INSTITUTO FEDERAL
Sul-rio-grandense | Câmpus Pelotas

LPW - Linguagem de Programação para WEB

CSS

Aula 9

Professor Sérgio Rodrigues

tsi tecnologia em sistemas para internet

CSS

Construção Layout

Construção Layout

Primeiramente, vamos iniciar com os estilos que reconfiguram definições padrão dos navegadores. Cada navegador possui sua própria folha de estilos com suas especificações de fonte, margens, bordas e outros. Ver reset.css

```
/*
-- Padrões e Reset
*/

* {margin: 0; padding: 0;}
ul {list-style-type: none;}
img, fieldset {border: 0;}
h1, h2, h3, h4, h5, h6 {font-weight: normal;}
```

Construção Layout

Definir a largura do site para que ele não ocupe sempre toda a largura da tela. Apesar da ideia de flexibilidade ser interessante, deixar o site ocupar toda a largura da tela sempre pode causar muitos problemas de layout e legibilidade, como, por exemplo, linhas muito longas. Por isso, vamos limitar a largura do site a 960 pixels, conforme o layout:

```
/*
-- Estrutura
*/

#cabecalho, #conteudo, #rodape {
    width: 960px;
    margin: 0 auto;
}
```

Construção Layout

No rodapé fazemos o fundo desta ocupar a largura da página Simples: no código HTML, adicionamos uma div extra para conter este fundo:

```
<div id="rodape-fora">
    <div id="rodape">
        [...]
    </div>
</div>
```

No código CSS, adicionamos a cor de fundo na div que acabou de ser criada:

```
/*
-- Estrutura
*/
[...]
#rodape-fora {
    background: #808080;
}
```

Construção Layout

A primeira coisa que iremos fazer é deixar o cabeçalho com posicionamento relativo, assim criando um contexto de posicionamento na div. Isso possibilitar o posicionamento do menu a direita com precisão:

```
/*
-- Estrutura
*/
[...]
#cabecalho {
    position: relative;
}
```

Construção Layout

Também iremos flutuar a logomarca para a esquerda, para que a logo, sendo bloco, não ocupe toda a largura da div "cabecalho". Se deixarmos a logo com display: block; e não limitarmos o seu tamanho, o link irá tomar conta de todo o espaço horizontal do cabeçalho e pode acabar interferindo com a navegação. Uma das formas de limitar a largura de um elemento bloco é flutuando-o, assim este encolhe automaticamente para se adequar ao tamanho do seu conteúdo interno.

```
[...]
#logo {
    padding: 10px 0;
    display: block;
    float: left;
}

#logo img {
    display: block;
}
```

Construção Layout

Flutuar um elemento traz o problema e floats: a div "cabecalho" não irá reconhecer o tamanho da logomarca caso a flutuação não seja finalizada. Como temos a logomarca flutuada e o menu será posicionado, o cabeçalho irá sumir a não ser que façamos algo.

```
/*
-- Classes de Uso Geral
*/

.grupo::before, .grupo::after  {
    content: " ";
    display: block;
}

.grupo::after {
    clear: both;
}
```

Construção Layout

Vamos usar position para colocar o menu à direita do cabeçalho. Estou aqui usando uma medida exata em **pixels**, mas podemos usar outras unidades, como **em** ou %.

```
/*
-- Estrutura
*/
[...]
#menu-principal {
    position: absolute;
    top: 26px;
    right: 0;
}
```

Construção Layout

Agora vamos estilizar os itens de lista e os links, conforme vistos anteriormente:

```
#menu-principal li {
    display: inline;
}

#menu-principal a {
    float: left;
    font-size: 18px;
    text-decoration: none;
    color: #333333;
    margin-left: 32px;
}
```

Construção Layout

Com isso, as regras de posicionamento do menu estão feitas. No entanto, ainda falta um pouco de *feedback* para o usuário: quando passamos o mouse em cima de um dos itens, nada acontece (além da mudança do cursor do mouse) e o item atual do menu não está destacado (o que ajuda o usuário a se localizar no site).

```
[...]
#menu-principal a:hover, #menu-principal a.ativo {
    color: #0f599f;
}

#menu-principal a.ativo {
    font-weight: bold;
}
```

Construção Layout

- A área de conteúdo do site é dividida em duas partes: conteúdo principal e conteúdo secundário, definidas visualmente como colunas. Vamos utilizar float, para estilizar e posicionar os elementos.
- Novamente, como vamos flutuar elementos, colocamos a classe "grupo" no contêiner, neste caso a div "conteudo":

```
<div id="conteudo" class="grupo">
    [...]
</div>
```

Construção Layout

- Consultando o layout, iremos colocar a div "conteudo-principal" com 620 pixels de largura, enquanto a div "conteudo-secundario" terá 280 pixels.
- Vamos incrementá-las, flutuando o conteúdo principal para a esquerda, o conteúdo secundário para a direita e definindo larguras.
- Colocaremos uma margem interna na div "conteudo" para que esta não fique grudada na foto ou no rodapé. Aproveitamos também para colocar a cor de fundo correta no conteúdo secundário (apenas na Home):

Construção Layout

```
/* -- Estrutura*/

#conteudo {
    margin: 30px auto;
}

#conteudo-principal {
    float: left;
    width: 620px;
}

#conteudo-secundario {
    float: right;
    width: 280px;
}

/*
-- Estilos específicos de Seção
*/

#bd-home #conteudo-secundario {
    background: #0f599f;
    color: #FFF;
}
```

Construção Layout

- Agora existe duas colunas no site. Apesar da estilização das colunas estar correta, ainda estamos com problemas na barra lateral: o link "Saiba Mais" sumiu, por estar da mesma cor do fundo. Além disso, o conteúdo está "colado" nas bordas da div, por isso precisaremos de margens internas.

Construção Layout

```
/*
-- Estrutura
*/
[...]
#conteudo-secundario {
    float: right;
    width: 280px;
    padding: 20px;
}
[...]
/*
-- Estilos específicos de Seção
*/
[...]
#bd-home #conteudo-secundario a {
    color: #FFF;
}
```

Inserir Margem Interna

Colocar o "a" do link

Estilizando o Rodapé

- O rodapé do site está dividido em três áreas: um parágrafo com o telefone e e-mail da empresa, outro com o endereço, e uma tag *small* com as informações de *copyright*.
- A estilização desta seção será bem simples. Verificando o layout, vemos que o telefone e o e-mail, estão à esquerda. As informações de *copyright* estão alinhadas à esquerda em baixo.
- Primeiramente, vamos estilizar e flutuar os parágrafos de telefone e e-mail e o de endereço:

Estilizando o Rodapé

```css
/*
-- Estrutura
*/
#telefone-email {
    float: left;
    font-size: 16px;
    color: #FFF;
    margin: 12px 0;
}

#telefone-email span.ddd {
    font-size: 0.8em;
}

#endereco {
    float: right;
    text-align: right;
    color: #FFF;
    font-size: 14px;
    margin: 12px 0;
}
```

Estilizando o Rodapé

- Veja como a div se reduziu à sua altura mínima porque os floats não foram finalizados. Quanto aos estilos em si, definimos tamanho, alinhamento e cor de fonte, além de algumas margens. Vamos resolver isso:

```
#copyright {
    font-size: 11px;
    color: #d9d9d9;
    clear: both;
    display: block;
    padding: 1em 0;
}
```

Estilizando as Fontes

- Realizando os ajustes finais vamos utilizar algumas fontes no Body, também ajustar algumas margens, justificar o texto e dimensionar melhor o tamanho de algumas fontes.

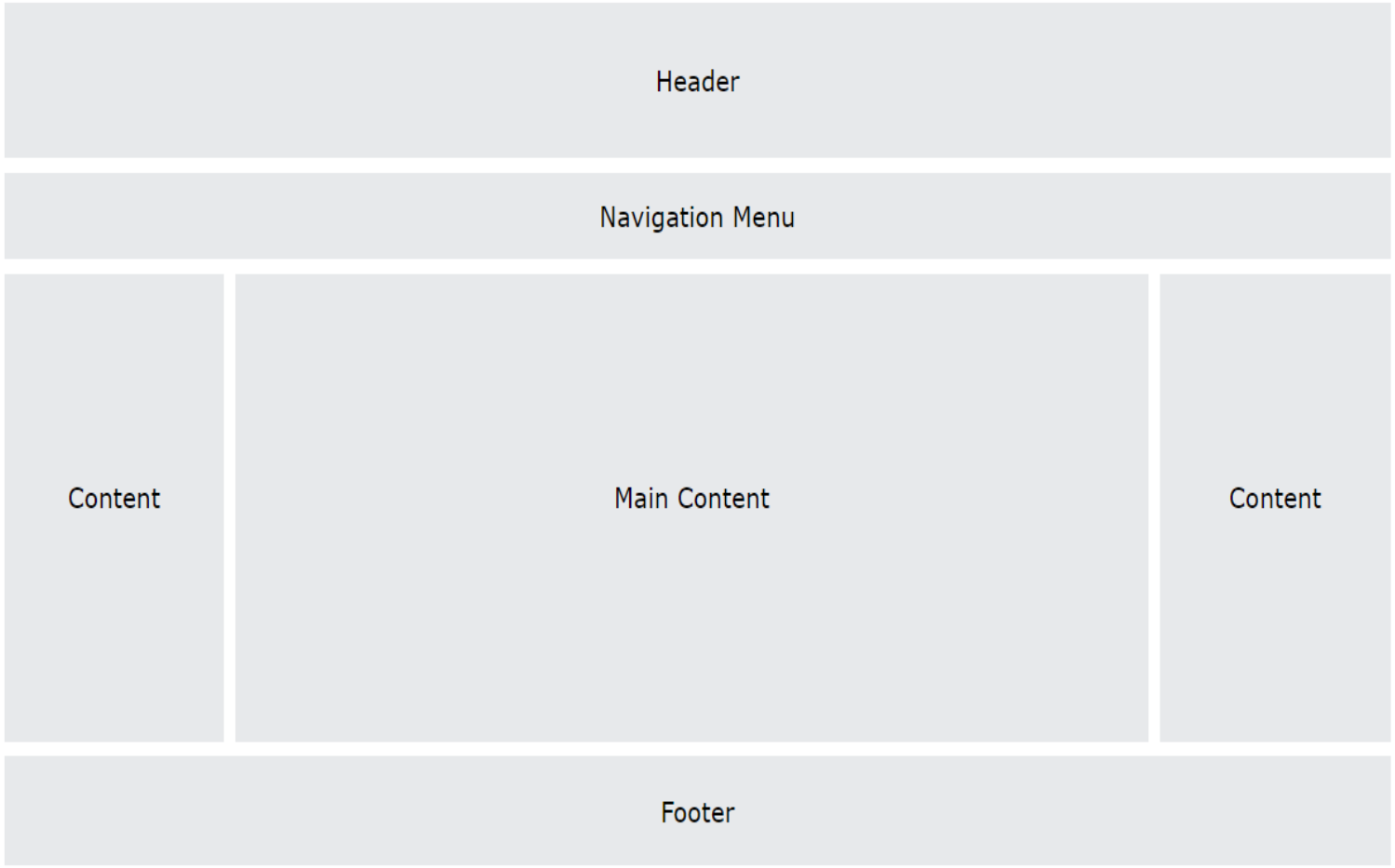
Estilizando as Fontes

```css
body {
    font-family: Candara, Verdana, sans-serif;
}

p {
    font-size: 16px;
    margin-bottom: 1.5em;
    line-height: 1.4em;
    text-align: justify;
}

#conteudo-principal h1 {
    font-size: 36px;
    margin-bottom: 0.5em;
  }

#conteudo-principal h2,
#conteudo-secundario h1 {
    font-size: 24px;
    margin-bottom: 0.5em;
}

#conteudo-secundario p {
    font-size: 14px;
}

#conteudo-principal ul,
#conteudo-principal ol {
    font-size: 16px;
    margin-left: 1.5em;
    margin-bottom: 1.5em;
}
```

TAG´s Semânticas

- Devido os novos elementos da HTML5, podemos atribuir valor semântico à estrutura do site.
- Perceba a estrutura que fizemos ao longo da disciplina: um cabeçalho, uma área de conteúdo, dividida entre conteúdo principal e conteúdo secundário, e um rodapé.
- Podemos substituir a div "cabecalho" por uma tag header e a div "menu" por uma tag nav e assim por diante.

TAG´s Semânticas



https://www.w3schools.com/css/css_website_layout.asp

TAG´s Semânticas

```
<div id="site">
    <header id="cabecalho" class="grupo">
        [...]
    </header>
    <nav id="menu-principal">
        [...]
    </nav>
</div>
```

TAG´s Semânticas

- Em seguida, na área de conteúdo, não substituímos a div "conteudo" em si, e sim as suas divs internas, "conteudo-principal" e "conteudo-secundario".
- A especificação que foi desenvolvida inicialmente pela WHATWG, comissão que concebeu a atualização da linguagem, diz que:

TAG´s Semânticas

- A div "conteudo", sendo um contêiner genérico, não deve ser substituído por section. Neste caso, usaremos section para o conteúdo principal e aside para o conteúdo secundário, desta forma:

```
<div id="conteudo" class="grupo">
    <section id="conteudo-principal">
        [...]
    </section>
    <aside id="conteudo-secundario">
        [...]
    </aside>
</div>
```

TAG´s Semânticas

- Por fim, podemos substituir a div "rodape" por uma tag footer:

```
<div id="rodape-fora">
    <footer id="rodape">
        [...]
    </footer>
</div>
```

TAG´s Semânticas

- Como pode ser observado não houve nenhuma mudança visual, por que a estrutura CSS é independente das tags HTML. O importante é que conseguimos um pouco mais de significado no nosso código HTML. A vantagem de usar as novas tags semânticas do HTML5 é que a leitura da página ficará mais fácil por programas especializados (como leitores de tela e buscadores).

- Buscador como o Google ou Bing irá ler a página e com base na semântica do código detectar onde é a navegação principal, e qual a hierarquia do conteúdo, entre outros.

Adicionando Flexibilidade ao Site

- Há algum tempo atrás, quando se construía um site, bastava planejá-lo para a resolução de tela 640×480 pixels. Depois, vieram as resoluções "padrão" 800×600 e 1024×768….
- Planejava-se o layout do site para a resolução "mais utilizada" e algo complicado, além de, levando em consideração os dispositivos portáteis.
- É tarefa de designers e desenvolvedores, criar sites que sejam compatíveis e adaptáveis com o maior número de navegadores possíveis.

Adicionando Flexibilidade ao Site

- Ao construir um site não quer dizer que tem que ser *igual* em todos os navegadores e dispositivos. Temos que considerar as restrições e capacidades de cada um e a forma de acesso.
- Assunto esse que fica fora do escopo dessa disciplina, mas que no futuro será tratado no decorrer do curso.

Adicionando Flexibilidade ao Site

- Para conseguirmos alguma flexibilidade no nosso site.
- Iremos explorar algumas das técnicas que, compiladas e difundidas por Ethan Marrote, que formam o que é chamado de Responsive Web Design.
- Alguns pré-requisitos serão necessários para um maior aprofundamento.

Adicionando Flexibilidade ao Site

- Grande parte da técnica de transformar um site em estrutura fluida e com unidades relativas.
- Requer apenas uma calculadora e a fórmula.

alvo ÷ contexto = resultado

- Esta simples conta nos leva a unidades relativas e, como consequência, independentes de resolução ou unidades fixas como pixels.

Adicionando Flexibilidade ao Site

- Por exemplo: se temos um parágrafo com texto com o tamanho de 14 pixels, e o tamanho de texto padrão é 16 pixels (como geralmente acontece nos navegadores), podemos escrever da seguinte forma:

```
font-size: 0.875em; /* o que é igual a 14 / 16 */
```

Adicionando Flexibilidade ao Site

- É fundamental no design responsivo transformar a estrutura de colunas do site e elementos como imagens em elementos que respondem ao tamanho da tela do navegador. Para isso, podemos usar **em** ou % como unidades.
- Assim trabalharemos, com conversões para **em**, com divisões por 16.

Adicionando Flexibilidade ao Site

- Redefinir a largura das divs "cabecalho", "conteudo" e "rodape" em **em´s**, desta forma: 960 /16 = 60

```
/*
-- Estrutura
*/

#cabecalho, #conteudo, #rodape {
    width: 60em;
    margin: 0 auto;
}
```

Adicionando Flexibilidade ao Site

- Porém, apenas definir a largura não torna o site flexível. Ao invés de apenas colocar uma largura, vamos especificar uma largura máxima e uma mínima, usando as propriedades CSS max-width e min-width:

```
#cabecalho, #conteudo, #rodape {
    max-width: 60em; /* 960 / 16 */
    min-width: 37.5em; /* 600 / 16 */
    margin: 0 auto;
}
```

Adicionando Flexibilidade ao Site

- Assim, quando a janela diminuir, o site irá diminuir junto, até um limite (este limite vai depender de cada site). Quando a janela aumentar, o site irá aumentar, até certo ponto.
- O max-width e min-width oferece alguma flexibilidade para o site, mas o conteúdo interno de cada contêiner também precisa se comportar de acordo.

Adicionando Flexibilidade ao Site

- Parágrafos e outros elementos que não tenham sua largura definida já irão fluir junto com a estrutura.
- O que normalmente pode ocasionar problemas são as colunas de um site (às quais normalmente definimos uma largura em pixels)
- As imagens e vídeos (tags object e embed) é simples de resolver, basta acrescentar algumas linhas de código.

Adicionando Flexibilidade ao Site

```
/*
-- Padrões e Reset
*/
[...]
img, object, embed {
    max-width: 100%;
}
```

A propriedade max-width aplicada a estes elementos faz o seguinte: se o tamanho do contêiner for maior do que o conteúdo, a imagem ou vídeo irá aumentar apenas até o seu tamanho normal (ao contrário de width: 100%; que faria com que a mídia se esticasse até preencher o contêiner).

Adicionando Flexibilidade ao Site

Quanto às colunas, vamos pegar o exemplo do nosso website: temos duas ("conteudo-principal" e "conteudo-secundario"), uma com 620 e outra com 280 pixels. Se conseguirmos uma porcentagem relativa ao contêiner "#conteudo", as colunas sempre irão manter a sua proporção! A nossa fórmula vem ao resgate: o contexto é a largura da div "conteudo" e o alvo é a largura da div que queremos redefinir.

Adicionando Flexibilidade ao Site

```
#conteudo-principal {
    float: left;
    width: 64.58333333333333%; /* 620 / 960 */
}

#conteudo-secundario {
    float: right;
    width: 29.16666666666667%; /* 280 / 960 */
    background: #0f599f;
    color: #FFF;
}
```

Adicionando Flexibilidade ao Site

O pixel é uma unidade fixa e exata de fácil entendimento. Apesar de ser importante se familiarizar com pixels, para um layout mais flexível é recomendável utilizar uma unidade relativa como o **em** ou %. Com fontes e margens em **em**, temos todos os tamanhos relativos ao tamanho da fonte do contexto (que normalmente é o padrão do site) e assim podemos aumentar ou diminuir textos e margens no site inteiro com mais simplicidade.

Adicionando Flexibilidade ao Site

```
body {
    font-size: 100%; /* 16 pixels */
}
```

Com isso, podemos basear quase todos os cálculos de fontes e margens sobre 16. Digo "quase" porque **em** é uma unidade relativa ao contexto e o contexto nem sempre é à tag body. Vamos ter como exemplo um parágrafo com 0.875em (14 pixels) de tamanho:

Adicionando Flexibilidade ao Site

```html
<p class="exemplo">
    Texto exemplo <strong>com um destaque</strong>.
</p>
```

```css
.exemplo {
    font-size: 0.875em; /* 14 / 16 */
}
```

O que acontece se colocarmos **2em** de tamanho no strong?

```css
.exemplo strong {
    font-size: 2em; /* será igual a 28 pixels, não 32! */
}
```

Atividade

1) Desenvolva as outras páginas do site obedecendo o mesmo padrão:

- Missão (home)
- Campus (lista)
- Cursos (tabela)
- Contato (formulário)

2) Utilize tag´s semânticas (HTML5).

3) Adicione flexibilidade ao site use unidades de medida (em ou %).