Programação Web Responsiva

Criando um Menu de Navegação e Trabalhando com Pseudo-Elementos

Prof. Bruno Azevedo

UNIP – Universidade Paulista

A Tag <nav>

- Representa uma seção de navegação dentro de uma página.
- Normalmente contém links para outras páginas ou seções do site.
- Um documento HTML pode conter vários elementos <nav>.
- A tag <nav> não deve ser usada para links não relacionados à navegação (como links para redes sociais ou outras seções auxiliares).
  - ▷ Exemplo de uso:

```
<nav>
  <ul>
    <li><a href="index.html">Início</a></li>
    <li><a href="sobre.html">Sobre</a></li>
    <li><a href="servicos.html">Serviços</a></li>
    <li><a href="contato.html">Contato</a></li>
  </ul>
</nav>
```

Seletores e Classes e IDs

- O atributo **class** é usado para estilizar múltiplos elementos ao mesmo tempo.
- O atributo **id** permite estilizar um único elemento de forma exclusiva.

```
<div id="noticia-principal">
    <h1 class="titulo">Nova Descoberta Científica</h1>
    <p class="resumo">Pesquisadores anunciam um avanço
                      revolucionário na medicina.</p>
</div>
```

- O **id** `noticia-principal` identifica exclusivamente a notícia mais importante da página.
- A **class** `titulo` estiliza o título da notícia.
- A **class** `resumo` estiliza o parágrafo que resume a notícia.

Seletores e IDs

- Vamos selecionar um elemento **através de seu id** e estilizá-lo:

```
<!DOCTYPE html>
<html>
    <head>
        <style>
            #meuid {
                font-family: monospace;
                font-size: 30px;
                color: blue;
            }
        </style>
    </head>
    <body>
        <p> Um parágrafo não-personalizado. </p>
        <p id="meuid"> Um parágrafo personalizado. </p>
    </body>
</html>
```

Seletores e Classes

- Vamos selecionar um elemento **através de sua classe** e estilizá-lo:

```
<!DOCTYPE html>
<html>
    <head>
        <style>
            .minhaclasse {
                font-family: monospace;
                font-size: 30px;
                color: blue;
            }
        </style>
    </head>
    <body>
        <p> Um parágrafo não-personalizado. </p>
        <p class="minhaclasse"> Um parágrafo personalizado. </p>
    </body>
</html>
```

Seletores, Classes e IDs

Resumindo:

- Usamos o nome do elemento para estilizá-lo.

```
p {
    font-family: monospace;
    font-size: 30px;
    color: blue;
}
```

- Se estamos estilizando um ID, usamos o símbolo # seguido do nome do identificador.

```
#meuid {
    font-family: monospace;
    font-size: 30px;
    color: blue;
}
```

- Se estamos estilizando uma classe, usamos o símbolo . seguido do nome da classe.

```
.minhaclasse {
    font-family: monospace;
    font-size: 30px;
    color: blue;
}
```

Seletores combinados em CSS

- Seletores combinados permitem aplicar estilos com mais precisão, considerando o tipo do elemento, sua posição na hierarquia do HTML e suas classes ou IDs.
- Principais combinações:
  - `seletor1 seletor2`: seleciona todos os elementos `seletor2` que estão dentro de `seletor1`, em qualquer nível de profundidade.
  - `seletor1 > seletor2`: seleciona apenas os elementos `seletor2` que são filhos diretos de `seletor1`.
  - `tipo.classe`: seleciona elementos do tipo especificado que possuem a classe indicada. Exemplo: `p.destaque` seleciona apenas parágrafos com a classe `destaque`.
  - `tipo .classe`: seleciona qualquer elemento com a classe indicada, desde que esteja dentro de um elemento do tipo especificado. Exemplo: `p .destaque` seleciona qualquer elemento com a classe `destaque` que esteja dentro de um parágrafo.

Exemplo: div ul li

- O seletor `div ul li` seleciona todos os `li` que estão dentro de uma `ul` que está dentro de uma `div`.
- Exemplo de CSS:

```
<-- CSS -->
div ul li {
  list-style-type: square;
}

<-- HTML -->
<div>
  <ul>
    <li>Item 1</li>
    <li>Item 2</li>
  </ul>
</div>
```

- Os itens da lista aparecerão com marcadores quadrados.

Exemplo: p.minhaclass span

- O seletor `p.minhaclass span` seleciona todos os elementos `<span>` que estão dentro de um parágrafo `<p>` com a classe `minhaclass`.
- Exemplo:

```
<-- CSS -->
p.minhaclass span {
  color: red;
}

<-- HTML -->
<p class="minhaclass">
  Olá <span>mundo</span>!
</p>
<p>
  Outro <span>exemplo</span> que não será afetado.
</p>
<div class="minhaclass">
  Fora do <span>alvo</span>.
</div>
```

- Apenas o `<span>` dentro do `<p class="minhaclass">` será estilizado em vermelho.
- O seletor não se aplica a parágrafos sem a classe `minhaclass` nem a outros elementos com essa classe que não sejam `<p>`.

Diferença: `tipo.classe` vs `tipo .classe`

- `tipo.classe`: seleciona elementos que são do tipo especificado e possuem a classe.
- `tipo .classe`: seleciona qualquer elemento com a classe, desde que esteja dentro do elemento do tipo indicado.
- Exemplo:

```
<-- CSS -->
p.destaque {
  color: red;
}
p .destaque {
  color: blue;
}

<-- HTML -->
<p class="destaque">Texto A</p>
<p>Texto B <span class="destaque">dentro do p</span></p>
```

- O primeiro <p> ficará com texto vermelho.
- O dentro do segundo <p> ficará com texto azul.

Pseudo-classes e Pseudo-elementos

- **Pseudo-classes**: Usadas para estilizar elementos com base em seu estado (ex: `:hover`, `:focus`).
- **Pseudo-elementos**: Usados para estilizar partes específicas de um elemento (ex: `::before`, `::after`).

Pseudo-classes

- Pseudo-classes são usadas para definir o estado de um elemento.
- ▷ Exemplos de pseudo-classes:
  - `:hover`: Quando o elemento é **hoverado** (passa o mouse sobre ele).
  - `:focus`: Quando o elemento recebe foco (campo de texto ativo).
  - `:nth-child()`: Seleciona elementos com base na sua posição no DOM.
  - `:active`: Quando o elemento está sendo clicado.

Pseudo-classes

- Vimos um exemplo de `:hover` no menu.
- ▷ No exemplo abaixo, quando o mouse passa sobre o link, a cor de fundo muda:

```
a:hover {
  background-color: yellow;
}
```

Pseudo-elementos

- Pseudo-elementos são usados para estilizar partes específicas de um elemento.
- ▷ Exemplos de pseudo-elementos:
  - `::first-letter`: Estiliza a primeira letra de um elemento.
  - `::first-line`: Estiliza a primeira linha de um elemento.
  - `::before`: Adiciona conteúdo antes de um elemento.
  - `::after`: Adiciona conteúdo após um elemento.
- Existem vários outros como `::cue`, `::selection`, `::slotted()`, `::backdrop`, `::placeholder`, `::marker`, `::spelling-error`, `::grammar-error`, etc.

Pseudo-elementos

- A primeira linha de cada parágrafo será vermelha e exibida em versaletes.

```
p::first-line {
  color: red;
  font-variant: small-caps;
}
```

- A primeira letra de cada parágrafo será azul e terá tamanho de 40 pixels.

```
p::first-letter {
  color: blue;
  font-size: 40px;
}
```

Combinação de Pseudo-classes e Pseudo-elementos

- Você pode combinar pseudo-classes e pseudo-elementos.
- ▷ Por exemplo, o código abaixo altera a cor da primeira letra de um parágrafo quando o usuário passa o mouse sobre ela.

```
p:hover::first-letter {
  color: red;
}
```

A propriedade `content`

- A propriedade `content` é usada com os pseudoelementos `::before` e `::after`.
- Serve para inserir conteúdo gerado automaticamente antes ou depois de um elemento.
- Não adiciona o conteúdo no HTML, apenas visualmente via CSS.
- Muito usada para adicionar ícones, setas, marcadores e efeitos decorativos.

Exemplo com ::before

```
<ul class="menu">
  <li>Início</li>
  <li>Serviços</li>
  <li>Contato</li>
</ul>

.menu li::before {
  content: url("icone.png");
  margin-right: 8px;
}
```

Criando um Menu

- Vamos criar um menu simples para o nosso site.
- O HTML contém:
  - Uma Tag de navegação
  - Uma lista não-ordenada de itens.
  - Links para cada item da lista.

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <title>Menu</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <nav>
        <ul>
            <li><a href="index.html">Início</a></li>
            <li><a href="sobre.html">Sobre</a></li>
            <li><a href="servicos.html">Serviços</a></li>
            <li><a href="contato.html">Contato</a></li>
        </ul>
    </nav>
</body>
</html>
```

Criando um Menu

- Agora vamos criar o CSS para tornar nosso menu visualmente interessante.
- Resetando margens e padding.

```
* {
    margin: 0;
    padding: 0;
}
```

  - Uma boa prática para garantir que todos os elementos comecem com o mesmo ponto de partida em termos de estilo, evitando surpresas devido aos estilos padrão dos navegadores.
  - O asterisco seleciona todos os elementos da página.

Criando um Menu

- Estilização para o corpo da página.

```
body {
    font-family: Arial, sans-serif;
    background-color: #f4f4f4;
}
```

- Estilização para o menu de navegação.

```
nav {
    background-color: black;
    padding: 10px 20px 10px 20px;
}
```

- ▷ Os valores de `padding` refere-se ao topo, direita, fundo, e esquerda.
- ▷ Alternativa: `padding: 10px 20px;`.

Criando um Menu

- Estilização para a lista de navegação.
  - O seletor `nav ul` aplica estilos à lista `<ul>` dentro do menu de navegação `<nav>`.

```
nav ul {
 list-style: none; /* Remove os marcadores da lista */
 text-align: center; /* Alinha o texto de forma centralizada */
}
```

- Estilização para os itens da lista.
  - O seletor `nav ul li` aplica estilo aos itens de lista `<li>` dentro da lista que está dentro do menu de navegação.

```
nav ul li {
 display: inline-block; /* Faz com que os itens da lista sejam exibidos
                           lado a lado, em vez de um abaixo do outro. */
 margin: 0 20px 0px 20px; /* Adiciona um espaçamento de 20px à esquerda e
                             direita de cada item. */
}
```

Criando um Menu

- Estilização para os links do menu.

```
nav ul li a {
    display: inline-block; /* Torna o link um bloco inline,
                              o que permite adicionar
                              preenchimento adequado. */
    color: white;
    text-decoration: none; /* Remove o sublinhado padrão dos links. */
    font-size: 18px;
    padding: 8px 15px 8px 15px;
}
```

- Efeito de hover nos links.

▷ O seletor `nav ul li a:hover` aplica estilo aos links quando o usuário passa o mouse (hover) sobre eles:

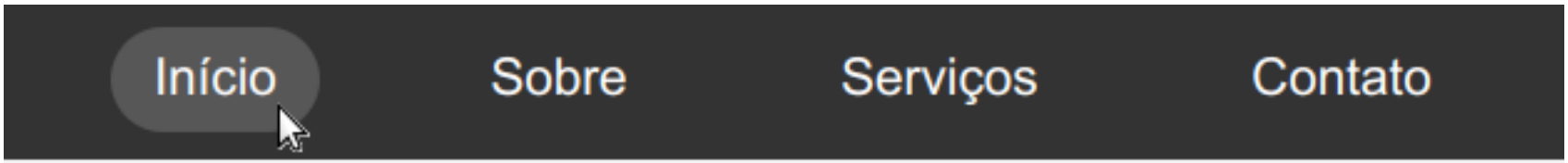
```
nav ul li a:hover {
    background-color: #575757;
    border-radius: 20px; /* Adiciona bordas arredondadas aos links,
                            com um raio de 20px, fazendo-os parecer
                            botões arredondados.*/
}
```

Código Completo

```html
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <title>Menu</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <nav>
        <ul>
            <li><a href="index.html">Início</a></li>
            <li><a href="sobre.html">Sobre</a></li>
            <li><a href="servicos.html">Serviços</a></li>
            <li><a href="contato.html">Contato</a></li>
        </ul>
    </nav>
</body>
</html>
```

```css
* {
    margin: 0;
    padding: 0;
}
body {
    font-family: Arial, sans-serif;
    background-color: #f4f4f4;
}
nav {
    background-color: black;
    padding: 10px 20px 10px 20px;
}
nav ul {
    list-style: none;
    text-align: center;
}
nav ul li {
    display: inline-block;
    margin: 0px 20px 0px 20px;
}
nav ul li a {
    display: inline-block;
    color: white;
    text-decoration: none;
    font-size: 18px;
    padding: 8px 15px;
}
nav ul li a:hover {
    background-color: #575757;
    border-radius: 20px;
}
```

Nosso Menu



Responsivo?

- Esse menu não está responsivo.
- Ele não se adapta a diferentes dispositivos.

O que é Responsividade

- Responsividade é a capacidade de uma página web se adaptar a diferentes tamanhos e resoluções de tela.
- O objetivo é garantir uma boa visualização e usabilidade em `smartphones`, `tablets`, `notebooks` e `monitores grandes`.
- Um site responsivo melhora a experiência do usuário.
- Elementos se reorganizam e redimensionam conforme o espaço disponível.
- A responsividade é essencial nos dias de hoje, considerando a ampla variedade de dispositivos utilizados para acessar a web.

Como tornar a página responsiva

- **Uso de unidades relativas como %, `em`, `rem`, e `vw/vh` ao invés de `px`.**
- **Utilização de `media queries` no CSS.**
- **Definição de `viewport` no HTML: `<meta name="viewport" content="width=device-width, initial-scale=1.0">`. Define que a largura da viewport deve ser igual à largura da tela do dispositivo, nível de zoom, etc.**
- Layouts flexíveis com `Flexbox` ou `Grid Layout`.
- Utilização de frameworks como **Bootstrap**.
- Entre outras estratégias.

Unidades em e rem no CSS

- As unidades em e rem são relativas e adaptáveis, ideais para design responsivo.
- 1em equivale ao tamanho da fonte do elemento pai.
- 1rem equivale ao tamanho da fonte da <html> (raiz do documento).
- Se o html tiver font-size: 16px, então:
  - 1rem = 16px.
  - 2rem = 32px, e assim por diante.
  - 1em depende da hierarquia de elementos.

O que são media queries

- As media queries permitem aplicar estilos CSS diferentes dependendo das características do dispositivo.
- A característica mais comum é a largura da tela, mas também podemos testar altura, resolução, orientação etc.
- A sintaxe é:

```
@media (condição) {
  /* estilos aqui */
}
```

- Podem ser usadas para tornar páginas responsivas sem alterar o HTML.

Exemplo de media query

- As regras dentro da @media só se aplicam se a condição for verdadeira.
- No exemplo abaixo, dispositivos com largura até 600px terão um layout adaptado.

```
/* Estilos para telas com largura menor que 600px */
@media (max-width: 600px) {
  .container {
    width: 100%;
    padding: 10px;
  }
}
/* Estilos padrão */
.container {
  width: 80%;
  margin: auto;
}
```

Condições comuns nas `media queries`

- `max-width`: aplica o estilo até uma largura máxima.
- `min-width`: aplica o estilo a partir de uma largura mínima.
- `orientation`: aplica o estilo conforme a orientação (`portrait` ou `landscape`).
- `resolution`: aplica conforme a densidade de pixels.

```
@media (min-width: 768px) { ... }
@media (max-width: 480px) { ... }
@media (orientation: portrait) { ... }
@media (min-resolution: 2dppx) { ... }
```

- dppx - Dots Per Pixel (1dppx = 96dpi) e dpi - Dots Per Inch.

Combinando condições nas media queries

- Podemos combinar várias condições com and:

```
@media (min-width: 600px) and (max-width: 900px) {
  /* Aplica entre 600px e 900px */
}
```

- Ou com vírgula (,) se apenas uma das condições precisar ser verdadeira:

```
@media (max-width: 480px), (orientation: portrait) {
  /* Aplica se qualquer uma das condições for verdadeira */
}
```

Código Responsivo

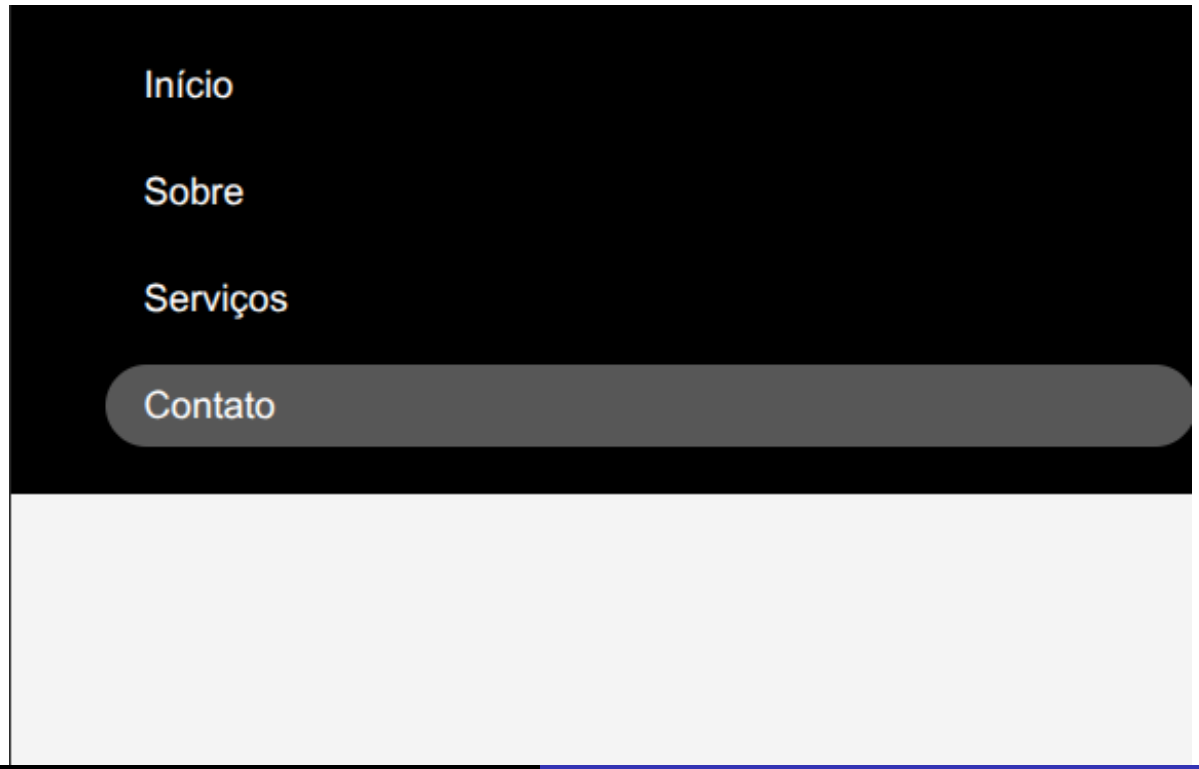
```
* {
    margin: 0;
    padding: 0;
}
body {
    font-family: Arial, sans-serif;
    background-color: #f4f4f4;
}
nav {
    background-color: black;
    padding: 0.625rem 1.25rem; /* 10px 20px */
}
nav ul {
    list-style: none;
    text-align: center;
    width: 100%;
}
nav ul li {
    display: inline-block;
    margin: 0 5%;
}
nav ul li a {
    display: inline-block;
    color: white;
    text-decoration: none;
    font-size: 1.125rem; /* 18px */
    padding: 0.5rem 0.9375rem; /* 8px 15px */
}
nav ul li a:hover {
    background-color: #575757;
    border-radius: 1.25rem; /* 20px */
}
```

```
/* Responsividade para telas menores */
@media screen and (max-width: 600px) {
    nav ul {
        text-align: left;
        padding-left: 1.25rem;
    }
    nav ul li {
        display: block;
        margin: 0.625rem 0; /* 10px em cima e embaixo */
    }
    nav ul li a {
        font-size: 1rem;
        padding: 0.5rem 1rem;
        /*Faz com que o link ocupe toda a largura disponível do seu
          contêiner. Isso é útil para menus em telas pequenas, pois
          permite que o link fique em sua própria linha e seja fácil
          de clicar.*/
        display: block;
    }
}
```

Comparando os Menus

- Menu não responsivo:

- Menu responsivo:



Atividade Prática 11

1. Criar uma página chamada menu.html contendo:
    1. Um menu horizontal com pelo menos 5 itens.
    2. Efeito de destaque ao passar o mouse nos itens do menu.
    3. Uso de pseudo-elementos para adicionar um ícone ou marcador antes dos itens.
    4. O menu deve estar estilizado para se adaptar a diferentes tamanhos de tela.
    5. Ampliação: Crie uma página para promover uma ONG da sua região, utilizando Menu de Navegação e Trabalhando com Pseudo-Elementos. O menu deve estar estilizado para se adaptar a diferentes tamanhos de tela.
2. Cada aluno deve produzir um relatório de 1 a 3 páginas contendo:
    - Resumo teórico: explicação sobre menus e pseudo-elementos no CSS.
    - Código-fonte comentado: explicação de cada estilização aplicada.