5. HTML: Marcação com significado

> Semântica

Semântica é o estudo do significado das palavras, frases e símbolos.

No HTML, a semântica é utilizada para definir o significado dos elementos, ou seja, o que cada elemento representa.

pedro.mateus@unicesumar.edu.br

A semântica do ***HTML*** é importante para que o **navegador** e os **mecanismos de busca** consigam interpretar o conteúdo da página, e também torna o conteúdo mais **acessível**

Possibilitando que usuários com deficiência visual, por exemplo, que utilizam **leitores de tela** (exemplo de tecnologia assistiva) para navegar na internet.

Como dito anteriormente, a Web sempre foi pensada para ser acessível a todos.

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>UNICESUMAR - Programação WEB 2024</title>
  </head>
  <body>
    <header>
      <h1>Toda página deve possuir um, e apenas um, h1!</h1>
      <img src="https://i.imgur.com/D5lZGis.png" alt="Logo da UNICESUMAR" />
    </header>
    <main>
      <section>
        <h2>
          Semântica bem intuitiva, basta se familiarizar com os elementos html
          (tags)
        </h2>
        <h3>
          A maioria dos browsers já possuem alguns estilos próprios para certos
          elementos, como os títulos e subtítulos
        </h3>
        <p>
          Também existem elementos específicos para
          <em>Itálico (Emphasis element)</em> e também textos em
          <strong> Negrito (Strong element) </strong>
        </p>
      </section>
    </main>
    <footer>
      <p>
        &copy; 2024 Fulano
        <a
          href="https://github.com/satinp"
          title="Github do autor"
          rel="external"
        >
          Âncora que vincula essa página ao meu Github
        </a>
      </p>
    </footer>
  </body>
</html>
```

> Conteúdo de uma página

Quando um navegador renderiza o HTML, ele transforma espaços extras, tabs e quebras de linha em um único espaço.

O HTML é restrito aos caracteres ASCII— letras e símbolos comuns. O que torna diversos caracteres especiais inválidos, como alguns acentos, cedilhas, etc.

Parte desse problema pode ser resolvido utilizando a codificação de caracteres **UTF-8**.

O Unicode utf-8 é um super conjunto do ASCII, que permite a utilização de alguns caracteres especiais.

No exemplo anterior, utilizamos a tag `<meta charset="utf-8" />` para informar ao navegador que o documento está codificado em UTF-8.

Mesmo assim, é comum utilizarmos **entidades HTML** para representar caracteres especiais, por exemplo: `&copy;` para representar o símbolo de direitos autorais ©.

Mais informações podem ser encontradas na documentação da W3C.

Aqui também temos uma Tabela de entidades HTML.

Elemento âncora

No exemplo do slide anterior também utilizamos a tag `<a>`, um elemento âncora, utilizado para criar links para outras páginas (externos), ou para outras partes da mesma página.

É o elemento responsável por fazer a web ser navegável, o que o torna um dos elementos mais importantes do HTML.

O elemento espera um atributo `href` que indica para onde a tag será direcionada. Podendo referenciar um arquivo local, um link externo, ou um elemento da mesma página.

Exemplos de âncoras.

ARIA

Os ARIA (Accessible Rich Internet Applications) são um conjunto de atributos em HTML que foram introduzidos pelo World Wide Web Consortium (W3C) para melhorar a acessibilidade de aplicativos web ricos e interativos para pessoas com deficiências.

Eles fornecem informações adicionais sobre a estrutura, função e comportamento de elementos HTML, permitindo que tecnologias assistivas, como leitores de tela, interpretem e apresentem corretamente o conteúdo para usuários com deficiências visuais, auditivas ou motoras.

Principais Atributos ARIA:

role (papel): Define o papel ou função semântica de um elemento HTML.

aria-* (propriedades ARIA): Define propriedades específicas relacionadas à acessibilidade de um elemento HTML, como estado, propriedade ou relação.

Por exemplo, o atributo `role="button"` pode ser aplicado a um `<div>` ou `<span>` para indicar que ele funciona como um botão, o que é útil para usuários de leitores de tela que precisam saber a função de um elemento interativo.

```
<div role="button" onclick="myFunction()">Clique aqui</div>
```

Por exemplo, o atributo aria-hidden="true" pode ser usado para elementos ocultados visualmente, mas ainda torná-lo acessível para leitores de tela.

```
<div aria-hidden="true">Indica que o conteúdo é visualmente oculto</div>
```

Mais informações podem ser encontradas na Referência de ARIA no MDN e também na Documentação oficial do W3C sobre ARIA

E alguns exemplos com Padrões ARIA

6. CSS: Estilizando a página

O CSS é uma linguagem de estilização, que permite a criação de regras para estilizar elementos HTML.

A estilização é feita através de **regras** aplicadas a **seletores**, esses indicam quais elementos serão formatados.

As regras são no formato chave e valor. Onde a chave é a propriedade a ser estilizada, e o valor é o estilo a ser aplicado.

> Construindo uma regra de estilo

```css
/* Comentário css */
h1 {
  color: red;
}
```

Onde o h1 é o **seletor**, e o conteúdo do bloco entre `{}` são os conjuntos de **regras**.

Os principais seletores utilizados serão os de **elementos**, **classes** e **ids**.

> Aplicando estilos com seletores de elementos

Conforme o exemplo acima, aplicamos um estilo para o elemento `h1`, onde definimos a cor do texto como vermelho.

Esse estilo será aplicado para todos os elementos `h1` da página.

Porque isso pode ser problemático?

> Aplicando estilos com seletores de classes

Para resolver o problema de estilização de todos os elementos `h1`, podemos utilizar **classes**.

As classes são utilizadas para agrupar elementos que possuem características em comum.

```
<!-- HTML -->
<h1>Título da página</h1>
<h1 class="titulo">Título da seção</h1>
```

Podemos, então, aplicar estilos para a classe `titulo`, que vai ser aplicada apenas para o segundo `h1` da página.

```
/* CSS */
h1 {
  color: red;
}

.titulo {
  color: green;
}
```

> Aplicando estilos com seletores de id

Os **ids** são utilizados para identificar um elemento de maneira única.

```
<!-- HTML -->
<h1>Título da página</h1>
<h1 class="titulo">Título da seção</h1>
<h1 id="outro-titulo">Outro título</h1>
```

```css
/* CSS */
h1 {
  color: red;
}

.titulo {
  color: green;
}

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

Maneiras de aplicar estilos

1. Atributo `style`:

```
<p style="color: red;">Texto do paragrafo vermelho</p>
```

2. Tag `<style>` dentro do `<head>`:

```
<head>
  <style>
    p {
      color: red;
    }
  </style>
</head>
```

3. Arquivo externo:

Podemos criar um arquivo com extensão `.css`, e referenciá-lo no documento HTML, também na tag `<head>`.

```
<head>
  <link rel="stylesheet" href="style.css" />
</head>
```

> Propriedades fundamentais

1. Cores

A versão mais moderna do css permite algumas maneiras de definir cores, como por exemplo: `rgb()`, `rgba()`, `hsl()`, `hsla()`, `hexadecimal`, `nome da cor`.

```
/* Define a cor do texto como um vermelho
    escuro usando um valor hexadecimal */

/* Indo de 00 (mais escuro) at´e FF (mais claro)
    para o valor de cada uma das cores. */
p {
  color: #8b0000;
}
```

```css
/* Define a cor de fundo como verde escuro
  usando valores RGB (Red, Green, Blue) */
div {
  background-color: rgb(0, 100, 0); /* Verde escuro */
}

/* Define a cor de fundo como azul com 50%
  de transparência usando valores RGBA (Alfa)*/
div {
  /* Azul com 50% de transparência */
  background-color: rgba(0, 0, 255, 0.5);
}

/* Define a cor de fundo como um azul claro usando
    valores HSL (Hue, Saturation, Lightness) */
div {
  background-color: hsl(210, 70%, 80%); /* Azul claro */
}
```

2. Margin e Padding

O atributo **margin** é a distância entre o elemento e os elementos vizinhos.

Já o **padding** é a distância entre o conteúdo do elemento e a borda do elemento.

Eles podem ser escritos de algumas maneiras.

- Uniforme:

```
.box {
  margin: 20px;
}
```

- Individuais:

```
.box {
  margin-top: 10px;
  margin-right: 20px;
  margin-bottom: 15px;
  margin-left: 25px;
}
```

- Horizontais e Verticais:

```
.box {
  /* Margem superior e inferior de 10 pixels,
     margem direita e esquerda de 20 pixels */
  margin: 10px 20px;
}
```

e

```
.box {
  /* Margem superior de 10 pixels,
     margem direita e esquerda de 20 pixels,
     margem inferior de 15 pixels */
  margin: 10px 20px 15px;
}
```

- Alternadas:

```
.box {
  /* Margem superior de 10 pixels,
     margem direita de 20 pixels,
     margem inferior de 15 pixels,
     margem esquerda de 25 pixels */
  margin: 10px 20px 15px 25px;
}
```

Importante: A dimensão apropriada para a propriedade padding ou margin pode ser definida automaticamente pelo navegador mediante uso do valor auto.

```
.box {
  margin: 0 auto;
}
```

> Algumas das principais unidades de medidas css

Unidade	Descrição
`px`	Pixels
`em`	Relativo ao tamanho da fonte do elemento pai
`rem`	Relativo ao tamanho da fonte do elemento raiz (root)
`vw`	Relativo à largura da viewport (1vw = 1% view width)
`vh`	Relativo à altura da viewport (1vh = 1% view height)
`%`	Percentual

Referências:

Livros:

- HTML5 e CSS3: guia prático e visual - Elizabeth Castro / Bruce Hyslop
- Guia de orientação e desenvolvimento de sites: HTML, XHTML, CSS e Javascript/Jscript - José A. Manzano / Suely Alves de Toledo

Sites/Documentações:

- W3C, MDN, W3Schools, Can I use, Open Graph.