**Estudos de front- HTML E CSS**

SITE PARA VERIFICAR A COMPATIBILIDADE DOS NAVEGADORES COM TECNOLOGIAS:

CANIUSE.COM....

[**DISPLAY- FLUXO DE LAYOUT-**](3.%20aula%20210-211)

**- inline:** São os elementos q vao ficando um do lado do outro, sem quebrar

-**Block**: inicia sempre uma nova linha e não permite continuar na mesma linha. Sempre com novos cheiros e etc.

-**in line block :** ele passa a receber um width e tudo mais, mas permanece na linha.

[Elementos independentes](4.%20exercicio%20214)

**- span :** é igual o div, mas não tem o display: block (dividido em bloco)

[Imagens](5.%20AULA%20216%20IMAGENS)

**.srv** : É o formato que mais vai ser utilizado pois é personalizável. Nunca perde qualidade.

PADDING E MARGIN

posso personalizar o meu **padding** e **margin** em relação ao tamanho que eu quero no *top, right, botton, left.*

padding: 20px 60px 40px 10px;

se eu coloco assim, segue sentido horário no tamanho.

margin: 20px 40px;

se coloco assim é top/bottom e right/left

mas posso tbm indicar, assim:

margin-top: 20px; margin-bottom: 20px; margin-right: 40px;   margin-left: 40px;

[Position](7.Posicionamento)

Relative- posiciona o objeto em relação a sua própria posição anterior- right, left, top and bottom

Absolute- posiciona o objeto em relação ao pai, ancestral, se não tiver, é em relação ao html.

Fixed- posiciona de acordo com a pagina do navegador em si

Semantica html

**Nav**- uso para indicar navegação, geralmente só para as mais importantes do site

**Section**- uma sessão que não faz sentido sozinha. Ela não marca o texto na acessibilidade. Só se eu colocar o aria-label=””.

**Article**- é um artigo, uma região da pagina que se tirada de contexto continua fazendo sentido.

Header- indica ai cabeçario, pode ser usado dentro de outras tags.

**Footer**- geralmente para rodapé. Tbm pode ser usado dentro de outras tags. Ambos marcam o texto no quesito acessibilidade.

**Aside-** é usado para informações adicionais no site.

**main-** é o que marca a parte principal do site.

Todos eles visualmente são como divs, porém alguns marcam o texto(acessibilidade).

Listas

* **ul**- cria uma lista desordenada. Dentro de ul coloca li para cada item. Posso colocar ul dentro de ul.
* **ol**- cria uma lista ordenada, padrão já vem numerado mas todos podem ser modificados. Usa li para cada item.
* **dl**- definition list. Listas de definição. Não usa li, usa lt e ld. dt é o item, e dd é a descrição.

Dicas de formatação básicas

* **a**- tiro o text decoration, coloco em display block/inline block, dou background e padding.
* **ul**- list style none, zero margin e padding, coloco em display flex (para listar lado a lado se quiser) e dou um gap.
* **Body**- margin 0px. Ele vem com margem de 8px padrão

Outros comandos/atributos

* **blockquote**- citação. Tipo uma div, mas já alinha como citação e permite colocar link do autor. Coloco o p dentro dele com texto e p fora com nome do autor.
* **q**- o que é uma citação interna, dentro do próprio p. tbm pode recer o cite=”https...”
* **cite**- usado para citar obras, filmes, autores. Já deixa em itálico. Sem link. Dentro do p.
* **em**- para dar ênfase a alguma parte do texto. Dentro do p.
* **Strong**- ênfase um pouco maior, coloca em negrito. Dentro do p.

HTML NÃO É PRA DEFINIR ESTILO E SIM DAR SEMANTICA. NÃO USAR O STRONG PARA COLOCAR EM NEGRITO E NEM O CITE PARA POR EM ITALICO OU ALGO ASSIM. USAR PARA ORGANIZAR O TEXTO.

Unidade de tamanho css

* **Px**- pixel
* **Rem**- 1 rem = 16px/
  + O rem ajuda na acessibilidade. Pois ele acompanha o navegador, se o navegador tiver sido modificado para fonte grande, ele acompanha.
  + O html em si é em rem, se eu mudar o font size do html ele altera o tamanho padrão do rem.

Comandos de altura e largura

* **Vh-**viewport height- considera toda a altura. Se eu coloca algo com 100vh, ele pega toda a tela.
* **Vw-**viewport width- considera toda a largura

Calc

* **Calc**- é um comando para calcular qualquer coisa dentro de css
  + Ex: height: calc(100vw / 3) – divide o tamanho da tela por 3.
  + height: calc(100vw -200px) diminui 200px da tela, indepdente do tamanho da tela.

Tipografia das letras

* **font-:**
  + **size-**tamanho da letra, posso por em rem agora
  + **Family-** a família, o tipo de letra. Posso colocar mais de uma, tipo, se não tiver a primeira, vai a segunda depois da virgula.
  + **Weight-** para colocar em negrito ou não. Da pra colocar valores de 100 a 900, a largura da letra
  + **Style-** italic, oblique. Colocar em itálico
* **Line-height:** altura da linha. Geralmente 1 é padrão
* **Text-:**
  + **decoration:** pra colocar underline, overline e etc
  + **transformation:** para colocar em maiúsculo, minúsculo, primeira maiusculuna (capitalize) e etc
  + **indent:** pra dar um recuo na primeira linha do paragrafo.
  + **Align:** alinhamento do texto, começa na esquerda
  + **Shadow:** colocar sombra no texto. 3 valores e a cor- primeiro px alinha a sombra pra esquerda ou direita, segundo alinha pra baixo ou pra cima, e terceiro o blush. E a cor
* **Letter-spacing**: para espalhar as letras

[Background](8.background)

**Background** sozinho me da a possibilidade de colocar todos os outros que vou citar, só ir pondo.

* **Background**-:
  + **Image**: url(‘*caminho da img*’). Pra colocar uma imagem como fundo.
  + **Repeat**: se quero que a imagem repita tanto pro lado quanto pra baixo. (padrão é repetir, então ponho “no repeat”)
  + **Size**:
    - cover- vai cobrir toda area da minha tela, mais ajustadamente com apenas uma imagem. Até corta pedaço dela se for preciso. Mais usado
    - Contain- preenche tudo, pode deixar espaço em branco. Ou repetir a imagem se ela for pequena, tanto pra um lado quanto pra baixo. Se usar esse, tem que usar o no repeat.
    - Poderia definir um valor especifico.

Pseudo classes de estado

Quando quero destacar geralmente um link quando o mouse passa em cima ou clica. Dentro do css fica:

* **a:hover{}-** se o mouse passa em cima ele muda a cor do botão. Defino a cor do background, da letra e se quiser da borda.
* **a:focus-** usando o tab o elemento fica em foco. Geralmente é bom por borda. A borda é o padrão dos navegadores.
* **a:active-**quando clicamos no elemento o que acontece.
* **a:visited-** quando já clicamos no site antes ai fica diferente. Esse se sobrepõe sobre os outros, então ter cuidado ao usa-lo

PARA LISTAS SELETORES:

Pintando os elementos selecionados. Serve pra listas ou pra divs mesmo... Só precisa ter pai e filho.

/\* SELECIONAR primeiro elemento \*/

.lista-1 li:*first-child* {

  background: #caf;}

/\* último elemento \*/

.lista-1 li:*last-child* {

  background: #cfa;}

/\* terceiro element, ou o element do numero que eu colocar\*/

.lista-1 li:*nth-child*(3) {

  background: #fda;}

/\* elementos ímpares \*/

.lista-2 li:*nth-child*(odd) {

  background: #cfa;}

/\* elementos pares \*/

.lista-2 li:*nth-child*(even) {

  background: #adf;}

/\* de 4 em 4 ou de 2 em 2 é 2n\*/

.lista-2 li:*nth-child*(4n) {

  background: #caf;

## :not

O :not nega a seleção de um elemento específico.

Lista com outras pseudo-classes: <https://developer.mozilla.org/en-US/docs/Web/CSS/Pseudo-classes>

/\* todos menos first-child e last-child \*/

ol li:*not*(:*first-child*, :*last-child*) {

  background: #adf;}

/\* todos h2's dentro de section, menos o que tiver a classe .contato \*/

section h2:*not*(.contato) {

  text-decoration: underline;}

pseudo elementos

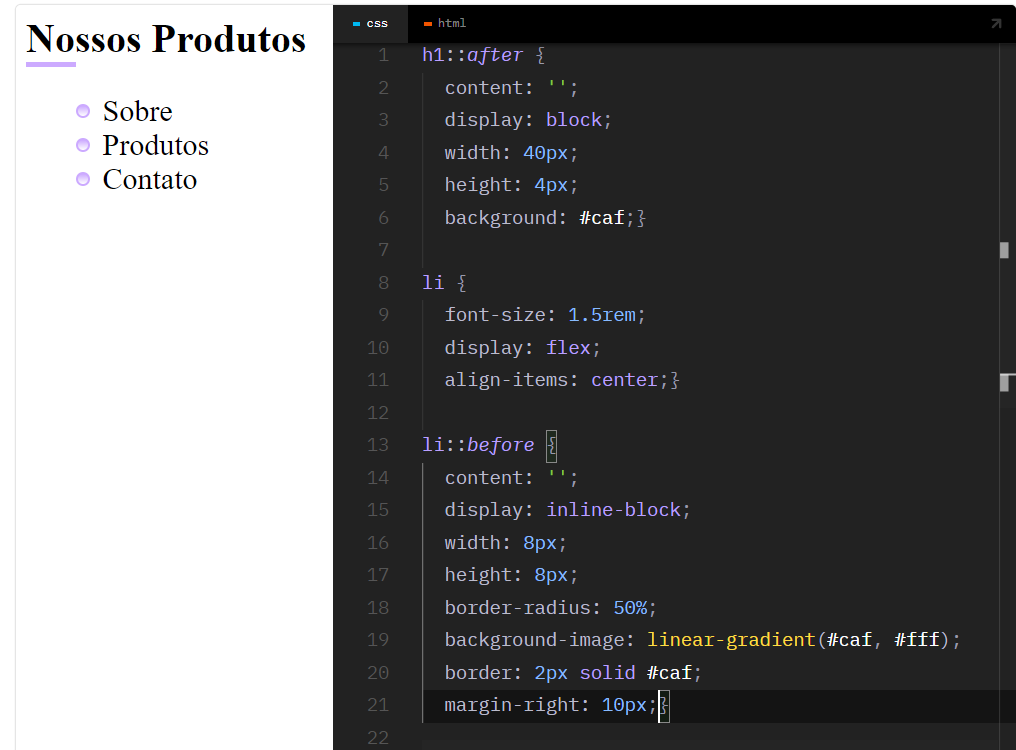
**Uso para estilizar meu html sem precisar criar uma div.**

* **::after-**fica depois do elemento
* **::before-** fica antes do elemento
* h1::*after* {
* content: '';
* display: block;
* width: 40px;
* height: 4px;
* background: #caf;}

**Nesse caso dentro do contente fica vazio por que quero colocar um detalhe e não um texto. Mas poderia colocar um texto normalmente que mudaria no html.**

**Nesse exemplo criou uma linha rocha.**

**USA MUITO PARA CRIAR OS DETALHES ANTES DE LISTA. AS BOLINHAS OU ALGO ASSIM.**

****

# [RESPONSIVIDADE](10.%20RESPONSIVIDADE)

Geralmente é bom começar pelo mobile, pra deixar o mais importante primeiro.

***ELEMENTOS DE RESPONSIVIDIADE***

1. ***FR-*** ajusta de acordo com o tamanho
2. ***Max-width: 100%***
3. ***Vw- viewport***

COLOCAR O META TAG – VIEWPORT, pois ela determina que a largura dos pixels e tudo mais vai acompanhar o do device, e não a qualidade da tela em si.

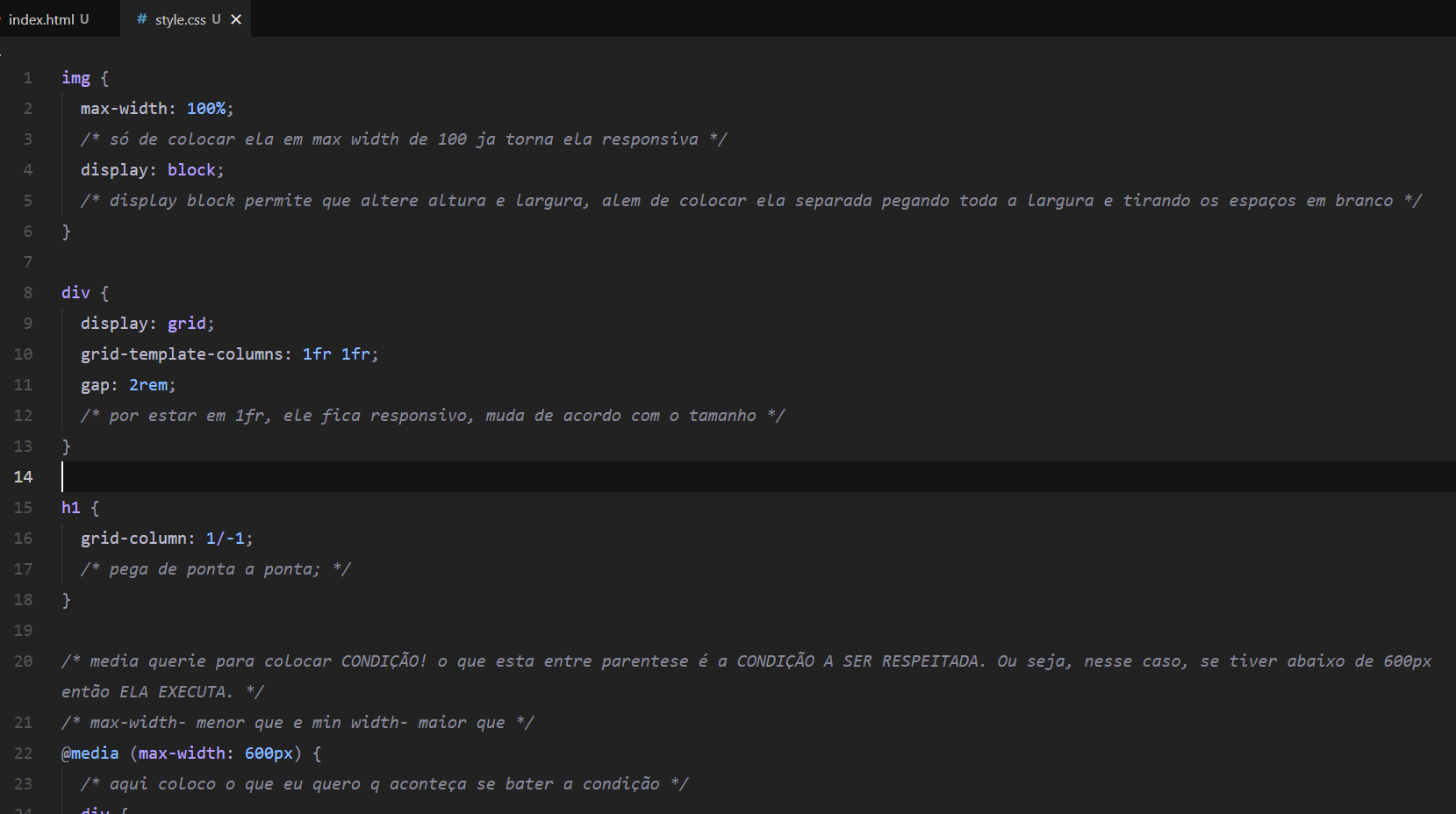


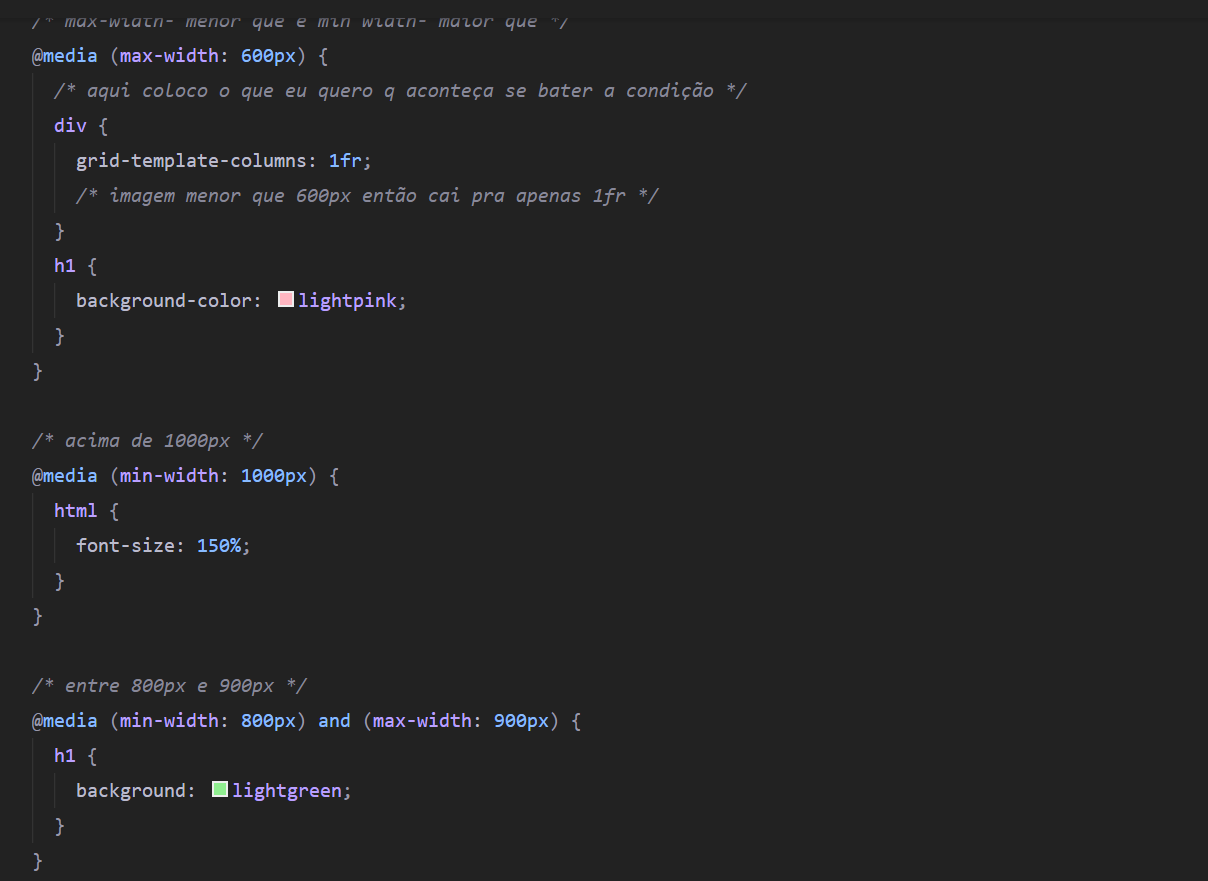
Width segue o do device, initial scale é 1, padrão. Pra selecionar ela é só digitar “meta-vp”

# [MEDIA QUERIES](10.%20RESPONSIVIDADE)

Utilizado para responsividade, como se fosse uma CONDIÇÃO. A partir de tal ponto, executa esse código.

* **A ORDEM IMPORTA, ENTÃO A PARTE RESPONSIVA DO MEU SITE EU SEMPRE ESCREVO DEPOIS DO GERAL.**
* Pra tirar algo é só colocar **display-none**

****

****

# [Grid responsivo](https://www.origamid.com/slide/html-e-css-para-iniciantes/#/0603-grid/1)

Eu consigo fazer meio que uma condição de media querie só que usando apenas o **repeat(auto-fit)** do grid.

.minmax {

  grid-template-columns: repeat(auto-fit, minmax(300px, 1fr));

}

Min max seria a imagem da bike e o texto, autofit faz com que ela não deixe espaços em brancos quando tiver com mínimo de 300px, e quando for maior deixa em 1 fr.

.galeria {

  grid-template-columns: repeat(auto-fit, 150px);

}

Varias imagens, uma galeria, máximo da coluna vai ser 150px, se tiver mais espaço que isso coloca outra coluna ao lado com outra imagem.

div {

  display: grid;

  gap: 1rem;

  margin-bottom: 1rem;

}

# Order

@media (min-width: 300px) {

  div img {

    order: 2;

  }

}

Mudo a ordem que algo aparece na tela colocando uma condição.

Poderia usar tbm o ***grid-row: 2;***