Universidade Federal do Espírito Santo
Centro de Ciências Exatas, Naturais e da Saúde
Departamento de Computação



# Desenvolvimento Web
# Dicas sobre CSS

**Fundamentos de Programação WEB**

Site: http://www.jeiks.net/fundpweb
E-mail: jacson.silva@ufes.br

# Trabalhando com CSS

- O mesmo código pode ser exibido de diferentes formas em diferentes navegadores. Exemplo:

```
<h1>Título</h1>
<fieldset>
    <legend>Campos do formulário</legend>
    <abbr title="Hypertext Markup Language">HTML</abbr>
    <p><label>Faça a sua pesquisa:</label><input
    type="search"></p>
    <textarea>Um pouco de texto</textarea>
</fieldset>
```

- Abra esse código no Firefox e no Chrome e veja a diferença.

# Trabalhando com CSS

- Já vimos que, para que um CSS seja exibido de forma regular em todos navegadores, devemos fazer um *reset* ou fazer a *normalização* da página.
- Há vários modelos prontos para isso, como:
  - Reset CSS (Eric Meyer) <http://meyerweb.com/eric/tools/css/reset/>
  - Yahoo! <http://yuilibrary.com/>
- Mas já estamos utilizando o Normalize.css

  <http://necolas.github.io/normalize.css/>
  - Ao invés de sobrescrever diversas propriedades dos navegadores, ele apenas adéqua os pontos diferentes.
  - Adicione-o no exemplo anterior e teste-o novamente.

# Compreendendo o layout

- Todos os elementos são mantidos em caixas.
  - Utilize o inspetor de código do navegador para visualizar isso.
- Porém, o tamanho real de uma caixa (box-sizing) é igual ao seu tamanho+*padding*. Ex.:

```
.caixa {
  background-color: red;
  width: 250px;
}
.caixa-com-padding {
  background-color: green;
  width: 250px;
  padding: 0 25px;
}
```

**Crie dois elementos DIV em um HTML, cada um com um desses estilos para ver essa diferença**

# Melhorias com CSS3

- Com o CSS3, isso pode ser melhorado.
  - Permite definir o **box-sizing**
    - De: **content-box**
    - Para: **border-box**
  - O que força o navegador a respeitar estes limites.
  - Experimente adicionando este CSS no exemplo do slide anterior:

```
* {
  box-sizing: border-box;
}
```

# Novos elementos

- Agora vamos trabalhar com pseudo-elementos que cuidam do visual de nossos itens:

  ::before

  ::after

  - Eles estão disponíveis para todas as *tags* adicionadas ao **body** da página.
  - Eles vivem antes e depois do **conteúdo** de uma tag (e não antes e depois da tag em si).
- Exemplo:
  - Crie o HTML:

```
<section>
    <h1>Utilizando pseudo elementos.</h1>
</section>
```

# Novos elementos

- **E o seguinte CSS:**

```
section {
  border: 1px solid #000;
  height: 100px;
  margin: 40px auto;
  width: 400px;
}
h1 {
  background-color: #990000;
  color: #FFF;
  font-size: 1.2em;
  left: -10px;  padding: 5px 0;
  position: relative;
  text-align: center;
  width: 420px;
}
```

- Depois, adicione este:

```
h1::before {
  border: 5px solid #7C0000;
  content: "";
  left: 0;
  position: absolute;
  top: -10px;
}
```

- Vamos melhorar com:

```
h1::before {
border-color:
    transparent #7C0000 #7C0000 transparent;
border-style: solid;   border-width: 5px;
content: "";           left: 0;
position: absolute;    top: -10px;
}
h1::after {
border-color:
    transparent transparent #7C0000 #7C0000;
border-style: solid;   border-width: 5px;
content: "";           right: 0;
position: absolute;    top: -10px;
}
```

# Criando caixas de ajuda

```
<p class='help'>
Preencha o campo com um e-mail válido,
assim poderemos entrar em contato com você
para eventuais problemas.
</p>
```

```
.help {
  background-color: #F1EFE6;
  border: 1px solid #D3CDAE;
  font-size: 0.9em;
  padding: 10px;
  position: relative;
  width: 300px;
}
```

```
.help::before {
   border-color:
      transparent #D3CDAE
      transparent transparent;
   border-style: solid;
   border-width: 14px;
   content: "";
   left: -28px;
   margin-top: -14px;
   position: absolute;
   top: 50%;
}
```

# Criando Conteúdos

- HTML:

```
<p class='tip'>Você sabia que...</p>
```

- CSS:

```
.tip::before {
  content: "\261E";
  margin-right: 10px;
}
```

# Adicionando mais estilos

- HTML:

```
<blockquote>
O problema com citações na
Internet é que você não pode
confirmar a sua veracidade.
</blockquote>
```

- CSS:

```
blockquote {
  color: #444;
  font-style: italic;
}
```

```
blockquote::before,
blockquote::after {
   color: #000;
   font-size: 3em;
}

blockquote::before {
   content: "\201C";
}

blockquote::after {
   content: "\201D";
}
```

# A regra @media

- A regra @media é usada para aplicar estilos diferentes para tipos/dispositivos diferentes.
- Existem formas de verificar diversas coisas, como:
  - largura e altura da janela de visualização;
  - largura e altura do dispositivo;
  - orientação (o tablet/telefone está no modo paisagem ou retrato?);
  - resolução.
- A capacidade de fazer isso se chama Consultas de Mídia:
  - Técnica popular;
  - Permite fornecer uma folha de estilo personalizada para diversos dispositivos.
- Também pode-se especificar que determinados estilos são apenas para documentos impressos ou para leitores de tela (tipo de mídia: impressão, tela ou fala).

# Sintaxe CSS

```
@media not|only mediatype and (mediafeature and|or|not mediafeature)
{
  Código CSS;
}
```

- Onde:
  - **not**: inverte o significado de uma consulta de mídia inteira.
  - **only**: impede que navegadores mais antigos que não suportam consultas de mídia com recursos de mídia apliquem os estilos especificados. Não tem efeito em navegadores modernos.
  - **and, or, not**: combina um recurso de mídia com um tipo de mídia ou outros recursos de mídia.
- São todos opcionais.
  No entanto, se você usar **not** ou **only**, também deverá especificar um tipo de mídia.

# Tipos de mídia

- Os tipos de mídia são:

  **all** (padrão): Utilizado para todos os tipos de dispositivos;
  **print**: usado pelas impressoras;

  **screen**: usado para telas de computadores, tables, smart-phones, etc.
  **speech**: utilizados por leitores de telas que leem a página em voz alta

# Opções

- Exemplos de diferentes folhas de estilos para diferentes mídias:

```
<link rel="stylesheet"
  media="screen and (min-width: 900px)"
  href="widescreen.css">

<link rel="stylesheet"
  media="screen and (max-width: 600px)"
  href="smallscreen.css">
```

# Recursos que podem ser utilizados

- Consulte a lista em:

  https://www.w3schools.com/cssref/css3_pr_mediaquery.asp

# Trabalhando com estilos de acordo com o tamanho da tela

- Exemplo para modificar a cor de fundo do body quando a tela do navegador for menor ou igual a 600px:

```
body {
  background-color: yellow;
}

@media screen and (max-width: 600px) {
  body {
    background-color: lightblue;
  }
  div.example {
    display: none;
  }
}
```

- Agora crie um HTML com texto e com um div que obedeça esse estilo

# Trabalhando com estilos de impressão

- Os navegadores nos permitem adicionar estilos específicos para quando uma página for impressa pelo usuário.
- Isto pode ser feito de duas maneiras:
  - Com o atributo media na tag link:

```
<link href="print.css" media="print">
```

  - Ou Criando um bloco de CSS dentro da diretiva @media print:

```
@media print {
   /* Este CSS só será aplicado quando a página for
   impressa */
}
```

- Exemplo:

```
@media print {
   * {
      background: transparent !important ;
      border-color: black !important ;
      box-shadow: none !important ;
      color: black !important ;
      text-shadow: none !important ;
   }
   a { text-decoration: underline !important;  }
   a[href]::after {
       content: ' (' attr(href) ')';
   }
   a[href^="javascript:"]::after, a[href^="#"]::after {
       content: "";
   }
}
```

# Impressão – Quebra de páginas

- Propriedades que podem ser aplicadas ao elemento:

  page-break-before;

  page-break-inside; ou

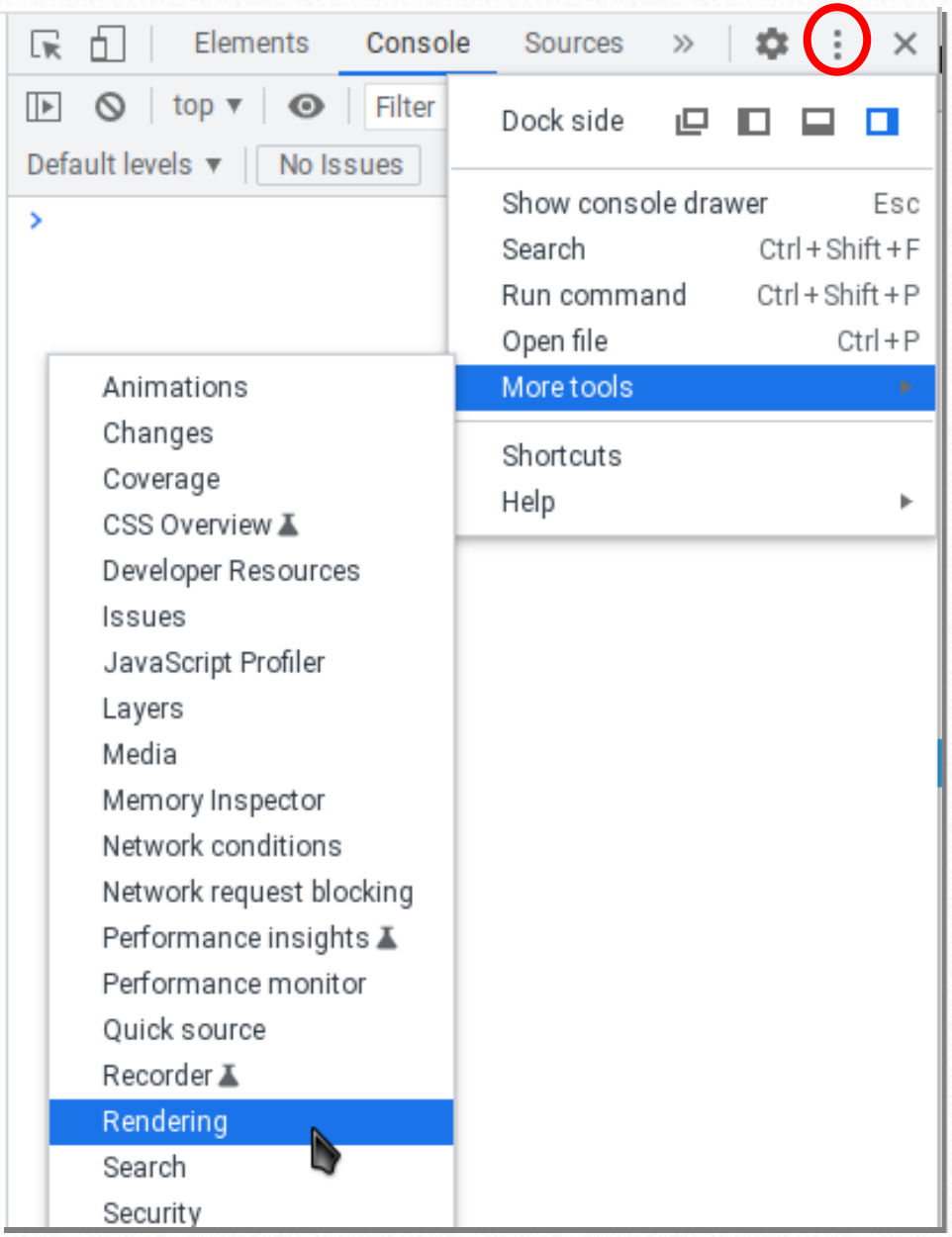
  page-break-after.

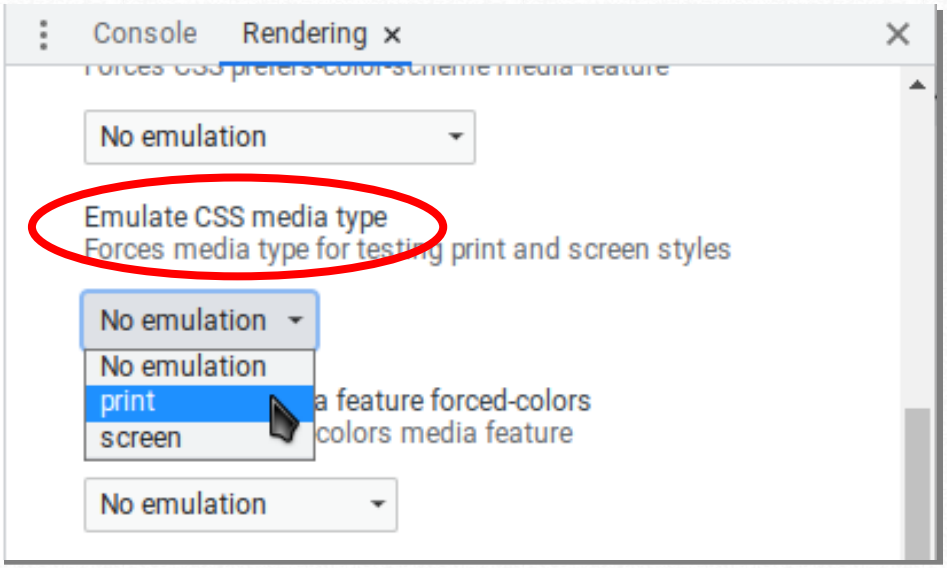
- Aceitam os valores:

  always: forçar a quebra de página;

  avoid: evitar a quebra de página.

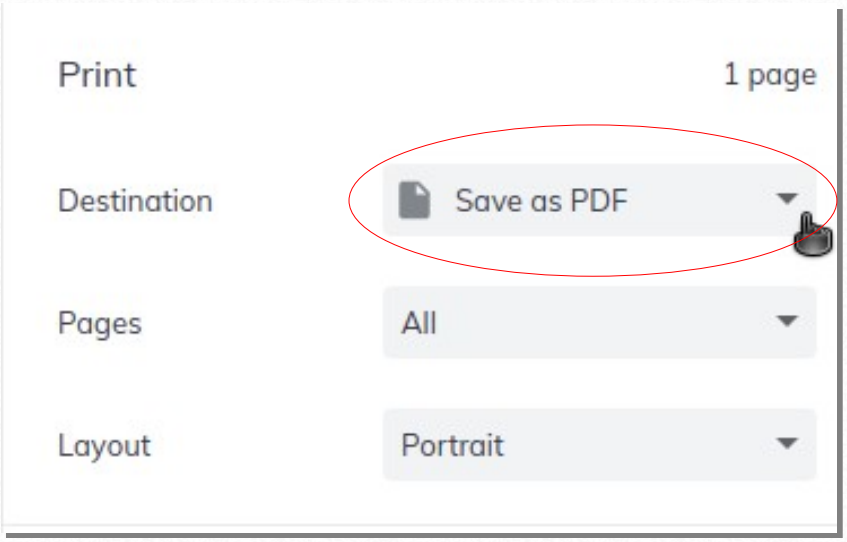
# Criando estilos em tempo real

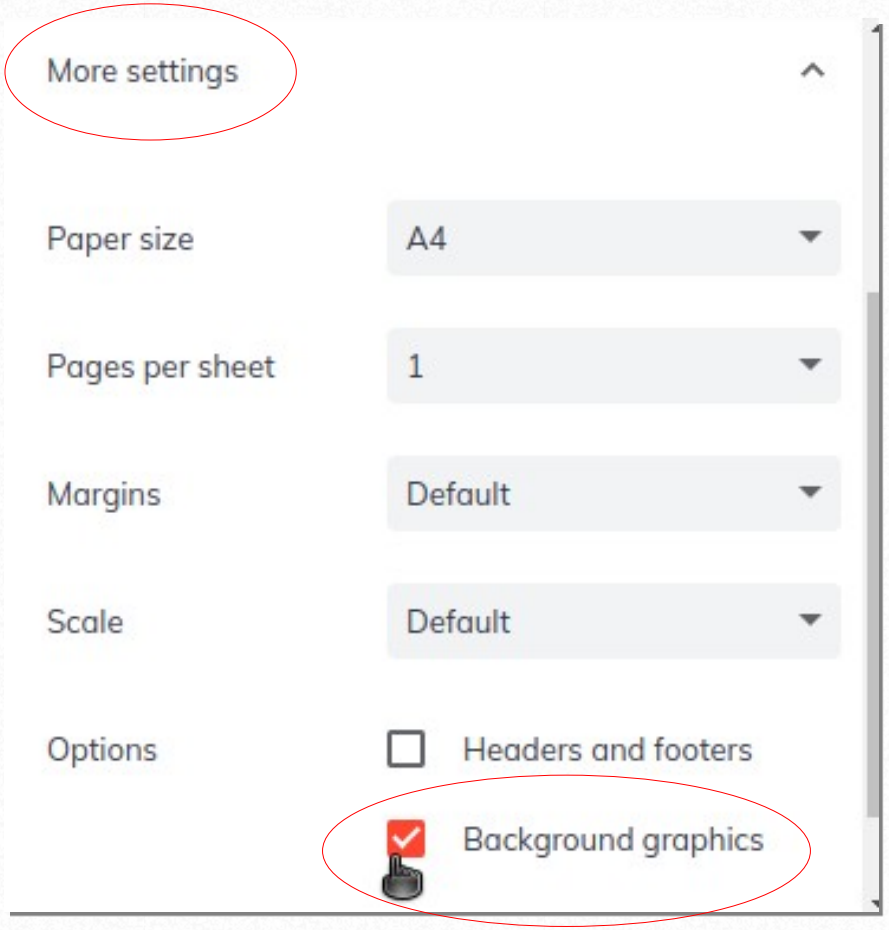


# E na hora de imprimir?

**CTRL+P e**



# Utilizar novas famílias de fontes

- Definindo a fonte:

```
@font-face {
font-family: "Lobster";
font-style: normal;
font-weight: 400;
src: local('Lobster'), url(/fonts/lobster.woff)
format('woff');
}
```

- Com a fonte definida, podemos utilizá-la normalmente:

```
h1 {
font-family: "Lobster", cursive;
}
```

# Locais para obter fontes/estilos

- Locais comuns:
  - Typekit: <https://typekit.com>
  - Google Web Fonts: <http://www.google.com/webfonts>
  - FontSpring: <http://www.fontspring.com/>
  - FontSquirrel: <http://www.fontsquirrel.com/>
- Exemplo de utilização:

```
<link href='http://fonts.googleapis.com/css?
family=Press+Start+2P' rel='stylesheet'>

h1 {
   font-family: 'Press Start 2P', cursive;
}
```

# Bordas

```
<h3 class='tnt'>TNT</h3>


.tnt {
  border-radius: 50%;
  border: 5px solid red;
  height: 50px;
  line-height: 50px;
  text-align: center;
  width: 50px;
  color: red;
}
```

# Bordas

```
<span class='counter'>07</span>
```

```
.counter {
    background-color: #000;
    border-top-left-radius: 25px 10px;
    border-top-right-radius: 25px 10px;
    color: white;          display: block;
    font-size: 1.7em;      height: 50px;
    line-height: 50px;     text-align: center;
    width: 50px;
}
```

# rgba

```
<figure>
   <img src="abstract-blue.jpg"
     alt="Abstração do Azul">
   <figcaption>
   Imagem de Wallpaper
   <br>
   <small>
      Por alguém desconhecido
   </small>
   </figcaption>
</figure>
```

```
figure {
  position: relative;
}
img {
  display:
  block;
}

figcaption {
  bottom: 5px;
  margin: 0 5px;
  padding: 5px;
  position: absolute;
  width: 300px;
  background-color:
     rgba(0,0,0,0.5);
  color: #FFF;
}
```

# Gradientes

- Exemplos:

```
.blue {
background:
  linear-gradient(to top, #4377FA, #0537B7);
}
.reverse-blue {
background:
  linear-gradient(to top, #0537B7, #4377FA);
}
.omg-pink {
background:
  linear-gradient(to bottom right, #FC0050, #FF79A3);
}
.stops {
background:
  linear-gradient(to top, #F5B951 48%, #F2A31C 56%);
}
```

# Sombras

```
<div>Caixas com sombras...</div>


div {
    box-shadow: 3px 3px 3px #AAA;
    /* box-shadow: -3px -3px 3px #AAA; */
    /* box-shadow: 2px 2px 2px rgba(0,0,0, 0.25) inset;
*/
    background-color: lightGreen;
    padding: 5px;
    width: 200px;
    text-align: center;
    font-weight: bold;
}'
```

Tente também:

```
input:focus {
 border-color: #35861B;
 box-shadow: 0 0 5px #35861B;
 outline: none;
}
```

# Sombras em Textos

```
<h1>HTML & CSS</h1>


h1 {
  background-color: #1D9AC0;
  display: inline-block;
  color: #FFF;
  padding: 10px;
  font-weight: bold;
  text-shadow: 2px 4px 2px rgba(0,0,0,0.5);
}
```

# Controlando o visual

- Três propriedades mais importantes para controlar o fluxo e a posição dos elementos:
  - display;
  - float;
  - position.

# Controlando o visual

- display:
  - Determina como um elemento será exibido pelo navegador.
- Propriedades de alguns elementos:
  - Elementos de texto, como: *a*, *span*, *small*:
    - Possuem um valor *inline* e permite que os seguintes elementos se mantenham na mesma linha.
  - Elementos de bloco, como: *div, p, section*:
    - Possuem um valor *block*, que quebra o fluxo dos elementos em uma nova linha e ocupa a largura máxima que lhe é permitida.

# Controlando o visual

- Para testar o comportamento dos elementos, basta inserir uma cor de fundo ou uma borda neles. Exemplo:

```
<span>inline</span>
<span>inline</span>
<p>block</p>

span { background-color: lightBlue; }
p { background-color: green; }
```

- Depois, adicione a propriedade **display**, com os valores:
  - **inline** para o **p**; e
  - **block** para o **span**.

Temos também os valores:
- **none**: some com o elemento;
- **inline-block**: mistura as características do inline e do block.

# Controlando o visual

- float:
  - Cuida do alinhamento e posicionamento de elementos;
  - Teve origem na diagramação de textos, no mundo da mídia impressa.
  - Na web essa técnica evoluiu e soluciona diversos casos de posicionamento de elementos.
  - Seus elementos são tratados como *block*, mas sem ocupar 100% de largura.
- Primeira utilização:

```
<h2>#1</h2>
<h4>500 pontos</h4>
<p>Prêmio para o campeão: Uma TV de LED 3D.</p>
h2 {
  float: left;
  margin: 0 20px 20px 0;
}
```

Se não quiser que o **p** acompanhe o **float**, adicione-o na classe **clear**.

# Controlando o visual

- O float pode também ser utilizado para colocar uma imagem do lado do texto:

```
img {
   float: left;
   margin-right: 10px;
}
```

- Agora, tente colocar um **div** do lado *direito*, um do lado *esquerdo* e outro no *meio* em sua página.

# Controlando o visual

- position:
  - O controle do posicionamento dos elementos pode ser feito com o position e com as propriedades de coordenadas *top, right, bottom* e *left*.
  - De acordo com o valor do position, as coordenadas são aplicadas de uma forma diferente.
- Valores possíveis:
  - static:
    - Padrão para todos os elementos,
    - Não afeta a posição por nenhuma das propriedades de coordenadas,
    - O navegador posiciona o elemento no seu lugar de origem.

# Controlando o visual

- Valores possíveis do position:
  - fixed:
    - Remove o elemento da sua posição original;
    - Reorganizando os elementos adjacentes, se necessário;
    - Fixa a sua posição na janela do navegador.
    - Mesmo que a página se mova, o elemento é mantido na mesma posição fixa, sobrepondo os demais elementos.
  - relative:
    - As coordenadas são aplicadas a partir do ponto original do elemento.
    - Não afeta a posição dos elementos ao seu redor.
  - absolute:
    - São aplicadas a partir do primeiro elemento pai que não tenha “position: static”;
    - São relativos a um elemento pai.

# Controlando a posição com CSS3

- Propriedades:

  column-count:

  - Quantidade de colunas que seu elemento deve ter;

  column-gap:

  - Espaçamento entre colunas;

  column-rule:

  - A borda que dividirá as colunas;

  column-fill:

  - Utilizado para distribuir o conteúdo entre as colunas.

# Controlando a posição com CSS3

- Crie um parágrafo com bastante texto e utilize o seguinte CSS:

```
p {
   column-count: 3;
   column-gap: 10px;
   column-rule: 1px solid #666;
   font-size: 0.9em;
   width: 500px;
}
```