Prova LMS

- Conteúdo

- [x] ~~HTML semântico~~
- [x] ~~CSS básico~~
- [ ] CSS de produções
- [ ] Transformações e posições

- Estrutura básica do HTML

Seção Head
o Título
o Recursos externos
o Metadados

Seção Body
o Texto
o Imagem
o Links

Exemplo

```
<!DOCTYPE html>
<html>
```

```
<head>
...
</head>
<body>
...
</body>
</html>
```

- Cabeçalhos

Cabeçalhos são definidos pelas tags <h1>...<h6>, adicionam espaços em branco antes e depois do heading.

O elemento `<hr>` em HTML é usado para criar uma **linha horizontal** que geralmente é utilizada para separar seções ou conteúdos dentro de uma página.

- Parágrafos

Adicionam espaços em branco antes e depois do parágrafo. O parágrafo vai remover quebras de linha e espaços extras no texto. O elemento para o parágrafo é .... . Para quebra de linha, utilizamos
...</br>

- Formatação de texto (define elementos INLINE)

<b> Texto em negrito

 Texto semanticamente forte

<i> Texto em itálico

 Texto semanticamente enfatizado

<mark> Texto marcado

<small> Texto pequeno

<del> Texto deletado

<ins> Texto sublinhado

<sub> Texto subscripto

<sup> Texto sobrescrito

- Imagens

Imagens são definidas pela tag `<img>`.

```
<img src="url" alt="some_text”/>
```

O atributo ALT é para uma mensagem alternativa caso a imagem não seja carregada.

- Links

O elemento <a> define links entre documentos.

```
<a href="url">Visite nosso site</a>
```

- As tabelas em HTML são definidas através da tag `<table>`;

<tr> define uma linha

<th> define header

<td> define uma célula (pode conter outros elementos como textos, imagens, listas e outras tabelas)

<table>

- Listas não ordenadas e ordenadas (ul e ol)

Listas não-ordenadas são aquelas que não possuem uma sequência de números e são definidas pela tag `<ul>`. Para criar qualquer item das listas, utiliza-se o `<li>`.

Listas ordenadas são com sequências de número (1, 2, 3) e são definidas pela tag `<ol>`. Podemos usar o atributo `type` para definir o estilo de marcação da lista.

type 1= com números

type A= letras maiúsculas

type a= letras minúsculas

type I= algarismos romanos maiúsculos

type i= algarismos romanos minúsculos

- Diferença entre elementos BLOCO e elementos INLINE

Elementos bloco sempre começam em uma nova linha. Exemplos:

* *

<h1><h6>

<p>

<form>

Já os elementos inline não começam uma nova linha. Exemplos:

**

<img>

<a>

- Elemento div e span

O elemento é utilizado como container para outros elementos HTML. É comum usar style e class nele.

```
<div style="background-color: black; color:cornsilk; padding
      <h1>Meu primeiro site</h1>
      <p>Lorem ipsum dolor sit amet. Qui magnam nisi hic mo
      <p>Qui nisi perspiciatis cum vitae omnis quo quia sin

      <h2>Meus interesses</h2>
      <ul>
          <li>Lorem ipsum</li>
          <li>Lorem ipsum</li>
          <li>Lorem ipsum</li>
      </ol>
      <table>
       <tr>
          <th>Firstname</th>
          <th>Lastname</th>
          <th>Age</th>
       </tr>
          <td>Jill</td>
          <td>Smith</td>
          <td>50</td>
      </table>
      </div>
```

Já o elemento é um container para texto, é muito utilizado para estilizar o texto. Exemplo:

```
<p>Lorem ipsum <span style="color:red">sit amet.</span> Qui m
        <p>Qui nisi perspiciatis cum vitae omnis quo quia sin
```

- Formulários em HTML

O elemento `<form>` serve como container para elementos de formulários. É um elemento bloco e envia informações para o servidor.

```
<form action=”/search” method=GET>
… Elementos do formulário …
</form>
```

O atributo `action` especifica a URL para a qual serão enviados os dados do formulário, e o atributo `method` especifica o modo de envio (GET,POST,PUT,DELETE)

Formulários podem conter parágrafos, listas, tabelas e imagens (Com exceção de outros formulários).

- Elementos pra entrada de dados

```
<input>, <select>, <textarea>, <button>, <datalist>, <keygen>, <output>
```

O elemento `<input>` é o mais importante de um formulário, pois define elementos de entrada de dados do usuário via atributo **type.**

O tipo `password` define o formato de senha no formulário, já o tipo `submit` submete o formulário para o servidor.

Há estilos que podem ser adicionados no formulário, como o tipo `radio` (seleção em bolinha) e o tipo `checkbox` (seleção em quadradinho).

O tipo `hidden` esconde alguma informação do usuário, e aparece apenas para o comando do navegador. Exemplo:

```
<input type=hidden name="codigo" value="123">
```

O tipo `number` estabelece um campo numérico e pode-se validar um intervalo entre eles (de 1 a 5, 1 a 9...) Exemplo:

```
<input type="number" name="quantidade" min="1" max="9" />
```

O tipo date estabelece um campo com barras (/). Exemplo:

```
<input type="date" name="aniv”
```

Alguns atributos `<input>` são:

● Atributo value
Especifica valor inicial no campo
● Atributo
readonly
Especifica que campo não pode ser mudado
● Atributo
disabled
Especifica que campo está desabilitado
● Atributo
size
Especifica o tamanho máximo em caracteres
● Atributo
maxlength
Especifica número máximo da caracteres permitidos

- Elementos semânticos e estruturais

Elementos não-semânticos: São elementos que não possuem um significado claro por si mesmos e não indicam o tipo de conteúdo que contêm. Exemplos:

`<div>, <span>`

Elementos semânticos: São elementos que têm significado explícito e descrevem claramente seu conteúdo ou função tanto para desenvolvedores quanto para navegadores e tecnologias assistivas (como leitores de tela). Exemplos:

`<table>, <forms> e <img>`

Alguns outros elementos semânticos estruturais são:

A tag `<section>` no HTML é um elemento **semântico** usado para agrupar conteúdo relacionado dentro de uma página web. Ela define uma **seção temática** do documento, geralmente com um título (`<h1>`, `<h2>`, etc.) que descreve o tópico ou propósito da seção. Exemplo:

```
<section>
<h2>Sobre Nós</h2>
<p>Somos uma empresa dedicada a fornecer soluções inovadoras e personalizadas para nossos clientes.</p>
</section>
```

```
<section>
<h2>Serviços</h2>
<ul>
<li>Consultoria de Negócios</li>
<li>Desenvolvimento de Software</li>
<li>Marketing Digital</li>
```

```
</ul>
</section>
```

A tag `<article>` no HTML é um elemento **semântico** usado para representar um conteúdo **independente e autocontido**, que pode ser reutilizado ou distribuído separadamente de sua página principal.

Ela é ideal para **artigos, postagens de blog, notícias, tutoriais ou qualquer conteúdo que tenha sentido por si só**.

```
<article>
<h2>Como Cuidar de Plantas de Interior</h2>
<p>Plantas de interior trazem vida e frescor a qualquer ambiente...</p>
<footer>
<p>Publicado em: 17 de dezembro de 2024</p>
</footer>
</article>
```

Qual usar?
a. O conteúdo contido faria mais sentido por si só num leitor de feed? Se sim, use
`<article>`
b. O conteúdo contido é relacionado? Se sim, use
`<section>`
c. Se não há relação semântica, use
`<div>`

A tag `<header>`, que pode ser usada junto com <section> e <article>, define um cabeçalho.

Já a tag `<nav>` define um conjunto de links de navegação. Exemplo:

```
<nav>
<a href="/home/">Home</a>
<a href="/sobre/">Sobre</a>
<a href="/servicos/">Serviços</a>
<a href="/contato/">Contato</a>
</nav>
```

O que é CSS

- CSS (Cascading Style Sheets) define como os elementos HTML são renderizados na tela.
- HTML organiza o **conteúdo**, enquanto o CSS define a **aparência**.

Modos de Adicionar CSS

1. **Inline:**
    - Adicionado diretamente no atributo `style` de um elemento.

```
h1 style="color:blue;margin-left:30px;">Título</h1>
```

Desvantagem: Mistura HTML e CSS, dificultando a manutenção.

2. **Interno:**
   - Definido dentro da tag `<style>` no `<head>` do documento.

```
<head>
  <style>
    body { background-color: linen; }
    h1 { color: maroon; margin-left: 40px; }
  </style>
</head>
```

3. **Externo:**
   - Estilo salvo em um arquivo `.css` separado, vinculado com `<link>`.

```
<head>
  <link rel="stylesheet" type="text/css" href="styles.cs
s">
</head>
```

Arquivo `styles.css`:

```
body { background-color: lightblue; }
h1 { color: navy; margin-left: 20px; }
```

Seletores CSS

1. **Por Elemento:**
   - Estilo aplicado a todos os elementos do mesmo tipo.

```
p { color: red; }
```

2. **Por ID:**
   - Identifica elementos únicos com `#`.

```
#titulo { color: blue; }
```

3. **Por Classe:**
   - Aplicado a múltiplos elementos com `.`.

```
.destaque { font-weight: bold; }
```

4. **Agrupamento de Seletores:**
   - Um estilo aplicado a vários elementos.

```
h1, h2, p { text-align: center; }
```

5. **Seletores Hierárquicos:**
   - Aplicado com base no relacionamento entre elementos.

```
div p { color: green; } /* Aplica a <p> dentro de <div>
*/
div > p { color: red; } /* Aplica a <p> filho direto de
```

```
<div> */
```

Modelo de Caixa CSS

Cada elemento é uma "caixa" composta por:

1. **Content:** O conteúdo.
2. **Padding:** Espaço entre o conteúdo e a borda.
3. **Border:** A borda da caixa.
4. **Margin:** Espaço externo, separa elementos.

Exemplo de cálculo:

```
div {
  width: 320px;
  padding: 10px;
  border: 5px solid gray;
  margin: 15px;
}
/* Largura total = 320 + (2x10) + (2x5) + (2x15) = 380px */
```

Propriedades de Texto

1. **Cor do Texto:**

```
h1 { color: green; }
```

2. **Alinhamento:**

```
h1 { text-align: center; }
```

3. **Decoração:**

```
h1 { text-decoration: underline; }
```

4. **Transformação de Texto:**

```
p { text-transform: uppercase; }
```

Exemplo para Revisão

HTML:

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
  <link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
  <div id="container">
    <h1 class="destaque">Título</h1>
    <p>Este é um parágrafo.</p>
  </div>
</body>
</html>
```

CSS (`styles.css`):

```
#container {
  width: 300px;
  margin: 0 auto;
```

```
  padding: 10px;
  border: 2px solid black;
}
.destaque {
  color: navy;
  text-align: center;
}
```

O que é Flexbox

- Modo eficiente para posicionar, alinhar e distribuir espaço entre elementos dentro de um container.
- **Características:**
  - Permite manipular tamanhos dos elementos filhos.
  - Funciona em layouts flexíveis (horizontal ou vertical).

Habilitar Flexbox

- Use a propriedade `display: flex` no container para ativar o flexbox.

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

Propriedades do Container

1. `flex-direction`**: Define o eixo principal.**
   - Valores:
     - `row` (padrão): Alinha horizontalmente, da esquerda para a direita.
     - `row-reverse`: Inverte horizontalmente, da direita para a esquerda.
     - `column`: Alinha verticalmente, de cima para baixo.
     - `column-reverse`: Inverte verticalmente, de baixo para cima.

```css
.container {
  flex-direction: row;
}
```

2. `flex-wrap`: **Define quebra de linha.**
   - Valores:
     - `nowrap` (padrão): Sem quebra de linha.
     - `wrap`: Permite que os itens quebrem para a próxima linha.
     - `wrap-reverse`: Quebra de linha invertida.

```css
.container {
  flex-wrap: wrap;
}
```

3. `justify-content`: **Alinha no eixo principal.**
   - Valores:
     - `flex-start` (padrão): Início do eixo.
     - `flex-end`: Final do eixo.
     - `center`: Centralizado no eixo.
     - `space-between`: Espaço igual entre os itens.
     - `space-around`: Espaço igual ao redor de cada item.
     - `space-evenly`: Espaço igual distribuído entre os itens e nas bordas.

```css
.container {
  justify-content: center;
}
```

4. `align-items`: **Alinha no eixo transversal (perpendicular).**
   - Valores:
     - `stretch` (padrão): Estica os itens.
     - `flex-start`: Alinha no início.
     - `flex-end`: Alinha no final.
     - `center`: Centraliza.
     - `baseline`: Alinha pela linha de base do texto.

```
.container {
  align-items: flex-end;
}
```

5. `align-content`: **Espaço entre linhas (quando há várias).**
   - Valores:
     - `stretch` (padrão), `flex-start`, `flex-end`, `center`, `space-between`, `space-around`.

```
.container {
  align-content: space-between;
}
```

Propriedades dos Itens

1. `order`: **Altera a ordem dos itens.**
   - Valor padrão: `0`.
   - Menor valor aparece antes.

```
.item {
  order: 1;
```

```
}
```

2. `flex-basis`: **Define o tamanho inicial antes de distribuir o espaço disponível.**

```
.item {
  flex-basis: 200px;
}
```

3. `flex-grow`: **Define quanto o item cresce em relação aos outros.**
   - Valor padrão: `0` (não cresce).

```
.item {
  flex-grow: 2;
}
```

4. `flex-shrink`: **Define quanto o item encolhe em relação aos outros.**
   - Valor padrão: `1` (encolhe).

```
.item {
  flex-shrink: 1;
}
```

5. `flex`: **Atalho para** `flex-grow`, `flex-shrink` **e** `flex-basis`.

```
.item {
  flex: 1 1 200px; /* grow, shrink, basis */
```

```
}
```

Exemplo Prático

HTML:

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

CSS:

```
.container {
  display: flex;
  flex-direction: row;
  justify-content: space-around;
  align-items: center;
}

.item {
  flex: 1;
  padding: 10px;
  border: 1px solid black;
  text-align: center;
}
```