Arquivos css

O CSS dispõe de diversas propriedades para alterar a aparência de elementos de texto. Com elas podemos customizar a cor, tamanho e fonte, bem como diversas outras características desses elementos.

A maior parte dessas propriedades estão agrupadas em font e, portanto, começaremos por elas.

**Negrito**

A propriedade font-weight é utilizada para aplicar ou remover negrito de um elemento de texto.

O valor de font-weight pode ser numérico ou uma palavra. Ambos correspondem a quão espessa será a fonte.

Os valores numéricos são 100, 200, 300, 400, 500, 600, 700, 800 e 900. Dentre eles, 400 corresponde a aparência padrão da maioria das fontes e 700 ao negrito.

No **Código 1** vemos o exemplo de uma regra que utiliza font-weight.

p {

font-weight: 400;

}

**Código 1**. Regra utilizando font-weight

Quando aplicados a elementos esses valores podem produzir o efeito observado na **Figura 1**.

Aplicação dos valores de 100 a 900 em elementos
p**Figura 1**. Aplicação dos valores de 100 a 900 em elementos p

Como podemos observar, os valores de 100 a 500 não produzem efeitos diferentes na fonte padrão do navegador. Podemos dizer o mesmo dos valores de 600 a 900. Portanto, algumas fontes podem não demonstrar alteração em sua aparência por não possuírem uma versão correspondente ao valor utilizado.

A fonte **Roboto**, distribuída pela Google, suporta todos essas variações de peso e alguns outros estilos. Demonstramos isso na **Figura 2** retirada da página dessa fonte no site [**Google Fonts**](https://fonts.google.com/specimen/Roboto).

Demonstração de uma fonte que suporta muitos
estilos**Figura 2**. Demonstração de uma fonte que suporta muitos estilos

Como podemos ver Roboto suporta todos os pesos, de 100 a 900, além de itálico, que é abordado na próxima seção.

Dentre os valores numéricos, 400 e 700 são os mais utilizados. Eles correspondem, como mencionado, ao peso padrão e em negrito de uma fonte.

Por esse motivo podemos usar a palavra-chave normal para 400 e bold para 700.

**Exemplos**

A seguir encontramos alguns exemplos práticos. Por exemplo, para remover negrito aplicando 400 ou normal a uma elemento, utilizamos o **Código 2**.

h1 {

font-weight: 400;

}

/\* ou \*/

h1 {

font-weight: normal;

}

**Código 2**. Removendo negrito do elemento h1

No **Código 2** removemos o negrito do elemento h1 que por padrão é aplicado pelo navegador.

No **Código 3** temos a aplicação de negrito com bold ou 700.

p {

font-weight: 700;

}

/\* ou \*/

p {

font-weight: bold;

}

**Código 3**. Aplicação de negrito ao elemento p

No **Código 3** aplicamos o negrito ao elemento p, sobrescrevendo o estilo padrão dele que é normal.

**Itálico**

Itálico é um estilo obtido com a propriedade font-style, utilizando o valor italic.

No **Código 4**vemos como utilizar essa propriedade em uma regra.

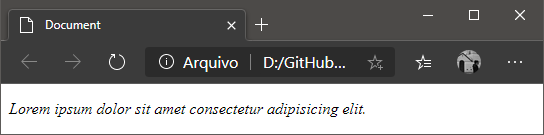
p {

font-style: italic;

}

**Código 4**. Regra utilizando font-style

O resultado desse código pode ser visto na **Figura 3**.

**Figura 3**. font-style: italic aplicado a um elemento p

**Exemplos**

Com o **Código 5** aplicamos o itálico a um elemento.

p {

font-style: italic;

}

**Código 5**. Aplicando itálico a um elemento

Com o **Código 6** removemos o itálico de um elemento.

p {

font-style: normal;

}

**Código 6**. Removendo o itálico de um elemento

Além de normal e italic, font-style também pode receber os valores oblique, mas poucas fontes dão suporte a ele.

Ao usar oblique em uma fonte que não suportar esse valor italic será automaticamente aplicado em seu lugar.

**Cor**

A cor de um texto é customizada utilizando a propriedade color, que pode receber valores numéricos ou textuais.

Os valores numéricos que podemos utilizar são escritos em hexadecimal e prefixados com cerquilha (#). O intervalo de valor vai de #000000 até #ffffff.

Alguns exemplo são, #000 (preto) e #fff (branco).

Podemos ainda usar valores textuais, que são nomes de cores em inglês, como black ou white. Esses nomes são pré-definidos e uma lista completa deles pode ser consultada [**aqui**](https://www.w3schools.com/colors/default.asp).

No **Código 7** vemos dois exemplo de como utilizar color em uma regra.

p {

color: red;

}

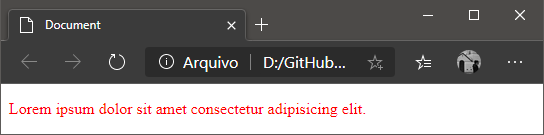
p {

color: #ff0000;

}

**Código 7**. Regra utilizando color

Na **Figura 4** está o resultado desta cor aplicada a um elemento p.

**Figura 4**. Cor aplicada ao elemento p

Os valores usados neste exemplo produzem a mesma cor vermelha, mas na primeira regra usamos o nome da cor e na segunda o seu valor hexadecimal.

**Funções de cor**

Algumas propriedade podem receber o resultado de uma função como valor. No caso de color, essas funções são rgb, rgba e hsl.

Aqui vale lembrar que o CSS não é uma linguagem de programação, mas dispõe de certas funções utilizadas para a geração de alguns valores, como uma cor específica, por exemplo.

Vamos conhecer rgb e rgba, as funções mais utilizadas nesse contexto.

**rgb e rgba**

RGB é um padrão de cor familiar a designers, onde se usa red (vermelho), green (verde) e blue (azul) para gerar uma cor.

A função rgb(red, green, blue) recebe como parâmetro três valores de 0 a 255 que correspondem a essas cores.

Dessa forma, para produzir apenas vermelho usamos rgb(255, 0, 0), verde rgb(0, 255, 0) e azul rgb(0, 0, 255).

Combinando valores para essas três cores podemos chegar a outras **16777216**.

No **Código 8** vemos um exemplo que usa rgb para definir a cor de um texto.

p {

color: rgb(81, 155, 0);

}

**Código 8**. Gerando uma cor com rgb

Nesse exemplo os valores correspondem a red=81, green=155 e blue=0 e como resultado temos a cor verde da **Figura 5**.

Cor verde correspondente a red=81, green=155 e
blue=0**Figura 5**. Cor verde correspondente a red=81, green=155 e blue=0

Além de rgb também podemos usar a função rgba para produzir uma cor. A diferença entre elas é que rgba recebe mais um parâmetro chamado alpha, que determina a opacidade da cor.

Quando alpha é 0 a cor está totalmente transparente e quando ele é 1 ela está completamente opaca. Assim, variando o alpha entre 0.0 e 1.0 conseguimos alcançar um conjunto maior de cores.

Essa função é utilizada para fins mais específicos como deixar um elemento semitransparente ou fazer ajustes mais finos a uma cor.

No **Código 9** vemos o mesmo exemplo anterior, dessa vez utilizando rgba para deixar a cor verde semitransparente.

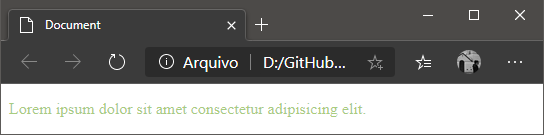
p {

color: rgba(81, 155, 0, 0.5);

}

**Código 9**. Gerando uma cor com rgba

Nesse exemplo os valores correspondem a red=81, green=155 e blue=0 e como resultado temos a cor verde da **Figura 6**.

**Figura 6**. Cor verde correspondente a red=81, green=155 e blue=0 e alpha=0.5

**Tamanho da fonte**

O tamanho da fonte pode ser customizado com a propriedade font-size.

Essa propriedade pode receber como valor palavras-chaves e números.

As palavras chave são xx-small, x-small, small, smaller, medium, larger, large, x-large e xx-large. Cada uma delas corresponde a um tamanho diferente de fonte, indo do menor para o maior.

Na prática, essas palavras-chave são pouco utilizadas, sendo mais comum utilizar valores numéricos para controlar o tamanho da fonte.

Ao utilizar um número como valor de font-size devemos informar em unidade do CSS ele está. A unidade de medida mais utilizada no CSS é o pixel (px).

Além de pixel existem outras unidades de medida que serão abordadas em um outro momento.

No **Código 10** vemos um exemplo de como modificar o tamanho de uma fonte para 22 pixels.

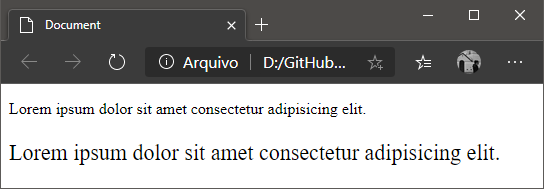
p {

font-size: 22px;

}

**Código 10**. Regra utilizando font-size

Na **Figura 7** vemos como ficaria um parágrafo antes e depois de receber essa regra.

**Figura 7**. Parágrafo antes de depois de receber essa regra

Os casos de uso dessa propriedade são muitos, mas não variam muito o exemplo apresentado no **Código 10**. Portanto, para gerar outros tamanhos de fonte basta variar o valor em pixels.

**Fonte**

Para customizar a fonte utilizada em um elemento de texto usamos a propriedade font-family que recebe como valor o nome de uma fonte.

Abaixo, no **Código 11**, vemos uma regra que utiliza a propriedade font-family.

p {

font-family: “Arial Black”, Gadget, sans-serif;

}

**Código 11**. Regra utilizando font-family

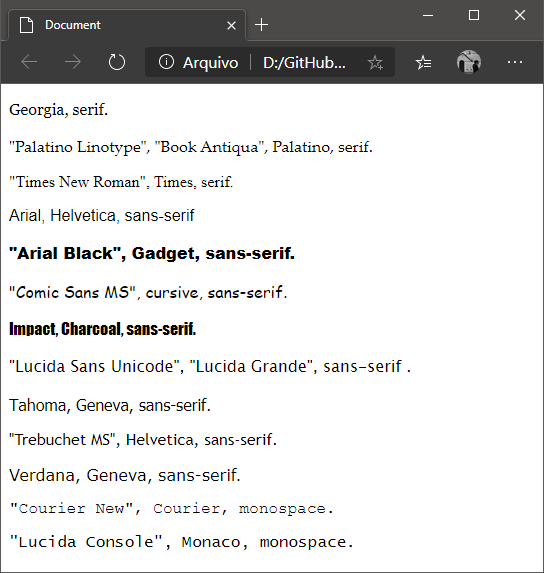
Note que nomes de fonte compostos por várias palavras devem estar entre aspas.

Na **Figura 8**, vemos um parágrafo que recebeu essa regra.

Parágrafo ao receber a fonte “Arial Black”,
Gadget, sans-serif**Figura 8**. Parágrafo ao receber a fonte “Arial Black”, Gadget, sans-serif

A fonte informada como valor para font-family deve estar presente no computador do usuário ou uma outra fonte padrão pode ser utilizada pelo navegador para garantir que o texto será, de alguma forma, apresentado.

Para reduzir esse risco de falha temos o que chamamos de "web safe fonts", um conjunto de fontes que existe na maioria dos navegadores (**Figura 9**).

**Figura 9**. As “web safe fonts”

Note que, exceto por Georgia, são utilizados mais de um nome de fonte em cada caso. Isso se dá para que o navegador tenha opções de fonte próximas do desejado para utilizar em caso de falha.

Por exemplo, no caso de Arial Black, caso essa fonte não exista no computador do usuário, o navegador tentará utilizar Gadget em seu lugar. E em último caso, o navegador buscará por uma fonte da família sans-serif, a qual Arial Black e Gadget pertencem, reduzindo assim o risco de uma fonte muito diferente daquela escolhida ser utilizada no final.

**Alinhamento**

Saindo agora da propriedade font, existem quatro formas de alinhar um texto utilizando a propriedade text-align. Elas são à direita (right), à esquerda (left), centralizado (center) e justificado (justify).

No **Código 12** vemos o exemplo de uma regra que usa o valor center para centralizar um texto.

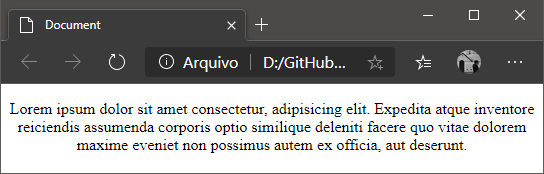
p {

text-align: center;

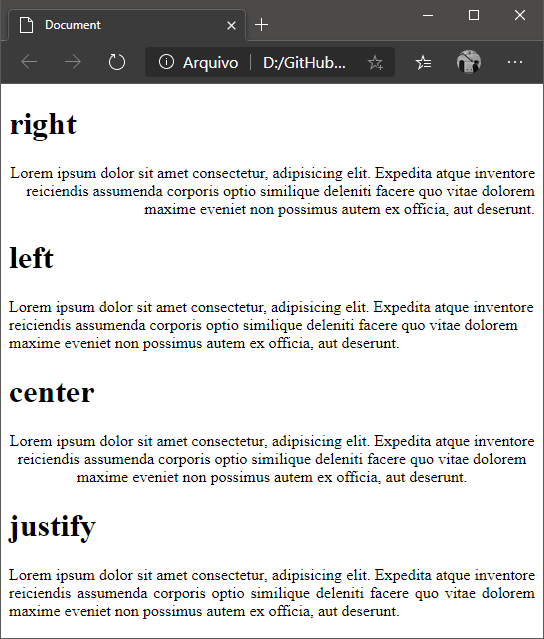
}

**Código 12**. Centralizando um texto com text-align

Na **Figura 10**vemos essa regra aplicada a um parágrafo.

**Figura 10**. Regra aplicada a um parágrafo

Na **Figura 11**vemos quatro parágrafos cada um utilizando uma dessas propriedades. Para facilitar a distinção entre os valores eles foram colocados em elementos h1.

**Figura 11**. Valores de text-align aplicados a parágrafos

Para alinhar o texto de outras formas basta utilizar um dos quatro valores de text-align, que são: right, left, justify e center.

**Conclusão**

O texto está presente em todo lugar. Nesse artigo fomos apresentados as principais propriedades disponíveis no CSS para customizá-lo em uma página web.

Tecnologias: