

HTML & CSS_

AXXES TRAINEESHIP 2021

AXXES_

WHO AM I?

Yannick Van der Jonckheyd

Frontend / Java Developer
Axxes since 2011

Project @ Liantis
Before: VRT, Bank Delen



TABLE OF CONTENTS

Booting

Introduction

Browsers

HTML

Introduction

Basics

Elements

CSS

Introduction

Basics

Boxes

Layouts

Animations & Transitions

Maintenance

Wrap up

Questions?

Interesting reads

HTML



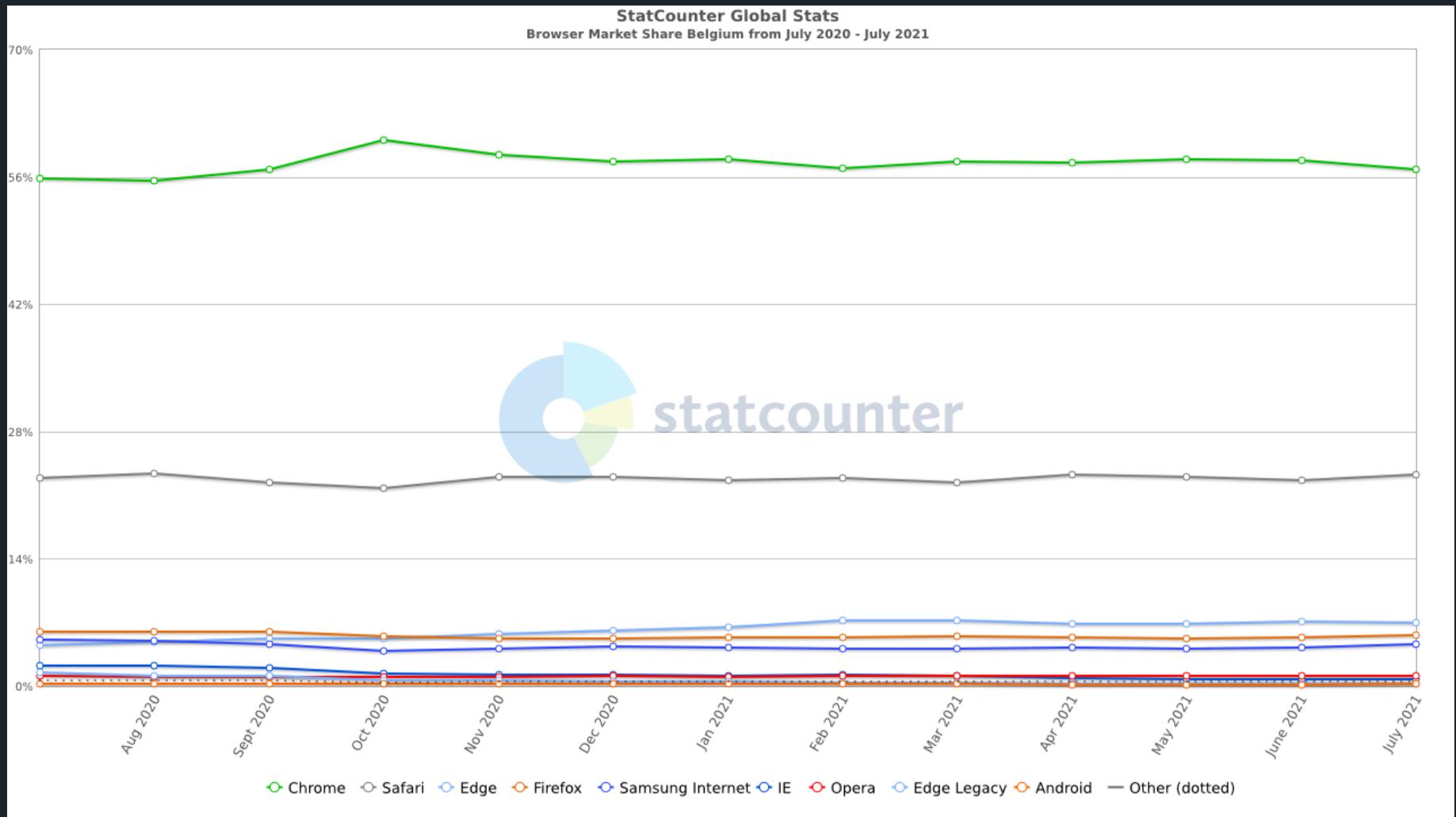
INTRODUCTION

- HyperText Markup Language
- Programming Language
- Structure of web page
- Consists of a series of elements

INTRODUCTION

- HyperText Markup Language
- ~~Programming Language~~
- Structure of web page
- Consists of a series of elements

BROWSERS



ANATOMY

```
<!-- element -->  
<tag attribute="value">content</tag>
```

ANATOMY

```
<!-- element -->  
<tag attribute="value">content</tag>
```

```
<p class="highlight">This is highlighted text</p>
```

EMPTY ELEMENTS

```
  

```



NESTING ELEMENTS

```
<p>This is a <em>cursive</em> text</p>
```

This is a *cursive* text

BLOCK VERSUS INLINE ELEMENTS

- Block-level elements form a visible block
 - Appear on a new line
 - Usually structural elements
- Inline elements are contained within block elements
 - Cause no new line
 - Typically text

```
<em>one</em><em>two</em><em>three</em>  
<p>four</p><p>five</p><p>six</p>
```

onetwothree

four

five

six

ATTRIBUTES

Attributes contain extra information about the element that won't appear in the content.

```
<a href="https://www.axxes.com"  
    title="Website of Axxes IT"  
    target="_blank">  
    Click here to visit Axxes  
</a>
```

Click here to visit Axxes

BOOLEAN ATTRIBUTES

Attributes without values

```
<input type="text" disabled="disabled" />  
  
<!-- shorthand -->  
<input type="text" disabled />
```



HTML COMMENTS

```
<!-- This is a comment -->
```

WHITESPACE IN HTML

HTML parser reduces each sequence of whitespace to
a single space

```
<p>This is a paragraph</p>
<p>This      is
    a      paragraph</p>
```

This is a paragraph

This is a paragraph

SPECIAL CHARACTERS

- <, >, ", ' and & are special characters because they are part of HTML syntax
- When needed in text, use character references.
- Starts with an ampersand (&) and ends with semicolon (;)

Literal character Character reference equivalent

<

<

>

>

"

"

'

'

&

&

DOCUMENT STRUCTURE

Individual HTML elements aren't useful on their own.
Specific individual elements combined form an HTML
page.

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My test page</title>
  </head>
  <body>
    <p>This is my page</p>
  </body>
</html>
```

- `<!doctype html>`: the doctype
 - back in the days, it acted as link to a set of rules
 - now it's a historical artifact that needs to be included for everything else to work right
- `<html>` element / root element
 - wraps all the content on the page
- `<head>` element
 - container for everything you want to include, that isn't the content
- `<body>` element
 - contains *all* the content that displays on page

INSIDE THE <head>

The <head>'s content is not displayed on the page, but it's job is to contain metadata about the document.

ELEMENTS IN THE HEAD

- **meta**
 - typically used to specify the character set, page description, keywords, ...
- **link**
 - defines the relationship between current document and external resource
- **title**
 - defines title of the document
- **base**
 - specifies the base URL and/or target for all relative URLs in a page

- **style**
 - define style information for single page
- **script**
 - define client-side JavaScripts
- **noscript**
 - defines an alternate content to be displayed to users that have disabled scripts in their browser or have a browser that doesn't support script

```
<!--  
  Set the character encoding for this document, so that  
  all characters within the UTF-8 space (such as emoji)  
  are rendered correctly.  
-->  
<meta charset="utf-8">  
  
<!-- Set the document's title -->  
<title>Page Title</title>  
  
<!--  
  Set the base URL for all relative URLs within  
  the document  
-->  
<base href="https://example.com/page.html">  
  
<!-- Link to an external CSS file -->  
<link rel="stylesheet" href="styles.css">  
  
<!-- Used for adding in-document CSS -->  
<style>  
  /* ... */  
</style>
```

RECOMMENDED MINIMUM

Essential elements for any web document

```
<meta charset="utf-8">
<meta name="viewport"
      content="width=device-width, initial-scale=1">
<title>Page Title</title>
```

WORTH MENTIONING ELEMENTS

```
<meta charset="utf-8">
<meta name="viewport"
      content="width=device-width, initial-scale=1">

<!-- Allows control over where resources are loaded from -->
<meta http-equiv="Content-Security-Policy"
      content="default-src 'self'">

<!-- Short description of the document -->
<meta name="description" content="A description of the page">

<!--
      Control the behavior of search engine crawling and indexing
-->
<meta name="robots" content="index,follow">
<meta name="googlebot" content="index,follow">

<!-- Allows control over how referrer information is passed -->
<meta name="referrer" content="no-referrer">

<!-- Icon in the highest resolution we need it for -->
<link rel="icon" sizes="192x192" href="/path/to/icon.png">
```

SOCIAL

Facebook Open Graph

```
<meta property="og:url"
      content="https://example.com/page.html">
<meta property="og:type" content="website">
<meta property="og:title" content="Content Title">
<meta property="og:image"
      content="https://example.com/image.jpg">
<meta property="og:image:alt"
      content="A description of what is in the image">
<meta property="og:description" content="Description Here">
<meta property="og:site_name" content="Site Name">
<meta property="og:locale" content="en_US">
```

Twitter Cards

```
<meta name="twitter:card" content="summary">
<meta name="twitter:site" content="@site_account">
<meta name="twitter:creator" content="@individual_account">
<meta name="twitter:url"
      content="https://example.com/page.html">
<meta name="twitter:title" content="Content Title">
<meta name="twitter:description"
      content="Content description less than 200 chars">
<meta name="twitter:image"
      content="https://example.com/image.jpg">
<meta name="twitter:image:alt"
      content="A text description of the image conveying
      the essential nature of an image to users who are
      visually impaired. Maximum 420 characters.">
```

ELEMENTS

CONTENT ELEMENTS

HEADINGS

```
<h1>Main heading</h1>
<h2>Sub heading</h2>
<h3>Sub-subheading</h3>
<h4>And so on heading</h4>
<h5>Small heading</h5>
<h6>Smaller heading</h6>
```

PARAGRAPHS AND TEXT

```
<p>This is a paragraph</p>
```

BREAKLINE

```
<p>This is a paragraph <br/> with break line</p>
```

This is a paragraph
with break line

HORIZONTAL RULE

```
<p>This piece of code has</p>
<hr/>
<p>a horizontal rule</p>
```

This piece of code has

a horizontal rule

PREFORMATTED TEXT

```
<pre>  
Twinkle, twinkle, little star,  
How I wonder what      you are!  
          Up above the world so high,  
Like a diamond in the sky.  
</pre>
```

Twinkle, twinkle, little star,
How I wonder what you are!
 Up above the world so high,
Like a diamond in the sky.

TEXT FORMATTING

```
<p>This is <b>bold text</b>.</p>
<p>This is <strong>strongly important text</strong>.</p>
<p>This is <i>italic text</i>.</p>
<p>This is <em>emphasized text</em>.</p>
<p>This is <mark>highlighted text</mark>.</p>
<p>This is <code>computer code</code>.</p>
<p>This is <small>smaller text</small>.</p>
<p>This is <sub>subscript</sub> text.</p>
<p>This is <sup>superscript</sup> text.</p>
<p>This is <del>deleted text</del>.</p>
<p>This is <ins>inserted text</ins>.</p>
```

QUOTES

```
<blockquote>
  <p>Patience you must have my young Padawan</p>
  <cite>- Yoda</cite>
</blockquote>
```

*Patience you must have my young
Padawan*

- Yoda

LISTS

Used to present list of information in well-formed and semantic way

- **Unordered list**
 - Used to create a list of related items, in no particular order.
- **Ordered list**
 - Used to create a list of related items, in a specific order.
- **Description list**
 - Used to create a list of terms and their descriptions.

UNORDERED LIST

```
1 <ul>
2   <li>Softened butter</li>
3   <li>Sugar</li>
4   <li>Eggs</li>
5   <li>Flour</li>
6 </ul>
```

UNORDERED LIST

```
1 <ul>
2   <li>Softened butter</li>
3   <li>Sugar</li>
4   <li>Eggs</li>
5   <li>Flour</li>
6 </ul>
```

UNORDERED LIST

```
1 <ul>
2   <li>Softened butter</li>
3   <li>Sugar</li>
4   <li>Eggs</li>
5   <li>Flour</li>
6 </ul>
```

ORDERED LIST

```
1 <ol>
2   <li>Heat the oven to 180 degrees C</li>
3   <li>Lightly grease round cake tin with butter</li>
4   <li>Put all the ingredients into a mixing bowl</li>
5   <li>Pour the mixture into the tin</li>
6 </ol>
```

ORDERED LIST

```
1 <ol>
2   <li>Heat the oven to 180 degrees C</li>
3   <li>Lightly grease round cake tin with butter</li>
4   <li>Put all the ingredients into a mixing bowl</li>
5   <li>Pour the mixture into the tin</li>
6 </ol>
```

NESTING LISTS

```
1 <ol>
2   <li>Remove the skin from the garlic, and chop coarsely.</li>
3   <li>Remove all the seeds and stalk from the pepper, and
4     <li>Add all the ingredients into a food processor.</li>
5     <li>Process all the ingredients into a paste.
6       <ul>
7         <li>If you want a coarse "chunky" hummus, process it
8         <li>If you want a smooth hummus, process it for a longer
9       </ul>
10    </li>
11 </ol>
```

NESTING LISTS

```
1 <ol>
2   <li>Remove the skin from the garlic, and chop coarsely.</li>
3   <li>Remove all the seeds and stalk from the pepper, and
4     <li>Add all the ingredients into a food processor.</li>
5     <li>Process all the ingredients into a paste.
6       <ul>
7         <li>If you want a coarse "chunky" hummus, process it
8         <li>If you want a smooth hummus, process it for a longer
9       </ul>
10    </li>
11 </ol>
```

NESTING LISTS

```
1 <ol>
2   <li>Remove the skin from the garlic, and chop coarsely.</li>
3   <li>Remove all the seeds and stalk from the pepper, and
4     <li>Add all the ingredients into a food processor.</li>
5     <li>Process all the ingredients into a paste.
6       <ul>
7         <li>If you want a coarse "chunky" hummus, process it
8         <li>If you want a smooth hummus, process it for a longer
9       </ul>
10    </li>
11 </ol>
```

NESTING LISTS

```
1 <ol>
2   <li>Remove the skin from the garlic, and chop coarsely.</li>
3   <li>Remove all the seeds and stalk from the pepper, and
4     <li>Add all the ingredients into a food processor.</li>
5     <li>Process all the ingredients into a paste.
6       <ul>
7         <li>If you want a coarse "chunky" hummus, process it
8         <li>If you want a smooth hummus, process it for a longer
9       </ul>
10    </li>
11 </ol>
```

DESCRIPTION LISTS

```
1 <dl>
2   <dt>Bread</dt>
3   <dd>A baked food made of flour.</dd>
4   <dt>Coffee</dt>
5   <dd>A drink made from roasted coffee beans.</dd>
6 </dl>
```

DESCRIPTION LISTS

```
1 <dl>
2   <dt>Bread</dt>
3   <dd>A baked food made of flour.</dd>
4   <dt>Coffee</dt>
5   <dd>A drink made from roasted coffee beans.</dd>
6 </dl>
```

DESCRIPTION LISTS

```
1 <dl>
2   <dt>Bread</dt>
3   <dd>A baked food made of flour.</dd>
4   <dt>Coffee</dt>
5   <dd>A drink made from roasted coffee beans.</dd>
6 </dl>
```

DESCRIPTION LISTS

```
1 <dl>
2   <dt>Bread</dt>
3   <dd>A baked food made of flour.</dd>
4   <dt>Coffee</dt>
5   <dd>A drink made from roasted coffee beans.</dd>
6 </dl>
```

Hyperlinks

The anchor element with its href attribute, creates a hyperlink to web pages, files, emails, etc..

```
<a href="https://www.axxes.com">Axxes Website</a>
```

Supporting information with title attribute

```
<a href="https://www.axxes.com"  
title="The official website of Axxes, a IT consul...">Axx
```

Links

BUTTONS

IMAGES

```

```

RESPONSIVE IMAGES

- Why responsive images?
 - Different screen sizes
 - Orientation of screen
 - High resolution screens

```

```

SECTION ELEMENTS

- <header>
- <footer>
- <navigation>
- <section>
- <main>
- <aside>

```
<body>
```

```
  <div id="container">
```

```
    <header>
```

```
      <nav>
```

```
    <section>
```

```
      <article>
```

```
      <article>
```

```
    <aside>
```

```
      <section>
```

```
      <section>
```

```
    <footer>
```

```
<!doctype html>
<html lang="nl">
<head>
</head>
<body>
  <header>
    <h1>Header</h1>
  </header>

  <nav>
    <ul>
      <li><a href="#">Home</a></li>
      <li><a href="#">Our team</a></li>
      <li><a href="#">Projects</a></li>
      <li><a href="#">Contact</a></li>
    </ul>
```

FORM ELEMENTS

FORM

```
<form action="" method="post" autocomplete="on">
  <!--  -->
</form>
```

LABEL & INPUT

Used to create interactive controls for web-based forms in order to accept data from the user

```
<label for="name">Name</label>
<input type="text" id="name" name="name">
```

- Great variety of input types:

- `text`
- `email`
- `search`
- `tel`
- `password`
- `number`
- `checkbox`
- `radio`
- `button`
- `submit`
- `color`
- `date`
- `file`
- `hidden`
- `range`

TEXT BASED INPUTS

```
<input type="text" minlength="2" maxlength="6">
<input type="email" required multiple>
<input type="search">
<input type="tel">
<input type="password" pattern="[0-9a-fA-F]{4,8}">
```

NUMBER

```
<input type="number">
```

CHECKBOX & RADIO

```
<div>
  <input type="checkbox" id="coding" name="coding">
  <label for="coding">Coding</label>
</div>
```

```
<p>Select a maintenance drone:</p>
<div>
  <input type="radio" id="huey" name="drone" value="huey" checked="checked">
  <label for="huey">Huey</label>
</div>
<div>
  <input type="radio" id="dewey" name="drone" value="dewey">
  <label for="dewey">Dewey</label>
</div>
<div>
  <input type="radio" id="louie" name="drone" value="louie">
  <label for="louie">Louie</label>
</div>
```

BUTTONS

```
<input type="reset">  
<button type="reset"></button>  
<input type="submit">  
<button type="reset"></button>
```

SELECT

```
<select name="name" id="id">
  <option value="dog">Dog</option>
  <option value="cat">Cat</option>
  <option value="hamster">Hamster</option>
  <option value="parrot">Parrot</option>
  <option value="spider">Spider</option>
  <option value="goldfish">Goldfish</option>
</select>
```

TEXT-AREA

```
<textarea name="name" id="id" cols="30" rows="10" ></textarea>
```

AUTOCOMPLETE ATTRIBUTE

- Autocomplete on form element
- Available on input, textarea, select and form elements
- Suggested values is generally up to the browser; typically values come from past values entered by the user, but they may also come from pre-configured values

<https://developer.mozilla.org/en-US/docs/Web/HTML/Attributes/autocomplete>

css_

INTRODUCTION

SYNTAX

ANATOMY

```
[selector] {  
    [property]: [value]; /* declaration */  
}  
/* rule */
```

ANATOMY

```
[selector] {  
    [property]: [value]; /* declaration */  
}  
/* rule */
```

```
p {  
    background-color: #f00;  
}
```

COMMENTS

```
/* Put your comment here */
```

COMMENTS

```
/* Put your comment here */
```

```
// Put your comment here  
<!-- -->
```

SHORTHANDS

Some properties like `margin`, `padding`,
`background`, `border`, ... can use shorthand
notations to set several property values in a single line

```
.foo {  
  margin: top right bottom left;  
  margin: top right-left bottom;  
  margin: top-bottom right-left;  
  margin: top-bottom-right-left;  
}
```

```
.foo {  
    /* top 2px, right 3px, bottom 4px, left 5px */  
    margin: 2px 3px 4px 5px;  
  
    /* top 4px, right 6px, bottom 8px, left 6px */  
    margin: 4px 6px 8px;  
  
    /* top 5px, right 10px, bottom 5px, left 10px */  
    margin: 5px 10px;  
  
    /* top 10px, right 10px, bottom 10px, left 10px */  
    margin: 10px;  
}
```

```
.foo {  
    background: #f00 url("image.png") 10px 10px  
              no-repeat fixed;  
    padding: 10px 20px 30px 40px;  
}
```

```
.foo {  
    background-color: hotpink;  
    background-image: url("image.png");  
    background-position: 10px 10px;  
    background-repeat: no-repeat;  
    background-attachment: fixed;  
  
    padding-top: 10px;  
    padding-right: 20px;  
    padding-bottom: 30px;  
    padding-left: 40px;  
}
```

INLINE, INTERNAL OR EXTERNAL

TODO

SELECTORS

BASIC SELECTORS

ELEMENT SELECTORS

Select all elements of the given type

```
1 <p>Example</p>
```

```
1 p {  
2     background-color: #f00;  
3 }
```

ELEMENT SELECTORS

Select all elements of the given type

```
1 <p>Example</p>
```

```
1 p {  
2     background-color: #f00;  
3 }
```

CLASS SELECTORS

Selects all elements with the given class

```
1 <p class="foo">Example</p>
```

```
1 .foo {  
2     background-color: #f00;  
3 }
```

ID SELECTOR

Selects the element with the given ID

```
1 <p id="bar">Example</p>
```

```
1 #bar {  
2     background-color: #f00;  
3 }
```

UNIVERSAL SELECTOR

The universal selector is the power selector, able to select all elements on the page.

```
1 <p id="foo">Example</p>
2 <div id="bar">Example 2</div>
```

```
1 * {
2     background-color: #f00;
3 }
```

UNIVERSAL SELECTOR

The universal selector is the power selector, able to select all elements on the page.

```
1 <p id="foo">Example</p>
2 <div id="bar">Example 2</div>
```

```
1 * {
2     background-color: #f00;
3 }
```

ATTRIBUTE SELECTORS

The attribute selectors allow you to select elements with specific attributes. The selector is extended with some very usefull features.

HAS ATTRIBUTE

[attr] selects all elements that have this attribute

```
1 <div data-attr="foo"></div>
```

```
1 [data-attr] {  
2     background-color: #f00;  
3 }
```

HAS ATTRIBUTE

[attr] selects all elements that have this attribute

```
1 <div data-attr="foo"></div>
```

```
1 [data-attr] {  
2     background-color: #f00;  
3 }
```

EQUALS

[attr="val"] selects all elements that have this attribute with the given val

```
1 <div data-attr="foo"></div>
```

```
1 [data-attr="foo"] {
2     background-color: #f00;
3 }
```

SPACE SEPARATED LIST

[attr~="val"]: Selects all elements that have this attribute and where val is part of the space separated list.

```
1 <div lang="en-us en-gb en-au en-nz">Hello World!</div>
```

```
1 [lang~="en-gb"] {  
2     background-color: #f00;  
3 }
```

EQUALS OR START OF DASHED-STRING

[attr | = "val"]: selects all elements that have this attribute with given val or value starts with val followed by hyphen

```
1 <div lang="nl">Hallo Nederlands sprekend persoon!</div>
2 <div lang="nl-BE">Hey Belg!</div>
3 <div lang="nl-NL">Hoi Hollander!</div>
```

```
1 [lang|="nl"] {
2     background-color: #f00;
3 }
```

STARTS WITH

[attr^="val"] : Select all elements that have this attribute and the value starts with val.

```
1 <a href="https://www.axxes.com">Axxes Website</a>
```

```
1 // secure links only
2 [href^="https://"] {
3     background-color: #f00;
4 }
```

ENDS WITH

[attr\$="val"]: Select all elements that have this attribute and the value ends with val.

```
1 <a href="https://www.axxes.com">Axxes Website</a>
```

```
1 [href$=".com"] {  
2     background-color: #f00;  
3 }
```

SUBSTRING

[attr*="val"]: Select all elements that have this attribute and the value contains the substring val.

```
1 <a href="https://www.axxes.com">Axxes Website</a>
```

```
1 [href*="axxes"] {  
2     background-color: #f00;  
3 }
```

PSEUDO CLASSES

A pseudo class is a selector used to select an element
in a certain state or context

:active

Selects an element with the active state which basically is the element you've clicked on until you release your mouse button.

```
1 <a class="button">Example</a>
```

```
1 .button {  
2     background-color: #f00;  
3 }  
4  
5 .button:active {  
6     background-color: #0f0;  
7 }
```

:active

Selects an element with the active state which basically is the element you've clicked on until you release your mouse button.

```
1 <a class="button">Example</a>
```

```
1 .button {  
2     background-color: #f00;  
3 }  
4  
5 .button:active {  
6     background-color: #0f0;  
7 }
```

:focus

Selects the element that has focus.

```
1 <input/>
```

```
1 input:focus {  
2     background-color: #f00;  
3 }
```

:hover

Selects the element that is hovered.

```
1 a:hover {  
2     background-color: #f00;  
3 }
```

`:empty`

Selects every element that has no children (including text nodes).

```
1 <div></div>
2 <div>I'm not selected</div>
3 <div>
4     <p>I'm also not selected</p>
5 </div>
```

```
1 div:empty {
2     background-color: #f00;
3 }
```

`:empty`

Selects every element that has no children (including text nodes).

```
1 <div></div>
2 <div>I'm not selected</div>
3 <div>
4     <p>I'm also not selected</p>
5 </div>
```

```
1 div:empty {
2     background-color: #f00;
3 }
```

:empty

Selects every element that has no children (including text nodes).

```
1 <div></div>
2 <div>I'm not selected</div>
3 <div>
4     <p>I'm also not selected</p>
5 </div>
```

```
1 div:empty {
2     background-color: #f00;
3 }
```

`:lang(language)`

Selects all elements that have a `lang` attribute with the given value on it.

```
1 <div lang="nl">Hallo Nederlands sprekend persoon!</div>
2 <div lang="nl-BE">Hey Belg!</div>
3 <div lang="nl-NL">Hoi Hollander!</div>
```

```
1 div:lang(nl) {
2     background-color: #f00;
3 }
```

:link & :visited

Selects all unvisited/visited links

```
1 <a href="https://www.axxes.com">Axxes website</a>
```

```
1 a:link {  
2     background-color: #f00;  
3 }  
4  
5 a:visited {  
6     background-color: #0f0;  
7 }
```

:link & :visited

Selects all unvisited/visited links

```
1 <a href="https://www.axxes.com">Axxes website</a>
```

```
1 a:link {  
2     background-color: #f00;  
3 }  
4  
5 a:visited {  
6     background-color: #0f0;  
7 }
```

`:not(selector)`

Represents elements that do not match a list of selectors. Since it prevents specific items from being selected, it is known as the negation pseudo-class.

```
1 <a class="example">Selected</a>
2 <p class="example">Not Selected</p>
```

```
1 .example:not(p) {
2     color: #f00;
3 }
```

`:not(selector)`

Represents elements that do not match a list of selectors. Since it prevents specific items from being selected, it is known as the negation pseudo-class.

```
1 <a class="example">Selected</a>
2 <p class="example">Not Selected</p>
```

```
1 .example:not(p) {
2     color: #f00;
3 }
```

read-only & read-write

read-only selects all elements that are read only
(have a readonly attribute) **read-write** selects all
elements that are not read only (have not a readonly
attribute)

```
1 <input readonly/>
```

```
1 input:read-only {  
2     background-color: #f00;  
3 }  
4 input:read-write {  
5     background-color: #0f0;  
6 }
```

:root

Selects root element of the document. In HTML, this will result in the <html> element

```
1 <html>
2   <body>
3     <p>Hello world</p>
4   </body>
5 </html>
```

```
1 :root {
2   background-color: #f00;
3 }
```

:root

Selects root element of the document. In HTML, this will result in the `<html>` element

```
1 <html>
2   <body>
3     <p>Hello world</p>
4   </body>
5 </html>
```

```
1 :root {
2   background-color: #f00;
3 }
```

:target

Selects the element with the same ID as the URL anchor.

```
https://www.awebsite.com/article/#comments
```

```
1 <div id="article">Not Selected</div>
2 <div id="comments">Selected</div>
```

```
1 div:target {
2     background-color: #f00;
3 }
```

:target

Selects the element with the same ID as the URL anchor.

```
https://www.awebsite.com/article/#comments
```

```
1 <div id="article">Not Selected</div>
2 <div id="comments">Selected</div>
```

```
1 div:target {
2     background-color: #f00;
3 }
```

`:is()` & `:matches()`

Takes a selector list as its argument. It selects all elements that match one of the selectors in that list.

```
1 :is(header, main, footer) p:hover {  
2   color: red;  
3   cursor: pointer;  
4 }  
5  
6 /* same as following */  
7 header p:hover,  
8 main p:hover,  
9 footer p:hover {  
10   color: red;  
11   cursor: pointer;  
12 }
```

`:is()` & `:matches()`

Takes a selector list as its argument. It selects all elements that match one of the selectors in that list.

```
1 :is(header, main, footer) p:hover {  
2   color: red;  
3   cursor: pointer;  
4 }  
5  
6 /* same as following */  
7 header p:hover,  
8 main p:hover,  
9 footer p:hover {  
10  color: red;  
11  cursor: pointer;  
12 }
```

```
/* Level 0 */
h1 {
    font-size: 30px;
}
/* Level 1 */
section h1, article h1, aside h1, nav h1 {
    font-size: 25px;
}
/* Level 2 */
section section h1, section article h1, section aside h1,
section nav h1, article section h1, article article h1,
article aside h1, article nav h1, aside section h1,
aside article h1, aside aside h1, aside nav h1, nav section h1,
nav article h1, nav aside h1, nav nav h1 {
    font-size: 20px;
}
/* Level 3 */
/* and so on... */
```

```
/* Level 0 */
h1 {
  font-size: 30px;
}
/* Level 1 */
:is(section, article, aside, nav) h1 {
  font-size: 25px;
}
/* Level 2 */
:is(section, article, aside, nav)
:is(section, article, aside, nav) h1 {
  font-size: 20px;
}
/* Level 3 */
:is(section, article, aside, nav)
:is(section, article, aside, nav)
:is(section, article, aside, nav) h1 {
  font-size: 15px;
}
```

`:fullscreen`

This selector matches every element which is currently in full-screen mode.

```
button { /* or button:not(:fullscreen) */
    color: #0f0;
}

button:fullscreen {
    color: #f00;
}
```

:checked

Selects an element with the checked state.

```
1 <input type="checkbox" checked/>
```

```
1 input:checked {  
2     background-color: #f00;  
3 }
```

`:disabled & :enabled`

Selects an element with the disabled / enabled
(default) state.

```
1 <input/>
2 <input disabled/>
```

```
1 input:disabled {
2     background-color: #f00;
3 }
4 input:enabled {
5     background-color: #0f0;
6 }
```

`:disabled & :enabled`

Selects an element with the disabled / enabled
(default) state.

```
1 <input/>
2 <input disabled/>
```

```
1 input:disabled {
2     background-color: #f00;
3 }
4 input:enabled {
5     background-color: #0f0;
6 }
```

:invalid & :valid

Selects an input element with a invalid / valid value

```
1 <input type="email" value="foo@bar.baz"/>
2 <input type="email" value="foobar.baz"/>
```

```
1 input:invalid {
2     background-color: #f00;
3 }
4 input:valid {
5     background-color: #0f0;
6 }
```

:invalid & :valid

Selects an input element with a invalid / valid value

```
1 <input type="email" value="foo@bar.baz"/>
2 <input type="email" value="foobar.baz"/>
```

```
1 input:invalid {
2     background-color: #f00;
3 }
4 input:valid {
5     background-color: #0f0;
6 }
```

:out-of-range & :in-range

Selects an input element that is in / out of range (value between min and max)

```
1 <input type="number" value="9" min="0" max="10"/>
2 <input type="number" value="11" min="0" max="10"/>
```

```
1 input:out-of-range {
2     background-color: #f00;
3 }
4 input:in-range {
5     background-color: #0f0;
6 }
```

:out-of-range & :in-range

Selects an input element that is in / out of range (value between min and max)

```
1 <input type="number" value="9" min="0" max="10"/>
2 <input type="number" value="11" min="0" max="10"/>
```

```
1 input:out-of-range {
2     background-color: #f00;
3 }
4 input:in-range {
5     background-color: #0f0;
6 }
```

:required & :optional

Select all input elements that are required / optional
(not required, default)

```
1 <input/>
2 <input required/>
```

```
1 input:required {
2     background-color: #f00;
3 }
4 input:optional {
5     background-color: #0f0;
6 }
```

:required & :optional

Select all input elements that are required / optional
(not required, default)

```
1 <input/>
2 <input required/>
```

```
1 input:required {
2     background-color: #f00;
3 }
4 input:optional {
5     background-color: #0f0;
6 }
```

:first-child

Selects every element that is the first child

```
1 <div>
2     <span>Selected</span>
3     <span>Not selected</span>
4 </div>
5 <div>
6     <span>Selected</span>
7     <span>Not selected</span>
8 </div>
9 <div>
10    <p>Not selected</p>
11    <span>Not selected</span>
12 </div>
```

```
1 span:first-child {
2     background-color: #f00;
3 }
```

`:first-child`

Selects every element that is the first child

```
1 <div>
2   <span>Selected</span>
3   <span>Not selected</span>
4 </div>
5 <div>
6   <span>Selected</span>
7   <span>Not selected</span>
8 </div>
9 <div>
10  <p>Not selected</p>
11  <span>Not selected</span>
12 </div>
```

```
1 span:first-child {
2   background-color: #f00;
3 }
```

:first-child

Selects every element that is the first child

```
1 <div>
2     <span>Selected</span>
3     <span>Not selected</span>
4 </div>
5 <div>
6     <span>Selected</span>
7     <span>Not selected</span>
8 </div>
9 <div>
10    <p>Not selected</p>
11    <span>Not selected</span>
12 </div>
```

```
1 span:first-child {
2     background-color: #f00;
3 }
```

:first-child

Selects every element that is the first child

```
1 <div>
2     <span>Selected</span>
3     <span>Not selected</span>
4 </div>
5 <div>
6     <span>Selected</span>
7     <span>Not selected</span>
8 </div>
9 <div>
10    <p>Not selected</p>
11    <span>Not selected</span>
12 </div>
```

```
1 span:first-child {
2     background-color: #f00;
3 }
```

`:first-of-type`

Selects every element that is the first child of given type

```
1 <div>
2   <span>Selected</span>
3   <span>Not selected</span>
4 </div>
5 <div>
6   <span>Selected</span>
7   <span>Not selected</span>
8 </div>
9 <div>
10  <p>Not selected</p>
11  <span>Selected</span>
12 </div>
```

```
1 span:first-of-type {
2   background-color: #f00;
3 }
```

`:first-of-type`

Selects every element that is the first child of given type

```
1 <div>
2   <span>Selected</span>
3   <span>Not selected</span>
4 </div>
5 <div>
6   <span>Selected</span>
7   <span>Not selected</span>
8 </div>
9 <div>
10  <p>Not selected</p>
11  <span>Selected</span>
12 </div>
```

```
1 span:first-of-type {
2   background-color: #f00;
3 }
```

`:first-of-type`

Selects every element that is the first child of given type

```
1 <div>
2   <span>Selected</span>
3   <span>Not selected</span>
4 </div>
5 <div>
6   <span>Selected</span>
7   <span>Not selected</span>
8 </div>
9 <div>
10  <p>Not selected</p>
11  <span>Selected</span>
12 </div>
```

```
1 span:first-of-type {
2   background-color: #f00;
3 }
```

:last-child

Selects every element that is the last child

```
1 <div>
2     <span>Not selected</span>
3     <span>Selected</span>
4 </div>
5 <div>
6     <span>Not selected</span>
7     <span>Selected</span>
8 </div>
9 <div>
10    <span>Not selected</span>
11    <p>Not selected</p>
12 </div>
```

```
1 span:last-child {
2     background-color: #f00;
3 }
```

:last-child

Selects every element that is the last child

```
1 <div>
2     <span>Not selected</span>
3     <span>Selected</span>
4 </div>
5 <div>
6     <span>Not selected</span>
7     <span>Selected</span>
8 </div>
9 <div>
10    <span>Not selected</span>
11    <p>Not selected</p>
12 </div>
```

```
1 span:last-child {
2     background-color: #f00;
3 }
```

:last-child

Selects every element that is the last child

```
1 <div>
2     <span>Not selected</span>
3     <span>Selected</span>
4 </div>
5 <div>
6     <span>Not selected</span>
7     <span>Selected</span>
8 </div>
9 <div>
10    <span>Not selected</span>
11    <p>Not selected</p>
12 </div>
```

```
1 span:last-child {
2     background-color: #f00;
3 }
```

:last-child

Selects every element that is the last child

```
1 <div>
2     <span>Not selected</span>
3     <span>Selected</span>
4 </div>
5 <div>
6     <span>Not selected</span>
7     <span>Selected</span>
8 </div>
9 <div>
10    <span>Not selected</span>
11    <p>Not selected</p>
12 </div>
```

```
1 span:last-child {
2     background-color: #f00;
3 }
```

`:last-of-type`

Selects every element that is the last child of given type

```
1 <div>
2   <span>Not selected</span>
3   <span>Selected</span>
4 </div>
5 <div>
6   <span>Not selected</span>
7   <span>Selected</span>
8 </div>
9 <div>
10  <span>Selected</span>
11  <p>Not selected</p>
12 </div>
```

```
1 span:last-of-type {
2   background-color: #f00;
3 }
```

`:last-of-type`

Selects every element that is the last child of given type

```
1 <div>
2   <span>Not selected</span>
3   <span>Selected</span>
4 </div>
5 <div>
6   <span>Not selected</span>
7   <span>Selected</span>
8 </div>
9 <div>
10  <span>Selected</span>
11  <p>Not selected</p>
12 </div>
```

```
1 span:last-of-type {
2   background-color: #f00;
3 }
```

`:last-of-type`

Selects every element that is the last child of given type

```
1 <div>
2     <span>Not selected</span>
3     <span>Selected</span>
4 </div>
5 <div>
6     <span>Not selected</span>
7     <span>Selected</span>
8 </div>
9 <div>
10    <span>Selected</span>
11    <p>Not selected</p>
12 </div>
```

```
1 span:last-of-type {
2     background-color: #f00;
3 }
```

`:nth-child(n)`

Selects an element based on the n parameter that is passed.

```
1 <div>
2   <p>Not selected</p>
3   <p>Selected</p>
4   <p>Not selected</p>
5   <p>Not selected</p>
6 </div>
```

```
1 p:nth-child(2) {
2   color: #f00;
3 }
```

`:nth-child(n)`

Selects an element based on the n parameter that is passed.

```
1 <div>
2   <p>Not selected</p>
3   <p>Selected</p>
4   <p>Not selected</p>
5   <p>Not selected</p>
6 </div>
```

```
1 p:nth-child(2) {
2   color: #f00;
3 }
```

You're also able to select multiple elements by using the n keyword.

```
1 <div>
2   <p>Not selected</p>
3   <p>Selected</p>
4   <p>Not selected</p>
5   <p>Selected</p>
6   <p>Not selected</p>
7 </div>
```

```
1 p:nth-child(2n) {
2   color: #f00;
3 }
```

You're also able to select multiple elements by using the n keyword.

```
1 <div>
2   <p>Not selected</p>
3   <p>Selected</p>
4   <p>Not selected</p>
5   <p>Selected</p>
6   <p>Not selected</p>
7 </div>
```

```
1 p:nth-child(2n) {
2   color: #f00;
3 }
```

```
1 <div>
2     <p>Selected</p>
3     <p>Not selected</p>
4     <p>Selected</p>
5     <p>Not selected</p>
6     <p>Selected</p>
7 </div>
```

```
1 p:nth-child(2n+1) {
2     color: #f00;
3 }
```

```
1 <div>
2     <p>Selected</p>
3     <p>Not selected</p>
4     <p>Selected</p>
5     <p>Not selected</p>
6     <p>Selected</p>
7 </div>
```

```
1 p:nth-child(2n+1) {
2     color: #f00;
3 }
```

`:nth-last-child(n)`

Selects an element based on the n parameter that is passed, counting from the end

```
1 <div>
2   <p>Not Selected</p>
3   <p>Not Selected</p>
4   <p>Selected</p>
5   <p>Not Selected</p>
6 </div>
```

```
1 p:nth-last-child(2) {
2   color: #f00;
3 }
```

`:nth-last-child(n)`

Selects an element based on the n parameter that is passed, counting from the end

```
1 <div>
2   <p>Not Selected</p>
3   <p>Not Selected</p>
4   <p>Selected</p>
5   <p>Not Selected</p>
6 </div>
```

```
1 p:nth-last-child(2) {
2   color: #f00;
3 }
```

`:nth-of-type(n)`

Works the same as `:nth-child(n)`, but only counts element of a specific type

```
1 <div>
2   <p>Not selected</p>
3   <span>Not selected</span>
4   <p>Selected</p>
5   <p>Not selected</p>
6   <p>Not selected</p>
7   <p>Not selected</p>
8 </div>
```

```
1 p:nth-of-type(2) {
2   color: #f00;
3 }
```

`:nth-of-type(n)`

Works the same as `:nth-child(n)`, but only counts element of a specific type

```
1 <div>
2   <p>Not selected</p>
3   <span>Not selected</span>
4   <p>Selected</p>
5   <p>Not selected</p>
6   <p>Not selected</p>
7   <p>Not selected</p>
8 </div>
```

```
1 p:nth-of-type(2) {
2   color: #f00;
3 }
```

`:nth-of-type(n)`

Works the same as `:nth-child(n)`, but only counts element of a specific type

```
1 <div>
2   <p>Not selected</p>
3   <span>Not selected</span>
4   <p>Selected</p>
5   <p>Not selected</p>
6   <p>Not selected</p>
7   <p>Not selected</p>
8 </div>
```

```
1 p:nth-of-type(2) {
2   color: #f00;
3 }
```

`:nth-last-of-type(n)`

Works the same as `:nth-last-child(n)`, but only counts element of a specific type

```
1 <div>
2   <p>Not selected</p>
3   <p>Not selected</p>
4   <p>Not selected</p>
5   <p>Selected</p>
6   <div>Not selected</div>
7     <p>Not selected</p>
8 </div>
```

```
1 p:nth-last-of-type(2) {
2   color: #f00;
3 }
```

`:nth-last-of-type(n)`

Works the same as `:nth-last-child(n)`, but only counts element of a specific type

```
1 <div>
2   <p>Not selected</p>
3   <p>Not selected</p>
4   <p>Not selected</p>
5   <p>Selected</p>
6   <div>Not selected</div>
7   <p>Not selected</p>
8 </div>
```

```
1 p:nth-last-of-type(2) {
2   color: #f00;
3 }
```

`:nth-last-of-type(n)`

Works the same as `:nth-last-child(n)`, but only counts element of a specific type

```
1 <div>
2   <p>Not selected</p>
3   <p>Not selected</p>
4   <p>Not selected</p>
5   <p>Selected</p>
6   <div>Not selected</div>
7   <p>Not selected</p>
8 </div>
```

```
1 p:nth-last-of-type(2) {
2   color: #f00;
3 }
```

:only-child

Selects every element that is the only child element of its parent element.

```
1 <div>
2     <p>Selected</p>
3 </div>
4 <div>
5     <p>Not Selected</p>
6     <div>Not Selected</div>
7 </div>
```

```
1 p:only-child {
2     color: #f00;
3 }
```

:only-child

Selects every element that is the only child element of its parent element.

```
1 <div>
2     <p>Selected</p>
3 </div>
4 <div>
5     <p>Not Selected</p>
6     <div>Not Selected</div>
7 </div>
```

```
1 p:only-child {
2     color: #f00;
3 }
```

:only-of-type

Selects every element that is the only child element of a given type of its parent element.

```
1 <div>
2   <p>Selected</p>
3 </div>
4 <div>
5   <p>Selected</p>
6   <div>Not Selected</div>
7 </div>
8 <div>
9   <p>Not Selected</p>
10  <p>Not Selected</p>
11   <div>Not Selected</div>
12 </div>
```

```
1 p:only-of-type {
2   color: #f00;
3 }
```


:only-of-type

Selects every element that is the only child element of a given type of its parent element.

```
1 <div>
2   <p>Selected</p>
3 </div>
4 <div>
5   <p>Selected</p>
6   <div>Not Selected</div>
7 </div>
8 <div>
9   <p>Not Selected</p>
10  <p>Not Selected</p>
11  <div>Not Selected</div>
12 </div>
```

```
1 p:only-of-type {
2   color: #f00;
3 }
```


PSEUDO ELEMENTS

- Selectors which allow you to access parts of your document without having to add extra classes or elements
- Elements are not present in the DOM, only in render tree
- Can't be selected by JavaScript

::after

This selector allows you to target the part after your element

```
<p>Hello</p>
```

```
p::after {  
  content: ' world!';  
  font-weight: bold;  
}
```

::before

This selector allows you to target the part before your element

```
<p class="name">Yannick</p>
```

```
p::before {  
    content: 'Hallo, ';  
    font-weight: bold;  
}
```

::first-letter

This selector allows you to target the first letter of your element

```
<p>Hello, world!</p>
```

```
p::first-letter {  
    font-size: 20px;  
    font-weight: bold;  
}
```

::first-line

This selector allows you to target the first line of your element.

```
<p>Hello,<br/> world!</p>
```

```
p::first-line {  
    font-size: 20px;  
    font-weight: bold;  
}
```

::selection

This selector allows you to target the selected text of your element

```
<p>Hello, world!</p>
```

```
p::selection {  
    color: #fff;  
    background: #f00;  
}
```

COMBINING SELECTORS

- For each element inside the DOM, the browser will check every selector to see if it matches the element
- By starting the matching of the selector on the right side, this process will be much more performant

```
/*
the CSS parser will read this starting from the right.
first it will collect all divs, then all divs in .quux,
then all divs in .baz who are also in .bar and so on.
*/
.foo .bar .baz. .quux div {
    // ...
}
```

SELECTOR LIST

List of multiple selectors, which will apply the style on all elements that match at least once

```
1 <div class="foo">yep</div>
2 <div class="bar">yep</div>
3 <div class="baz">nope</div>
```

```
1 .foo, .bar {
2     color: #f00;
3 }
```

SELECTOR LIST

List of multiple selectors, which will apply the style on all elements that match at least once

```
1 <div class="foo">yep</div>
2 <div class="bar">yep</div>
3 <div class="baz">nope</div>
```

```
1 .foo, .bar {
2     color: #f00;
3 }
```

SELECTOR COMBINATORS

DESCENDANT COMBINATOR

matches all elements that are descendants of a specified element

```
1 <div class="example">
2   <p>text</p>
3   <div>
4     <span>text</span>
5     <p>text</p>
6   </div>
7 </div>
8 <div class="other">
9   <p>text</p>
10 </div>
```

```
1 .example p {
2   background-color: #f00;
3 }
```

DESCENDANT COMBINATOR

matches all elements that are descendants of a specified element

```
1 <div class="example">
2   <p>text</p>
3   <div>
4     <span>text</span>
5     <p>text</p>
6   </div>
7 </div>
8 <div class="other">
9   <p>text</p>
10 </div>
```

```
1 .example p {
2   background-color: #f00;
3 }
```

CHILD COMBINATOR

matches all elements that are direct children of specified element

```
1 <div class="example">
2   <p>text</p>
3   <div>
4     <span>text</span>
5     <p>text</p>
6   </div>
7 </div>
8 <div class="other">
9   <p>text</p>
10 </div>
```

```
1 .example > p {
2   background-color: #f00;
3 }
```

CHILD COMBINATOR

matches all elements that are direct children of specified element

```
1 <div class="example">
2   <p>text</p>
3   <div>
4     <span>text</span>
5     <p>text</p>
6   </div>
7 </div>
8 <div class="other">
9   <p>text</p>
10 </div>
```

```
1 .example > p {
2   background-color: #f00;
3 }
```

GENERAL SIBLING COMBINATOR

separates two selectors and matches all iterations of the second element, that are following the first element

```
1 <div>
2   <p>text</p>
3   <p class="foo">text</p>
4   <p>text</p>
5   <span>other text</span>
6   <p>text</p>
7 </div>
```

```
1 .foo ~ p {
2   background-color: #f00;
3 }
```

GENERAL SIBLING COMBINATOR

separates two selectors and matches all iterations of the second element, that are following the first element

```
1 <div>
2   <p>text</p>
3   <p class="foo">text</p>
4   <p>text</p>
5   <span>other text</span>
6   <p>text</p>
7 </div>
```

```
1 .foo ~ p {
2   background-color: #f00;
3 }
```

ADJACENT SIBLING COMBINATOR

separates two selectors and matches the second element only if it immediately follows the first element, and both are children of the same parent element

```
1 <div>
2   <p>text</p>
3   <p class="foo">text</p>
4   <p>text</p>
5   <span>other text</span>
6   <p>text</p>
7 </div>
```

```
1 .foo ~ p {
2   background-color: #f00;
3 }
```

ADJACENT SIBLING COMBINATOR

separates two selectors and matches the second element only if it immediately follows the first element, and both are children of the same parent element

```
1 <div>
2   <p>text</p>
3   <p class="foo">text</p>
4   <p>text</p>
5   <span>other text</span>
6   <p>text</p>
7 </div>
```

```
1 .foo ~ p {
2   background-color: #f00;
3 }
```

CONCEPTS

CASCADE

- Styles applied top to bottom, which allows styles to be added or overridden
- Every browser has default styles, so you use this all the time

```
<p>Lorem ipsum dolor sit amet.</p>
```

```
1 p {  
2     background: #f00;  
3     background: #0f0;  
4 }  
5 p {  
6     background: #00f;  
7 }
```

CASCADE

- Styles applied top to bottom, which allows styles to be added or overridden
- Every browser has default styles, so you use this all the time

```
<p>Lorem ipsum dolor sit amet.</p>
```

```
1 p {  
2     background: #f00;  
3     background: #0f0;  
4 }  
5 p {  
6     background: #00f;  
7 }
```

SOURCE ORDER

- Source order also plays role in parsing of the CSS
- Last one will always win

SPECIFICITY

Five levels of specificity in CSS

- css rule followed by `! important`
- inline CSS using `style` attribute
- ID selectors
- Class selectors (incl pseudo class selectors and attribute selectors)
- Element selectors (incl pseudo element selectors)

INHERITANCE

Apart from cascading, inheritance is the phenomenon that elements inherit properties without being explicitly defined in the CSS

```
div {  
    font-size: 14px;  
}
```

```
<div>  
    I have a font-size of 14px.  
    <p>  
        But so have I!  
    </p>  
</div>
```

VALUES & UNITS

COLORS

KEYWORD

- Keyword colors, mapped to a specific value
- Examples: `black`, `green`, `darkblue`, `salmon`...
- Limited use (+- 165 keyword colors), so not recommended way

```
p {  
    color: hotpink;  
}
```

- 2 special keywords
 - `transparent`: sets color to transparent
 - `currentColor`: represents the value of elements `color` property, so you can use it on other properties of same element

HEXADECIMAL

- Hexadecimal colors, combination of hash **#** followed by 3, 6 or 8 characters (0-9a-f)
- **#RRGGBB [AA]**: reds, greens, blues, (optional) alpha channel
- In case of repeating values, you can shorthand it (**#fffb33 -> #fb3**)

```
p {  
    color: #bada55;  
    background: #fb3;  
    text-shadow: #00000044;  
}
```

RGB & RGBA

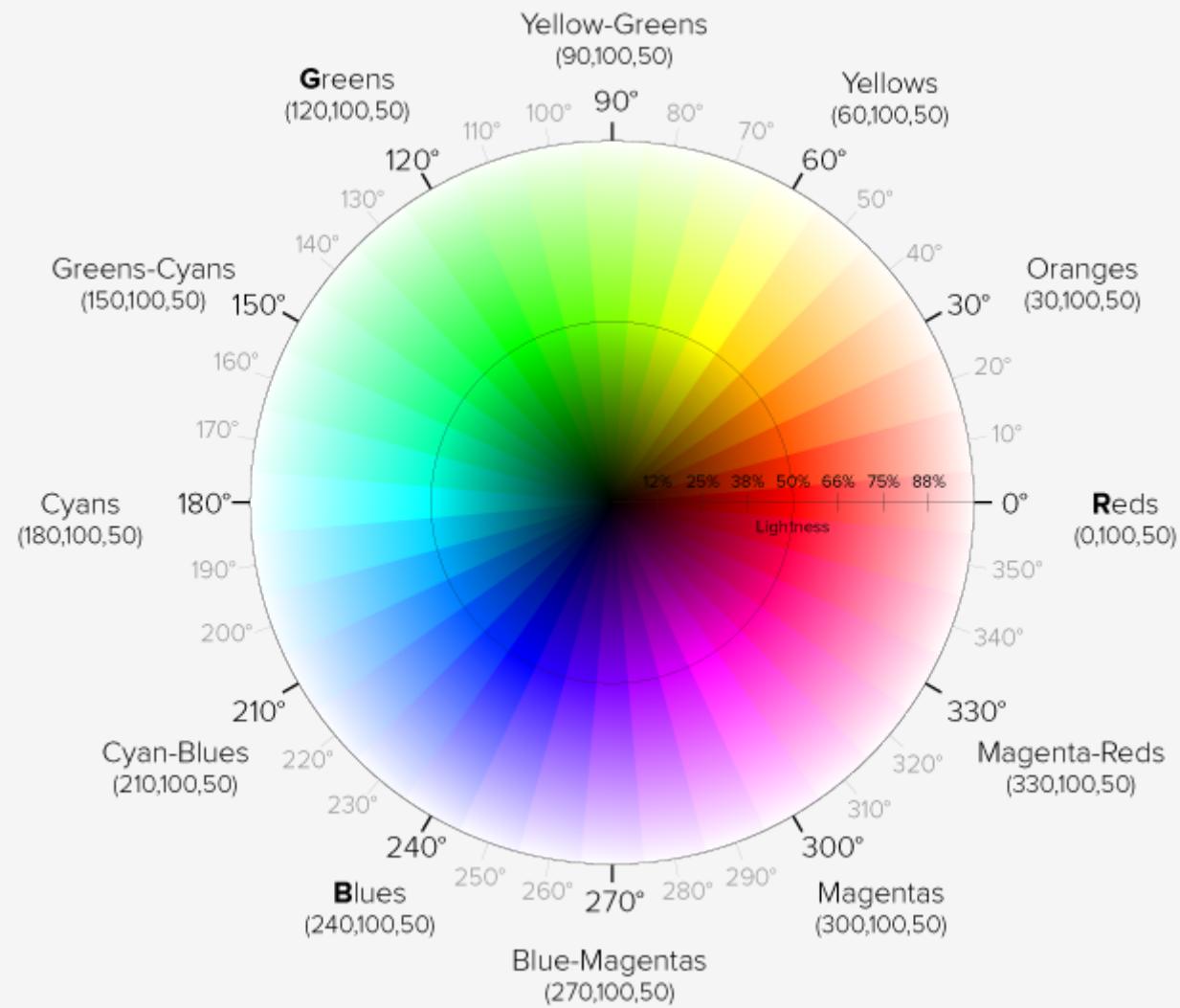
- CSS function `rgb()` and `rgba()`, which accept respectably 3 or 4 parameters
- `rgb(red, green, blue)` where each parameter defines intensity of that color
 - Integer between 0 and 255 or percentage value (0% to 100%)
- `rgba(red, green, blue, alpha)` same as `rgb()` but has extra alpha channel
 - Value between 0 and 1

```
p {  
    color: rgb(186, 218, 85);  
    background: rgba(255, 187, 51, 0.44);  
}
```

HSL & HSLA

- Similar to the `rgb()` and `rgba()` CSS functions, but params represent hue, saturation and lightness
- `hsl(hue, saturation, light)`
 - *hue* is integer between 0 - 360 (color wheel)
 - *saturation* is percentage (0% is shade of gray, 100% is color)
 - *lightness* is percentage (0% black, 100% white, 50% normal)

```
p {  
  background-color: hsla(180, 50%, 50%, .42);  
  color: hsl(0, 0%, 100%);  
}
```



NUMERIC

Different numeric values

- Absolute units
- Relative units

ABSOLUTE UNITS

- Most known absolute length unit in CSS is **px**
- one pixel = one dot on the screen
- Other absolute units are **mm**, **cm**, **in**, **pt** and **pc**, but are less used because no relationship to screen.

RELATIVE UNITS

- **em** is most frequently used font based value
 - represents the font-size (or inherited font-size) of element
 - $2\text{em} = 2 * \text{font-size}$
- **rem** is similar to **em**, but represents font-size of the root element
- **%** is relative to the size of the parent element
- Others are **ex**, **ch** (rarely used)

- Instead relative to font-size or parent, also viewport relative values
- `vh` and `vw` represent 1% of either the viewport height or width.
- `vmin` and `vmax` represent 1% of the smaller / larger dimension of viewport

UNITLESS

Sometimes unitless is okay and valid

```
.foo {  
  /* setting a value to 0 is always valid without a unit  
   since 0px and 0em has the same outcome */  
  margin: 0;  
  
  /* unitless line-height represent a multiplying factor  
   based on the inherited line-height */  
  line-height: 1.5;  
  
  /* in case of animation counts it is just a simple  
   number that sets the how many times the animation  
   should play */  
  animation-iteration-count: 5;  
}
```

VARIABLES

CSS also provides a way to set variables and re-use/re-assign them on the fly.

```
:root {  
    --somecolor: #f00;  
}  
  
.foo {  
    color: var(--somecolor);  
}
```

FUNCTIONS

- CSS has functions build-in
- We can't create our own (or can we?)

```
.foo {  
  /* calculate the new position of an element after it  
   has been rotated by 45 degress */  
  transform: rotate(45deg);  
  
  /* calculate the computed value of 90% of the current  
   width minus 15px */  
  width: calc(90% - 15px);  
  
  /* create a gradient and use it as a background */  
  background-image: linear-gradient(to left, teal, red);  
}
```

CSS STYLES

BASICS

BACKGROUND

```
1 .foo {  
2   background-color: #f00;  
3   background-image: url("image.png");  
4   background-position: 50px 25px;  
5   background-position: left bottom;  
6   background-repeat: repeat;  
7             /* no-repeat, repeat-x, repeat-y */  
8  
9   background: #f00 url("image.png") left bottom repeat;  
10 }
```

BACKGROUND

```
1 .foo {  
2   background-color: #f00;  
3   background-image: url("image.png");  
4   background-position: 50px 25px;  
5   background-position: left bottom;  
6   background-repeat: repeat;  
7     /* no-repeat, repeat-x, repeat-y */  
8  
9   background: #f00 url("image.png") left bottom repeat;  
10 }
```

BACKGROUND

```
1 .foo {  
2   background-color: #f00;  
3   background-image: url("image.png");  
4   background-position: 50px 25px;  
5   background-position: left bottom;  
6   background-repeat: repeat;  
7     /* no-repeat, repeat-x, repeat-y */  
8  
9   background: #f00 url("image.png") left bottom repeat;  
10 }
```

BACKGROUND

```
1 .foo {  
2   background-color: #f00;  
3   background-image: url("image.png");  
4   background-position: 50px 25px;  
5   background-position: left bottom;  
6   background-repeat: repeat;  
7             /* no-repeat, repeat-x, repeat-y */  
8  
9   background: #f00 url("image.png") left bottom repeat;  
10 }
```

BACKGROUND

```
1 .foo {  
2   background-color: #f00;  
3   background-image: url("image.png");  
4   background-position: 50px 25px;  
5   background-position: left bottom;  
6   background-repeat: repeat;  
7     /* no-repeat, repeat-x, repeat-y */  
8  
9   background: #f00 url("image.png") left bottom repeat;  
10 }
```

BACKGROUND GRADIENTS

It's possible to set gradients as `background-image`

```
.foo {  
  background-image: linear-gradient(to left, red, blue);  
  background-image: radial-gradient(crimson, skyblue);  
}
```

A single element can have multiple backgrounds

```
.foo {  
  background-image:  
    url("image"),  
    linear-gradient(to right, black, yellow, red);  
}
```

BORDER

- sets visible border between margin & padding
- default size of 0, so not visible
- border consist of 3 things
 - width
 - style (solid, dashed, double, ...)
 - color

```
.foo {  
  border: 1px solid #f00;  
  /* ... */  
  border-color: #f00;  
  border-style: solid;  
  border-width: 1px;  
}
```

```
.foo {  
  /* if you want to go funky */  
  border: 5px solid olivedrab;  
  border-bottom-style: dot-dot-dash;  
  border-top-color: olive;  
  border-left-width: 3px;  
  border-right: 4px dashed goldenrod;  
}
```

BORDER-RADIUS

Makes rounded corners on the box

```
.foo {  
  border-radius: 10px;  
}
```

Also possible to define elliptical borders by providing 2 radii, separated by a slash /. The radii define the x and y radius of the corner

```
.foo {  
  border-radius: 10px / 20px;  
  border-radius: 4px 3px 6px / 2px 4px;  
}
```

BOX-SHADOW

box-shadow allows you to set one or more shadows on an element.

Value exists of 2 to 4 length values, optional color value and optional **inset** keyword

```
.foo {  
  box-shadow: 5px 5px 5px rgba(0, 0, 0, .2);  
  box-shadow: inset 5px 5px 5px 5px rgba(0, 0, 0, .2);  
}
```

It's possible to define multiple shadows on a single element

```
.foo {  
  box-shadow:  
    -5px -5px 5px rgba(0, 0, 0, .2),  
    5px 5px 5px rgba(0, 0, 0, .2);  
}
```

BOXES

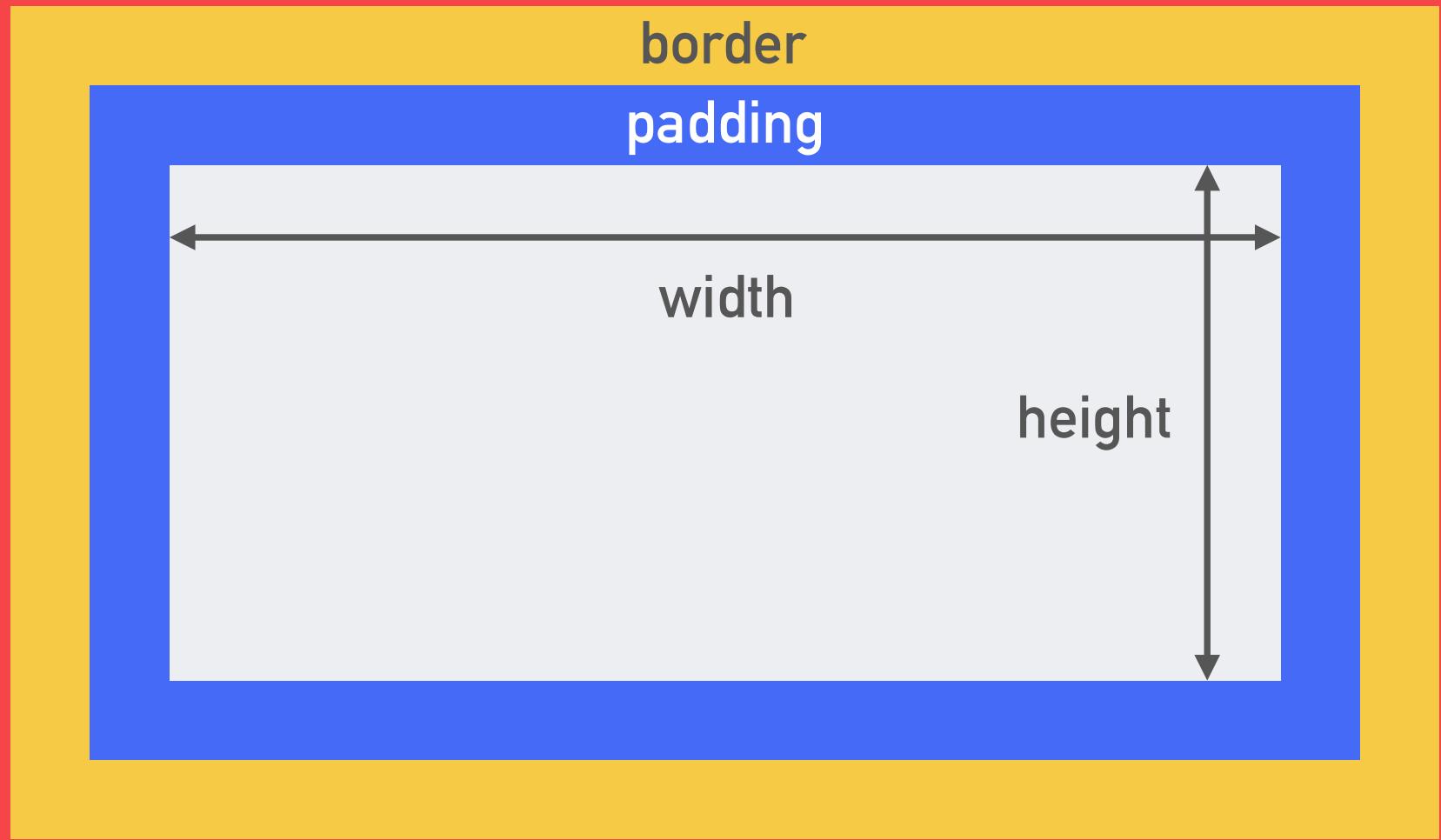
margin

border

padding

width

height



CSS BOX MODEL

- **width & height** define dimensions of the content box
- **padding** is inner spacing of the content box
- **border** sits on the outside of the content box.
 - By default, not visible, enable through **border** property
- **margin** is outer spacing around the element
 - margins "collapse", when 2 boxes (with margin) touch, the space is largest of the 2

```
div {  
    border: 5px solid red;  
    height: 50px;  
    margin: 10px 20px 15px;  
    padding: 5px 10px;  
    width: 25%;  
}
```

DISPLAY TYPES

```
1 div {  
2   display: block;  
3   display: inline;  
4   display: inline-block;  
5   display: flex;  
6   display: inline-flex;  
7   display: grid;  
8   display: inline-grid;  
9  
10  display: none;  
11  
12  display: table;  
13  display: list-item;  
14 }
```

DISPLAY TYPES

```
1 div {  
2   display: block;  
3   display: inline;  
4   display: inline-block;  
5   display: flex;  
6   display: inline-flex;  
7   display: grid;  
8   display: inline-grid;  
9  
10  display: none;  
11  
12  display: table;  
13  display: list-item;  
14 }
```

DISPLAY TYPES

```
1 div {  
2   display: block;  
3   display: inline;  
4   display: inline-block;  
5   display: flex;  
6   display: inline-flex;  
7   display: grid;  
8   display: inline-grid;  
9  
10  display: none;  
11  
12  display: table;  
13  display: list-item;  
14 }
```

DISPLAY TYPES

```
1 div {  
2   display: block;  
3   display: inline;  
4   display: inline-block;  
5   display: flex;  
6   display: inline-flex;  
7   display: grid;  
8   display: inline-grid;  
9  
10  display: none;  
11  
12  display: table;  
13  display: list-item;  
14 }
```

DISPLAY TYPES

```
1 div {  
2   display: block;  
3   display: inline;  
4   display: inline-block;  
5   display: flex;  
6   display: inline-flex;  
7   display: grid;  
8   display: inline-grid;  
9  
10  display: none;  
11  
12  display: table;  
13  display: list-item;  
14 }
```

DISPLAY TYPES

```
1 div {  
2   display: block;  
3   display: inline;  
4   display: inline-block;  
5   display: flex;  
6   display: inline-flex;  
7   display: grid;  
8   display: inline-grid;  
9  
10  display: none;  
11  
12  display: table;  
13  display: list-item;  
14 }
```

BOX-SIZING PROPERTY

- **box-sizing** property sets how total width & height is calculated
- **content-box**
 - default & initial value
 - dimensions are calculated as: $width = width \text{ of content}$, $height = height \text{ of content}$
- **border-box**
 - dimensions are calculated as: $width = border + padding + width \text{ of content}$, $height = border + padding + height \text{ of content}$

FONTS & TEXT

FONT-FAMILY

- Property declares which font you want to use.
- Accepts multiple, comma-separated values

```
.foo {  
  font-family: Verdana, Arial, Helvetica, sans-serif;  
}
```

FONTSIZE

Sets size of font

```
.foo {  
  font-size: 14px;  
}
```

FONT-STYLE

Sets font to italic

```
.foo {  
  font-style: normal;  
  font-style: italic;  
  font-style: oblique;  
}
```

FONT-WEIGHT

Sets the weight/thickness of the font

```
.foo {  
  font-weight: normal;  
  font-weight: bold;  
  
  /* relative to parent */  
  font-weight: lighter;  
  font-weight: bolder;  
}
```

You can also set the value ranging from 100 to 900, with steps of 100, where 100 is the lightest and 900 the heaviest.

```
.foo {  
  font-weight: 100;  
  font-weight: 200;  
  font-weight: 300;  
  font-weight: 400; /* normal */  
  font-weight: 500;  
  font-weight: 600;  
  font-weight: 700; /* bold */  
  font-weight: 800;  
  font-weight: 900;  
}
```

LINE-HEIGHT

- Sets the distance between 2 lines of text
- Most browsers set a default `line-height` of 1.2 (multiplying factor)

```
.foo {  
  line-height: 1.5;  
  
  line-height: 30px;  
  height: 30px;  
}
```

Sometimes used to center text vertically by setting line-height same as height of block

TEXT-DECORATION

- Allows adding extra styling to text (like underlines)
- By default, no text-decoration
 - with some exceptions like the <a>-tag

```
.foo {  
    text-decoration: underline;  
  
    text-decoration: wavy overline lime;  
    text-decoration: underline dotted red 3px;  
}
```

FEW HONOURABLE MENTIONS

- **font-variant**: specifies whether or not a text should be displayed in a small-caps font.
- **letter-spacing**: increases or decreases the space between characters in a text.
- **text-transform**: controls the capitalization of text

CUSTOM FONTS

Not limited to system fonts, we can embed our own using @font-face at rule

```
@font-face {  
    font-family: "Open Sans";  
    src:  
        url("/fonts/OpenSans-Regular.woff2") format("woff2"),  
        url("/fonts/OpenSans-Regular.woff") format("woff");  
}
```

```
@font-face {  
    font-family: MyHelvetica;  
    src: local("Helvetica Neue"),  
        local("HelveticaNeue"),  
        url(MgOpenModernaRegular.ttf);  
}  
  
@font-face {  
    font-family: MyHelvetica;  
    src: local("Helvetica Neue Bold"),  
        local("HelveticaNeue-Bold"),  
        url(MgOpenModernaBold.ttf);  
    font-weight: bold;  
}
```

```
@font-face {  
    font-family: 'Dosis';  
    src: url('Dosis-VarFont_wght.ttf')  
         format("truetype-variations");  
    font-weight: 125 950;  
    font-stretch: 75% 125%;  
    font-style: oblique 0deg 20deg;  
}
```

LAYOUT

POSITIONING

static

- Positioned according to the normal flow of the document
 - `top`, `right`, `bottom`, `left` and `z-index` have no effect

```
.element {  
  position: static; /* default */  
}
```

relative

- Acts if it has no positioning.
 - It changes positioning, but does not change surrounding layout

```
.element {  
  position: relative;  
}
```

absolute

- Reserves no space for the element
- Relative positioning to its closest positioned parent

```
.element {  
  position: absolute;  
  left: 40px;  
  top: 20%  
}
```

fixed

- Reserves no space for the element
- Relative positioning to viewport (keeps same location when scrolled)

```
.element {  
  position: fixed;  
  left: 40px;  
  top: 20%  
}
```

sticky

- Relative positioning until element crosses certain point

```
.header {  
  position: sticky;  
  top: 30px  
}
```

`z-index`

- Used when elements start overlapping each other
- By default, order by placing in HTML
- With `z-index` this stacking order is overruled

```
.foo {  
  z-index: 2;  
}
```

FLOAT

- The way to create grid and layouts before modern solutions
- Originally designed to float images in text

Float property specifies that element should be taken from regular flow and should be positioned to left or right side of it's container or another floating element

```
.foo {  
    float: none | left | right;  
}
```

Clearing floats defines the element should be next to another floating element or should move down below them.

```
.quux {  
  clear: none | left | right | both;  
}
```

- When floating a large element that sits in the same container as a smaller element, the parent container does not wrap around the size of the floating container.
- Solution: clearfix hack

```
.wrapper::after {  
  clear: both;  
  content: '';  
  display: block;  
}
```

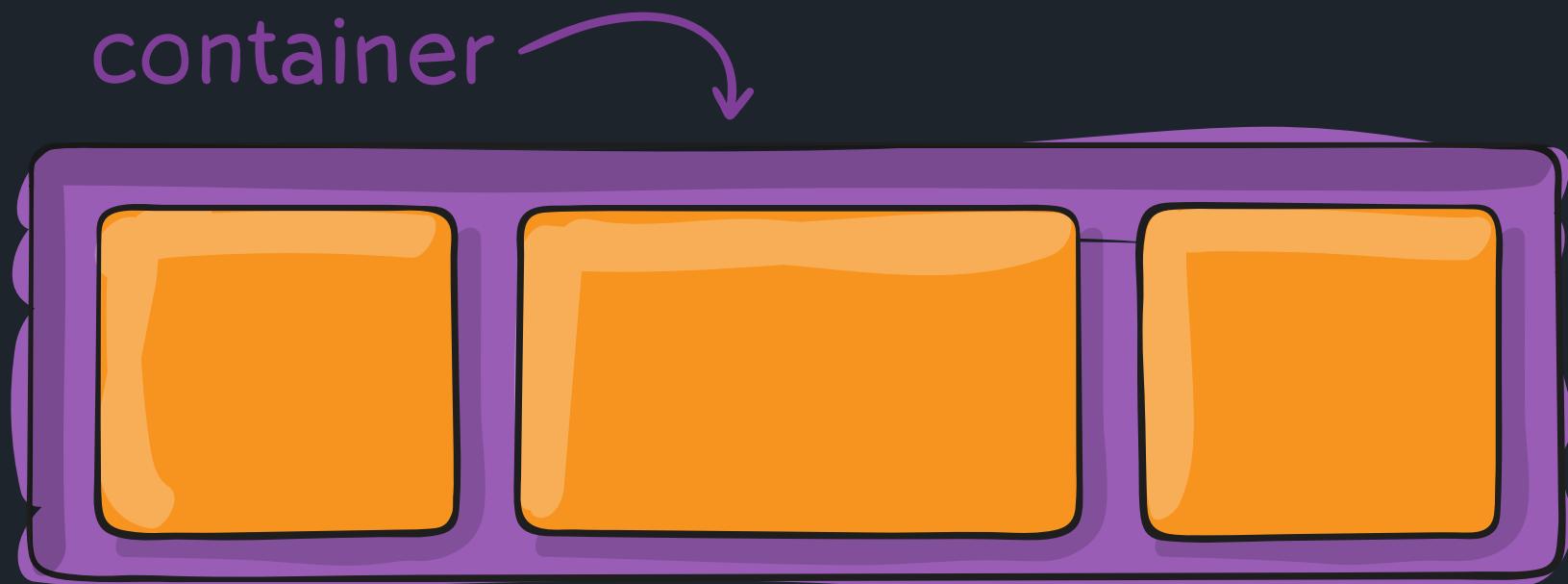
- More modern approach: flow-root
- It creates a block formatting context which contains the floating element.

```
.wrapper {  
    display: flow-root;  
}
```

LAYOUT: FLEXBOX

- More modern approach to layouts of webpages
- Ability to alter its items width/height (and order) to best fill the available space
- Expands items to fill available free space or shrinks them to prevent overflow
- Flexbox is a whole module and not a single property

FLEX CONTAINER (PARENT)



```
.container {  
  display: flex; /* or inline-flex */  
}
```

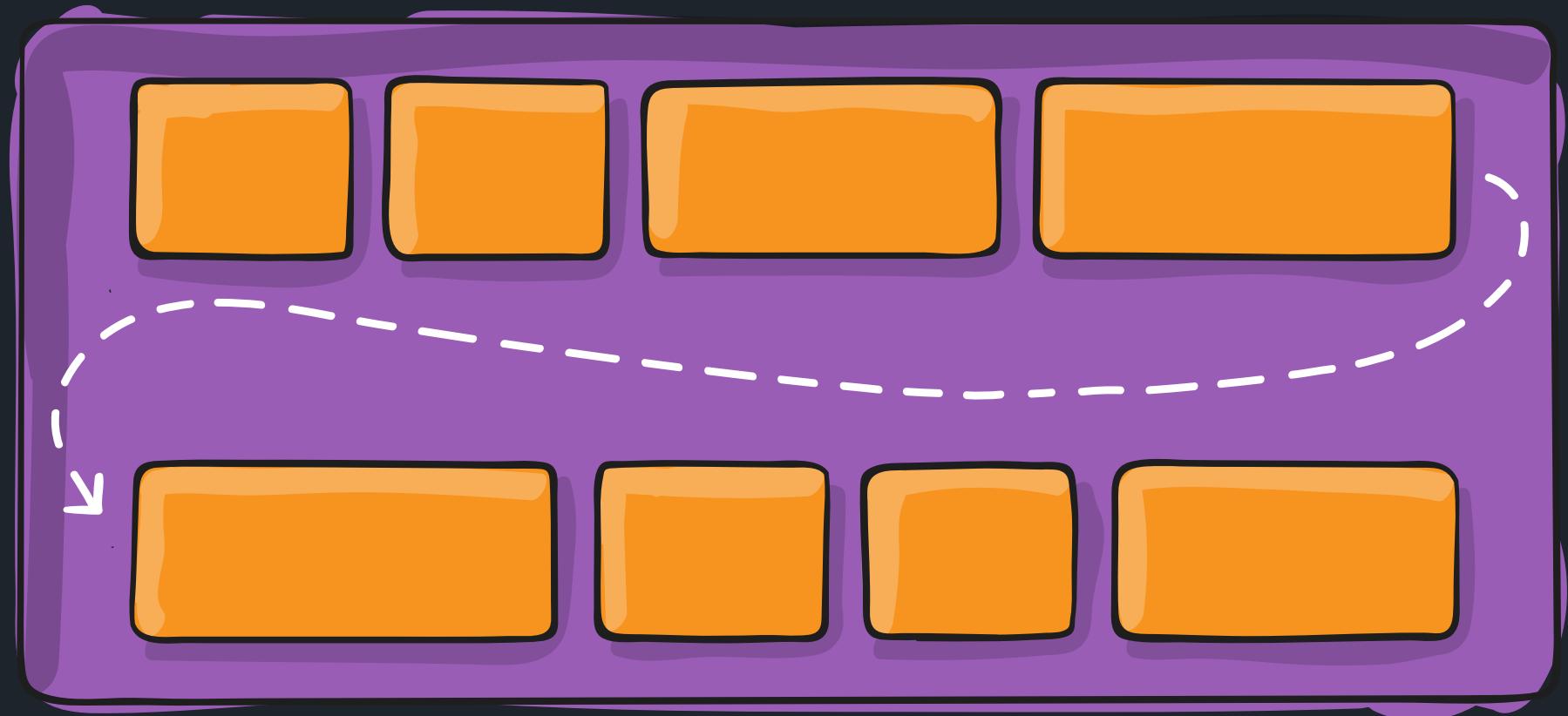
FLEX-DIRECTION



flex-direction: column;

```
.container {  
  flex-direction: row | row-reverse | column | column-reverse  
}
```

FLEX-WRAP



```
.container {  
  flex-wrap: nowrap | wrap | wrap-reverse;  
}
```

FLEX-FLOW

- Shorthand for the `flex-direction` and `flex-wrap` properties
- Defines the flex containers main and cross axes
- Default: `row nowrap`

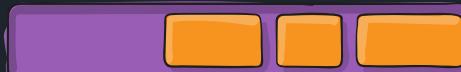
```
.container {  
  flex-flow: column wrap;  
}
```

JUSTIFY CONTENT

flex-start



flex-end



center



space-between



space-around



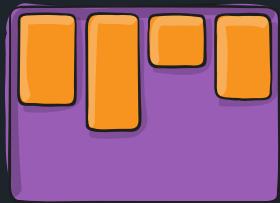
space-evenly




```
.container {  
  justify-content: flex-start  
    | flex-end  
    | center  
    | space-between  
    | space-around  
    | space-evenly;  
}
```

ALIGN-ITEMS

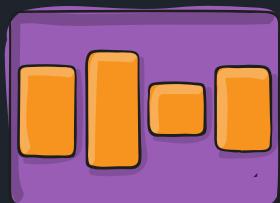
flex-start



flex-end



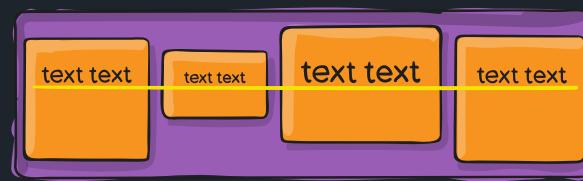
center



stretch



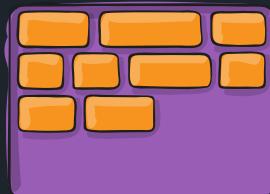
baseline



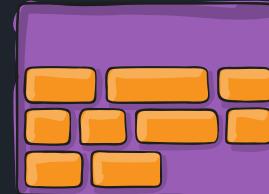

```
.container {  
  align-items: stretch  
    | flex-start  
    | flex-end  
    | center  
    | baseline  
    | first baseline  
    | last baseline;  
}
```

ALIGN-CONTENT

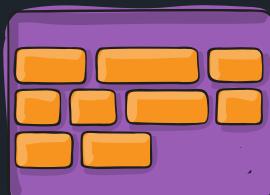
flex-start



flex-end



center



stretch



space-between

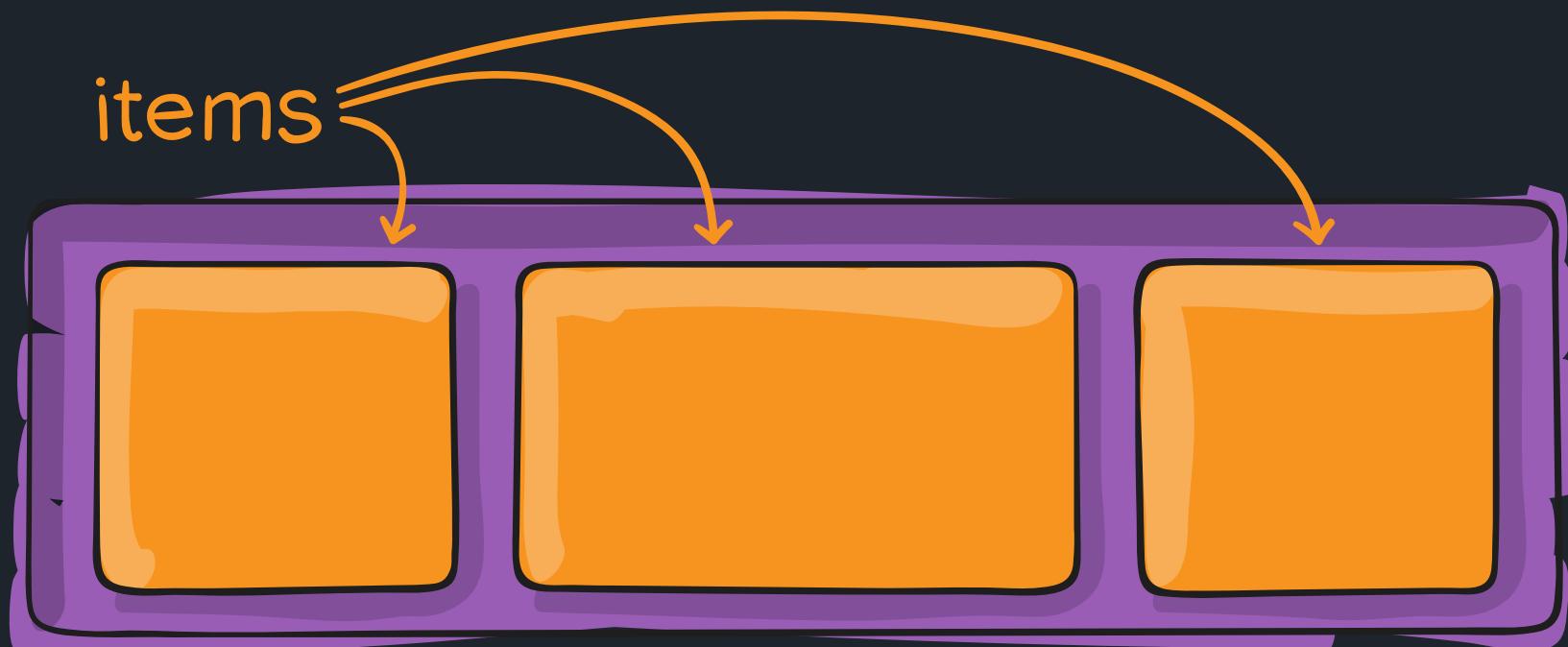


space-around

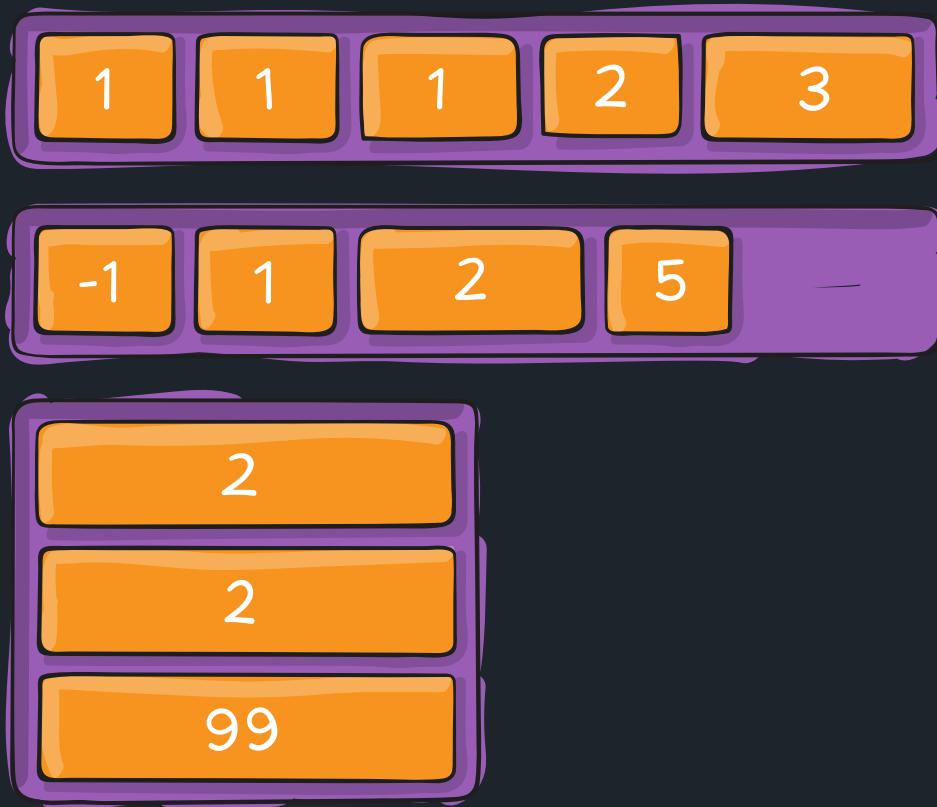



```
.container {  
  align-content: flex-start  
    | flex-end  
    | center  
    | space-between  
    | space-around  
    | space-evenly  
    | stretch;  
}
```

FLEX ITEMS (CHILDREN)

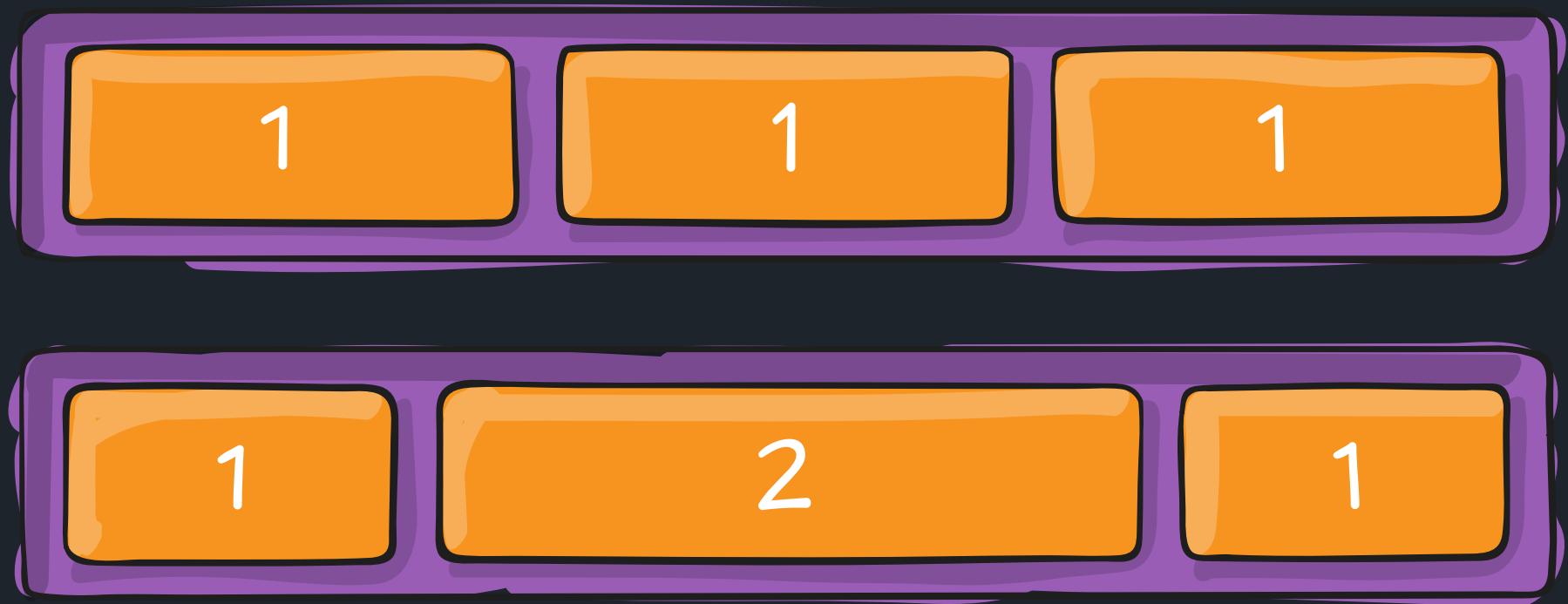


ORDER




```
.item {  
  order: 5; /* default is 0 */  
}
```

FLEX-GROW



```
.item {  
  flex-grow: 4; /* default 0 */  
}
```

FLEX-SHRINK

```
.item {  
  flex-shrink: 3; /* default 1 */  
}
```

FLEX-BASIS

```
.item {  
  flex-basis:  | auto; /* default auto */  
}
```

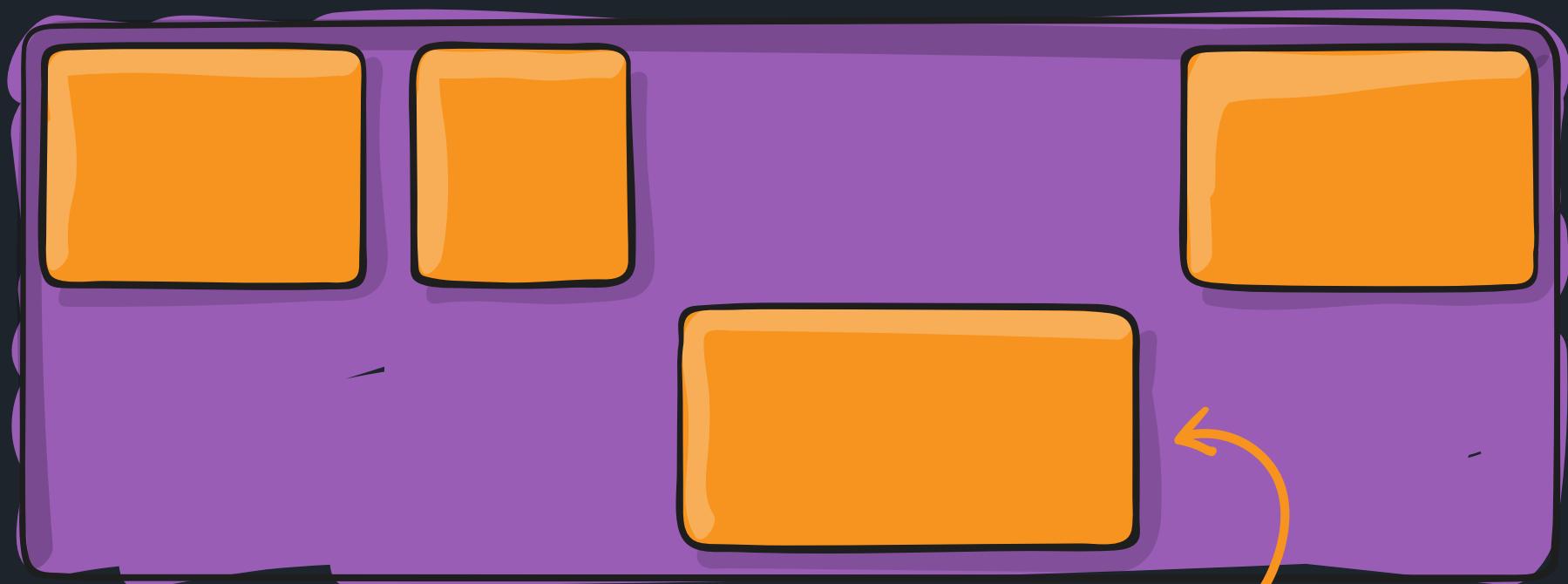
FLEX

- Shorthand for flex-grow, flex-shrink and flex-basis combined
- Default is 0 1 auto

```
.item {  
  flex: none | [ <'flex-grow'> <'flex-shrink'>? || <'flex-bas  
 }  
}
```

ALIGN-SELF

flex-start



flex-end

- allows the default alignment (or the one specified by align-items) to be overridden for individual flex items

```
.item {  
  align-self: auto  
    | flex-start  
    | flex-end  
    | center  
    | baseline  
    | stretch;  
}
```

<https://flexboxfroggy.com/>

LAYOUT: GRID

- Grid Layout is a two-dimensional grid-based layout system
- Different use-cases than flexbox (which has a more one-directional flow)

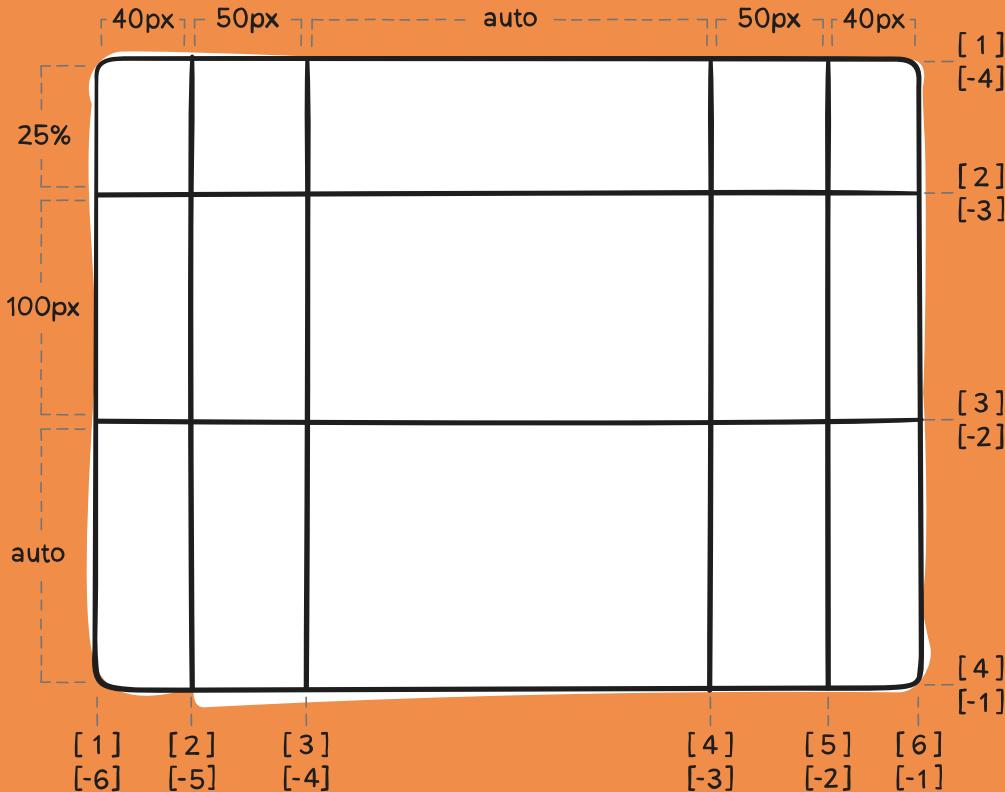
GRID CONTAINER (PARENT)

```
.container {  
  display: grid | inline-grid;  
}
```

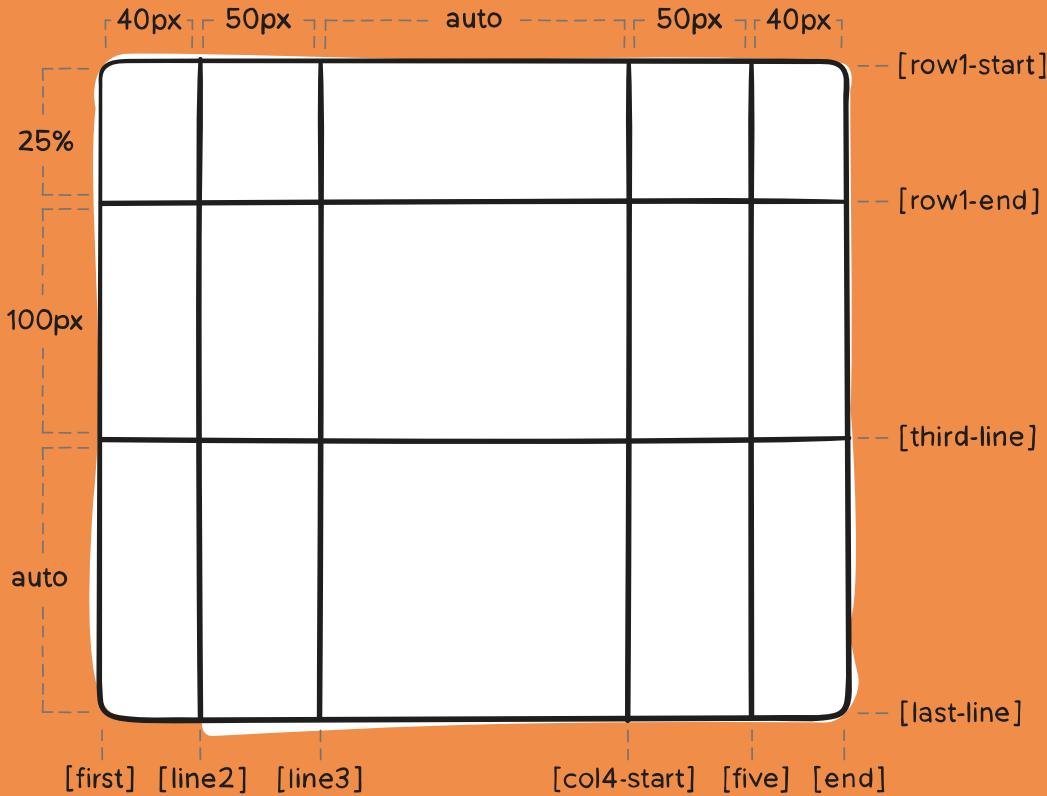
grid-template-columns & grid-templates-rows

Defines the columns and rows of the grid with a space-separated list of values. The values represent the track size, and the space between them represents the grid line.

```
.container {  
    grid-template-columns: ... ...;  
    /* e.g.  
       1fr 1fr  
       minmax(10px, 1fr) 3fr  
       repeat(5, 1fr)  
       50px auto 100px 1fr  
    */  
    grid-template-rows: ... ...;  
    /* e.g.  
       min-content 1fr min-content  
       100px 1fr max-content  
    */  
}
```




```
.container {  
    grid-template-columns: [first] 40px [line2] 50px [line3] au  
    grid-template-rows: [row1-start] 25% [row1-end] 100px [thir  
}  
}
```



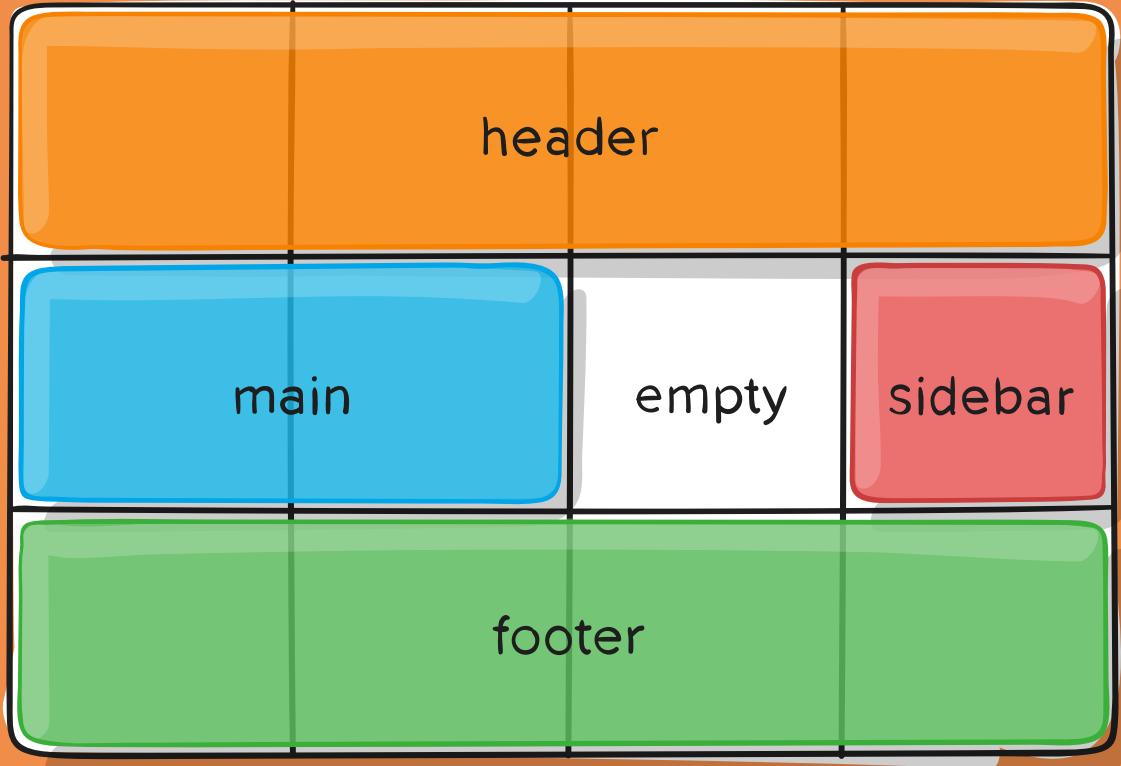
grid-template-areas

Defines a grid template by referencing the names of the grid areas which are specified with the grid-area property.

```
.container {  
  grid-template-areas:  
    "<grid-area-name> | . | none | ..."   
    "...";  
}
```

```
.item-a {
  grid-area: header;
}
.item-b {
  grid-area: main;
}
.item-c {
  grid-area: sidebar;
}
.item-d {
  grid-area: footer;
}

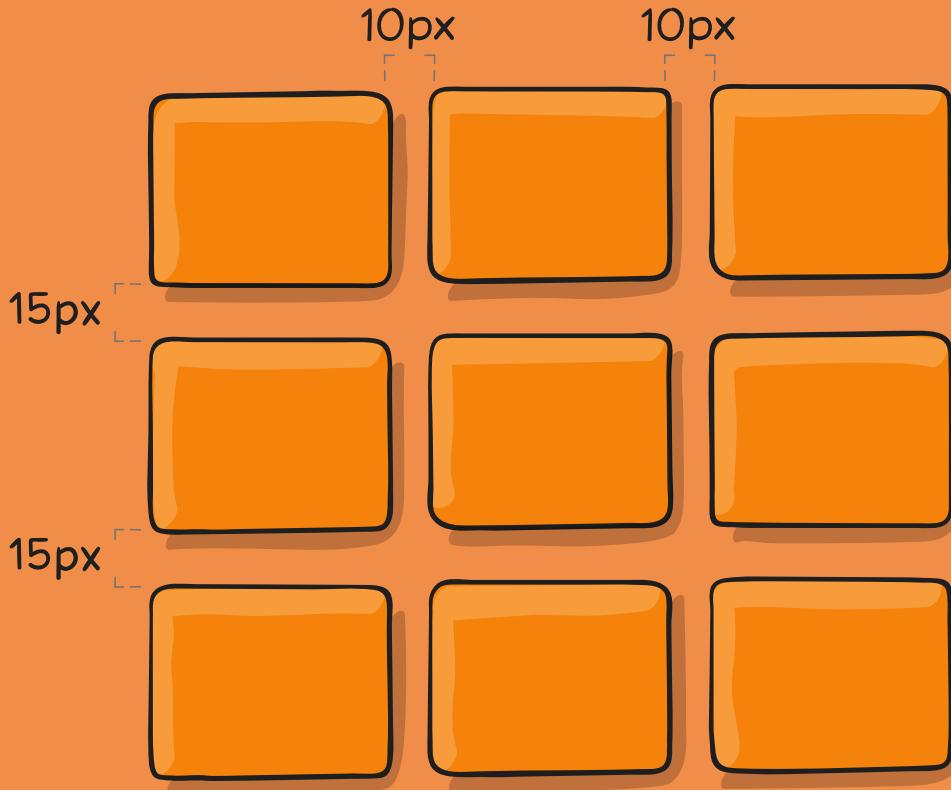
.container {
  display: grid;
  grid-template-columns: 50px 50px 50px 50px;
  grid-template-rows: auto;
  grid-template-areas:
    "header header header header"
    "main main . sidebar"
    "footer footer footer footer";
}
```



column-gap & row-gap

Specifies the size of the grid lines. You can think of it like setting the width of the gutters between the columns/rows.

```
.container {  
  column-gap: 10px;  
  row-gap: 15px;  
}
```



gap

- shorthand for `row-gap` & `column-gap`

```
.container {  
  gap: <grid-row-gap> <grid-column-gap>;  
}
```

justify-items

- Aligns items along the inline (row) axis
- Value applies to all grid items inside the container

```
.container {  
  justify-items: start | end | center | stretch;  
}
```

align-items

- Aligns items along the block (column) axis
- Value applies to all grid items inside the container

```
.container {  
  align-items: start | end | center | stretch;  
}
```

place-items

- Shorthand for align-items and justify-items

```
.container {  
  place-items: center start;  
}  
/* same as */  
.container {  
  align-items: center;  
  justify-items: start;  
}
```

- If second value is omitted, first value is assigned to both properties

justify-content

- Sometimes total size of grid might be less than size of container
- Sets alignment of grid within the grid container
- Aligns the grid along the inline (row) axis

```
.container {  
  justify-content: start  
    | end  
    | center  
    | stretch  
    | space-around  
    | space-between  
    | space-evenly;  
}
```

align-content

- Sometimes total size of grid might be less than size of container
- Sets alignment of grid within the grid container
- Aligns the grid along the block (column) axis

```
.container {  
  align-content: start  
    | end  
    | center  
    | stretch  
    | space-around  
    | space-between  
    | space-evenly;  
}
```

place-content

- Shorthand for `and`

```
.container {  
  place-content: space-between center;  
}  
/* same as */  
.container {  
  align-content: space-between;  
  justify-content: start;  
}
```

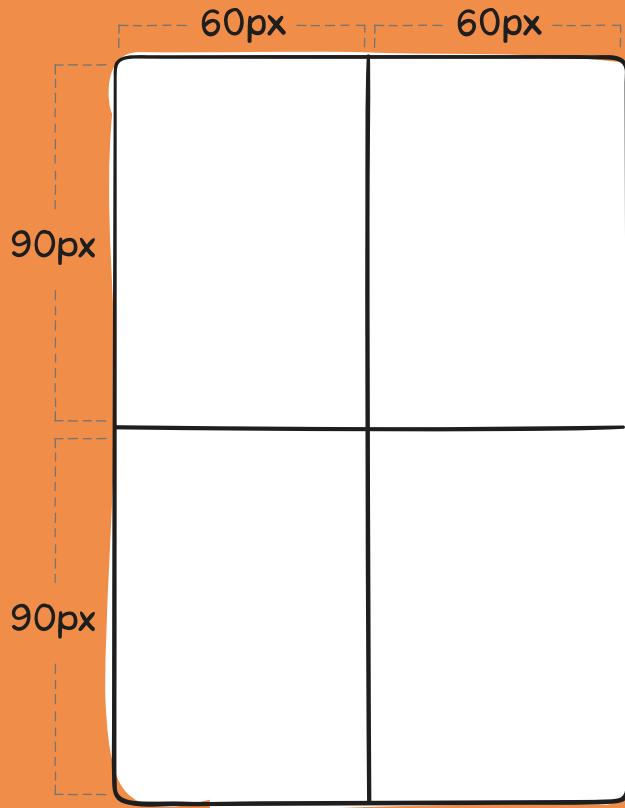
grid-auto-columns & grid-auto-rows

- Specifies the size of any auto-generated grid tracks (implicit grid tracks)
- Implicit tracks get created when there are more grid items than cells in the grid (or item is placed outside the explicit grid)

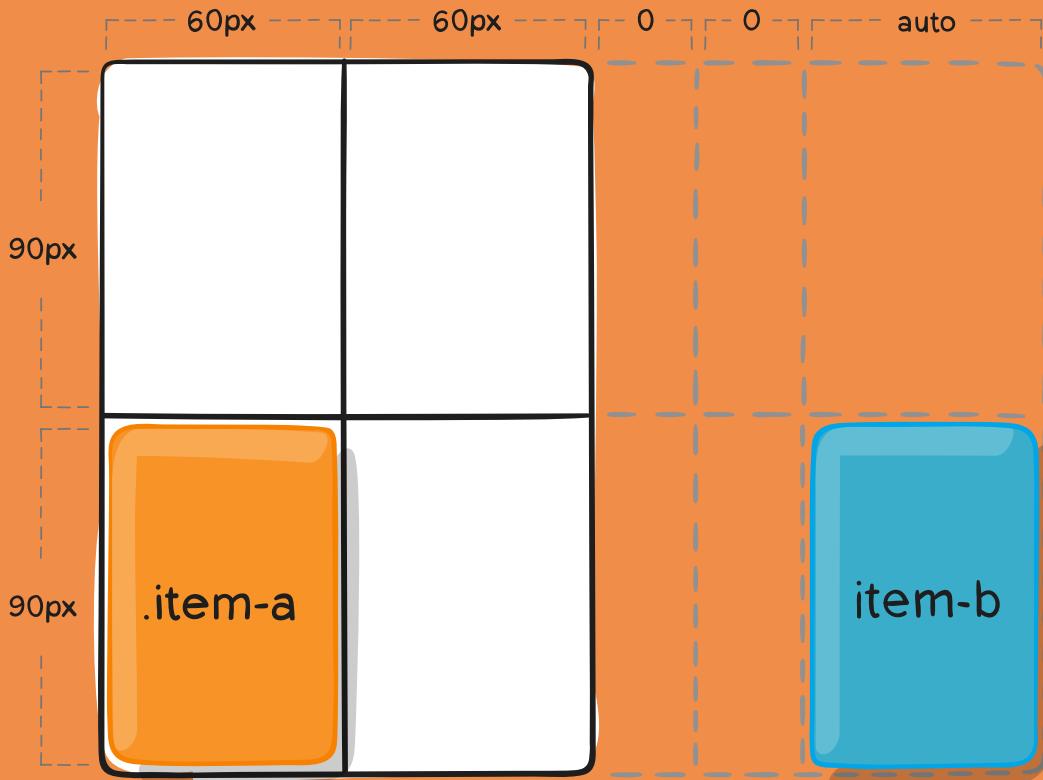
```
.container {  
  grid-auto-columns: <track-size> ...;  
  grid-auto-rows: <track-size> ...;  
}
```

Illustration of the implicit grid track

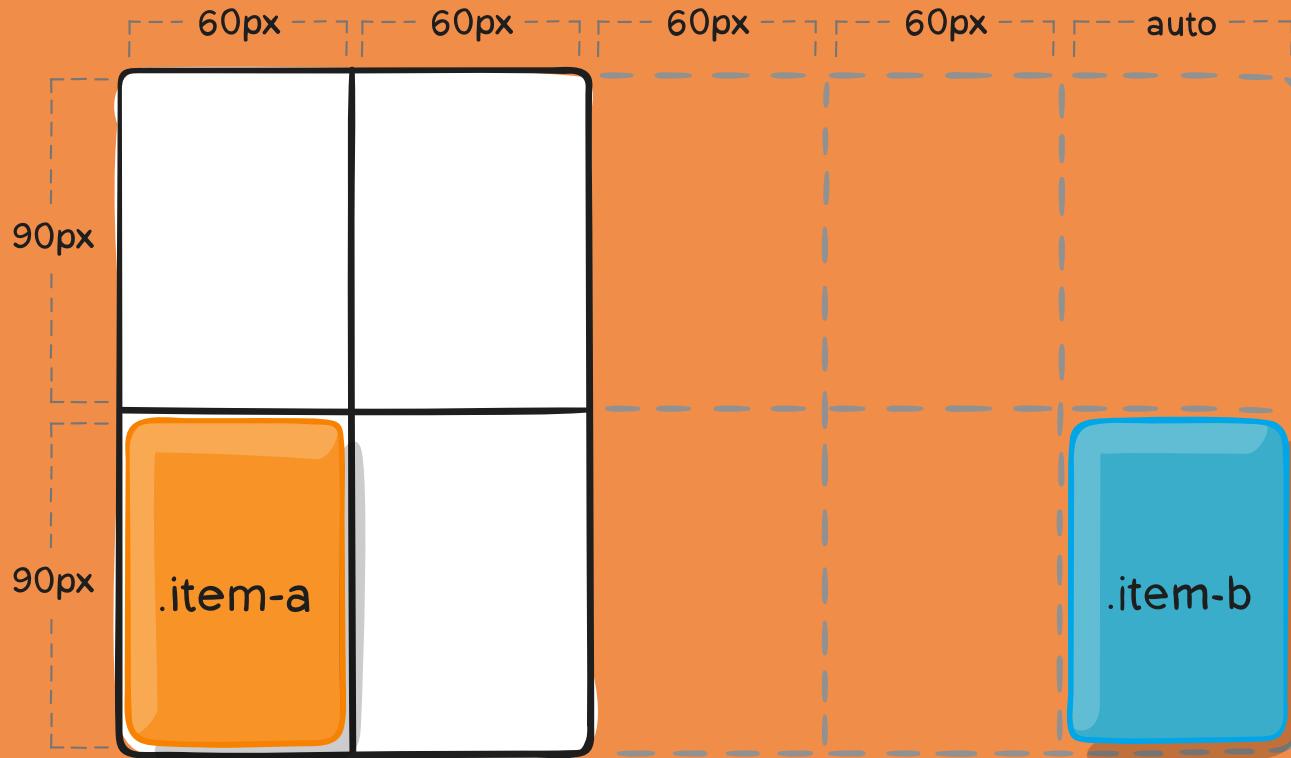
```
.container {  
  grid-template-columns: 60px 60px;  
  grid-template-rows: 90px 90px;  
}
```




```
.item-a {  
  grid-column: 1 / 2;  
  grid-row: 2 / 3;  
}  
.item-b {  
  grid-column: 5 / 6;  
  grid-row: 2 / 3;  
}
```




```
.container {  
    grid-auto-columns: 60px;  
}
```



grid-auto-flow

- If you have grid items that you don't explicitly place on the grid, the auto-placement algorithm kicks in
- **grid-auto-flow** controls how this algorithm works

```
.container {  
  grid-auto-flow: row | column | row dense | column dense;  
}
```

Example

```
<section class="container">
  <div class="item-a">item-a</div>
  <div class="item-b">item-b</div>
  <div class="item-c">item-c</div>
  <div class="item-d">item-d</div>
  <div class="item-e">item-e</div>
</section>
```

```
.container {
  display: grid;
  grid-template-columns: 60px 60px 60px 60px 60px;
  grid-template-rows: 30px 30px;
  grid-auto-flow: row; /* also default */
}

.item-a {
  grid-column: 1;
  grid-row: 1 / 3;
}

.item-e {
  grid-column: 5;
  grid-row: 1 / 3;
}
```

.item-a

.item-b

.item-c

.item-d

.item-e

```
.container {  
  display: grid;  
  grid-template-columns: 60px 60px 60px 60px 60px;  
  grid-template-rows: 30px 30px;  
  grid-auto-flow: column;  
}
```

.item-a

.item-b

.item-d

.item-c

.item-e

grid

- grid property is "shorthand" for setting the following:
 - `grid-template-rows`
 - `grid-template-columns`
 - `grid-template-areas`
 - `grid-auto-rows`
 - `grid-auto-columns`
 - `grid-auto-flow`

Example:

```
.container {  
    grid: 100px 300px / 3fr 1fr;  
}  
  
.container {  
    grid-template-rows: 100px 300px;  
    grid-template-columns: 3fr 1fr;  
}
```

```
.container {  
    grid: [row1-start] "header header header" 1fr [row1-end]  
          [row2-start] "footer footer footer" 25px [row2-end]  
          / auto 50px auto;  
}  
  
.container {  
    grid-template-areas:  
        "header header header"  
        "footer footer footer";  
    grid-template-rows: [row1-start] 1fr [row1-end row2-start]  
    grid-template-columns: auto 50px auto;  
}
```

GRID ITEMS (CHILDREN)

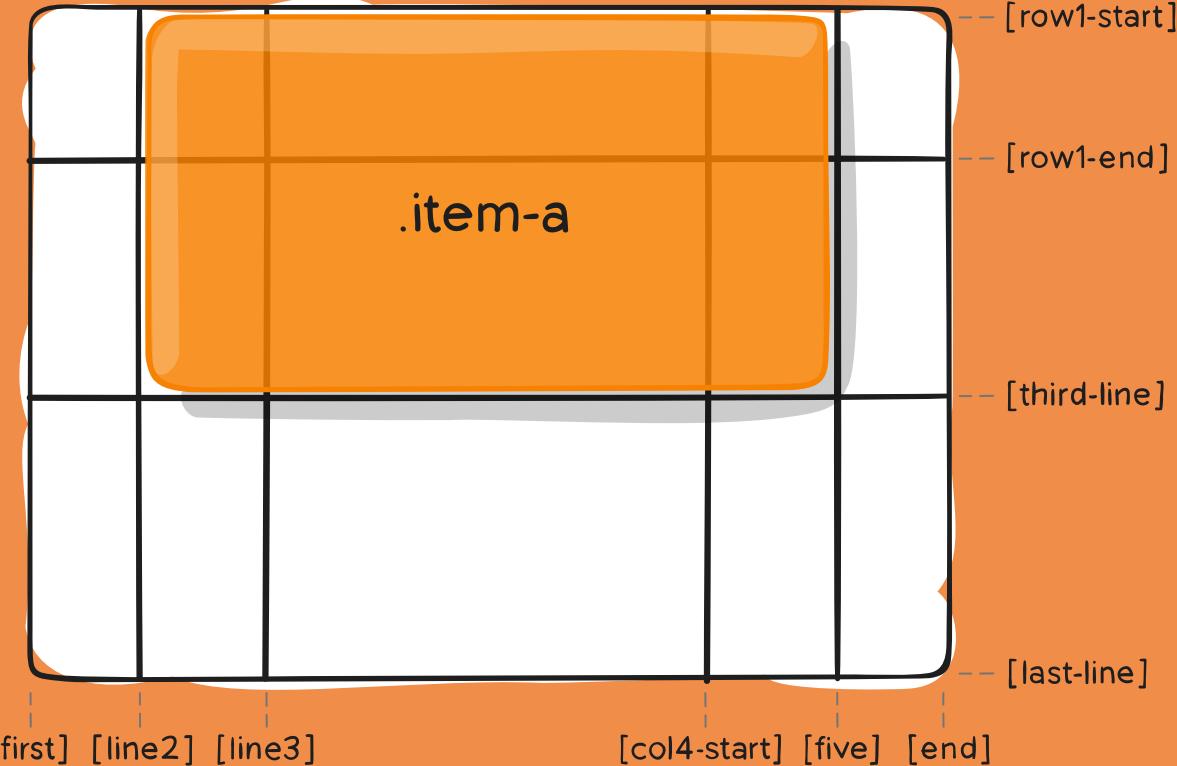
- Determine items location within the grid referring to specific grid lines

- `grid-column-start`
- `grid-column-end`
- `grid-row-start`
- `grid-row-end`

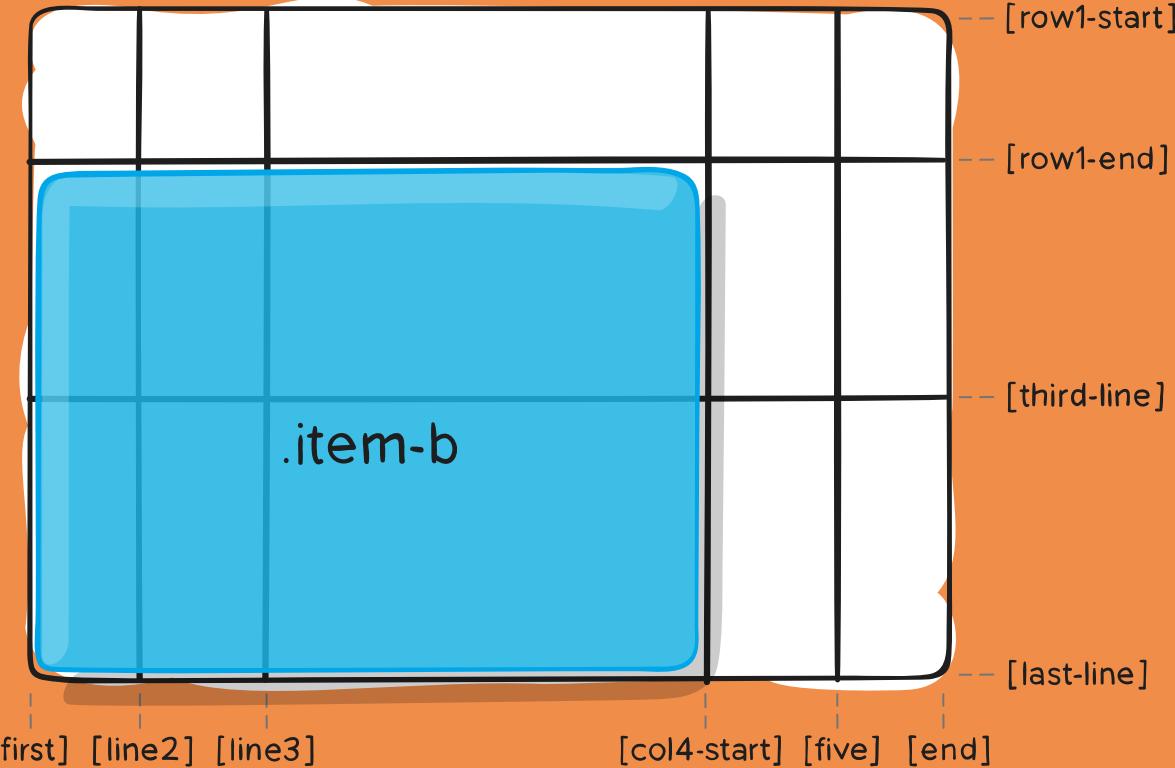
```
.item {  
  grid-column-start: <number> | <name> | span <number> | span <name>;  
  grid-column-end: <number> | <name> | span <number> | span <name>;  
  grid-row-start: <number> | <name> | span <number> | span <name>;  
  grid-row-end: <number> | <name> | span <number> | span <name>;  
}
```

Example:

```
.item-a {  
  grid-column-start: 2;  
  grid-column-end: five;  
  grid-row-start: row1-start;  
  grid-row-end: 3;  
}
```




```
.item-b {  
    grid-column-start: 1;  
    grid-column-end: span col4-start;  
    grid-row-start: 2;  
    grid-row-end: span 2;  
}
```



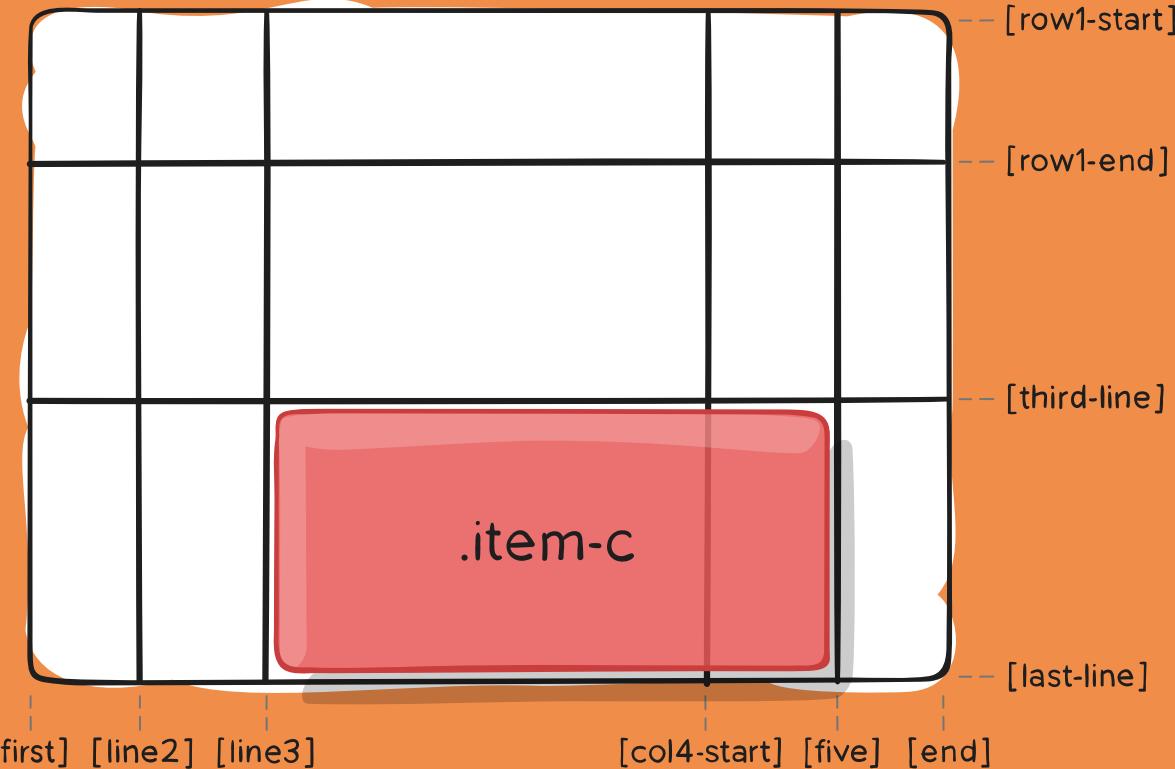
grid-column & grid-row

Shorthand for `grid-column-start + grid-column-end` & `grid-row-start + grid-row-end`

```
.item {  
  grid-column: <start-line> / <end-line> | <start-line> / span <  
  grid-row: <start-line> / <end-line> | <start-line> / span <  
}
```

Example:

```
.item-c {  
  grid-column: 3 / span 2;  
  grid-row: third-line / 4;  
}
```



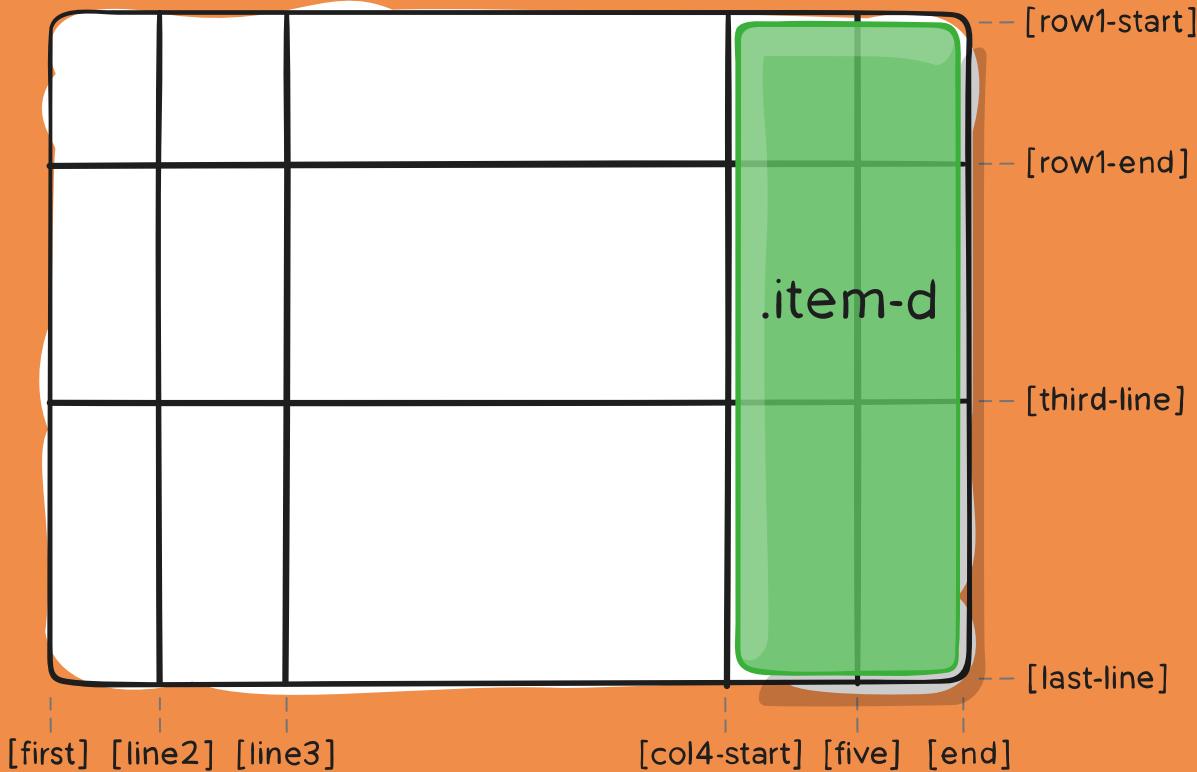
grid-area

- Gives an item a name so that it can be referenced by `grid-template-areas`
- Can also be used as shorthand for `grid-row-start + grid-column-start + grid-row-end + grid-column-end`

```
.item {  
  grid-area: <name> | <row-start> / <column-start> / <row-end>  
}
```

```
.item-d {  
  grid-area: header;  
}
```

```
.item-d {  
  grid-area: 1 / col4-start / last-line / 6;  
}
```



justify-self & align-self

- Aligns a grid item inside a cell along the inline (row) axis / the block (column) axis
- Applies to a grid item inside a single cell.

```
.item {  
  justify-self: start | end | center | stretch;  
  align-self: start | end | center | stretch;  
}
```

place-self

Shorthand for the align-self and justify-self

```
.item-a {  
  place-self: end center;  
}
```

```
.item-a {  
  align-self: end;  
  justify-self: center;  
}
```

SPECIAL UNITS & FUNCTIONS

- **fr** units or fractional units.
- "the portion of the remaining space"

```
/* example */
.container {
  grid-template-columns: 1fr 3fr; /* loosely 25% 75% */
}
```

- Sizing keywords
 - min-content
 - max-content
 - auto
 - fit-content

- Sizing functions

- `minmax()`
- `min()`
- `max()`

```
.container {  
  grid-template-columns: minmax(100px, 1fr) 3fr;  
}
```

- `repeat()` function and keywords

```
.container {  
  grid-template-columns: 1fr 1fr 1fr 1fr 1fr 1fr 1fr 1fr;  
  
  /* easier: */  
  grid-template-columns: repeat(8, 1fr);  
  
  /* especially when: */  
  grid-template-columns: repeat(8, minmax(10px, 1fr));  
}
```

- `repeat()` function can get extra fancy when combined with keywords
 - `auto-fill`: Fit as many possible columns as possible on a row, even if they are empty
 - `auto-fit`: Fit whatever columns there are into the space. Prefer expanding columns to fill space rather than empty columns.

```
.container {  
  grid-template-columns:  
    repeat(auto-fit, minmax(250px, 1fr));  
}
```

MEDIA QUERIES

Media queries are used to apply different styles for different media types/devices

- A media query consist out of a media type and media feature
- Different queries can be combined into a more complex query

MEDIA TYPES

- **all**
 - All different media types
- **screen**
 - Only when displayed on a screen
- **print**
 - Only when in print mode
- **speech**
 - Only when displayed via screen reader mode

MEDIA FEATURES

- **height [min-/max-/device-]**
 - describes the height of the output device's rendering surface
- **width [min-/max-/device-]**
 - describes the width of the rendering surface of the output device
- **orientation**
 - Indicates whether the viewport is in *landscape* or *portrait* mode
- **aspect-ratio [min-/max-/device-]**
 - Describes the aspect ratio of the targeted display area of the output device

- **color [min-/max-]**
 - Indicates the number of bits per color component of the output device
- **grid**
 - Determines whether the output device is a grid device or a bitmap device
- **monochrome [min-/max-]**
 - Indicates the number of bits per pixel on a monochrome (greyscale) device
- **resolution [min-/max-]**
 - Indicates the resolution (pixel density) of the output device

@media

When we want to use media query in CSS we use the
@media rule

```
@media screen and (min-width: 30em) and (orientation: landscape) {  
    .container {  
        background: #f00;  
    }  
}
```

```
@media (min-height: 680px), screen and (orientation: portrait
```

```
@media not screen and (color), print and (color) { /* */ }
```

```
@media (min-width: 30em) and (max-width: 50em) { /* */ }
```

```
@supports (display: flex) {
  @media screen and (min-width: 900px) {
    article {
      display: flex;
    }
  }
}
```

MEDIA ATTRIBUTE

You can also use media queries in HTML when linking
to our CSS files

```
<link href="stylesheet.css" rel="stylesheet"/>
<link href="print.css" media="print" rel="stylesheet"/>
<link href="stylesheet-large.css" media="(min-width: 40em)" r
```

RESPONSIVE VS ADAPTIVE

In adaptive design, we use breakpoints to create new
fixed layouts

```
.example {  
    margin: 0 auto;  
    width: 1400px;  
}  
  
@media (max-width: 1000px) {  
    .example {  
        width: 960px;  
    }  
}  
  
@media (max-width: 800px) {  
    .example {  
        width: 760px;  
    }  
}
```

In responsive design, we use breakpoints to create new **fluid** layouts

```
.example {  
  width: 80%;  
  margin: 0 auto;  
}  
  
@media (max-width: 1000px) {  
  .example {  
    width: 90%;  
  }  
}  
  
@media (max-width: 800px) {  
  .example {  
    width: 95%;  
  }  
}
```

CSS ANIMATIONS & TRANSITIONS

TRANSITIONS

A transition is there to alter the appearance of an element when a state change happens (eg. hover, focus, ...).

No transition

With transition

transition-property

- When transition is set, by default all properties are affected
- You can define specific properties, which will be affected by transition

```
.foo {  
  background: hotpink;  
  border-radius: 2%;  
  transition-property: background, border-radius;  
  width: 40px;  
}  
  
.foo:hover {  
  background: cyan;  
  border-radius: 50%;  
  width: 80px;  
}
```

transition-duration

- You can set duration of the transition
 - in seconds (**s**) or milliseconds (**ms**). May also be fractional (**.3s**)
- Default is **0s**, so if not set, no transition

```
.foo {  
  background: hotpink;  
  transition-property: background;  
  transition-duration: 1s;  
}  
  
.foo:hover {  
  background: cyan;  
}
```

transition-delay

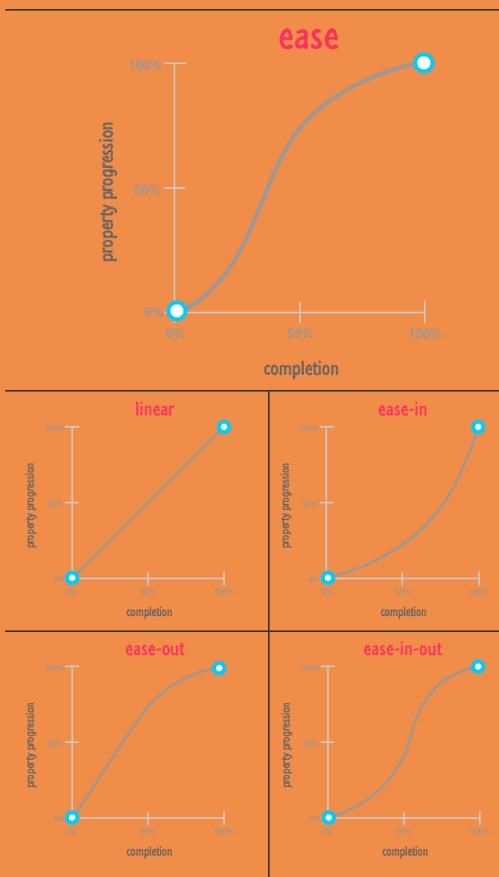
- Stall the transition before playing

```
.foo {  
  background: hotpink;  
  transition-property: background;  
  transition-duration: 1s;  
  transition-delay: 1s;  
}  
  
.foo:hover {  
  background: cyan;  
}
```

transition-timing-function

- sets the speed in which a transition is happening
- Keywords:
 - **linear**
 - **ease**
 - **ease-in**
 - **ease-out**
 - **ease-in-out**
 - **steps**

```
.foo {  
  transition-timing-function: ease-in-out;  
}
```



You also use custom **cubic-bezier**-function

```
.quux {  
  transition-timing-function: cubic-bezier(0.0, 0.0, 1.0, 1  
}
```

<https://cubic-bezier.com/>

SHORTHAND

```
.foo {  
  transition: .4s background-color ease-out;  
  
  /* same as */  
  transition-delay: 0s;  
  transition-duration: .4s;  
  transition-property: background-color;  
  transition-timing-function: ease-out;  
}
```

ANIMATIONS

Animations are for changing appearances over time by using keyframes. This way you can chain multiple transitions inside those different keyframes.

KEYFRAMES

- Build up of the animation using the at-rule
@keyframes
- Use of breakpoints

```
@keyframes foo { /* with keyword */
  from {
    color: hotpink;
  }
  to {
    color: cyan;
  }
}

@keyframes bar { /* only percentages */
  0% {
    color: hotpink;
  }
  50% {
    color: rebeccapurple;
  }
  100% {
    color: cyan;
  }
}
```

animation-name

Link created **@keyframes** to element

```
.container {  
  animation-name: foo;  
}
```

animation-duration

Sets the duration of a single animation

```
.container {  
  animation-duration: 4s;  
}
```

animation-timing-function

Sets the speed curve of a single animation

```
.container {  
  animation-timing-function: linear;  
  /* linear | ease | ease-in | ease-out | ease-in-out |  
   cubic-bezier(n,n,n,n) */  
}
```

animation-delay

Sets the time between loading the element and start of animation

```
.container {  
    animation-delay: 1s;  
}
```

animation-iteration-count

By default, an animation runs only one. With **animation-iteration-count**, you can define the number of cycles

You can set a number or the **infinite** keyword

```
.container {  
    animation-iteration-count: 20;  
    animation-iteration-count: infinite;  
}
```

ANIMATION-DIRECTION

Sets the direction of the animation

- Possible values:

- normal
- reverse
- alternate
- alternate-reverse

```
.container {  
  animation-direction: alternate;  
}
```

animation-play-state

You can set an animation in **running** state or a **paused** state, when in an state of an element (fe. hover)

```
.container:hover {  
    animation-play-state: paused;  
}
```

animation-fill-mode

This property defines if the styling of the element should be done before, after or before and after the animation.

- **forwards**
 - The last styles applied at the end of the animation are retained afterwards
- **backwards**
 - The animation's styles will already be applied before the animation actually starts
- **both**
 - The styles are applied before and after the animation plays

```
.container {  
  animation-fill-mode: both;  
}
```

TRANSFORM

Transform CSS property let you rotate, scale, skew or translate an element. It modifies the coordinate space of the CSS visual formatting model

rotate()

Rotate function defines transformation that rotates an element around a fixed point on the 2D plane, without deforming it.

```
.foo {  
  transform: rotate(30deg);  
  transform: rotate(.5turn);  
}
```

scale() - RESIZING

Resizes an element on the 2D plane. Because the amount of scaling is defined by a vector, it can resize the horizontal and vertical dimensions at different scales.

```
.foo {  
  transform: scale(0.7);  
  transform: scale(1.4, 0.5); /* scaleX, scaleY */  
}
```

skew() - DISORTION

skews an element on the 2D plane

```
.foo {  
  transform: skew(10deg); /* skewX(10deg) */  
  transform: skew(10deg, 10deg); /* skewX, skewY */  
}
```

translate() - MOVING

repositions an element in the horizontal and/or
vertical directions

```
.foo {  
  transform: translate(60px); /* translateX(60px) */  
  transform: translate(100px, 200%); /* translateX, translateY */  
}
```

COMBINED

transform functions can be combined

```
.foo {  
  transform: rotate(45deg) translateX(180px);  
}
```

MAINTAINABLE CSS

PROJECT STRUCTURE

```
body {  
    font-family: Arial;  
}  
  
h1 {  
    font-size: 30px;  
}  
  
h2 {  
    font-size: 24px;  
}
```

```
body {
    font-family: Arial;
}

h1 {
    font-size: 30px;
}

h2 {
    font-size: 24px;
}

.btn {
    background-color: hotpink;
    border-radius: 3px;
    color: white;
    padding: 10px;
}

.navbar {
    position: fixed;
    top: 0;
    left: 0;
    right: 0.
```

```
body {
    font-family: Arial;
}

h1 {
    font-size: 30px;
}

h2 {
    font-size: 24px;
}

.btn {
    background-color: hotpink;
    border-radius: 3px;
    color: white;
    padding: 10px;
}

.navbar {
    position: fixed;
    top: 0;
    left: 0;
    right: 0.
```

```
- src
  - components
    - button
      - button.html
      - button.css
    - navbar
      - navbar.html
      - navbar.css
  - app
    - featureA
      - featureA.html
      - featureA.css
      - featureA.js
  - common
    - common.css
```

```
body {  
    font-family: Arial;  
}  
  
h1 {  
    font-size: 30px;  
}  
  
h2 {  
    font-size: 24px;  
}
```

button.css

```
.btn {  
    background-color: hotpink;  
    border-radius: 3px;  
    color: white;  
    padding: 10px;  
}  
  
.btn-primary {  
    background-color: rebeccapurple;  
}
```

common.css

```
.navbar {  
    position: fixed;  
    top: 0;  
    left: 0;  
    right: 0;  
    height: 50px;  
    background-color: #f00;  
}
```

navbar.css

PREPROCESSORS

Tools that make CSS development easier

AVAILABLE PREPROCESSORS

- Sass
- Less
- Stylus
- Compass
- ...

COMPILING

Now that we are using a preprocessor we'll need to compile our source files into css. There are several tools to do this (such as grunt, gulp, webpack, ...) and most modern IDE's have a compiler built in.

- gulp-sass
- sass-loader
- grunt-sass

QUICK OVERVIEW OF SASS

- Syntactically Awesome Style Sheets
- Originally ruby based, but now also js compiler
- 2 synthax: sass & scss
 - sass (old syntax) -> indent based
 - scss -> css based syntax (feels more natural like css)

VARIABLES

```
$my-favorite-color: #ff69b4;  
  
.example {  
    color: $my-favorite-color;  
}
```

```
.example {  
    color: #ff69b4;  
}
```

LOOPING

```
@for $i from 1 through 3 {  
  .item-#${$i} {  
    width: 100px * $i;  
  }  
}
```

```
.item-1 {  
  width: 100px;  
}  
  
.item-2 {  
  width: 200px;  
}  
  
.item-3 {  
  width: 300px;  
}
```

NESTING

```
.long-class-name {  
  ul {  
    margin: 0;  
    padding: 0;  
    list-style: none;  
  }  
  li { display: inline-block; }  
  &.link {  
    display: block;  
    padding: 6px 12px;  
    text-decoration: none;  
  }  
}
```

```
.long-class-name ul {  
    margin: 0;  
    padding: 0;  
    list-style: none;  
}  
.long-class-name li {  
    display: inline-block;  
}  
.long-class-name.link {  
    display: block;  
    padding: 6px 12px;  
    text-decoration: none;  
}
```

MIXINS

```
@mixin theme($theme: darkgrey) {  
  background: $theme;  
  box-shadow: 0 0 1px rgba($theme, .25);  
  color: #fff;  
}  
.info {  
  @include theme;  
}  
.alert {  
  @include theme($theme: darkred);  
}  
.success {  
  @include theme($theme: darkgreen);  
}
```

```
.info {  
    background: darkgrey;  
    box-shadow: 0 0 1px rgba(169, 169, 169, 0.25);  
    color: #fff;  
}  
  
.alert {  
    background: darkred;  
    box-shadow: 0 0 1px rgba(139, 0, 0, 0.25);  
    color: #fff;  
}  
  
.success {  
    background: darkgreen;  
    box-shadow: 0 0 1px rgba(0, 100, 0, 0.25);  
    color: #fff;
```

EXTEND / INHERITANCE

```
/* This CSS will print because %message-shared is extended. */
%message-shared {
  border: 1px solid #ccc;
  padding: 10px;
  color: #333;
}

// This CSS won't print because %equal-heights is never extended.
%equal-heights {
  display: flex;
  flex-wrap: wrap;
}

.message {
  @extend %message-shared;
}

.success {
  @extend %message-shared;
  border-color: green;
}
```

```
/* This CSS will print because %message-shared is extended. */
.message, .success, .error, .warning {
    border: 1px solid #ccc;
    padding: 10px;
    color: #333;
}

.success {
    border-color: green;
}

.error {
    border-color: red;
}

.warning {
    border-color: yellow;
}
```

OPERATORS

```
.container {  
    width: 100%;  
}  
  
article[role="main"] {  
    float: left;  
    width: 600px / 960px * 100%;  
}  
  
aside[role="complementary"] {  
    float: right;  
    width: 300px / 960px * 100%;  
}
```

```
.container {  
    width: 100%;  
}  
  
article[role="main"] {  
    float: left;  
    width: 62.5%;  
}  
  
aside[role="complementary"] {  
    float: right;  
    width: 31.25%;  
}
```

A FEW WORDS

A FEW WORDS

- More functionality than written here

A FEW WORDS

- More functionality than written here
 - But it's a quick overview

A FEW WORDS

- More functionality than written here
 - But it's a quick overview
- Be cautious when using extends / inheritance

A FEW WORDS

- More functionality than written here
 - But it's a quick overview
- Be cautious when using extends / inheritance
 - Compiled css can give long selectors

A FEW WORDS

- More functionality than written here
 - But it's a quick overview
- Be cautious when using extends / inheritance
 - Compiled css can give long selectors
 - Better solutions available

A FEW WORDS

- More functionality than written here
 - But it's a quick overview
- Be cautious when using extends / inheritance
 - Compiled css can give long selectors
 - Better solutions available
- Keep it readable

A FEW WORDS

- More functionality than written here
 - But it's a quick overview
- Be cautious when using extends / inheritance
 - Compiled css can give long selectors
 - Better solutions available
- Keep it readable
 - Nesting can become unreadable very quickly

A FEW WORDS

- More functionality than written here
 - But it's a quick overview
- Be cautious when using extends / inheritance
 - Compiled css can give long selectors
 - Better solutions available
- Keep it readable
 - Nesting can become unreadable very quickly
 - Minimize the use of &

BEM

- BEM is frontend methodology
 - Block
 - Element
 - Modifier
- A way of giving clearer names to CSS classes to improve clarity and readability

BLOCK

A block represents the higher level of a component

```
/* eg. .navigation */
.block {
  ...
}
```

ELEMENT

An element is a descendent of a block that performs a specific function.

```
/* eg. .navigation */
.block_element {
  ...
}
```

MODIFIER

The modifier represents a different state of a block or an element.

```
/* eg. .navigation__item--active */  
.block__element--modifier {  
  ...  
}
```

QUESTIONS? 