Responsive.Real-World.Websites.with.HTML.and.CSS

01 Welcome and First Steps DONE

004 A High-Level Overview of Web Development.en\_US

006 Your Very First Webpage!

! + Tab = set the basic structure.

010 Introduction to HTML

An element is Everything from the beginning tag to the end tag including them.

02 HTML Fundamentals

009 Section Intro

010 Introduction to HTML

011 HTML Document Structure

012 Text Elements

Really good practice to have just one of <h1>

<b> Bold tag </b> replaced with <strong> **strong** </strong>

<i> Italic </i> replced by <em> emphatized </em>

013 More Text Elements\_ Lists

<ol> ordered list (1-, 2-, 3-) //(parent element)

<li> List item </li> // child element

</ol>

<ul> unordered list (bullet points)

<li> List item </li>

</ul>

Text inside body is displayed without any tags by HTML (one line without any meaning).

014 Images and Attributes.

<img/> element does not hava content or closing tag

* Attributes = pieces of data use to describe elemnts (source attribute)

<img src = “image.jpg” />

* **Alt** attribute to describe the image ( also for blind people)
* **Witth/hight**  attribute

015 Hyperlinks

One of the fundamental building blocks of the internet

Hyperlinks or links are what actually enables the internet to be a world wide web.

Anchor element : <a> Name of the URL </a> + href (the URL attribute) =>

<a href = “ x” > Name </a>

Target attribute 🡪 <a href = “ x” target = “\_blank” > Name </a>

WHAT MAKES AN ANCHOR ELEMENT REALLY A LINK IS THE *HREF* PROPERTY.

But we can make a link which is not going anywhere

<a href = “#”> Challenges </a>

This behavior is going to be usefull later in the course.

016 Structuring our Page

<a href="blog.html">Blog</a>

<a href="#">Challenges</a>

<a href="#">Flexbox</a>

<a href="#">CSS Grid</a>

All of these could be grouped on <nav> </nav> element. (navigation – a invisible box that now contains these elements.

Also we can use <header > to group these in header

Also <article> can be used in our body. Also inside <article> we can have <header>. But also in the end of the article we have the footer ( on the same level as <article>)

Stay aware for semantic HTML.

Copyright symbol = HTML entity

We write an ampersand (&)

*&copy*

017 A Note on Semantic HTML

So in Html when we talk about sematics what we mean is that certain elements had actually a meaning or a purpose attached to them.

We shoul not think how the element looks like on the page but we should think about what that element actually means and what it stands for.

But not all elements in HTML are semantic (for exemple “div”)

Why we actually do that ?

Multiple reasons:

* But one important reason is that, the search engine optimization (search engine as google) will be able to understand the structure of your content
* Also for accessibility and especially for people who rely on screen readers to consume on web pages.

Html = what elements mean and what they stand for.

018 Installing Additional VS Code Extensions

019 CHALLENGE #1

<aside> </aside> element

The aside is generally used for secondary information tha compliments the information in the main part of the page.

Basically some related post to the article( the main part)

Usually we use the aside element as a sidebar but It doesn t have to be that way.

75x75 px

020 CHALLENGE #2

Codepan.io – this is a place online where we can basically write our code and it will then automatically execute the code and render it and it makes it also easy to share that code.

A good way of sharing code on udemy in case of any problems.(and not only on udemy)

<https://codingheroes.io/resources/>

<https://css-tricks.com/snippets/html/glyphs/>

03 CSS Fundamentals

021 Section Intro

022 Introduction to CSS

CSS ( cascadind style sheets) describes the ciseal style and presentation of the content written In HTML

CSS consists of countless properties that developers use to format the content: properties about font, text, spacing, layout, etc.

Declaration block:

Selector {

Declaration/style

Declaration/style

Declaration/style

}

023 Inline, Internal and External CSS

Internal CSS <style> tag </style> in the head of html document.

The sepatation of cencerns . (writing css separately)

Css file could have any name but usually is called style.css

Tell our html file about css file—we do that in html head  **because that is exactly where all the information about the page goes.** 🡪link element

The rel attribute defines the relationship between a linked resource and the current document. Valid on <link>, <a>, <area>, and <form>, the supported values depend on the element on which the attribute is found.

024 Styling Text

025 Combining Selectors

Sans – serif is repetitive instace for all selelctor so:

h1,

h2,

h3,

h4,

p,

li {

font-family: sans-serif;

}

Descendent selector :

* + Footer p {

}

026 Class and ID Selectors

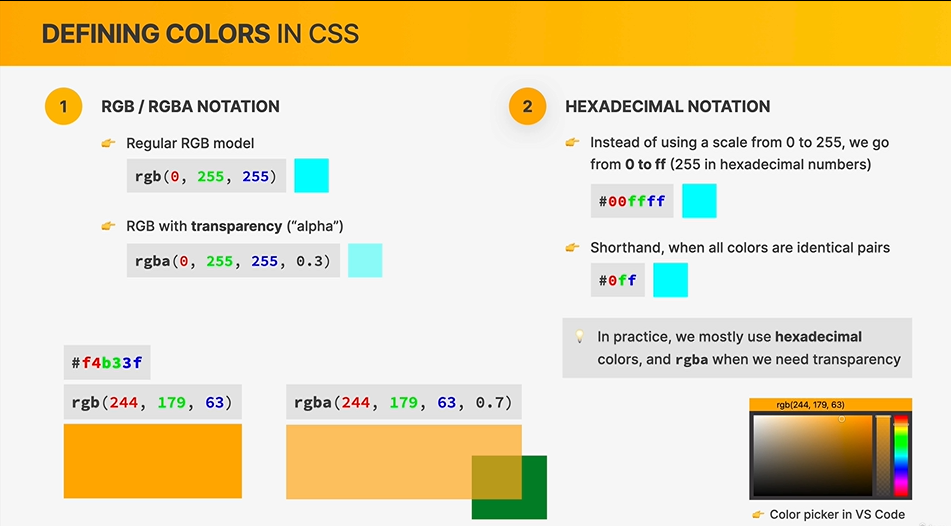
Why to use a class instead of ID even we use that only one time. It bcz we are preparing for the future. At some point later maybe is we are gonna use this class again for extra content on our page.

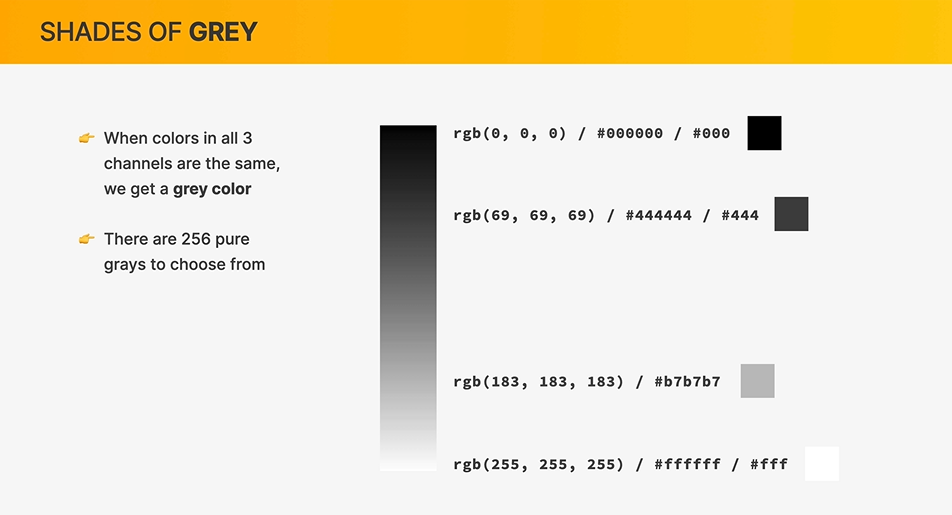
It will save us a lot from many changes in the future.

027 Working With Colors

RBG model: every color can be represented by a combination of RED,GREEN and BLUE.

Each of these colors can take a value between 0 and 255, ehich leads to 16.8 million different colors.

2 ways of writing colors in css . 



028 Pseudo-classes

Css automatically figure out which is the first element inside a cointainer:

Ex. : li:first-child {}

A CSS pseudo-class is a keyword added to a selector that specifies a special state of the selected element(s). For example, the pseudo-class :hover can be used to select a button when a user's pointer hovers over the button and this selected button can then be styled.

article p:first-child {

color: red;

}

This is not gonna work if the <p> element is not the first element in the article/.

029 Styling Hyperlinks

/\* Styling links \*/

a:link {

/\*anchor with a href attribute \*/ }

a:visited {

/\* color: #777; \*/

}

Most important one:

a:hover {

/\* Styles applied to the anchor as soon the element is hovered by the mouse\*/

}

a:active {

/ \* the state in which we are actually clicking\*/

}

/\* Alwyas this order: LVHA\*/

030 Using Chrome DevTools

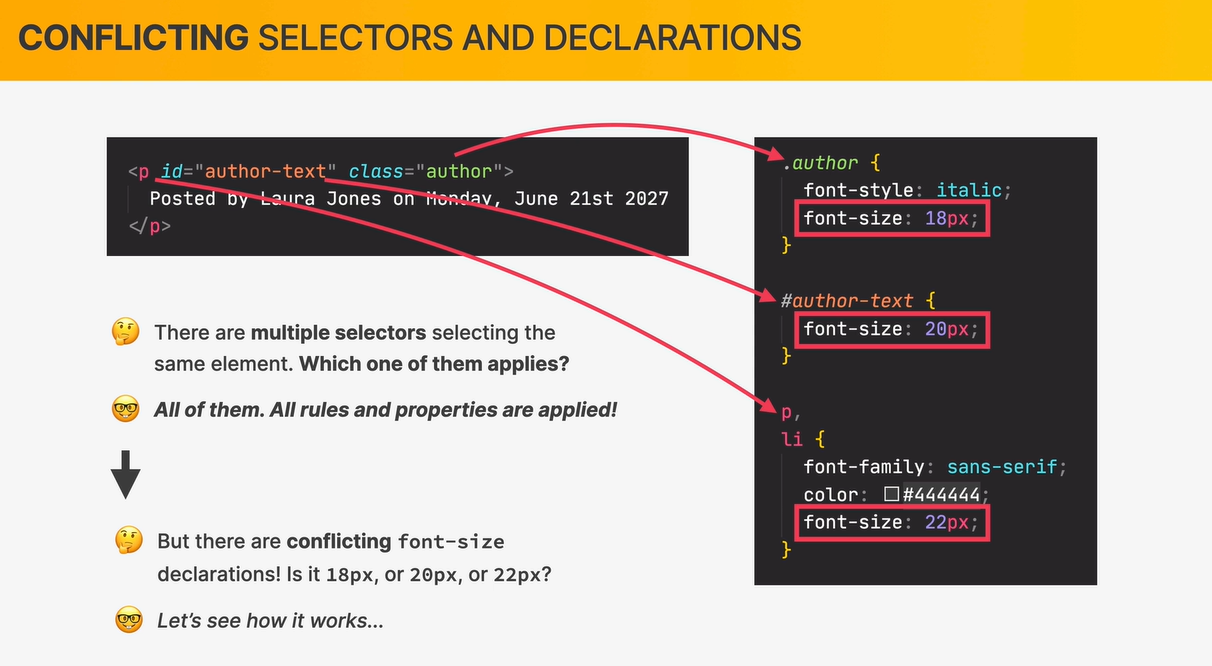
user agent stylesheet = default styles

hover =: to position (a computer cursor) over something (such as an image or icon) without selecting it.

031 CSS Theory #1\_ Conflicts Between Selectors

“What happens when there are multiple CSS rules?” that applies to the same element.

Conflicting selectors.



Conflicting declarations

Highest priority

* Declaration marked !important
* Inline styles
* ID # selector 🡪 Multiple 🡪 Last selector in code applies
* Class (.) or pseudo-class (:) selector  Last selector in code applies
* Element selector (p, div, li, etc.) Last selector in code applies
* Universal selector (\*)
* It’s a bit more complicated in reality

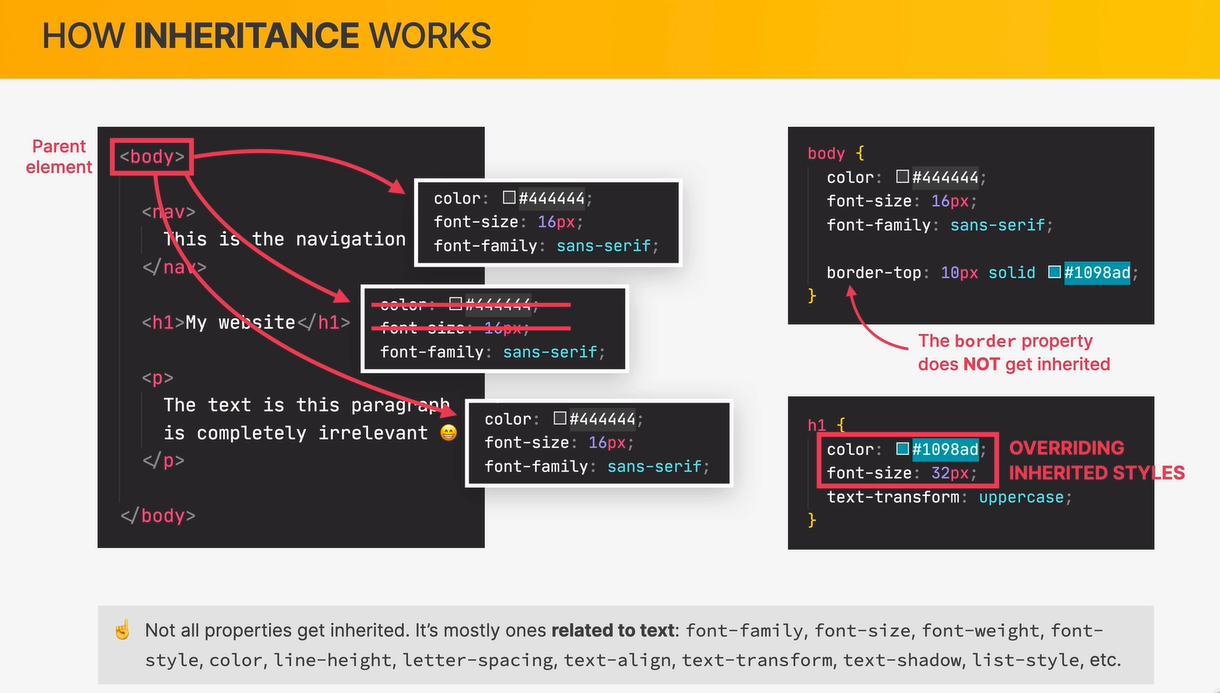
032 CSS Theory #2\_ Inheritance and the Universal Selector

= inheritance is a mechanism by which some styles so some properties get values inherited from parents elements to child elements.

Setting some properties on the body will be inherited by all the child elements.

NOT all the properties do actually get inherited, it is mostly the ones that are about text. Exemple: border-top property.

Inherited values are the ones who have the lowest priority.



Universal selector simply applies to all elements and there is NO INHERITENCE AT ALL.

033 CHALLENGE #1

033 CHALLENGE #2

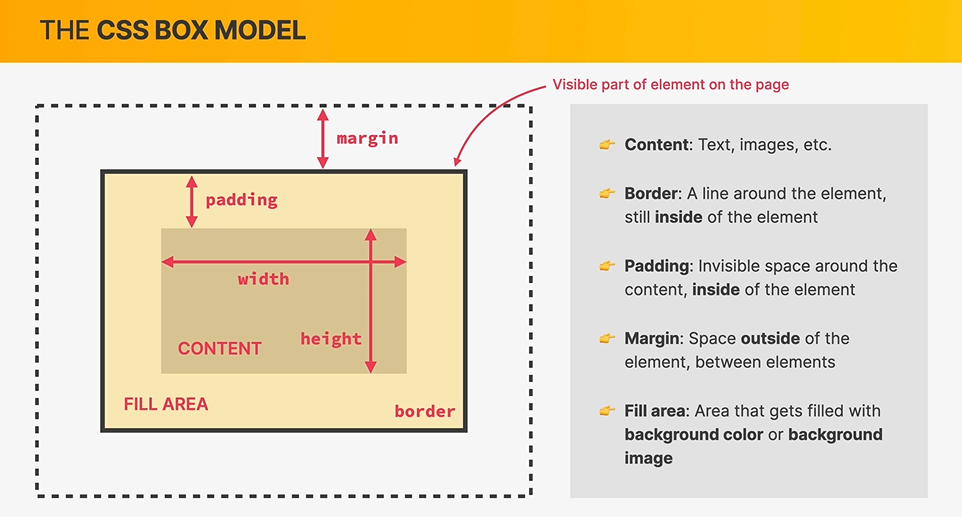
033 CHALLENGE #3

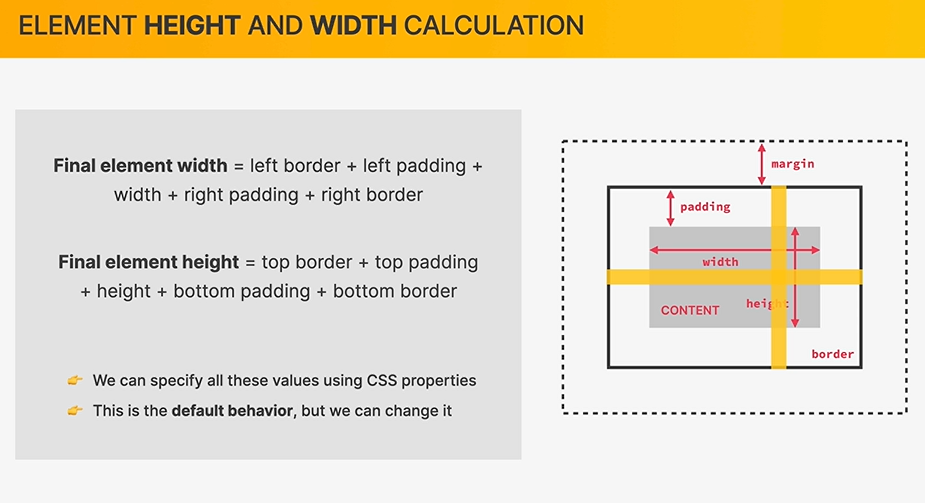
034 CSS Theory #3\_ The CSS Box Model

The box model defines how elements are displayed on a web page and how they are sized. So in this box model each and every element on a webpage can be seen as a rectangular box and each of these boxes can have content, a border and space inside and outside of it.

Padding and border are inside element

Marigin are around elements.





035 Using Margins and Paddings

Reset the margins and padding

* Margin Is not a property that is related to text so it is not a property that gets inherited.

To create space between elements: margin-top or margin-bottom.

**Collapsing magins** – basically when we have two margins that occupied the same space, only one of them is actually visible on the page. ( usually the larger one- they are not being summed).

036 Adding Dimensions

Height: auto (if we set width) = this is only necessaire is the height was already specified in html before;

If we don’t specify any image dimensions in html, then if we set height or the width using css the other onw will automatically adapt in order to account the original aspect ratio of the image.

Using procentages is better for responsive (will take the size of the parent container).

037 Centering our page .\

<div> container

.container {

width: 700px;

margin-left: auto;

margin-right: auto;

}

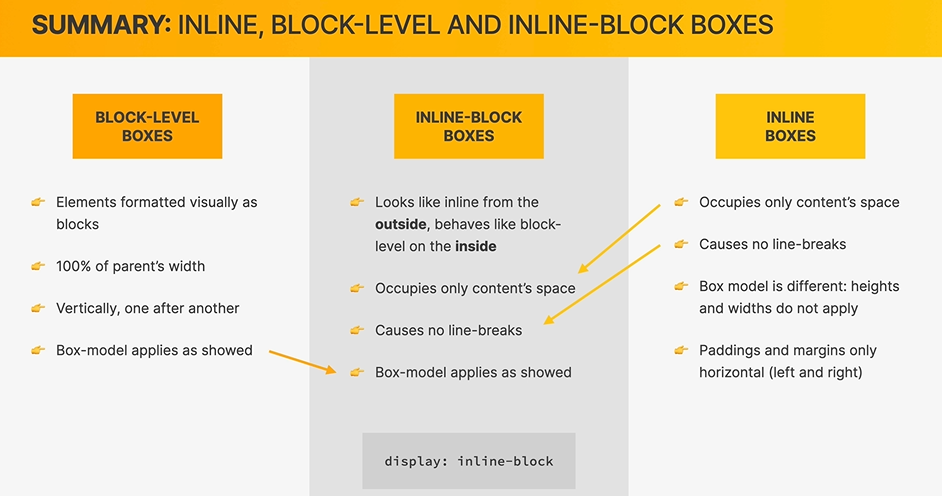
038 Challenge #2

You can’t use margins on the small elements like <a> link…

039 CSS Theory #4\_ Types of Boxes

The type of boxes that only occupied exactly the space that they need for its content is what we call inline boxes.

In the other hand all the boxes like <p> <h3> we call them block level boxes or block level elements



* The anchor element is an inline element where the box model does not create any vertical space (margin or padding). Actually for padding some space will be created but that is only true for inside of the element, because in fact, the anchor content itself it’s still in the exact same place. If this was a block-level element, then all od this would have moved down a little bit.

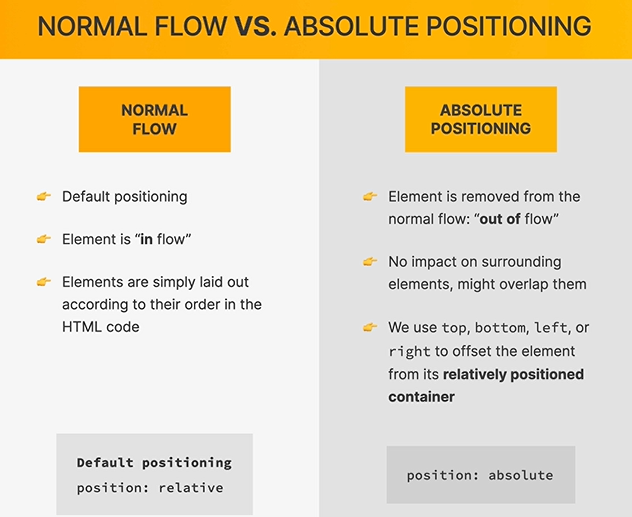
In conclusion the space has been expanded an dilled with the background color, **but no new space was in fact created.**

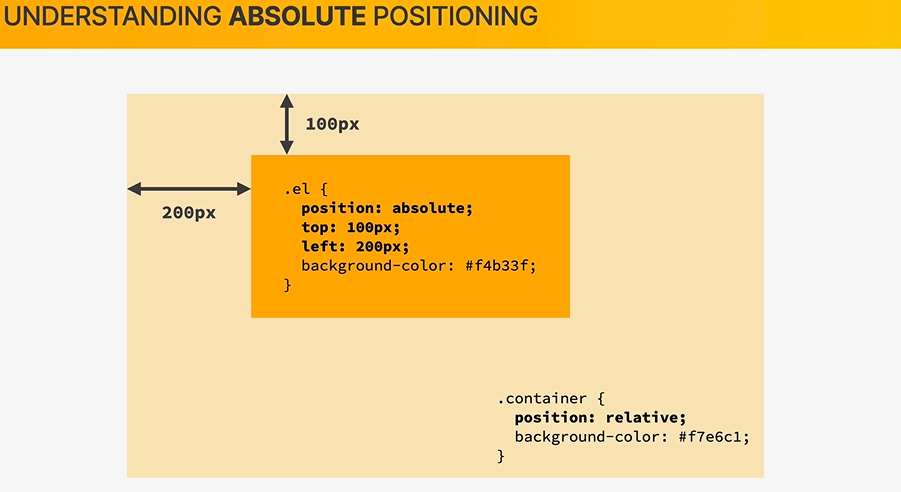
* **Images** are actually inline-block boxes. They do behave like inline-block elements and we can easily demonstrate this by adding some margin.

As a take away from this lecture most of the time we will let the display setting alone and simply use the predefined box type because by default they do actually make a lot of sense.

However, we can use display set to block in order to make a small inline element, for exemple, occupy an entire line.(is something that we will sometimes need in the real world.

040 CSS Theory #5\_ Absolute Positioning





041 Pseudo-elements.

= are essentially elements that don’t exist in HTML but that we can still select and style in CSS.

Some common examples like that are the first letter of a paragraph or the first line of a pargraph.

🡪pseudo class -> one :

🡪pseudo element -> two ::

A siblillng element is basically an element that is part of the same parent.

The adjacent siblings is a sibling that is actually the very next element.(right after it)

🡪MOST USED AND MOST IMPORTANT pseudo element:  **after and before pseudo elements.**

By defauld any pseudo element is actually an inline element. 🡪 so we need display: inline-block !

* **The after pseudo element creates a pseudo element that will automatically be the very first child of the selected element.**
* **The before works exactly in the same way but it will be before/\.**

042 Developer Skill #1\_ Googling and Reading Documentation

043 Developer Skill #2\_ Debugging and Asking Questions

Html validator

Validator.v3.org

Div checker

Basically in vs code – the higher the number the more that selector apply.

044.CHALLENGE #3

04 Layouts\_ Floats, Flexbox, and CSS Grid Fundamentals

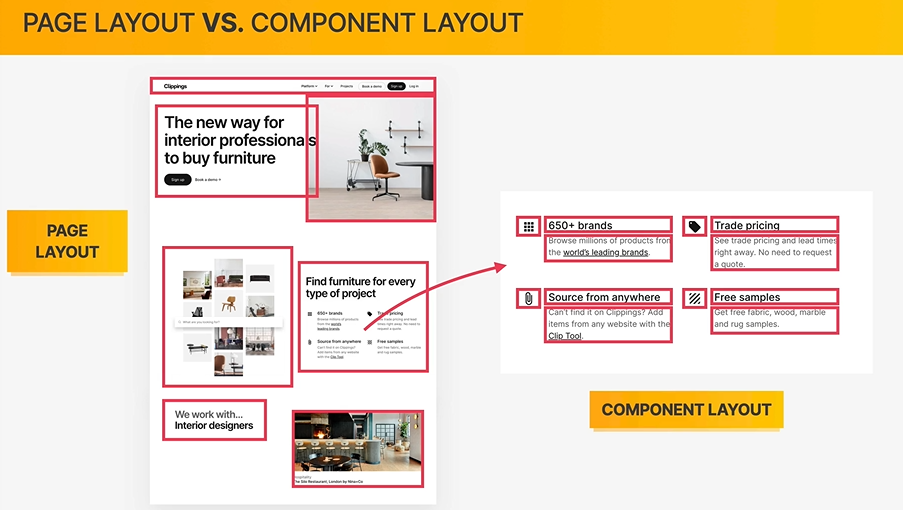
045 Section intro

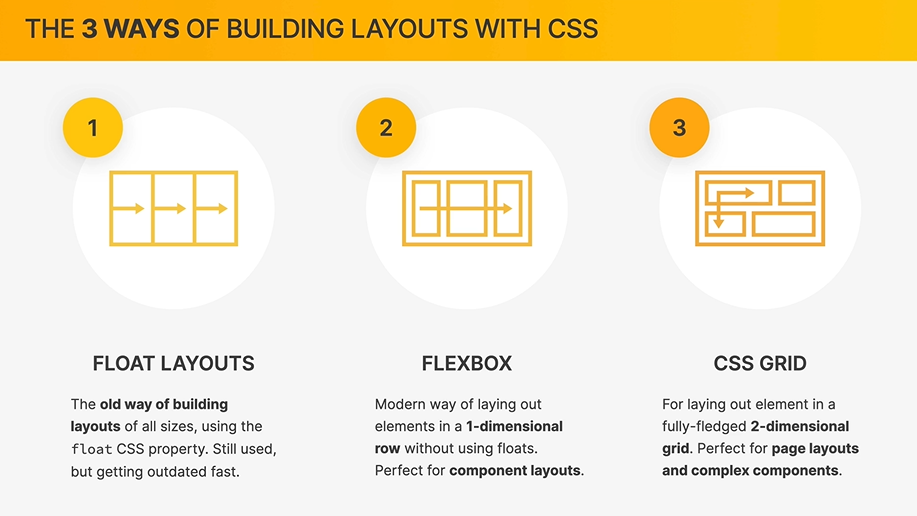
* Build layouts 🡪 how to build layouts using floats and olso using two modern technologies (**Flex Box and CSS Grid**)

046 The 3 Ways of Building Layouts

3 ways in which we can build layouts right now

What does a layout mean ?

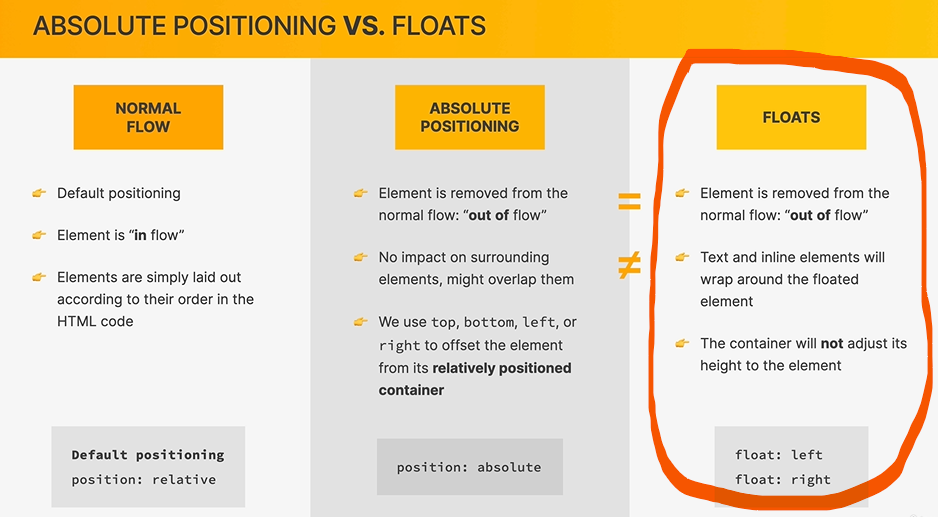




047 Using Floats

**How the flow property actually works ? (the goal of this lecture)**

LOREM (trick in vs code) 🡪 lorem ipsum fake text



048 Clearing Floats

**Quickly learn how to clear floats**

Building float layouts is kinda outdated… so you do NOT need to understand 100% how all of this works, specially for the “clear”. And usind a div and for clearing clear property is kinda a bad practice.

Trick: ( to have idea about it) **clearfix hack.**

* So the clearfix hack is something that has been widely used for many years in the CSS community.
* And the reason for that is exactly to avoid having “these empty divs” in the html file

How clearfix hack works ?

.clearfix::after = basically create a new element,which will be the less child element of the container (header)

* Remember for pseudo elements: it only appear if we declare something for content property( even a empty text)
* Also pseudo elements need a block level element (display:block)

049 Building a Simple Float Layout

article {

width: 825px;

float: left;

}

aside {

width: 300px;

float: right;

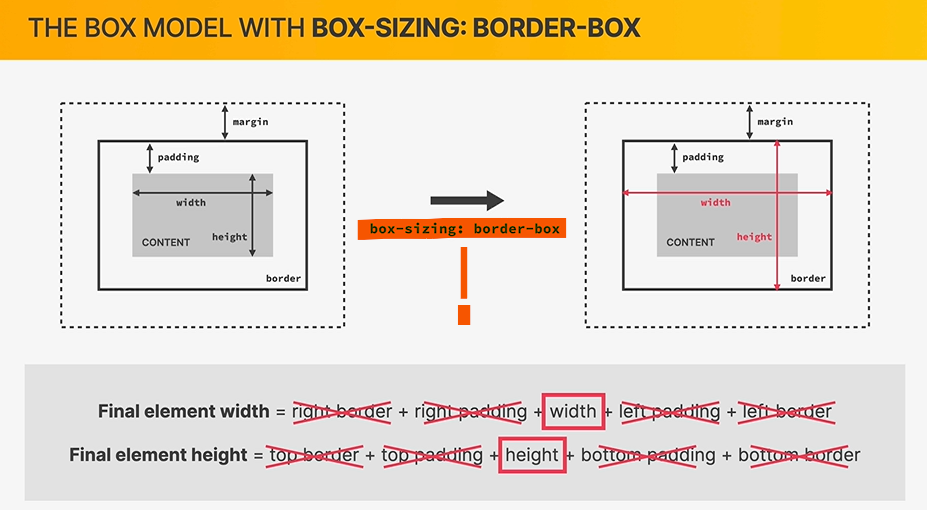
}

footer {

clear: both;

}

050 box-sizing\_ border-box



051 CHALLENGE #1

Done.

052 Introduction to Flexbox

PRACTICE

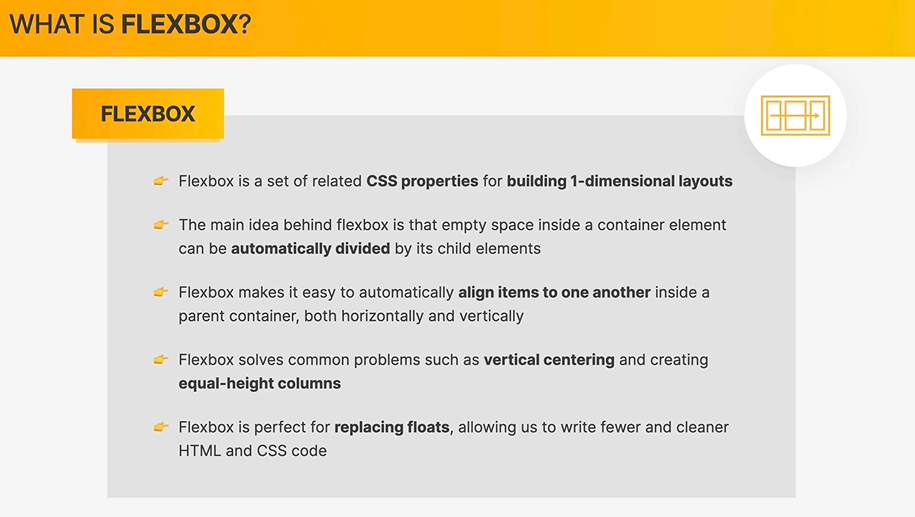
/\* FLEXBOX \*/

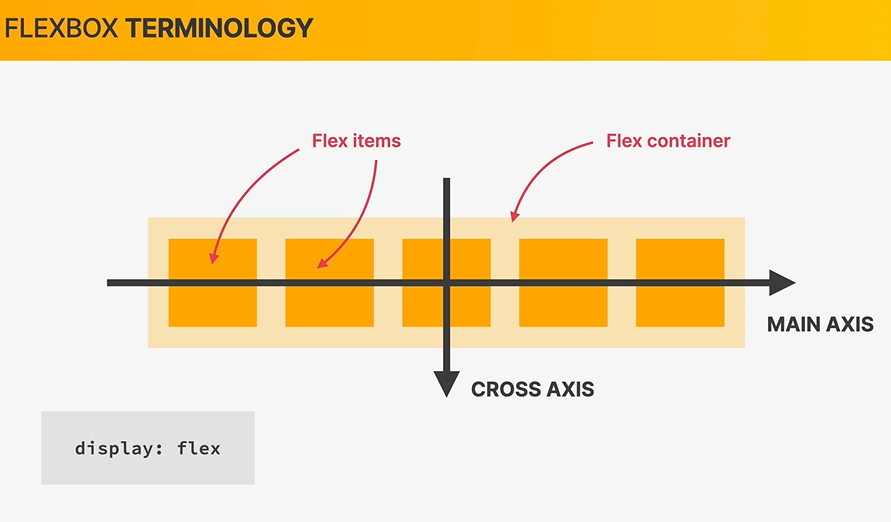
display: flex;

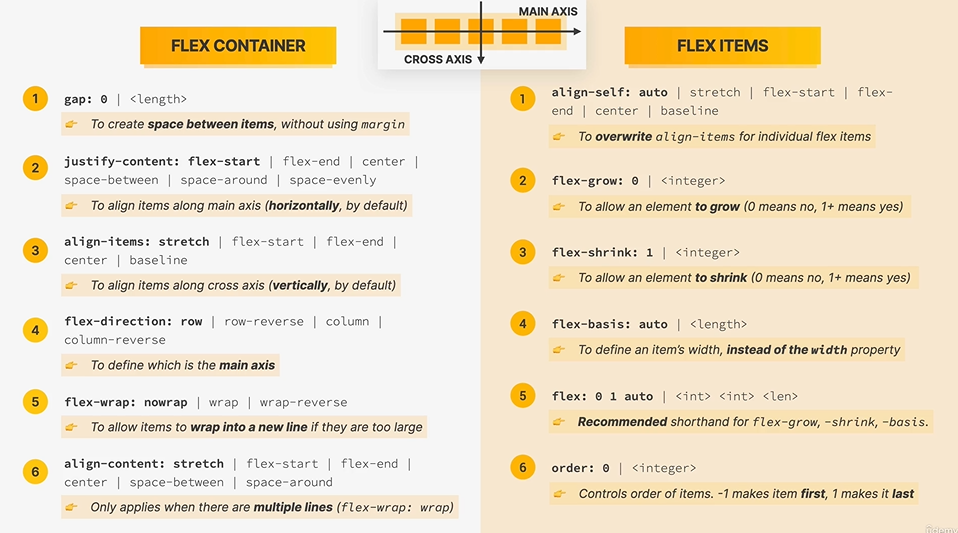
align-items: center;

justify-content: space-between;

053 A Flexbox Overview







054 Spacing and Aligning Flex Items

Align-items & justify-content 🡪 are both properties that apply to the flex container.

Justify-content is used to align items along main axis. (horizontally by default) – (“space between ”, “centered”, “flex-strat”.

Order property 🡪 changing the order in which flex items are displayed on the page can be very helpful when we are adapting a bigger layout to a small layout for mobile screens.

The default value for all elements is always 0.

The most prominent property for justify-content: center, space-between or space-around.

055 The flex Property

**= which is the property that we use in order to actually size flex items.**

Default value for properties bellow:

flex-grow: 0;

flex-shrink: 1;

flex-basis: auto;

* So when we want to size flex items and in particular with a width, then we usually do not use the witdh property, **but instead we use flex basis.**
* By default flex box is allowed to shrink elements so that they fit the container.( this is why flex-shrink is set to 1.)
* Flex-grow is basically the opposite of that(shrink) and it could be 1(the empty space is equal distributed to all elements) or it could be 2,3 ,4 etc
  + Exemple: if the empty space is gonna be 600 and we have 5 elements
    - For el1 {flex-grow: 2 🡪 =200px} 🡪 el2-5 = 100 px

Flex property itself which is a shorthand for these 3 above

* Flex: 0 0 200px

056 Adding Flexbox to Our Project

We will use flex box: the element that is gonna be the flex container is the parent of the elements that we want to use flex box on it .( so .main-header in out case)

Whenever we have just two flex items and we want to push each of them to one side then the pefect thing to do is *justify-content.*

057 Building a Simple Flexbox Layout

Done

**058 CHALLENGE #2**

Done.

059 Introduction to CSS Grid

So Css Grid is right now the most modern way of building layouts and its also the most complete one and in Jonas opinios is even the easiest way of building layouts (at least if you use only the fundamentals).

Like in FlexBox in CSS Grid we have Grid container and Grid items.

/\* CSS GRID \*/

display: grid;

grid-template-columns: 200px 200px 100px 100px;

grid-template-rows: 300px 200px;

/\* gap: 30px; \*/ (=grid-gap)

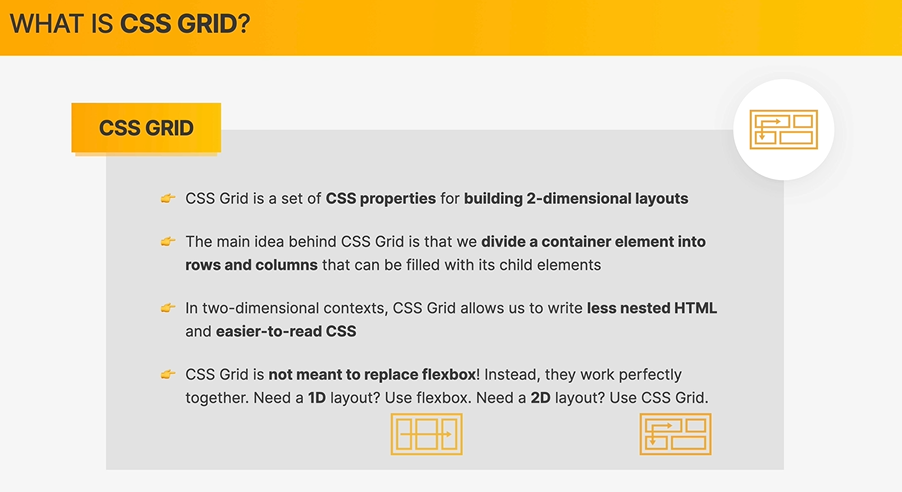
column-gap: 30px;

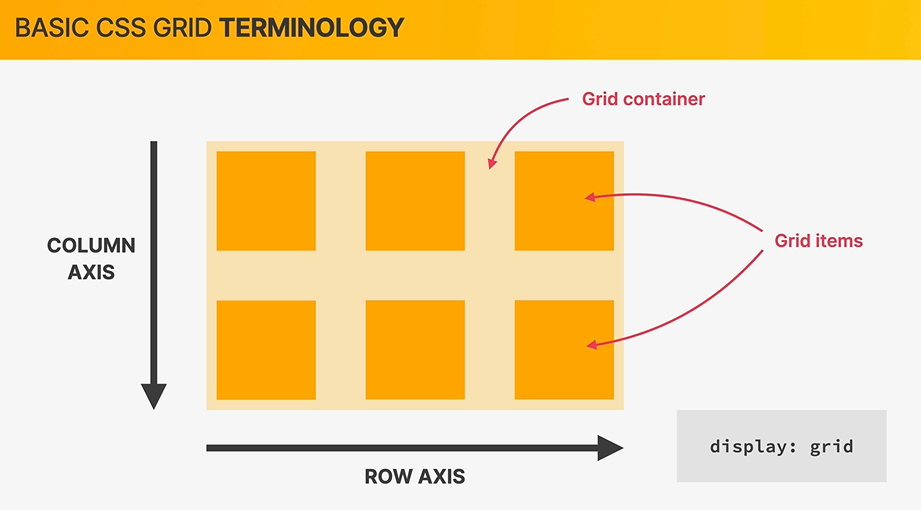
row-gap: 60px;

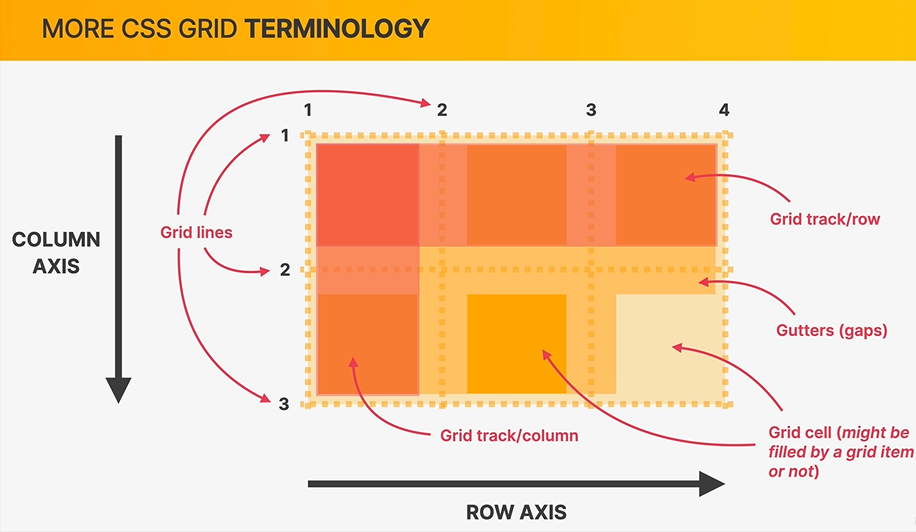
060 A CSS Grid Overview

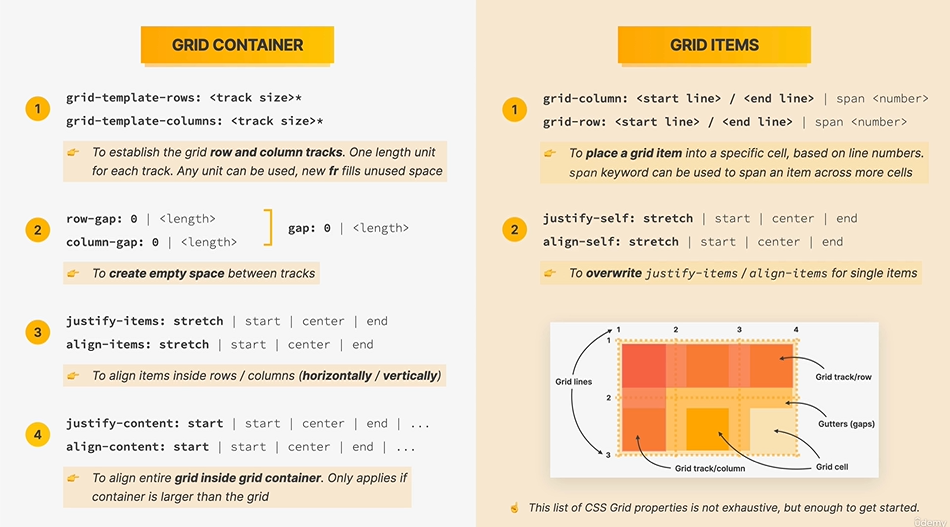
What actually is CSS Grid ?

CSS Grid is a set of CSS properties for building 2-dimensional layouts









061 Sizing Grid Columns and Rows

Flexible columns and units – for that we use the new *fr unit.*

Fr = fractional (or just fraction) – will try to fill the remaining space

1fr for 4 columns so wach of them will be divided equally

*Auto* for one of the columns: it will only take exactly the size that is necessary to fill its content. (many time in practice this is exactly what we need)

Explicit rows – cand sunt definite de noi

Implicit rows – cand trece pe un nou rand pentru ca nu mai are loc

We also have a way to style an implicit row. But we will see.

Usually its actually enough to define columns and let the rows be filled automatically.

062 Placing and Spanning Grid Items

How to place a certain grid item, into another grid cell

Properties that always need to be used on the grid items.

grid-column: 2/3;

grid-row: 1/2;

And the next ones are properties used on the grid container.

grid-template-columns: repeat(4, 1fr);

grid-template-rows: 1fr 1fr;

/\* gap: 30px; \*/

column-gap: 20px;

row-gap: 40px;

[…]

Grid-column: 1 / span 3;

= instead of specifying the final value of the  **grid line,**  we can simply say: ‘start at 1 and then span across three cells’.

Whenever we don’t know how many columns we have we can use the next trick:

grid-column: 1 / -1;

* It works because we have grid lines from 1 to 5 for example(left to right) but also the other way from -1 to -5 (right to left)

063 Aligning Grid Items and Tracks

Lets now lear how to align items

/\* CSS GRID \*/

display: grid;

grid-template-columns: 125px 200px 125px;

grid-template-rows: 250px 100px;

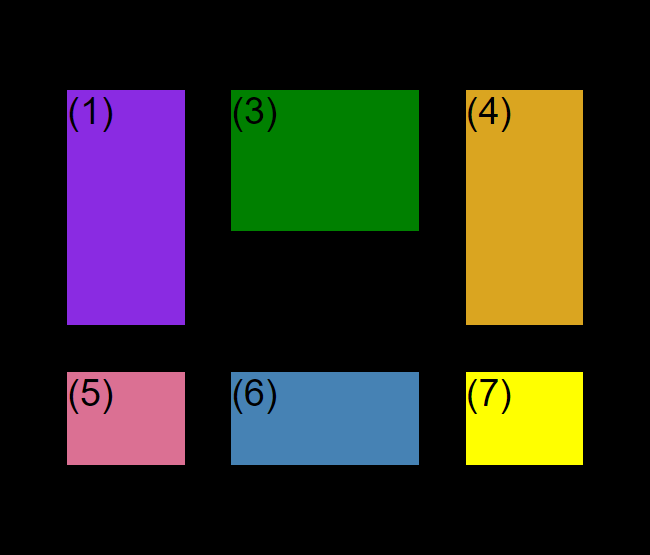
gap: 50px;

/\* Alinging tracks inside container: Distribute empty space \*/

justify-content: center;

align-content: center;

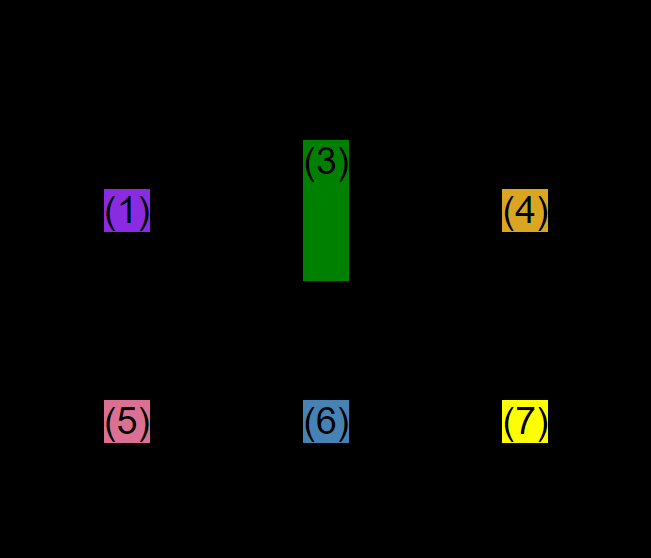
\*it only work because the container is bigger than the grid and we have empty space



/\* Alinging items INSIDE the cells: moving items around inside cells \*/

align-items: center;

justify-items: center;



\*3 is higher because we set inside the element a hight of 150 px

064 Building a Simple CSS Grid Layout

Blog spot layout

Css Grid works perfectly together with FlexBox. So we can keep FlexBox when is just one dimensional (one row)

065 CHALLENGE #3