afecarvalho@ipt.pt

CSS Flexbox

Layout de elementos

Flexbox

- *Flexbox* é um modelo de layout de conteúdos *uni-dimensional*.
- Permite:
  - Dispor elementos em linhas ou colunas num contentor;
  - Controlar o tamanho (relativo ou absoluto) de cada elemento:
    - Tamanho inicial;
    - Proporção de tamanhos;
    - Prioridades para que os elementos “estiquem” ou “encolham” dependendo do espaço;
    - Distribuição dos conteúdos.
  - Controlar a ordem dos elementos, independentemente da ordem no HTML;
  - Facilitar o alinhamento dos conteúdos no contentor (ex: centrar verticalmente uma div noutra).

Flexbox – Tipos de elementos

- Existem dois tipos de elementos em flexbox:
- Container: Um container *flex* engloba e distribui os seus elementos-filho.
  - Para definir um container *flex*, usa-se a propriedade display: **display: flex;**
  - **Sem display: flex; nada do que vem a seguir funciona!!**
- Filho: O filho de um container *flex* é todo e qualquer elemento que é um descendente *direto* de um elemento que tem **display: flex;**.
  - Os elementos-filho não precisam de nenhuma propriedade especial para poderem usar flexbox, desde que estejam dentro de um container *flex*.

Flexbox – Tipos de propriedades

Propriedades do contentor:

- Afetam o layout dos conteúdos presentes nele:
  - Direção;
  - Distribuição nos dois eixos;
  - *Wrapping* (“mudança de linha ou coluna”).

Propriedades do elemento “filho”

- Permitem controlar aspetos relativos a cada elemento contido num container flex:
  - Ordem (relativamente aos outros);
  - Tamanho (incluindo a possibilidade de “esticar” ou “encolher” com base no espaço);
  - Alinhamento no eixo secundário.

afecarvalho@ipt.pt

Flexbox

Conceitos e propriedades do **container**

(Aquele que fica com display: flex)

Eixos

- Flexbox define dois eixos no container:
  - Principal: Define o eixo pelo qual os elementos serão distribuídos.
  - Secundário: Define o eixo perpendicular ao principal.
- O eixo **principal** é definido através da propriedade **flex-direction** no **container**:
  - **flex-direction: column**: O eixo principal toma uma direção **vertical** (regra geral, de cima para baixo).
  - **flex-direction: row**: O eixo principal toma uma direção **horizontal, definida pela direção do texto.**
- **O eixo secundário depende sempre do principal!**

afecarvalho@ipt.pt

flex-direction – Direção horizontal do eixo principal

- **flex-direction: row;**
- Os elementos são dispostos segundo **linhas**.
- Cada filho (a, b, c) comporta-se como fosse "display: inline";
- Valor por defeito nos browsers!
- Dependente da linguagem!
- **Eixo secundário: vertical**

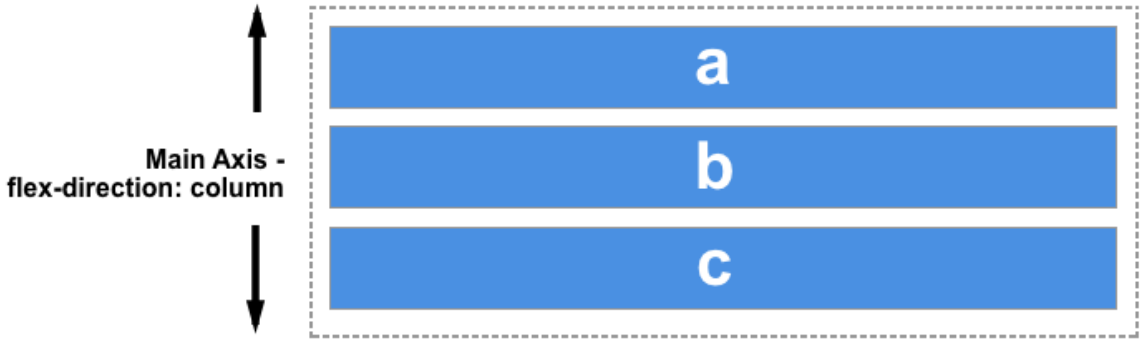
Numa linguagem esquerda-direita (ex: Português)

Numa linguagem direita-esquerda (ex: Árabe)

afecarvalho@ipt.pt

flex-direction – Direção vertical do eixo principal

- **flex-direction: column;**
- Os elementos são dispostos verticalmente, segundo uma **coluna**;
- Cada filho (a, b, c) comporta-se como fosse "display: block";
- **Eixo secundário: horizontal**



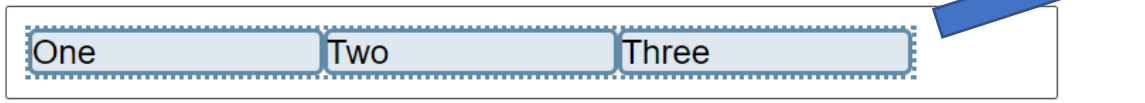
flex-direction – Inversão da ordem no eixo principal

- Flexbox permite *inverter* a ordem dos elementos, isto é, os elementos começam a ser dispostos **a partir do fim do eixo principal**:
  - Controlado novamente através do "**flex-direction**" no **container**.
  - **row-reverse**: Elementos dispostos em linha, mas por ordem inversa à do texto (ao contrário de **row**).
  - **column-reverse**: Elementos dispostos em coluna, mas de baixo para cima (ao contrário de **column**).
- **O uso de row-reverse e column-reverse não afeta a distribuição dos elementos no eixo secundário!**

flex-wrap – Disposição de elementos no eixo secundário

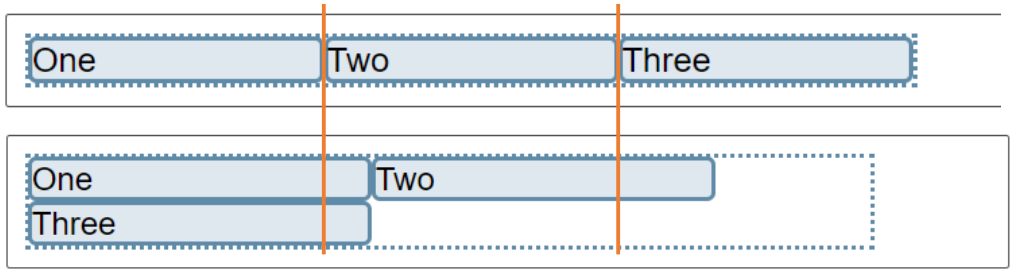
- O Flexbox, **por defeito**, distribui os elementos sobre **uma única linha/coluna**, (depende do eixo principal) mesmo que signifique que os elementos tenham que ser "encolhidos"!
- Este comportamento pode ser substituído através da propriedade "**flex-wrap**":
  - "**flex-wrap: nowrap**" é o comportamento por defeito, não há "mudanças de linha/coluna". Os elementos "encolhem" para caberem todos.
  - "**flex-wrap: wrap**" permite que os elementos se distribuam também pelo eixo secundário. Os elementos não encolhem (exceto quando só existe 1 elemento por linha/coluna do eixo principal e já não há espaço).

flex-wrap: Comparação




```
.box {
  display: flex;
  flex-wrap: nowrap;
}
```

```
<div class="box">
  <div>One</div>
  <div>Two</div>
  <div>Three</div>
</div>
```




```
.box {
  display: flex;
  flex-wrap: wrap;
}
```

```
<div class="box">
  <div>One</div>
  <div>Two</div>
  <div>Three</div>
</div>
```

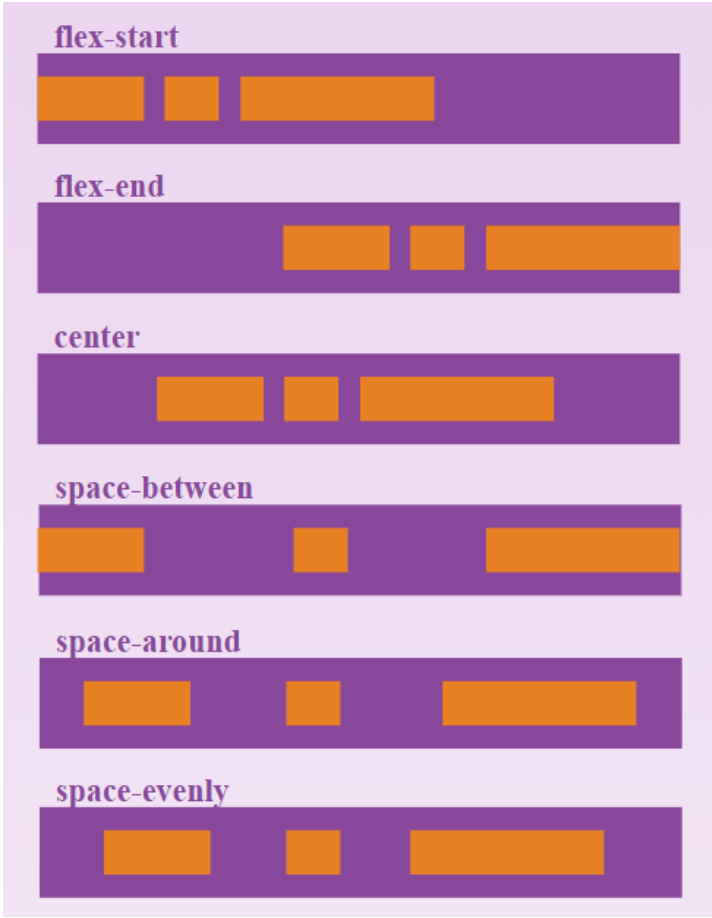
- **Sem** flex-wrap (default).
- Os elementos são dispostos só numa linha.
- Neste caso eles encolhem também para caberem todos (comparar com imagem à direita)

- **Com** flex-wrap.
- Os elementos são dispostos por quantas linhas forem precisas.
- Existe menor prioridade em encolher os elementos.

afecarvalho@ipt.pt

justify-content: Disposição de elementos no eixo principal

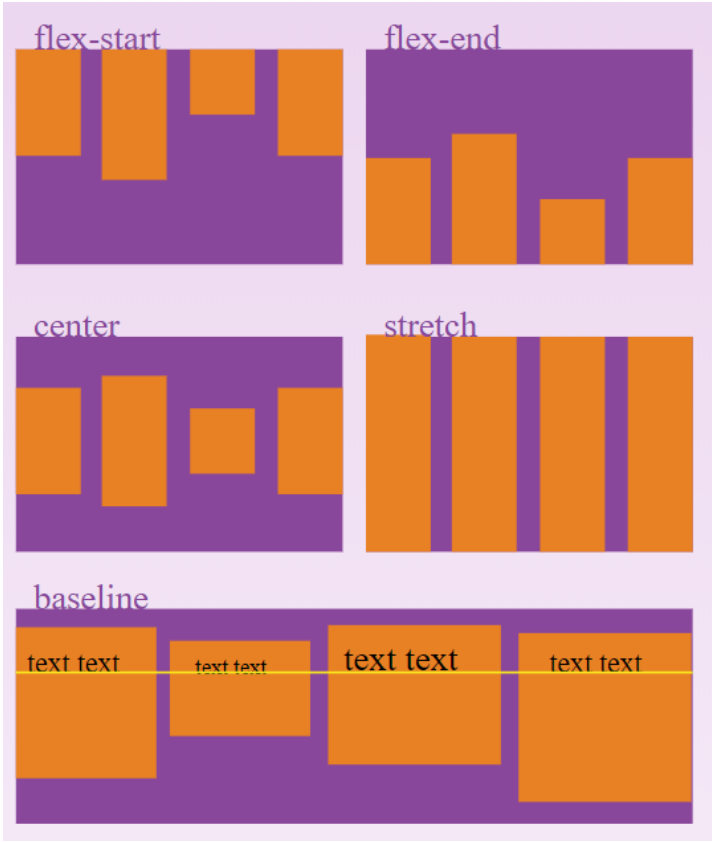
- O **justify-content** permite distribuir elementos segundo cada linha/coluna **do eixo principal**, desde alinhá-los ao início, centro, ou fim do eixo, a distribuí-los com espaço equitativo entre eles.
- A imagem à direita mostra os possíveis valores num **container** com **flex-direction: row**.



afecarvalho@ipt.pt

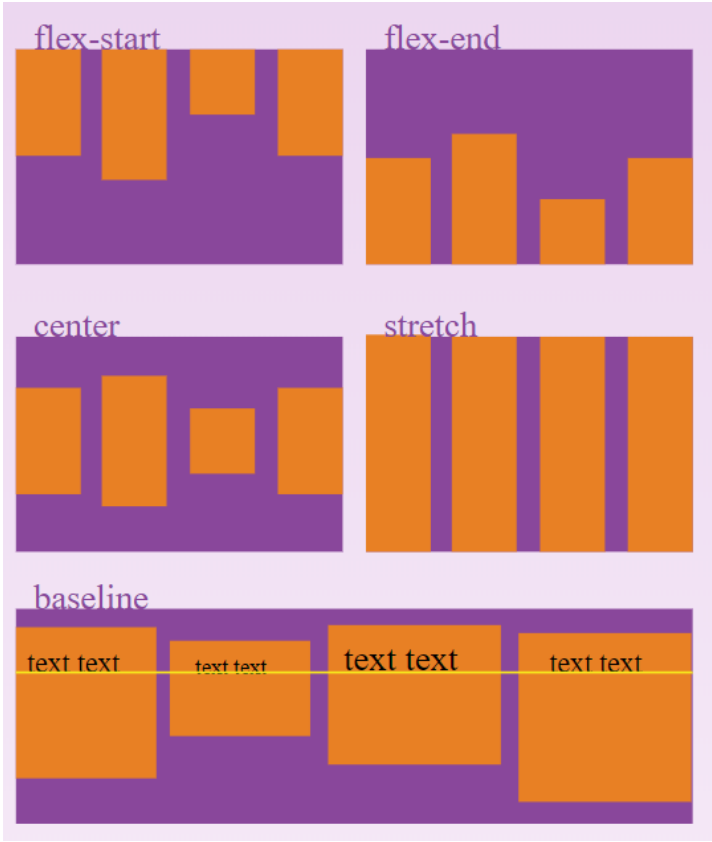
align-items: Disposição de elementos no eixo secundário

- O **align-items** permite alinhar elementos segundo cada linha/coluna **do eixo secundário**, como ao início, centro, ou fim do eixo, esticá-los, ou com base nas linhas de texto (**baseline**)
- A imagem à direita mostra os possíveis valores num **container** com **flex-direction: row**. (Eixo secundário: vertical, cima-baixo).



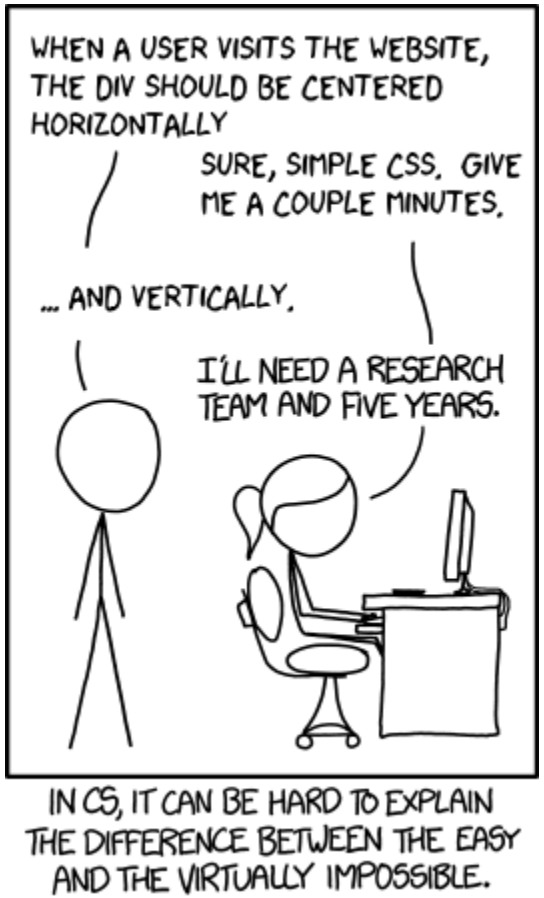
align-items: Disposição de elementos no eixo secundário

- Valor por defeito: **stretch** (os elementos “esticam-se” para preencher o espaço.
- Especialmente útil para alinhar elementos ao centro (verticalmente).



Alinhamento vertical em CSS antes de flexbox

Soluções:
margin: auto
OU
justify-content: center
align-items: center
(Depende do use case)



Fonte: https://i.imgur.com/2sY5JFr.png;
Original: https://xkcd.com/1425/

afecarvalho@ipt.pt

Flexbox

Conceitos e propriedades dos filhos

(Descendentes diretos de um elemento com display: flex)

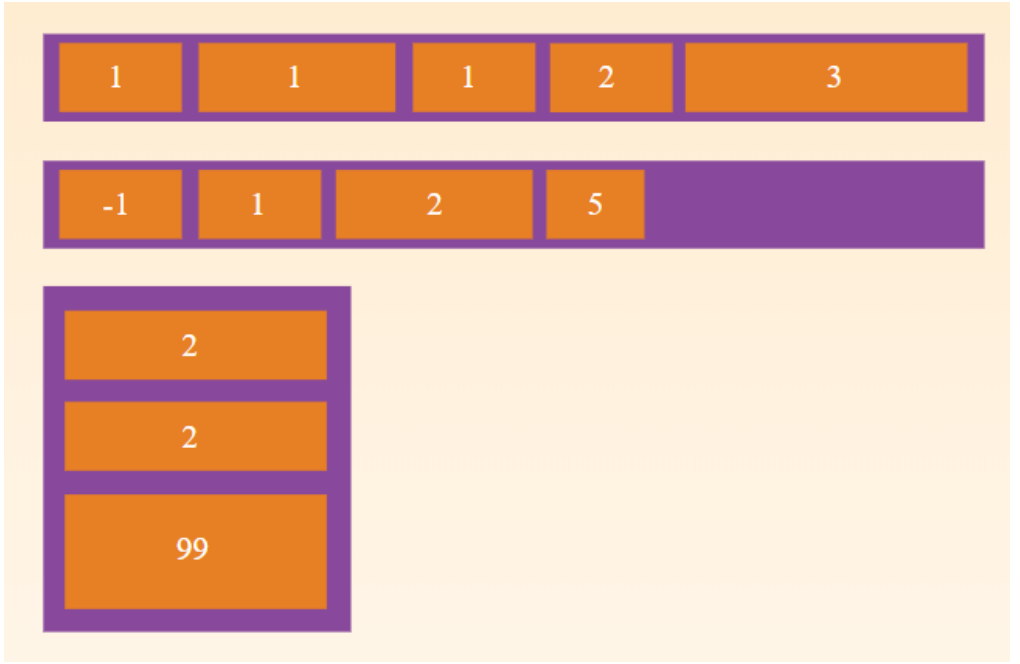
order: Ordenação de elementos

- O "**order**" é uma propriedade que recebe um valor **inteiro** (zero, negativo ou positivo) e permite alterar a ordem de um elemento.
- **Todos os elementos começam com order: 0**.
  - Dois elementos com o mesmo nº de ordem são dispostos pela **ordem no documento**.
- Valores < 0 (ex: **order: -1**) fazem com que o elemento fique **antes de outros com valores superiores**.
- Valores > 0 (ex: **order: 1**) fazem com que o elemento fique **depois de outros com valores inferiores**.

afecarvalho@ipt.pt

order: Ordenação de elementos

- O **order** é útil quando nós queremos que um elemento em particular fique antes ou depois de outros, **mesmo quando não conseguimos controlar o HTML que o colocou lá**.
- A imagem à direita mostra vários exemplos de **order**. O nº em cada caixa indica o valor do order.



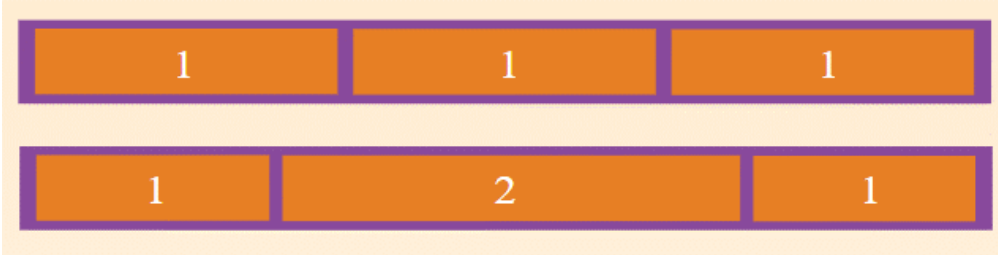
flex-grow: Capacidade de um elemento "esticar"

- O **flex-grow** é uma propriedade de cada elemento que recebe um valor **inteiro e maior ou igual a zero** que serve como **proporção** de crescimento.
- A proporção serve para "preencher" o espaço vazio.
- A proporção é definida com base no **espaço restante** no container:

  "Espaço restante" é o espaço **não ocupado** pelos elementos **antes** de se ter em conta as suas propriedades de **flex-grow** e **flex-shrink**.
- Valor por defeito: 0.
  - Faz com que o elemento **não "estique"** para preencher o espaço livre!

flex-grow: Capacidade de um elemento "esticar"

- **A proporção de espaço restante é adicionada a cada elemento!**
  - **Dois elementos com o mesmo flex-grow não terão a mesma dimensão se tiverem dimensão inicial (ex: width ou conteúdo diferente)!**
- A imagem à direita mostra um exemplo com 3 elementos, e dois diferentes casos de **flex-grow**.



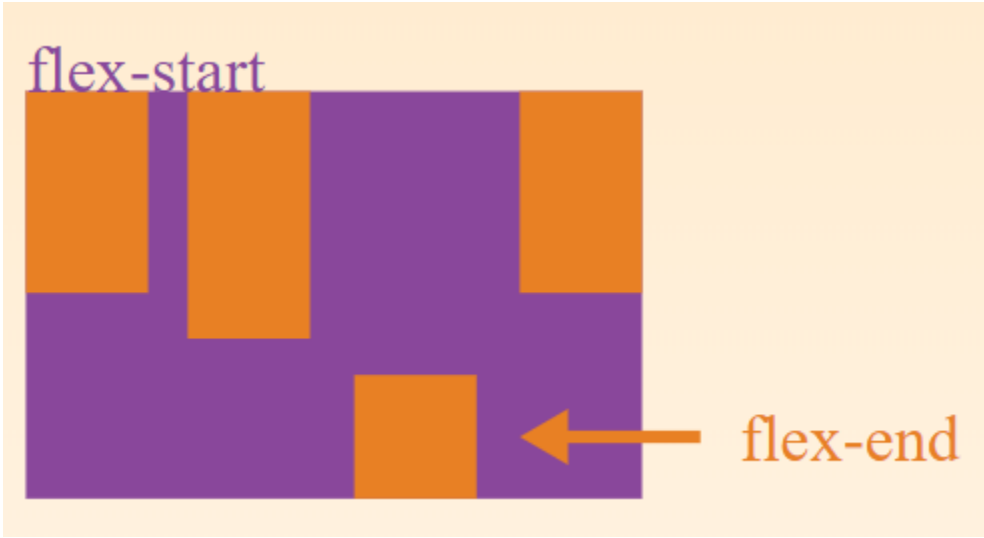
flex-shrink: Capacidade de um elemento “encolher”

- Quando não há espaço suficiente, o container tentará **encolher** os seus conteúdos de forma a que caibam todos **numa só linha/coluna do eixo principal** (ver: **flex-wrap**).
- O **flex-shrink**, tal como o **flex-grow**, recebe um valor **inteiro maior ou igual a zero**, e define a **proporção de “encolhimento”** de um elemento.
- Por defeito, este valor é **1**. (Todos os elementos podem encolher, e encolhem “por igual”).
- **Um valor de 0 impede com que o elemento encolha.**

afecarvalho@ipt.pt

align-self: Alinhamento de um elemento no eixo secundário

- O **align-self** pode ser usado para **sobrepor** o valor do **align-items** (ver slides anteriores) **de um único elemento**.
- Exemplo: elementos dispostos sob uma coluna, centrados (**align-items: center**), mas um deles precisa de se esticar para preencher a largura (**align-self: stretch**)



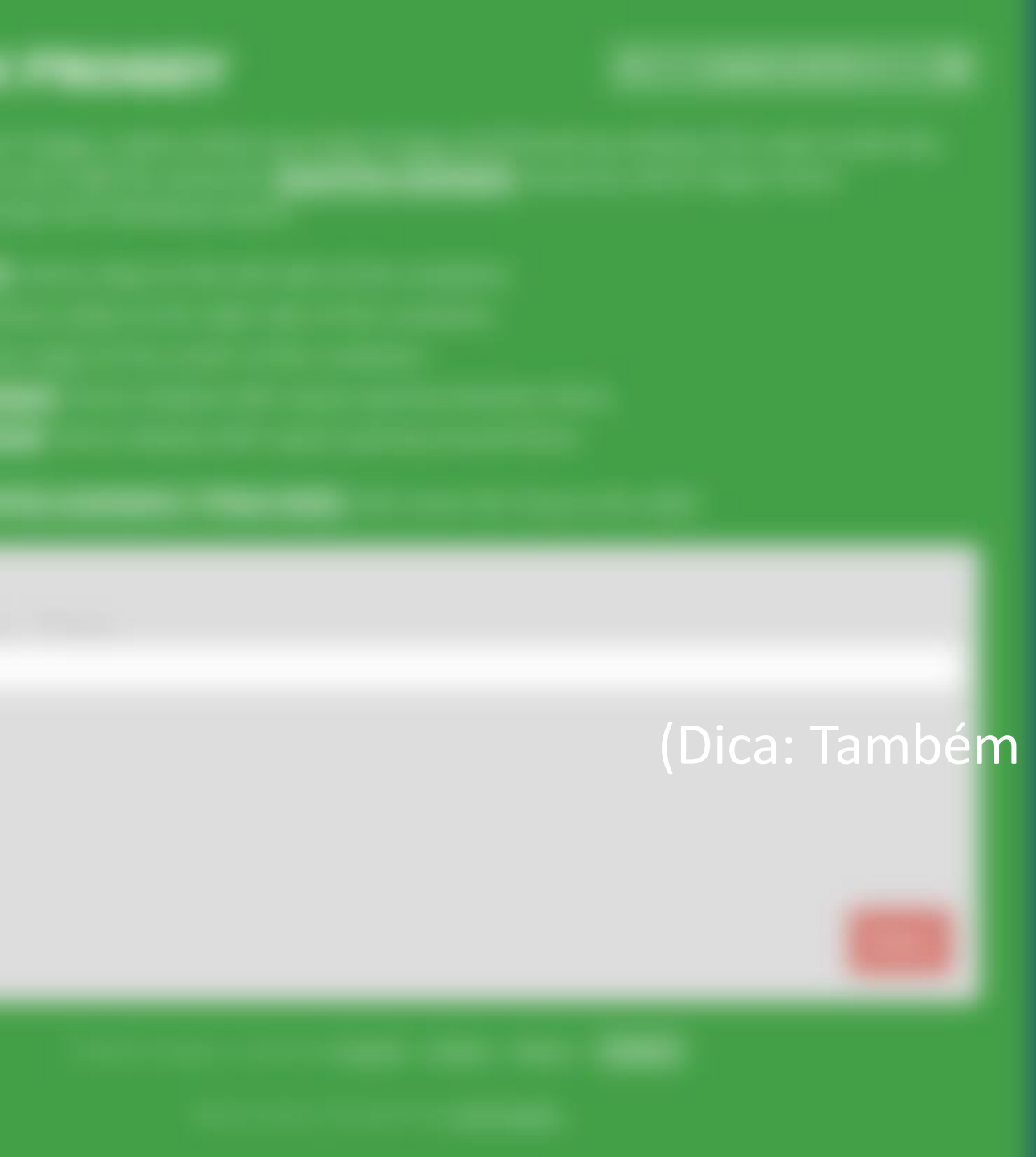
Elementos-filho: Notas

- Os valores de width e height continuam a poder ser usados para definir as dimensões de cada elemento.
- Quando se trabalha com flexbox, os conceitos de floats e vertical-align perdem significado (são ignorados).
- O uso de margens e posicionamento pode ser usado para afinar a posição de cada elemento-filho.
- **Elementos filho podem ter display: flex! Layouts flex podem ser colocados uns dentro dos outros!**

afecarvalho@ipt.pt

TPC: *Flexbox Froggy*

- *Flexbox Froggy* é um jogo que mostra como o Flexbox funciona.
- O objetivo do jogo é fazer com que os sapos fiquem em cima dos seus nenúfares, através do uso de Flexbox.
- Tem 20+ níveis, e faz uso dos conteúdos destes slides (e mais!).



Link

http://flexboxfroggy.com/

(Dica: Também há um de CSS Grid na mesma página!)

Material de estudo

- A complete guide to Flexbox: https://css-tricks.com/snippets/css/a-guide-to-flexbox/
- Basic concepts of Flexbox: https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Flexible_Box_Layout/Basic_Concepts_of_Flexbox

(As imagens dos slides anteriores foram retiradas destes dois links)

Imagem: https://desket.co/products/css-is-awesome-mug

Obrigado!

André Carvalho

Email: afecarvalho@ipt.pt