

# TRYBE

## Modulo II – Front-end

### Bloco 6 – HTML, CSS Flexbox, Bibliotecas JS

Refêrencia completa para HTML, CSS e JS: <https://www.w3schools.com/> .

#### 1) HTML&CSS - FORMS

Criar formulários em HTML com as tags: input, button, textarea, select, form.

##### FORM

<form action="" method="">

Method sendo get ou post.

Para campos destacados:

<fieldset>

<legend>

</fieldset>

##### INPUTS

Types of inputs:

<input/>

Lista completa [https://www.w3schools.com/html/html\\_form\\_input\\_types.asp](https://www.w3schools.com/html/html_form_input_types.asp)

type="text" sendo o mais comum

type="checkbox" para deixar selecionar mais do que uma opção

type="radio" para deixar selecionar apenas uma opção

type="file" para enviar arquivo no form

type="password" para esconder input com \*

Attributes:

Lista completa [https://www.w3schools.com/html/html\\_form\\_attributes.asp](https://www.w3schools.com/html/html_form_attributes.asp)

Ex: readonly, disabled, size, maxlength, min, max, multiple, pattern, placeholder, required="required", step, autofocus, height, width, list, autocomplete.

Atributos mais comuns dos inputs:

type, placeholder, required, class, value, id, name.

##### LABELS

Título dos campos a serem preenchidos.

<label for="idcampo">

permite todo clicavel

##### TEXTAREA

Um input de maior tamanho.

`<textarea>` com mesmos atributos do que input e também `cols` e `rows` para determinar o tamanho via nº de colunas e linhas.

## DROPDOWN LIST

Campo de seleção.

```
<select>
<option1>
<option2>
</select>
```

Nessas tags podem ser introduzidas os atributos de sempre.

## BUTTON

```
<button>Texto desejado</button>
```

`type="submit"` para enviar no button final

Tem todos os mesmos atributos dos inputs.

## LEGENDS

```
<small>
```

Recursos adicionais completos:

[https://www.w3schools.com/howto/howto\\_css\\_register\\_form.asp](https://www.w3schools.com/howto/howto_css_register_form.asp)

<https://www.freecodecamp.org/news/a-step-by-step-guide-to-getting-started-with-html-forms-7f77ae4522b5/>

Dicas diversas a partir do exercício

**prevent.Default()** <https://developer.mozilla.org/pt-BR/docs/Web/API/Event/preventDefault>

Não esquecer do `window.onload` para por ex. validar a chamada de function

---

## 2) Bibliotecas JS e frameworks CSS

### Objetivo geral

Entender e incluir esses pacotes de códigos prontos reutilizáveis nos nossos projetos.

## Frameworks CSS

### Exemplos

Bulma, Bootstrap, Semantic UI, Materialize

### Como integrar framework

Integrar no head html:

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
<link href="css/yourpathtodownloadedframework" rel="stylesheet" type="text/css">
```

```
<script type="text/javascript" src="pathtoit"></script>
```

### Como usar framework para estilizar

Dar para os elementos html as class da documentação do framework específico.

## Bibliotecas JS

### Opções de uso

Usar biblioteca JS a partir do seu **próprio servidor**  
VS  
incluir-la a partir de uma **CDN externa**.

(= "**Content Delivery Network**", servidor especializado para servir arquivos estáticos (como bibliotecas JS) rapidamente. Vantagens do CDN: rapidez do carregamento, segurança.)

**URL relativa** que se refere a uma pasta interna  
VS

**URL absoluta**

com protocolo httpS, dica de usar mecanismo de segurança CSP para garantir confiabilidade dos servidores de onde traz recursos (como bootstrapcdn.com, rawgit.com, googleapis.com, jsdelivr.net e cdnjs.com.)

### Diferentes tipos e usos de bibliotecas

- para compatibilidade de navegadores: Prototype, Mobtools, YUI, jQuery
- para linguagem mais agradável: underscore (trabalhar com vetores)
- para consertar problemas na linguagem: moment (trabalhar com datas)
- outros diversos: cheet (criar easter eggs).

### Como integrar biblioteca

Integrar biblioteca js na tag script.

Exemplo:

```
<script src='https://cdn.com/fortune.js'></script>
<script>
var fortune = fortune.teller();
</script>
```

### Ordem de inclusão:

- a biblioteca antes do meu código
- as bibliotecas por ordem de hierarquia (por exemplo, jQuery acima de bootstrap).

Uma página web é composta pela **interface** do usuário (HTML e CSS), pela **interatividade** (JS + DOM) e pelos **dados** (os quais normalmente obtemos usando JS).

## Dicas diversas

Excluir as bibliotecas das análises do code climate

—

Lista longa de bibliotecas e frameworks:

<https://pt.khanacademy.org/computing/computer-programming/html-css-js/using-js-libraries-in-your-webpage/a/the-world-of-js-libraries>

Critérios de escolha: considerar qualidade da experiência do desenvolvedor e do usuário.

Atalho VsCode: CTRL+D para selecionar todas tags com mesmo nome.

---

### 3) 4) CSS Flexbox (Part 1 & Part 2)

#### Uso

Para organizar elementos dentro de um container pai de jeito **flexível** e assim, **responsivo**.

#### Primeira etapa

Indicar e possibilitar o uso do flexbox em um elemento, que seja o container pai ou o item, com o *display: flex*;

```
.flex-container {  
  display: flex;  
}
```

#### Flex axes

Pode ser **horizontal**, quando flex-direction é row ou row-reverse, e **vertical**, quando é column ou column-reverse. Por padrão é o row.

#### Flex lines

Dependendo da propriedade flex-wrap, pode ser **single** ou **multi line**.

### Propriedades para o elemento pai: flex container

**Flex-direction.** Esta propriedade CSS define a direção em que os itens são posicionados no container e aceita os seguintes valores:

*row*: Itens são posicionados na mesma direção do texto.

*row-reverse*: Itens são posicionados na direção oposta à do texto.

*column*: Itens são posicionados de cima para baixo.

*column-reverse*: Itens são posicionados de baixo para cima.

**Flex-wrap** aceita os seguintes valores:

*nowrap*: Todos os itens são apertados em uma única linha.

*wrap*: Itens se separam em linhas adicionais.

*wrap-reverse*: Itens se separam em linhas adicionais em reverso.

**Flex-flow:** As duas propriedades `flex-direction` e `flex-wrap` são usadas tão frequentemente juntas que uma propriedade abreviada `flex-flow` foi criada para combiná-las. Essa propriedade aceita o valor das duas propriedades separados por um espaço.

Por exemplo, você pode usar `flex-flow: row wrap` para aplicar a direção de linha e quebrar em múltiplas linhas.

### **Justify-content:**

*flex-start:* Itens se alinham à esquerda do container.

*flex-end:* Itens se alinham à direita do container.

*center:* Itens se alinham no centro do container.

*space-between:* Itens se alinham com distância igual entre eles.

*space-around:* Itens se alinham com distância igual em torno deles.

**Align-items.** Essa propriedade CSS alinha os itens verticalmente e aceita os seguintes valores:

*flex-start:* Itens se alinham na parte de cima do container.

*flex-end:* Itens se alinham na parte de baixo do container.

*center:* Itens se alinham no centro vertical do container.

*baseline:* Itens se alinham na linha da base do container.

*stretch:* Itens se esticam para preencher o container. É seu padrão.

**Align-content** para definir como múltiplas linhas devem ser espaçadas uma das outras. Valores:

*flex-start:* Linhas são agrupadas no topo do container.

*flex-end:* Linhas são agrupadas no fundo do container.

*center:* Linhas são agrupadas no centro vertical do container.

*space-between:* Linhas são posicionadas com espaço igual entre elas.

*space-around:* Linhas são posicionadas com espaço igual em torno delas.

*stretch:* Linhas se esticam para preencher o container.

**align-content determina o espaçamento entre linhas, enquanto align-items determina como as linhas como um todo são alinhadas dentro do container.** Quando há só uma linha, `align-content` não tem nenhum efeito.

## **Propriedades para o elemento filho: flex item**

### **Order**

Às vezes, reverter a ordem de uma coluna ou de um container não é suficiente. Nesses casos, podemos aplicar a propriedade `order` para itens individuais. Por padrão, itens tem um valor de 0, mas nós podemos usar essa propriedade para alterar para um *valor inteiro positivo ou negativo*.

```
#pond {  
  display: flex;  
}  
  
.yellow {  
  order: 1;  
}
```

**Align-self** alinha um item ao longo do eixo cruzado, substituindo o valor de `align-items`.

Aceita os valores: *flex-start*, *flex-end*, *center*, *baseline*, *stretch* (os mesmos valores que `align-items`) e também *auto*, *initial*.

Anotar bem que *float*, *clear* e *vertical-align* não têm efeito sobre um item flex.

### **Flex-grow**

Habilidade máxima para um flex item de crescer. Valores: *n°* (>0, 1 sendo o mesmo tamanho para todos itens do container, 2 3 indo aumentando o tamanho) e *initial*, *auto*, *inherit*.

## Flex-shrink

Habilidade máxima para um flex item de encolher. Valores:  $n^{\circ}$  ( $>0$ , 1 sendo o mesmo tamanho para todos itens do container, 2 3 etc indo diminuindo o tamanho) e *initial*, *auto*, *inherit*.

## Flex-basis

Define o tamanho padrão de um item antes que o resto do espaço seja distribuído.

Valor pode ser um tamanho (*%*, *rem*, *px* etc) o keyword como *auto* (que olha para tamanho definido), *initial*, *inherit*, *content*, *max-content*, *min-content*, *fit-content*.

Diferença entre 0 e auto: se for 0, o espaço extra não é redistribuído.

## Flex

flex-grow, flex-shrink and flex-basis combinados, nesta ordem.

Syntax: pode usar com um, dois ou três valores.

Padrão é *0 1 auto*.

```
/* Keyword values */
flex: auto;
flex: initial;
flex: none;

/* One value, unitless number: flex-grow */
flex: 2;

/* One value, width/height: flex-basis */
flex: 10em;
flex: 30%;
flex: min-content;

/* Two values: flex-grow | flex-basis */
flex: 1 30px;

/* Two values: flex-grow | flex-shrink */
flex: 2 2;

/* Three values: flex-grow | flex-shrink | flex-basis */
flex: 2 2 10%;

/* Global values */
flex: inherit;
flex: initial;
flex: unset;
```

## Margin

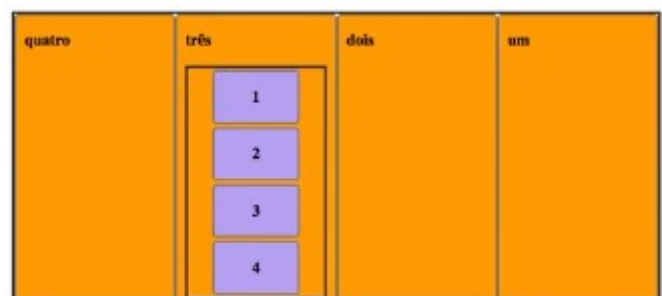
Pode aplicar no container pai e no elemento filho normal.

## Subcontainers e subitens

É possível criar **containers dentro de containers**, para fazer agrupamento e para facilitar a organização da página.

Atenção particular para:

- filho que herde propriedades do pai;
- altura, sendo soma dos filhos que vai se tornar a do pai.



## Dicas diversas

### **Para ver a estrutura do seu code:**

\* pode usar a Outline Chrome Extension

\*ou no CSS dar um border (1px solid red por exemplo) para o \* {}.

—

Note que quando você define a direção para uma linha ou coluna reversa, start e end também são reversos.

—

**CSS tem margens padrão** por elementos, pensar nisso quando seu layout não está como queria.

—

Dar atributos CSS para children de elementos:

o seletor **:nth-child(i)** onde i começa com 1 (equivalente a first-child)

**Referência:** [https://www.w3schools.com/cssref/selector\\_nth-child.asp](https://www.w3schools.com/cssref/selector_nth-child.asp)

**Exemplo:**



```
▼ .flex-container:nth-child(2) >
  .box:first-child {
    padding-top: 40px;
  }
```

```
header li:last-child {
}
```

—

Unidade no CSS para ajustar tamanhos relativamente ao navegador:

vh = viewport height

vw = viewport width

—

*Diferença entre initial e auto:* initial é o padrão da propriedade, auto é calculado com outros fatores.

*Diferença entre initial e inherit:*

“initial-Sets this property to its default value.

inherit-Inherits this property from its parent element.”

--

**Sites para visualizar** as propriedades css flexbox de maneira interativa:

<https://codepen.io/enxaneta/full/adLPwv>

<https://flexboxfroggy.com/#pt-br>

<https://the-echoplex.net/flexyboxes/>

<http://www.flexboxdefense.com/>

<https://demos.scotch.io/visual-guide-to-css3-flexbox-flexbox-playground/demos/>

## Content completo:

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

<https://origamid.com/projetos/flexbox-guia-completo/>

<https://developer.mozilla.org/en-US/docs/Glossary/Flexbox>

Com tricks <https://devhints.io/css-flexbox> .

---

## 5) CSS Responsivo – mobile first

Vantagem da versão desktop primeiro: pior caso, será o único design da página;

Benefícios de **codar o CSS focado para o mobile primeiro**: tempo de carregamento, menos linhas.

### Media queries

#### Sintax base

```
@media (max-width: 600px) {  
  elementos e estilos escritos como no css  
}
```

#### Exemplos de palavras chave para condicionar o media

*only* previne que navegadores antigos que não suportam media queries com media features de aplicar os estilos dados;

#### Media types:

*screen* aplica a condição que aquele css vai funcionar apenas para smartphone, tablet ou desktop, *print* aplicado apenas em momentos de impressões.

Value	Description
all	Default. Used for all media type devices
print	Used for printers
screen	Used for computer screens, tablets, smart-phones etc.
speech	Used for screenreaders that "reads" the page out loud

#### Media features:

Além de max e min-width, ver lista [https://www.w3schools.com/cssref/css3\\_pr\\_mediaquery.asp](https://www.w3schools.com/cssref/css3_pr_mediaquery.asp) .

*@media (orientation:portrait)* ou *(orientation:landscape)* para prever estilos quando a orientação da página mudar para vertical ou horizontal.

#### Operadores lógicos:

*and*

, equivalente ao operador or

#### Todas syntax completas

[https://developer.mozilla.org/pt-BR/docs/Web/Guide/CSS/CSS\\_Media\\_queries](https://developer.mozilla.org/pt-BR/docs/Web/Guide/CSS/CSS_Media_queries)



## Onde escrever

Dica: escrever os media queries no final do arquivo CSS, segundo regras de prioridade do CSS. Saber que todo o que for fora do media, quer dizer que se aplica para todos.

## Viewport

Estrutura no head do html importante aqui, garantir que tem:

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

## Breakpoints: tamanhos padrão de devices

320-420px (mobile)

>768px (tablet vertical)


>1280px (desktop / tablet horizontal)

<https://www.freecodecamp.org/news/the-100-correct-way-to-do-css-breakpoints-88d6a5ba1862/>

## Dicas diversas:

*height: 100%* não funciona

—

Visualizar no inspect mudando o tamanho da tela manualmente ou apertando 

—

Para tirar *position:fixed* : voltar para o valor original padrão com *position:static*;  
( [https://www.w3schools.com/css/css\\_positioning.asp](https://www.w3schools.com/css/css_positioning.asp) )

—

“!important” para valorizar ordem de execução de algo no CSS.

—

para tirar um elemento no media, *display:none*;

—

Em certos casos, pode ser pertinente preferir unidades relativas.

—

Dicas de estilo gerais:

