

CSS (Cascading style sheet)

- **Linked** with HTML using 3 ways
 - External : <link> tag (We always use this)
 - Maybe you will have multiple external links inside html file , Note that : they override themselves sequentially
 - Internal : <style> tag inside <head>
 - Inline : within the tag using style attribute "The most powerful style"

بمعنى ان لودة موجود على عنصر معين و في الفايل الخارجي بيعمل حاجة مختلفة inline " اللي هيتنفذ هو ال
- **Comment** using ' /* */'
- **Syntax:** Selector {

Property: value;

 }
 - Selector can be:
 - **CSS Selectors**
 - * => ALL elements
 - Element => [p, div, h2]
 - Element OtherElement => div p => We will style any paragraph which is inside a div
 - .class-name => Style all elements have that class
 - #id-name => Style an element has that id
 - .parent .child => Targets the child class which is inside a parent class (the style will be applied to .child)
 - .class-one.class-two => That means we target The element which has 2 classes (class-one and class-two)
 - .class-name div, .class-name p => Grouping two selectors that have the same rules (all dives that have that class name and all paragraphs which have that class name)
 - Element.class-name => p.class-name (we will search for the class name which is inside the paragraph)
 - .parent > .child => Direct Child => we have 2 concepts (direct child and nesting child) here we target the direct child of the parent and any grandchild will be neglected
 - **Element + Other Element => [div + p] => next element , here we will style any paragraph that is next to div , ex: <div></div><p></p> that paragraph will be styled**
 - **Element ~ Other Elements => [p ~ div] => here we target siblings (الاشقاء) , which means all divs that siblings to a paragraph will be styled**
 - [Attribute] => we can style elements based on it's attribute
 - Element[Attribute] => we can style specific element based on it's attribute
 - [Attribute=Value] => we can style based on specific value of attribute
 - Element[Attribute=Value] => input[type="submit"] => marge all properties together
 - [Attribute~=Value] => Contains A Word
 - [Attribute*=Value] => Contains A Atring
 - [Attribute^=Value] => Start With A String
 - **Advanced selectors**
 - **:first-child** => Targets the first child of the parent (p: first-child => here there are 2 conditions to satisfy (1: the element is paragraph , 2: That paragraph is the first child of its parent))
 - **:last-child** => Targets the last child of the parent (p: last-child => here there are 2 conditions to satisfy (1: the element is paragraph , 2: That paragraph is the last child of its parent))
 - **:first-of-type** => This selector targets the element which is the first of its type reference to its parent (ex: p:first-of-type{this means we will style any paragraph take the first prize in being the first paragraph in his parent })

هنا انني بتقولي الاول من نوعه حتى لو () ماكنش اول ابن
 - **:last-of-type** => This selector targets the element which is the last of its type reference to its parent (ex: p:last-of-type{this means we will style any paragraph take the last prize in being the last paragraph in his parent })

هنا انني بتقولي الاخير من نوعه حتى لو () ماكنش اول ابن

- **:nth-child(n)** => by using this selector you can target the number of wanted element by using the same concept of (Child) , here we are counting from start to end (Forward-counting)
- **:nth-last-child(n)** => Same as nth-child but here the counting start from end to start (Backward counting)
- **:nth-of-type(n)** => Same explanation of nth-child but here we are using the concept (of-type)
- **:nth-last-of-type(n)** => Same explanation of nth-last-child but here we are using the concept (of-type)
- **not(Selectors)** => used to exclude any selector from specific styling
- **:only-child** => Here we target the unique elements (العناصر الوحيدة بمعنى اصح) (p:only-child {Here we will style any paragraph that exist "ALONE" in his parent})
- **:root** => Targets all element between <html> tag
- **:checked** => Targets checked attribute (as we said in form lesson)
- **:empty** => (mentioned before in pseudo classes)
- **:disabled** => Targets the element which is disabled (we mention the disabled attribute in forms)
- **:required** => Targets the element which is required (we mention the required attribute in forms)
- **:focus** => (mentioned before in pseudo classes)
- **::selection** => (mentioned before in pseudo elements)
- **::placeholder** => Targets the element which have a placeholder (we mention the placeholder attribute in forms)

○ The compiler apply the last style in CSS file (بمعنى ان لو عندك استايل بيقول لون الفقرة ب احمر و تحته امر بيقول لونها باخضر اللي هيتفد هو الاخضر)

➤ **Identifiers:** They are consider as special word that make some element/s unique 'such as Class & ID'

- In Css file, If you want to group some elements or classes or ids , we sperate between them by (,).
- Ex : .one , #two , div {
Color :red; }

➤ **Name conventions and rules :**

- Don't start the naming of any class or id with (Special character or number)
- In Classes, We name them based on kabab case (ex: user-test) 'https://developer.mozilla.org/en-US/docs/Glossary/Kebab_case'

➤ **What is the meaning of (Nesting):**

- Nesting is a way to reach the element base on sequence of other elements and classes
- Ex: you want to reach a paragraph that has a class name 'story-content' and this paragraph is inside a div , To reach it we use nesting
- Div .story-content{
Style here
}

➤ What is the usage of !important in CSS ?

- As you know there are some levels to apply the CSS rules (Priority), The highest level is inline styling. If you want to upgrade some style to an exist inline styling, Beside the property you write (!important) hence you have the upper hand

Properties and Values :

- **Background** : This consider as a shorthand (You can write background only as a property and the value you want, Based on that value the browser will understand which specific background you want)
 - **Background-color** => This property used to put a background color for an element, You can apply it in 3 ways
 1. Direct name of color , ex: background-color: red;
 2. Rgb (red, green, blue, alpha "opacity of the color"), ex: background-color: rgb(255,0,0,1);
 3. Hex decimal number #000000 to #ffffff, ex: background-color: #ff0000;
 - **Background-image** => This property is used to insert an image in form of background , It has an attribute URL(), inside it we add the

path of the image, now we are going to mention some properties that related mainly to image background

- **Background-repeat** => This property is related with Background-image property, where with Background-repeat we can control the repetition of a background image, it has multiple values:
 - **Repeat** -> **which is the default value**, The image is repeated through over the page
 - **No-repeat** -> the image will appear once
 - **Repeat-x** -> the repetition of the image will be in horizontal way only
 - **Repeat-y** -> the repetition of the image will be in vertical way only
- **Background-attachment**: This property which controls if the background will scroll with me if I scrolled in webpage or not **بمعنى** (هل الخلفية ستتزلز معي لما انزل ولا لا (الاساسي انها ماتتزلز), It has some values :
 - **Scroll** : **default value** (Where the image remains at the same position if I scrolled) **مش هتتزلز معايا يعني**
 - **Fixed** : Where the image will go downside with me (الخلفية هتتزلز معاك لو انت نزلت)
- **Background-position**: This property controls the position of the image within the page, it can we multiple values:
 - **X and Y directions** : we wrote it in the same simple form of (x,y), where X has (Left, center, right) values, and Y has (top, center, bottom) valued. Note that the default value of Y is center "You don't need to write it"
 - **Pixel**: The same concept of X, Y is applied but with numbers , first we write the position "Horizontally -> X axis " and then we write the position "vertically -> Y axis". Example : background-position: 100px 200px
 - **Percentage** : X, Y idea (ex: 50% 60%)
- **Background-size**: controls the size of the background-image, we have multiple values for this :
 - **Auto** : **default value**, No change in image's size
 - **Cover**: cover value covers all the content in which the image can be stretched OR cropped depend on the content and also the size of image
 - **Contain**: This value also TRY TO cover the content but the image hence be FULLY Visible however you stretch or decrease the size of the webpage it will stay fully visible
- **Padding**: This is a shorthand for (the upcoming 4 properties), You can write padding : 4 values for the 4 Directions 'Top, Right, Bottom, left' which is clockwise. In case you didn't write the 4 values each direction will take from the opposite direction (Top& bottom) (Right& left)
 - **Padding-top**: Add **internal** space on the top of the container or page
 - **Padding-right**: Add internal space on the right of the container or page
 - **Padding-bottom**: Add internal space on the bottom of the container or page
 - **Padding-left**: Add internal space on the left of the container or page
 - **Important Note** : Padding property can accept (px, percentage) and **Don't accept Negative value**
- **Margin**: Adding an **external** space to the container, margin is a shorthand also such as padding and all mentioned information related to padding are also applied here BUT there are some differences:
 - **Margin** : Can **accept Negative value** (ساعتها هينفذ الامر بالعكس يعني بدل ما يسبب مسافة هياخد مسافة)
 - **Margin: Auto** , This value works to make sure that your element on center (**Super important**), **Till now we now that it works horizontally only**
- **Border**: Is the property we use to control the border of any element, we can control (width, style, and color), border is a shorthand by which I can merge all 3 properties (Ex: **border: 10px solid red**). By using this way you controlled the whole style, (هنا انت مش بتقدر تخلي (مثلا كل جهة بلون او بشكل او بحجم هنا كل الجهات ليها نفس الحجم والشكل واللون # طب لو عايز تلعب ؟)
 - **Border-width** : Control the width of border and can take 4 value such as padding and margin and you can control one side by **border-top-width** and so on
 - **Border -style** : Control the style of border and can take 4 value such as padding and margin and you can control one side by **border-right-style** and so on
 - **Border- color**: Control the color of border and can take 4 value such as padding and margin and you can control one side by **border-bottom-color** and so on
- **Outline**: This property adding an outline to the element **which is NOT a part of that element** (ف عرض البوردر بيضاف للعرض الكلي للعنصر ودة) (على عكس البوردر اللي بيتعتبر جزء من العنصر مش بيحصل لما بتستخدم ال اوتلاين), Outline is also **Does not support the concept of 4 values** , The value will be applied to all sides of box , Shorthand (**outline: 10px solid red**)
 - **Outline-width**: add a width to outline (**accept one value ONLY**)
 - **Outline- style**: add a style to outline (accept one value ONLY)
 - **Outline-color**: add a color to outline (accept one value ONLY)

- **Display:** This property controls how the element will be displayed in webpage we have multiple values:
 - **Display: block** => The block elements have some characteristics :
 - Take the full width of the webpage if you don't select a width for it
 - Add a break line before and after them (يمتد سطر برضو)
 - Respect padding& margin[applied to all directions (top,right,bottom,left)], width, height
 - **Display: inline** => The inline elements have some characteristics:
 - Allow elements before and after them and take the exact width of the content + (مش يمتد سطر)
 - Do Not respect width and height
 - Respect padding and margin in specific direction only [right + left]
 - **Display: Inline-block** => It's a combination between inline and block , and also have some characteristics:
 - Allow elements before and after them and take the exact width of the content + (مش يمتد سطر) "Inline property"
 - Respect padding& margin[applied to all directions (top, right, bottom, left)], width, height "Block property"
 - **Display : none** => This value works on disappearance of the element where the next element will take its place in normal flow
- **Visibility:** This property controls the visibility of the element and its **default value is 'visible'** , but if you want to disappear it we put the value '**hidden**' , hence the element will disappear but its place will remain as it is. The usage of this property is mainly with animation to add some flexibility
- **Width :** This property controls the width of the element, Here we have value (**Fit-content**) that the width will exactly will fit the content of the element **specially whenever that element is a block element**
- **Min-width:** Determine the minimum width the element that will be applied
- **Max-width:** Determine the maximum width that the element will reach
- **Height:** Determine the height of the element (**default value is auto**) which fits the content of the element
- **Min-height:** Same as min-width
- **max-height:** Same as max-width
- **Overflow:** This property determine what will happen in overflowed content, This property has multiple values:
 - **Overflow: visible**, **which is the default value** where the content will normally appear
 - **Overflow: hidden**, hence the content will be hidden
 - **Overflow: Scroll**, add a scroll bar even if there is no overflow
 - **Overflow: auto**, exactly suit the content, add a scroll bar only if there is an overflow
- **Overflow-x :** This Property controls the content in horizontal axis (X- axis)
- **Overflow-y:** This Property controls the content in vertical axis (Y- axis)
- **Text:** Now we are going to talk about properties that is related to Text :
 - **Color :** Determine the color of text
 - **Text-shadow:** This property add some shadowing to the text, It takes 4 values as pixels
 - **Horizontal-Shadow (pixel)** : related to the motion of shadow left and right , it will be right if (Positive pixel) and left if (Negative pixel)
 - **Vertical-Shadow (pixel)** : related to the motion of shadow up and down , it will be down if (Positive pixel) and up if (Negative pixel)
 - **Blur (pixel)**: Determine the degree of blur
 - **Color (normal color)**: Determine the color
 - **Text-align:** Determine the POSITION of the text , It can have multiple values form those values (**Left, center, right**). **In that property it doesn't matter if the direction of the page (ltr or rtl)**
 - **Direction:** Determine the Direction of the text, It can have multiple values form those values (**ltr** 'Left to Right', **rtl** 'Right to Left')
 - **Vertical-align:** This property is used when there is an image inside text, the value of that property determine the position of the text related to that image, For example: **Vertical-align: top** (Text will be on upper level than image) and so on
 - **Text-decoration:** This property add some decoration to the text, it has multiple values such as (**underline, line-through, overline, none** (**which is used to remove the line in links**))

- **Text-transform**: This property change in the form of text, for example (**Capitalize** 'first letter capital', **Uppercase** 'all letters capital', **lowercase** 'all letters small')
 - **Letter-spacing**: Determine the space between letters in webpage (accept + and - pixel's value)
 - **Text-indent**: Determine the space that be left before the words will be written (accept + and - pixel's value)
 - **word-space** : Determine the space between words in webpage (accept + and - pixel's value)
 - **Line-height** : This property determine the height between lines in webpage (The optimal is 1.6 which is equal to 160%)
 - **White -space** : This property determine if the content will be wrapped if there is overflow happened or not , if **white-space: normal** then the wrap will happen , if **white-space: nowrap** it will remain on the same line
 - **Word-break**: This property is used when you need to control or match a specific width but you content is one word (بمعنى ان عندك عرض معين و عايز تناسب العرض دة و لكن المحتوى بتاعك هو كلمة واحدة و كمان اكبر من العرض ف لازم تستخدم الخاصية دي علشان تكسر الكلمة و لكن تظل كلمة واحدة و دي غالبا بتكون مع الينك) This property has multiple values (**Normal** 'no break' , **break-all**, **break-word**)
 - **Text-overflow** : This property is used to convert the overflowed text into 3 dots (...) with respect to padding (بمعنى ان ال 3 نقط دول padding , To do that you have to assign the value to **ellipsis**, **Important Note**: This property to work the **overflow** property must set to **hidden** . Later we will learn how to made that content visible when user hover, we can do it using:
 - Use **title** attribute in the element and inside it put the whole content
 - Use events, for example (div: hover {**overflow**: **visible** })
 - **Inherit Value** :
 - The concept of inheritance exist in some properties and others not, If you want the element to inherit some property, you have to write the **parent and the child** then { the **property** and give it **inherit** as a value }
 - **Font- Family** : This property determine the shape of the font based of chosen family, as a value we write **font option 1** then **font option 2** , etc. and then the **family** (sans-serif & serif), We put 2 options just in case the browser didn't recognize one of them
 - Much better way : Use google fonts (<https://fonts.google.com/>) add the wanted font in head of html page
 - We have something called (**@font-face**) this is used to add some font that specify for your project the user don't need to download the font, It's atomically downloaded from the server that host your website
 - How to use it ?
- ```
@font-face {
 font-family: 'Outfit';
 src: url('fonts/Outfit-VariableFont_wght.ttf') format('truetype');
 font-weight: 100 900;
 font-style: normal;
}
body {
 font-family: 'Outfit', sans-serif;
}
```
- **Font-size**: This property determine the size of the font, It can take multiple values from **CSS units** so let's talk about them
    - **Px** : Pixel is a dynamic unite that take specific partition from screen
    - **Em** : em indicates 'Time', this value inherit the size from parent base on how many times you want, **ex(div {font-size: 20px}/ div span {font-size: 2em "Which is equal to 40px" })**
    - **Rem**: rem indicates "Root time". Who is the root ? ==> **<HTML> tag is the root and the default size of the webpage is 16px**, which means the element will inherit from the size of html base on the number beside rem (2rem == 16\*2 = 32px )
    - **Percentage**: Based on that percentage it will inherit from its parent (div {font-size: 50px} div span {font-size: 50% == 25px})
    - **Vw**: **view port width**, what is view port ? ==> Is the whole white screen that exist 'This concept is mainly used in responsive designs' **1 vw means 1% from the current view port**, ex (current view port = 900 px , div {font-size: 5vw == 5\*9 = 45 px})
      - Note that: same with vmin (Indicates minimum view port) , vh (view port height)
  - **Font-style**: Add some style into the font such as (**italic and oblique**) Not used but in case you need them (The default situation **font-style: normal** )

- **Font-variant:** Actually, we know one value only 'Small-caps' this converts all letter to capital letters , wait what is the difference between it and between text-transform: uppercase ?
  - **Font-variant:** 'small-caps' has smaller font size that text-transform: uppercase
- **Font-weight:** Determine the weight of the font (سمك الخط نفسه) , this property has values from (100 to 900 and bold, bolder ) we will use it and to reset it we use (font-weight: normal)
- **Cursor:** This property determine the shape of the cursor, it has several values (pointer, grap, move, etc.)
- **Float:** This property is used to structure the layout of the webpage ( بتخلي العناصر عايمة كدة في الصفحة بناا على طلبك انت ) this property has sepecific ruels to work, First let's talk values (left, right and other values)
- **Clear:** This property remove the effect of float property, its values (left, right, both 'we use both')

- Let's see an **example** of float and clear:

```
<div class="parent">
 <div>1</div>
 <div>2</div>
 <div>3</div>
 <div>4</div>
 <div>5</div>
 <div>6</div>
 <p class="clear"></p>
</div>
<p> Rana </p>
```

```
.parent{
 background-color: aqua;
}
.parent div{
 float: left;
 width: 18%;
}
.clear{
 clear: both;
}
```

- **Calc ():** **This Is NOT a property**, It is a method used in properties who needs complex numbers as a value, such as width , ex: width: calc(100% -90px )
  - width: calc((100% - 120px)/6);
  - margin-left: 20px;
- **Opacity:** This property controls the transparency of all components of that element (such as color, text, image, etc.). Note that the alpha in color controls only the appearance of the color and nothing else
- **Position:** One of the most important properties, It controls the position of several elements , let's talk about its values :
  - **Static:** **The default value**, The element with that value remains in the normal workflow and not affected by (Top, bottom, right, left) properties which we used to control the exact position of the element
  - **Relative :** This value makes the element move based on **ITSELF** and normally affected by (Top, bottom, right, left) properties
  - **Absolute:** This value makes the element move based on ITS **PARENT** (the parent may be the webpage itself or another element **(Super important note : If you will made it follow another element, That element must take position : relative )**) and also affected by (Top, bottom, right, left) properties
  - **Fixed:** This value makes the element to stay in specific place based on the main webpage even if you scroll it will be fixed to the same place with you , and also affected by (Top, bottom, right, left) properties
  - **Sticky:** This value makes the element fixed But whenever you reach it at the first time using Scroll and also affected by (Top, bottom, right, left) properties
    - Recourse "<https://developer.mozilla.org/en-US/docs/Web/CSS/position>"
- **Z-index :** This property is responsible for layering, Which element will be above or below which element. Without z-index, the first written element in html will be at the bottom of layers. When you use z-index, the element who has the biggest number of z-index will be at the top of page.
  - **What if there is 2 element have the same z-index value ? =>** Then we return to the normal structured sequence in HTML file.
  - **Note that :** This property works ONLY then you used **position** property also
- **List-Styling**

- **List-style-type** : This property is used to style the pullets of the list (We have thousands of values ) just know the property
- **List-style-position** : This property determine if the pullets will be **inside/ outside** the margin an padding that will we applied into the list
- **List-style-img**: Instead of pullets you put an image and put the **URL** of it as value
- **List-style** : Short-hand, waiting from you 3 values (style of pullets , Position of pullets (inside/outside) and finally an URL of the image )

□ **Border-radius**: This property is used to add some curved borders to the elements , This property accept 4 values normally like others

□ **Border-top-left-radius**: This property targets the top-left edge of the border, This property can accept 2 values

□ **Box-shadowing**: This property is used to add one or more shadows to the element, This property takes 6 values

1. **Horizontal-shadow** (number must have an unit)
2. **vertical-shadow** (number must have an unit)
3. **Blur** (number must have an unit)
4. **Spread** (number must have an unit)
5. **Color**
6. **Inset or outset**(Outset is the default value)

**Note that you can add multiple shadows to element like that example**

```
box-shadow: 10px 10px 0.1px 0px #deae21 inset, 5px 0px 0.2px 0px #5f82ff ;
```

□ **Box-sizing**: In Normal cases, the padding and borders are added to the width and height of the element if you select specific width or height for that element or they are added to the parent element, That scenario happens when the (**box-sizing: Content-box ;**). To Change that scenario (The padding and border will be out the calculations of width and height ) set (**box-sizing: border-box;**)

□ **Transition-duration**: Any element you decide to change its behavior based on some action, For smooth transition we used this property, This determine the time of the **ACTION** ( الوقت اللي هياخده العنصر و هو بيتحول من حالة لحالة تانية )

□ **Transition- delay**: This is the **WAITING** time ( اللي وقت الى العنصر بينتظره عقبال ما ينفذ التغيرات اللي طرأت عليه )

□ **Transition- Property** : Determine which exactly properties that we will apply it's transition ( بمعنى حدد بالظبط اية الخواص اللي هنتغير لما حدث ) (معين يحصل علشان ماتقفلش صفحة على زميلك)

□ **Transition-timing-function**: This property which determine how the transition will happen (slow, fast, slow) "**ease**", (slow, normal) "**ease-in**", (normal, slow) "**ease-out**", (the speed is constant all over the transition) "

□ " and so on

□ **Transition**: This is a short hand, You can put all 4 values (Note that the first number is for the duration ) and even you can add for more than property using

```
transition:1s width,padding-left ease-in , 2s margin ease;
```

□ **Flex**: This concept is super important, It helps you to manage and style your layout in the optimal and easiest way ever.

1. **What is the difference between float and flex concepts ?** => with using (Float property) You have to rearrange everything manually (There is no dynamics) But while using (**display: flex**) everything is dynamically calculated
2. How to use flexbox and what are the properties that related to them?

◇ To use flexbox you have first to add all elements you want to layout into a one div(Parent):

□ **In that parent we write,**

◆ **Display :flex** => which is responsible to enable the flexbox mood, **Super important point=> If you want another element to be next to the flex-container, then we write (display: inline-flex)** Then you write whatever the properties you want:

◇ **Flex-direction**: This property is used to determine whether you want to align the elements in (**row "default value"**, **row-reverse**, **column**, **column-reverse**). **(1 Dimension )**

**Here there is super important note** : The flexbox concept is following the direction of the page (which is dynamic concept), where float is only follow the value of float property (**float: right/ left**)

◇ **Flex-wrap**: This property determines whether the elements will be wrapped or not ( تنزل تحت يعني لو العرض خلس ) The default value in case flexmood (**flex-wrap: nowrap**) which means flex mood always works to fit the contant to

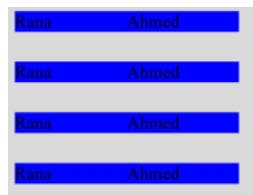
the available width in the webpage without need to start a new line, to change that we set (**flex-wrap: wrap**) and also there is a value called (**wrap-reverse**)

**Note that:** This property by logic will work incase you chose the direction: row / row-reverse

- ◇ **Flex-flow:** This property is a short-hand for both (direction and wrap) so we put the value of the direction first then the wrap value
- ◇ **Justify-content:** (Horizontally when flex-direction is row, Here we arrange the items Horizontally (based to x axis) so here we deal with (width) ) This property main controls the arrangement of the element through the flex, We have several values
  - ▶ **Justify-content: flex-start** => This is the default value where the elements normally start from the direction of the webpage ( في حالة ان لغة الموقع انجليزي هيكونوا من الشمال لليمين عادي جدا )
  - ▶ **Justify-content: flex-end** => This value where the elements have the same sequence BUT they are located at the end of the page ( في حالة ان لغة الموقع انجليزي هيكونوا بنفس ترتيبهم عادي بس على ناحية اليمين )
  - ▶ **Justify-content: center** => Centered the elements to the container
  - ▶ **Justify-content: space-between** => if you want to add a space between elements (this space will be calculated dynamically ( بياخد الباقي من العرض و يقسمه على انه يكون مسافة متساوية بين العناصر )
  - ▶ **Justify-content: space-around** => if you want to add a space before and after each elements (this space will be calculated dynamically ( بياخد الباقي من العرض و يقسمه على انه يكون مسافة متساوية بين العناصر (قبل و بعد كل عنصر) )
  - ▶ **Justify-content: space-evenly** => if you want to add a space between elements && a space before and after each elements (this space will be calculated dynamically (combination between space- between and space-around )

- ◇ **Aline-items :**(Vertically, when flex-direction is row Here we deal with (Height)) This property is used to arrange the items which is inside the flex's parent based on (y-axis) Note that -> This property controls the Items inside the container, this works on items level , We have several values:

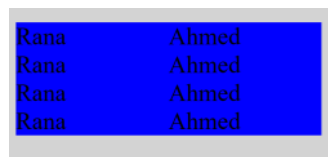
- ▶ **Aline-items: Stretch** => This is the default value where the elements will be stretched to fit the whole height of the page
- ▶ **Aline-items: flex-start** => here the elements normally start from the top of the page (The start of the webpage (vertically ))



- ▶ **Aline-items: flex-end** => This value where the elements have the same sequence BUT they are located at the end of the height of the page
- ▶ **Aline-items: center** => Centered the elements to the container based on y-axis (height)

- ◇ **Aline-content:**(Vertically, Here we deal with (Height)) This property is used to arrange the **WHOLE CONTENT INSIDE THE CONTAINER AS ONE BLOCK** which is inside the flex's parent based on (y-axis) Note that -> This property controls the **WHOLE BLOCK** which is inside the container, this works on **WHOEL CONTENT level** , We have several values:

- ▶ **Aline-content: Stretch** => This is the default value where the elements will be stretched to fit the whole height of the page
- ▶ **Aline-content: flex-start** => here the all block of elements normally start from the top of the page (The start of the webpage (vertically ))



- ▶ **Aline-content: flex-end** => This value where all block of elements have the same sequence BUT they are located at the end of the height of the page
- ▶ **Aline-content: center** => Centered the all block of elements to the container based on y-axis (height)
- ▶ **Aline-content: space-between** => if you want to add a space between elements(Vertically) (this space will be calculated dynamically ( بياخد الباقي من الارتفاع و يقسمه على انه يكون مسافة متساوية بين العناصر )
- ▶ **Aline-content: space-around** => if you want to add a space before and after each elements (Vertically) (this

space will be calculated dynamically (قبل و بعد كل العناصر) (عنصر)

- ▶ **Aline-content:** `space - evenly` => if you want to add a space between elements && a space before and after each elements (this space will be calculated dynamically (combination between space- between and space-around )

◆ **Notice that when the flex direction is a column, justify-content changes to the vertical and align-items to the horizontal.**

□ **In that Child we write,**

There are many properties we can write In the child related to flex-mood:

- ◆ **Flex-grow:** **The default value is (0)**, when using the default value the width will normally distributed, But when you write a number this number will be like a multiplication of increment (أو زيادة) the given width but for that element exactly (Whenever there is available width) (بكل بساطة انتي بتدي اهمية لعنصر انه يكون اكبر من باقي العناصر طول مافي عرض بيسمح بده اما لو ( Whenever there is available width) مافيش تعتمد على برضو اللي باقي من المساحة دي قد اية يعني هو هيفضل برضو اكبر من الباقي و لكن في نقطة معينة لما العرض يقل العناصر كلها هتكون (نفس الحجم)
- ◆ **Flex-shrink:** **The default value is (1)**, when using the default value the width will normally distributed, But when you write a number this number will be like a multiplication of deduction (أو نقصان) the given width but for that element exactly (Whenever there is available width) (بكل بساطة انتي بتقولي لو حصل و العرض قل لاي سبب من الاسباب العنصر دة هو اللي هيقل بنسبة اكبر ( Whenever there is available width) (لحد برضو نسبة معينة من النقصان ساعتها كل العناصر هيكون ليها نفس الحجم)
- ◆ **Flex-basis:** **The default value is (auto)**, This property is super important (and works **ONLY with flex child**), This property controls how much (width or height will be saved for that element)!! yes, it controls both the width and height based on the (flex-direction property)  
**Important notes** about (Flex-basis): Note that=> **This property CAN NOT overcome the max-width property**
- ◆ **Flex:** This property is a short-hand to combine 3 properties in one (flex-grow , flex-shrink, flex-basis) with the same sequence (**0 1 auto**), **Note that : Both properties (flex-grow and flex-shrink) have no UNIT but flex-basis has**
- ◆ **Order:** by using this property, you can control the order of the element BUT note that, **in case you want to change the order you must re-set the order for the rest of element** (لما تغير مكان عنصر و تحطه مكان عنصر ثاني لازم تحدد بقي العنصر الثاني دة هيروح (فمين و خلي بالك ان تاثير الترتيب مش هيبان غير لما ترتبي باقي العناصر الثانية كلها Note that: You can add (negative value for that property)
- ◆ **Align-self:** This property is used to control one element in the flex-container (بمعنى انك تقدر تتحكم في عنصر لوحده و تديله مثلا ) , This property has multiple values such as (**flex-start, flex-end, center**, etc.)

- **Filter:** This property is used to add some effects into images so we write the property and give it one of the values (**grayscale**(Percentage here), **invert**(Percentage here), **blur**(write a unite here )), We use this property to add some style to images.

- **Gradient:** This concept is used to add style in colors and mix them up, To use this property we write (**background-image: linear-gradient**(here we mainly write 3 things ++))
1. The direction or angle (to right, to left, to bottom, to top) or the equivalent to those in angles (0deg, 90deg, 180 and 270)
  2. The first color stop + the percentage of that color in the container and if you want to made it sharp you have to write the start and the end (In this way, you prevent the mixing of colors)
  3. The second color and so on

Example =>

```
background-image: linear-gradient(to right, ■rgb(137, 13, 87, 0.687)30%, ■rgb(255,0,0,1))
```

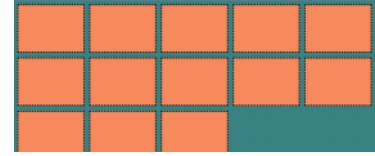
- **Caret-color:** This property is used to change the color of the cursor in writeable places , example

```
input{
 caret-color: ■rgb(255, 247, 29);
}
```

- **Pointer-event:** This property is used to change in the behavior of clickable elements (ex: a { **pointer-events: none**} // Then the link will be none-clickable )

- **Grid system** => This Concept is one of the most powerful tools in CSS , Frist let's talk about grid system and note that now we will

understand the concept only, This image describes the concept of grid system , It consists of columns and Row (2D layout system)



### ○ As usual, Let's talk about what we write in parent

- **Display:** `grid` , or `inline-grid`. Exactly as flex mood, by writing those we enable the grid system. Using **display:** `inline-grid` in case you want to add some element beside the grid **(2 Dimension )** . Now , let's talk about other properties:

- ◆ **grid-template-columns:** This property is used to determine the number of columns and decide the

**width(Horizontally)** of each column, This property has multiple values :

- **Px** : Normal pixel unit , Note that we write the units based on the number of column we want, ex: `grid-template-columns: 100px 100px 100px` (This means we have 3 columns each is 100px width)
- **%**: It also available to write the wanted width as a percentage of the whole width (25%(means 1/4 from the whole width) 50% 25%)
- **Auto** : In case there in NO property controls the layout horizontally, Them the element will be automatic distributed through the width, BUT if there is anything that controls the horizontal layout then the auto became shy (بتأخذ اللي على ) قد المحتوى بتاعها و بس و خلي بالك انتي عندك حالتين بتكون فيهم شاي اول حالة هي تدخل من خاصية تانية هتتحكم في الافقي و تاني حاجة هي تتدخل الفراكنش
- **Fr**: This indicates fraction, it sees what is the available width and take them all (It's a greedy value)
- **Minmax()**: **This method is used to control the minimum and maximum acceptable width , this also can be inserted in repeat()**
- **Repeat()**: we used this value in case we have multiple columns have the same width, First we write the number of columns then the width of each. Ex, `Repeat(3,1fr)`.
  - ◇ **OR we can let the screen's width decide that Repeat(auto-fill, "the wanted width"), In that way the browser will divide the current viewport width into columns their width equal to wanted width**

**Note that: We can mix between all units, it's okay**

- ◆ **grid-template-rows:** This property is used to determine the **height(Vertically)** of each row, This property has multiple values exactly same as `grid-template-column`
- ◆ **Row-gap:** Indicates the space between the rows
- ◆ **column-gap:** Indicates the space between the columns
- ◆ **Gap:** this is a short-hand of row-gap and column gap with the same sequence
- ◆ **Justify-content:** As you know, This property targets the **horizontal axis which is the width (so this targets the grid-columns)**, Whenever this property appears the auto value of columns being shy (بتأخذ عرض على قد المحتوى بالظبط), It's values same as mentioned in flex-mood
- ◆ **Align-content:** As you know, This property targets the **vertical axis which is the height (so this targets the grid-rows)**, Whenever this property appears the auto value of rows being shy (بتأخذ عرض على قد المحتوى بالظبط), It's values same as mentioned in flex-mood
- ◆ **grid-template-areas:** This property controls which element will take which row and how many columns, This is super important in dividing the layout, to understand it focus on example,  
**Important note: In case you want to leave one column empty you have to put " " instead of writing the grid-area value**
- ◆ **Grid-area: (child property)** The value of that property is equal to the value which is written in `grid-template-areas`  
Example :

```

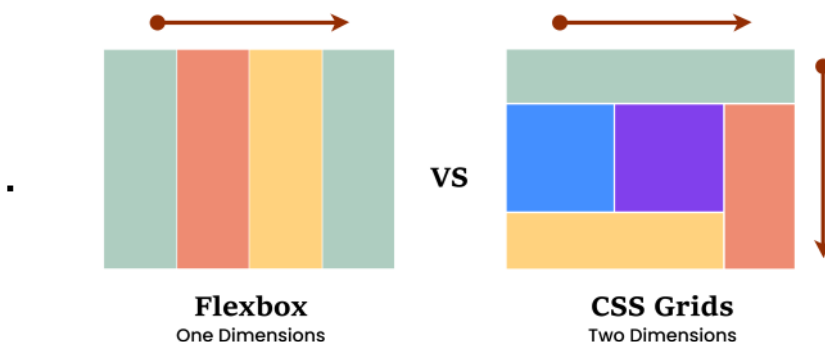
.parent{
 height: 800px;
 display: grid;
 grid-template-columns: repeat(5,auto);
 grid-template-rows: auto auto auto;
 /* "logo logo nav nav nav" => Means this is the first row with logo
 takes 2 columns and nav takes 3 */
 grid-template-areas: "logo logo nav nav nav"
 "side side cont cont cont"
 "foot foot foot foot foot";
}
h2{
 grid-area: logo;
 background-color: brown;
}

```

## ○ Now, Let's talk about what we write in Childs

- **Grid-row:** This property is used to determine how many rows the element will take, so this accept (row start and row end) example: **grid-row: 2/5** this means the element will take 3 rows (the end index is excluded as usual). You can do the same function using span, example : **grid-row : span 3**, Note that based on what write you may increase the number of rows and it's okay here (**In rows**)
- **Grid-column:** This property is used to determine how many columns the element will take, so this accept (column start and column end) example: **grid-column: 2/5** this means the element will take 3 columns (the end index is excluded as usual), **using the normal start/ end may case an increment in the number of actual column which may destroy the layout, then You can do the same function using span**, example : **grid-column : span 3**, hence no new column will be added , may be a new row added and its okay
- **Grid-area :** We talk about this property in **grid-template-areas** property while dividing the grid into areas, now we can do the same thing but using (**row-start, column start, row-end, column-end**) with that sequence, by using that way to told how much space that element will take from grid

○ **Finally Note that :**



□ Now we will talk about transform properties (Computer Graphics If you remember) **Remember to use prefixes because those methods not supported in all browsers :**

- 2D transform (x-axis, y-axis)
  - **Transform : ScaleX()** -> This method determine the scale of the increasing (incase + value), decreasing (value less than 1 ), or flapping (incase - value) and all of that based on X-axis , **The default value of scaleX is 1**
  - **Transform : ScaleY()** -> This method determines the scale of the increasing (incase + value), decreasing (value less than 1 ), or flapping (incase - value) and all of that based on Y-axis, **The default value of scaleY is 1**

- **Transform : Scale()** -> This is a short hand of both (**scalex** and **scaley**), you write the two values and in case you wrote one only the second value will be equal to the first (ببساطة القيمة الثانية من القيمة الاولى)

- **Important note => What is the difference between width, height and scalex and scaley ?**

In width, height , when increase or decreasing the width/ height the container is the only thing be stretched or shrink But the text inside, the margin, padding all of those things remain constant. In scalex and scaley the WHOLE container with which is inside (text, margin, padding) all of them will be stretched or shrink also

- **Transform : Rotate()** => This method is used to rotate elements around the Z-axis (The third dimension) so it's equal to **RotateZ()**, This property accept the angle of rotation in form of(deg, rad, grad, and turn) we will use deg or turn

**Important note =>**

- In case the angle value is '+' Then the rotation became (clockwise) and in case '-' value the rotation became (anti clockwise)
- Whenever you want to apply more than one transform on the element, you write them sequentially in transform property example: Transform: scale(1.3) Rotate(90deg) => then both functions will be applied into the element

- **Transform: translate()** -> This method is used to move an element from some position to other position, This method accept 2 values (Move in x-axis, move in y-axis) Base on those values the element will move

- **Note that:**

- ◆ Translate accepts (px, em, rem, %) all those units are accepted.
- ◆ + Value for x-axis means it will move to right, and - value means left .
- ◆ + Value for y-axis means it will move to bottom, and - value means top.
- ◆ In case you wrote one value only in translate them the other value became (Zero)
- ◆ **The translate method is the best for performance in animation world and much more simpler to use translate than using position and top/right/bottom/left properties**

- **Transform: skew()** -> This method is used to add (انحراف) for the element, mainly this method is used to add good appearance to the webpage, It accepts 2 value (x-axis deg/rad/grad/turn, y-axis deg/rad/grad/turn). The + value الانحراف سيكون في اتجاه معين and the - value الانحراف سيكون في الاتجاه المعاكس

- **Transform : matrix** -> we can say this is a short-hand of 6 methods , "You write only 6 numbers without writing any method's name". You have to write the numbers base of this sequence (**scaleX(), skewY(), skewX(), scaleY(), translateX(), translateY()**), as we said we write a number only , example :

**transform: matrix(1.2, 0.2679, 0, 1.2, 20, 20);** // 15deg is equal to 0.2679

The above line is exactly same as:

**transform: translateX(20px) translateY(20px) scaleX(1.2) skewY(15deg) skewX(0deg) scaleY(1.2);** // Here we put the translate at first to give us the exactly behavior

- **Transform-origin: This is a property not a method**, This property is used to determine the start or the origin point at which the element will start It's transformation (methods we talk about), **The default value of the property is transform-origin: (50%,50%)** for sure we are talking about x and y values.

- **Note that :**

- ◆ The property can accept as a value:
  - 1) CSS units => px, em, rem, etc.
  - 2) %
  - 3) Keywords-> x-axis place (left ==0%) (center == 50%) (right == 100%) , y-axis place (top ==0%) (center == 50%) (bottom == 100%) , **ALWAYS REMEMBER -> You screen starts from the top left corner (x increases by going to right and y increases by going to bottom)**

- As we said, we use the transform-origin to setup the place where we will start transformation

- Now we will talk about **3D transform** properties **Remember to use prefixes because those methods not supported in all browsers :**

➤ **Add To all methods the third dimension (Z-axis)**, The direction of z-axis is the direction from the screen of your laptop to you (This is a super important note)

➤ **Rotation options**

- **RotateX** (the rotation will be around yz plane) , **RotateY** (the rotation will be around xz plane), **RotateZ** == to rotate() (the rotation will be around xy plane),
- **Rotate3d()**: This method accept 4 values (x, y, z, degree) , We put 1 in the axis where we want the rotation to be around, Example: **Rotate3d(0,0,1,45deg) == rotatez(45) == rotate(45deg)**
- **Apply the same concept on the other methods**

➤ **Translate options :**

- This method as we know used to move some element to another place, okay now we have translate3d (x-value, y-value, **z-value**) Nice talk., How we get the value of z?!
- **Perspective: Parent property**, The value of this property determine the distance between you and the screen (The whole z-axis available distance for you), Then you can decide the translate z-axis value based on available number
- **Perspective origin: Parent property**, This property accept from you the **same values of transform-origin**,  
هي اللي بتحدد المكان اللي بتبص على العنصر منه ?  
What is the purpose of it ?
- **Backface-visibility**: This property (**child property**) is mainly used whenever you flapped some element and you want when it flapped be disappeared يختفي بمعني اننا عايزين لما عنصر مايتقلب ساعتها بدل ما يفصل ظاهر و هو مقلب  
The default value of that property is **visible**, then we convert it to **hidden** to remove element from screen
- **transform- style**: This is a (**parent property**), This property used to save the position of an element in 3d sapce, The default value is **falt** which is not saving a place, so we use **preserve-3d** value to keep the place even if it disappears
- **Transform-origin**: This property is used here too Normally such is 2d

## □ Animations : USE prefixes to avoid crashing :

- First we have to decide which animation we want to do, and we do that by :  
**@keyframes** (name of the animation){  
word 'from' / percentage{  
here you write what the animation you want IF there aren't apart from the style ( بمعني ان لو اللي هتكتبه هنا هو اصلا اللي في الستايل العادي ممكن )  
(نعمل سكب للفروم خالص)  
}  
word 'To' / percentage{  
here you write what the animation you want  
}  
}  
}
- Then in the element you want to animate we start writing the animation properties:
  - **Animation-name**: as a value of this property we write the name of animation (the same written name in **@keyframes**)
  - **Animation-duration**: This property determines the duration of how long the animation will continue
  - **Animation-timing-function**: Is the same at transition property, it determines the way of growth the animation will have (**ease**, **ease-in**, **ease-out**, **linear**)
  - **Animation-delay**: This property indicates the time of waiting (**incase the given value is +**) or the time of skipping from the main time of animation and start over after that CUTTED time (**incase the given value is -**) ( بمعني ان لو الوقت بالسالب ده معناه ان هو هيقطع )  
(الوقت ده من المدة بتاعت الانيميشن اصلا و بعد كدة يعرضلك الانيميشن من بعد الوقت المقطوع ده)
  - **Animation-iteration-count**: This property determines the number of REPEATION of an animation
  - **Animation-direction**: This property controls the direction of animation's execution, it has multiple values (**Normal** : normal execution 'from then to', **reverse** 'to then from', **alternative** 'forward execution then backward', **alternative-reverse** 'backward execution then forward')
  - **Animation-fill-mode**: This property used in case you want the animation to stay at his last state (Don't return to the basic state) Then it depends on animation-direction property, so set its value to **both** to remove conflict
  - **Animation-play-state**: The default value of this property is **running** (the animation will run over and over "based on number of iterations") if you want to pause it give it value **paused**
  - **Animation**: This is the short-hand is waiting from you "**name duration timing-function delay iteration-count direction fill-mode**"

## Pseudos'

- **Classes**: The meaning of pseudo class is (They are not real classes, They are actually events or actions). Based on some action you decide to change the behavior of the element. We write it using ": ", Let's see some examples
  - **:hover** => This pseudo class is used when you hover some element (ex, a:hover{ color: red})
  - **:Visited** => Mainly used for links to determine which link is visited and which one is not (a: visited {/\*write your style here\*/})
  - **:Checked** => Mainly used in forms to change in style of the checked element ==> **(HTML => <input class = 'ch' type="checkbox"> <label> </label>, CSS=> .ch: checked + label {color : red } ) // This code means when the checkbox with class (ch) Is checked then convert the color of label that next to input into red color**
  - **:empty** => This class targets all empty elements
  - **:Focus** => Is applied whenever the element is on focus
  - **Important Note** => You can change those classes from google chrome by click the element and chose Force state
- **Elements**: This is same as classes but here to write pseudo elements we write the actual element then (:: the pseudo element)

- **::First-letter** => This pseudo element targets the first letter of any actual element and add whatever style
  - **::First-line** => This pseudo element targets the first line of any actual element and add whatever style
  - **::selection** => Targets when you select some words then you can write the wanted style
  - **::After** => You can add any content or shape after the actual element , Some notes => **you have to add content property to that pseudo element to work** , By logic to controls that element you must give to its parent (**position : relative**) and to the (element::**after** {**position: absolute**})
  - **::Before** => You can add any content or shape before the actual element , Some notes => **you have to add content property to that pseudo element to work** , By logic to controls that element you must give to its parent (**position : relative**) and to the (element::**before** {**position: absolute**})
- We said, You have to add **content** property to the pseudo element to work . Now let's discover what the values of that property:
1. **Normal text between the (" ")** : This text will be added normal to the content and appears in HTML page
  2. **Counter()** : This method is used whenever you want to count some elements, you have to insert (**counter-increment**) property in the main element and give it any value , then write these value in **counter( )**

```

p{
 counter-increment: rana;
}
i) p::before{
 content: counter(rana);
 color: magenta;
}

```

3. **Attr()**: By using this method you can catch whatever the element you want and add content and style that content also

```

<p data="This data is inserted into the page ">
i) Lorem ipsum dolor sit amet consectetur adipisicing elit. Ex sit praesentium, recusandae molestiae quae sint adipisci,
</p>

```

```

ii) p::before{
 content: attr(data);
 color: magenta;
}

```

- iii) **This data is inserted into the page** Lorem ipsum dolor sit amet consectetur adipisicing elit.  
Ex sit praesentium, recusandae molestiae quae sint adipisci,

4. **URL()** : This method is to insert multimedia to the element but we do not use it.

## Vendors Prefix

What is the meaning of vendors prefix ?

- [1] New CSS Feature Appeared
- [2] Browsers Tests The Feature In Version X
- [3] Browsers Add Prefix To Give Developers The Ability To Use It
- [4] The Feature Is Fully Supported In Version Y, No Need For Prefix

So we add some prefix to the properties that maybe new to make sure that they will be supported, Those prefixes are:

- **-webkit-** => **Chrome, Safari, New Opera Version**
- **-moz-** => **Firefox**
- **-ms-** => **ie, Edge**
- **-o-** => **Old Version of Opera**

Note that :

- In visual studio code we have extension to add this "css-auto-prefix" , So don't bother yourself

## The Margin Collapse

- [1] Margin Collapse Happens Only with Vertical
- [2] Bigger Margin Wins
- [3] Margin Collapsing With Elements Without Anything Between الدمج يحصل في حالة ان مافيش عنصر في النص بين العنصرين التانيين

## CSS variables

- As you know we have 2 types of variables (global and local)

1. **Global CSS variable** -> hence you wrote the Variable in root element (which is html tag ) but we write it such as that way (:root{ }), then you have to write the variable (-- variable\_name : value) and for sure to use it we pass the variable name in the matched property, Note that we use var(variable name, fallback 'which is used whenever something wrong happened with the variable "Backup" ')

Example:

a)

```
:root{
 --mainColor: Red;
}

p{
 color: var(--mainColor, blue);
}
```

2. **Local Variable** => This variable is created and used locally within the element, as you know if the variable within the scope the value will be applied, if not then we search in outer scope (Global scope)

a.

```
:root{
 --mainColor: purple;
}

p{
 --mainColor: Red;
 color: var(--mainColor, blue);
}
```

The Red Color will be normally applied

```
:root{
 --mainColor: purple;
}

p{
 /* --mainColor:Red; */
 color: var(--mainColor, blue);
}
```

The purple color will be applied

```
:root{
 /* --mainColor:purple; */
}

p{
 /* --mainColor:Red; */
 color: var(--mainColor, blue);
}
```

The Blue color will be applied

## CSS Global values

- Each Property we talked about has global values , those values are :

1. **inherit** : as we said to tell the element that he will inherit this property from his parent , ex: color: inherit
  2. **initial** : Each property has an initial value , for example the initial value of display is inline , so if I wrote display : initial == display: inline
  3. **unset** => This value may case a confusion for you
    - If property is Inherited => then in case you set the property value to unset the property will inherit the value from the parent
    - If Not => then the property value will be = to the initial value
  4. **revert** CSS Level [4] => This value give the property the value the by default given by the agent user
- All** : If you want to group all properties together and give it a global value you can write , all: unset;

# **CSS Responsive designs**

## **What is the meaning of creating Responsive webpage ?**

- This means to design webpage suits all devices, we apply it by set some conditions, each condition represent type of screens and based on each screen we change in style

## **How we apply responsive designs ?**

- We apply it by writing the "`@media (condition){ }`", This is the format of media query. The condition here represent the action which is when TRUE the styling will be applied, Example: `@media print { }` => The inner styling will be applied in case the user printed the page only  
`@media (min-width: 500px) =>` this means the style will be applied from 500px width till (ماشاء الله)  
`@media (max-width: 500px) =>` this means the style will be applied from 0 px width till 500px  
For sure we can add range by using both (min-width:) and (max-width:)

## **Where we write the media query?**

- We have 3 places to write the media :
  1. Below the element in the same CSS file
  2. Made a new file for media only and link it in HTML using link tag and media attribute and the value of that attribute is the condition of media (External)
  3. In style tag in html file and use also media attribute

## **What are the standards of the screens ?**

- `/* Mobile */ => @media (max-width: 767px) { }`
- `/* Small Screens */ => @media (min-width: 768px) and (max-width: 991px) { }`
- `/* Medium Screens */ => @media (min-width: 992px) { }`
- `/* Large Screens */ => @media (min-width: 1200px) { }`
- `/* Custom */ => @media (max-width: 1199px) { }`

## **ADVACE : Create your own Frame work**

- Frame works stand on the concept of built in classes , so merge the properties you always want into one class and insert that class to the elements
- (Note that : Don't create class for ONLY ONE property!! That is not correct)