1 NOÇÕES DE DESIGN E LINGUAGEM CSS

Web design é uma vertente do design gráfico voltada para a criação de websites e documentos para o ambiente online, valendo-se de alguns preceitos como arquitetura da informação, usabilidade, acessibilidade, layout, entre outros. O objetivo do design voltado para web é expor da forma mais ágil, atrativa e simples um produto ou informação aos usuários que visitarem aquele site.

1.1 ARQUITETURA DA INFORMAÇÃO

É a técnica empregada para dar forma a produtos e experiências de informação de modo a suportar a facilidade de acesso ao conteúdo e a usabilidade. Sem um bom entendimento do propósito e das informações que irão ser dispostas no site, não será possível realizar um bom trabalho quanto a usabilidade, acessibilidade e ao próprio layout da página (arte) em si. Boas práticas em arquitetura levam em conta, por exemplo: analisar como seria o tipo de navegação do usuário que irá acessar o site; fortalecer os mecanismos de busca dentro do próprio site para que se necessário, o usuário tenha facilidade ao fazer suas pesquisas, principalmente quando o conteúdo apresentado for extenso; rotular e particionar o conteúdo de maneira a tornar a navegação do usuário mais intuitiva e natural; evitar ao máximo que o usuário tenha que descer muitos níveis dentro do site para encontrar o que deseja. O ideal é que tenhamos sempre que possível 3 níveis de navegação. Por exemplo:

```
Home > Categorias de Produtos > Detalhes de um produto
```

1.1.1 USABILIDADE

A usabilidade é um atributo de qualidade de um produto que permite medir o grau de facilidade que um usuário tem ao utilizar uma interface. Também é empregada para se referir ao conjunto de métodos destinados a melhorar a utilização de um produto ou aplicação seja ela qual for. A usabilidade é definida em 5 dimensões:

1.1.1.1 APRENDIZAGEM

Quão fácil é para os usuários realizarem tarefas básicas no primeiro contato que têm com a interface ou web site?

1.1.1.2 EFICIÊNCIA

Depois dos usuários se tornarem experientes na utilização da interface ou website, quão rápido conseguem realizar as tarefas?

1.1.1.3 MEMORIZAÇÃO

Depois de um longo período de ausência, quão facilmente os usuários conseguem restabelecer o seu nível de proficiência?

1.1.1.4 ROBUSTEZ

Quantos erros cometem os usuários, quão graves são esses erros, e qual a facilidade com que conseguem se recuperar desses erros?

No caso de um web site, as informações são encontradas? O acesso a essas informações é simples? (Note que a usabilidade tem uma grande relação com a arquitetura da informação).

1.1.1.5 SATISFAÇÃO

Na web, a usabilidade é um fator de sobrevivência crucial. Se um website é difícil de utilizar, as pessoas desistem. Se a homepage não apresenta, de forma clara, o que a empresa tem para oferecer e o que os usuários podem fazer no site, as pessoas desistem. Se os usuários se perdem num website, desistem. Se a informação contida num website é de difícil leitura e não responde às questões-chave dos utilizadores, eles desistem. É irreal pensar que um usuário irá ler um manual para aprender a navegar em um determinado site ou plataforma. Existem numerosas alternativas online; desistir e abandonar o site é a primeira defesa que um usuário tem quando encontra dificuldades de sua utilização.

1.1.2 ACESSIBILIDADE

Quando um projeto web vai ao ar, é de responsabilidade do desenvolvedor atender as necessidades do usuário e que ele possa usar o web site de forma clara. Entre esses usuários, é possível que existam pessoas com necessidades especificas, que estejam navegando através de um dispositivo móvel, ou que dispõem de uma conexão precária com a internet. Sempre que tentamos tratar essas necessidades, aliadas ao objetivo de um projeto, falamos em acessibilidade.

Quando se trata de acessibilidade, principalmente com relação a sites governamentais, a primeira noção que nos vem à mente é um conjunto de ferramentas que facilitem a

navegação para pessoas com algum tipo de necessidade especial, por exemplo, uma pessoa cega ou que seja extremamente leiga quanto ao funcionamento da internet. Porém, acessibilidade engloba variadas ferramentas e boas práticas paralelas a essas empregadas no caso acima, como o crescimento da utilização de celulares para navegar na web, que, necessita de adaptação de conteúdo, pois se tratando de um website convencional, a navegação é dificultada por cortar o conteúdo ou apresenta-lo de forma que não propícia à navegação mobile, assim utilizando o design responsivo, que se adapta naturalmente ao tamanho de tela e capacidade do dispositivo móvel do usuário. É natural que o usuário irá optar mais comumente em retornar ao segundo site para novas visitas. Outro fator importante que está ligado a acessibilidade é a visibilidade de um site perante aos mecanismos de busca (Google, por exemplo). Para tal é necessário um uso aprofundado de algumas técnicas de programação para tornar o site mais acessível a esses mecanismos e melhorar o seu ranking nas pesquisas.

1.1.3 LAYOUT

Layout é uma palavra inglesa, que significa plano, arranjo, esquema, design ou projeto. Na área de arte gráfica, o layout é um esboço ou rascunho que mostra a estrutura física de uma página de um jornal, revista ou página na internet (como um blog, por exemplo). O layout engloba elementos como texto, gráficos, imagens e a forma como eles se encontram em um determinado espaço. O layout gráfico pressupõe o trabalho de um designer gráfico, que vai trabalhar no formato e números de página e suas margens, números de colunas de texto e outros aspectos relevantes.

O layout de uma página vai depender da criatividade e do conteúdo que vai conter. Por esse motivo, muitas vezes o cliente dá indicações precisas ao designer, para que ele possa através da arquitetura de informação, técnicas de usabilidade e acessibilidade, trabalhar no layout. Assim, o layout consiste em um rascunho, esboço ou projeto, um trabalho prévio que dá uma ideia de como será a aparência final da página em questão. Pode ser um desenho simples numa folha ou algo mais evoluído, quando o projeto já está em uma fase mais avançada.

1.2 A LINGUAGEM CSS

1.2.1 CONCEITO

Com a evolução da internet, as páginas web tinham a necessidade de atrair mais atenção do usuário, de modo que ele obtivesse uma experiência mais satisfatória na absorção do conteúdo apresentado, seja ele qual for. Uma das peças responsáveis por trazer essa melhor visualização de páginas HTML foi a linguagem CSS - Cascading Style Sheets ou folhas de estilo em camadas.

Através do CSS é possível manipular tanto visual (a maneira como são impressos os elementos no navegador), quanto estruturalmente (como o elemento se comporta em relação ao demais elementos da página) às propriedades dos elementos HTML. Abaixo vemos um exemplo de como é a estrutura CSS e outro exemplo prático do seu uso.

```
propriedade: valor;

seletor {
     propriedade: valor;
     }
```

```
color: red;
font-size: 10px;
```

1.2.2 FORMA DE APLICAÇÃO DO CSS E SUAS PRECEDÊNCIAS

O CSS pode ser aplicado as tags HTML de três formas diferentes.

- ✓ **Estilos locais (in-line):**

Neste modelo, o estilo é aplicado diretamente à TAG HTML e deve ser utilizado dentro do atributo HTML "style" junto com a TAG desejada. Esta utilização deve ser evitada de modo geral, ou ter uma utilização muito pontual, pois por ser inserida diretamente na tag não permite nenhuma reutilização, levando, por exemplo, a um arquivo mais extenso e de maior dificuldade na hora manutenção. Exemplo:

```
     <p  style="color:red;  margin-left:20px;">  Este  é  um
parágrafo vermelho</p>
```

✓ **Estilo Incorporado:**

Nesta forma de incluir CSS no HTML todos os estilos são inseridos em um único local da página. Estes CSSs devem ser incluídos sempre no cabeçalho (head)[1] da página HTML dentro de uma TAG style. Exemplo:

```
<head>
     <style type="text/css">
          p {
               color: red;
               margin-left: 20px;
          }
     </style>
</head>
```

✓ **Estilo Linkado ou Externo:**

Nesta forma de inclusão do CSS, todo o estilo é escrito em um ou mais arquivos (.css) externos a página HTML, sendo os mesmos referenciados através da TAG <link/> no cabeçalho da página. Esta forma de aplicar estilos a uma página apresenta vantagens sobre as demais por permitir que uma mesma folha de estilo seja aplicada a diversas páginas HTML, e permitindo também que estes estilos fiquem salvo no cache[2] do navegador por exemplo.

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

[1] O motivo [illegible] sim como lemos este texto, uma página HTML é lida pelo navegador de maneira descendente (de cima para baixo) de modo que ao incluirmos o CSS no topo evitamos o chamado FOUC (flash of unstyled content - lampejo de conteúdo não estilizado). Evitando assim que o browser imprima a página na tela, primeiramente sem estilo, para somente depois aplicar as correções e propriedades referentes a cada elemento.

[2] O cache pode ser compreendido como uma área de armazenamento onde dados ou processos frequentemente utilizados são guardados para um acesso futuro mais rápido, perceba que um site que você normalmente acessa carrega mais rapidamente que um site nunca acessado. Isso se deve ao fato do cache do navegador salvar a estrutura básica das páginas que você está habituado a visitar, poupando o tempo de download delas em exibições futuras. Ele salva os planos de fundo das páginas, principais links, arquivos utilizados pela página (.css, .js) e diversos outros dados dela, o que torna a navegação mais rápida.

```
p {
      color: red;
      margin-left: 20px;
}
h3{
      color:#00FFFF;
}
```

Precedência de Estilos

As três formas de aplicação de estilos que vimos anteriormente (in-line, in-page, externo) podem ser inclusas na mesma página referindo-se ao mesmo elemento. Quando isso ocorre, qual das formas de aplicação irá efetivamente estilizar o elemento? Exemplo:

```
<html>
      <head>
            <link   rel="stylesheet"   href="arquivo.css"
type="text/css"/>
            <style>
            div {
                  background-color: #000000;
            }
            </style>
      </head>
      <body>
            <div style=”background-color: #ffff00”></div>
      </body>
</html>
```

```
div {
      background-color: blue;
}
```

Para que o browser decida qual regra irá aplicar, ele segue algumas regras gerais de precedência. Abaixo segue sua ordem de relevância, onde por exemplo um estilo in-line tem mais "força" que um estilo externo.

- Estilo In-line;
- Estilo In-page;

- Folha de Estilo Externa (linkada);
- Estilo padrão do navegador[3].

1.2.3 AS TAGS HTML DIV E SPAN

Tag Div

Esta tag permite uma divisão no documento HTML, possibilitando a inclusão de tag's HTML no seu interior. Exemplo:

```
<div>
     <h3> Ultima noticia </h3>
     <p>conteúdo de um paragrafo </p>
</div>
```

Tag Span

Permite a inclusão de um estilo a um texto que não seja especificamente todo o conteúdo de um elemento HTML. Exemplo:

```
     <p> Este é um texto em <span style="font-
style:italic"> italico </span></p>
```

1.2.4 SELETORES

Quando fazemos uso de um estilo in-page ou externo na nossa página, necessitamos de uma forma de casarmos nossas regras de CSS com os elementos presentes na página. Isso se faz através do uso de seletores css, sendo os principais:

[3] Mesmo respeitando a precedência de código todo browser tem por padrão sua própria definição de propriedades iniciais para cada elemento, o que varia de browser para browser. As propriedades iniciais do browser são sobrepostas naturalmente por qualquer uma das 3 precedências apresentadas, porém vale ressaltar que “propriedade não setada" difere de um atributo igual a "zero" ou "none" pois não setado remete as propriedades iniciais do browser. Uma maneira encontrada por desenvolvedores para contornar este problema é utilizar um arquivo css para "resetar" todas as propriedades e torna-las iguais em todos browsers, zerando as propriedades iniciais do navegador.

1.2.4.1 SELETOR DE TIPO

Um seletor tipo, combina com qualquer instância de um determinado tipo de elemento. Como no exemplo a seguir onde define-se que qualquer elemento (do tipo) parágrafo no documento terá a cor de texto vermelha:

```
p {
        color:red;
}
```

1.2.4.2 SELETOR DESCENDENTE

Com este tipo de seletor, podemos escolher um ou mais elementos que estão dentro de outro elemento, ou seja, que são descendentes do elemento principal. Exemplo:

```
p span {
     color: blue;
}
```

Com isso, atribuímos a cor de texto azul a todas as TAGs span que estão dentro de parágrafos (p). É possível selecionar com ainda mais especificidade, escrevendo mais elementos, como:

```
div p a span {
     font-size: 14px;
}
```

Neste exemplo, atribuímos um tamanho de fonte de 14px(pixels) a todas TAGs span que estão dentro de link (a), situados em parágrafos (p), dentro de uma TAG div.

Não há um limite para esta utilização, porém, o uso em demasia tanto de seletores de tipo simples quanto de descendência, acaba necessitando um maior processamento do navegador para organizar a página e localizar o elemento a ser estilizado, o que por sua vez, torna a impressão da página mais lenta. Recomenda-se que ao invés de utilizar seletores de tipo em uma descendência, se use seletores de classe ou ID sempre que possível e que um seletor de descendência não possua mais que 4 níveis. Exemplo:

```
p span {
     color: blue;
}
```

Se possível, opte por estes:

```
texto span {
     color: blue;
}
```

Na página teremos:

```
     <p class=”texto”>este texto tem uma palavra em
<span>azul</span></p>
      <p class=”texto”>este texto tem uma palavra em
<span class=”span-texto”>azul</span></p>
```

1.2.4.3 SELETOR DE ID

Um seletor de ID aplica-se a um elemento especificado na página através do atributo ID=”nome_id”; todo ID é único, podendo portanto ser utilizado somente uma vez no mesmo documento. Exemplo:

Na página temos:

```
     <p id=”texto”>Meu texto com ID</p>
```

No arquivo css temos:

```
#texto {
     Font-weight: bold:
}
```

Note que dentro do arquivo css a diferenciação de um seletor de ID é feita através do caractere “#” antes do nome do seletor.

1.2.4.4 SELETOR DE CLASSE

Um seletor de classe aplica-se a todo e qualquer elemento especificado na página pelo atributo class=”nome_da_class”. A grande vantagem do uso de seletores de classe é que ela pode ser utilizada sem restrições dentro de um mesmo documento. Fazendo com que o uso de css através de classes torne o código mais “reciclável”, já que pode ser usado mais de uma vez na página permitindo que se digite menos código. Exemplo:

Na página:

```
<div class=”content”> </div>
```

No arquivo CSS:

```
.content{
     width:200px;
}
```

Note que dentro do arquivo css a diferenciação de um seletor de classe é feita através do caractere “.” antes do nome do seletor.

1.2.4.5 SELETOR UNIVERSAL

Este seletor é usado como o nome sugere para atribuir propriedades a todos os elementos da página. Ele é definido por um asterisco antes das chaves que abrem e fecham a declaração. Exemplo de arquivo CSS:

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

Recomenda-se que não se atribua muitas propriedades para este tipo de seletor pois ele também interfere no carregamento da página; normalmente é usado para zerar margens e padding (espaçamento interno) dos elementos.

1.2.4.6 PSEUDO-CLASSES

Pseudo-classes são tipos de classes especiais que NÃO são definidas pelo desenvolvedor (já são pré-definidas). São mais utilizadas para atribuir propriedades que destacam links (TAGs [a]) novos ou já visitado, para alterar as propriedades de um elemento quando o cursor do mouse estiver sobre ele entre outros. O seletor de pseudo-classe é escrito com o nome do elemento ou seletor + dois pontos + nome da pseudo-classe. Exemplo:

```
a:hover {
        color: blue;
}
```

Outras pseudo-classes de links:

1. Com o seletor *a:link*, estilizamos apenas os links não-visitados, ou seja, links no seu estado normal.
2. Com o seletor *a:visited*, estilizamos apenas links visitados, ou seja, que já foram clicados.
3. Com o seletor *a:hover*, estilizamos links quando o mouse está em cima do mesmo. Com esta pseudo-classe podemos fazer diversos efeitos interessantes. Esta pseudo-classe pode ser aplicada a qualquer elemento, não apenas links, o que a torna ainda mais útil.
4. Com o seletor *a:focus*, estilizamos links quando os selecionamos com o teclado, através da tecla Tab. Esta pseudo-classe é útil para estilizar links para pessoas que possuem habilidade limitada e não conseguem utilizar o mouse, por exemplo.
5. Com o seletor *a:active*, estilizamos um link quando o mouse está sendo clicado ou se pressionarmos a tecla “Enter”, ativando o link.

1.2.5 UNIDADES DE MEDIDA CSS

As unidades de medida de comprimento CSS referem-se a medidas na horizontal ou na vertical (e em sentido mais amplo, em qualquer direção).

O formato para declarar o valor de uma unidade de medida CSS é um número com ou sem ponto decimal podendo ser precedido por um sinal ‘-' (menos), sendo o sinal '+' (mais) o

valor "default" e imediatamente seguido por uma unidade identificadora. A unidade identificadora é opcional quando se declara um valor '0' (zero) e para algumas propriedades em especifico.

Algumas das propriedades CSS permitem que sejam declarados valores negativos para unidades de medida. A adoção de valores negativos podem complicar a formatação do elemento e devem ser usados com cautela. Se valores negativos não forem suportados pela aplicação de usuário, eles serão convertidos para o valor mais próximo suportado (e isso pode tornar-se desastroso para um layout).

1.2.5.1 UNIDADE DE MEDIDA RELATIVA

É aquela tomada em relação a uma outra medida. Folhas de Estilo em Cascata que usam unidades de comprimento relativas são mais apropriadas para ajustes de uso em diferentes tipos de mídia. (Por exemplo, de uma tela de monitor para uma impressora laser).

O valor é tomado em relação a:

em: ao tamanho da fonte ('font-size') herdado;

ex: a altura da letra x (xis) da fonte herdado;

px: ao dispositivo de exibição medida mais comumente usada);

%: a uma medida previamente definida (normalmente relacionada ao elemento pai).

1.2.5.1.1 UNIDADE DE MEDIDA – PIXEL

A unidade de medida de comprimento pixel é relativa a resolução do dispositivo de exibição (a tela de um monitor, por exemplo).

A mais simples definição de pixel seria:

Pixel é o menor elemento em um dispositivo de exibição, ao qual é possível atribuir-se uma cor.

Considere um dispositivo de exibição construído com uma densidade de 90 dpi (dpi = dots per inch = pontos por polegada). Por definição, a referência padrão para pixel é igual a um ponto no citado dispositivo. Daí pode-se concluir que 1 pixel naquele dispositivo de exibição é igual a 1/90 inch = 0,28 mm.

Para uma densidade de 300 dpi 1 pixel é igual a 1/300 inch = 0,085mm.

Assim, pixel é uma medida ligada a resolução do dispositivo de exibição.

1.2.5.2 UNIDADE DE MEDIDA ABSOLUTA

É aquela que não está referenciada a qualquer outra unidade e nem é herdada. São unidades de medida de comprimento definidas nos sistemas de medidas pela física e em fim são os conhecidos "centímetros, polegadas etc.). São indicadas para serem usadas quando as mídias de exibição são perfeitamente conhecidas.

pt - ponto

pc - pica

mm - milímetro

cm - centímetro

in - polegada

1.2.6 ESTILOS PARA FORMATAÇÃO DE TEXTO

Abaixo uma seleção com as propriedades mais comumente usadas para formatação de texto:

Color:

A propriedade color é utilizada para definir uma cor para um texto.

Em CSS, uma cor de fonte pode ser definida de três maneiras:

- Com um valor em hexadecimal, como #ffff00 (vermelho);
- Com um valor em RGB, como rgb (255,255,0) (vermelho);
- Com um nome, como red;

```
<html>
	<head>
		<style>
			.texto-1{
				color:red;
			}
			.texto-2{
				color:#0000ff;
			}
			.texto-3{
				color:rgb(0,255,0);
			}
		</style>
	</head>
	<body>
		<p class="texto-1">cor vermelha</p>
		<p class="texto-2">hexadecimal da cor azul</p>
		<p class="texto-3">RGB da cor vermelha</p>
	</body>
</html>
```

Text-align:

A propriedade text-align faz exatamente o que o nome sugere: alinha o texto da mesma forma que ocorre com os editores de texto.

Assim como na maioria dos editores de texto, existem quatro tipos de alinhamento para textos em HTML com CSS:

text-align: center: centraliza o texto.

text-align: left: alinha o texto à esquerda.

text-align: right: alinha o texto à direita.

text-align: justify: ajusta o texto ao espaço determinado, ajustando os espaços entre fontes e entre palavras.

```
<html>
	<head>
		<style>
			.esquerda{
				text-align:left;
			}
			.central{
				text-align:center;
			}
			.direita{
				text-align:right;
			}
			.justificado{
				text-align:justify;
			}
		</style>
	</head>
	<body>
		<p class="esquerda">alnhamento a esquerda</p>
		<p class="central">centralizado</p>
		<p class="direita">alinhamento a direita</p>
		<p class="justificado">este paragrafo esta usando
um alinhamento
		justificado. Isso fara com que o texto se adapte
ao container onde
		estiver inserido (note como o espacamento entre as
letras e palavras varia)</p>
	</body>
</html>
```

Text-transform:

A propriedade text-transform pode ser muito útil, principalmente na elaboração de menus.

Existem três tipos de transformação de textos que podem ser realizadas através dessa propriedade CSS:

uppercase: transforma o texto em questão para maiúsculas.

lowercase: transforma o texto para minúsculas.

capitalize: transforma cada primeira letra das palavras do texto em maiúscula.

```
<html>
      <head>
            <style>
                  .texto-1{
                        text-transform:uppercade;
                  }
                  .texto-2{
                        text-transform:lowercase;
                  }
                  .texto-3{
                        text-transform:capitalize;
                  }
            </style>
      </head>
      <body>
            <p class="texto-1">caracteres em minusculo</p>
            <p class="texto-2">caracteres em maiusculo</p>
            <p class="texto-3">a primeira letra de cada
palavra ser maiuscula</p>
      </body>
```

Letter-spacing:

Esta propriedade define o espaçamento entre as letras de um texto na página.

```
<html>
      <head>
            <style>
                  .texto-1{
                        letter-spacing:-2px;
                  }
            </style>
      </head>
      <body>
            <p class="texto-1">texto com espacamento
reduzido</p>
      </body>
```

Word-spacing:

Como o nome sugere, essa propriedade define o espaçamento entre as palavras de um texto.

```
<html>
      <head>
            <style>
                  .texto-1{
                        word-spacing:30px;
                  }
            </style>
      </head>
      <body>
            <p class="texto-1">espacamento entre palavras
ampliado</p>
      </body>
```

Text-shadow:

Especifica uma sombra para o texto. Sendo composta por dois valores obrigatórios e dois valores opcionais. O valor padrão desta propriedade é none (neste caso, sem sombra):

Sintaxe:

h-shadow: sombra horizontal (valores positivos aplicam a sombra a direita do texto, valores negativos aplicam a esquerda do texto);

v-shadow: sombra vertical (valores positivos aplicam a sombra abaixo do texto, valores negativos aplicam acima do texto);

blur: desfoque (quanto maior o valor, maior o desfoque);

color: cor da sombra;

```
text-shadow: 2px 3px 5px rgb(0,0,0);
```

Text-decoration:

Especifica uma decoração para o texto, normalmente usada para sublinhar. Possui os seguintes valores:

none: nenhum (valor padrão de texto);

underline: sublinhado[4];

overline: sobrelinhado

line-through: riscado;

[4] Todo texto dentro de uma TAG 'a' (link) tem como padrão o valor sublinhado, para retirar este padrão é só atribuir o valor none a esta propriedade.

```
<html>
	<head>
		<style>
			.texto-1{
				text-decotarion:none;
			}
			.texto-2{
				text-decotarion:underline;
			}
			.texto-3{
				text-decotarion:overline;
			}
			.texto-4{
				text-decotarion:line-through;
			}

		</style>
	</head>
	<body>
		<p class="texto-1">texto normal</p>
		<p class="texto-2">texto sublinhado</p>
		<p class="texto-3">texto sobrelinhado</p>
		<p class="texto-4">texto riscado</p>
	</body>
</html>
```

Font-Family:

Especifica uma família de fontes a ser atribuída ao documento ou determinado bloco de texto. Exemplo:

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

No exemplo, utilizamos a família “Times New Roman”[5], para aplicar esta fonte a página o navegador procura nos diretórios locais pela fonte, se esta fonte não existir no computador do usuário ela será substituída pela fonte padrão do navegador: por este motivo esta propriedade deve ser descrita sempre iniciando pela fonte desejada seguida de uma ou mais fontes genéricas similares a desejada. Garantindo assim que o navegador sempre chegue a um resultado no mínimo próximo ao desejado.

[5] Se o nome de uma família de fonte possuir mais de uma palavra deve ser descrito entre aspas.

Font-size:

Especifica o tamanho da fonte. Pode ser atribuído uma medida tanto relativa, quanto absoluta. Quando este valor não é setado pelo desenvolvedor acaba sendo orientado pelas propriedades padrões do navegador.

```
<html>
	<head>
		<style>
			.texto-1{
				/*Tamanho defindo pelo navegador*/
			}
			.texto-2{
				font-size:50%; /*Porcentagem em relacao
ao valor padrao do navegador*/
			}
			.texto-3{
				font-size:10pt; /*Tamanho definido por
pontos. 10pt equivalr a 13 px aproxximadamente*/
			}

		</style>
	</head>
	<body>
		<p class="texto-1">tamanho da fonte</p>
		<p class="texto-2">tamanho da fonte</p>
		<p class="texto-3">tamanho da fonte </p>
	</body>
</html>
```

Font-weight:

Especifica o peso do tipo (caracteres da fonte).

Pode ser atribuído através de valores como:

ligther

normal

bold

bolder

A propriedade font-weight pode ser atribuída numericamente através dos valores

100, 200, 300, 400, 500, 600, 700, 800, 900.

Sendo 400 o valor referente a normal e 700 o valor referente a bold.

```
<html>
	<head>
		<style>
			.texto-1{
				font-weight:bold;
			}
			.texto-2{
				font-weight:400;
			}

		</style>
	</head>
	<body>
		<p class="texto-1">peso da fonte em 700 ou bold</p>
		<p class="texto-2">peso da fonte em 400 ou normal</p>
	</body>
</html>
```

Font-style:

Usado para alterar o estilo de renderização da fonte. Possui três valores:

Normal: o texto e renderizado normalmente.

Itálico: o texto e renderizado em italico.

Oblique: o texto e renderizado relativamente "deitado" (valor similar ao itálico porém menos suportado).

```
<html>
	<head>
		<style>
			.texto-1{
				font-style:normal;
			}
			.texto-2{
				font-style:italic;
			}
			.texto-3{
				font-style:oblique;
			}
		</style>
	</head>
	<body>
		<p class="texto-1">estilo normal</p>
		<p class="texto-2">estilo italico</p>
		<p class="texto-2">estilo obliquo</p>
	</body>
</html>
```

1.2.7 FLUXO DA PÁGINA

Se a grosso modo fossemos quantificar o grau de importância em uma escala de 0 a 100%; o bom entendimento do conceito de fluxo de página teria uma importância de 60% no real entendimento de css. Trate-se de um conceito básico, porém as vezes pouco estudado na hora de manipularmos nossos elementos html através de CSS.

Sabemos que as páginas web são codificadas em HTML e os elementos aparecem no código em uma determinada posição. O navegador, no momento que interpreta o código HTML da página, vai imprimindo na tela os elementos de acordo com a sua ordem na página (como as palavras neste texto) e as propriedades estruturais padrão de cada TAG (como o elemento fica disposto na página).

A disposição padrão (display) dos elementos e a posição destes no código resulta na visualização dos elementos de uma forma especifica. Esta forma de disposição e visualização é o que chamamos de fluxo de página.

Com relação ao fluxo da página (disposição e visualização) existem 3 tipos elementos.

1.2.7.1 ELEMENTOS INLINE

Possuem a propriedade css display:inline; como padrão, ocupam somente a largura necessária para exibir seu próprio conteúdo. Só podem conter informações ou outros elementos “inline”. Diferente dos elementos nível de bloco os elementos inline não começam em nova linha. Ficando dispostos lado a lado, até ocuparem toda largura disponível na página.

Exemplos de elementos (TAGs) inline:

a

img

input

label

span

1.2.7.2 ELEMENTOS DE BLOCO

Possuem a propriedade css display:block; como padrão, ocupam toda largura disponível na página e criam uma linha invisível antes e depois de si próprios. Elementos de bloco sempre começam em nova linha. Um elemento que venha antes ou depois de um elemento de bloco é renderizado acima ou abaixo do elemento de bloco, nunca ao lado. Elementos de bloco podem conter elementos inline e outros elementos de bloco.

Exemplos de elementos (TAGs) de bloco:

div

h1

h2

ul

li

p (a TAG p é inicialmente usada como um “bloco” de texto)

1.2.7.3 ELEMENTOS DE BLOCO

Possuem a propriedade css display: none; como padrão, são definidos como elementos invisíveis porque estão no documento mas o usuário não os vê.

Exemplos de elementos (TAGs) de bloco:

meta

link

style

title

Todo elemento presente na página pode ter sua renderização padrão alterada através da propriedade “display”, o que interfere no fluxo da página.

1.2.7.4 DISPLAY, POSITION E FLOAT

Display, position e float são as três propriedades que mais interferem com o fluxo da página podendo altera-lo drasticamente:

1.2.7.4.1 DISPLAY

A propriedade display determina como um elemento se comporta no fluxo da página, através desta propriedade podemos alterar o modo de renderização padrão dos elementos descritos no fluxo da página. Exemplo:

```
display: block;
```

Os valores mais usados são:

Inline:

Como mencionado anteriormente, elementos com display inline se comportam como palavras em um arquivo de texto, colocando-se lado a lado até ocuparem toda largura disponível. Elementos inline são flexíveis quanto a sua altura e largura, sempre forçando a visualização de TODO seu conteúdo. Um elemento inline pode recebem propriedades como margem e espaçamento interno assim como elementos de bloco.

```
<div>
      <span>ESSE E NOSSO PRIMEIRO SPAN</span>
      <a href="">seguido de um link</a>
      <span>esse é nosso segundo span</span>
</div>
```

Block:

Elementos com display block, sempre promove uma quebra de linha no fluxo da página Um bloco contém um espaço em branco tanto em cima quanto embaixo e não permite outros elementos HTML ao lado, exceto quando tiver sido declarado o contrário (por exemplo, declarar a propriedade “float” para o elemento próximo ao bloco). São mais rígidos que elementos inline quanto a suas dimensões, principalmente quando possuírem altura e largura definidas por alguma medida especifica, o que pode ocasionar o “vazamento” do conteúdo para fora do bloco, ou, que o bloco corte parte do conteúdo que ficará escondida durante a visualização da página.

Table:

Display table força o elemento a se comportar como uma tabela html, transformando o elemento em um elemento de bloco, provoca uma quebra de linha acima e abaixo de sí próprio. A maior diferença entre um elemento com display table e outro com display block é que uma tabela sempre irá acomodar todo seu conteúdo ou elementos filhos em seu interior. Não permitindo "vazamentos" mesmo que tenha uma altura e largura definida a tabela irá crescer (se o conteúdo exceder seu tamanho) de acordo com a necessidade de seu conteúdo.

None:

Quando atribuímos esta propriedade a um elemento, ou ele já a possui por padrão, ele deixa de fazer parte do fluxo da página, atuando como se não existisse. Um ponto importante sobre display none;

Mesmo que um elemento esteja com display none ele será lido pelo navegador e influenciando o carregamento da página; Sempre que for possível remova os elementos da página, ao invés de atribuir display none a elementos que não serão usados ou não irão influenciar na renderização da página.

Inline-Block E Inline-Table:

Ambos comportam-se como elementos de bloco com relação a suas dimensões e aninhamento (relação entre elemento pai e elemento filho) porém, coloca-se como elementos inline no fluxo da página (ou seja, na mesma linha do conteúdo adjacente).

```
      <html>
            <head>
                  <style>
                        div                                    {width:50px;
height:50px;margin:0px;display:inline-table;}
                        .box-1{
                              background-color:red;
                        }
                        .box-2{
                              background-color:blue;
                        }
                        .box-3{
                              background-color:green;
                        }
                  </style>
            </head>
            <body>
                  <div class="box-2"></div>
                  <div class="box-2"></div>
                  <div class="box-3"></div>
            </body>
      </html>
```

1.2.7.4.2 POSITION

A propriedade position trabalha juntamente com propriedades de coordenadas (top, left, right ou bottom) para diagramar* e também para posicionar elementos no fluxo da página; os quatro principais valores de posicionamento são:

Position static:

Este é o posicionamento padrão que o browser atribui a todos elementos. Como o nome sugere nada acontece ao fluxo da página; permanecendo o elemento em sua posição original dentro do fluxo.

Position fixed:

Quando atribuímos um valor diferente de static ao posicionamento de um elemento, iremos precisar de um ponto de referência de onde irão partir nossas coordenadas. No caso de um position fixed, o ponto de referência será o canto superior esquerdo da tela. Se atribuirmos os valores abaixo:

```
div {
     display: block;
     position: fixed;
     top:0;
     left:0;
}
```

No exemplo acima, a div irá colar ao canto superior esquerdo da tela. Mesmo que a página seja rolada o elemento irá ficar fixo ao canto.

Position Relative:

Este valor posiciona o elemento em relação a si mesmo dentro do fluxo da página; Seguindo a regra de que todo elemento precisa de um ponto de referência para iniciar seu cálculo de coordenadas e ao contrário do que muitos pensam, esse ponto não é o ponto central do elemento, o ponto base é o canto superior esquerdo do elemento, e ele começara a contar a partir dali. Por posicionar-se em relação a sí mesmo, um elemento com position relative se comporta de maneira similar a um position static, diferindo somente no fato de que o position relative pode influenciar o posicionamento de seus elementos filhos. Quando se deseja manter

o elemento em seu local de origem no fluxo da página não há necessidade de setar coordenadas tendo em vista que o valor das mesmas é zero por padrão.

```
<html>
	<head>
		<style>
			div {width:50px;
height:50px;margin:0px;position:relative;}
			.box-1{
				background-color:red;
			}
			.box-2{
				background-color:blue;
				position:absolute;
				top:-10px;
				lef:-10px;
			}
			.box-3{
				background-color:green;
			}
		</style>
	</head>
	<body>
		<div class=”box-2”></div>
		<div class=”box-2”></div>
		<div class=”box-3”></div>
	</body>
</html>
```

Position Absolute:

Ao contrário do de um position relative, um elemento com position absolute adota como referência o canto superior esquerdo de seu elemento pai, ou do elemento diretamente anterior a ele que tiver um position diferente de static, saindo do fluxo da página. Por ficar fora do fluxo da página, elementos com este tipo de posicionamento não irão ocupar espaço dentro do elemento pai. Quando não houver nenhum elemento com position setado na árvore de elementos precedentes a este elemento, ele irá se orientar pelo body, mesmo que seja o último elemento descrito no código HTML. Só é necessário setar duas coordenadas para cada elemento. Uma no eixo X (left e right) e uma no eixo Y (top e bottom).

```
<html>
      <head>
            <style>
                  div {width:50px; height:50px;margin:0px;}
                  .box-1{
                        background-color:red;
                  }
                  .box-2{
                        background-color:blue;
                        position:absolute;
                        top:20px;
                        lef:20px;
                  }
                  .box-3{
                        background-color:green;
                  }
            </style>
      </head>
      <body>
            <div class=”box-1”></div>
            <div class=”box-2”></div>
            <div class=”box-3”></div>
      </body>
</html>
```

1.2.7.4.3 FLOAT

Float é uma propriedade que retira o elemento do fluxo da página e o força a se posicionar a esquerda ou a direita da página, dependendo do valor setado. A propriedade float deve ser atribuída a elementos com position relative ou static. Quando dois ou mais elementos estiverem flutuando, eles irão se acomodar lado a lado enquanto houver espaço, prosseguindo logo abaixo. Possui três valores:

left

right

none (padrão)

```
<html>
      <head>
            <style>
                  div                                    {width:50px;
height:50px;margin:0px;float:left;}
                  .box-1{
                        background-color:red;
                  }
                  .box-2{
                        background-color:blue;
                  }
                  .box-3{
                        background-color:green;
                  }
                  .box-4{
                        background-color:cyan;
                  }
                  .box-5{
                        background-color:yellow;
                  }
                  .box-6{
                        background-
color:black;width:80px;height:80px;float:nome;
                  }
            </style>
      </head>
      <body>
            <div class=”box-1”></div>
            <div class=”box-2”></div>
            <div class=”box-3”></div>
            <div class=”box-4”></div>
            <div class=”box-5”></div>
            <div class=”box-6”></div>
      </body>
</html>
```

A propriedade ***CLEAR*** é usada para controlar o comportamento dos elementos que se seguem aos elementos float no documento.

Por padrão, o elemento subsequente a um float, ocupa o espaço livre ao lado do elemento flutuado.

A propriedade clear pode assumir os valores left, right, both (ambos) ou none. A regra geral é: se clear, for por exemplo definido both para um box, a margem superior deste box será posicionada sempre abaixo da margem inferior dos boxes flutuados que estejam antes dele no código (ocasionando uma espécie de quebra de linha).

```
      <html>
            <head>
                  <style>
                        div                                    {width:50px;
height:50px;margin:0px;float:left;}
                        .box-1{
                              background-color:red;
                        }
                        .box-2{
                              background-color:blue;
                        }
                        .box-3{
                              background-color:green;
                        }
                        .box-4{
                              background-color:cyan;
                              float:nome;
                              clear:left;
      /*O estilo clear permite limpar a flutuação de um element que
vier antes o depois de determinado element, proporcionando uma
especie de quebra de pagina*/
                        }
                        .box-5{
                              background-color:yellow;
                        }
                        .box-6{
                              background-
color:black;width:80px;height:80px;
                        }
                  </style>
            </head>
            <body>
                  <div class=”box-1”></div>
                  <div class=”box-2”></div>
                  <div class=”box-3”></div>
                  <div class=”box-4”></div>
                  <div class=”box-5”></div>
```

1.2.8 PROPRIEDADES PARA FORMATAÇÃO DE ELEMENTOS DE BLOCO

Assim como elementos inline elementos de bloco podem receber propriedades que irão influenciar sua apresentação e comportamento:

width: especifica a largura de um elemento.

max-width: especifica a largura máxima de um elemento.

min-width: especifica a largura mínima de um elemento.

height: especifica a altura de um elemento .

max-height: especifica a altura máxima de um elemento.

min-height: especifica a altura mínima de um elemento.

A largura e altura de um elemento podem conter ao invés de uma medida absoluta ou uma medida relativa o valor auto. Este valor sugere que o elemento irá tomar um tamanho necessário para acomodar seu conteúdo.

border: ajusta a largura, a cor e estilo da borda de todos os lados de um elemento.

Sintaxe:

propriedade: largura estilo cor;

Exemplo:

```
border: 2px solid rgb(0,0,0);
```

As definições de valores podem ser atribuídas separadamente para cada lado no elemento através da adição de sufixos a propriedade:

border-top

border-right

border-bottom

border-left

Os estilos de borda mais comuns são:

solid (borda lisa)

dashed (tracejada)

dotted (pontilhada)

```
<html>
	<head>
		<style>
			div {
				width:100px;
				height:100px;
				background-color:red;
				margin:0px 0px 20px 20px;
			}
			.box-1{
				border:5px solid #ff0;
			}
			.box-2{
				border-bottom: 10px solid blue;
			}
			.box-3{
				border-bottom: 5px solid blue;
				border-top: 5px dashed black;
				border-left: 5px dotted blue;
				border-right: 5px dashed red;
			}
		</style>
	</head>
	<body>
		<div class=”box-1”></div>
		<div class=”box-2”></div>
		<div class=”box-3”></div>
	</body>
</html>
```

margin[6]: especifica uma margem para cada lado de um elemento.

padding: especifica o espaçamento interno para cada lado de um elemento.

Estas duas propriedades seguem o mesmo padrão na hora de serem aplicadas a um elemento:

- podem receber o valor auto;

- podem ter somente um valor, que será atribuído a todos os lados do elemento;

- podem ter dois valores, que serão atribuídos da seguinte forma; o primeiro valor para os lados superior e inferior e, o segundo valor para os lados direito e esquerdo;

[6] Atribuindo-se o valor “auto” para a borda esquerda e direita de um elemento ela irá automaticamente facilmente centralizar na página ou no elemento pai.

- podem ter 4 valores, que serão atribuídos aos lados do elemento no sentido horário; topo, direito, inferior e esquerdo;

- podem tratar cada lado separadamente através de sufixos como a propriedade border.

```
<html>
      <head>
            <style>
                  div {
                        width:100px;
                        height:100px;
                        background-color:red;
                  }
                  .box-1{
                        margin:0px 0px 20px 20px;
                  }
                  .box-2{
                        background-color:blue;
                        margin: 20px 100px;
                  }
                  .box-3{
                        margin-left: 50px;
                  }
            </style>
      </head>
      <body>
            <div class=”box-1”></div>
            <div class=”box-2”></div>
            <div class=”box-3”></div>
      </body>
</html>
```

Background-Color:

Atribui uma cor de fundo ao elemento, cobrindo todo seu interior; usa os mesmo valores que a cor de texto. Um background por padrão leva o valor transparente na maioria dos elementos.

Background-Image:

Atribui uma imagem de fundo ao elemento.

```
<html>
      <head>
            <style>
                  .box-1{
                        width:300px;
                        height:250px;
                        margin:20px 10px;
                        background-color:green;
                        background-
image:url(http://www.teste.com.br/imagem.png);
                  }

            </style>
      </head>
      <body>
            <div class=”box-1”></div>

      </body>
</html>
```

Background-Repeat:

Ajusta o tipo de repetição que a imagem de fundo usará. É possível que seja atribuída como imagem de fundo de um elemento uma imagem menor que o próprio elemento, isso faz com que por padrão a imagem seja repetida até cobrir todo fundo do elemento. É possível alterar este padrão com esta propriedade:

- repeat: a imagem irá repetir em ambas as direções;

- repeat-x: a imagem irá repetir somente no eixo x (horizontalmente);

- repeat-y: a imagem irá repetir somente no eixo y (verticalmente);

- no-repeat: a imagem não irá ser repetida;

```
<html>
      <head>
            <style>
                  div{
                        width:300px;
                        height:250px;
                        margin:20px 10px;
                        background-color:green;
                        background-
image:url(http://www.teste.com.br/imagem.png);
                  }
                  .box-1{
                        background-repeat:no-repeat;
                  }
                  .box-2{
                        background-repeat:repeat-x;
                  }
            </style>
      </head>
      <body>
            <div class=”box-1”></div>
            <div class=”box-2”></div>
      </body>
</html>
```

Background-Position:

Posiciona através de coordenadas onde ficará a primeira aplicação de uma imagem de fundo. Ao contrário da propriedade position, esta propriedade pega com referência o ponto central do elemento como referência. Pode receber valores numéricos para X e Y ou apresentar os seguintes valores:

-Para o eixo x: left, center, right;

-Para o eixo Y: top, center, bottom;

- deve conter uma valor para x e um para y

- se o valor for igual para ambos eixos pode declarar somente um valor, como no caso do valor center ou uma medida numérica.

```
<html>
      <head>
            <style>
                  div{
                        width:300px;
                        height:250px;
                        margin:20px 10px;
                        background-color:green;
                        background-
image:url(http://www.teste.com.br/imagem.png);
                        background-repeat:no-repeat;
                  }
                  .box-1{
                        background-position:110px center;
                  }
                  .box-2{
                        background-position:-93px -30px;
                  }
            </style>
      </head>
      <body>
            <div class="box-1"></div>
            <div class="box-2"></div>
      </body>
</html>
```

Background:

Esta propriedade resume todas as propriedades relacionadas ao plano de fundo de um elemento.

Sintaxe:

`Background: url (caminho da imagem), repetição, nome da cor, posicionamento;`

Exemplo:

```
background url (./imagem.jpg) no-repeat #ff0
```