**RESENTANDO O CSS**

\*, \*::before, ::after {

margin: 0;

paddign: 0;

border: 0;

box-sizign: border- box;

}

**FUNÇÃO REPEAT()**

grid-template-columns: 1fr 1fr;

ou

grid-template-columns: repeat(2, 1fr);

**:NTH-CHILD()**

<div class=”container”>

<!-- Oi em várias línguas -->

<p>Oi</p>

<p>Hello</p>

<p>Hallo</p>

<p>Hola</p>

<p>Hej</p>

<p>Bonjour</p>

<p>Ciao</p>

<p>Debry</p>

<p>Dobrý den</p>

<p>Bună ziua</p>

</div>

div.container:nth-child(1) {

font-size: 1.2rem;

}

/\* nth-child(1) pega o primeiro filho da div.container \*/

div.container:nth-child(n + 5) {

font-weight: bold;

}

/\*

nth-child(n + 5) pega todos os elementos filhos da div.container a partir do quinto

Neste caso, seriam todos os últimos 6 <p> (pois conta o quinto elemento também)

\*/

**:NOT**

div.container:not(:last-child) {

margin-bottom: 20px;

/\*

:not = não , :last-child = último filho

Neste caso, todos os filhos receberão uma margin-bottom de 20px, com exceção do último filho

\*/

**ELEMENTO NO CENTRO COM POSITION ABSOLUTE**

div {

position: absolute;

top: 50%;

left: 50%;

transform: translate(-50%, -50%);

**OBJECT-FIT**

div > img {

object-fit: cover;

}

/\*

Faz com que objeto cubra todo o espaço disponível do elemento pai sem que haja uma distorção

\*/

**SCROLL-SNAP-TYPE JUNTO COM SCROLL-SNAP-ALIGN**

Faz com que um objeto filho desalinhado pelo scroll se alinhe automaticamente dentro do objeto pai pelo browser

<div class=”list”>

<div class=”list-item”></div>

</div>

PRIMEIRAMENTE NO OBJETO PAI

.list {

scroll-snap-type: x mandatory;

-webkit-overflow-scrolling: touch;

}

DEPOIS NO OBJETO FILHO

.list .list-item {

scroll-snap-align: start;

}

/\*

start para que o objeto seja alinhado a partir do seu início

-webkit-overflow-scrolling: touch é necessário para garantir o funcionamento em alguns dispositivos

\*/

**REMOVENDO EVENTOS ONDE CURSOR ESTIVER**

div {

pointer-events: none;

}

**SCROLL SUAVE**

div {

scroll-behavior; smooth;

}

**DICA PARA SLIDER/CAROUSEL**

.slider {

display: flex;

width: 800px;

heigh: 400px;

background-color: lightgray;

border: 1px solid gray;

margin: 0 auto;

}

.slider .item {

background-color: tomato;

flex: 1 0 100%;

transform: translateX(-800px);

}

/\*

O truque aqui está no flex: 1 0 100% e no transform: translateX(-800px)

flex-grow: 1; para expandir o item por todo o espaço disponível da div.slide

flex-shrink: 0; para os itens que extrapolam a largura da div.slide não reduzam de tamanho

flex-basis: 100%; para que os itens tenha no mínimo 100% da largura da div.slide

transform: translateX(-800px); é o que fará os itens se deslocarem

800px porque é o valor da largura da div.slide (ou seja, do próprio slide) e o sinal negativo é para os itens se desloquem para a esquerda

\*/

**PSEUDO-ELEMENTO ::BEFORE E ::AFTER**

p::after {

content: ‘’;

width: 6px;

height: 6px;

border-radius: 50%;

background-color: blue;

}

/\*

Um pseudo-elemento necessita de um conteúdo (content) para existir, por isso o content: ‘’;

Neste exemplo será criada uma bolinha depois do parágrafo

\*/

**LINEAR GRADIENT (DEGRADÊ)**

div {

background-image: linear-gradient(to bottom, blue, red);

}

/\*

to bottom = de cima para baixo

do azul (blue) para o vermelho (red)

\*/

**IMAGENS DE FUNDO (SEM VALOR SEMÂNTICO)**

div {

background-image: url(‘assets/img/imagem-fundo.png’)

background-size: 100px 100px:

}

/\*

Teremos uma imagem de fundo. Por padrão ela irá se repetir

O background-size (opcional) se refere ao tamanho que você quer delimitar para a imagem. Podemos colocar valores como cover e contain também

\*/

div {

background-image: url(‘assets/img/imagem-fundo.png’)

background-repeat: repeat; /\* padrão \*/

ou

background-repeat: no-repeat; /\* não repete \*/

ou

background-repeat: repeat-y; /\* repete vertical \*/

ou

background-repeat: repeat-x; /\* repete horizontal \*/

}

div {

background-image: url(‘assets/img/imagem-fundo.png’)

background-position: left top;

ou

background-position: center center;

}

/\*

Sendo a imagem maior que a div

left top 🡪 posiciona a imagem a partir da parte superior esquerda da imagem (padrão)

center center 🡪 posiciona a imagem a partir do centro da imagem

\*/

div {

background-image: url(‘assets/img/imagem-fundo.png’)

background-attachment: fixed;

}

/\*

fixed deixa a imagem fixa no fundo

a propriedade padrão é o scroll

\*/

**SHORTHAND BACKGROUND**

div {

background: black url(‘assets/img/imagem-fundo.png’) center center no-repeat cover fixed

}

/\*

ORDEM:

color 🡪 image 🡪 position 🡪 repeat 🡪 size\* 🡪 attachment

\*: o size pode não funcionar, mas está na shorthand de acordo com o W3C

\*/

**DICA PARA RESPONSSIVIDADE DE IFRAME**

iframe {

width: 560px

max-width: 100%

aspect-ratio: 16 / 9;

}

/\*

É importante tirar as propriedades width e height que já vem do youtube dentro da tag <iframe>

width: 560px; 🡪 padrão do YouTube

max-width: 100%; para gerar a responsividade;

aspect-ratio: 16 / 9; 🡪 para manter o iframe sempre nesta proporção, que é a padrão do YouTube

\*/

**FOCO NO INPUT**

form input:focus-within {

background-color: white;

}

/\*

:focus-within 🡪 o que acontece quando o input estiver em foco

Neste exemplo, ao caixinha de digitar do input o input ficará branco

\*/

**SELECIONANDO ELEMENTO QUE VEM DEPOIS DE UM OUTRO ESPECÍFICO**

.nav li + li {

margin-left: 30px;

}

/\*

li + li 🡪 pega todos os <li> que vem após de um <li>

Por exemplo, se tivéssemos 3 <li> a margin-left iria ser aplicada somente nos 2 últimos <li>

O sinal de + aqui poderia significar “após”

\*/

**DIMINUINDO OU AUMENTANDO O ESPAÇO ENTRE AS LETRAS**

p {

letter-spacing: -0.005em; /\* diminuindo \*/

ou

letter-spacing: 0.005em; /\* aumentando \*/

}

/\*

letter-spacing altera o espaçamento entre as letras, diferentemente do line-height que altera o espaçamento entre as linhas do texto

\*/