Shift

Aula 02

HTML & CSS

Parte 2

O QUE O MATERIAL COBRE

1. TAGS HTML: DIVS
2. PROPRIEDADES CSS: LAYOUTS, GERAIS E UNID. DE MEDIDA
3. NORMALIZAÇÃO DE ESTILOS
4. TAGS HTML: SEMÂNTICA NO HTML

O QUE O MATERIAL COBRE

5 PROPRIEDADE CSS: FLEX

6 PROPRIEDADE CSS: GRID

7 SUGESTÕES DE EXERCÍCIOS

1

Tags HTML

Divs

Divs: estruturando o conteúdo no HTML

No HTML, **“div”** é uma abreviação para “division”, ou seja, divisão.

Quando falamos da tag ** ** você pode pensar em elementos genéricos, como se fossem containers ou caixas que podem armazenar qualquer tipo de conteúdo que você queira.

O conteúdo pode ser texto, imagens, links, ou algo maior como um header, um conteúdo lateral ou qualquer outra tag HTML ou estrutura que faça sentido na construção do conteúdo do site.

Divs: estruturando o conteúdo no HTML

Ao juntar as Divs com o CSS é possível criar estruturas de layout organizadas, com flexibilidade para funcionar em diferentes dispositivos e com design moderno.

Por padrão as divs são blocantes (no CSS “display: block”), ou seja, elas sempre ocupam o espaço inteiro referente ao seu elemento pai até que uma regra CSS personalizada diga ao contrário.

Divs: estruturando o conteúdo no HTML

```
<div>conteúdo</div>
```

Ao juntar as divs com o CSS é possível criar estruturas de layout organizadas, com flexibilidade para funcionar em diferentes dispositivos e com design moderno.

Por padrão as divs são blocantes (no CSS “display: block”), ou seja, elas sempre ocupam o espaço inteiro referente ao seu elemento pai até que uma regra CSS personalizada diga ao contrário.

2

Propriedades CSS

Layouts, Gerais e Unid. de Medidas

Propriedades CSS: width & height

```
<style>
  div {
    width: 100px;
    height: 100px;
  }
</style>

<div>conteúdo</div>
```

As divs sozinhas não conseguem fazer muitas coisas, mas juntando o poder do CSS nós podemos dar forma para elas.

Um exemplo é utilizando o **width** e **height** que definem largura e altura respectivamente.

Propriedades CSS: margin

```
<style>
  div {
    margin-top: 10px;
    margin-bottom: 10px;
    margin-right: 10px;
    margin-left: 10px;
  }
</style>

<div>conteúdo</div>
```

Para definir espaçamento externo ao redor da div (ou qualquer outro elemento) utilizamos a propriedade **margin**.

As margens possíveis são:

- top (cima)
- Bottom (baixo)
- Right (direita)
- Left (esquerda)

Propriedades CSS: padding

```
<style>
  div {
    padding-top: 10px;
    padding-bottom: 10px;
    padding-right: 10px;
    padding-left: 10px;
  }
</style>

<div>conteúdo</div>
```

Para definir espaçamento interno da div (ou qualquer outro elemento) utilizamos a propriedade **padding**.

Os paddings possíveis são:

- top (cima)
- Bottom (baixo)
- Right (direita)
- Left (esquerda)

margin & padding: outras maneiras de usar

```
div {
  /* top,bottom,right,left */
  /* Define o valor 10px para os quatro lados igualmente */
  margin: 10px;
  padding: 10px;

  /* top,bottom right,left */
  /* Define o valor 10px para top e bottom e 20px para right e left */
  margin: 10px 20px;
  padding: 10px 20px;

  /* top right,left bottom */
  /* Define o valor 10px para top, 25px para bottom e 20px para right e left */
  margin: 10px 20px 25px;
  padding: 10px 20px 25px;

  /* top right bottom left */
  /* Define o valor 10px para top, 25px para bottom, 20px para right e 30px left */
  margin: 10px 20px 25px 30px;
  padding: 10px 20px 25px 30px;
}
```

Propriedades CSS: box-sizing

```
<style>
  div {
    box-sizing: border-box;
  }
</style>

<div>conteúdo</div>
```

Você pode ter reparado que ao adicionar padding em uma div, o tamanho dela (width/height) cresceu junto.

Isso não é algo que buscamos ao criar layouts, porque pode gerar muita quebra e irá necessitar muitos calculos do nosso lado para resolver.

Felizmente o CSS tem a propriedade e valor **box-sizing: border-box** que garante que o tamanho total do elemento seja calculado de forma que o padding não influencie na largura e altura dos elementos.

Propriedades CSS: border

```
<style>
  div {
    border: 1px solid #000;

    /* Pode ser definido separadamente também */
    border-width: 1px;
    border-style: solid;
    border-color: #000;
  }
</style>

<div>conteúdo</div>
```

É possível e bem fácil definir bordas para seus elementos HTML. Utilizando a propriedade **border** e passando todos os valores de uma vez, ou passando item por item:

- border-width: espessura da borda;
- border-style: estilo da borda – seus valores são solid, dashed e outros;
- border-color: muito parecido com a cor de fonte, pode usar cores com palavra-chave, hexadecimal ou rgba

Propriedades CSS: border-radius

```
<style>
  div {
    border-radius: 8px;
  }
</style>

<div>conteúdo</div>
```

Para arredondar os cantos de um elemento pode ser utilizada a propriedade **border-radius**.

Tem uma forma de atribuição muito parecida com o margin/padding. Ela pode ser geral, ou separadamente para cada canto:

- top-left (topo esquerdo);
- top-right (topo direito);
- bottom-right (base direita);
- bottom-left (base esquerda);

border-radius: outras maneiras de usar

```
<style>
  div {
    border-radius: 8px;

    /* Pode definir os cantos separadamente de uma vez */
    /* topo-esquerda topo-direita base-direita base-esquerda */
    border-radius: 1px 2px 3px 4px;

    /* Pode ser definido separadamente também */
    border-top-left: 8px;
    border-top-right: 8px;
    border-bottom-right: 8px;
    border-bottom-left: 8px;
  }
</style>

<div>conteúdo</div>
```

Propriedades CSS: box-shadow

```
<style>
  div {
    box-shadow: 2px 2px 10px rgba(0,0,0,0.4);
  }
</style>

<div>conteúdo</div>
```

Para colocar sombra em algum elemento é possível utilizer a propriedade **box-shadow** passando como valores nessa ordem:

- valor eixo x
- valor eixo y
- valor do blur
- cor (normalmente em rgba)

Propriedades CSS: background-color

```
<style>
  div {
    background-color: #000000;
  }
</style>

<div>conteúdo</div>
```

Podemos colocar cores de fundo para os elementos, seja div ou qualquer outro. Para isso basta utilizar a propriedade **background-color**.

Como as outras propriedades de cor, pode ser utilizado palavras-chave, hexadecimal ou rgb/rgba.

Propriedades CSS: background-image e outros

```
<style>
  div {
    background-image: url('URL_DA_IMAGEM');
    background-repeat: no-repeat;
    background-position: center;
    background-size: cover;
  }
</style>

<div>conteúdo</div>
```

Além da tag <img /> também é possível utilizar imagens de fundo nos elementos, isso nos proporciona muita flexibilidade para receber imagens das quais não temos controle de largura e altura.

Propriedades CSS: background-image e outros

```
<style>
  div {
    background-image: url('URL_DA_IMAGEM');
    background-repeat: no-repeat;
    background-position: center;
    background-size: cover;
  }
</style>

<div>conteúdo</div>
```

Podemos usar uma combinação legal de propriedades:

- **background-image** – define a URL da imagem
- **background-repeat** – define a repetição da imagem no eixo x e y;
- **background-position** – define a posição da imagem, pode usar palavras-chave ou valores;
- **background-size** – pode ser utilizado palavras-chaves ou valores.

Unidades de medida: px, %, wh e vh

```
<style>
  div {
    width: 10px;
    width: 100%;
    width: 100wh;
    height: 100vh;
  }
</style>

<div>conteúdo</div>
```

Unidades de medida são muito uteis para os valores saberem quais tamanhos, distâncias e proporções tomarem. Elas são absolutas ou relavitas. Até agora apenas o píxel foi utilizado, mas existem outras unidades muito úteis:

- **px** – 1 pixel representa 1 ponto da tela do dispositivo;
- **%** – porcentagem relativa ao element pai, muito útil para responsividade;
- **wh** – porcentagem da largura da visualização da tela
- **vh** – porcentagem da altura da visualização da tela

Unidades de medida: rem

```
<style>
  html {
    font-size: 10px;
  }

  div {
    /* 1rem = 10px */
    width: 1rem;
    height: 1rem;
  }
</style>

<div>conteúdo</div>
```

Essa unidade é uma das mais utilizadas, e está na categoria de unidades relativas. Para você utilizar o **rem** é necessário definir o valor base no element html através da propriedade **font-size**.

No caso ao lado 1rem será equivalente a 10px, 2rem será 20px e assim por diante.

É muito útil porque isso gera um controle maior sobre todo o site, caso queira mudar os todos os tamanhos, apenas uma mudança no **root**, ou seja, no selector **html** e todo o site será atualizado.

Pseudo-seletores: :hover

```
<style>
  button {
    background-color: #ef4444;
    /* opcional, mas legal */
    transition: all 200ms linear;
  }

  button:hover {
    background-color: #f43f5e;
  }
</style>

<button>meu botão</button>
```

O CSS nos da seletores especiais que permitem estilizar partes específicas dos elementos, ou seja, partes que não estão especificadas no próprio HTML. Esses seletores são chamados **pseudo-seletores**.

Um dos mais famosos é o **:hover**, que altera o estilo do elemento ao passar o cursor em cima do mesmo.

A maioria das propriedades CSS podem ser alteradas.

Dica: para deixar a transição de propriedade com efeito e mais “smooth”, pode-se usar a propriedade transition.

Pseudo-seletores: :first-child, :last-child e :nth-child

```
<style>
  ul li {
    background-color: #0ea5e9;
  }

  ul li:first-child {
    background-color: #6366f1;
  }

  ul li:last-child {
    background-color: #f97316;
  }

  ul li:nth-child(2) {
    background-color: #84cc16;
  }
</style>

<ul>
  <li>item 1</li>
  <li>item 2</li>
  <li>item 3</li>
</ul>
```

Uma outra categoria de pseudo-seletores são os “child”, com eles podemos estilizar elementos específicos de um certo grupo de elementos, contando com:

- **:first-child** – estiliza o primeiro filho;
- **:last-child** – estiliza o último filho;
- **:nth-child(valor)** – estiliza o filho de acordo com o valor passado no parâmetro.

Dicionário de propriedades CSS

Aqui cobrimos as principais, porém existem muitas outras mais propriedades CSS que podem ser utilizadas para chegar no layout desejado para o seu site.

O CSS-Tricks, um grande site voltado para CSS disponibilizou um dicionário de propriedades bem completo, e você pode acessar nesse link: https://css-tricks.com/almanac/.

3

Normalização de **Estilos**

Benefícios de utilizar normalização

Existem diversos navegadores e dispositivos disponíveis no mundo todo e cada dia existem mais e mais atualizações. Hoje em dia temos mais padrões definidos entre os navegadores, mas cada um ainda possui suas peculiaridades e estilos padrões, por exemplo: o tamanho de um <h1>, o padding do body, entre outros elementos.

Ao utilizar a **normalização de estilos** da para garantir que os elementos HTML tenham uma mesma configuração padrão, resetando todos os principais estilos e mantendo os padrões acessíveis e sem surpresas durante o desenvolvimento.

Normalize.css

O normalize.css

O programador **Nicolas Gallagher** criou e disponibilizou de maneira open source o **normalize.css** que pode ser consumido via CDN utilizando a tag **<link />**, você pode acessar nesse link: https://necolas.github.io/normalize.css/.

Normalize.css

A modern, HTML5-ready alternative to CSS resets

Normalize.css makes browsers render all elements more consistently and in line with modern standards. It precisely targets only the styles that need normalizing.

Download v8.0.1

Chrome, Edge, Firefox ESR+, IE 10+, Safari 8+, Opera

See the CHANGELOG

```
npm install normalize.css
```

```
<link
    rel="stylesheet"
    href="https://necolas.github.io/normalize.css/8.0.1/normalize.css"
/>
```

4

Tags HTML

Semântica no HTML

O que é HTML semântico

Até agora vimos que a tag pode ser utilizada para criar diversos tipos de caixas que possibilitam criar layouts para nossos sites.

Na versão 5 o HTML criou novas tags que são muito parecidas com as divs, mas que tem mais significado principalmente para quem necessita de acessibilidade, mas também para um melhor ranqueamento em sistemas de busca.

Em vez de usar apenas usar divs o HTML semântico enfatiza o significado dos elementos criados no site através de várias tags e algumas principais que não pode faltar.

Lembrando que todos esses elementos vão dentro da tag <body>.

HTML semântico: <header>

<header></header>

Define o cabeçalho de uma seção ou de um documento. Pode ser utilizada mais de uma vez. Ela pode ser utilizada onde fica o logotipo e o navegador do site, e também por exemplo para criar uma lista de posts de um blog.

Foque em utilizar o header em informações introdutórias.

HTML semântico: <footer>

<footer></footer>

Define o rodapé de uma seção ou de um documento. Pode ser utilizada mais de uma vez. Ela pode ser utilizada onde fica o final do site, normalmente com “copyright” e também por exemplo para criar uma lista de posts de um blog, onde você coloca informações adicionais ao post, por exemplo “tempo de leitura”.

HTML semântico: <main>

<main></main>

Define o local onde ficará o conteúdo principal do site, normalmente temos as tags <header> e <footer> principais e logo no meio temos a tag <main>.

Por ser um elemento mais específico, apenas um deve ser criado no site.

HTML semântico: <nav>

<nav></nav>

Define uma seção de navegação, seja o menu principal do site ou qualquer outro conjunto de links que formem um espaço onde é possível navegar por ele. Pode ser utilizado multiplas vezes.

HTML semântico: <article>

<article></article>

Representa um conteúdo independente e que não necessita de nenhum outro relacionado, por exemplo um post de um blog, ou informações de uma página de sobre e relacionados.

HTML semântico: <section>

<section></section>

Define uma seção dentro do documento. Normalmente é criado em conjunto com um article, mas não é uma regra, por exemplo: um post de um blog pode ter a seção do conteúdo, e uma outra seção de comentários.

Utilize a tag <section> para criar grupos de conteúdos.

HTML semântico: <aside>

<aside></aside>

Define um conteúdo relacionado, mas que não é essencial ao conteúdo principal, normalmente localizado na tag <article>. Normalmente é utilizado para exibir links para navegar no post, ou então posts relacionados.

5

Propriedades CSS

Flex

Propriedades CSS: display: flex

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

<div class="flex">
  <div class="flex-item">
    Elemento 1
  </div>
  <div class="flex-item">
    Elemento 2
  </div>
  <div class="flex-item">
    Elemento 3
  </div>
</div>
```

O **display: flex** é um dos tipos de display que existem no CSS, com ele é possível criar layouts flexíveis que são muito úteis para criação de layouts que serão executados em multiplos dispositivos, seguindo o conceito de responsividade.

Quando o display flex é aplicado no elemento pai, automaticamente todos os elementos filhos se tornam flexíveis.

As principais caracteristicas do display flex incluem: alinhamento flexível, distribuição do espaço, tamanho e reorganização dos itens.

Outras propriedades podem ser utilizadas em conjunto com o display flex para criação do layout.

Propriedades CSS: gap

```
<style>
  .flex {
    display: flex;
    gap: 2rem;
  }
</style>

<div class="flex">
  <div class="flex-item">
    Elemento 1
  </div>
  <div class="flex-item">
    Elemento 2
  </div>
  <div class="flex-item">
    Elemento 3
  </div>
</div>
```

Define o espaçamento entre os itens do container.

Podem ser utilizados valores númericos seguido de uma unidade de medida.

Propriedades CSS: align-items

```
<style>
  .flex {
    display: flex;
    height: 10rem;
    align-items: center;
  }
</style>

<div class="flex">
  <div class="flex-item">
    Elemento 1
  </div>
  <div class="flex-item">
    Elemento 2
  </div>
  <div class="flex-item">
    Elemento 3
  </div>
</div>
```

Alinha os itens verticalmente dentro do container, o valor padrão é **flex-start**, e existem as opções:

- **flex-start** – alinhamento no início;
- **flex-end** – alinhamento no final;
- **center** – alinhamento no centro;
- **baseline** – alinhados na linha da base;
- **stretch** – esticados para preencher o container.

Propriedades CSS: justify-content

```
<style>
  .flex {
    display: flex;
    justify-content: center;
  }
</style>

<div class="flex">
  <div class="flex-item">
    Elemento 1
  </div>
  <div class="flex-item">
    Elemento 2
  </div>
  <div class="flex-item">
    Elemento 3
  </div>
</div>
```

Alinha os itens horizontalmente dentro do container, o valor padrão é **flex-start**, e existem as opções:

- **flex-start** – alinhamento no início;
- **flex-end** – alinhamento no final;
- **center** – alinhamento no centro;
- **space-between** – espaçados uniformemente;
- **space-around** – espaçados com espaços iguais em torno deles.

Propriedades CSS: flex-direction

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

  .flex .flex-item {
    flex-direction: column;
  }
</style>

<div class="flex">
  <div class="flex-item">
    Elemento 1
  </div>
  <div class="flex-item">
    Elemento 2
  </div>
  <div class="flex-item">
    Elemento 3
  </div>
</div>
```

Define a direção dos itens dentro do container, por padrão a direção é “row” e existem mais opções:

- **row** – horizontal;
- **column** – vertical;
- **row-reverse** – horizontal invertido;
- **column-reverse** – vertical invertido.

Propriedades CSS: flex-wrap

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

  .flex .flex-item {
    flex-wrap: wrap;
  }
</style>

<div class="flex">
  <div class="flex-item">
    Elemento 1
  </div>
  <div class="flex-item">
    Elemento 2
  </div>
  <div class="flex-item">
    Elemento 3
  </div>
</div>
```

Determina se os itens podem quebrar em várias linhas dentro do container, por padrão a opção é **“nowrap”**, e existem as opções:

- **nowrap** – não quebrar;
- **wrap** – quebrar;
- **wrap-reverse** – quebrar invertido.

Propriedades CSS: flex-grow

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

  .flex .flex-item {
    flex-grow: 1;
  }

  .flex .flex-item:first-item,
  .flex .flex-item:last-item {
    flex-grow: 3;
  }
</style>

<div class="flex">
  <div class="flex-item">
    Elemento 1
  </div>
  <div class="flex-item">
    Elemento 2
  </div>
  <div class="flex-item">
    Elemento 3
  </div>
</div>
```

Determina como os itens flexíveis crescem para ocupar o espaço no container, um maior valor aumenta a proporção em relação aos outros, os valores de entrada são numéricos.

Propriedades CSS: flex-shrink

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

  .flex .flex-item {
    flex-grow: 1;
  }

  .flex .flex-item:first-item {
    flex-shrink: 3;
  }
</style>

<div class="flex">
  <div class="flex-item">
    Elemento 1
  </div>
  <div class="flex-item">
    Elemento 2
  </div>
  <div class="flex-item">
    Elemento 3
  </div>
</div>
```

Determina como os itens flexíveis encolhem em relação aos outros itens no container, quanto maior o valor, mais o item vai encolher.

Propriedades CSS: flex-basis

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

  .flex .flex-item {
    flex-grow: 1;
  }

  .flex .flex-item:first-item {
    flex-shrink: 3;
  }
</style>

<div class="flex">
  <div class="flex-item">
    Elemento 1
  </div>
  <div class="flex-item">
    Elemento 2
  </div>
  <div class="flex-item">
    Elemento 3
  </div>
</div>
```

Determina o tamanho do item dentro do container antes da distribuição do espaço restante. Podem ser definidos valores numéricos seguidos de uma unidade de medida como “px” ou “rem”.

Propriedades CSS: order

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

  .flex .flex-item {
    order: 1;
  }

  .flex .flex-item:nth-child(2) {
    order: 3;
  }

  .flex .flex-item:nth-child(3) {
    order: 2;
  }
</style>

<div class="flex">
  <div class="flex-item">
    Elemento 1
  </div>
  <div class="flex-item">
    Elemento 2
  </div>
  <div class="flex-item">
    Elemento 3
  </div>
</div>
```

Define a ordem em que os itens serão exibidos dentro do container, pode ser um número inteiro positive ou negativo.

6

Propriedades CSS

Grid

Propriedades CSS: display: grid

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

<div class="flex">
  <div class="flex-item">
    Elemento 1
  </div>
  <div class="flex-item">
    Elemento 2
  </div>
  <div class="flex-item">
    Elemento 3
  </div>
</div>
```

O **display: grid** permite criar layours de forma bidimensional, organizando em uma grade de linhas e colunas.

A propriedade deve ser definida no container pai e logo em seguida definir a quantidade de colunas (e linhas caso existam) do grid.

Normalmente a grid é mais utilizada para criar uma linha com multiplas colunas, essa prática deixa mais flexível para responsividade.

As propriedades **align-items, justify-content, gap e order** podem ser utilizadas também com o display: grid.

Propriedades CSS: grid-template-columns

```
<style>
  .grid {
    display: grid;
    grid-template-columns: 1fr 1fr 20rem;
  }
</style>

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

Define as colunas de uma grid. Essa propriedade é bem dinâmica e pode definir o tamanho de cada coluna. É possível utilizar diferentes unidades de medida.

A grid conta com uma unidade especifica chamada “fr” (fração fracionada), que define dinâmicamente quantas colunas o elemento irá ocupar.

É importante que a quantidade de elementos seja equivalente com a quantidade de valores definidos no **grid-template-columns**.

7

Sugestões de **Exercícios**

Exercícios para treinar Divs, CSS, Flex e Grid

Com o site criado na última aula

- Defina um header, main, nav e footer para o seu site – utilize display flex para organizar os itens do header e navegador;
- Crie uma página com listagem de posts para o blog – utilize display grid;
- Crie um página para exibir o post do blog com um aside;
- Use a criatividade

Para se inspirar

- Lee Robinson Personal Website: https://leerob.io/
- Zeno Rocha Personal Website: https://zenorocha.com/
- Gustavo Sales Personal Website: https://gsales.io/

Bons estudos! 😀

OBRIGADO

@guscsales - gsales.io

FIAP