</h1>


/* CSS */
.solid {
    border-bottom: 3px solid hotpink;
}

.dotted {
    border-bottom: 3px dotted hotpink;
}

.dashed {
    border-bottom: 3px dashed hotpink;
}
```

box-shadow

- Use the box-shadow property to create shadows for your elements
- The first two parameters of box-shadow define the shadow offset on the x and y axis
- The third parameter defines the blur radius
- Example
- Box-shadow generator
- box-shadow property on MDN

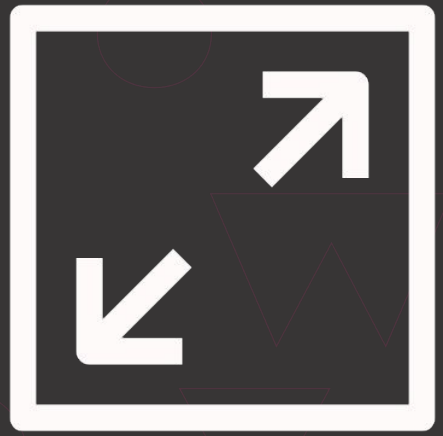
<code>

```
/* CSS */
.shadow {
  background-color: #333;
  padding: 10px;
  color: #fff;
  box-shadow: 5px 5px 0px hotpink;
}


<!-- HTML -->
<h1 class="shadow">Look at this
shadow!</h1>
```

Sprint 3: Colors and Links

Sprint 3: Colors and links

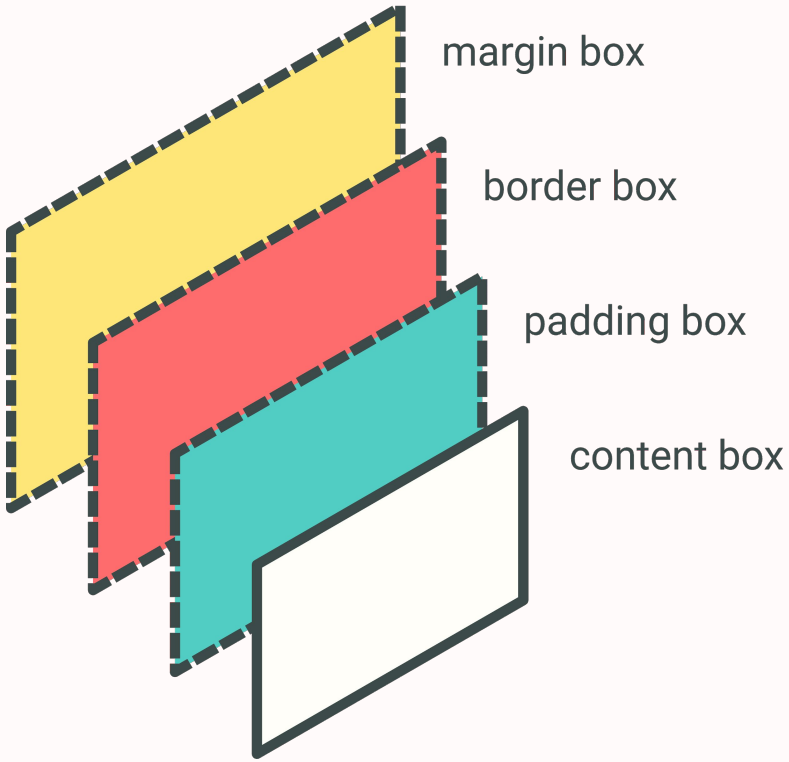
- Add some color to your website
  - Use pseudo classes to make some colorful links
  - Use borders, backgrounds and/or box-shadows to spice up your elements
  - Use the ::selection pseudo element to customize the appearance of selected text



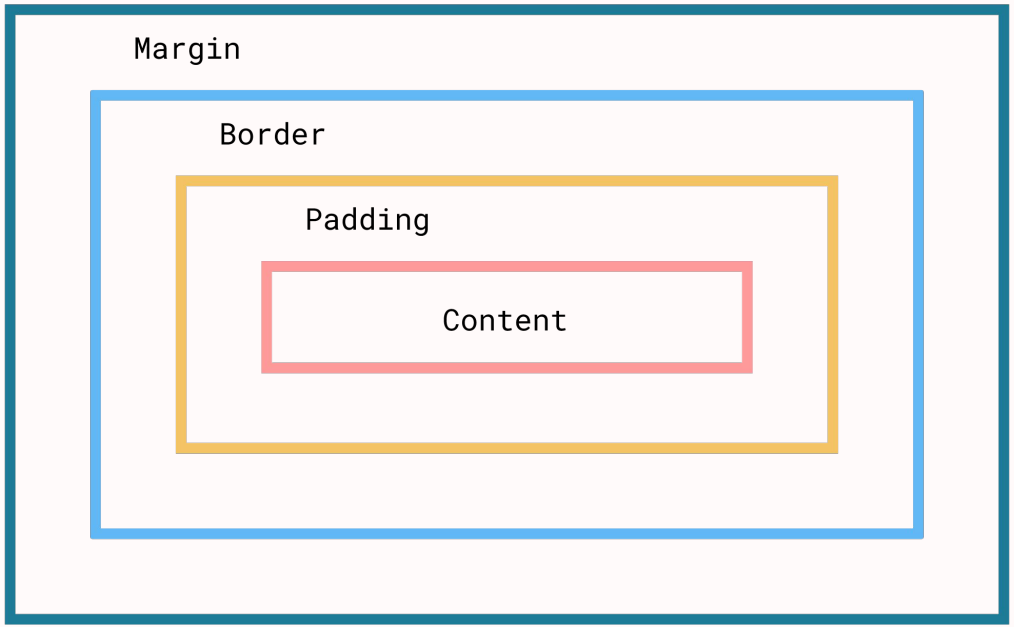
Box Model and Box Sizing

Box Model

- Every element is defined by the box model
- The box model is like the blueprint of the web
- Each element is made out of the box's
  - content
  - padding
  - border
  - margin
- MDN article about the box model



Box Model



Box model properties

- the margin property sets the elements outer margin
- the border property sets the elements border
- the padding property sets the elements inner padding
- the width property sets width of the element
    - yes, no, it's complicated… we will come to that soon 😅
- It's all about spacing 🥳
- Example

<code>

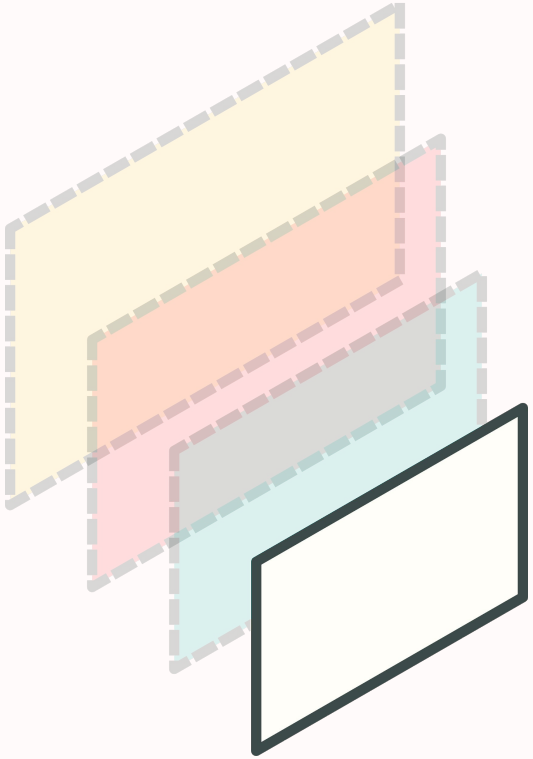
```
.box {
    margin: 50px;
    border: 10px solid black;
    padding: 20px;
}
```

Sizing of elements

- There are two different ways to calculate the size of an element
- The calculation is based on the box-sizing property
- The two possible values and calculations are
  - content-box (default)
  - border-box

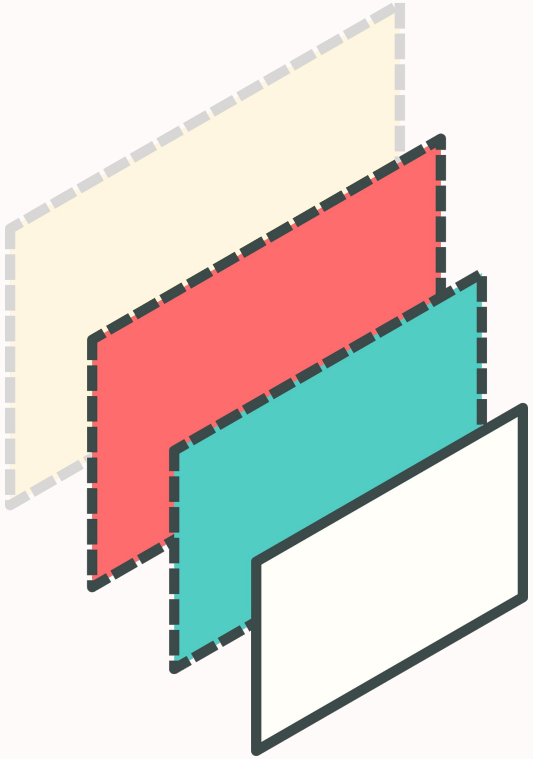
box-sizing: content-box

- Width and height property only set the size of the elements content-box
- Values for padding and border are added to the total dimensions
- Feels quirky, but is the default behavior

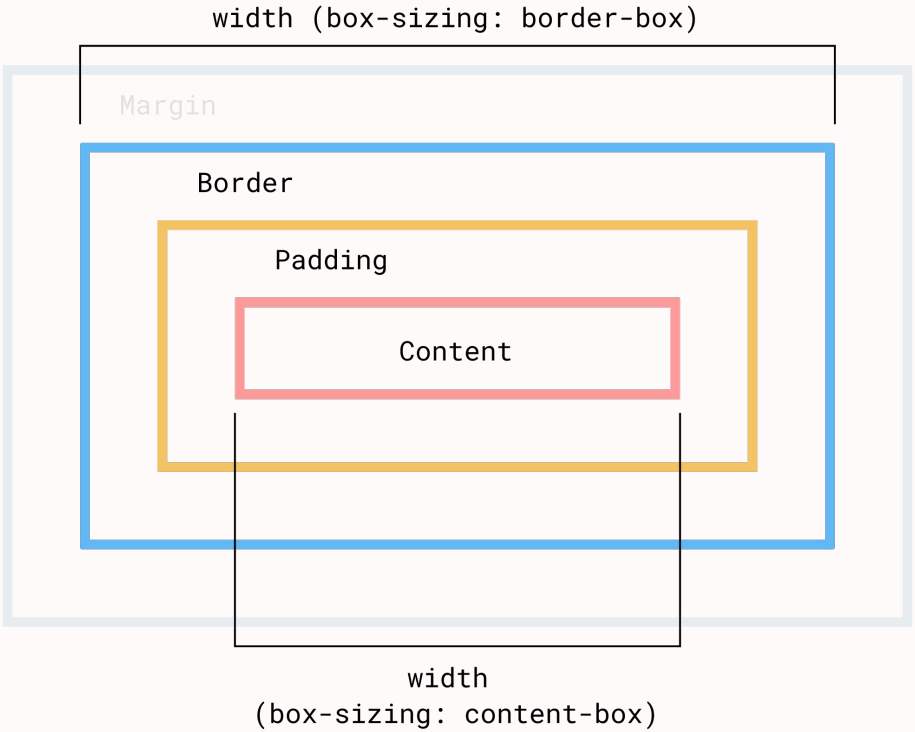


box-sizing: border-box

- Width and height properties define the dimensions of the element including values for padding and border
- Feels more intuitive



box-sizing


width (box-sizing: border-box)
Margin
Border
Padding
Content
width
(box-sizing: content-box)

Box sizing example

- Example

<code>

```
<!-- HTML -->
<div>Whaat?😳</div>
<div class="border-box">Oha!😘</div>


/* CSS */
div {
    margin: 20px;
    border: 10px solid black;
    padding: 40px;
    width: 200px;
}

.border-box {
    box-sizing: border-box;
}
```

padding, margin & border shorthands

- padding, margin & border property consists of four single properties for each side of the element (top, right, bottom, left)
- Padding, margin and border can be set for each side or with the shorthand

<code>

```
padding: 10px;
/*
padding-top: 10px;
padding-right: 10px;
padding-bottom: 10px;
padding-left: 10px;
*/

margin: 0 10px;
/*
margin-top: 0;
margin-right: 10px;
margin-bottom: 0;
margin-left: 10px;
*/
```

Display Property and Flow Layout

Display property

- The display property defines how an element participates in the layout process
- Each tag has a default display property
- List of all display types on MDN

display: block

- creates a new line before and after the element
- takes up the full width of the parent element by default
- adjusts to content height by default
- width and height can be set
- can contain any elements

display: block

- Most elements are block elements, e.g.
  - <p>
  - <header>
  - <main>
  - <footer>
  -
- List of all block elements on MDN
- Example

<code>

```
/* CSS */
.special-div {
  width: 400px;
  border: 5px solid black;
  padding: 20px;
}

p {
  margin: 10px 0;
  background-color: lightgrey;
}

<!-- HTML -->
<p>Hey there</p>
<div>Whats up!</div>
<div class="special-div">
    <p>Mic check... 1,2,3</p>
    <p>😾😿🙀😹😼😻😹😸😺</p>
</div>
```

display: inline

- width and height properties are ignored
- as big as the elements content
- overflows to a new line
- creates no new lines
- can only contain other inline elements or text

display: inline

- Mostly text and form related elements are inline elements, e.g.
  - `<span>`
  - `<label>`
  - `<strong>`
- List of all inline elements on MDN
- Example

<code>

```
<!-- HTML -->
<span class="big">Ene</span>
<span>Mene</span>
<span>Muh</span>
<div>
    <span>Und raus bist du</span>
</div>


/* CSS */
span {
  background-color: salmon;
}

.big {
  width: 1000px;
  height: 5000px;
}
```

display: inline-block

- behaves like display: inline, but the width and height can be set
- Example

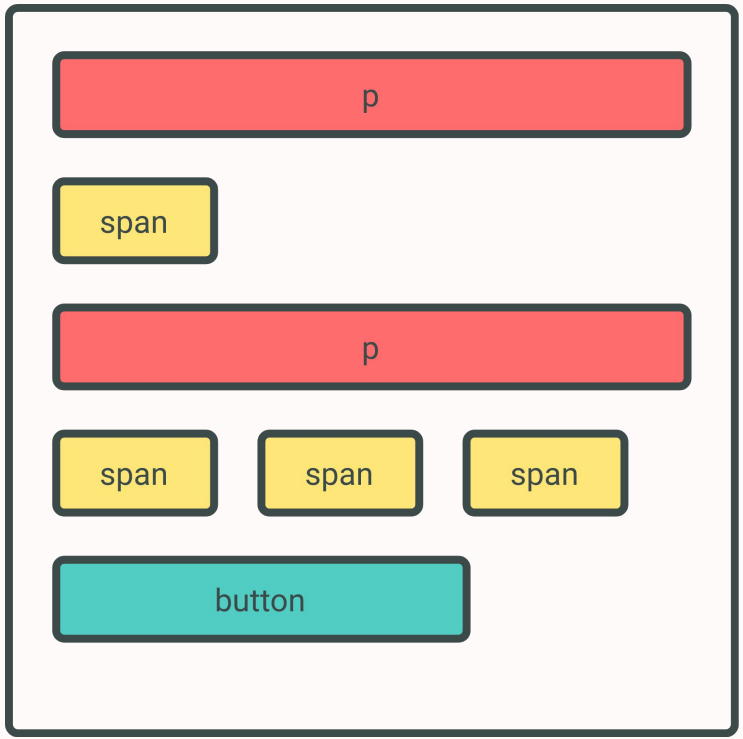
<code>

```
<-- HTML -->
<span>Ene</span>
<span>Mene</span>
<span>Muh</span>


/* CSS */
span {
  display: inline-block;
  width: 100px;
  background-color: papayawhip;
}
```

Normal document flow

- In the normal document flow, all elements are
  - laid out according to their appearance in the document
  - respecting the individual display type
- Example

Normal document flow



<code>

```
<main>
    <p></p>
    <span></span>
    <p></p>
    <span></span>
    <span></span>
    <span></span>
    <button></button>
</main>
```

Position

Position theory

- Defines how an element is positioned
- The top, bottom, left and right properties define the final location of **positioned** elements
- Position values
  - static (default)
  - relative
  - absolute
  - *fixed*
  - *sticky*

position: static

- The element **is not** positioned
- Default position of all elements in the normal document flow
- Top, right, bottom and left properties are ignored
- Example

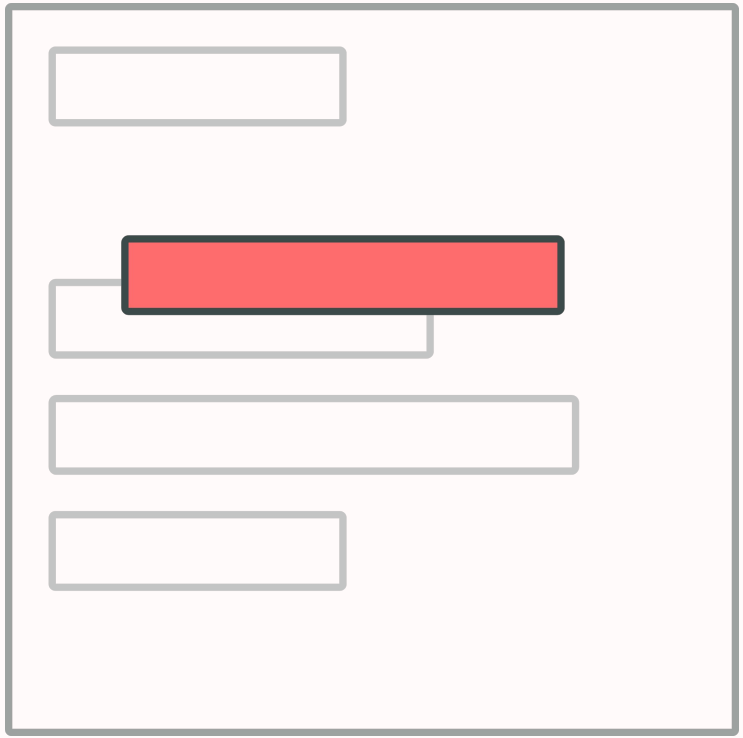
position: static


```
<!-- HTML -->
<main>
  <div class="div1">div 1</div>
  <div class="div2">div 2</div>
  <div class="div3">div 3</div>
  <div class="div4">div 4</div>
  <div class="div5">div 5</div>
</main>


/* CSS */
body, main, div {
  padding: 10px;
}

main,div {
  border: 5px solid black;
  margin: 10px;
}

.div2 {
  position: static;
  top: 50px;
  left: 50px;
  background-color: salmon;
}
```

position: relative

- The element **is** positioned
- Still takes up the original space in the normal document flow
- top **or*** bottom properties set vertical offset
- left **or*** right properties set horizontal offset
- * depends on the current direction (LTR or RTL)
- Example

position: relative



<code>

```
<!-- HTML -->
<main>
  <div class="div1">div 1</div>
  <div class="div2">div 2</div>
  <div class="div3">div 3</div>
  <div class="div4">div 4</div>
  <div class="div5">div 5</div>
</main>


/* CSS */
body, main, div {
  padding: 10px;
}

main,div {
  border: 5px solid black;
  margin: 10px;
}

.div2 {
  position: relative;
  top: 50px;
  left: 50px;
  background-color: salmon;
}
```

position: absolute

- The element **is** positioned
- Behavior of position: absolute depends on
  - the **containing block** of the element
  - weather top, bottom, left or right properties are set

Containing block

- The containing block for absolute positioned elements is the nearest **positioned** parent element
- A positioned element has a position value other than static
- There are some exceptions
  - but for now ignore them 🤷‍♀️

position: absolute (without top, bottom, left or right values)

- Element is removed from the normal document flow
- Stays at its initial position from the normal document flow
- Shrinks to its content size (when no width or height value is set)
- Example

<code>

```
<!-- HTML -->
<main>
  <div class="div1">div 1</div>
  <div class="div2">div 2</div>
  <div class="div3">div 3</div>
  <div class="div4">div 4</div>
  <div class="div5">div 5</div>
</main>

/* CSS */
body, main, div {
  padding: 10px;
}

main,div {
  border: 5px solid black;
  margin: 10px;
}

.div2 {
  position: absolute;
  background-color: salmon;
}
```

position: absolute (without top, bottom, left or right values)

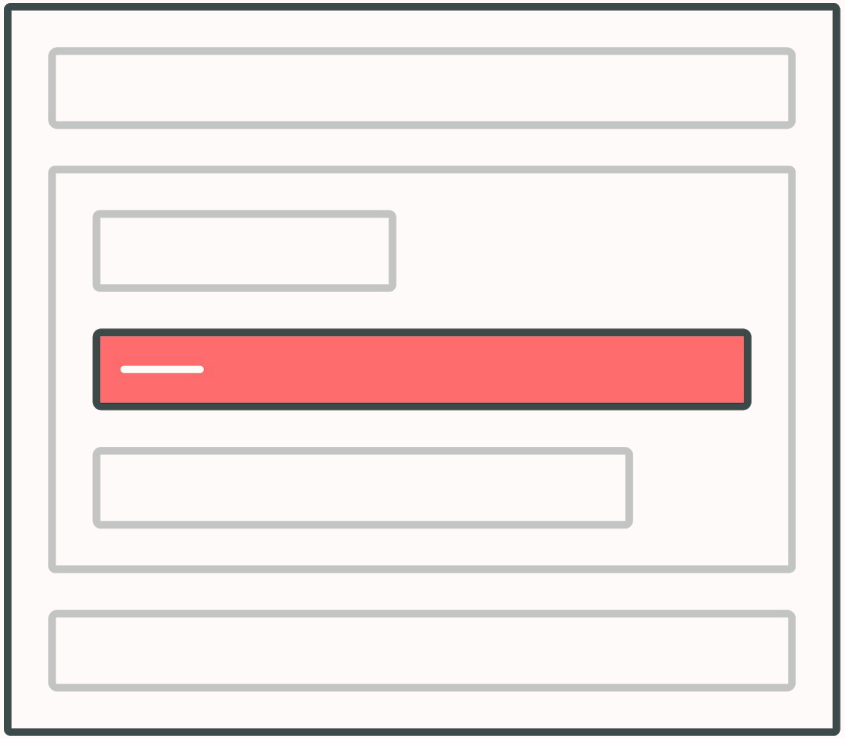
position: absolute (without top: 0px and left: 0px)

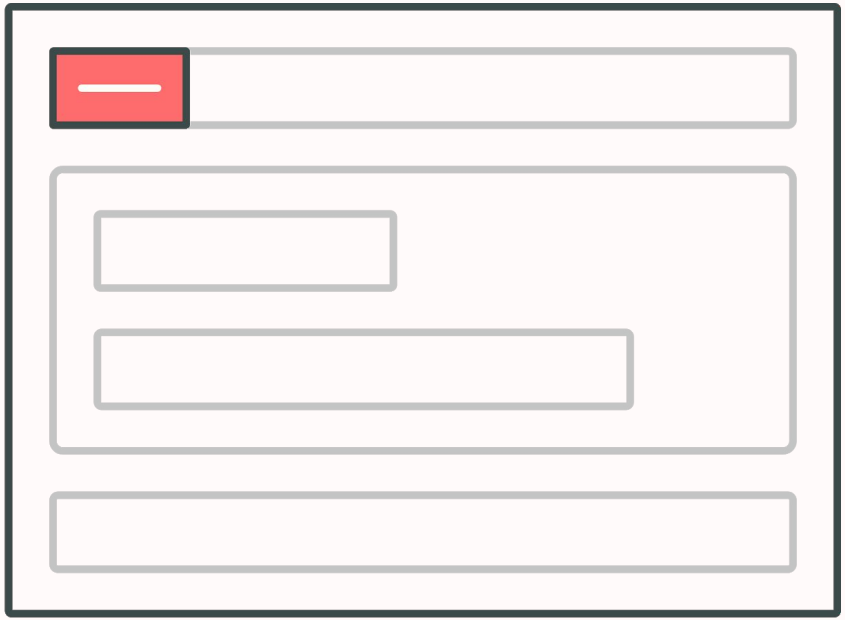
- Element is removed from the normal document flow
- Element is positioned according to the offsets within the padding box of the containing block
- Shrinks to its content size (when no width or height value is set)
- Example

<code>

```
<!-- HTML -->
<main>
  <div class="div1">div 1</div>
  <div class="div2">div 2</div>
  <div class="div3">div 3</div>
  <div class="div4">div 4</div>
  <div class="div5">div 5</div>
</main>

/* CSS */
body, main, div {
  padding: 10px;
}

main,div {
  border: 5px solid black;
  margin: 10px;
}

main {
  position: relative;
}

.div2 {
  position: absolute;
  top: 50px;
  left: 50px;
  background-color: salmon;
}
```

position: absolute (without top: 0px and left: 0px)





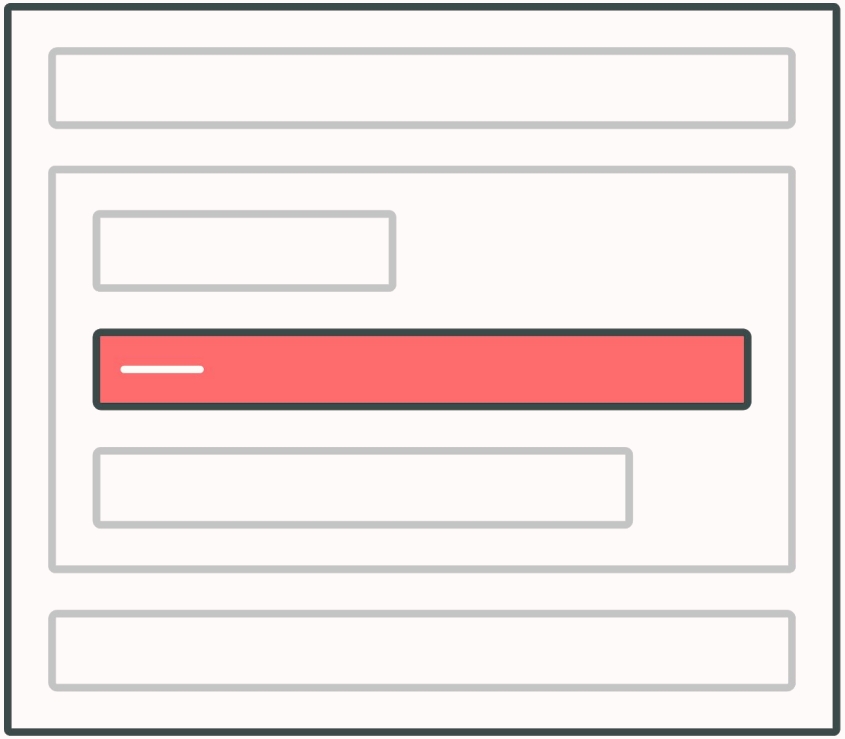
position: absolute (without top: 0px, bottom: 0px, left: 0px and right: 0px)

- Element is removed from the normal document flow
- Element is positioned according to the offsets within the padding box of the containing block
- Example

<code>

```
<!-- HTML -->
<main>
  <div class="div1">div 1</div>
  <div class="div2">div 2</div>
  <div class="div3">div 3</div>
  <div class="div4">div 4</div>
  <div class="div5">div 5</div>
</main>

/* CSS */
body, main, div {
  padding: 10px;
}

main,div {
  border: 5px solid black;
  margin: 10px;
}

main { position: relative; }

.div2 {
  position: absolute;
  top: 0px;
  bottom: 0px;
  left: 0px;
  right: 50px;
  background-color: salmon;
}
```

position: absolute (without top: 0px, bottom: 0px, left: 0px and right: 0px)



background-*

background-image

- use the background-image property to add a background image to an element
- Accepts an url as argument
- By default the background image is repeated, if the dimensions of the image don't match the element
- Example

<code>

```
/* CSS */
header {
  height: 100px;
  background-image:
    url(http://domain.tld/image.png);
}
```

background-size

- Use the background-size property to set the size of the background image
- Useful values for photo backgrounds are
  - cover
  - contain
- background-size: cover will try to cover the whole element
- background-size: contain scales the image as large as possible without cropping or stretching it
- Example

<code>

```
/* CSS */
div {
  width: 500px;
  height: 200px;
  margin: 10px;
  background-image:
url(https://www.fillmurray.com/200/200);
  border: 5px solid hotpink;
}

.cover {
  background-size: cover;
}

.contain {
  background-size: contain;
}

<!-- HTML -->
<div class="cover"></div>
<div class="contain"></div>
```

background-color

- The background-color property sets background color of an element 😂
- Can be used as a backup for a background image (that maybe doesn't load)
- Example

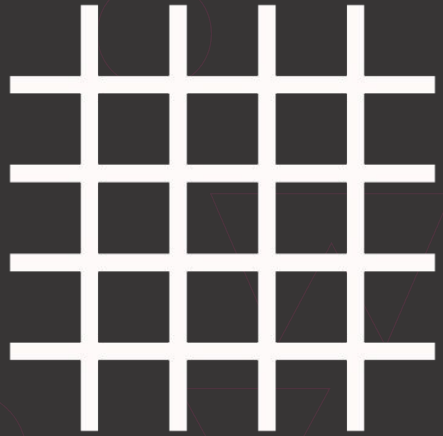
<code>

```
/* CSS */
div {
  width: 500px;
  height: 200px;
  margin: 10px;
  background-color: hotpink;
  background-image:
url(xhttps://www.fillmurray.com/200/200);
  border: 5px solid hotpink;
}

<!-- HTML -->
<div></div>
```

Sprint 4: Spice up your header

Sprint 4: Spice up your header

- Use a fixed height for the websites header
- Use a background image for your header
  - You can find free images on pexels
- Put your logo on the top left
- Put your links on the top right
  - Also make your links (should be ul and li elements) display: inline



CSS Grid

CSS Grid

- Efficient and modern way for building complex layouts
- display: grid defines a grid container
- The element that has display: grid becomes the grid container
- Child elements of the grid container become the grid items

grid-template-columns

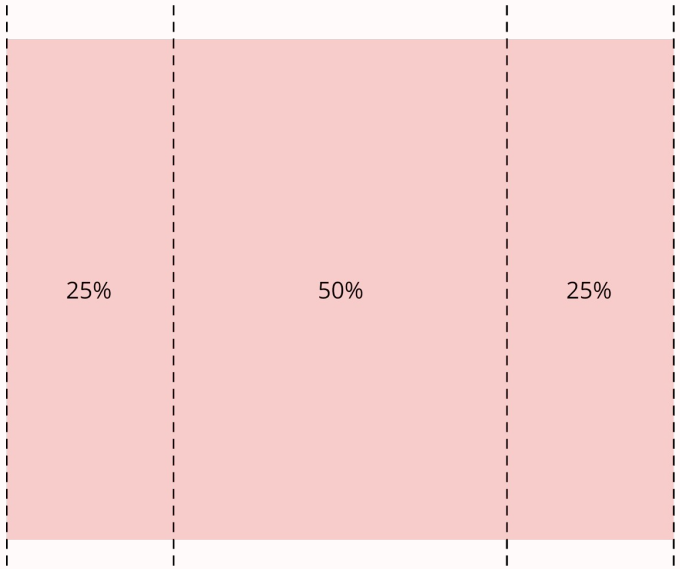
- grid-template-columns property defines the columns of the grid
- Accepts a list of values defining the width of columns
- Possible units
  - Absolute pixels values
  - Percentage values
  - Fractional values
  - auto keyword
- Example

<code>

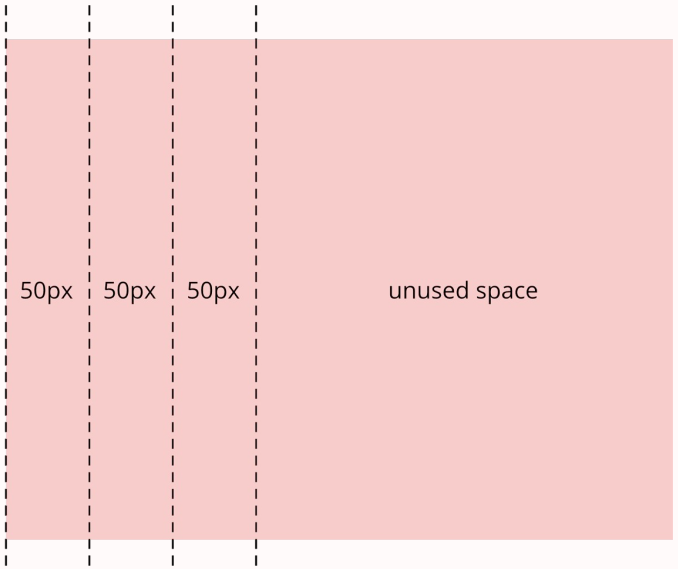
```
/* absolute px values */
.grid {
    display: grid;
    grid-template-columns: 100px 200px 100px;
}

/* percentage values */
.grid-2 {
    display: grid;
    grid-template-columns: 10% 80% 10%;
}
```

grid-template-columns

`grid-template-columns: 25% 50% 25%`

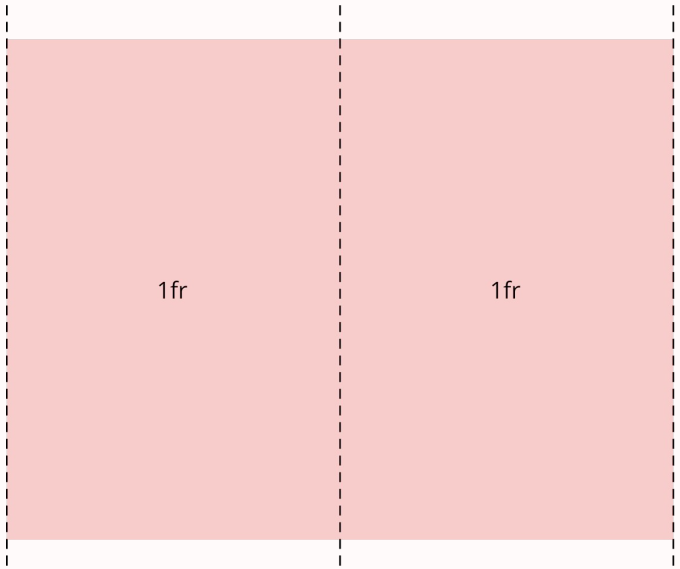


`grid-template-columns: 50px 50px 50px`

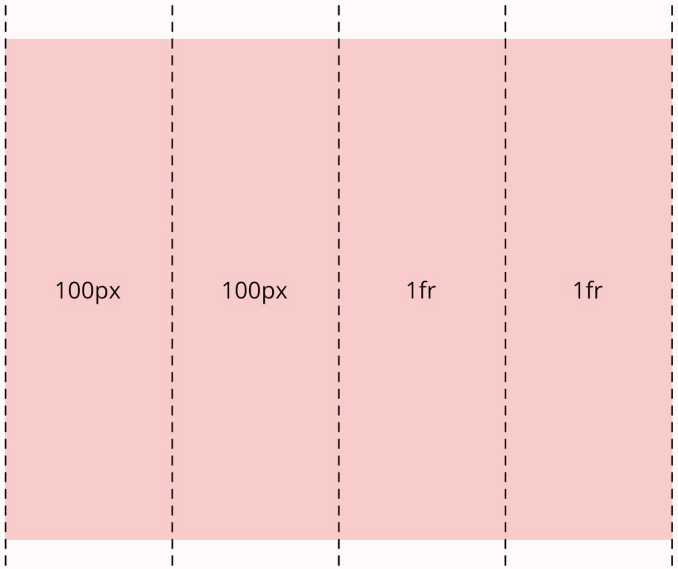


grid-template-columns

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



`grid-template-columns: 100px 100px 1fr 1fr`



grid-template-rows

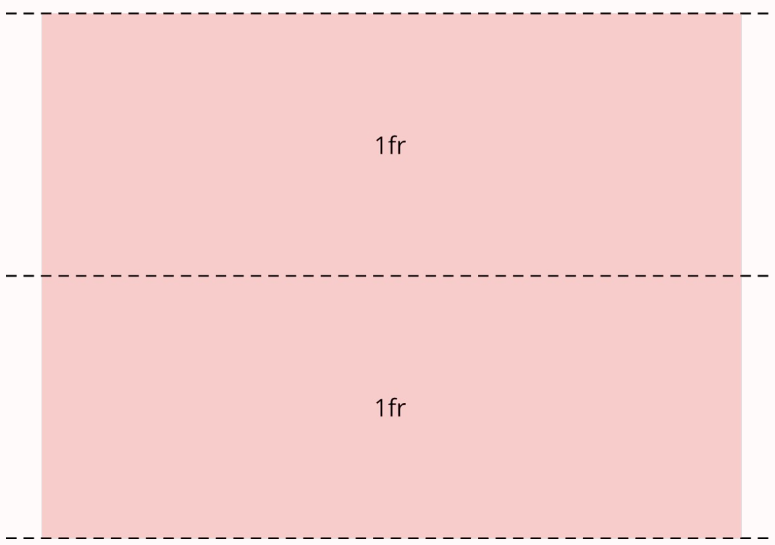
- grid-template-rows defines the rows of a grid
- behaves like grid-template-columns

<code>

```
/* grid-template-rows */
.grid {
    display: grid;
    grid-template-rows: 100px 1fr auto;
}
```

grid-template-rows

`grid-template-rows: 1fr 1fr`



`grid-template-rows: 100px 1fr 50px`

grid-*-gap

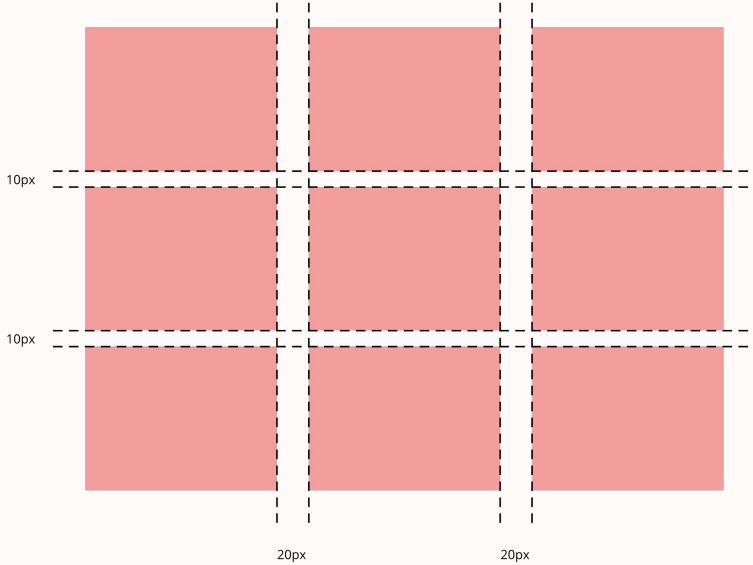
- Grid columns and rows can have a gap
  - Also called gutter
- The grid-column-gap property defines gaps between the columns
- The grid-row-gap property defines gaps between the rows
- Use the grid-gap property shorthand to set both values at once
- Example

<code>

```
.grid {
    display: grid;
     grid-template-columns: 100px 200px 100px;

    grid-column-gap: 10px;
    grid-row-gap: 20px;

    /* grid-gap shorthand */
    grid-gap: 10px 20px;
}
```

grid-gap

grid-gap: 20px 10px

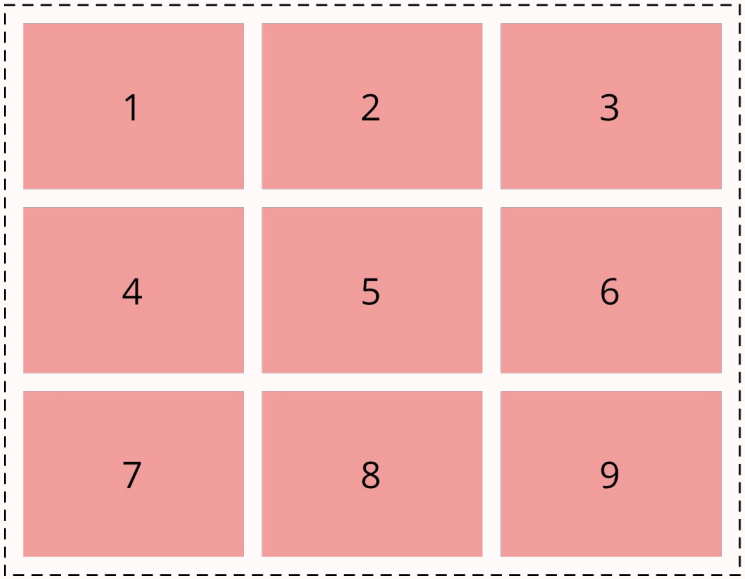


grid-auto-flow

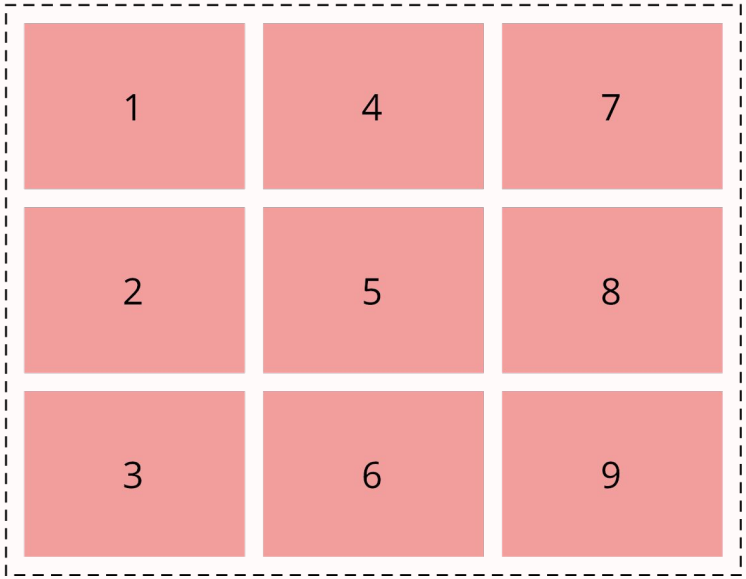
- grid-auto-flow property defines how auto-placement for grid items is done
- possible values
  - row (default): grid items are auto-placed on the grid rows
  - column: grid items are auto-placed on the grid columns
- dense keyword can be added to the row and column keywords
  - dense will try to fill any holes in the grid
  - will probably mix up original order of grid items
- Example

grid-auto-flow

`grid-auto-flow: row`

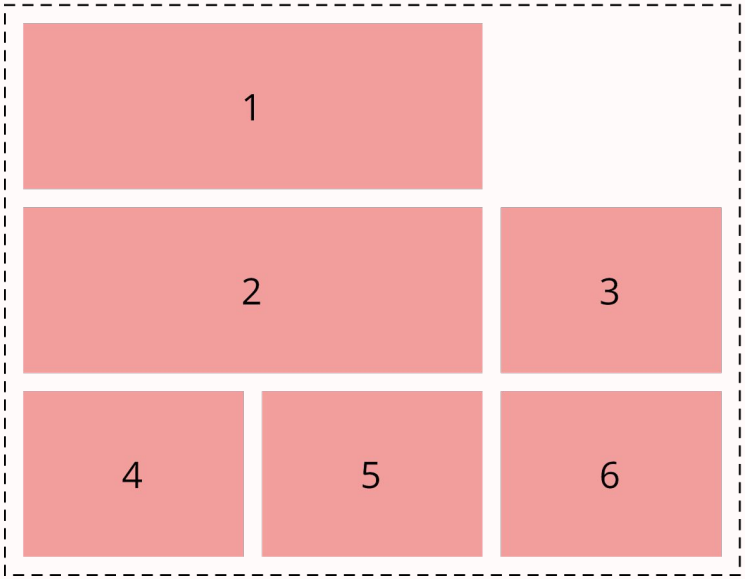


`grid-auto-flow: column`

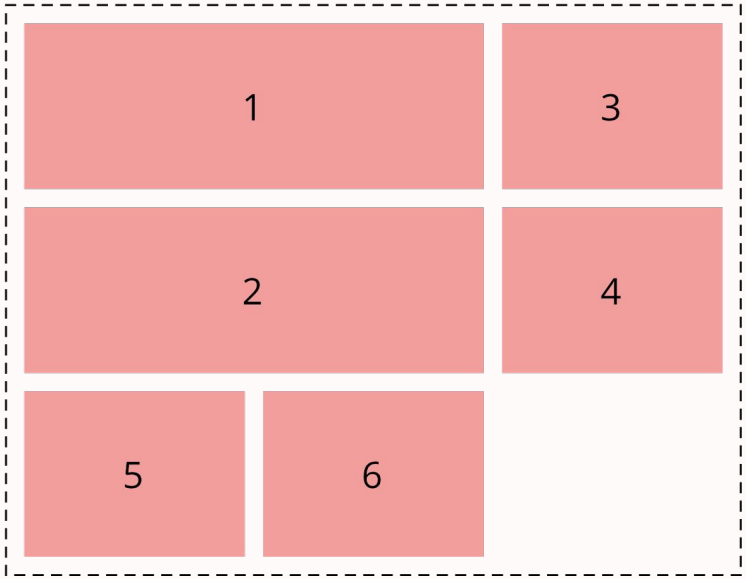


grid-auto-flow

`grid-auto-flow: row`

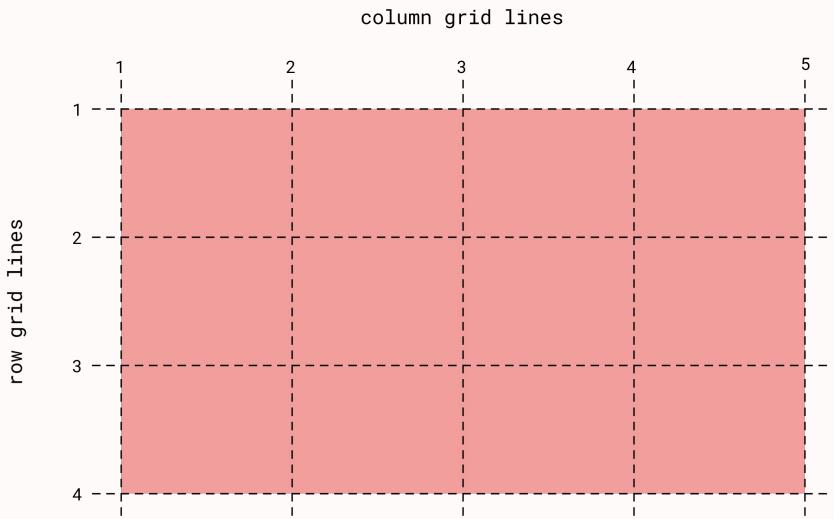


`grid-auto-flow: row dense`



Grid lines

- Grid items can be placed according to the grid lines
- Grid lines start before the first column / row
- Grid lines are between columns / rows
- Grid lines end after the last column / row



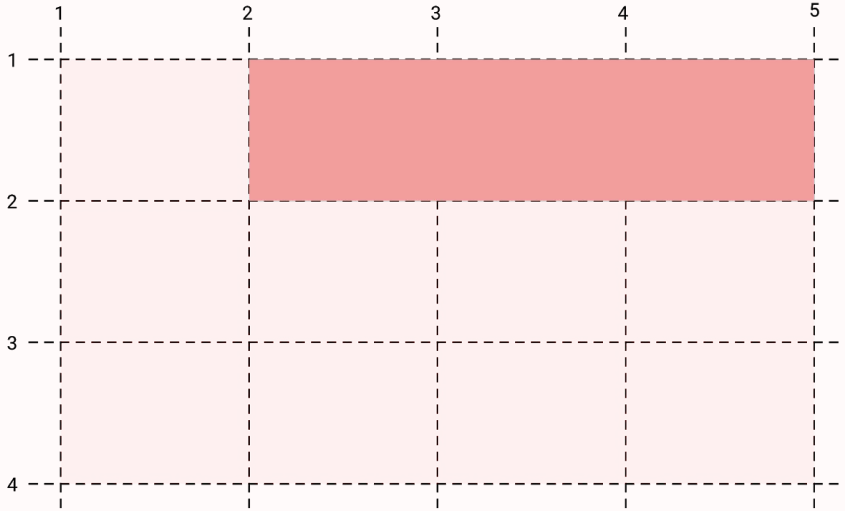
grid-column-start & grid-column-end

- The properties grid-column-start and grid-column-end define the start and end positions of grid items in the grid
- grid-column-start: 1 sets the starting position on the first grid line
- grid-column-end: 4 sets the end position on the forth grid line
- grid-column property shorthand can set both properties
- Example

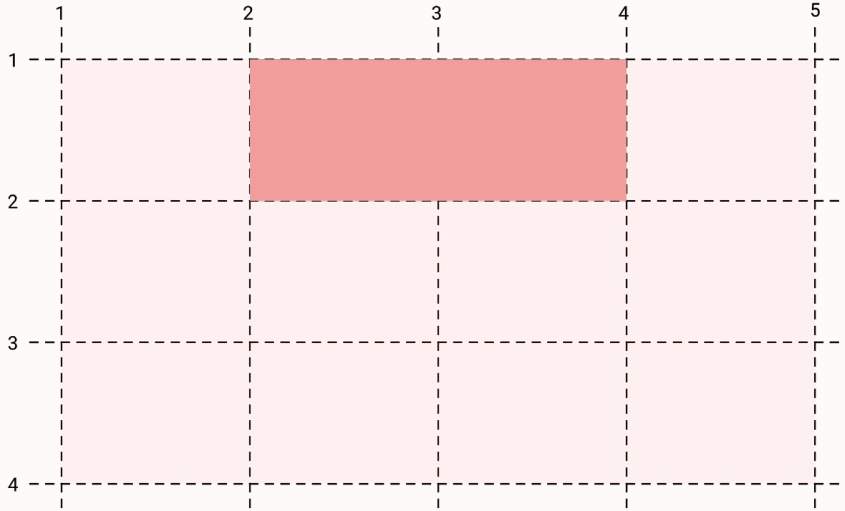

```
.grid {
    display: grid;
    grid-template-columns: 100px 100px 100px;
}

/* item will span across all columns */
.item {
    grid-column-start: 1;
    grid-column-end: 4;
}
```

grid-column-start & grid-column-end

`grid-column: 2 / 5`



`grid-column: 2 / -2`

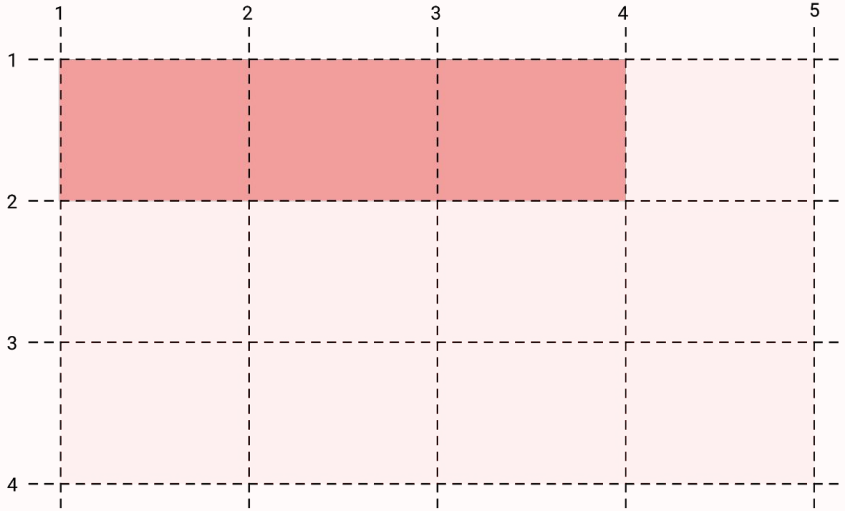


grid-column-start & grid-column-end

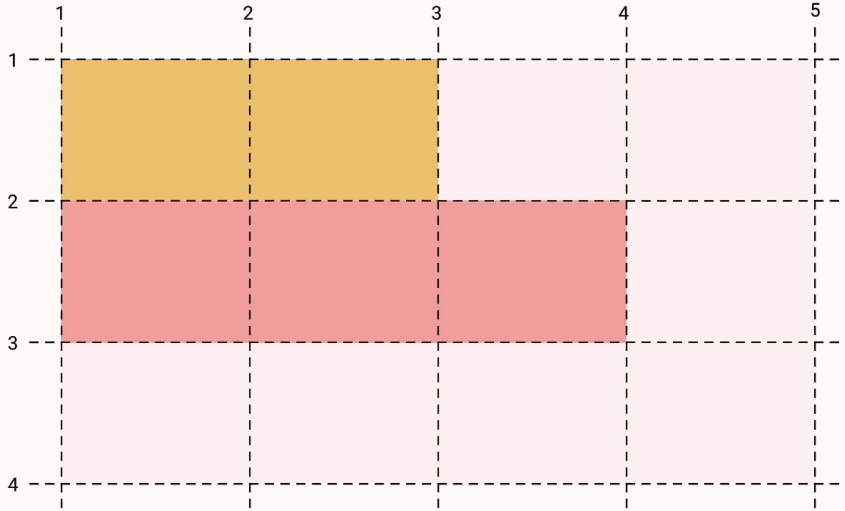
- grid-column-start and grid-column-end can be combined with the span keyword
- grid-column-start: span 2 will span the item from its (automatic) start position across two grid lines
- grid-column-end: span 3 will span the item from its (automatic) end position across 3 grid lines to the start

grid-column-start & grid-column-end

`grid-column-start: span 3`



`grid-column-start: span 3`



grid-row-start & grid-row-end

- The properties grid-row-start and grid-row-end work like the properties for column, but for rows instead
- grid-row property shorthand can set both properties

Sprint 5: Use the power of Grid

Sprint 5: Use the power of grid

- Turn your nav from display: inline to grid
- Get some images from pexels and create an (thumbnail) image gallery
- Use the grid to format your sections / articles
- Go crazy 🥳

Resources

Resources

- CSS Diner
- Flexbox Froggy
- CSS Grid Garden
- The Noun Project
- Google Fonts
- MDN
- Web Bos Courses (Grid, Flexbox and it's freeeee 💯)