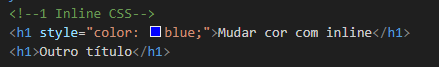
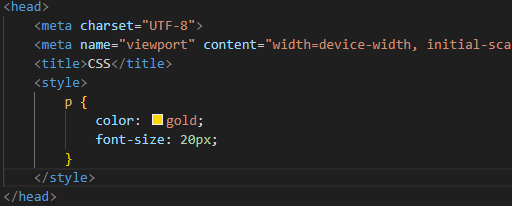
## **CSS**

* **Maneiras de adicionar o Css;**   
  Temos algumas maneiras de adicionar o CSS no Html;

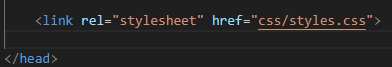
1. **Inline:** quando os estilos são adicionados como atributos dentro da tag html;

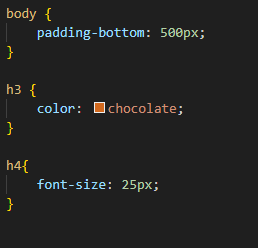


1. **Internal:** quando o CSS é adicionado dentro da tag <head> como exemplo abaixo;



1. **External:** quando o Css é adicionado através de um arquivo externo, e depois importado para o Html com a tag <link>; Ex:<link rel="stylesheet" href="css/styles.css">





### Ordem do CSS

O Html tem uma ordem de preferência por estilização.

Inline > internal/External > padrão do navegador.

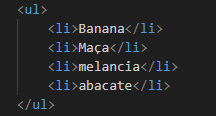
O inline sempre vai prevalecer sobre o Internal/External.

Internal/External: Sempre irá pegar oque foi lido primeiro.



### Múltiplos arquivos de CSS

É possível colocar dois arquivos estilizados em uma página CSS.





### Comentários no CSS

Para utilizar basta colocar os símbolos

/\* E digitar sua mensagem dentro \*/.



### Classes e ID

Classes e Ids são atribuídos a tags do HTML, mas estão diretamente relacionados ao CSS;

Podemos especificar elementos específicos com ele;

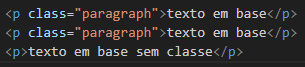
Ids são utilizados para elementos únicos, já as classes servem para um ou mais elementos.

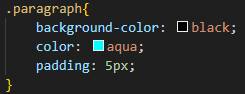
1. **Classes**: Elas são inseridas através de um atributo de

HTML. O valor do atributo é de nossa escolha.

para adicionar no css é necessário colocar

( .nome\_da\_classe { } ).



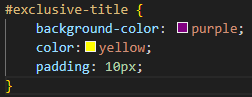


1. **Ids:** Também são atributos no HTML:

Podemos escrever qualquer dado como valor, porém os Ids são únicos podendo usar um por vez sem repetir o mesmo.

para adicionar o ID no CSS basta colocar ( #nome\_ID );



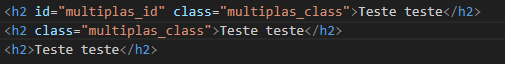


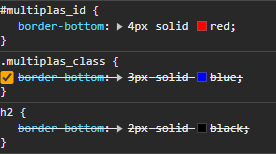
### A ordem dos Seletores

Quando houver alguma regra de CSS em conflito prevalecerá

essa ordem:

ID > Class > Seletor de tag:

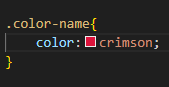
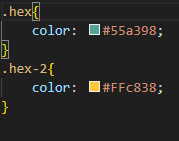


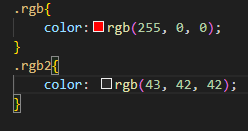


### A cores do CSS

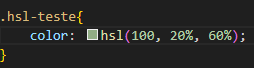
Em css as cores são divididas em grupos.

Temos:

1. **Nomes de cor:** como red ou blue.  
   
2. **Hexadecimal(HEX):**uma união de letras e números que podem criar uma cor. Podemos resumir caso ele se repita ex: #FF5533 podemos escrever como #F53  
   
3. **Rgb:** Configuramos as tonalidades de red,green e blue.



1. **RGBA:** Seria a mesma coisa que RGB porém o A Seria a opacidade.  
   
2. **HSL:** hue,saturation e lightness. Mudando estes valores   
   temos uma cor.

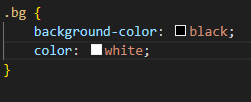


### Background color

Quase todo elemento tem um background, e podemos

mudar a cor dele.

background-color: yellow;

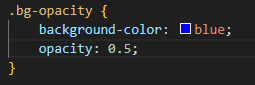


### Background opacity

Podemos alterar a opacidade de uma cor de fundo com Css;

A regra é a opacity;

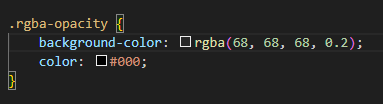
os valores vão de 0 a 1: Sendo 1 visivel e 0 removendo a cor;



### Resolvendo o problema da opacidade

Podemos utilizar o RGBA para estar diferenciando o

background;



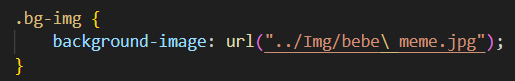
### 

### Background Images

Nessa regra de Css podemos adicionar uma imagem como

fundo de um background;

background-image:url(“local\_do\_arquivo”);

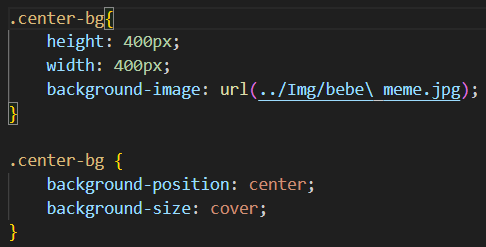


### Centralizando Background

Podemos efetuar esse processo com a utilização do.

background-position: center; (ou outros valores)

background-size: cover;



### Box model

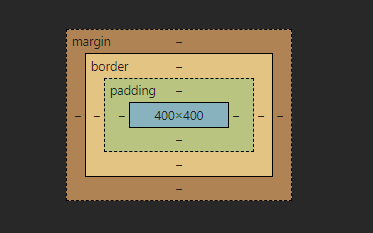
Box model é uma entidade criada em todo documento Html.

Ela entra em alguns tópicos.

1. Altura e Largura;
2. padding;
3. border e margin;

Alguns elementos já vem com essas configurações como

padrão.

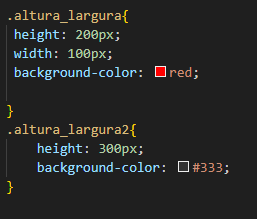


### Height e Width

Altura e largura é o coração do box model;  
Esta estrutura é feita em base no tamanho do elemento,

porém a partir do momento que editamos o css podemos

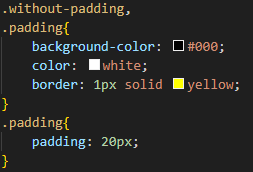
alterar o mesmo.



### Padding

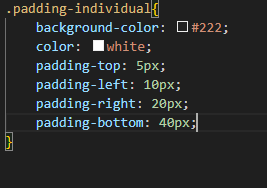
É o espaço entre o conteúdo e a borda do elemento, também conhecimento como espaçamento interno;

Esse é conhecido também por criar uma distância entre o conteúdo e a extremidade do elemento.



### **Padding individuais**

Podemos utilizar o padding separadamente selecionando o lado que vai ter o espaçamento utilizando o “padding-”;  
Exemplo: padding-left, padding-top, padding-right, padding-bottom.



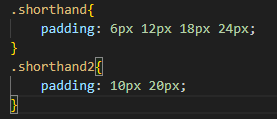
### Shorthand properties

As propriedades de shorthand nos permitem adicionar

padding a todas as direções com uma regra;

Apenas precisamos utilizar a regra padding e configurar como

top, right, bottom e left nesta ordem;  
Exemplo: padding: 10px 5px 12px 20px;  
Esta regra pode ser aplicada em margin também;



### Padding e Width

O padding é adicionado à largura do elemento, e isso pode

ser um problema. Como um exemplo: Necessitamos que

seguir um layout perfeitamente;

Um elemento com 200px de width e 25px de padding tem um

tamanho total de 250px na horizontal;

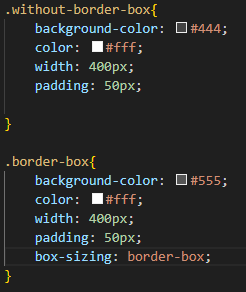
Para resolver o problema poderia ser diminuído o tamanho do

elemento, porém isso dificulta o cálculo também.

Para resolvermos podemos usar a regra de box-sizing e o

valor de border-box, isso faz o elemento respeitar o tamanho

que está em width;



### Border

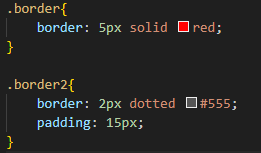
A bordar é o elemento central. Fica entre o padding e

margin;

É geralmente utilizado com propósito decorativo;

A regra é defina em algumas partes: Tamanho, aspecto

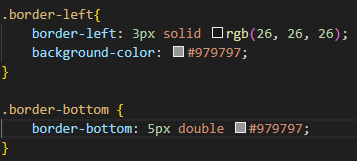
e cor da borda;



### Bordas Individuais

Podemos usar borda aos lados específicos de um

elemento;  
Podemos usar border- “top, right, bottom e left”;



### Border radius “Arredondada”

Com a propriedade “border-radius:” podemos

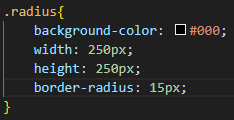
arredondar os cantos de um elemento;

Podemos aplicá-la assim: “border-radius:5px;”

Isso faz que o conto seja arredondado em 5px;

Obs: podemos arredondar mesmo sem especificar uma

borda;



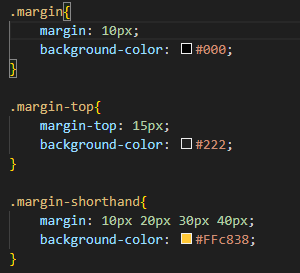
### Margin

A propriedade de margin é responsável pelo

espaçamento externo do elemento;

Podemos aplicar o recurso como aplicamos padding;

Ou seja: lados individuais e também o shorthand;



### Elementos do box model juntos

Em alguns elementos vamos utilizar todos os recursos

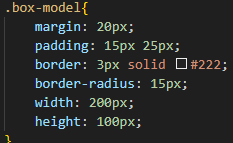
do box model;

Ou seja vamos definir o tamanho (width e height);

Um espaçamento interno (padding);

Decorar o elemento com bordas (border);

Afastar o elemento de outras (margin);



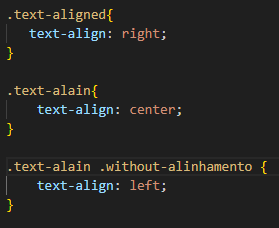
### Alinhamento de texto

Nossos textos podem ser alinhados em várias direções;

Por padrão ele é alinhado a esquerda;

Porém com a regra ”text-align: ” configuramos center

(centro) ou right (direita) ou valor default;

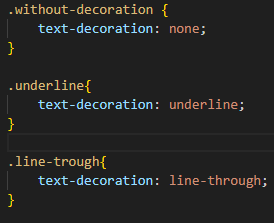


### Text decoration

Com a regra “decoration:” podemos adicionar efeitos ao texto;

Esta regra é utilizada em casos específicos;

A tag “a” tem um underline por padrão, podemos remover isso com a regra de text-decoration eo valor de none;



### Transformação de texto

Com a regra text-transform podemos alterar com o texto é exibido;

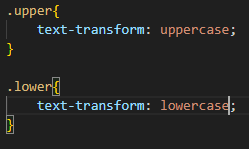
é possível alterar para uppercase ou lowercase

(letras maiúsculas ou minúsculas);

Não há muitos valores para esta regra;

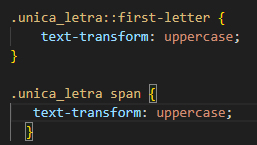
obs: nunca escreva o texto em “capslock” no HTML

Sempre use o uppercase.



Obs: podemos selecionar somente a primeira letra do parágrafo ou outra letra no meio da frase usando esse exemplo.



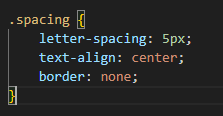


### Espaçamento de letras

Com a regra letter-spacing podemos alterar o espaçamento entre as letras de todo o texto;

Isso é interessante em situações que o layout pede esta mudança;

A regra é aplicada da seguinte maneira: letter-spacing: 5px;



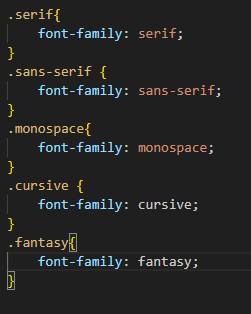
### Fontes

Podemos alterar o tipo da fonte, com a regra font-family;

As regras disponíveis são: Serif, Sans-serif, Monospace, Cursive e Fantasy;

Todos os navegadores tem várias fontes que podemos utilizar;

Podemos adicionar fonts externas, por exemplo com o google Fonts;

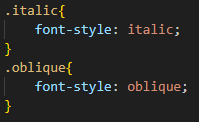


### Estilo de fonte

Nós podemos utilizar a propriedade font-style para mudar o aspecto das letras;

Os valores são: normal, italic e oblique;

O padrão default sempre é o “normal”.

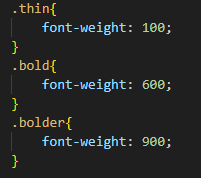


### Font Weight

A propriedade font-weight pode deixar a fonte de texto mais fina ou mais grossa;

Os valores vão de 100 a 900, sendo 100 o mais fino;

Algumas palavras também funcionam com valores, por exemplo: bold (600);

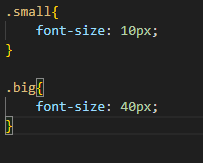


### Font Size

A regra “font-size” é responsável por deixar a nossa fonte grande ou pequena;

Quanto menor o valor declarado, menor a fonte e vice-versa;

Esta regra é declarada em unidade de medida “px” = pixel;

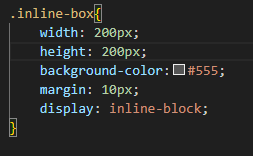


### Display

Em HTML e CSS temos alguns elementos que são considerados de bloco e outros inline;

A tag div é um exemplo de block e span um exemplo de inline;

Com a regra display podemos mudar este comportamento, ou seja deixar uma div como inline;

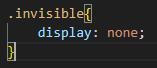


### Escondendo elementos

Existem situações que precisamos ocultar elementos;

Basta adicionar a regra display com o valor de none;

Então o elemento não é mais exibido, porém ainda consta no HTML;



### Posição dos elementos

A regra position e seus valores são responsáveis por posicionar o elemento na tela;

Temos algumas possibilidades: relative, fixed, absolute, sticky e mais;

O valor padrão é static, todo elemento começa com esta posição;

Esta regra é essencial quando precisamos mudar onde o elemento deve ficar no projeto;

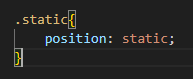
1. **Position Static:**

O valor de static na position não faz nada com o elemento;

Pois este é o valor padrão, ou seja, todo elemento já tem este tipo de position;

Outros valores são afetados pelas regras top,left,right e bottom porem static não.

Static apenas segue o fluxo do HTML;



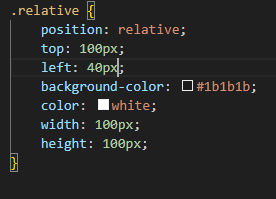
1. **Position relative:**

Com a position configurada como relative temos mais possibilidades que static;

Agora as regras top,left,right e bottom, movem o elemento na tela;

O elemento ainda segue o fluxo do HTML;

OBS: Normalmente não utilizamos estas regras de posição com relative;



1. **Position absolute:**

Com o valor de absolute em position, o elemento pode ser

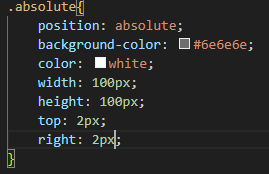
movido pela tela toa;

Ou seja, Quebrarmos o fluxo do HTML;

Esta regra também é afetada por top,left, right e bottom;

Posicionar com absolute pode ser uma solução ou um

problema, dependendo do ponto de vista;



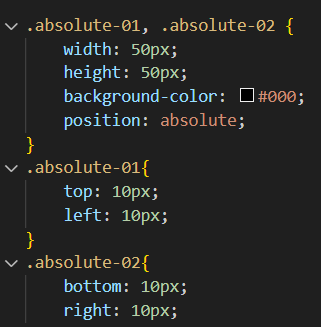
**Relative com absolute:**

Podemos resolver o problema de absolute com relative;   
 Um elemento com position absolute é ligado ao elemento

mais próximo com posição relativa, se não ele é ligado ao

body;

Então com um container com posição relative, podemos controlar melhor a área de ação dos elementos com absolute;

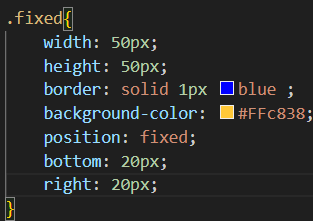


1. Position fixed:

Com fixed o elemento pode ser fixado na tela;

Mesmo colocando o scroll para baixo ou para cima ele ficara fixo na mesma posição;

O recurso é sempre utilizado para colocar uma barra de navegação fixa;



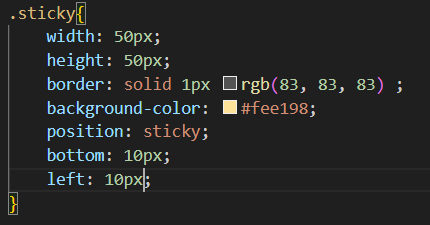
1. **Position sticky:**

Sticky faz com que o elemento fique fixo na tela:

Mas tem outra função referente ao mesmo quando ele volta

para sua posição original ele se comporta como relative:

A posição seria onde ele foi posicionado no HTML:



### Z-INDEX

Se temos dois elementos com as mesmas posições ou se

eles se colidem na página, podemos escolher qual que ficará

mais visível ou exibido.

Utilizamos o z-index para isso.

O elemento com maior valor prevalece;



## **RESPONSIVIDADE**

### O'Que seria responsividade ?

Responsividade é a técnica de adaptar uma página web para vários dispositivos diferentes;

Ou seja, temos mudanças de CSS baseada na resolução;

Detectamos o que o usuário está utilizando, e adaptamos o nosso site a resolução;

As regras de CSS são as mesma, porém dentro de um recurso chamado media query;

### Configurando a responsividade:

Primeiro vamos adicionar uma meta tag ao nosso head;

Ela é:   
<meta name=”viewport” content=”width=device-width”, initial-scale=1.0”>

Isso faz com que o conteúdo se adapte com os dispositivos;

Adicionamos também uma escala de 1:1;

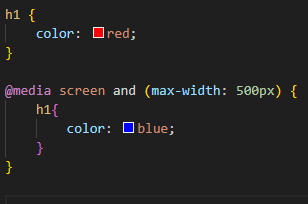


### Media query:

Media query é o recurso que utilizamos para criar os breakpoints;

Os breakpoints adaptam o nosso projeto para diferentes tipos de tela;

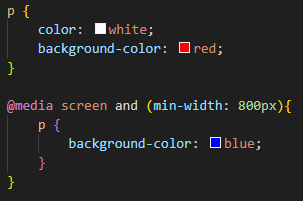
Configuramos uma largura, e depois as regras começam a serem alternadas dependendo do tamanho da tela;



### Media query com min-width:

Se utilizamos min-width em vez de max-width, a media query funciona ao contrário;   
Então podemos desenvolver projetos com uma técnica chamada mobile first;

Os projetos que tem mais usuários mobile geralmente são feitos em mobile first;



### Padrão de breakpoint:

Estes breakpoints são utilizados frequentemente:

600px e abaixo: celular;

768px até 600px: tablets;

992px até 768px: mini laptops:

992px e acima: laptops e desktop;

É comum em projetos profissionais utilizarmos estes valores

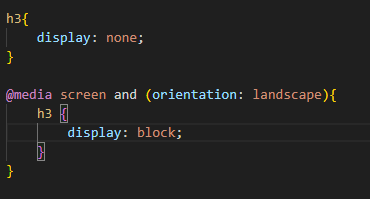
para desenvolvermos para diferentes dispositivos;

### Adaptação para landscape:

Podemos alterar CSS apenas para usuários com visão em landscape;

Para isso precisamos de um atributo chamado orientation com o valor de landscape, na media query;

O código é ativado quando a orientação é alterada;



## **FLEXBOX**

### O que é flexbox ?

Flexbox é um valor da propriedade display;

Esta é a maneira mais utilizada para acondicionar elementos

em um container;  
Temos diversas regras dentro do flex, veremos nas aulas

seguintes;

O felx deve ser considerado no elemento do pai, e os elementos filhos serão os movimentados;

Os elementos filhos também podem ter regras específicas;

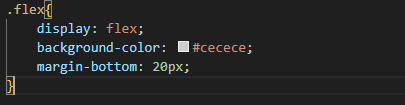
### Aplicando Flex

Para aplicar o flex vamos precisar de uma estrutura base;

Consiste em um container e elementos filhos dentro do container;

Colocamos a regra display com o valor de flex no container, e agora os elementos estão seguindo o fluxo do flex;

Com felx todos os elementos se comportam parecido com os elementos inline;

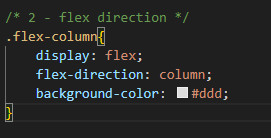


### Flex direction

Podemos mudar o comportamento inline do flex (chamado de row);

Alterando o flex-direction para column, teremos agora os itens se comportando como elementos de bloco;

O valor padrão de direction é row;



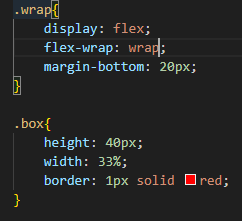
### Flex Wrap:

O flex tenta colocar todos os elementos na mesma linha por padrão;

Porém há situações que queremos x elementos por linha;

Para isso acontecer devemos aplicar a regra flex-wrap com o valor de wrap;

Agora as linhas respeitam a largura dos elementos;

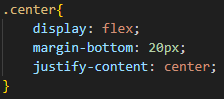


### Posicionando conteúdo:

Com justify-content é possível mudar como o conteúdo é posicionado no eixo horizontal;

Temos alguns valores interessantes nesta propriedade;

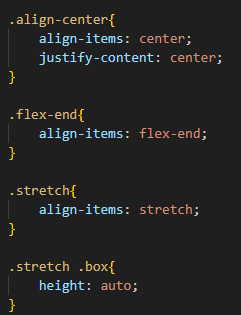
Por exemplo: centralizar os elementos na horizontal, basta colocar o valor de center;



### Posicionamento na vertical:

Com o justify-content modificamos os elementos na horizontal, já align-items nos permite mudá-los na vertical;

A propriedade tem vários valores, como o center;

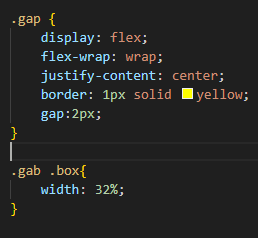


### Gap:

O gap é uma regra que serve para colocar espaço entre elementos

que estão no flex;

Nós especificamos a medida em px, por exemplo;   
E este valor é adicionado entre cada um dos elementos;

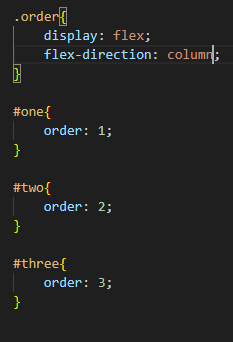


### Order:

Com a order podemos mudar a ordem dos elementos;

Esta propriedade é utilizada nos elementos filhos;

Agora começamos com as regras que são aplicadas aos elementos filhos, não ao elemento pai;

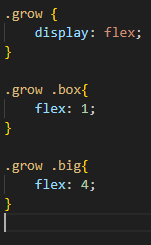


### Grow:

Com a regra flex-grow podemos mudar a proporção de um ou mais elementos filhos;

Nota: A width precisa estar sem valor, com automática;

Exemplos: Se colocarmos grow como 2, o elemento vai crescer duas vezes mais que os outros quando estiver se adaptando no container;

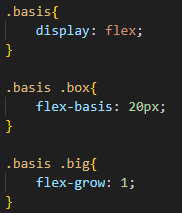


### Basis:

A regra flex-basis configura a largura base do elemento;

Nós podemos trabalhar com basis e grow juntas!

Grow vai preencher toda a largura que basis deixar vazia;

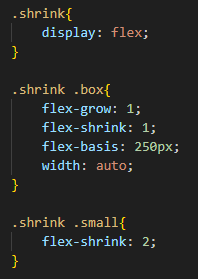
****

### Shrink:

Shrink é o oposto de grow;

Quando utilizamos precisamos manter o tamanho dos outros elementos, então o elemento do shrink diminui o seu tamanho para manter o dos outros;

Nota: Usamos esta regra em conjunto de basis e grow;



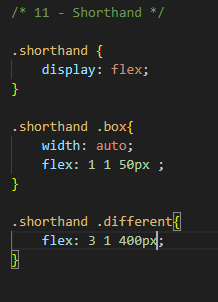
### Flex shorthand:

Utilizando apenas flex conseguimos configurar grow,shrink e basis;

Colocamos os valores nesta ordem também;

Desta maneira: flex 2 1 100px;

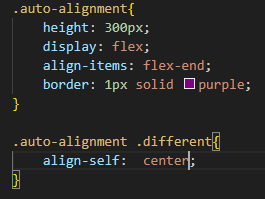
Isso dá ao elemento: grow = 2, shrink = 1 e basis = 100px ;



### Auto alinhamento:

A regra align-self alinha um elemento diferente dos demais;

Podemos centralizar um elemento enquanto os outros seguem a regra de alinhamento do container;

****

## Estilizando a página

### Scrool do site (Barra de rolagem):

Podemos estilizar da forma que acharmos melhor a barra de rolagem na lateral da página.   
Deixar ela oculta entre outros objetivos que sua imaginação desejar.   
Iremos aplicar regras de css em um objeto específico.

::-webkit-scrollbar

::-webkit-scrollbar-track

::-webkit-scrollbar-thumb

