História

Formatar informações dos sites não é algo novo. Por volta de 1970, no começo da trajetória do SGML (Standard Generalized Markup Language - Padrão Generalizado de Linguagens de Marcação), vários browsers já personalizavam as aparências dos documentos, cada um com seu estilo próprio.

Hakon Wium Lie estudava e percebia as dificuldades que se tinham ao desenvolver um site e resolveu criar uma maneira fácil para formatar a informação do HTML. Foi aí que ele propôs a criação do CSS ou Cascading Style Sheets. Esse era o ano de 1994.

Como a web não tinha sido projetada para desenvolver os criativos ambientes gráficos que temos atualmente, naturalmente, os recursos de desenvolvimento eram limitados e os criadores faziam das tripas coração para criar seus sites. Entre as muitas idéias que surgiram para ultrapassar ao ambiente de "apenas texto" da internet, estava aquela de utilizar tabelas de HTML para posicionar os elementos no layout, utilizando slices de imagem, gifs transparentes e a técnica de aninhamento de tabelas para contornar os problemas que os padrões proprietários traziam. A esse tipo de técnica, que foi usada pela maior parte dos websites, chamamos de layout com tabelas.

HTML

Em 1980 Tim Berners-Lee propôs um projeto baseado no conceito de hipertexto denominado ENQUIRE. Este projeto foi inicialmente todo desenvolvido em linguagem Pascal. Em 1989 Tim Berners-Lee com a ajuda de um estudante do *CERN* chamado Robert Cailliau conseguiu implementar a primeira comunicação bem sucedida entre um cliente HTTP e um servidor

através da internet. Surgia então a World Wide Web. Porém o protocolo HTTP só foi implementado em fevereiro de 1993.

A sigla HTML significa *HyperText Markup Language* em português, linguagem de marcação de hipertexto. A primeira versão do *HTML* foi baseada na linguagem *SGML*. O *SGML* era utilizado para a estruturação de documentos e foi dele que o *HTML* herdou diversas tags tais como: título <h1> ao <h6>, cabeçalho <head> e parágrafo <*p*>. A maior diferença entre essas duas linguagens de marcação é que o *HTML* implementava a tag <*a*> com o atributo*href*, permitindo assim a ligação (links) de uma página a outra. Esse conceito de interligação entre documentos é a base do funcionamento de toda Web.

O HTML surgiu em 1990 e até o seu quinto ano de vida sofreu várias revisões e alterações na sua especificação. Nesta época, quem controlava o padrão era o *CERN* e a *IETF*. Após 1995, o padrão passou a ser regularizado pela *W3C*, entidade que regula os padrões Web e que será detalhada com maior profundidade posteriormente.

Em 1993, Dave Raggett propôs uma evolução do padrão *HTML*, denominada *HTML+*. Entretanto tal proposta nunca foi implementada.

O verdadeiro sucessor do *HTML* foi o *HTML* 2.0, o qual foi apresentado na primeira conferência mundial sobre *Web*, a *World Wide Web Conference*. Essa versão do *HTML* pode ser vista apenas como uma correção da versão anterior, ou seja, apenas formalizava as características do *HTML* que já estavam em uso.

Depois, Dave Raggett escreveu o *HTML 3.0* baseado em seu mais recente rascunho do*HTML+*. Entretanto, mais uma vez, a versão do *HTML* não foi implementada. Esta especificação foi superada pela versão 3.2, a qual novamente foi uma correção e providenciou total compatibilidade com a versão 2. O *HTML 3.2* implementou características tais como tabelas, *applets* e texto flutuante ao redor de imagens. Posteriormente, Raggest foi co-author do *HTML 4* e ajudou com o desenvolvimento das linguagens como *XHTML*,*XForms*, *MathML* e além de outras especificações mais modernas da *W3C*. Em dezembro de 1999 o *HTML 4.01* foi

publicado trazendo como principal premissa a compatibilidade com as suas versões anteriores através de 3 implementações:

- *Scrict* (estrita) – implementação na qual fica proibida a utilização de elementos obsoletos da linguagem;
- *Transitional* (transitória) – implementação na qual os elementos obsoletos são permitidos;
- *Frameset* – implementação direcionada para site que usam *frames*.

W3C – World Wide Web Consortium

Em agosto de 1994 foi fundada a *W3C*, um consórcio formado por instituições comerciais e educacionais, com o objetivo de definir padrões para as respectivas áreas relacionadas à*Web*. Em dezembro de 1997 a *W3C* publicou o *HTML 4* e após dois meses publicou o *XML 1*. A *W3C* reformulou o grupo responsável pelo *HTML* para criar uma "suíte de *tags XML*". O primeiro passo foi dado em dezembro de 1998 quando o grupo reescreveu o *HTML* em *XML*sem adicionar elementos ou atributos novos. Essa especificação foi chamada de *XHTML 1*.

O objetivo seguinte da *W3C* foi a reestruturação dos formulários *Web*. Em agosto de 1999 o mesmo grupo responsável pelo *HTML* publicou o primeiro rascunho da extensão dos formulários para *XHTML*. Alguns meses depois, essa "extensão dos formulários para *XHTML*" foi rebatizada de "*XForms*". Foi criado um grupo específico, responsável pelo *XForms*. Esse grupo trabalhou em paralelo com o grupo de desenvolvimento do *HTML* com a finalidade de publicar a primeira versão do *XForms* em outubro de 2003.

Enquanto isso, com a transição do *XML* completa, os membros do grupo do *HTML* criaram "a nova geração do *HTML*". Em maio de 2001 eles publicaram a primeira versão do *XHTML 1.1*.

Em julho de 2004 a *W3C* organizou um *workshop* chamado "*Workshop on Web Applications and Compound Documents*". Nesse *workshop* estavam presentes membros da *W3C* e companhias desenvolvedoras de navegadores como a fundação *Mozilla* e a *Opera*.

Neste*workshop* foi apresentado uma visão do futuro da *web* com uma evolução do padrão *HTML 4* incluindo novas características para aplicações modernas.

No workshop a *W3C* deixou claro que não seriam desenvolvidas novas tecnologias de extensões do *HTML* e *CSS* que não fossem criadas pelo atual grupo de trabalho da *W3C*. Diante desse impasse, o grupo que desenvolvia o *HTML* e os formulários *HTML* tinha duas escolhas: ou se uniam à *W3C*, ou se separavam. Dessa forma, em julho de 2004 surgiu o*WHATWG*.

Os padrões W3C

A Web foi criada para ser um ambiente onde fosse possível trocar informações livremente, e que essas informações pudessem ser acessadas ao redor do planeta por qualquer pessoa. Em 1994, foi criado o W3C (World Wide Web Consortium): um consórcio internacional, onde são desenvolvidas os padrões para a web (Web Standards) tais como: HTML, CSS, XML, XSLT, entre outros.

Em 1994 Håkon Wium Lie propôs a criação do CSS ou Cascading Style Sheets como vimos no tópico anterior. E em 1995 apresentou sua proposta e finalmente, o W3C – World Wide Web Consortium – que estava acabando de nascer, se interessou pelo projeto e resolveu criar uma equipe, obviamente liderada por Håkon e Bert Bos. O resultado apareceu logo, em 1996, eles lançaram a recomendação oficial pelo W3C do CSS Level 1 (CSS 1).

Dois anos depois, no dia 12 de Maio de 1998, eles lançaram a recomendação do CSS de nível 2. A segunda versão das Folhas de Estilo para web.

Naquela época, no mercado de browsers, as opções ainda eram poucas: consistiam apenas em Lynx, Mozaic e Netscape Navigator, da Netscape Communications, então liderada por James Clark. A Microsoft, de Bill

Gates, resolveu entrar nesse mercado lançando o Internet Explorer. A partir daí, o Netscape e o Internet Explorer começaram a travar uma guerra atrás de adeptos. A concorrência entre os dois browsers é chamada até hoje de Guerra dos Browsers. Durante essa "guerra", os padrões do W3C ainda eram meros rascunhos. Por conta disso, as duas empresas que não podiam esperar que esses rascunhos ficassem prontos começaram a lançar seus browsers com padrões proprietários.

Agora o impasse: Os browsers tinham seus próprios padrões… Já os desenvolvedores não conseguiam criar um único código que funcionasse nos dois navegadores. Por este motivo, eles eram obrigados a desenvolver, na maioria das vezes, para apenas um browser.

Isso trouxe mais um problema, agora para os usuários. O usuário que usava Netscape, não conseguia acessar sites que eram feitos para Internet Explorer, e vice-versa.

Hoje em dia, o nível de compatibilidade entre os browsers é muito parecido, de forma que se você implementar algo específico em um browser, é muito provável que em outro browser esteja igual.

Portanto, o desenvolvedor pode ficar tranquilo quanto a maioria dos problemas causados por diferenças entre browsers.

Os sites que seguem os Padrões Web utilizam uma metodologia de desenvolvimento baseado em 3 camadas, são elas:

- 1. Informação – A informação do site é exibida utilizando código HTML.
- 2. Formatação – O HTML que exibe a informação é formatada com CSS (Folhas de Estilo). É com CSS que comandamos todo o visual do site. Tudo que é visual e decorativo deve ser feito por CSS.

- 3. Comportamento – Definida por Javascript. É a camada que define como os elementos irão se comportar de acordo com as ações do usuário.

WHATWG – Web Hypertext Application Technology Working Group

O *WHATWG* é o grupo de trabalho tecnológico de aplicações de hipertexto para *Web*. Trata-se de um grupo livre, não oficial e de colaboração dos desenvolvedores de navegadores e de seus interessados. O *WHATWG* desenvolve especificações baseadas no *HTML* e em tecnologias relacionadas para facilitar o desenvolvimento e compatibilidade das aplicações web, com a intenção de submeter os resultados para um padrão organizacional. Essas submissões podem então fazer parte do padrão formal do *HTML*.

O *WHATWG* trabalhou por muito tempo somente por email e alcançaram seu maior feito com os formulários *HTML 4* que passou a incorporar tecnologias dos autores do grupo, sem quebrar o padrão de compatibilidade com o conteúdo existente. O grupo foi criado para assegurar que o futuro desenvolvimento destas especificações fosse completamente livre, através de arquivos públicos e uma lista de discussão aberta.

Por muitos anos o *W3C* e o *WHATWG* ignoraram um ao outro. Enquanto o *WHATWG* focava nos formulários e nas novas características no *HTML*, o *W3C* estava ocupado com a versão 2.0 do *XHTML*. Mas em outubro de 2006, ficou claro que a *WHATWG* estava vivendo um momento mais sério, enquanto o *XHTML 2.0* estava sendo lançado como rascunho e não estava sendo implementado pela maioria dos navegadores. Em outubro de 2006, Tim Berners-Lee, o fundador da W3C, anunciou que a *W3C* deveria trabalhar em conjunto com a *WHATWG*. O primeiro passo do trabalho em conjunto foi renomear o “*Web Application 1.0*” para “*HTML5*”.

O que é HTML?

HTML é uma linguagem de marcação (não de programação), e uma linguagem de marcação é um conjunto de tags de marcação, simples assim…

Mas e as tags?

As tags são estruturas da linguagem de marcação que contém instruções para que os navegadores (e alguns dispositivos) possam renderizar uma página, elas normalmente vêm em pares, como **<p>** e **</ p>,** e geralmente são palavras em inglês ou simplesmente abreviações de palavras da língua norte-americana.

A primeira é a tag de início, a segunda é a tag final , a tag final é escrita exatamente como a tag inicial, mas com uma barra antes do nome, a barra representa o fechamento da tag e o encerramento daquela determinada marcação.O Início e fim das tags também são chamados de tags de abertura e fechamento.

O código abaixo mostra a estrutura básica do HTML.

```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>Minha primeira pagina em HTML</title>
  </head>
  <body>

    <h1>Meu primeiro título</h1>

    <p>Meu primeiro parágrafo</p>

  </body>
</html>
```

O que vemos no Exemplo?

- A declaração DOCTYPE define o tipo de documento ;
- As informações entre <html> e </html> vão gerar uma página web ;
- As informações entre <head> e </head> formam o cabeçalho da página;
- As tag <meta> são informações mais técnicas que você vai ver em aula;
- O texto entre <title> e </title> é exibido como título da página;

- As informações entre <body> e </body> formam o conteúdo da página visível ;
- O texto entre <h1> e </h1> é exibido como um título de conteúdo;
- O texto entre e é exibido como um parágrafo.

Teste o código acima. digite ele usando seu editor de texto favorito e salve com a extensão .html, após abra o arquivo no seu navegador.

Seu navegador deverá exibir:

Meu primeiro título

Meu primeiro parágrafo

Parabéns agora você fez a sua primeira página para internet. Que seja a primeira de muitas!

Título de uma Página

Você aprendeu as partes e função das tags em um documento HTML.

Agora vamos aprender a criar um documento HTML simples.

Em qualquer documento HTML, você usa uma tag chamada HTML no começo do documento.

Esta tag informa ao browser que o documento é um documento formatado em HTML.

A criação de um documento HTML envolve a criação do cabeçalho e o corpo do documento.

Você usa a tag HEAD para criar o cabeçalho de um documento HTML.

A sintaxe desta tag é exibida abaixo:

```
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Minha primeira pagina em HTML</title>
</head>
```

A parte do cabeçalho de um documento HTML contém o **titulo** da página Web.

Para especificar o título, você usa a **tag TITLE** entre a de inicio e de finalização da tag HEAD. As matas tag's servem para informar ao seu navedor muita coisas, dentre elas a localização de arquivos externos e o padrão de codificação de seu texto.

Outros elementos do HTML

Veremos agora, os elementos de HTML mais utilizados e os exemplos de como devemos aplicar os padrões web.

Títulos ou Headings

As posições são definidas com a tag <h1> até a tag <h6>.

<h1> define o título mais importante e a tag <h6> define o título menos importante.

Por padrão a tag <H1> é utilizada somente uma vez por cada página.

```
<h1>This is a heading</h1>
<h2>This is a heading</h2>
<h3>This is a heading</h3>
```

Links

O atributo **target** especifica onde abrir o documento vinculado.

Temos o" _blank", o "_self" e o "_parent".

O exemplo abaixo irá abrir o documento vinculado em uma nova janela do navegador ou uma nova aba:

```
<a href="http://targettrust.com.br" target="_blank">Visite a Target!</a>
```

Listas não ordenadas

Uma lista não ordenada começa com a tag **<ul>** e cada item da lista começa com a tag **<li>**.

Os itens da lista serão marcados com ***bullets*** (pequenos círculos pretos).

```
<ul>
  <li>Java Script</li>
  <li>CSS</li>
  <li>HTML</li>
</ul>
```

Listas ordenadas

Uma lista ordenada começa com a tag **<ol>**. Cada item da lista começa com a tag **<li>**.

Os itens da lista serão marcadas com números.

```
<ol>
  <li>Java Script</li>
  <li>CSS</li>
  <li>HTML</li>
</ol>
```

Faça um teste! Escreva no mesmo arquivo HTML os dois tipos de listas e veja a difrença entre elas.

Imagens

Em HTML, imagens são definidas com a tag <img>.

A tag **<img>** é vazia, o que significa que ela contém apenas atributos, e não tem tag de fechamento.

Para exibir uma imagem em uma página, você precisa usar o atributo src. Src significa "source". O valor do atributo src é a URL da imagem que você deseja exibir.

```
<img src="meugato.jpg" alt="Maru - Gato de estimação" height="42" width="42">
```

Tabelas

Definimos tabelas com a tag **<table>**.

As tabelas são divididas em linhas chamadas de **table rows** com a tag **<tr>**.

As linhas são divididas em “celulas” chamadas **table data** com a tag **<td>**.

As linhas das tabelsa também podem ser divididas em títulos chamados de **table headings** com a tag **<th>.**

```
<table border=”1”>
  <tr>
    <td>Nome</td>
    <td>Sobrenome</td>
    <td>Idade</td>
  </tr>
  <tr>
    <td>Angelina</td>
    <td>Jolie</td>
    <td>39</td>
  </tr>
</table>
```

Mesmo código acima, mas agora usando a tag <th>.

```
<table border=”1”>
  <tr>
    <th>Nome</th>
    <th>Sobrenome</th>
    <th>Idade</th>
```

```
  </tr>
  <tr>
    <td>Angelina</td>
    <td>Jolie</td>
    <td>39</td>
  </tr>
</table>
```

Formulários

Formulários HTML são usados para transmitir dados a um servidor.

Um formulário HTML pode conter elementos de entrada, como campos de texto, caixas de seleção, botões de rádio, botões de envio e demais elementos como: listas de seleção, textarea, fieldset, legendas e elementos de rotulagem.

Para se criar um form usamos as tags **<form>** e **</form>**.

Elementos de entrada

O elemento mais importante de um form é o elemento <input>, ele é usado para selecionar as informações do usuário.

Um elemento <input> pode variar de muitas formas, dependendo do tipo de atributo, podendo ser do tipo campo de texto, caixa de verificação, senha, botão de rádio, botão enviar, e outros.

Os tipos mais comuns de elementos de entrada encontram-se descritos a seguir:

Campos de texto

<input type = "text"> define um campo de entrada de uma linha que um usuário pode inserir texto. Veja o exemplo abaixo:

```
<form>
  Nome: <input type="text" name="nome"> <br/>
  Sobre nome: <input type="text" name="sobre_nome">
</form>
```

Campos de Senha

<input type = "password"> define um campo de senha

```
<form>
  Senha: <input type="password" name="senha">
</form>
```

Botões de Rádio (Radio Buttons)

<input type = "radio"> define um botão de rádio. Os botões de rádio permitem que o usuário selecione **somente uma** de um número limitado de opções:

```
<form>
  <input type="radio" name="sexo" value="masculino"> Masculino <br/>
  <input type="radio" name="sexo" value="feminino"> Feminino
</form>
```

Caixa de seleção (Checkbox)

<input type = "checkbox"> define uma caixa de seleção. As caixas de seleção permitem que o usuário selecione **nenhuma ou mais** opções de um número limitado de opções.

```
<form>
    Qual você gosta mais? <br/>
  <input type="checkbox" name="veiculo" value="carro"> Carro <br/>
  <input type="checkbox" name="veiculo" value="moto"> Moto
</form>
```

Lista de seleção (SELECT)

<select> define o início de uma lista de seleção. As listas de seleções permitem que o usuário selecione **uma das** opções de um número limitado de opções.

```
<form>
   Qual você gosta mais? <br/>
  <select name="veiculos">
     <option value="carro"> Carro </option>
     <option value="moto"> Moto </option>
     <option value="onibus"> Ônibus </option>
  </select>
</form>
```

Caixas de texto (TEXTAREA)

<textarea> define o início de uma caixa de texto. As caixas permitem que o usuário escreva um texto com múltiplas linhas.

```
<form>
   Deixe seu comentário <br/>
  <textarea name="comentario" rows="10" cols="30">
   Escreva aqui o seu comentário...
  </textarea>
</form>
```

Botões (BUTTON)

<input type = "button"> define um botão

```
<form>
  <input type="button" name="botao" value="Botão Legal">
</form>
```

Botões de envio (SUBMIT)

<input type = “submit"> define um botão de envio.

Um botão de envio é usado para enviar dados de formulário para um servidor.

Os dados são enviados para a página especificada no atributo action do formulário. O arquivo definido no atributo action geralmente faz algo com a entrada recebida:

```
<form name="pesquisa" action="pesquisa.php" method="POST">
  <input type="submit" value="Enviar">
</form>
```

CSS

Cascading Style Sheets (CSS) é uma linguagem de folhas de estilo que surgiu na época do HTML 4.0 e é utilizada para definir como os elementos de uma linguagem de marcação, como o próprio HTML ou XML, serão apresentados visualmente pelos navegadores.

A vantagem é que o CSS é bem mais robusto que o HTML para estilização destes elementos e isto aumenta a possibilidade de criar layouts mais atrativos visualmente e por consequência aumentando o poder comercial do site.

Agora você vai aprender como utilizar o CSS para controlar o estilo e layout de várias páginas em HTML ao mesmo tempo.

Como tudo começou:

O HTML nunca foi destinado a conter tags para a formatação de documentos web, ele foi concebido para definir o conteúdo de um documento, como:

<h1> Este é um título </ h1> ou <p> Este é um parágrafo. </ p>

Quando tags como <font>, e atributos de cor foram adicionados no HTML 3.2, começou um pesadelo para os desenvolvedores web. O desenvolvimento de grandes web sites, onde fontes e informações de cores foram adicionadas a cada página individualmente, tornou-se um processo longo e caro. Para resolver este problema, o World Wide Web Consortium (W3C) criou o CSS.

Quando o HTML 4.0 surgiu, toda a formatação pode ser removido do documento HTML, colocada em um arquivo separado com extenção .css , as famosas folha de estilo.

Atualmente todos os browsers suportam CSS.

Criando uma folha de estilo

Existem três maneiras de aplicarmos nossas formatações em CSS, duas delas não devem ser usadas por questões de boas práticas, mas é importante você ter conhecimento de todas , pois certamente você vai se deparar com elas durante em algum momento dos seus trabalhos futuros.

Atributo style (Inline style)

A primeira maneira de se criar estilo em CSS é como um atributo style no próprio elemento:

```
<p style=”color:#000; font-size:12px;”>
  Olá, sou um exemplo de texto. :)
</p>
```

Uma das grandes vantagens do CSS é manter as regras de estilo fora

do HTML e usando o atributo style não fazemos isso. Por este motivo não se recomenda esse tipo de uso na prática. Além do fato que cada vez que formos usar a tag <p> temos que repetir o código de formatação dela, isso duplica o tempo de codificação de cada página e o nosso trabalho.

Tag style (Internal style sheet)

A segunda maneira de se utilizar o CSS é declarando suas propriedades dentro de uma tag **<style>**.

Como estamos declarando as propriedades visuais de um elemento no header do nosso documento, precisamos indicar qual elemento nos referimos. fazemos isso utilizando um seletor CSS que é basicamente uma forma de buscar certos elementos dentro da página que receberão os estilos visuais desejados.

Veja abaixo como funciona as regras do CSS para o seletor e seu bloco de declaração:

*No exemplo acima estamos dizendo que o texto contigo dentro da tag **<h1>** vai ficar na cor azul e o tamanho da fonte vai ser de **12px**.*

O bloco de declaração contém uma ou mais declarações separadas por ponto e vírgula. Cada declaração inclui um nome de propriedade e um valor, separados por dois pontos.

No exemplo a seguir vamos ver como se aplicaria o bloco de seleção acima dentro do código html:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset=utf-8" />
    <title>Minha primeira pagina em HTML e CSS</title>
   <style>
     h1 {color:blue; font-size:12px}
    </style>
  </head>
  <body>
    <h1>Meu primeiro título formatado por CSS</h1>
```

```
    <p>Meu primeiro parágrafo</p>
  </body>
</html>
```

Arquivo Externo (External style sheet)

A terceira maneira e ultima maneira de aplicarmos os estilos de CSS em uma página web, é através de um um arquivo externo, com a extensão **.css**.

Este arquivo externo deve conter apenas estilos em CSS e nunca deve conter informações em HTML ou outras linguagens.

Veja no código abaixo como chamamos o arquivo externo de CSS dentro do header no arquivo .HTML

```
<head>
<link rel="stylesheet" type="text/css" href="mystyle.css">
</head>
```

Seletores em CSS

Seletores CSS permitem selecionar e manipular elemento (s) HTML, eles são utilizadas para "encontrar" (ou selecionar) elementos HTML com base no ID, classes, tipos, atributos, valores de atributos e assim por diante.

Seletor por elemento (TAG)

O seletor de elemento seleciona elementos com base no nome do elemento.

Você pode selecionar todos <p> elementos em uma página como esta: (todos os elementos <p> serão centro-alinhados, com a cor de texto em vermelho)

```
p {
   text-align: center;
   color: red;
}
```

Seletor por ID

O seletor ID usa o atributo id de uma tag HTML para encontrar o elemento específico.

Um id deve ser único dentro de uma página, então só devemos usar o seletor de id quando desejamos encontrar um único elemento.

Para selecionar um elemento com um ID específico, devemos criar um ID usando o simbulo **#** mais o nome do id que desejamos.

A regra de estilo abaixo será aplicada ao elemento HTML com id = "textoPrincipal":

```
#textoCentralizado {
    text-align: center;
    color: red;
}
```

IMPORTANTE: **Não inicie** um nome de ID com um número!

Seletor por classe

O seletor ID usa o atributo id de uma tag HTML para encontrar o elemento específico.

Um id deve ser único dentro de uma página, então só devemos usar o seletor de id quando desejamos encontrar um único elemento.

```
#textoCentralizado {
    text-align: center;
    color: red;
}
```

Podemos também especificar quais elementos HTML específicos devem ser afetado por uma classe. No exemplo a seguir, todos os elementos p com a classe "textoCentralizado" serão centralizados.

```
p.textoCentralizado {
   text-align: center;
   color: red;
}
```

IMPORTANTE: **Não inicie** um nome de uma classe com um número!

Principais propriedades do CSS

Agora vamos ver propriedades mais utilizadas durante a codificação de um projeto ou site na web.

Cor de fundo

A propriedade background-color especifica a cor de fundo de um elemento.

No exemplo abaixo, aplicamos uma cor de fundo no elemento body:

```
body {
    background-color: #b0c4de;
}
```

Imagem de Fundo

A propriedade background-image especifica uma imagem para usar como pano de fundo de um elemento.

Por padrão, a imagem é repetida para que ele cobre todo o elemento, mas podemos definir como ela se repete ou mesmo fazer com que a imagem não se repita, usando a propriedade background-repeat, e os valores repeat-x, repeat-y no-repeat .

```
body {
    background-image: url("paper.gif");
    background-repeat: repeat-x;
}
```

No CSS podemos utilizar uma codificação reduzida, como no exemplo abaixo,

```
body {
    background: #ffffff url("paper.gif") no-repeat right top;
}
```

Ao usar a propriedade abreviada a ordem dos valores das propriedades é equivalente a:

- background-color
- background-image
- background-repeat
- background-attachment
- background-position

Não é necessário preencher todos os valores, mas os valores inseridos vão ser interpretados pelo navegador nesta ordem.

Formatando textos

Cores

A propriedade color é utilizada para definir a cor do texto. No CSS as cores são mais frequentemente especificadas por:

um valor HEX (hexadecimal) - como "# FF0000"

um valor RGB (red green blue)- como "rgb (255.0.0)"

nome de uma cor - como "blue"

Veja os exemplos abaixo:

```
body {
    color: blue;
}

h1 {
    color: #00ff00;
}

h2 {
    color: rgb(255,0,0);
}
```

Alinhamento de texto

A propriedade text-align é usada para definir o alinhamento horizontal de um texto.

O texto pode ser centralizado ou alinhado à esquerda ou à direita ou justificado.

Quando a text-align é definida como "justificar", as linhas são esticadas para que cada uma tenha a mesma largura, e as margens esquerda e direita são retas (como acontece nos jornais e revistas)

Veja os exemplos abaixo:

```
h1 {
    text-align: center;
}

p.data {
    text-align: right;
```

```
}

p.texto {
    text-align: justify;
}
```

Text Decoration

Usamos a propriedade text-decoration para definir ou remover decorações nos texto. Veja exemplos abaixo:

```
a {
    text-decoration: none;
}

h1 {
    text-decoration: overline;
}

h2 {
    text-decoration: line-through;
}

h3 {
    text-decoration: underline;
}
```

Text Transformation

Esta propriedade pode ser usada para transformar tudo em letras maiúsculas ou minúsculas, ou capitalizar a primeira letra de cada palavra:

```
p.maiusculo {
    text-transform: uppercase;
}

p.minusculo {
    text-transform: lowercase;
}
```

```
p.capitalizado {
    text-transform: capitalize;
}
```

Fontes

No CSS a propriedade de fonte define a família das fontes, negritos, tamanho e estilo de um texto.

Diferença entre fontes com serifa e sem serifa

Fontes

No CSS a propriedade de fonte define a família das fontes, negritos, tamanho e estilo de um texto existem dois tipos de nomes para família de fontes:

generic family - um grupo de famílias de fontes com um olhar similar (como "Serif" ou "Monospace")

font family - uma família de fontes específicas (como "Times New Roman" ou "Arial")

Dica: Em telas de computador, fontes sans-serif são considerados mais fáceis de ler do que fontes com serifa.

Familia da fonte

A família da fonte de um texto é definida com a propriedade font-family.

A propriedade font-family deve conter vários nomes de fontes como um sistema de "fallback". Se o navegador não suportar a primeira fonte, ele vai tentar a próxima fonte.

Sepre comece com a fonte que você quer, e termine com uma família genérica, assim você vai deixar o navegador escolher uma fonte semelhante na família genérica, parecida com as que você definiu primeiramente caso nenhuma destas fontes estejam disponíveis no dispositivo do usuário.

Nota: Se o nome de uma família de fonte é mais do que uma palavra, ele deve estar entre aspas, como: "Times New Roman".

Quando há mais do que uma família de fontes, devemos especificar cada uma delas separada por vírgulas, como no exemplo:

```
p {
    font-family: "Times New Roman", Times, serif;
}
```

Estilo da fonte

A propriedade font-style é usada principalmente para especificar texto em itálico.

Esta propriedade tem três valores:

Normal - O texto é exibido normalmente

Itálico - O texto é mostrado em itálico

Oblíquo - O texto é "inclinar-se" (oblíqua é muito semelhante ao itálico, mas menos utilizado e suportado)

```
p.normal {
    font-style: normal;
}
```

```
p.italico {
    font-style: italic;
}

p.obliquo {
    font-style: oblique;
}
```

Tamanho da fonte

A propriedade font-size define o tamanho do texto. Ser capaz de gerenciar o tamanho do texto é importante no design para web. No entanto, você não deve usar essa propriedade para fazer parágrafos se parecerem como títulos, ou cabeçalhos se parecem com parágrafos. Neste caso, utilize sempre as tags HTML adequadas para isso, como o <h1> até o <h6> para títulos e <p> para parágrafos.

O valor da font-size pode ser um tamanho absoluto ou relativo.

Tamanho absoluto:

Define o texto para um tamanho especificado

Não permite que o usuário altere o tamanho do texto em todos os navegadores (isto se torna ruim por questões de acessibilidade)

Tamanho absoluto é útil quando o tamanho físico de impressão (na tela) da fonte é conhecida.

Tamanho relativo:

Define o tamanho em relação a elementos e, sua volta.

Permite que o usuário altere o tamanho do texto nos navegadores. (Bom para acessibilidade)

Atenção: Se você não especificar um tamanho de fonte, o tamanho padrão para texto normal, como parágrafos, é 16px (16px = 1em).

font-size em pixels:

Definir o tamanho com pixels lhe dá controle total sobre o tamanho do texto:

```
h1 {
    font-size: 40px;
}
```

font-size em Em:

Para permitir que os usuários possam redimensionar o texto (no menu do navegador), muitos desenvolvedores usam “em” em vez de pixels.

O tamanho da unidade “em” é recomendado pela W3C.

1em é igual ao tamanho da fonte de corrente. O tamanho padrão do texto nos navegadores é 16px. Assim, o tamanho padrão de 1em é 16px.

O tamanho pode ser calculado a partir de pixels para ”em” usando esta fórmula: pixels / 16 = EM

```
h1 {
    font-size: 2.5em; /* 40px/16=2.5em */
}
```

Links

Os links podem ser decorados com qualquer propriedade do CSS (por exemplo, cor, font-family, fundo, etc).

Além disso, os links podem ser decorados de forma diferente dependendo do estado em que se encontram.

Os 4 estados dos links são:

a: link - link normal, não visitado

a: visited - link que o usuário já tenha visitado antes

a: hover - link quando o usuário passa o mouse por cima

a: active - link no momento que o usuário clica

```
/* link normal (Não Visitado) */
a:link {
    color: #FF0000;
}

/* link visitado */
a:visited {
    color: #00FF00;
}

/* Link ao passar o mouse por cima */
a:hover {
    color: #FF00FF;
}

/*link ao clicar */
a:active {
    color: #0000FF;
}
```

Ao definir o estilos para vários estados de um link, existem algumas regras que devemos seguir:

a: hover deve vir depois de a: link e a: visited

a: active deve vir depois de a: hover

Listas no CSS

Vimos no HTML, que existem dois tipos de listas, as listas não ordenadas onde os itens são assinalados com os bullets e as listas ordenadas onde os itens desta lista são assinalados com números ou letras

Com CSS, as listas podem ser decorados mais coisas, e imagens por exemplo podem ser usadas como marcadores de itens da lista. Como vemos no exemplo abaixo.

```
ul {
   list-style-image: url('teste.gif');
}
```

Teste outros exemplos de estilos:

```
ul.circulo {
    list-style-type: circle;
}

ul.quadrado {
    list-style-type: square;
}

ol.romano {
    list-style-type: upper-roman;
}

ol.alfabeto {
    list-style-type: lower-alpha;
}
```