howest
hogeschool

Web Technology
Responsive (by) design

Dieter Mourisse

(some slides by Jill VandenDriessche & Machteld De Groef)

howest
hogeschool

What is Responsive design

What is responsive design

"shrinking stuff to make it fit on the user's screen!"

What is responsive design

"shrinking stuff to make it fit on the user's screen!"

What is responsive design

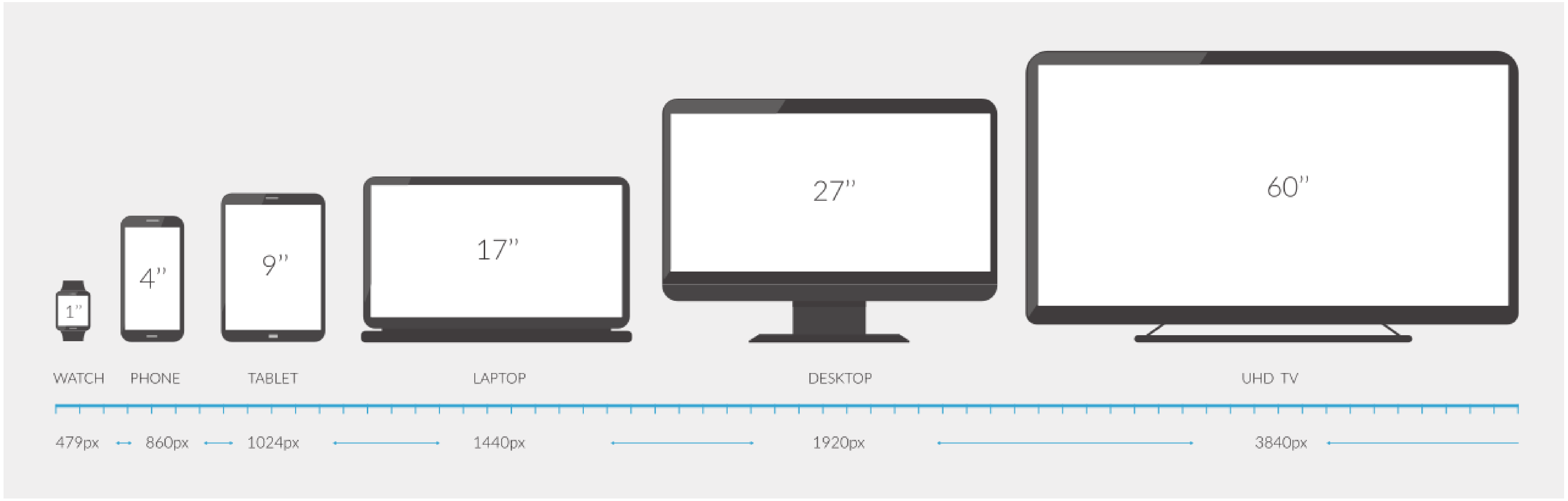
An approach that suggests that

design and development should respond to the user’s behaviour and environment.

https://www.smashingmagazine.com/2011/01/guidelines-for-responsive-web-design/

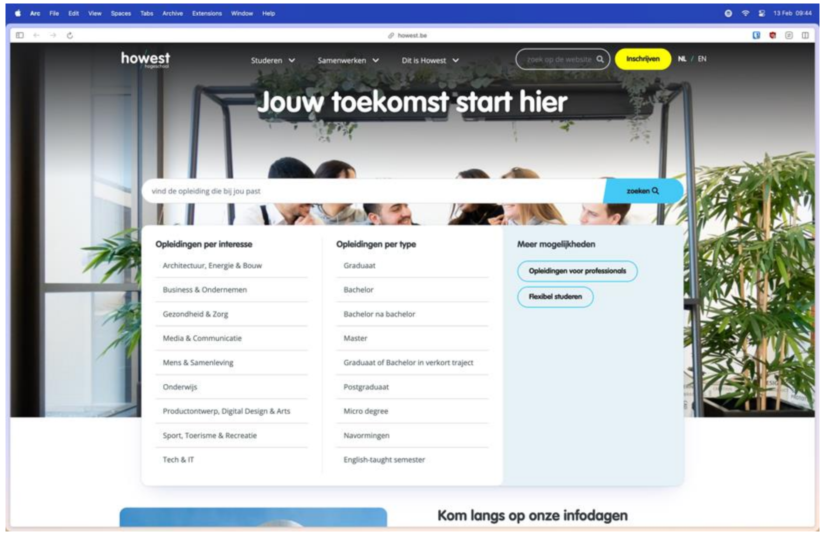
What is responsive design

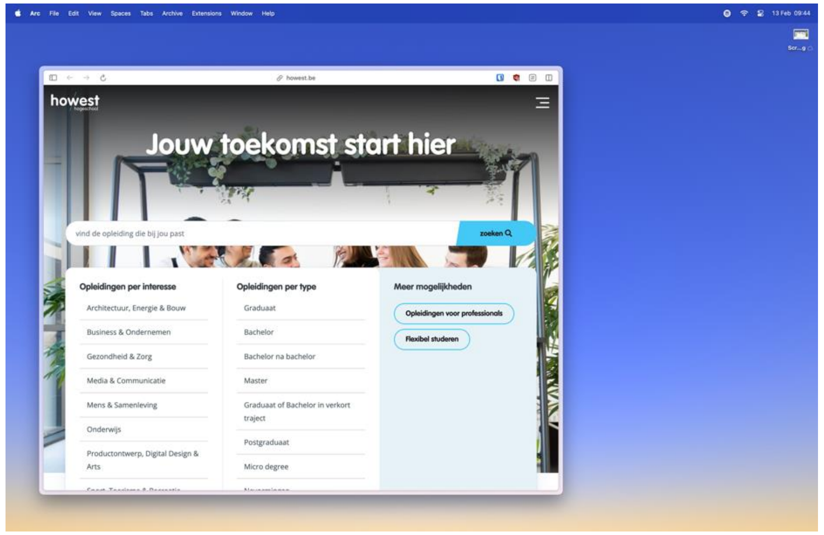
Difference in screen sizes

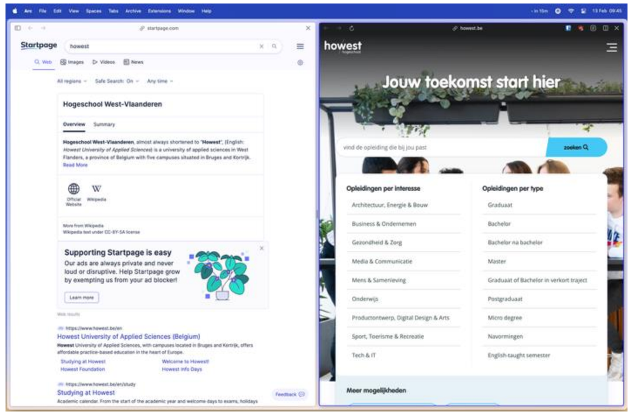


What is responsive design

Difference in window (viewport) sizes





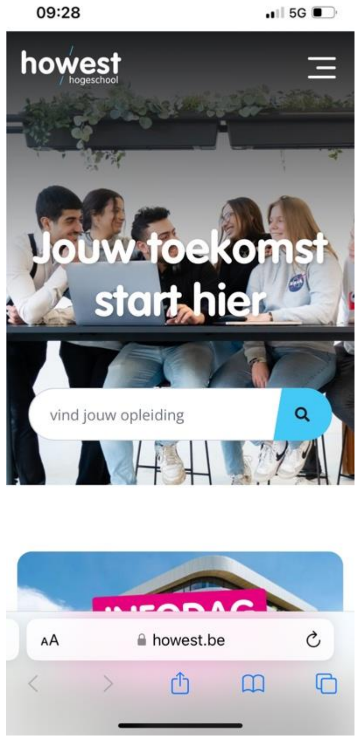

Startpage
howest
Hogeschool West-Vlaanderen
Supporting Startpage is easy
Jouw toekomst start hier
Opleidingen per interesse
Opleidingen per type
Meer mogelijkheden

Full screen

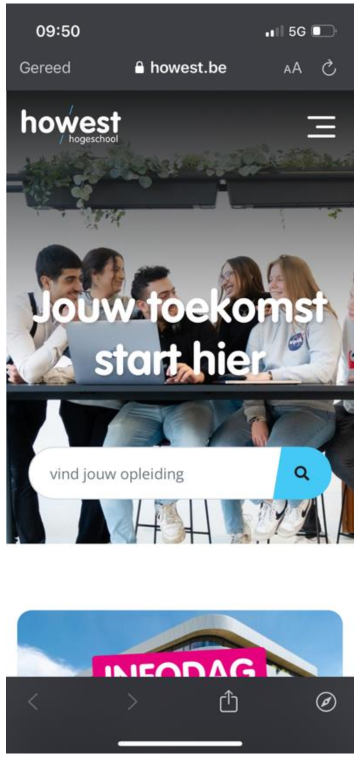
Partial / split screen

What is responsive design

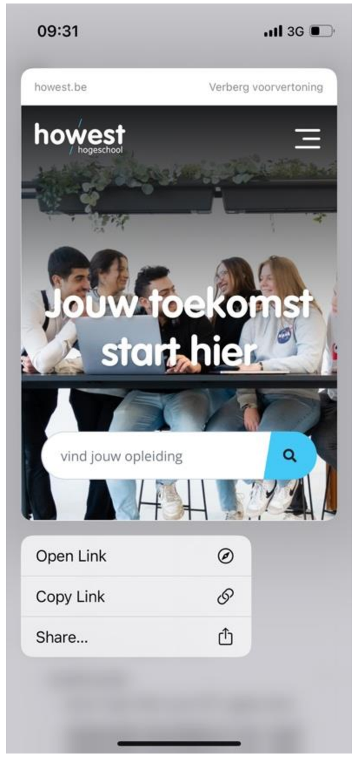
Difference in environments (e.g. iOS)



Browser



In-app browser



3D touch preview

What is responsive design

Making sure that your web application
delivers a good user experience
in all circumstances

First: let it go

"You **do not know what conditions your website will be visited on** and you have little to no control over that.

Accept that lack of control and use the limitations to breed creativity and also, laser focus your UX work."

https://www.viewports.fyi

howest
hogeschool

"Be the browser’s mentor, not its micromanager"

https://buildexcellentwebsit.es

Don't micromanage the browser

Give the browser **some solid rules & hints**

then **let it make the right decisions**

for the people that visit it, based on their device,
connection quality, capabilities and preferences.

Don't micromanage the browser

Give the browser **some solid rules & hints** →

then **let it make the right decisions**

Modern CSS
you shall not write overly specific rules

Flexible layouts
you shall not match devices

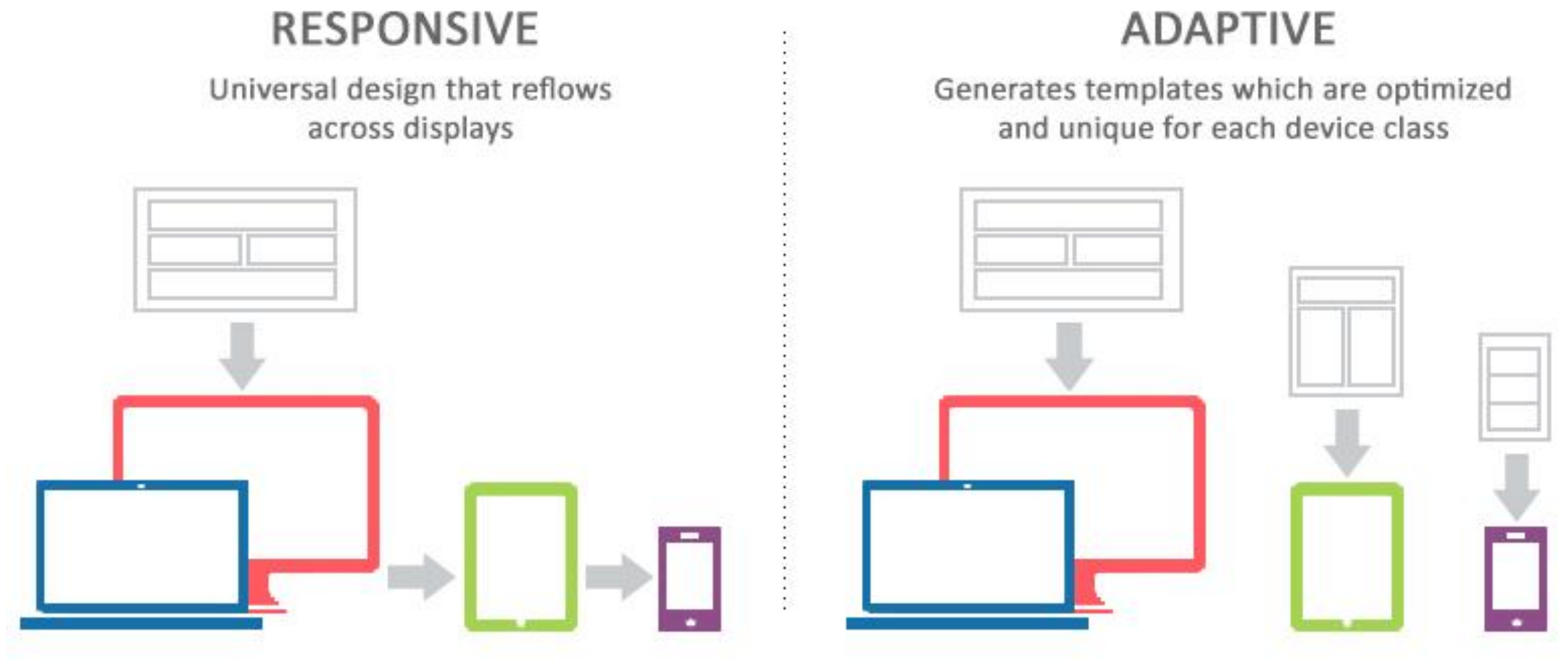
Fluid font sizes & spacing
you shall not use absolute units

Progressive enhancement
you shall not start from the best-case scenario

howest
hogeschool

Adaptive design

Responsive vs Adaptive Design


RESPONSIVE
Universal design that reflows across displays
ADAPTIVE
Generates templates which are optimized and unique for each device class

One codebase for all devices

Serve different codebase depending on the device

The advantages and disadvantages of each approach are a lesson in itself.

We'll be focussing on the responsive design techniques only, and mainly on those that are responsive by design.

So we need one code base to do all this

Block vs inline

Block vs Inline axis

Inline
Parallel to the flow of text within a line.
For standard Dutch text = horizontal.

Block
Perpendicular to the flow of text within a line.
For standard Dutch text = vertical.

https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Logical_Properties#block_vs._inline

Block vs Inline axis

Inline
Parallel to the flow of text within a line.
For standard Dutch text = horizontal.

Block
Perpendicular to the flow of text within a line.
For standard Dutch text = vertical.

Depends on the writing mode!

https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Logical_Properties#block_vs._inline
https://www.w3.org/International/questions/qa-scripts.en

Block vs inline axis

CSS was initially designed with only physical coordinates in its controls. Some physical properties now have **logical equivalents**. The transition to logical axes is **ongoing**.

Physical properties	Flow-relative logical values
Top	block-start
Bottom	block-end
Left	inline-start
Right	inline-end
Height	block-size
Width	inline-size

e.g. margin-top => margin-block-start

howest
hogeschool

The Viewport

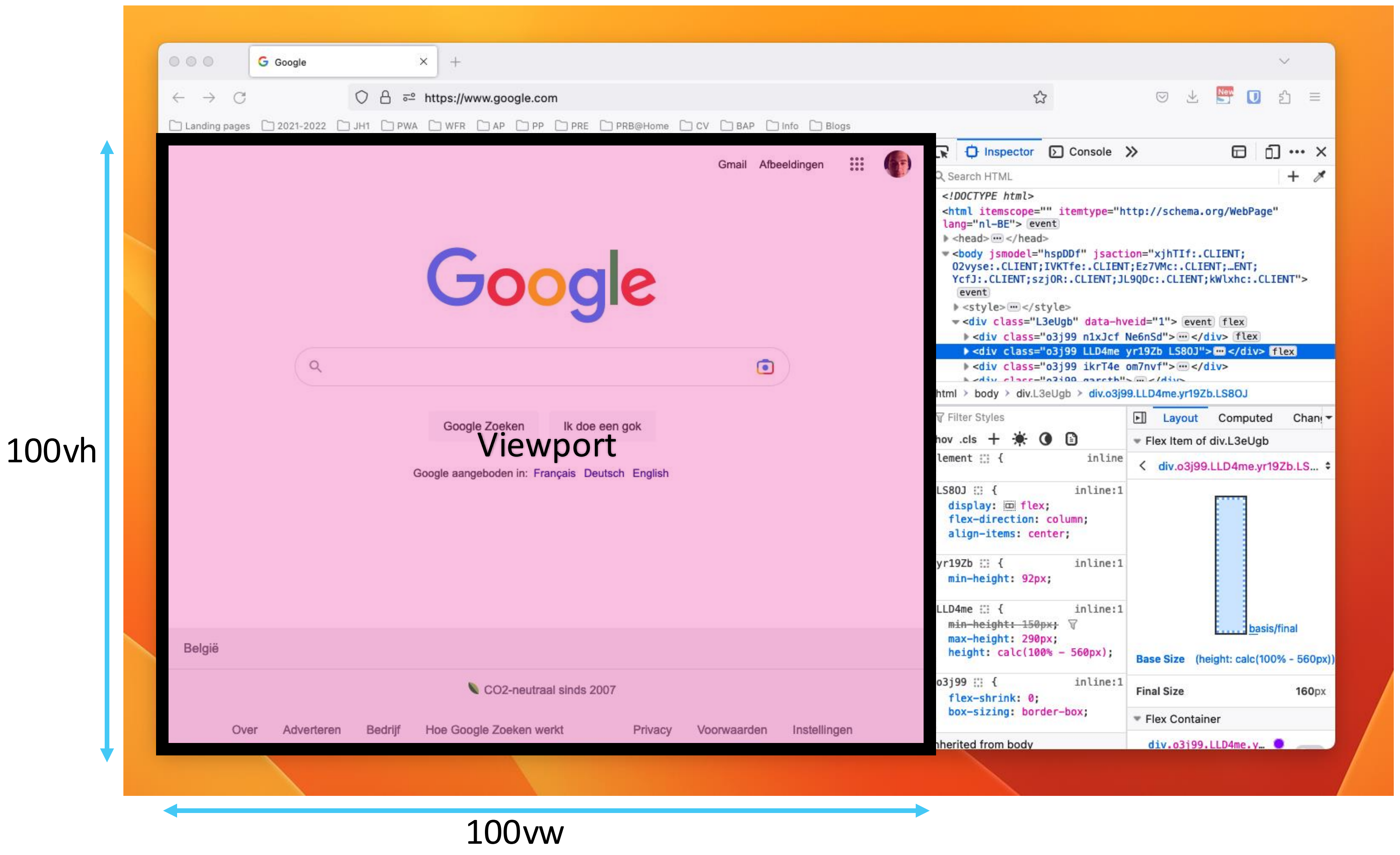
The Viewport

A viewport represents the area in computer graphics being currently viewed. In web browser terms, it is generally the same as the browser window, excluding the UI, menu bar, etc. That is the part of the document you are viewing. Your viewport is everything that is currently visible

Viewport units

Unit	Meaning
1vw	1% of the viewport's width
1vh	1% of the viewport's height
1vmin	1% of the viewport's smaller dimension
1vmax	1% of the viewport's larger dimension
1vb	1% of the initial containing block in the direction of the root element's block axis
1vi	1% of the initial containing block in the direction of the root element's inline axis

https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/Values_and_units



```
<!DOCTYPE html>
<html itemscope="" itemtype="http://schema.org/WebPage"
lang="nl-BE"> event
  <head> ··· </head>
  <body jsmodel="hspDDf" jsaction="xjhTIf:.CLIENT;
  O2vyse:.CLIENT;IVKTfe:.CLIENT;Ez7VMc:.CLIENT;…ENT;
  YcfJ:.CLIENT;szjOR:.CLIENT;JL9QDc:.CLIENT;kWlxhc:.CLIENT">
   event
    <style> ··· </style>
    <div class="L3eUgb" data-hveid="1"> event flex
      <div class="o3j99 n1xJcf Ne6nSd"> ··· </div> flex
      <div class="o3j99 LLD4me yr19Zb LS8OJ"> ··· </div> flex
      <div class="o3j99 ikrT4e om7nvf"> ··· </div>
html > body > div.L3eUgb > div.o3j99.LLD4me.yr19Zb.LS8OJ
```

```
lement {                          inline
}
LS8OJ {                          inline:1
  display: flex;
  flex-direction: column;
  align-items: center;
}
yr19Zb {                         inline:1
  min-height: 92px;
}
LLD4me {                         inline:1
  ~~min-height: 150px;~~
  max-height: 290px;
  height: calc(100% - 560px);
}
o3j99 {                          inline:1
  flex-shrink: 0;
  box-sizing: border-box;
}
nherited from body
```

```
div.o3j99.LLD4me.yr19Zb.LS...
Base Size    (height: calc(100% - 560px))
Final Size                          160px
div.o3j99.LLD4me.y…
```

On desktop browsers, the concept is ”easy” (if you ignore scrollbars exist)

howest hogeschool

On mobile browsers, it's a bit more difficult.

The small viewport is the viewport sized assuming any dynamic UA interfaces to be expanded.

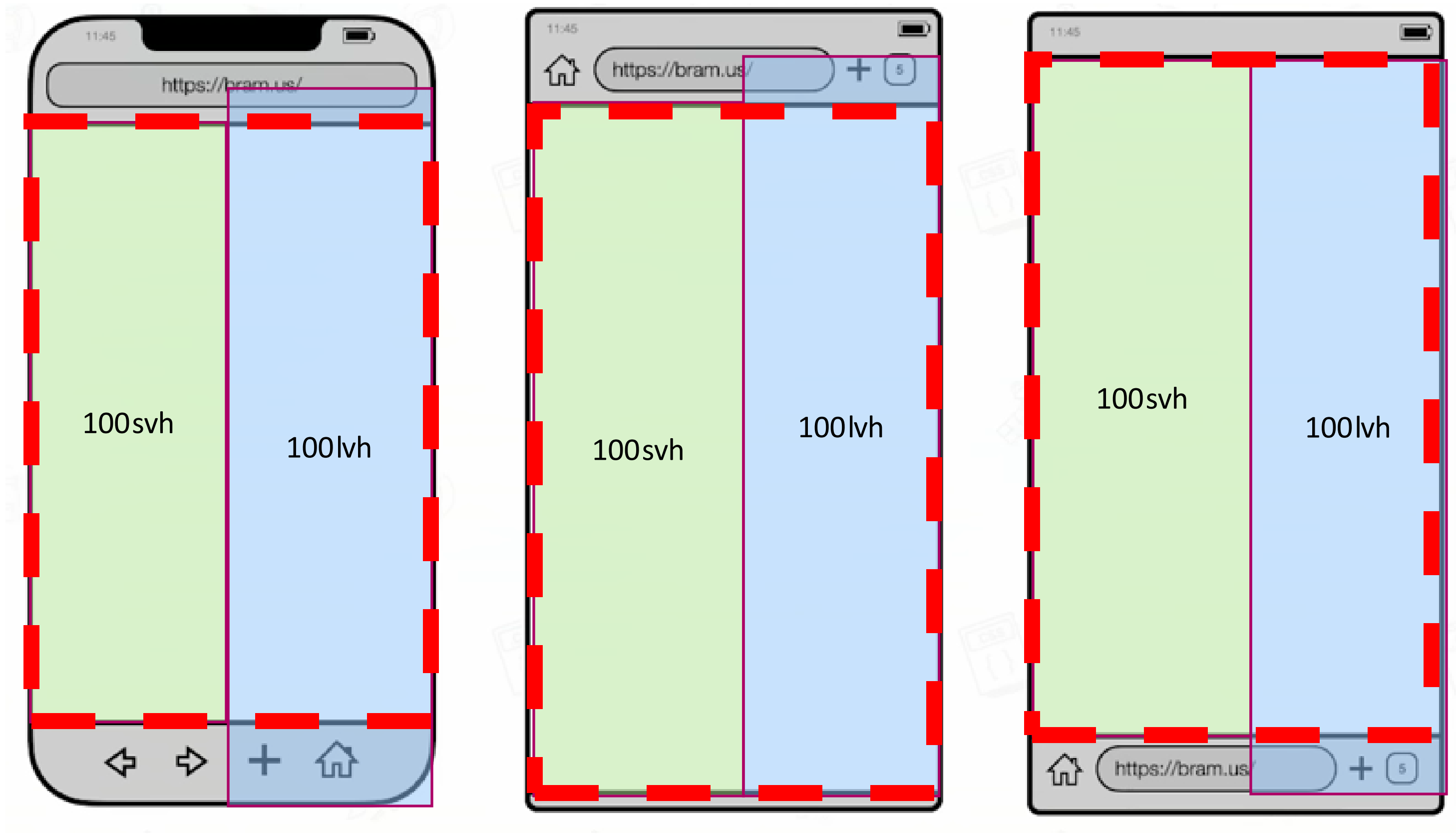
The large viewport is the viewport sized assuming any dynamic UA interfaces to be retracted.

The Dynamic Viewport is the viewport sized with dynamic consideration of any UA interfaces that are dynamically expanded and retracted.

mobile safari

mobile chrome

mobile firefox



Since mobile browsers often have interfaces that expand and retract dynamically there are four types of viewports:

- Small
- Large
- Dynamic
- Default

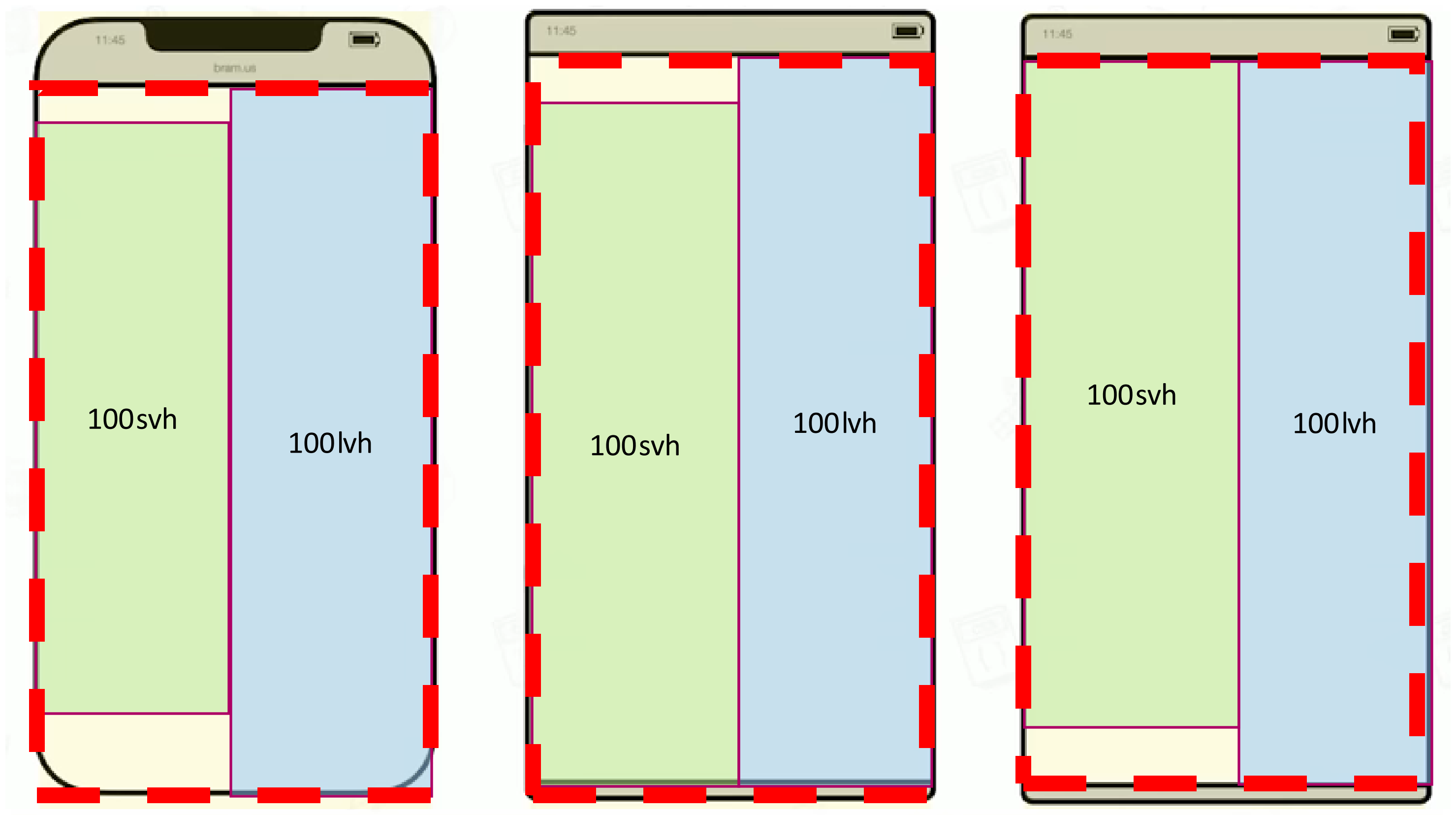
On mobile browsers, it's a bit more difficult.

https://www.youtube.com/watch?v=xl9R8aTOW_I

mobile safari

mobile chrome

mobile firefox



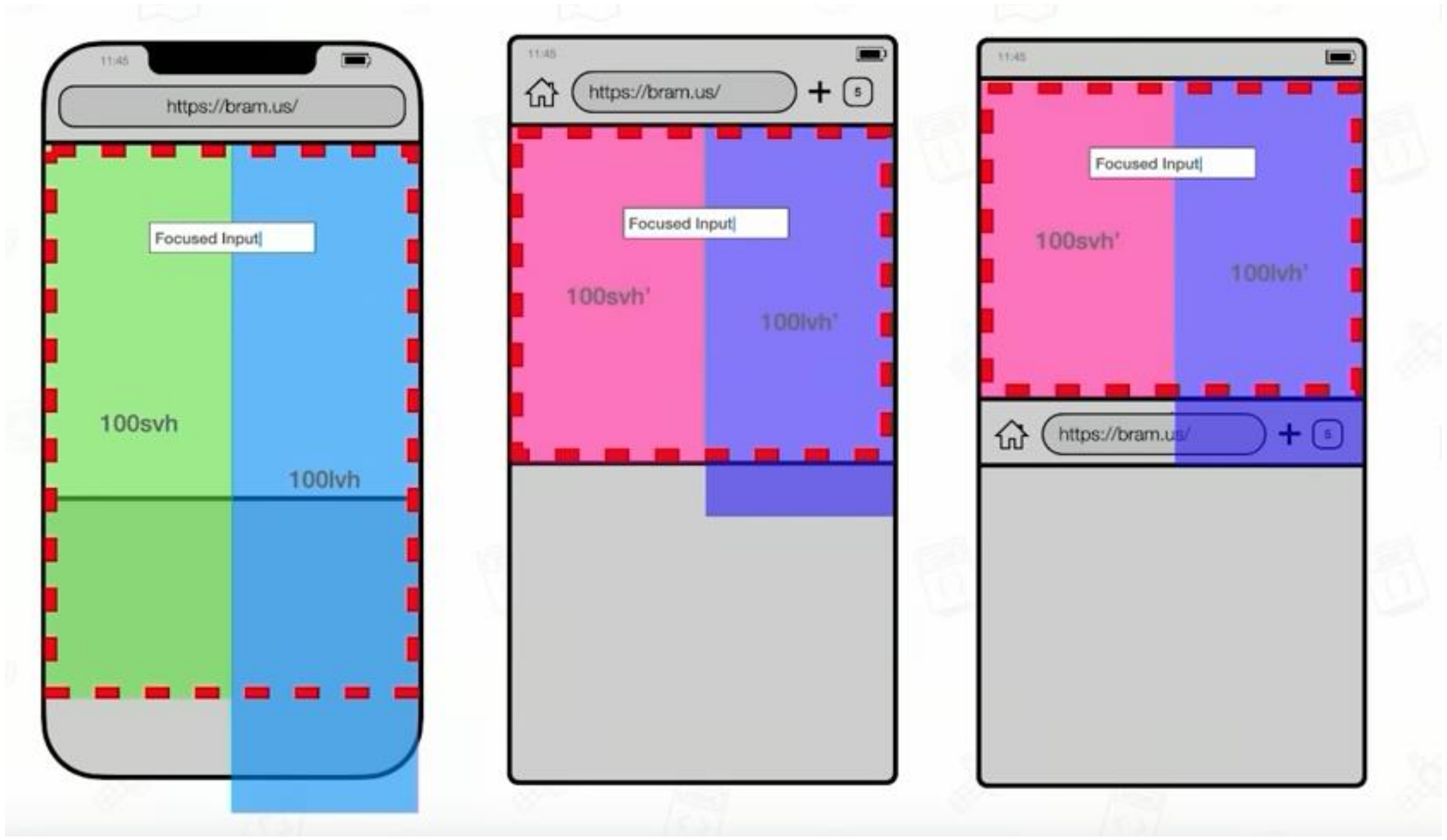
Since mobile browsers often have interfaces that expand and retract dynamically there are four types of viewports:

- Small
- Large
- Dynamic
- Default

On mobile browsers, it's a bit more difficult.

https://www.youtube.com/watch?v=xl9R8aTOW_I

mobile safari mobile chrome mobile firefox



On mobile browsers, it's a bit more difficult.

Since mobile browsers often have interfaces that expand and retract dynamically there are four types of viewports:

- Small
- Large
- Dynamic
- Default

https://www.youtube.com/watch?v=xl9R8aTOW_I

Viewport units

Viewport	Units
Small viewport	svh, svw, svmin, svmax, svi, svb
Large viewport	lvh, lvw, lvmin, lvmax, lvi, lvb
Dynamic Viewport	dvh, dvw, dvmin, dvmax, dvi, dvb
UA-default viewport	vh, vw, vmin, vmax, vi, vb

The viewport meta tag

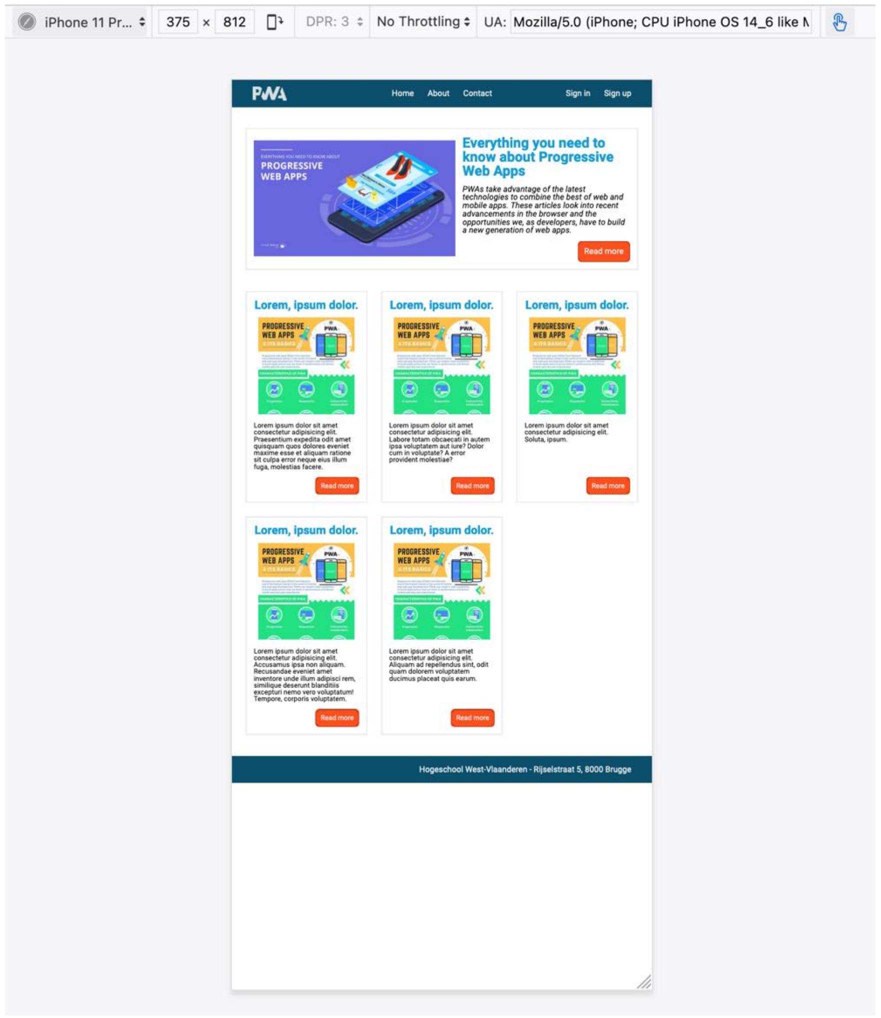
Always place a viewport meta tag inside the head element of your html pages.

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

Some mobile devices and other narrow screens render pages in a virtual window or viewport, which is usually wider than the screen, and then shrink the rendered result down so it can all be seen at once. Users can then pan and zoom to see different areas of the page.

Adding the viewport meta tag overrides this behaviour.

https://developer.mozilla.org/en-US/docs/Web/HTML/Viewport_meta_tag without viewport meta tag



iPhone 11 Pr... 375 × 812 DPR: 3 No Throttling UA: Mozilla/5.0 (iPhone; CPU iPhone OS 14_6 like M

PWA Home About Contact Sign in Sign up

Everything you need to know about Progressive Web Apps

PWAs take advantage of the latest technologies to combine the best of web and mobile apps. These articles look into recent advancements in the browser and the opportunities we, as developers, have to build a new generation of web apps.

Read more

Lorem, ipsum dolor.

Lorem ipsum dolor sit amet consectetur adipisicing elit. Praesentium expedita odit amet quisquam quos dolores eveniet maxime esse et aliquam ratione sit culpa error neque eius illum fuga, molestias facere.

Read more

Lorem, ipsum dolor.

Lorem ipsum dolor sit amet consectetur adipisicing elit. Labore totam obcaecati in autem ipsa voluptatem aut iure? Dolor cum in voluptate? A error provident molestiae?

Read more

Lorem, ipsum dolor.

Lorem ipsum dolor sit amet consectetur adipisicing elit. Soluta, ipsum.

Read more

Lorem, ipsum dolor.

Lorem ipsum dolor sit amet consectetur adipisicing elit. Accusamus ipsa non aliquam. Recusandae eveniet amet inventore unde illum adipisci rem, similique deserunt blanditiis excepturi nemo vero voluptatum! Tempore, corporis voluptatem.

Read more

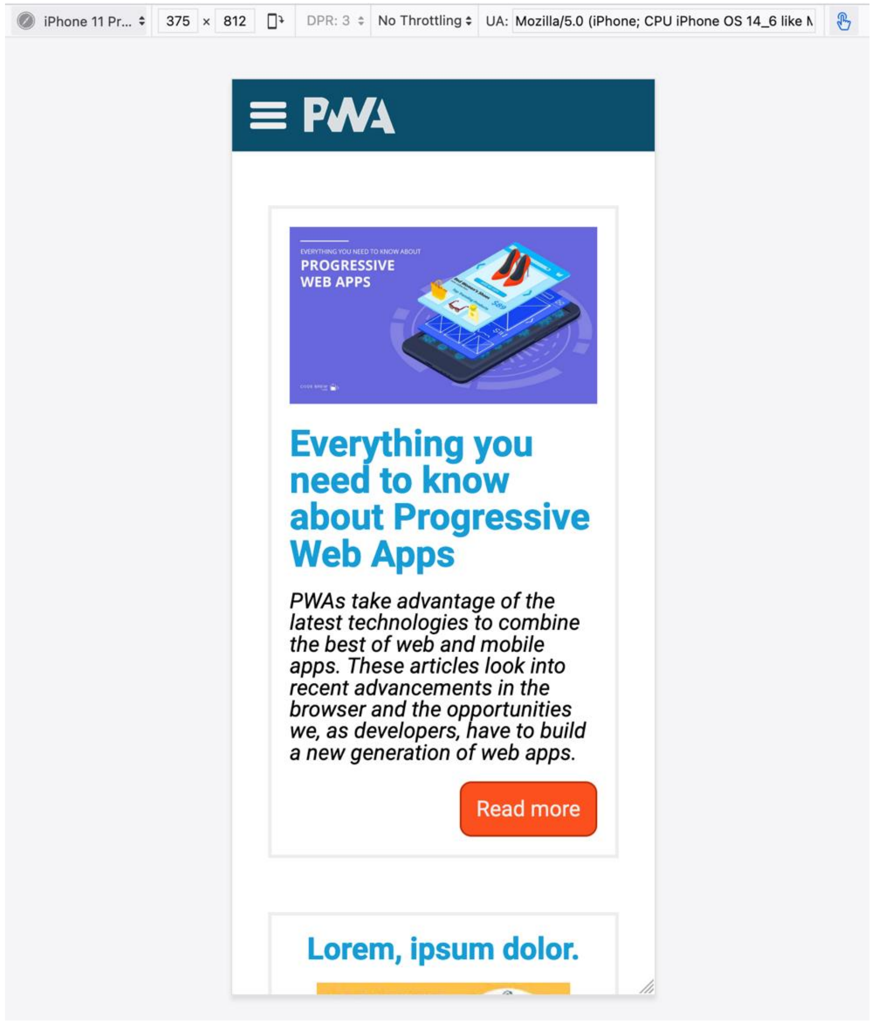
Lorem, ipsum dolor.

Lorem ipsum dolor sit amet consectetur adipisicing elit. Aliquam ad repellendus sint, odit quam dolorem voluptatem ducimus placeat quis earum.

Read more

Hogeschool West-Vlaanderen - Rijselstraat 5, 8000 Brugge

with viewport meta tag



iPhone 11 Pr... 375 × 812 DPR: 3 No Throttling UA: Mozilla/5.0 (iPhone; CPU iPhone OS 14_6 like M

PWA

Everything you need to know about Progressive Web Apps

PWAs take advantage of the latest technologies to combine the best of web and mobile apps. These articles look into recent advancements in the browser and the opportunities we, as developers, have to build a new generation of web apps.

Read more

Lorem, ipsum dolor.

Absolute vs relative units

Absolute vs relative units

Absolute
not relative to anything else, and are generally considered to always be the same size
e.g. cm, mm, px, pt, pc

Relative
relative to something else, perhaps the size of the parent element's font, or the size of the viewport, or the containing element.
e.g. em, ex, ch, rem, ...

Absolute vs relative units

Absolute
not relative to anything else, and are generally considered to always be the same size
e.g. cm, mm, px, pt, pc

Relative
relative to something else, perhaps the size of the parent element's font, or the size of the viewport, or the containing element.
e.g. em, ex, ch, rem, ...

https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/Values_and_units

Absolute vs relative units

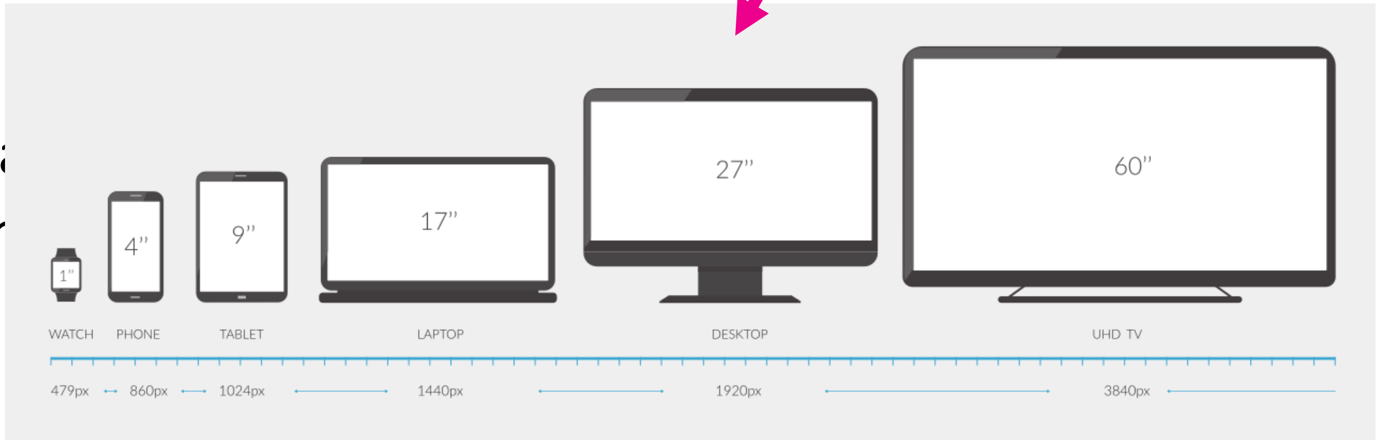
Absolute

not relative to anything else, and are generally considered to always be the same size

e.g. cm, mm, px, pt, pc

Relative

relative to something else, perh
of the viewport, or the containi
e.g. em, ex, ch, rem, ...



Absolute vs relative units

Absolute
not relative to anything else, and are generally considered to always be the same size
e.g. cm, mm, px, pt, pc

Relative
relative to something else, perhaps the size of the parent element's font, or the size of the viewport, or the containing element.
e.g. em, ex, ch, rem, vw, cqw, ...

→ with some careful planning you can make it so that
elements scale relative to everything else on the page

howest hogeschool

Fluid font sizes & spacing

You shall not use pixels

A pixel is not a pixel (1 CSS px != 1 device px)

Absolute units are not affected by font metrics, inherited property values, or the **viewport.**

Some users in/decrease the default font size of their browser to improve legibility. Pixel values disregard that change.

Rem and em

em
1em represents the calculated font-size of the element. If used on the font-size property itself, it represents the *inherited* font-size of the element.

rem
1rem represents the font-size of the root element (typically <html>). When used within the root element font-size, it represents its initial value (a common browser default is 16px, but user-defined preferences may modify this).

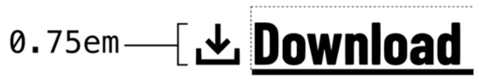
Rem and em

rem
calculates size relative to the font-size value of the **root** element (typically <html>). When used within the root element font-size, it represents its initial value (a common browser default is 16px, but users may modify this).

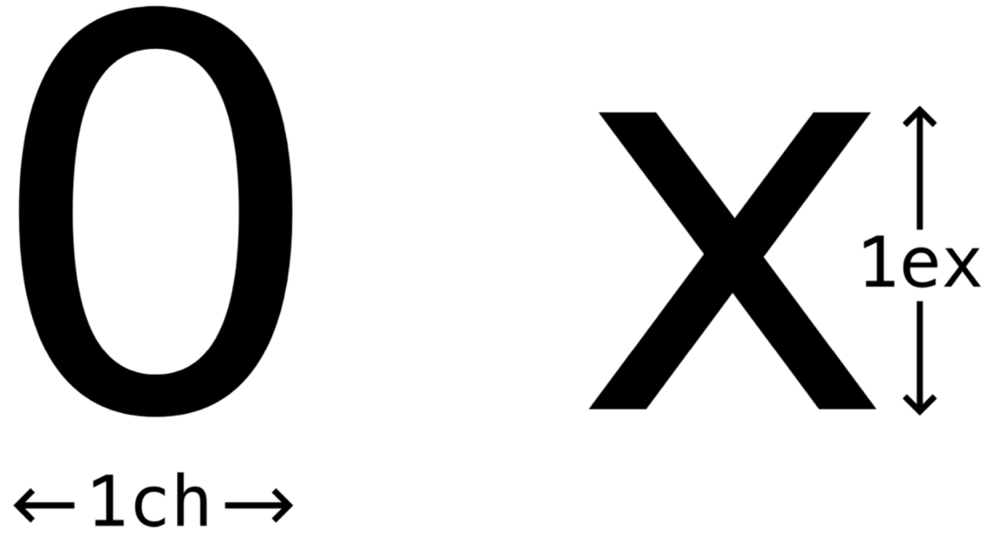
Better for sizing block elements

em
represents a proportion of the computed value of the font-size property for the element. It pertains to the **immediate context** rather than the outer document.

Better for sizing inline elements
e.g. SVG-icons:



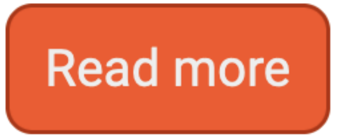
Other useful units: ch and ex

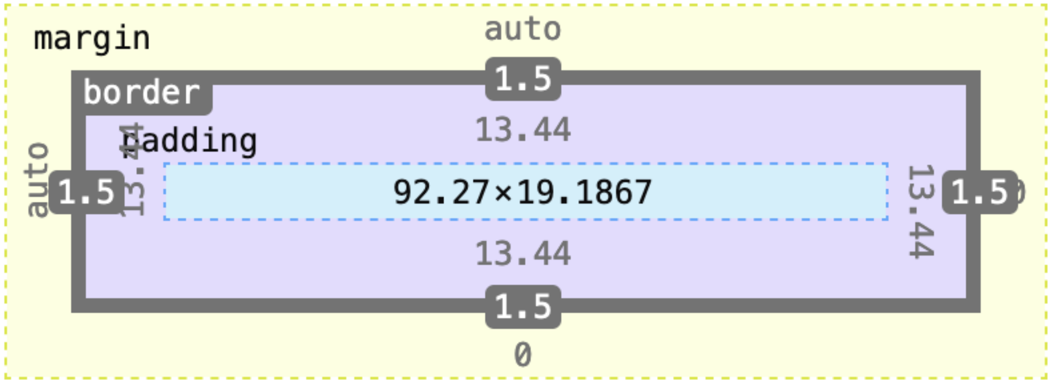


ch
Setting line-length of long text (because shorter lines improve legibility). Optimally around 55ch.
Also: headings, column widths, ...

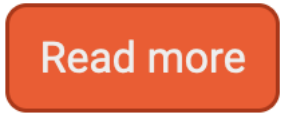
ex
adjusting letter-spacing, underline offset

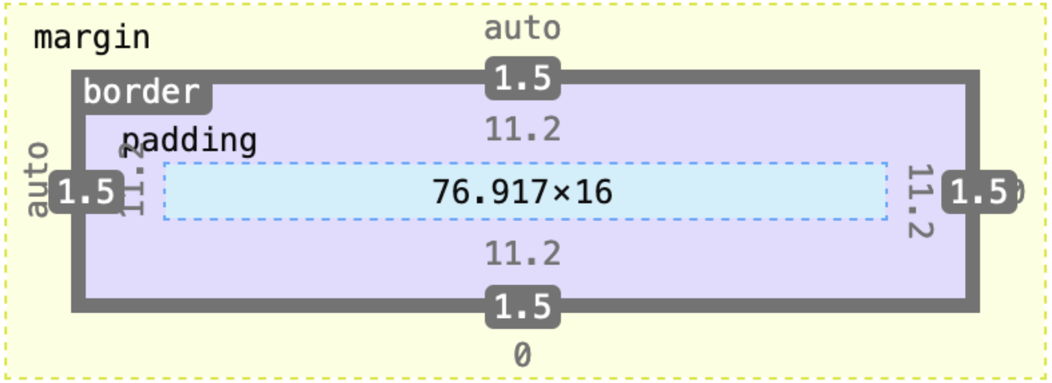
Buttons with dynamic padding





```
.read-more-button {
   padding: 0.7em;
}
```





Fluid font sizes

```
p {
   font-size: calc(1rem + 5vw);
}
```

The font size will grow as the screen gets wider.

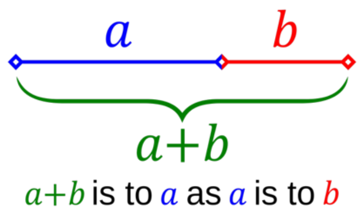
```
p {
   font-size: clamp(0.5rem, 4vw , 2rem);
}
```

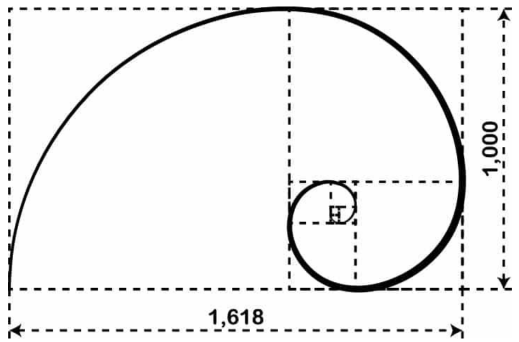
The **clamp()** CSS function clamps a middle value within a range of values between a defined minimum bound and a maximum bound. The function takes three parameters: a minimum value, a preferred value, and a maximum allowed value.

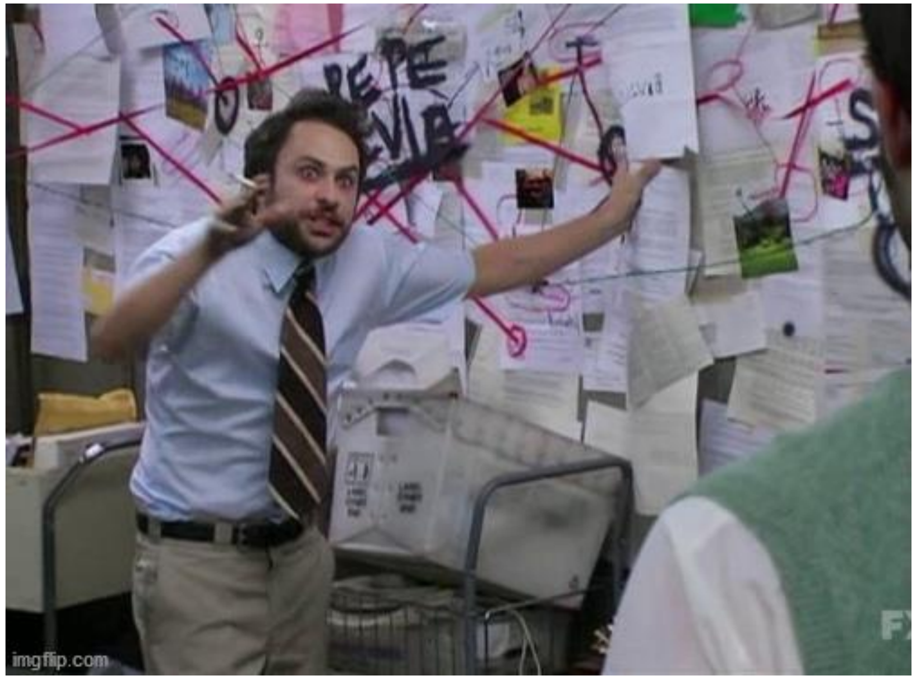
https://developer.mozilla.org/en-US/docs/Web/CSS/clamp

Fluid (font) sizes: taking it further

Visual harmony using a "modular scale" =
a sequence of numbers that relate to one another in a meaningful way.





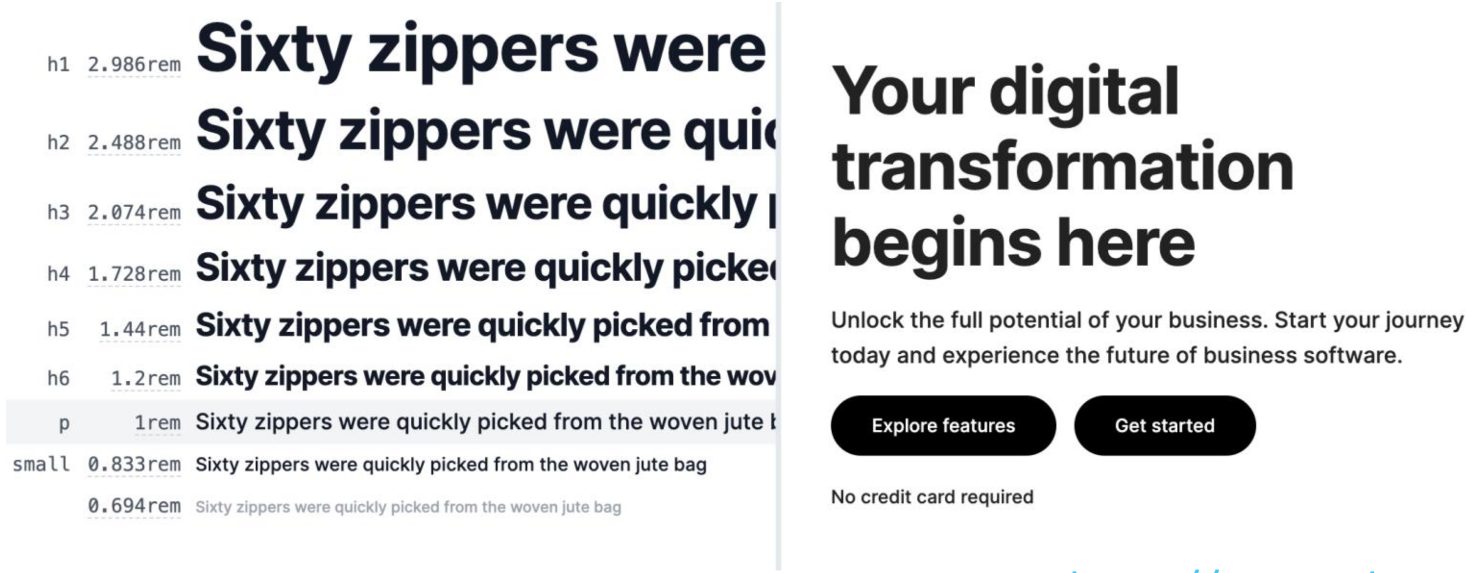


Using the **golden ratio**, for example, we can produce values for a modular scale by multiplying by 1.618 to arrive at the next highest number, or dividing by 1.618 to arrive at the next number down.

Harmonious fluid sizing: modular scale

Then choose numbers from the scale for font sizes, line height, line length, margins, paddings, column widths, ...

User benefit: page will always look “right”, regardless of the device being used



h1	2.986rem	Sixty zippers were
h2	2.488rem	Sixty zippers were qui
h3	2.074rem	Sixty zippers were quickly
h4	1.728rem	Sixty zippers were quickly picke
h5	1.44rem	Sixty zippers were quickly picked from
h6	1.2rem	Sixty zippers were quickly picked from the wov
p	1rem	Sixty zippers were quickly picked from the woven jute
small	0.833rem	Sixty zippers were quickly picked from the woven jute bag
	0.694rem	Sixty zippers were quickly picked from the woven jute bag

Your digital transformation begins here

Unlock the full potential of your business. Start your journey today and experience the future of business software.

No credit card required

https://typescale.com/

Creating a modular scale

Save yourself the headache: https://www.utopia.fyi

```
:root {
  --step--2: clamp(0.7813rem, 0.7747rem + 0.0326vi, 0.8rem);
  --step--1: clamp(0.9375rem, 0.9158rem + 0.1087vi, 1rem);
  --step-0: clamp(1.125rem, 1.0815rem + 0.2174vi, 1.25rem);
  --step-1: clamp(1.35rem, 1.2761rem + 0.3696vi, 1.5625rem);
  --step-2: clamp(1.62rem, 1.5041rem + 0.5793vi, 1.9531rem);
  --step-3: clamp(1.944rem, 1.771rem + 0.8651vi, 2.4414rem);
  --step-4: clamp(2.3328rem, 2.0827rem + 1.2504vi, 3.0518rem);
  --step-5: clamp(2.7994rem, 2.4462rem + 1.7658vi, 3.8147rem);
}
```

```
p {
  font-size: var(--step-0);
     line-height: var(--step-0);
}
```

```
main > * + * {
  margin-block-start: var(--step-1);
}
```

Intrinsic size

Intrinsic size

In CSS, the *intrinsic size* of an element is the size if it would be based on its content, if no external factors were applied to it.

min-content
The min-content sizing keyword represents the intrinsic minimum width of the content. For text content this means that the content will take all soft-wrapping opportunities, becoming as small as the longest word.

max-content
The max-content sizing keyword represents the intrinsic maximum width or height of the content. For text content this means that the content will not wrap at all even if it causes overflows.

https://developer.mozilla.org/en-US/docs/Glossary/Intrinsic_Size

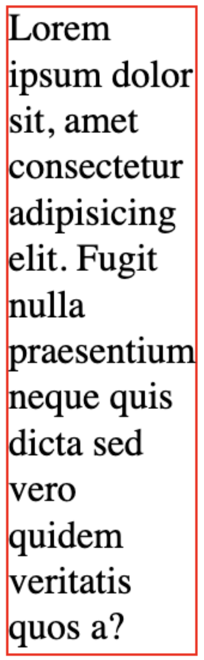
https://developer.mozilla.org/en-US/docs/Web/CSS/min-content

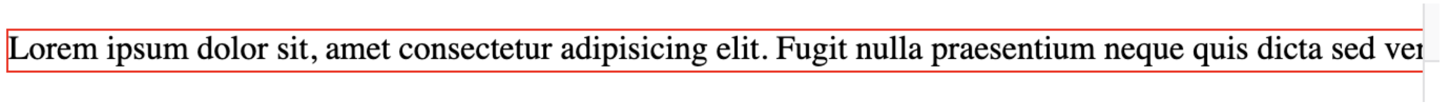
https://developer.mozilla.org/en-US/docs/Web/CSS/max-content

Intrinsic size

```
p {
  width: min-content;
  border: 1px solid red;
}
```

```
p {
  width: max-content;
  border: 1px solid red;
}
```


Lorem
ipsum dolor
sit, amet
consectetur
adipisicing
elit. Fugit
nulla
praesentium
neque quis
dicta sed
vero
quidem
veritatis
quos a?


Lorem ipsum dolor sit, amet consectetur adipisicing elit. Fugit nulla praesentium neque quis dicta sed ver

Intrinsic size

The **fit-content** behaves as fit-content(stretch). In practice this means that the box will use the available space, but never more than max-content.

The **fit-content()** CSS function clamps a given size to an available size according to the formula min(maximum size, max(minimum size, argument)).

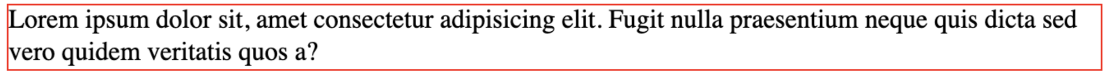
Not supported for width yet

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

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

Intrinsic size

```
p {
   width: fit-content;
   border: 1px solid red;
}
```


Lorem ipsum dolor sit, amet consectetur adipisicing elit. Fugit nulla praesentium neque quis dicta sed vero quidem veritatis quos a?

howest
hogeschool

Flexbox

Flexbox

Flexbox is a one-dimensional layout method for arranging items in rows or columns. Items *flex* (expand) to fill additional space or shrink to fit into smaller spaces.

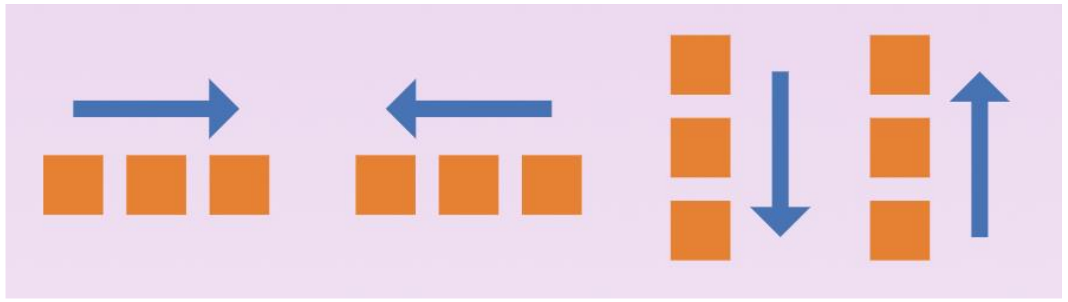
	CSS property	Some possible values
flex-flow	flex-direction	row, column, row-reverse, column-reverse
	flex-wrap	nowrap, wrap, wrap-reverse
	align-items	stretch, flex-start, flex-end, center
	justify-content	flex-start, flex-end, center, space-around, space-between
	align-content	stretch, flex-start, flex-end, center, space-between, space-around
flex	flex-grow	number
	flex-shrink	number
	flex-basis	length
	flex-gap	length

https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Flexbox

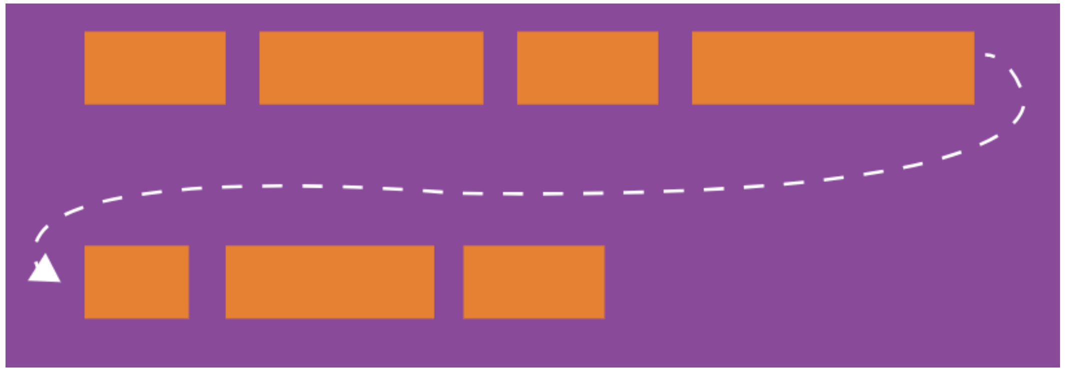
FLEX DIRECTION

determines the direction in which the flex should flow

```
main {
   display: flex;
   flex-direction: row | row-reverse | column |
column-reverse;
}
```



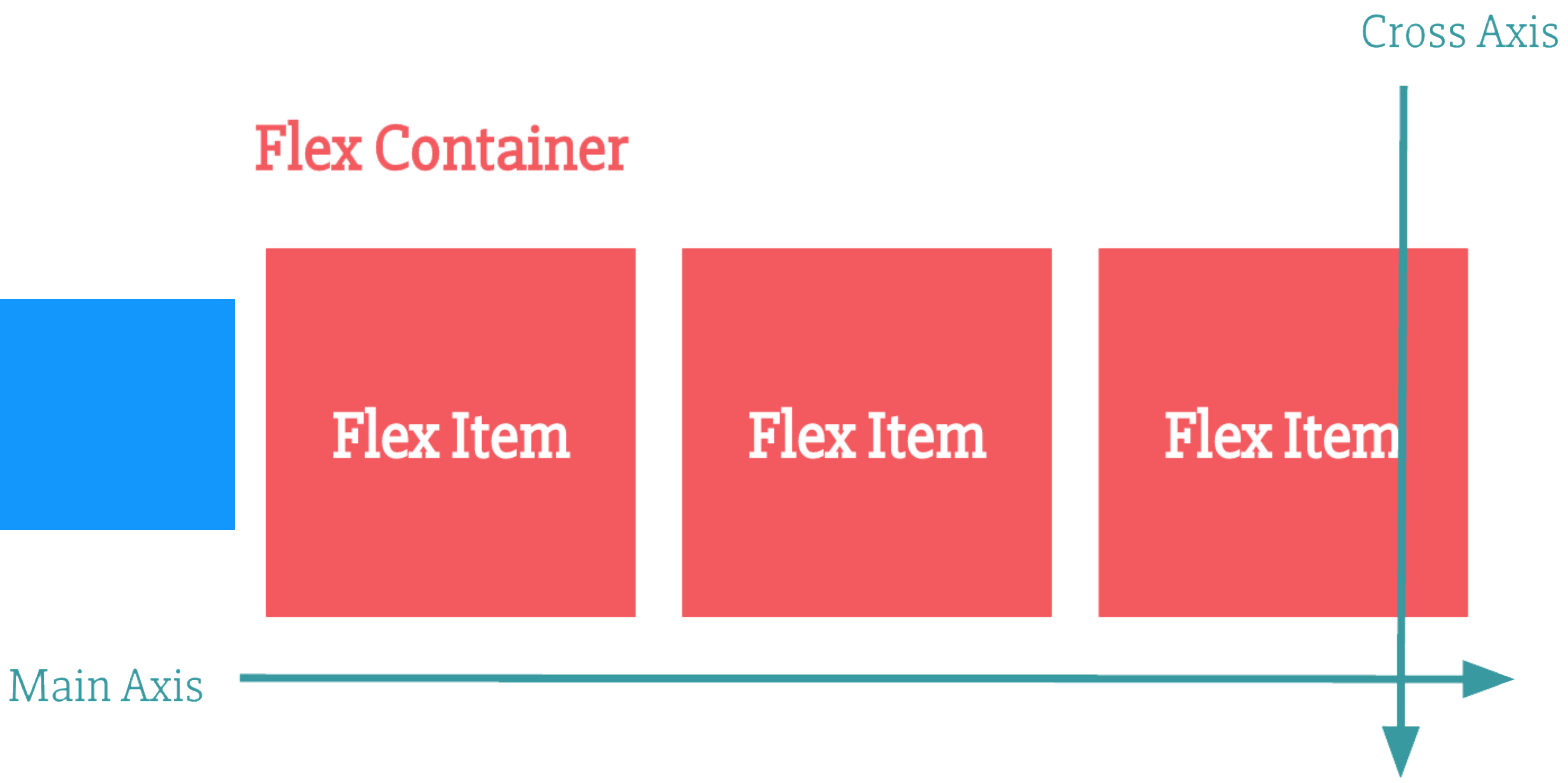
```
main {
  display: flex;
  flex-wrap: nowrap | wrap | wrap-reverse;
}
```



FLEX WRAP

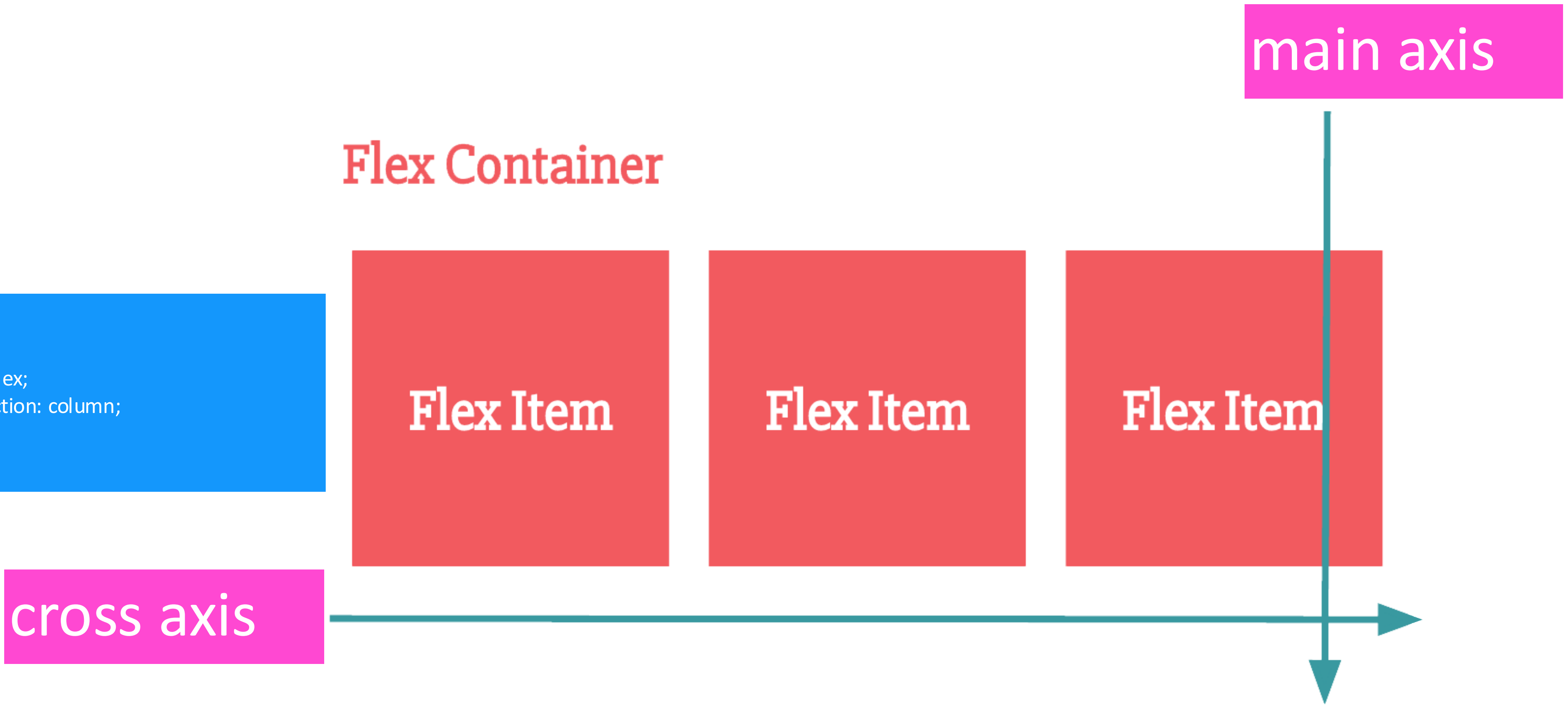
determines whether or not the items are allowed to wrap or forced on a single line (column widths will scale for the latter)

```
main {
    display: flex;
    flex-direction: row;
}
```



BOX ALIGNMENT SPECIFICATION

```
main {
    display: flex;
    flex-direction: column;
}
```



REMEMBER: AXIS + MAIN SIZE REVERSE WHEN CHANGING FLEX DIRECTION

	flex container	flex item
main axis	justify-content	
cross axis	align-items	align-self
main axis multiple line container	align-content	

ALIGNMENT PROPERTIES

The flex property

The **flex** CSS shorthand property sets how a flex *item* will grow or shrink to fit the space available in its flex container.

This property is a shorthand for the following CSS properties:

- flex-grow
- flex-shrink
- flex-basis

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

The flex property

CSS property	What does it do	Default value
flex-grow	sets the flex grow factor of a flex item's main size.	0
flex-shrink	sets the flex shrink factor of a flex item.	1
flex-basis	sets the initial main size of a flex item	auto (it’s complicated)

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

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

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

The flex property

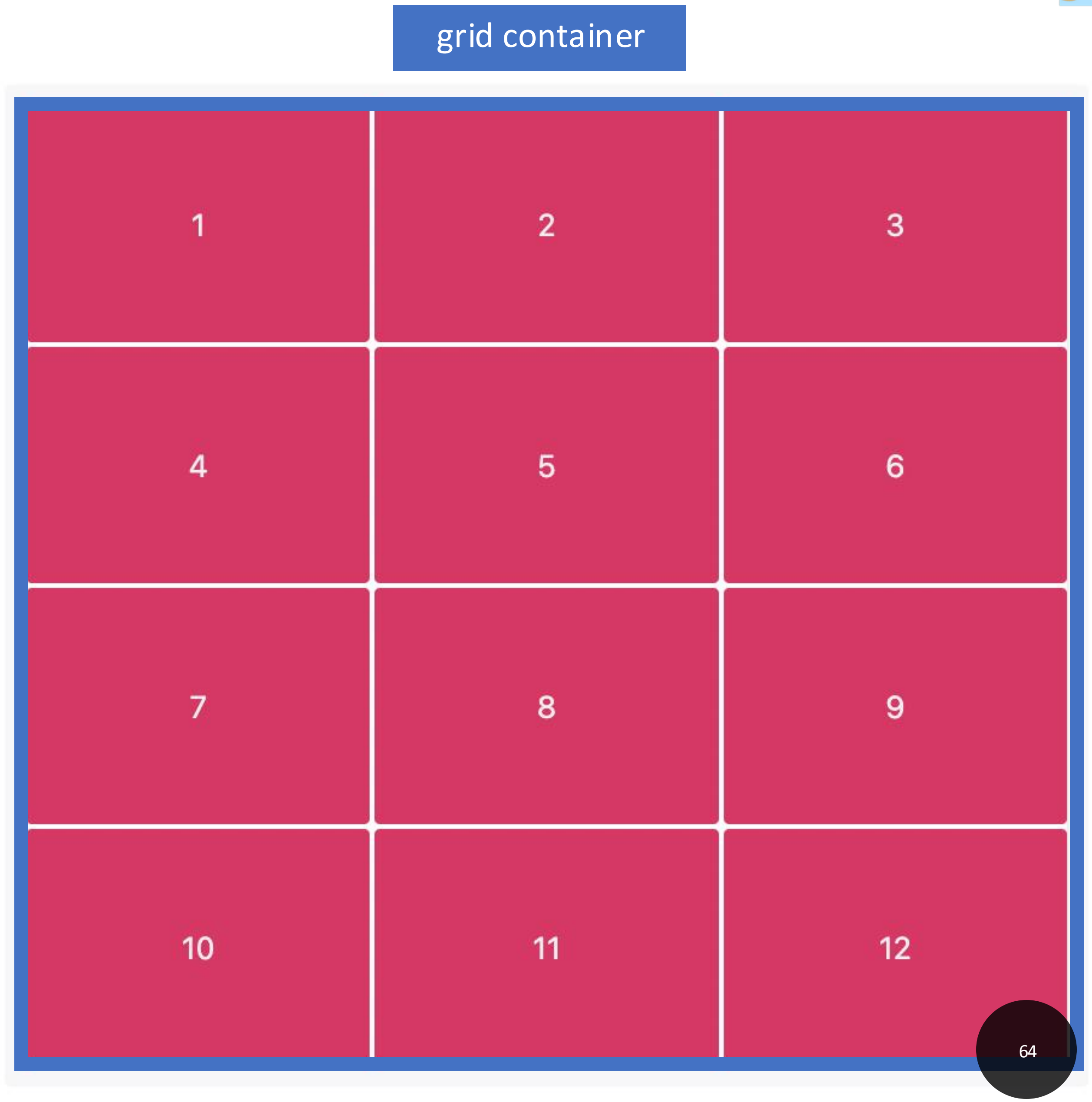
- Think about it in the following way
  - Initially the size of the flex-items is set according to their flex-basis value
  - It there isn't enough space, items are shrunk according to their flex-shrink value
    - When an item has a flex-shrink value of two, it will shrink twice as much as an item with a flex-shrink value of one
  - If the flex container has empty space, items grow according to their flex-grow value
    - When all flex items have a flex-grow of zero, nothing happens.
    - When an item has a flex-grow value of two, it will grow twice as much as an item with a flex-grow value of one

howest
hogeschool

CSS Grid

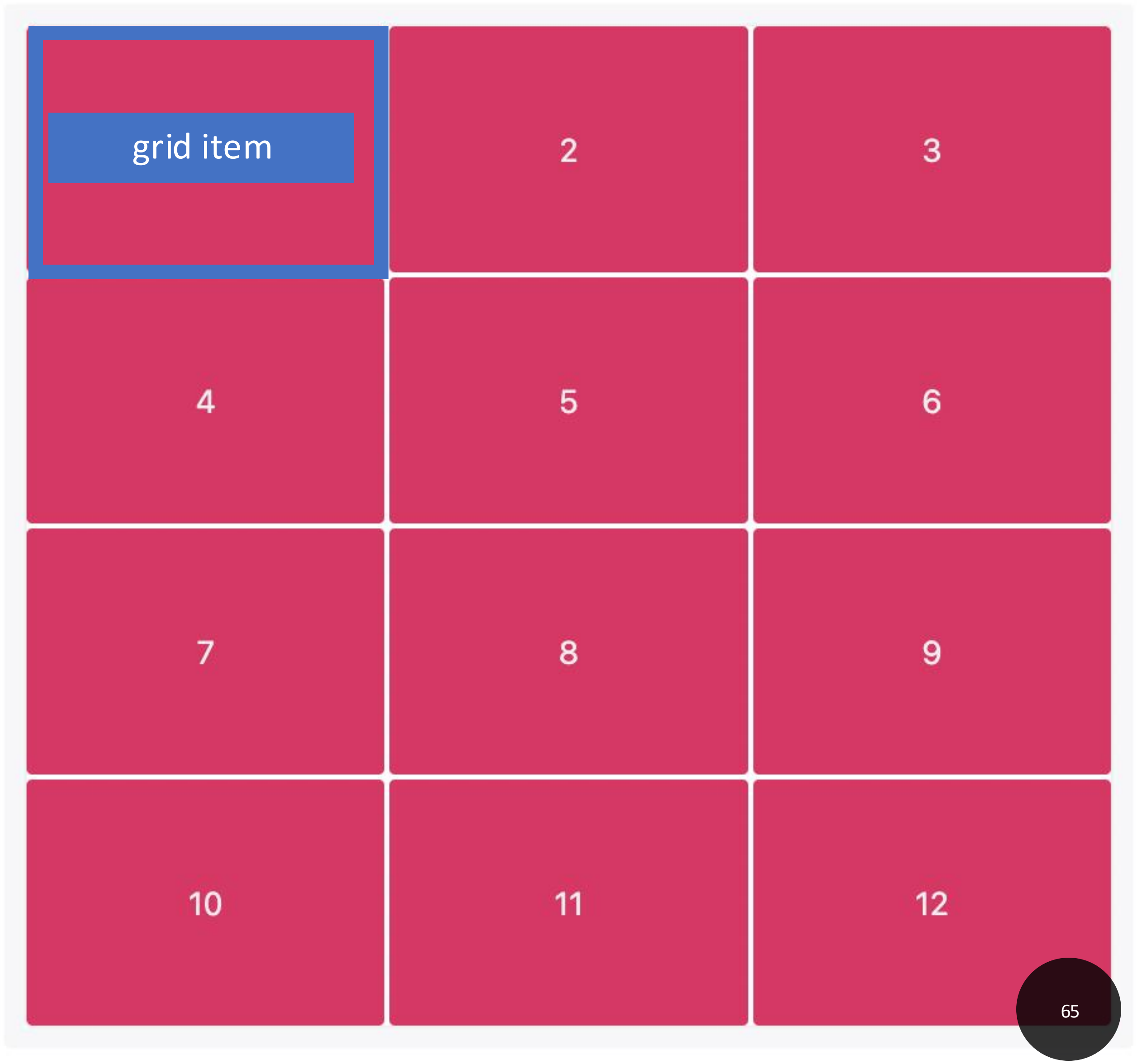
Basic concepts: grid

Container-based, just like flexbox



Basic concepts: grid

Container-based, just like flexbox



Grid container properties

The following properties should be applied to the grid container

How?

Apply display: grid; to the container

```
ul {
    display: grid;
}
```

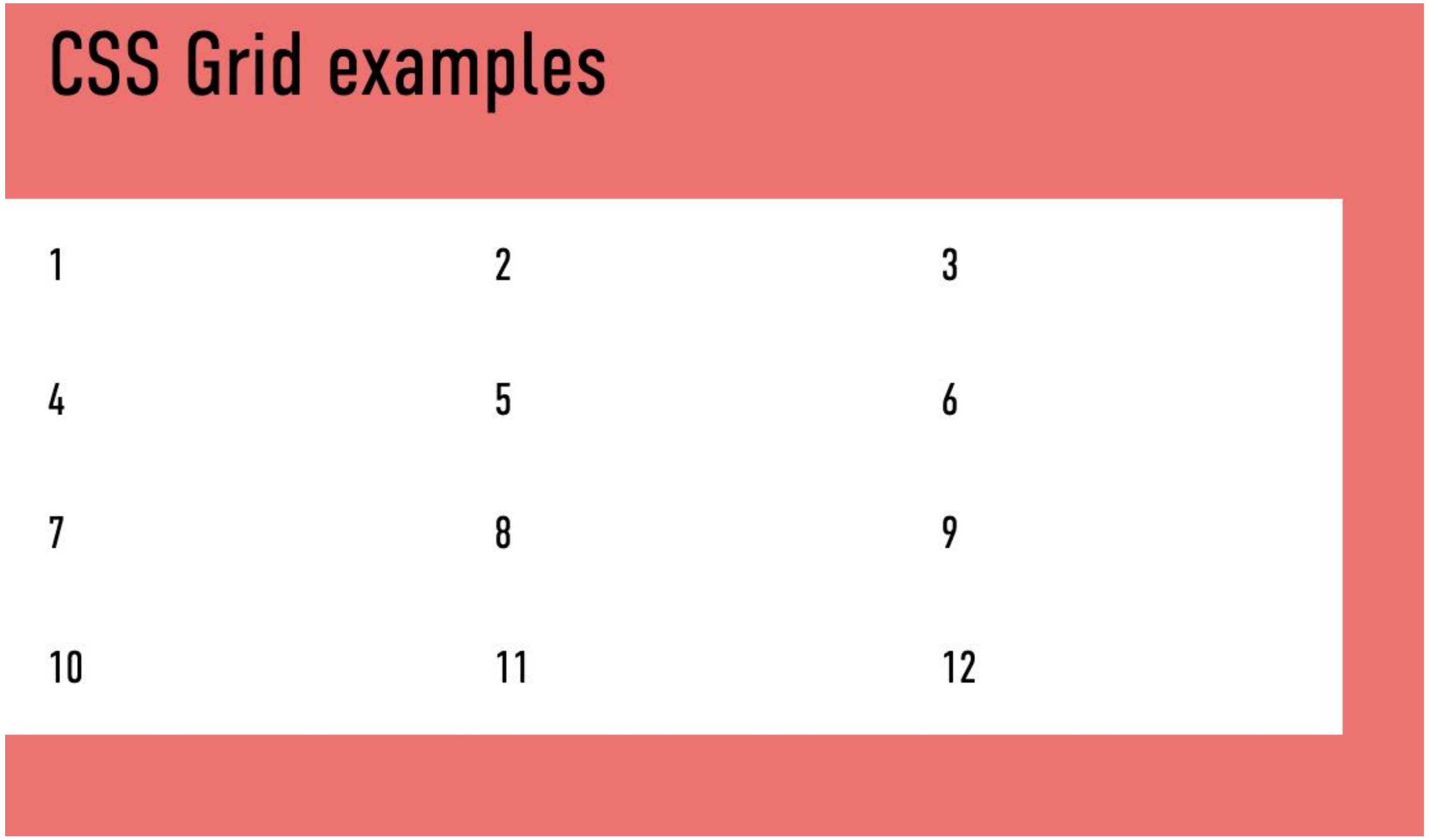
It’s a little different form flex

You need to specify the column size

```
ul {
    display: grid;
        grid-template-columns:
                        10rem 10rem 10rem;
}
```

This will give you a three-column layout

Each column being **10rem wide**, rows being **automatically determined** depending on the content.



CSS Grid examples

1	2	3
4	5	6
7	8	9
10	11	12

Grid has a special relative unit size

fr stands for “fraction”

It represents a fraction of the available space in the grid container (works like Flexbox’s unitless values).

Works on *columns* as well as *rows*.

Even though it grows and shrinks like the flex-shrink/flex-grow property, it is not the same.

```css
ul {
    display: grid;
      grid-template-columns:
                    1fr 5fr 3fr;

}
```

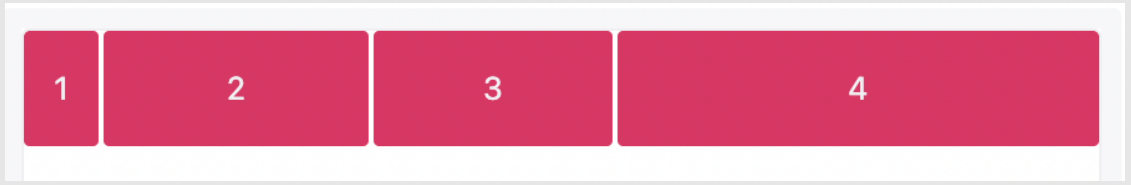
When combined with other values

fr will be **calculated based on the remaining space.**

In this example, 3rem and 25% will be subtracted from the available space before the size of fr is calculated:

1fr = *((width of grid) - (3rem) - (25% of width of grid)) / 3*

```
ul {
    display: grid;
        grid-template-columns:
                3rem 25% 1fr 2fr;

}
```



Repeating grid tracks

Define repeating grid tracks using the **repeat()** notation. This is useful for **grids with items with equal sizes** or many items.

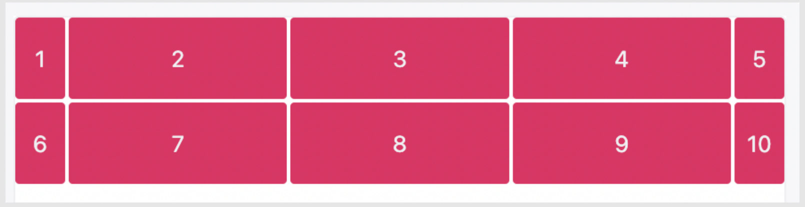
The repeat() notation accepts 2 arguments: the first represents ***the number of times the defined tracks should repeat***, and the second is the ***track definition***.

```
ul {

 display: grid;
 grid-template-columns: repeat(3, 1fr);
 grid-template-rows: repeat(4, 10rem);

}
```

Can be used in track listing

You can combine repeat in track listings as well, along with other values.

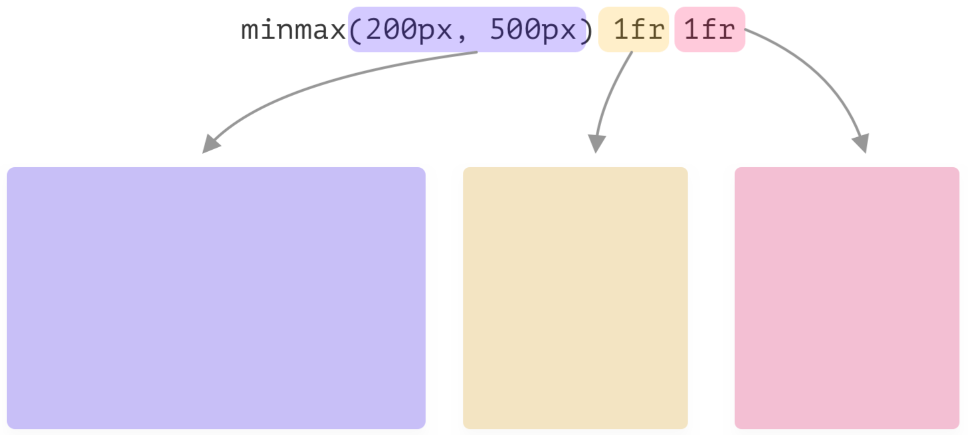
```
ul {

 display: grid;
 grid-template-columns:
                        2rem repeat(3, 1fr) 2rem;
 grid-template-rows:
                        repeat(4, 10rem);
}
```



Minmax

The **minmax()** CSS function defines a size range greater than or equal to *min* and less than or equal to *max*. It is used with CSS Grids.

```
#grid-container {
   display: grid;
   grid-template-columns:
            minmax(200px, 500px)
            1fr
            1fr;
   grid-gap: 1rem;
}
```



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

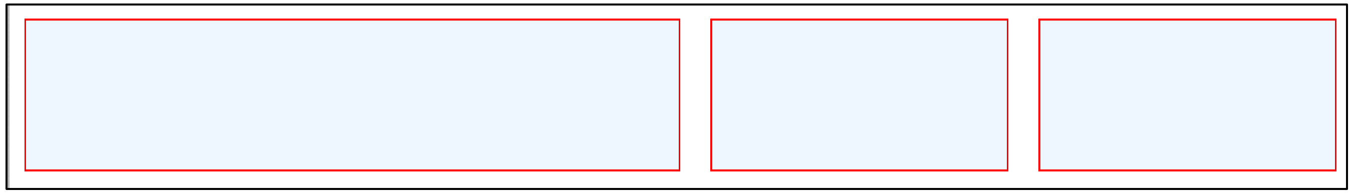
Minmax

Let the grid sizing algorithm figure it out

https://w3c.github.io/csswg-drafts/css-grid/#algo-overview

Some use cases

```
grid-template-columns: minmax(12rem, 1fr) 10rem 10rem;
```

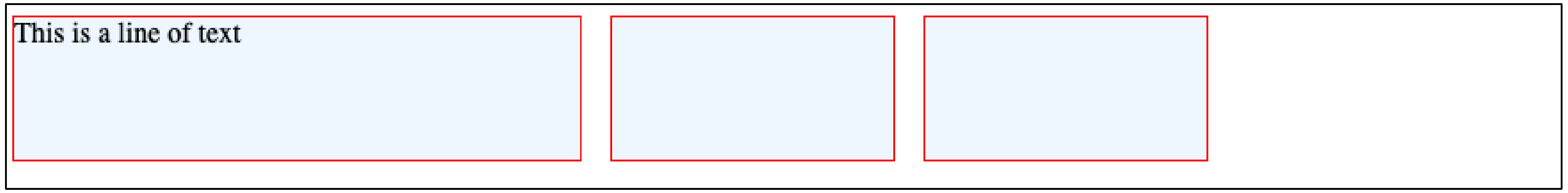


Last two columns are 10rem wide. First one takes remaining space, but is at least 12rem.

When the viewport width gets smaller than 32rem, horizontal scrollbars will appear

Some use cases

```
grid-template-columns: minmax(max-content, 20rem) 10rem 10rem;
```

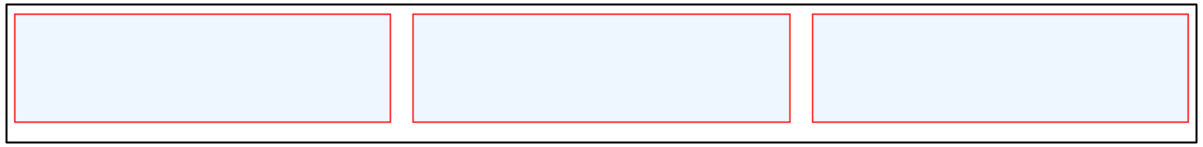


Last two columns are 10rem, first one at most 20rem and at least the value of max-content (no text is wrapped)

When the viewport width is smaller than 40rem, the grid will not span the full width

Some use cases

```
grid-template-columns: repeat(3, minmax(12rem, 1fr));
```



3 equal with columns that are at least 12rem wide

When the viewport width gets smaller than 36rem, horizontal scrollbars will appear

auto-fit / auto-fill

auto-fit and auto-fill are keywords that can be used as first parameter for the repeat function. They can be used to make the number of columns in your grid responsive.

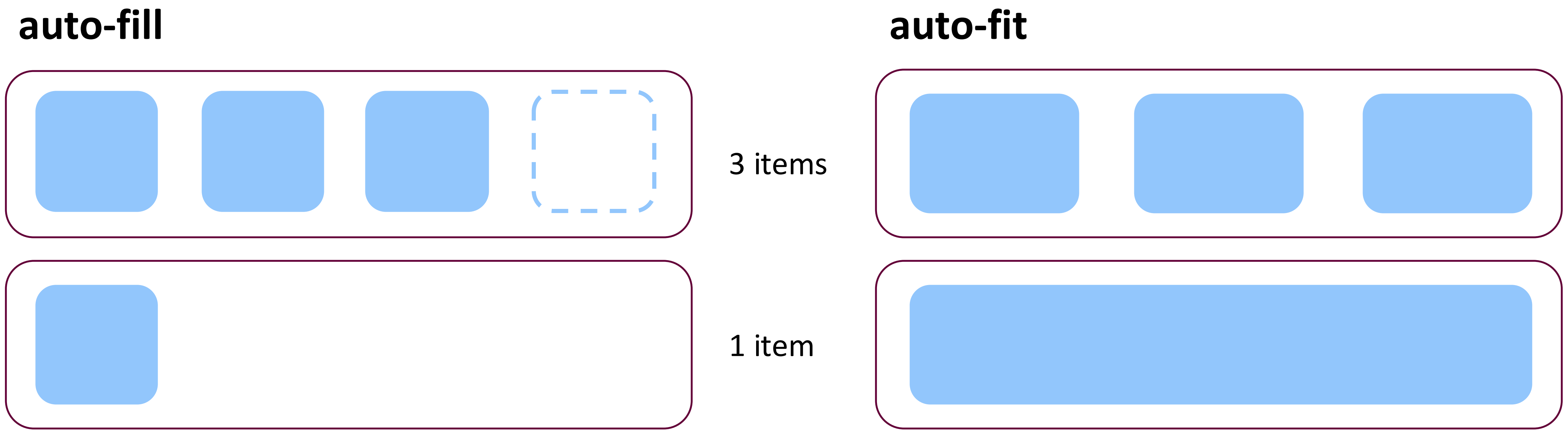
auto-fill
If the grid container has a definite or maximal size in the relevant axis, then the number of repetitions is the largest possible positive integer that does not cause the grid to overflow its grid container.

auto-fit
Behaves the same as auto-fill, except that after placing the grid items any empty repeated tracks are collapsed. An empty track is one with no in-flow grid items placed into or spanning across it.

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

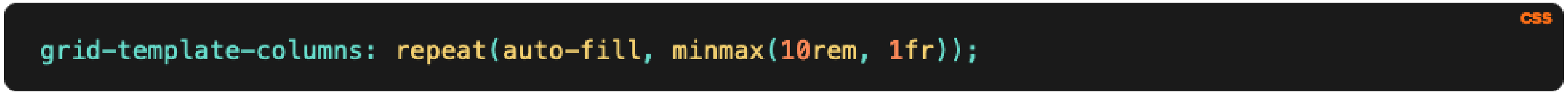
auto-fit / auto-fill

keywords that can be used as first parameter for the repeat function. They can be used to make the number of columns in your grid responsive.



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

The most famous line of code to have come out of CSS grid so far is:



```css
grid-template-columns: repeat(auto-fill, minmax(10rem, 1fr));
```

Without any media queries, that will set up a grid container that has a flexible number of columns. The columns will stretch a little, until there is enough room for another one, and then a new column is added, and in reverse.

All columns will have a minimum width of 10rem

https://css-tricks.com/intrinsically-responsive-css-grid-with-minmax-and-min/

Media queries

Media queries

A media query allows you to write CSS for a specific situation

You can target media-types

all
print
screen
…

You can target media-features

width
height
orientation
…

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

Media queries

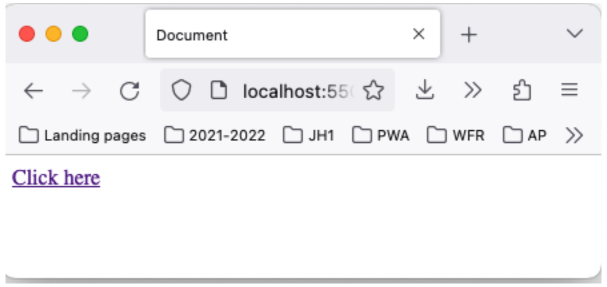
Rules put inside a media query will be applied when it's requirements (media type and features) are met.

```
@media print {
	/* RULES FOR A PRINT CSS */

	/* REMOVE NOT IMPORTANT INFORMATION HERE */
	/* e.g. navigation menu, links, banners */
}
```

Adding links to print css

```
<body>
  <a href="http://www.google.be">Click here</a>
</body>
```

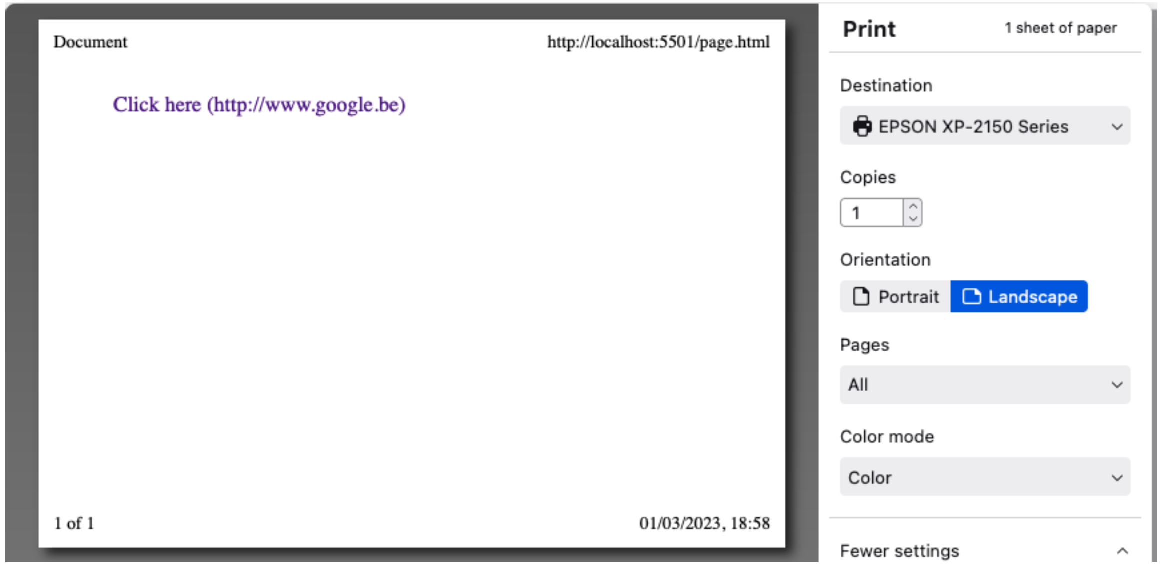
```
@media print {
  a::after{
    content: " (" attr(href) ") ";
  }
}
```


Document

localhost:55

Landing pages 2021-2022 JH1 PWA WFR AP

Click here


Document   http://localhost:5501/page.html

Click here (http://www.google.be)

1 of 1   01/03/2023, 18:58

Print   1 sheet of paper

Destination: EPSON XP-2150 Series

Copies: 1

Orientation: Portrait [✓ Landscape]

Pages: All

Color mode: Color

Fewer settings

Targetting screen size

```
@media screen and (max-width: 12450px) {
	/*
		HERE YOU CAN WRITE CSS RULES THAT BECOME ACTIVE
		WHEN THE SCREEN SIZE IS SMALLER THAN 12450px
	*/
}


@media screen and (min-width: 500px) {
	/*
		HERE YOU CAN WRITE CSS RULES THAT BECOME ACTIVE
		WHEN THE SCREEN SIZE IS LARGER THAN 500px
	*/
}
```

How to add them

1. You can link them seperately to your html page

```
<link rel="stylesheet" media="screen and (max-width: 640px)" href="smallscreen.css" type="text/css">
```

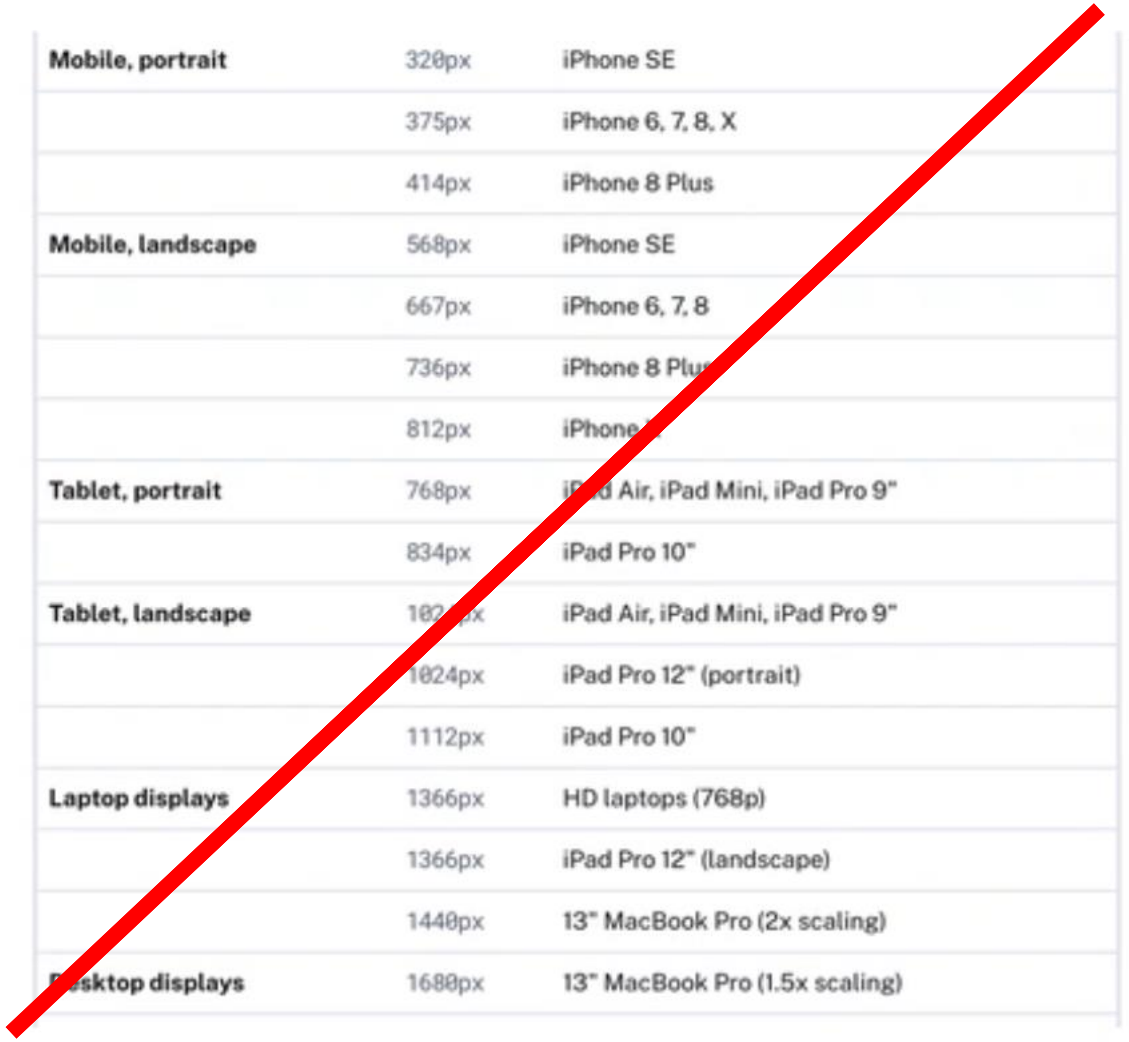
2. You can write them directly in your css using the @media directive

```
@media screen and (max-width: 640px) {

}

@media (400px <= width <= 700px) {


}
```

new since 2023

How to choose your breakpoints

- Old techniques involved target specific devices by specifying a device width
  - This does no longer work: too much devices, screen resolutions differ greatly, …



Mobile, portrait	320px	iPhone SE
	375px	iPhone 6, 7, 8, X
	414px	iPhone 8 Plus
Mobile, landscape	568px	iPhone SE
	667px	iPhone 6, 7, 8
	736px	iPhone 8 Plus
	812px	iPhone X
Tablet, portrait	768px	iPad Air, iPad Mini, iPad Pro 9"
	834px	iPad Pro 10"
Tablet, landscape	1024px	iPad Air, iPad Mini, iPad Pro 9"
	1024px	iPad Pro 12" (portrait)
	1112px	iPad Pro 10"
Laptop displays	1366px	HD laptops (768p)
	1366px	iPad Pro 12" (landscape)
	1440px	13" MacBook Pro (2x scaling)
Desktop displays	1680px	13" MacBook Pro (1.5x scaling)

How to choose your breakpoints

Modern Responsive Design Rule #1

You shall not use pixel-based breakpoints

Use relative units instead (e.g. rem)

How to choose your breakpoints

Modern Responsive Design Rule #2

You shall not match devices

Instead, look for when your design falls apart/ fails to work properly and target **that**

How to choose your breakpoints

Modern Responsive Design Rule #3

Only use media queries to tweak where necessary

Use as many **default** responsive behaviours as you can.