

5

XML

Apresentação com CSS

Helder da Rocha
(helder@argonavis.com.br)

Apresentação do XML

- *Por que apresentar os dados?*
 - *Impressão*
 - *Web*
 - *Verificação*
 - *Edição*
- *Soluções*
 - *Cascading Style Sheets (CSS)*
 - *Document Style Semantics and Specification Language (DSSSL)*
 - *eXtensible Stylesheet Language Transformations (XSLT): transforma em qualquer coisa: PDF, SVG, XSL-FO, etc.*



O que é uma folha de estilos?

- *Conjunto de regras para formatar*
 - *um documento*
 - *vários documentos*
- *Comum em*
 - *processadores de texto (Word: .DOT)*
 - *DTP (Framemaker: .FOS, Ventura Pub: .STY)*
- *Separa estrutura e conteúdo da apresentação*
 - *portabilidade maior*
 - *maior facilidade para gerar visões diferentes dos dados*
 - *facilita manutenção e promove reutilização*
- *Veja exemplos (HTML + CSS)*



- *Principais linguagens de folhas de estilo disponíveis para XML*
- **CSS**
 - *utiliza a estrutura existente do documento*
 - *regras informam ao browser como ele deve formatar cada elemento da árvore*
 - *interpretada em tempo de execução (não pode guardar o resultado da transformação)*
- **XSL (XSLT + XSL-FO)**
 - *altera a estrutura do documento (transforma)*
 - *regras informam ao processador elementos e atributos que devem ser substituídos*
 - *formatação estilo-CSS via especificação XSL-FO*
 - *interpretada em tempo de execução ou previamente (pode guardar o resultado da transformação)*



Por que usar CSS?

■ *Em HTML*

- *Alterar a forma original determinada pelo estilo do browser*
- *Separar conteúdo da forma de apresentação*
- *Simplificar a manutenção de um grande site*
- *Ter grande controle sobre cores, fontes, layout*
- *Tornar as páginas mais leves e o site mais rápido*

■ *Em XML*

- *Dar forma a elementos que **não têm forma predefinida***
- *Solução para a Web (browsers que suportam XML e CSS)*
- *Suporte: **depende do visualizador** (nem tudo que funciona para HTML em um browser funciona para XML)*
- *Aplicações como SVG e XHTML definem em especificação suporte formal a recursos do CSS (suporte é parcial em SVG)*



- *Em XML (inclusive XHTML)*

- *Folha de estilos externa*

- ```
<?xml-stylesheet type="text/css" href="abc.css" ?>
```

- *Em HTML (e também XHTML)*

- *Folha de estilos externa (use dentro de <head>)*

- ```
<link rel="stylesheet" type="text/css"
      href="abc.css">
```

- *Folha de estilos embutida (use dentro de <head>)*

- ```
<style>
 p {color: red; font-size: 12pt}
</style>
```

- *CSS aplicado em elementos individuais*

- ```
<p style="color: red; font-size: 12pt">texto</p>
```



CSS essencial: regras

- Uma folha de estilo CSS é um **conjunto de regras**. Cada regra tem a forma

```
seletor1, ..., seletorn { propriedade: valor;  
                               ...;  
                               propriedade: valor  
}
```

- Além das regras, uma folha de estilos CSS pode ter

- **Comentários:**

```
/* seletor ignorado {font: 12pt} */
```

- **Instruções:**

```
@import  
@media  
@page  
@font-face  
@charset  
@nome
```



- *Identifica um ou mais elementos*
 - *Um elemento identificado por **ID** (**xml:id** ou **id** definido pel DTD da aplicação SVG, XHTML, XSL-FO, HTML, etc.)*
 - *Um ou mais elementos determinados por*
 - *nome do elemento*
 - *atributos que contém*
 - *valores dos atributos*
 - *padrões encontrados nos valores dos atributos (predicados)*
 - *contexto hierárquico*
 - *contexto de posição no documento*
 - *modificador (pseudo-classe)*
 - *atributo de classe (vale para SVG, HTML e XHTML)*
- *Exemplo de seletores*
`h1, #id3, a[href], img[alt][src~='.jpg'] {border: solid}`



Seletores elementares

- Os seletores mais simples são **nomes** de elementos
- Definem estilo para **todos** os elementos identificados com o mesmo nome

```
titulo {color: blue;  
        font-size: 23pt;}
```

```
p { text-indent: 1.5em !important }
```

```
td, li, ul { font: 12pt sans-serif}
```

```
table { font-size: 24pt }
```

```
* {font-size: 20pt} /* seletor universal*/
```

- Estilo aplicado é herdado pelos elementos filho



Resolução de conflitos

- Ao importar folhas de estilo, definir novas regras, pode haver conflitos de precedência
- Regras básicas
 - Estilos **mais específicos** predominam: ser mais específico é mais importante que chegar depois
 - Propriedades não sobrepostas são herdadas
 - Elementos filho herdam propriedades dos pais
 - exceto quando filhos definem suas propriedades (mesmo que antes das dos pais): vale a regra do 'mais específico'
 - Regra aplicada via * (seletor universal) é considerada mais específica que uma regra herdada pela estrutura
 - O seletor mais específico de todos é o **ID**



- *Usam valor do atributo para identificar elemento*

- *Exemplos*

- a [name] {color: red}
- a [name = "inicio_da_pagina"] {color: red}
- a [name ~= "pagina"] {color: red}
- a [name |= "inicio_"] {color: red}
- a [name][href~="meusite.com"] {color: red}

- *Podem ser aplicados no eletor universal*

- *Exemplo: todos os elementos que tenham atributo **name** com valor "**coisa**"*

*[name = "coisa"]



- Atributos especificados na **DTD**, ou no **esquema**, ou por **xml:id**, como sendo do tipo ID
- Podem ser referenciados diretamente usando o identificador do elemento no documento
- *Sintaxe*
 - elemento#identificador
 - #identificador *(recomendada)*
- *Exemplo:*
 - `<div id="fred"> ... </div>`
 - `#fred {color: green}`



Seleção por contexto hierárquico

- *Dois ou mais nomes correspondem a um seletor*
- *Relação ancestral-descendente*
 - `ancestral descendente {color: red}`
 - `table td, ul b {color:yellow}`
 - `body p img {border-width: 2px}`
- *Descendência direta, do tipo pai-filho*
 - `elemento_pai > elemento_filho {color: red}`
 - `table > tr > td {color: blue}`
- *Outra forma (há pelo menos **um** elemento entre ancestral e descendente):*
 - `ancestral * descendente {color: gray}`



Seleção por contexto de posição

- *Pode-se selecionar um elemento com base no seu vizinho anterior usando-se o símbolo “+”.*
- *O elemento vizinho anterior (preceding-sibling) é um elemento irmão*
- *Exemplo:*
 - `nome[id="25"] + endereço {color: blue}`
 - *seleciona o elemento <endereço> que está no mesmo nível de <nome id="25"> e imediatamente após o mesmo.*



- *Pseudo-classes permitem selecionar elementos marginais e acrescentar texto antes ou depois de um elemento.*
- *Uma pseudo-classe liga-se ao nome de um elemento através de “:”*
 - `tr:first-child {color: green}`
`/* afeta primeiro <td> */`
 - `p:before {content: "<div class='paragrafo'>"}`
`p:after {content: "</div>"}`
 - `assunto:first-child:before`
`{content: "Assuntos: "; font: bold}`
- *Em HTML, pseudo-classes alteram elemento `<a>`*
 - `a:visited`, `a:link`, `a:active`, `a:hover`



- *Classes permitem agrupar vários elementos*
 - *Depende de suporte formal por parte da aplicação XML: requer a definição de um atributo **class** (logo não funcionam com qualquer XML)*
 - *São suportadas por XHTML (e HTML) e SVG*
- *Exemplo usando classes em XHTML:*

```
<p class="padre">Eu retiro o que disse, João</p>  
<p class="grilo">Retirando ou não retirando, o fato é  
que o cachorro enterrou-se em latim</p>  
<p class="bispo">Um cachorro? Enterrado em latim? </p>  
<p class="padre">Enterrado latindo, Senhor Bispo, Au,  
au, au, não sabe? </p>
```

- *Para dar a cada parágrafo de um mesmo personagem (mesma classe) os mesmos atributos de estilo, usa-se:*

```
.grilo { color: maroon }  
.padre { color: black }  
.bispo { color: navy }
```



- *Importa outra folha de estilos*
 - *Implementa a cascata*
 - *Regras podem ser herdadas ou sobrepostas*
 - *Regras mais específicas persistem*
 - *Exemplo:* `<body><p>texto</p></body>`
 - *Aplicar estilo em body:*
`body {color: red}`
afeta p somente se p já não tiver estilo definido
- *Exemplo*
 - `@import (estilo.css);`
 - `@import (http://www.estilos.org/estilo.css);`



- *Descreve uma fonte para uso na página*

- `@font-face {
 font-family: "Charter";
 src: url("http://site/fonts/charter")
}`

- `@font-face {
 font-family: "Swiss 721";
 src: url(swiss721.pfr); /* Swiss 721 */
}`



- `titulo { font-family: "Swiss 721", sans-serif }`
`paragrafo { font-family: "Charter", serif }`



@page e @media

- **@page** *controla a aparência de mídia paginada*
- **@media** *define estilos diferentes para mídias diferentes*
 - *Opções: all, aural, braille, embossed, handheld, print, projection, screen, tty, tv*

```
@page { size: 8.5in 11in;
        margin: 1in }
@page { size: 210mm 297mm;
        margin: 2.5cm }
@media handheld {
  @page {
    size: 120px 120px;
    margin: 5px;
  }
  p {font-size: 8pt;}
}
```



Propriedades de estilo

- *Atributos que alteram a aparência dos dados, e que são aplicados aos seletores*
 - *Sempre dependem de suporte da aplicação XML e do processador*
 - *Aplicar propriedades CSS em um XML genérico qualquer **poderá funcionar** em um browser que saiba como aplicá-las*
 - *Aplicar propriedades em um XML que especifica formalmente suporte a CSS **deve funcionar** em um visualizador compatível*
- *Sintaxe de declarações*
 - *Quando usadas dentro de folhas de estilos*
{nome: valor}
 - *Quando usadas em atributos style de aplicações XML que o suportam (XHTML, SVG, etc.)*
<elemento style="nome: valor">
- **Valores** *válidos dependem de suporte da aplicação*
 - *Para browsers, aplicações XHTML e SVG, valores válidos incluem nomes, unidades, porcentagens, cores, URIs, etc.*



Propriedades de classificação

- **display**
 - *muda o papel do elemento*
 - *bloco, tabela, lista, inline, invisível*
 - *essencial para formatar XML genérico em um browser, pois elementos não têm estrutura default*
- **white-space**
 - *definem a forma de tratamento de espaços*
- **list-style-***
 - *marcadores, números, etc.*
 - *estilos para listas de tópicos*
- **content**
 - *substitui seletor por outro conteúdo*



- **font-***
 - *Alteram propriedades relativas a fontes*
- **font-family**
 - *família (tipo)*
- **font-size**
 - *tamanho, em várias unidades:* pt, cm, in, px, em, ex, pc
- **font-weight**
 - *peso:* bold, light, 100, 200, ... , 900
- **font-style**
 - *estilo de grifo:* italic, oblique
- **font-variant**
 - *variação:* small-caps
- **font-stretch**
 - *expande ou condensa a fonte*
- **font**
 - *atalho para especificar várias propriedades de uma vez*



Atributos de texto

- *text-transform*
 - capitalize, uppercase, lowercase
- *text-decoration*
 - underline, overline, blink, line-through
- *text-align*
 - left, right, justify, center
- *vertical-align*
 - baseline, top, text-top, middle, bottom, text-bottom
 - sub, super
 - porcentagem
- *text-indent*
 - *valor ou porcentagem para endentar primeira linha*
- *line-height*
 - *leading (valor ou porcentagem)*
- *letter-spacing e word-spacing*
 - *valor*



■ *Unidades*

- `rgb(255,255,255)`
- `rgb(100%,100%,100%)`
- `#rrggbb`
- `nome_de_cor`

■ *color*

- *cor do texto*

■ *background-color*

- *cor do fundo (para qualquer elemento)*
- *default:* `transparent`

■ *CSS 2 suporta também RGBA (transparência)*

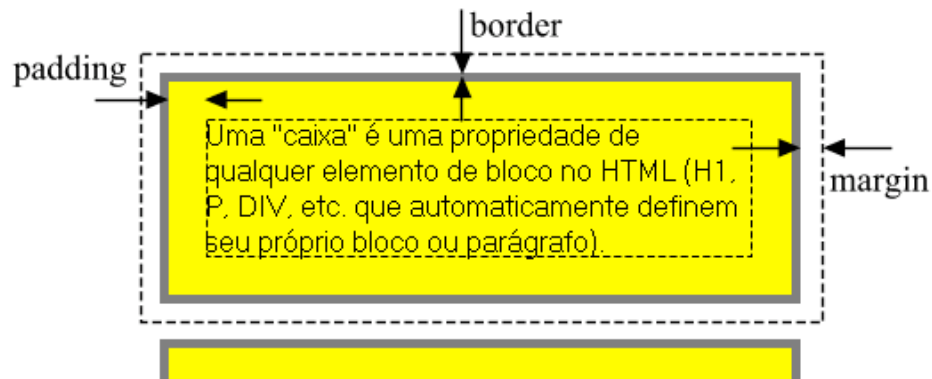
■ *SVG usa **fill** e **stroke** para preenchimento de objetos e traços, em vez de **color***



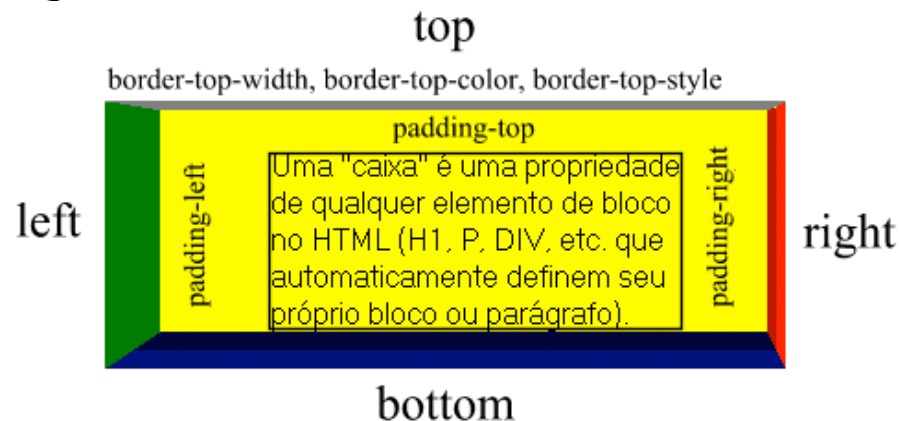
- *background-image*
 - `url(imagem)`
 - `url(http://www.imagens.org/imagem.gif)`
- *background-repeat*
 - *Como tratar a imagem de fundo*
 - `repeat`, `repeat-x`, `repeat-y`, `no-repeat`
- *background-position*
 - *Posicionamento da imagem de fundo*
 - `background-position: pos_h pos_v`
 - `pos_h`: valor, % ou `left`, `center`, `right`
 - `pos_v`: valor, % ou `top`, `center`, `bottom`
- *background-attachment*
 - `fixed` - *preso à janela*
 - `scroll` - *preso à página*
- *background (atalho)*



■ Caixa do elemento



■ Posições



Blocos: propriedades (I)

- *padding (margem interna)*
 - padding-top, padding-bottom, padding-left, padding-right
- *margin (margem externa)*
 - margin-top, margin-bottom, margin-left, margin-right
- *border-color*
 - border-top-color, border-bottom-color, border-left-color, border-right-color
- *border-style*
 - border-top-style, border-bottom-style, border-left-style, border-right-style
- *border-width*
 - border-top-width, border-bottom-width, border-left-width, border-right-width
- *border (atalho)*



Blocos: propriedades (2)

- *width*
 - *largura do elemento*
- *height*
 - *altura do elemento*
- *float*
 - *flutua para esquerda ou direita (resto do conteúdo flui)*
 - none, right, left
- *clear*
 - *quando quebrar linha quando vizinho de bloco float*
 - none, left, right *ou* both
- *visibility*
 - hidden *ou* visible



- **position**
 - absolute: *relativo ao contexto (absoluto se contexto for a página)*
 - relative: *relativo à posição anterior*
 - static: *relativo ao texto da página*
- **top**
 - *coordenada y (0 é canto superior)*
- **left**
 - *coordenada x (0 é canto esquerdo)*
- **z-index**
 - *coordenada z (layers)*

