KORU

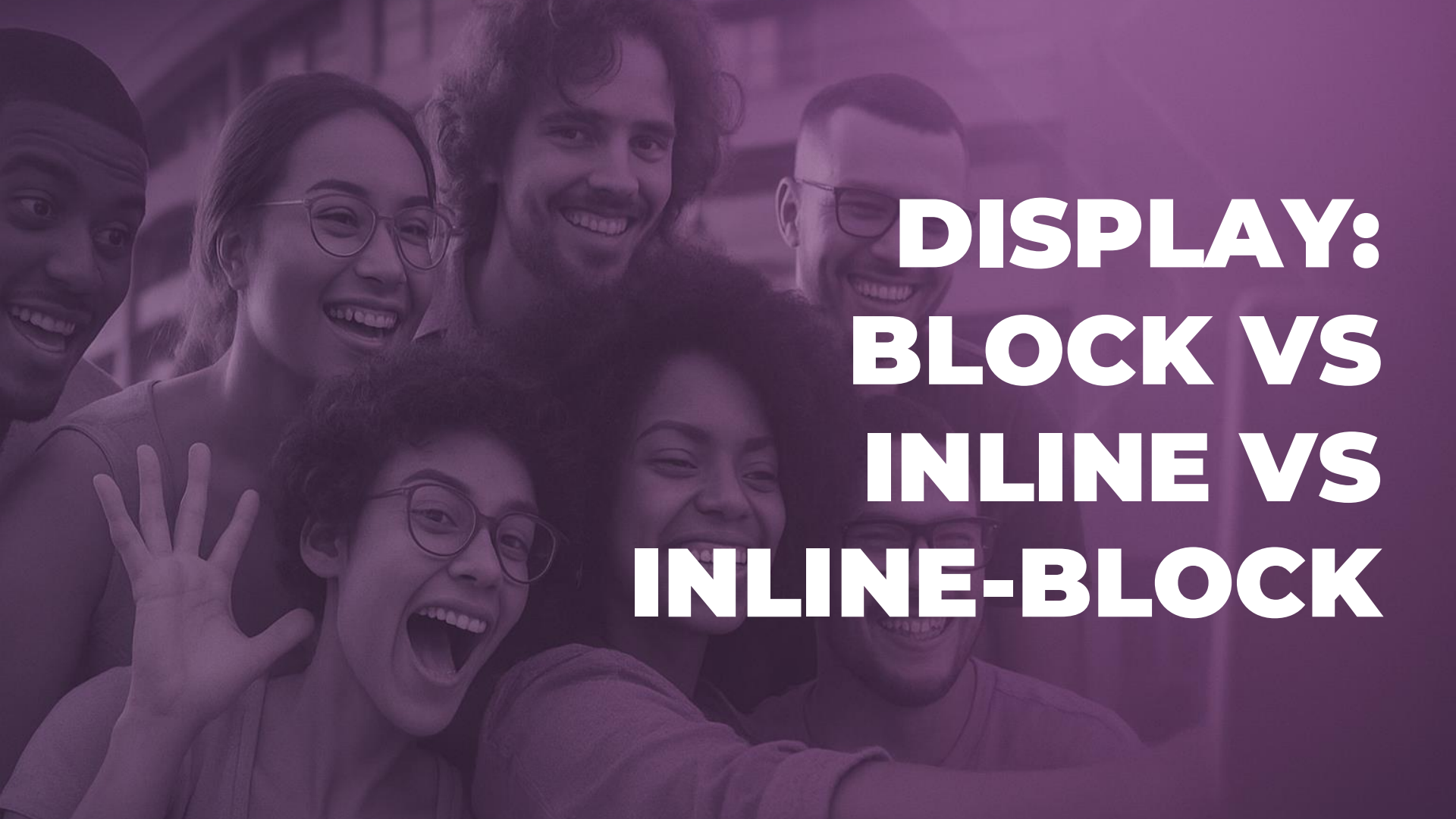

DESENVOLVE
programa de formação
e inclusão de talentos
em tecnologia
Grupo
Boticário
KORU


BOAS-VINDAS

BOAS-VINDAS

Recap Rápido: vimos, na última aula, SS Básico (sintaxe, seletores, box model, cores e fontes).

Hoje: Posicionar elementos e fazer o website ser responsivo (funcionar em qualquer dispositivo)


DISPLAY:
BLOCK VS
INLINE VS
INLINE-BLOCK

DISPLAY:

inline

O elemento não começa em uma nova linha e ocupa apenas a largura necessária. Você não pode definir a largura ou a altura.

block

O elemento começa em uma nova linha e ocupa toda a largura disponível. E você pode definir valores de largura e altura.

inline-block

É formatado como um elemento inline, onde não começa em uma nova linha. MAS, você pode definir valores de largura e altura.

DISPLAY:

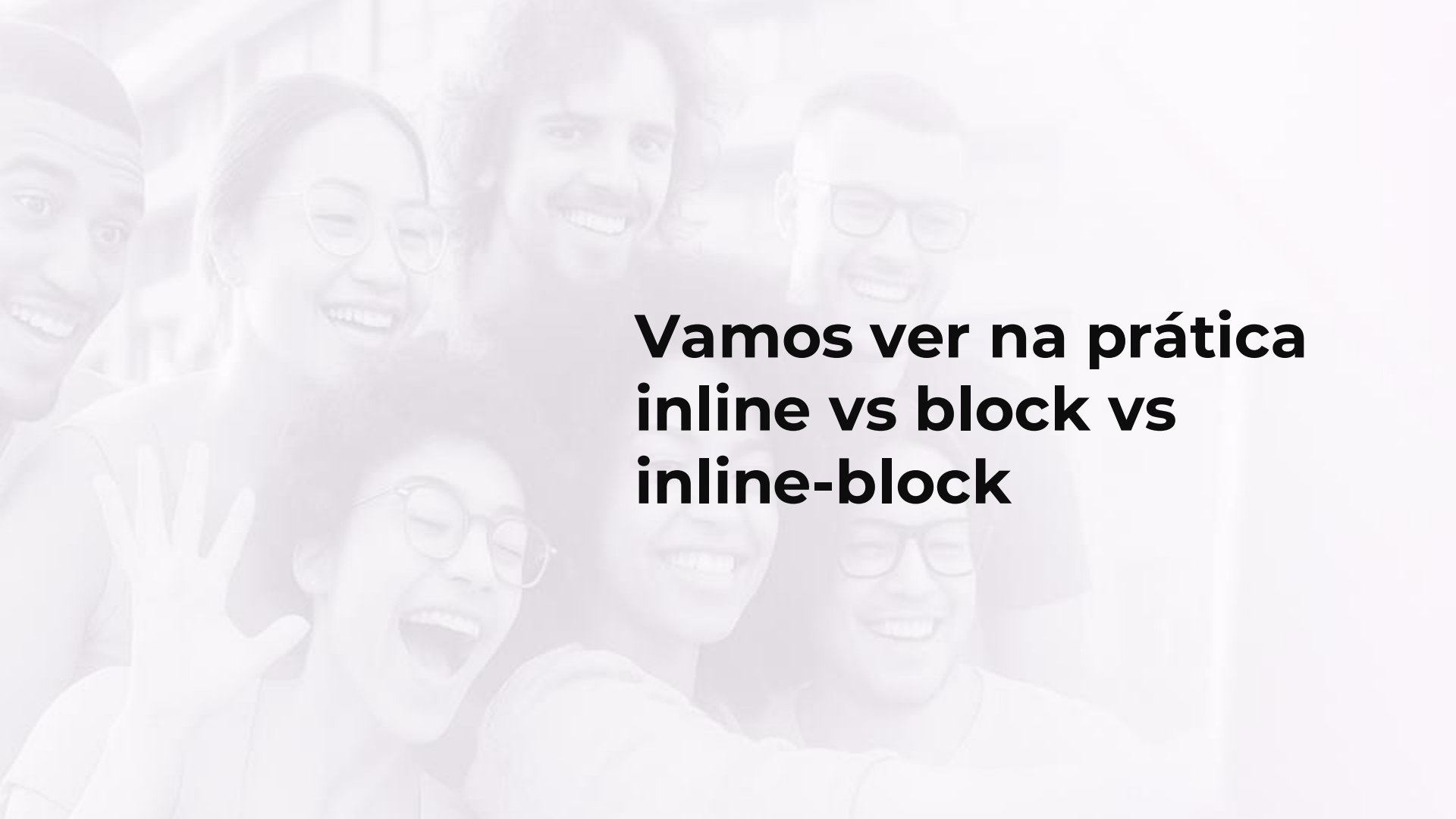
```html
<!-- Principais elementos com display inline por padrão -->
<a href=""></a>
<span></span>
<img src="" alt="">
<strong></strong>
<small></small>
<input type="text">
<button></button>

<!-- Principais elementos com display block por padrão -->
<div></div>
<p></p>
<h1></h1> <!-- Todos os títulos -->
<ul></ul>
<ol></ol>
<li></li>
<section></section>
<article></article>
<!-- e vários outros -->
```

Nenhum elemento HTML por padrão é inline-block, flex ou grid. 👀

DISPLAY:

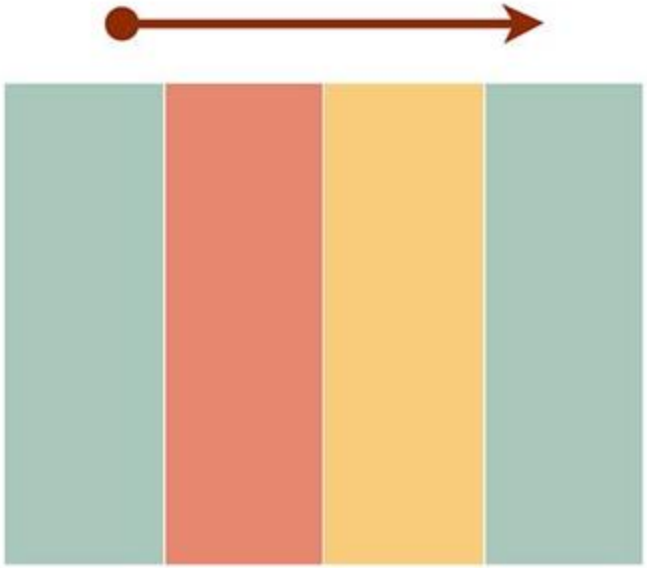
none

O que você acha que significa display: none?



Vamos ver na prática inline vs block vs inline-block

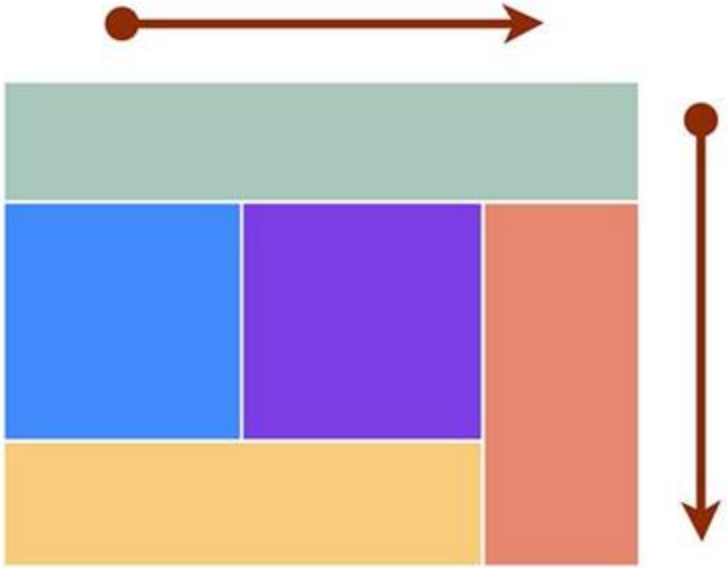

DISPLAY: FLEX
DISPLAY: GRID



Flexbox

One Dimensions

VS



CSS Grids

Two Dimensions


DISPLAY: FLEX

DISPLAY: FLEX

flex

o uso do flex é organização de itens em uma dimensão (esquerda → direita / cima → baixo)

Uma boa analogia é organização de itens em uma prateleira.

DISPLAY: FLEX

Container e item

a distinção mais fundamental no flexbox.

Primeiro você deverá possuir um **flex container**.

Os filhos diretos do container são **flex items**.

DISPLAY: FLEX

```
<div class="container">
 <div class="item">Item 1</div>
 <div class="item">Item 2</div>
 <div class="item">Item 3</div>
</div>

<style>
 .container {
   display: flex;
 }
</style>
```

flex container

Cada um dos filhos do container serão automaticamente **flex-items**

DISPLAY: FLEX

Eixo no flexbox

segunda distinção mais fundamental do flexbox. (flex-direction)

DISPLAY: FLEX

```html
<div class="container">
  <div class="item">Item 1</div>
  <div class="item">Item 2</div>
  <div class="item">Item 3</div>
</div>

<style>
  .container {
    display: flex;
    flex-direction: column; /* default: row*/
  }
</style>
```

O eixo será de cima para baixo

PROPRIEDADES DO CONTAINER

flex-direction

- Row
- Column
- row-reverse
- column-reverse

justify-content

- flex-start
- flex-end
- Center
- space-between
- space-around
- space-evenly

align-items

- flex-start
- flex-end
- Center
- Stretch
- baseline

PROPRIEDADES DO CONTAINER

flex-wrap

- Nowrap
- Wrap
- wrap-reverse

gap

- espaçamento entre items

PROPRIEDADES DOS ITENS

flex-direction

capacidade de "crescimento" do item

flex-shrink

capacidade de "encolhimento" do item

flex-basis

Tamanho inicial do item antes de crescer ou encolher

order

ordem visual dos itens


VAMOS VER NA
PRÁTICA O
FLEX

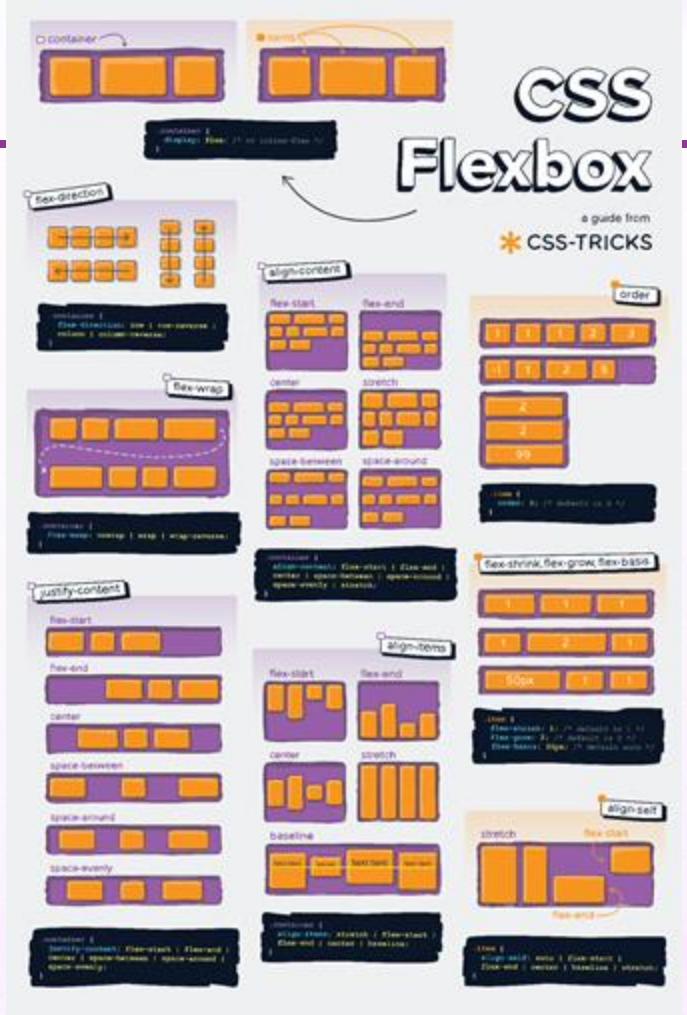
FLEX NA PRÁTICA

Um dos melhores recursos escritos
é o guia do CSS Tricks:

https://css-tricks.com/snippets/css/a-guide-to-flexbox/

FLEX NA PRÁTICA

Cheat-sheet do CSS Tricks



https://css-tricks.com/wp-content/uploads/2022/02/css-flexbox-poster.png

FLEX NA PRÁTICA

Flexbox exige muita prática. Uma dica:

https://flexboxfroggy.com/


DISPLAY: GRID

DISPLAY: FLEX

grid

o uso do grid é organização de itens em duas dimensões (grade)

Uma boa analogia é a ideia de um tabuleiro.

DISPLAY: GRID

Container e item

assim como no flexbox, essa distinção é fundamental.

Primeiro você deverá possuir um **grid container**.

Os filhos diretos do container são **grid items**.

DISPLAY: GRID

```
<div class="grid-container">
  <div class="grid-item">1</div>
  <div class="grid-item">2</div>
  <div class="grid-item">3</div>
  <div class="grid-item">4</div>
</div>

<style>
  .container {
    display: grid;
  }
</style>
```

grid container

DISPLAY: FLEX

Estrutura do grid

primeiro é importante definir a estrutura do grid (quantas colunas, quantas linhas?)

grid-template-columns: define a largura das colunas (exemplo: **100px 1fr 2fr;**)

grid-template-rows: define a altura das linhas (exemplo: **auto 200px repeat(2, 100px)**;)

gap: espaçamento

DISPLAY: FLEX

Posicionamento básico

Os itens se posicionam automaticamente por padrão

grid-column: posicionamento em relação às colunas

grid-row - posicionamento em relação às linhas

FLEX NA PRÁTICA

Um dos melhores recursos
escritos é o guia do CSS Tricks:

https://css-tricks.com/snippets/css/complete-guide-grid/

FLEX NA PRÁTICA

Grid também exige muita prática. Uma dica:

https://cssgridgarden.com/

VAMOS VER NA PRÁTICA O GRID

DESIGN RESPONSIVO E MEDIA QUERIES

DESIGN RESPONSIVO E MEDIA QUERIES

Media Queries

Quase sempre um website deve se comportar diferente dependendo do tamanho da tela (mobile vs desktop, por exemplo)

@media (max-width: 1000px){};

Pode-se utilizar tanto *max-width* como *min-width*

VAMOS DEIXAR O CÓDIGO ANTERIOR RESPONSIVO PARA CELULARES:

Vamos **avaliar** o encontro?

CSAT - 0062_TEC_Grupo Boticário_01