

Curso de **CSS 3**

CSS



Este material é exclusivo do canal (youtube): canalfessorbruno e do site www.cfbcursos.com.br
não pode ser distribuido ou comercializado por outra fonte.

Apostila: versão 1.0

08/02/2016

www.youtube.com/canalfessorbruno

www.cfbcursos.com.br

canalfessorbruno@gmail.com

www.facebook.com/canalfessorbruno

twitter: @fessorBruno

Sumário

Introdução	8
Inserindo CSS	8
Incorporado	8
Externo	8
Sintaxe	9
Seletores CSS	9
Seletor para tag	10
Seletor para id	10
Seletor para classes	10
Vários seletores	11
Cores.....	12
background-color – Cor de fundo	18
background-image.....	19
background-size.....	19
background-repeat	20
background-position.....	21
background-origin.....	21
background-clip	22
background-attachment	23
Multiplas imagens como background-image	24
background - Fundo	25
Imagem de fundo em tela inteira	25
border – bordas / linhas de contorno	26
border-radius / Cantos arredondada	29
border-image – CSS3	30
Cores RGB decimal.....	31
rgba – Transparência – CSS3.....	32
margin – margens	33
Centralizar a página na janela e seus elementos	35
padding - margens internas	38
Gradientes – CSS3	40
linear-gradient	41
radial-gradient	42
repeating-linear-gradient & repeat-radial-gradient.....	42
Aplicando os gradientes.....	42

shadow – Efeito de sombra – CSS3	43
text-shadow	43
box-shadow	45
Comentários	46
width e height / largura e altura	46
width – largura.....	47
height – Altura	47
max-width & min-width	48
Formatações para textos	48
color – cor	49
text-align - alinhamento	49
text-decoration – Decoração	50
text-transform – Transformações	51
text-indent – Endentação	52
letter-spacing – Espaçamento entre as letras	52
line-height – Espaçamento entre as linhas	53
direction – Direção do texto	54
word-spacing – Espaçamento entre as palavras	54
white-space	55
overflow – CSS3 - Texto transbordado.....	56
word-wrap – CSS3 – Quebra de texto longo	58
word-break – CSS3 – Quebra a palavra	58
text-align-last – CSS3 – Alinhamento da última linha do texto	59
Formatações para fontes	59
font-family	59
font-style	60
font-size.....	60
font-weight	61
font-variant.....	61
@font-face – CSS3	62
Formatação de links	63
Formatando o link como um botão	64
Transformações 2D – CSS3	65
translate()	65
rotate()	66
scale()	66

skewX()	67
skewY()	68
skew()	68
matrix()	69
transform-origin	69
Transformações 3D – CSS3	72
perspective()	73
backface-visibility	74
Listas	75
list-style-type	76
list-style-image	77
list-style-position	78
list-style	79
Menu horizontal com listas	79
Tabelas	80
Formatações comuns	81
vertical-align	81
border-collapse	81
border-spacing	82
empty-cells	82
table-layout	82
nth-child() – CSS3	82
outline	84
outline-color	85
outline-offset	85
outline-style	85
outline-width	85
display	86
block	86
inline	86
none	86
inline-block	86
inline-table	86
list-item	86
run-in	86
table	86

table-caption	86
table-header-group.....	87
table-footer-group.....	87
table-row-group.....	87
table-cell	87
table-column.....	87
table-row	87
position.....	91
static	91
relative.....	92
absolute.....	93
fixed.....	94
z-index	96
float	99
clear	101
overflow.....	104
Seletores after e before	106
Menu dropDown sem script	107
transition – Transições – CSS3	109
transition-timing-function	110
transition-delay.....	111
transition + transform.....	111
animation – CSS3	111
animation-duration.....	113
animation-direction	114
animation-timing-function.....	114
animation-iteration-count	114
Metapropriedade transition	114
animation-play-state	114
Colunas – CSS3.....	114
column-count	115
column-gap.....	115
column-rule	115
column-width	116
resize	116
box-sizing – CSS3	117

Filter – CSS3	119
grayscale	120
blur	120
brightness	121
contrast	121
drop-shadow	121
hue-rotate	122
invert	122
opacity	123
saturate	123
sepia	124
Aplicando vários filtros	124
opacity	125
@media – layouts responsivos	125
Unidades de medida	134
inherit, initial	134
Tabela geral dos parâmetros CSS	135
Tabela de compatibilidade, CSS x Browser	141
Considerações finais	144

Introdução

CSS é uma linguagem para formatação de documentos de marcação, tais como HTML ou XML, arquivos CSS são denominados como folhas de estilo, pois, contém todo código para formatação visual / estilo da página web. Saber trabalhar bem com CSS é fundamental para que possamos formatar bem uma página web, hoje em dia, nos padrões atuais de desenvolvimento web, não usamos mais códigos “misturados”, ou seja, tipos diferentes de linguagem como HTML e CSS misturados pelo código afora, tudo no seu devido lugar, separados, isso facilita no desenvolvimento, manutenção e portabilidade do código.

Nesta apostila você vai encontrar todas as informações necessárias para entender como funciona esta linguagem de formatação, comandos antigos que foram continuados e novos comandos da versão 3.

Então vamos começar que temos muito a aprender.

Inserindo CSS

Podemos trabalhar com os códigos de CSS no próprio documento (incorporado) ou em um arquivo externo, a melhor forma de trabalho em com o arquivo “.css” externo, porém, nesta apostila, simplesmente por facilidade e para deixar a didática mais ágil, vou mostrar a maioria dos trabalhos com o código incorporado no documento.

Incorporado

No modo de trabalho incorporado, o código CSS é inserido dentro da tag `<style>` que é inserida no `<head>` da página, veja o código de exemplo a seguir.

```
<!doctype html>
<html lang="pt-br">
  <head>
    <title>Curso de CSS</title>
    <meta charset="UTF-8">
    <style rel="stylesheet">
      ...
    </style>
  </head>
  <body>
    ...
  </body>
</html>
```

Externo

No modo de trabalho externo, criamos um arquivo separado denominado folha de estilo com extensão .css, neste arquivo serão inseridos todos os comandos relacionados à formatação da página, em nosso código de exemplo anexamos uma folha de estilos chamada “estilos.css” onde todo código de formatação será inserido.

```
<!doctype html>
<html lang="pt-br">
  <head>
    <title>Curso de CSS</title>
    <meta charset="UTF-8">
    <style rel="stylesheet" href="estilos.css">
  </head>
  <body>
    ...
  </body>
</html>
```

Sintaxe

A sintaxe padrão de CSS é bastante simples, precisamos indicar um seletor que é uma tag, id ou classe, e dentro das chaves inserimos os comandos referente à formatação, veja um exemplo da sintaxe básica a seguir.

seletor { propriedade:valor; propriedade:valor; }

p { color:#F00; font-size:20pt; }

Observe o código de exemplo a seguir onde aplicamos uma formatação de cor e tamanho à tag <p>.

```
<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de CSS</title>
        <meta charset="UTF-8">
        <style rel="stylesheet">
            p{ color:#F00; font-size:20pt; }
        </style>
    </head>
    <body>
        <p>Canal Fessor Bruno</p>
    </body>
</html>
```

Podemos inserir os comandos em linha ou um em baixo do outro como no exemplo a seguir.

```
p{
    color:#F00;
    font-size:20pt;
}
```

Observe um detalhe importante, no final de cada comando precisamos inserir o ponto e vírgula ";" para sinalizar que o comando terminou.

Outra observação importante é o operador que separa a propriedade do valor, neste caso usamos dois pontos ":".

O resultado da formatação anterior é a ilustração a seguir.



Seletores CSS

Existem vários tipos de seletores CSS, os padrões são os seletores para tags, ids e classes, neste capítulo vamos entender como utilizar estes seletores e vou apresentar uma tabela com diversas formas de utilização.

Seletor para tag

O seletor de tag é o mais simples de todos, basta informar o nome da tag, neste caso, todas as tags <p> existentes no documento receberão a formatação indicada.

```
<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de CSS</title>
        <meta charset="UTF-8">
        <style rel="stylesheet">
            p{
                color:#F00;
                font-size:20pt;
            }
        </style>
    </head>
    <body>
        <p>Canal Fessor Bruno</p>
    </body>
</html>
```

Seletor para id

O seletor para id é mais específico do que o de tags, formata somente a tag que possui o id especificado, para indicar que o seletor será para um id basta anteceder o nome do id com o caractere hash "#".

```
<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de CSS</title>
        <meta charset="UTF-8">
        <style rel="stylesheet">
            #txt{
                color:#F00;
                font-size:20pt;
            }
        </style>
    </head>
    <body>
        <p id="txt">Canal Fessor Bruno</p>
    </body>
</html>
```

Seletor para classes

Este selector formata todos os elementos que usam a classe indicada, desta maneira podemos ter várias tags <p> por exemplo, somente receberão a formatação aquelas que tiverem o atributo class com o nome da classe configurada, para identificar este seletor, basta usar um ponto “.” antes do nome da classe.

```
<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de CSS</title>
        <meta charset="UTF-8">
        <style rel="stylesheet">
            .txt{
                color:#F00;
                font-size:20pt;
            }
        </style>
    </head>
    <body>
        <p class="txt">Canal Fessor Bruno</p>
    </body>
</html>
```

Vários seletores

Podemos indicar mais de um seletor para receber um conjunto de formatações CSS, basta indicar estes seletores separados por vírgulas, como no código de exemplo.

```
<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de CSS</title>
        <meta charset="UTF-8">
        <style rel="stylesheet">
            h1, h2, h3, p{
                color:#F00;
                font-size:20pt;
            }
        </style>
    </head>
    <body>
        <h1>Canal Fessor Bruno</h1>
        <h2>Canal Fessor Bruno</h2>
        <h3>Canal Fessor Bruno</h3>
        <p>Canal Fessor Bruno</p>
    </body>
</html>
```

Mesmo que os seletores sejam de tipos diferentes podemos indicar como grupos.

```
<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de CSS</title>
        <meta charset="UTF-8">
        <style rel="stylesheet">
            h1, #txth2, .txt, p{
                color:#F00;
                font-size:20pt;
            }
        </style>
    </head>
    <body>
        <h1>Canal Fessor Bruno</h1>
        <h2 id="txth2">Canal Fessor Bruno</h2>
        <h3 class="txt">Canal Fessor Bruno</h3>
        <p>Canal Fessor Bruno</p>
    </body>
</html>
```

São várias as formas de selecionar um elemento para formatação CSS, a seguir disponibilizo uma tabela com os possíveis seletores CSS.

Seletor CSS	Exemplo	Descrição
.classe	.intro	Seleciona todos os elementos que usam a classe intro / class="intro"
#id	#primeiroNome	Seleciona o elemento com o id primeiroNome / id="primeiroNome"
*	*	Seleciona todos os elementos
elemento	p	Seleciona todos os elementos com a tag <p>
elemento,elemento	div, p	Seleciona todos os elementos <div> e <p>
elemento elemento	div p	Seleciona todos os elementos <p> que estão dentro de elementos <div>
elemento>elemento	div > p	Seleciona todos os elementos <p> onde o pai seja um elemento <div>
elemento+elemento	div + p	Seleciona todos os elementos <p> que estão posicionados imediatamente após o elemento <div>
elemento1~elemento2	p ~ ul	Seleciona todos os elementos que são precedidos por um elemento <p>
[atributo]	[target]	Seleciona todos os elementos com o atributo target
[atributo=valor]	[target=_blank]	Seleciona todos os elementos com o atributo target configurado em _blank / target="_blank"
[atributo~=valor]	[title~=curso]	Seleciona todos os elementos cujo o atributo title contenha a palavra "curso"
[atributo =valor]	[lang =pt]	Seleciona todos os elementos com um valor de atributo lang começando com "pt"
[atributo^=valor]	a[href^="https"]	Seleciona cada elemento <a> cujo valor do atributo href começa com "https"
[atributo\$=valor]	a[href\$=".rar"]	Seleciona cada elemento <a> cujo valor do atributo href termina com ".rar"
[atributo*=valor]	a[href*="cursos"]	Seleciona a cada <uma> elemento cujo valor do atributo href contenha a substring "cursos"
:active	a:active	Seleciona o link ativo
::after	p::after	Insira algo depois do conteúdo de cada elemento
::before	p::before	Insira algo antes do conteúdo de cada elemento
:checked	input:checked	Seleciona todos os elementos <input> que tem a propriedade checked

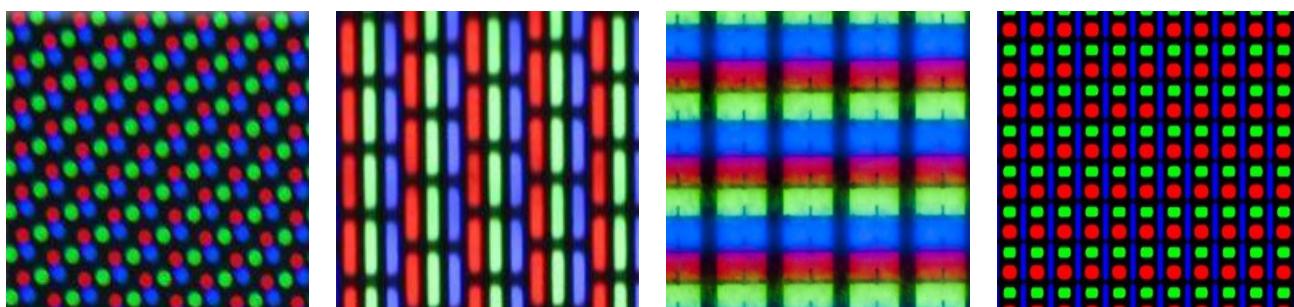
:disabled	input:disabled	Seleciona todos os elementos <input> que tem a propriedade disabled
:empty	p:empty	Seleciona todos os elementos <p> que não tem filhos (incluindo os nós de texto)
:enabled	input:enabled	Seleciona todos os elementos <input> que tem a propriedade enabled
:first-child	p:first-child	Seleciona todos os elementos <p> que são first-child de seu elemento pai
::first-letter	p::first-letter	Seleciona a primeira letra de cada elemento <p>
::first-line	p::first-line	Selects the first line of every <p> element
::first-of-type	p:first-of-type	Seleciona a primeira linha de cada elemento <p>
:focus	input:focus	Seleciona o elemento de entrada que tem o foco
:hover	a:hover	Seleciona os links que o tem a propriedade hover configurada
:in-range	input:in-range	Seleciona elementos de entrada com um valor dentro de um intervalo especificado
:invalid	input:invalid	Seleciona todos os elementos <input> que tem a propriedade invalid
:lang(language)	p:lang(en)	Seleciona todos os elementos <p> com um atributo lang igual a "en" (inglês)
:last-child	p:last-child	Seleciona todos os elementos <p> que é o último filho do seu elemento pai
:last-of-type	p:last-of-type	Seleciona todos os elementos <p> que é o último elemento <p> de seu elemento pai
:link	a:link	Seleciona todos os links não clicados
:not(selector)	:not(p)	Seleciona todos os elementos que não são um elemento <p>
:nth-child(n)	p:nth-child(2)	Seleciona o elemento <p> que é o segundo filho de seu elemento pai
:nth-last-child(n)	p:nth-last-child(2)	Seleciona o elemento <p> que é o segundo filho de seu elemento pai, a contar do último elemento filho
:nth-last-of-type(n)	p:nth-last-of-type(2)	Seleciona o elemento <p> que é o segundo elemento <p> de seu elemento pai, a contar do último elemento filho
:nth-of-type(n)	p:nth-of-type(2)	Seleciona o elemento <p> que é o segundo elemento <p> de seu elemento pai
:only-of-type	p:only-of-type	Seleciona todos os elementos <p> de seu elemento pai
:only-child	p:only-child	Seleciona todos os elementos <p> que só tem um elemento como filho
:optional	input:optional	Seleciona elementos input com nenhum atributo "required"
:out-of-range	input:out-of-range	Seleciona elementos input com um valor fora de um intervalo especificado
:read-only	input:read-only	Seleciona elementos input com o atributo "readonly" especificado
:read-write	input:read-write	Seleciona elementos input com o atributo "readonly" NÃO especificado
:required	input:required	Seleciona elementos input com o atributo "required" especificado
:root	:root	Seleciona o elemento raiz do documento
::selection	::selection	Seleciona a parte de um elemento que é selecionado pelo usuário
:target	#notícias:target	Seleciona o elemento #notícias atual (clicou em um URL que contém esse nome de âncora)
:valid	input:valid	Seleciona todos os elementos input com um valor válido
:visited	a:visited	Seleciona todos os links visitados

Cores

Trabalhar com cores para web é uma tarefa um pouco trabalhosa, principalmente no início quando ainda não se tem o costume e o domínio do assunto, neste capítulo vou mostrar como podemos configurar as cores usando caracteres hexadecimais.

Podemos indicar o nome da cor em inglês, o que parece ser uma tarefa mais simples, mas entenda a grande limitação desta forma de trabalho. Pense e conte quantas cores você conhece? Agora, destas cores que você pensou, conte somente quantas você sabe o nome em inglês? Quantas você conseguiu contar? Por mais cores que sejam, o número de aproximou de 16.777.216? Claro que não! Aposto que nem pensou inicialmente nesta quantidade de variações de cores!

O sistema de cores usado na web se chama RGB, que são as letras para Red Green Blue (Vermelho, Verde e Azul). Esse é o sistema de cores usado por qualquer tela que não seja monocromática, seja CRT (tubo), LCD, Led, enfim, independente do tipo, veja a seguir algumas formações de pixels RGB.



Mas se as telas são compostas somente de pixels de três cores, de onde vem as outras cores? Acredite, todas as cores que vemos no monitor vem da mistura somente destas três cores, do branco ao preto!

O que acontece é que alteramos a intensidade destes pixels para formar as demais cores, veja bem, para formar o vermelho, basta diminuir a intensidade ao mínimo dos pixels Green (verde) e Blue (azul) e aumentar ao máximo a intensidade do pixel Red (vermelho), então teremos a cor vermelha, a mesma coisa para o verde e para o azul.

Podemos controlar a intensidade dos pixels a uma profundidade de 8bits, ou seja, cada canal, pode ser controlado com um valor de 8bits, então, 8bits para o Red, 8bits para o Green e 8bits para o Blue, totalizando 24bits de profundidade, ou seja, usamos o sistema de cores RGB de 24bits, alguns programas como o Photoshop, informam este sistema como RGB 8bits, já o CorelDRAW como RGB 24bits, mas qual a diferença? A diferença é que o Photoshop mostra a quantidade de bits por canal e o CorelDRAW mostra a quantidade de bits total ($8+8+8=24$).

Assim, a profundidade indica quantas variações de nuance são possíveis em cada canal, desta maneira, com um canal de 8bits conseguimos 256 variações, nossa agora deu um nó na cabeça? De onde vem este número 256? Vejamos.

Em nosso computador 8bits é equivalente a 1byte, que por sua vez é equivalente a um caractere, como estamos falando de números, vamos equivaler 1byte a um número, decimal.

Veja como representamos 1byte em sistema binário:

00000000	00000001	00000010	00000011	00000100	00000101	00000110	00000111	00001000
----------	----------	----------	----------	----------	----------	----------	----------	----------

Como podemos observar são formações de 8bits para formar 1byte, ou um numeral, desta maneira, no sistema binário de 8bits conseguimos quantas combinações diferentes? Eis a resposta $2^8 = 256$, ou seja (binário elevado a 8bits = 256 combinações).

Então, convertendo de binário 8bits para decimal temos:

```

00000000 = 0
00000001 = 1
00000010 = 2
00000011 = 3
00000100 = 4
00000101 = 5
00000110 = 6
00000111 = 7
00001000 = 8
00001001 = 9
00001010 = 10
até
11111111 = 255

```

De 0 a 255 temos 256 combinações diferentes, assim, 256 para Red, 256 para Green e 256 para Blue conseguimos $16.777.216$ combinações diferentes, pois, $256^3 = 16.777.216$.

OK, achamos os número, mas só falamos até agora de binário (base 2) e decimal (base 10), precisamos de hexadecimal, ou seja, base 16.

Vamos entender mais alguns detalhes simples sobre os sistemas de numeração.

Binário (base 2) = 0, 1

Decimal (base 10 – Nossa sistema de numeração padrão) = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Hexadecimal (base 16) = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

Não vou mostrar os cálculos para conversão, mas vou mostrar as equivalências a seguir.

Binário	Decimal	Hexadecimal
00000000	0	0
00000001	1	1
00000010	2	2
00000011	3	3
00000100	4	4
00000101	5	5
00000110	6	6
00000111	7	7
00001000	8	8
00001001	9	9
00001010	10	A
00001011	11	B
00001100	12	C
00001101	13	D
00001110	14	E
00001111	15	F
00010000	16	10
00010001	17	11
00010010	18	12
00010011	19	13
00010100	20	14
00010101	21	15
00010110	22	16
00010111	23	17
00011000	24	18
00011001	25	19
00011010	26	1A
00011011	27	1B
00011100	28	1C
00011101	29	1D
00011111	30	1E
00100000	31	1F
00100001	32	20
00100010	33	21
00100011	34	22
00100100	35	23
00100101	36	24
00100110	37	25
00100111	38	26
00101000	39	27
00101001	40	28
00101010	41	29
00101011	42	2A
até...	até...	até...
11111111	255	FF

Vou fazer um resumo usando os sistemas decimal e hexadecimal

Decimal	hexadecimal
0	00
1	01
2	02
3	03
4	04
5	05
6	06
7	07
8	08
9	09
10	0A
11	0B
12	0C
13	0D
14	0E
15	0F

Decimal	hexadecimal
16	10
17	11
18	12
19	13
20	14
21	15
22	16
23	17
24	18
25	19
26	1A
27	1B
28	1C
29	1D
30	1E
31	1F

Decimal	hexadecimal
32	20
33	21
34	22
35	23
36	24
37	25
38	26
39	27
40	28
41	29
42	2A
43	2B
44	2C
45	2D
46	2E
47	2F

Decimal	hexadecimal
48	30
49	31
50	32
51	33
52	34
53	35
54	36
55	37
56	38
57	39
58	3A
59	3B
60	3C
61	3D
62	3E
63	3F

Decimal	hexadecimal
64	40
...	...
73	49
74	4A
75	4B
76	4C
77	4D
78	4E
79	4F
80	50
...	...
90	5A
91	5B
92	5C
...	...
95	5F

Decimal	hexadecimal
96	60
...	...
105	69
106	6A
...	...
111	6F
112	70
...	...
127	7F
128	80
...	...
143	8F
144	90
...	...
159	9F
160	A0

Decimal	hexadecimal
161	A1
...	...
175	AF
176	B0
...	...
191	BF
192	C0
...	...
207	CF
208	D0
...	...

Decimal	hexadecimal
223	DF
224	E0
...	...
239	EF
240	F0
255	FF

Com todos estes valores podemos chegar à conclusão sobre os valores mínimo, médio e máximo.

	Decimal	Hexadefinal
Mínimo	00	00
Médio	128	80
Máximo	255	FF

Vamos finalmente formar cores.

Para formar as cores em RGB (vermelho, verde, azul), basta “misturar” as cores, pela intensidade de cada canal. Sendo “00” o valor mínimo e “FF” o valor máximo, respectivamente, a intensidade (força) mínima e máxima do canal. Assim, se informarmos Red=00, Green=00 e Blue=00, intensidade mínimo para todos os canais, vamos formar a cor preto, o contrário Red=FF, Green=FF e Blue=FF, intensidade máxima para todos os canais, vamos formar o branco.

Desta maneira vamos ver as cores básicas.

Preto = #000000

Branco = #FFFFFF

Vermelho = #FF0000

Verde = #00FF00

Azul = #0000FF

Misturando dois canais podemos obter as cores do sistema CMYK (Ciano, Magenta, Amarelo e Preto), vejamos

Ciano = #00FFFF

Magenta = #FF00FF

Amarelo = #FFFF00

Agora que já vimos as 8 possibilidades de cores usando as intensidades mínima (00) e máxima (FF), vamos formar outras cores usando intensidades diferentes.

Laranjado = #FF8000 Confuso?

Vou explicar, quais tintas misturamos para criar a cor laranja? Vermelho e amarelo! Aqui aconteceu a mesma coisa! Vejamos.

Vermelho = #FF0000;

Amarelo = #FFFF00;

Agora vamos adicionar um canal ao outro.

Cor	Red = vermelho	Green = Verde	Blue = Azul
Vermelho	FF	00	00
Amarelo	FF	FF	00
+	+	+	+
Resultado	FF	FF	00

R = Como o valor máximo é FF, então a soma de FF com FF é FF.

G = 00 + FF = FF

B = 00 + 00 = 00

Note então que a soma de vermelho (#FF0000) com amarelo (#FFFF00) foi igual ao próprio amarelo (#FFFF00), mas isso não resultou em laranjado! Calma, vamos corrigir.

O que deu errado é que usamos o amarelo na sua intensidade máxima, vamos diminuir a intensidade do amarelo pela metade.

Amarelo menos intenso = #808000 (Red = 80, Green = 80, Blue = 00)

Agora vamos somar novamente.

Cor	Red = vermelho	Green = Verde	Blue = Azul
Vermelho	FF	00	00
Amarelo menos intenso	80	80	00
+	+	+	+
Resultado	FF	80	00

$$R = FF + 80 = FF$$

$$G = 00 + 80 = 80$$

$$B = 00 + 00 = 00$$

Então, agora obtivemos o laranjado #FF8000.

Vamos a mais uma soma? Como obter a cor roxo? Misturando vermelho e azul! Vamos a esta mistura.

Vermelho menos intenso = #800000

Azul = #0000FF

Cor	Red = vermelho	Green = Verde	Blue = Azul
Vermelho menos intenso	80	00	00
Azul	00	00	FF
+	+	+	+
Resultado	80	00	FF

$$\text{Roxo} = \#8000FF (\text{Red} = 80, \text{Green} = 00, \text{Blue} = FF)$$

Para escurecer uma cor basta diminuir a intensidade dos canais, veja a seguir as intensidades de vermelho diminuídas até chegar ao preto.

Vermelho = #FF0000;

Vermelho = #C00000;

Vermelho = #800000;

Vermelho = #400000;

Preto = #000000;

Deste maneira com todas as cores, basta diminuir o valor dos canais para escurecer e aumentar para clarear.

E quanto aos tons de cinza?

Para formar tons de cinza basta usar valores iguais nos três canais, veja os exemplos iniciando do branco e terminando no preto

Branco = #FFFFFF;

Cinza = #COCOCO;

Cinza = #808080;
 Cinza = #404040;
 Preto = #000000;

Uma facilidade que podemos usufruir para criar cores é usar a forma resumida, ou seja, usar somente um dígito para cada canal, veja alguns exemplo.

#FFFFFF = #FFF #FFCC88 = #FC8 #AA44BB = #A4B #AABBCC = #ABC

Em nosso curso vamos usar na grande maioria das vezes o format resumido.

#FF0000 = #FO0	#FFFF00 = #FF0	#000000 = #000
#00FF00 = #0F0	#FF00FF = #FOF	#FF8800 = #F80
#0000FF = #00F	#00FFFF = #OFF	#8800FF = #80F

Pronto, agora já sabemos as informações necessárias para conseguirmos formar qualquer cor que desejarmos, vamos a um pouco de prática.

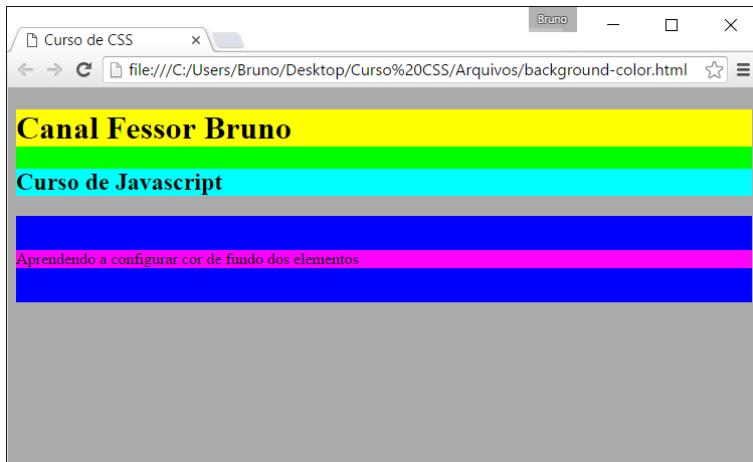
background-color – Cor de fundo

A propriedade background-color permite configurar uma cor de fundo a um elemento, podemos aplicar praticamente a qualquer elementos que desejamos mudar a cor do fundo, body, p, div, h1 a h6, header, section, etc.

Observe o código de exemplo.

```
<!doctype html>
<html lang="pt-br">
  <head>
    <title>Curso de CSS</title>
    <meta charset="UTF-8">
    <style rel="stylesheet">
      body{
        background-color:#AAA;
      }
      header{
        background-color:#0F0;
      }
      section{
        background-color:#00F;
      }
      h1{
        background-color:#FF0;
      }
      h2{
        background-color:#OFF;
      }
      p{
        background-color:#F0F;
      }
    </style>
  </head>
  <body>
    <header>
      <h1>Canal Fessor Bruno</h1>
      <h2>Curso de Javascript</h3>
    </header>
    <section>
      <br>
      <p>Aprendendo a configurar cor de fundo dos elementos</p>
      <br>
    </section>
  </body>
</html>
```

Resultado do código acima.



background-image

Caso precise adicionar uma imagem de fundo basta usar `background-image`, sem delongas veja o código a seguir onde inserimos a imagem “cfb.jpg” como imagem de fundo da página.

```
<!doctype html>
<html lang="pt-br">
  <head>
    <title>Curso de CSS</title>
    <meta charset="UTF-8">
    <style rel="stylesheet">
      body{
        background-image:url(cfb.jpg);
      }
    </style>
  </head>
  <body>
  </body>
</html>
```

Veja o resultado.



Note que o padrão de aplicação da imagem é que preencha toda a janela, repetindo na horizontal “x” e vertical “y”.

Podemos configurar algumas propriedades da imagem de fundo, como tamanho, posição inicial, repetição e recorte, vejamos.

background-size

Com a propriedade `background-size` podemos configurar o tamanho da imagem de fundo. Adicionando esta propriedade podemos controlar o tamanho da largura e altura da imagem, em nosso código de exemplo usamos o valor 200px para definir a largura, quando usamos somente um valor para largura, automaticamente a altura aumenta ou diminui na proporção devida.

```
<style rel="stylesheet">
    body{
        background-image:url(cfb.jpg);
        background-size: 200px;
    }
</style>
```



Para mudar largura e altura basta informar o valor para a altura, veja um exemplo usando 200px de largura por 100px de altura.

```
background-size: 200px 100px;
```

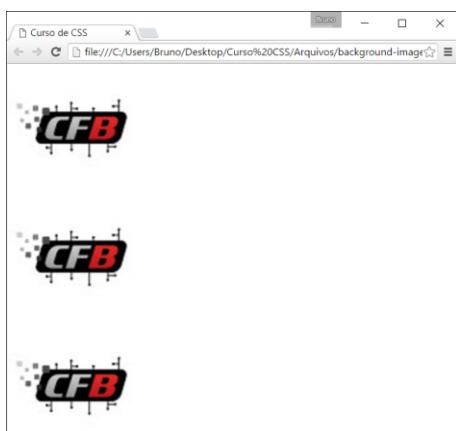
background-repeat

Este propriedade configura a repetição da imagem, por padrão a imagem de fundo se repete até que preencha toda área da janela, mas podemos controlar se vai repetir somente na horizontal, vertical ou sem repetição.

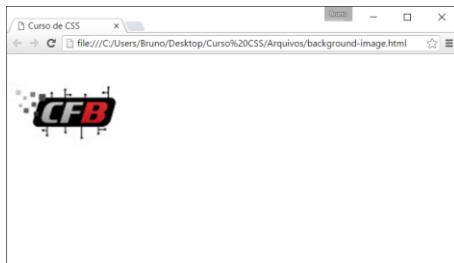
```
<style rel="stylesheet">
    body{
        background-image:url(cfb.jpg);
        background-size: 200px;
        background-repeat: repeat-x;
    }
</style>
```



```
<style rel="stylesheet">
    body{
        background-image:url(cfb.jpg);
        background-size: 200px;
        background-repeat: repeat-y;
    }
</style>
```



```
<style rel="stylesheet">
    body{
        background-image:url(cfb.jpg);
        background-size: 200px;
        background-repeat: no-repeat;
    }
</style>
```

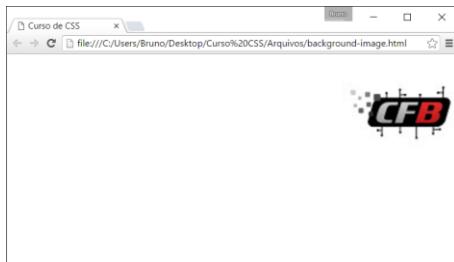


background-position

Esta propriedade configura o posicionamento da imagem de fundo, é importante configurar a imagem sem repetição para que possamos ver com clareza a propriedade atuando, não é obrigatório, mas é mais fácil de entender quando a imagem é pequena, em nosso código de exemplo posicionamos a imagem no lado direito e na parte superior da tela.

```
<style rel="stylesheet">
    body{
        background-image:url(cfb.jpg);
        background-size: 200px;
        background-repeat: no-repeat;
        background-position: right top;
    }
</style>
```

Note que informamos dois valores para position, o primeiro em relação ao posicionamento horizontal e o segundo para o posicionamento vertical.



Confira os valores possíveis para **background-position**

- left top = Esquerda em cima
- left center = Esquerda no centro
- left bottom = Esquerda em baixo
- right top = Direita em cima
- right center = Direita no centro
- right bottom = Direita em baixo
- center top = Centro no topo
- center center = Centro no centro
- center bottom = Centro em baixo

background-origin

A propriedade **background-origin** especifica onde será a origem do posicionamento da imagem de fundo, veja os valores possíveis.

border-box = A imagem inicia à esquerda no topo

padding-box = (padrão) A imagem inicia à esquerda no topo levando em consideração o padding aplicado ao elemento pai.

content-box = A imagem inicia à esquerda no topo do conteúdo do container onde a imagem está aplicada.

background-clip

A propriedade background-clip especifica a área do fundo, veja o código de exemplo.

```
<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de CSS</title>
        <meta charset="UTF-8">
        <style rel="stylesheet">
            #ex1 {
                border: 10px dotted black;
                padding: 35px;
                background: #0F0;
                background-clip: border-box;
            }

            #ex2 {
                border: 10px dotted black;
                padding: 35px;
                background: #0F0;
                background-clip: padding-box;
            }

            #ex3 {
                border: 10px dotted black;
                padding: 35px;
                background: #0F0;
                background-clip: content-box;
            }
        </style>
    </head>
    <body>
        <div id="ex1">
            <h2>border-box</h2>
            <p>Canal Fessor Bruno - Curso de CSS</p>
        </div>
        <br>
        <div id="ex2">
            <h2>padding-box</h2>
            <p>Canal Fessor Bruno - Curso de CSS</p>
        </div>
        <br>
        <div id="ex3">
            <h2>content-box</h2>
            <p>Canal Fessor Bruno - Curso de CSS</p>
        </div>
    </body>
</html>
```



Note que os valores são os mesmos da propriedade background-position o que muda é que o valor padrão é "border-box".

border-box = (padrão) O fundo inicia à esquerda no topo

padding-box = O fundo inicia à esquerda no topo levando em consideração o padding aplicado ao elemento pai.

content-box = O fundo inicia inicia à esquerda no topo do conteúdo do container.

background-attachment

Especifica como a imagem de fundo se comportará na rolagem da tela, com este propriedade é possível configurar que a imagem de fundo não role junto com a tela quando a barra de rolagem for usada.

```
<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de CSS</title>
        <meta charset="UTF-8">
        <style rel="stylesheet">
            body{
                background-image:url(cfb.jpg);
                background-repeat: no-repeat;
                background-position: center top;
                background-size: 400px;
                background-attachment: fixed;
            }

            p{margin-bottom:2000px;}
        </style>
    </head>
    <body>

        <p>Canal Fessor Bruno</p>

    </body>
</html>
```

Com a formatação acima a imagem de fundo ficará fixa e não irá se mover quando a tela for rolada.



Os valores possíveis para esta propriedade são.

scroll = (padrão) A imagem rola junto com o conteúdo da janela.

fixed = A imagem fica fixa e não rola junto com o conteúdo.

local = A imagem rola junto com o conteúdo da janela.

Initial e inherit

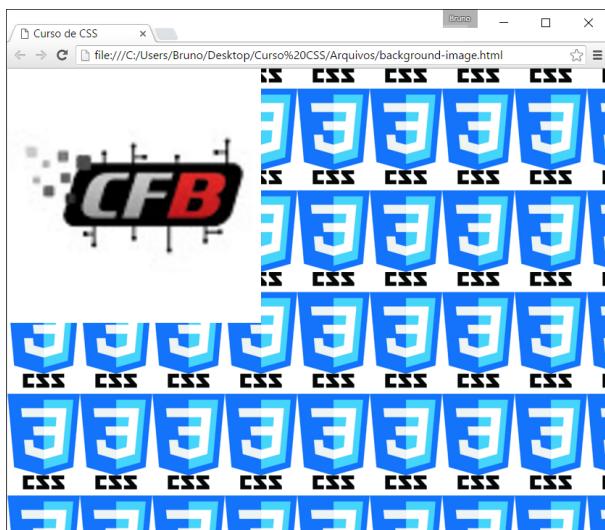
Multiplas imagens como background-image

Podemos configurar mais de uma imagem como plano de fundo em nossas páginas, basta separar por vírgula, simples assim, veja o código de exemplo.

```
<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de CSS</title>
        <meta charset="UTF-8">
        <style rel="stylesheet">
            body{
                background-image: url(cfb.jpg), url(logo-CSS3.jpg);
                background-repeat: no-repeat, repeat;
                background-size: 350px, 100px;
            }
        </style>
    </head>
    <body>

    </body>
</html>
```

Note que separamos as imagens pela vírgula e as outras propriedades também.



background - Fundo

Background é uma metapropriedade para configuração do fundo da página, a vantagem é que podemos usar para aplica mais de uma propriedade no mesmo comando, podemos definir cor e imagem de fundo e algumas configurações da imagem ao mesmo tempo, sem ter que usar um comando para cada alteração.

A sintaxe é: background: cor imagem posição/tamanho repeat origem clip attachment initial | inherit;

Veja o código de exemplo onde aplicamos cor de fundo preto, imagem de fundo “cfb.jpg” para repetir na horizontal, configuramos a imagem em scroll para poder rolar normalmente e alinhamento horizontal ao centro e vertical no topo.

```
<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de CSS</title>
        <meta charset="UTF-8">
        <style rel="stylesheet">
            body{
                background:#000 url(cfb.jpg) repeat-x scroll center top;
            }
        </style>
    </head>
    <body>

    </body>
</html>
```



Imagen de fundo em tela inteira

Para configurar a imagem de fundo de forma que preencha toda a janela basta usar o comando a seguir.

```
<style rel="stylesheet">
    body{
        background:url(cfb.jpg) 0px 0px / 100%;
    }
</style>
```

url = imagem

0px 0px = Posição (left,top)

100% = Tamanho



Desta maneira a imagem de fundo será ajustada automaticamente de acordo com o tamanho a tela.

border – bordas / linhas de contorno

A metapropriedade border define as configurações das bordas do elemento, podemos definir o estilo da borda, largura e cor de forma simples.

A sintaxe é a seguinte: border: estilo largura cor;

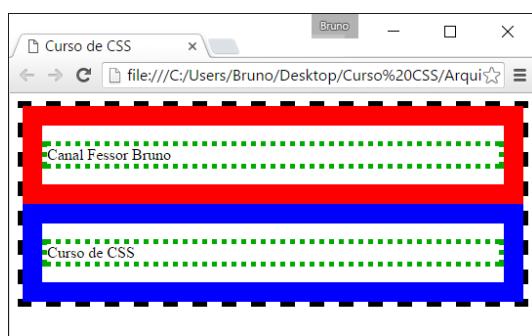
Vamos ao código de exemplo.

```
<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de CSS</title>
        <meta charset="UTF-8">
        <style rel="stylesheet">
            body{
                border: dashed 5px #000;
            }
            header{
                border: solid 20px #F00;
            }
            div{
                border: solid 20px #00F;
            }
            p{
                border: dotted 5px #0A0;
            }
        </style>
    </head>
    <body>

        <header>
            <p>Canal Fessor Bruno</p>
        </header>

        <div>
            <p>Curso de CSS</p>
        </div>

    </body>
</html>
```



Body = Bordas tracejadas com largura de 5 pixels de cor preto.

Header = Bordas sólidas com largura de 20 pixels de cor vermelho.

Div = Bordas sólidas com largura de 20 pixels de cor azul.

P = Bordas pontilhadas com largura de 5 pixels de cor verde.

Existem vários estilos que podemos aplicar, veja a seguir.

- dotted – Bordas pontilhadas
- dashed – Bordas tracejadas
- solid – Bordas com linha sólida / contínua

- double – Bordas com linhas duplas
- groove – Bordas em estilo 3D
- ridge – Bordas em estilo 3D com as cores ao contrário de groove
- inset – Efeito 3D no estilo baixo relevo.
- outset – Efeito 3D no estilo alto relevo.
- none – Sem bordas.
- hidden – Bordas ocultas.

Podemos usar as configurações separadas também, como no exemplo a seguir.

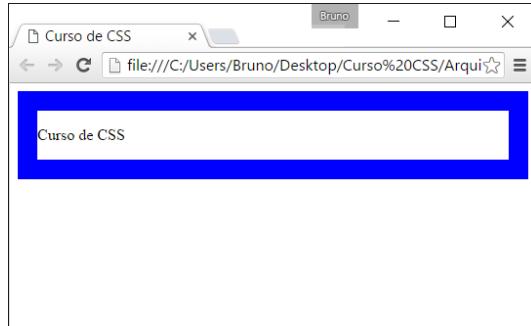
```
<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de CSS</title>
        <meta charset="UTF-8">
        <style rel="stylesheet">

            div{
                border-style: solid;
                border-width: 20px;
                border-color: #00F;
            }

        </style>
    </head>
    <body>

        <div>
            <p>Curso de CSS</p>
        </div>

    </body>
</html>
```



Lembre-se que podemos aplicar bordas a qualquer elemento, veja uma variação do código anterior aplicado a uma imagem.

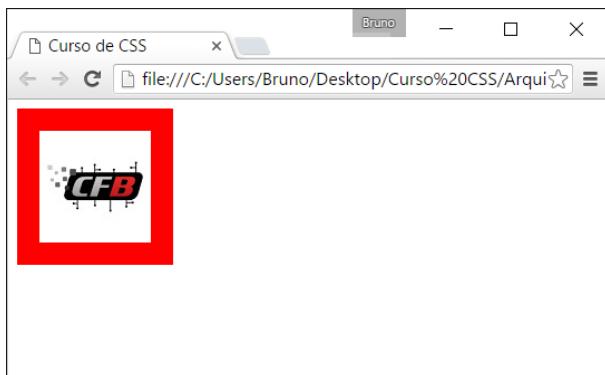
```
<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de CSS</title>
        <meta charset="UTF-8">
        <style rel="stylesheet">

            img{
                border: solid 20px #F00;
            }

        </style>
    </head>
    <body>

    </body>
</html>
```



Podemos trabalhar com as bordas de forma individual, ou seja cada borda com sua própria configuração, neste caso devemos usar uma propriedade para cada borda.

`border-top` = Borda superior.

`border-right` = Borda direita.

`border-bottom` = Borda inferior.

`border-left`: esquerda.

Veja o exemplo.

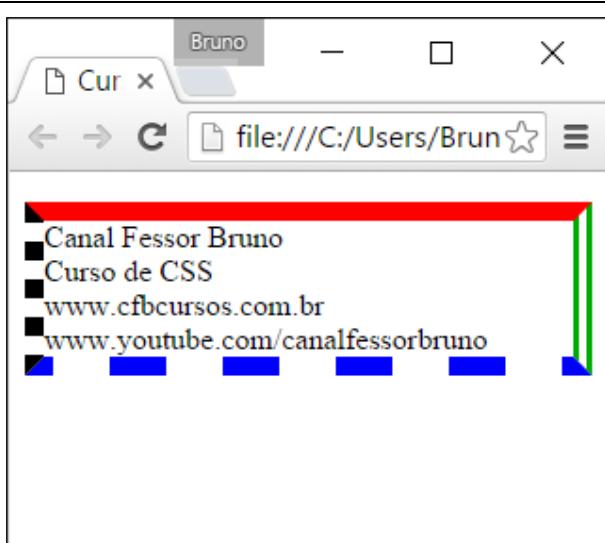
```
<!doctype html>
<html lang="pt-br">
<head>
    <title>Curso de CSS</title>
    <meta charset="UTF-8">
    <style rel="stylesheet">

        p{
            border-top: solid 10px #F00;
            border-right: double 10px #0A0;
            border-bottom: dashed 10px #00F;
            border-left: dotted 10px #000;
        }

    </style>
</head>
<body>

    <p>Canal Fessor Bruno<br>Curso de CSS<br>www.cfbcursos.com.br<br>www.youtube.com/canalfessorbruno</p>

</body>
</html>
```



border-radius / Cantos arredondada

Uma novidade de CSS3 em relação aos cantos arredondados, onde podemos trabalhar de forma que cada canto tenha um arredondamento ou de forma que todos os os cantos tenham o mesmo valor.

O uso da propriedade é bem simples, basta informar o valor do arredondamento com a unidade de medida desejada, pixel, %, em, etc.

O código de exemplo arredonda os quatro cantos em 20 pixels.

```
<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de CSS</title>
        <meta charset="UTF-8">
        <style rel="stylesheet">

            p{
                border: solid 1px #000;
                border-radius: 20px;
            }

        </style>
    </head>
    <body>

        <p>Canal Fessor Bruno<br>Curso de CSS<br>www.cfbcursos.com.br</p>

    </body>
</html>
```



Para arredondar os cantos com valores diferentes basta informar um valor para cada canto, começando do canto superior esquerdo.

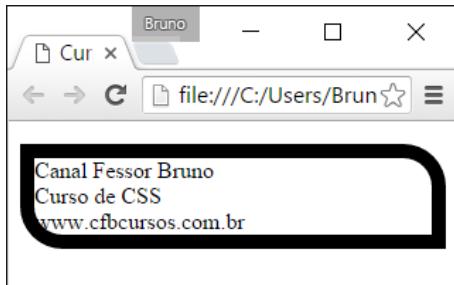
border-radius: superioEsquerdo superiorDireito inferiorDireito inferiorEsquerdo;

No nosso exemplo arredondamos em 30 pixels os cantos superior direito e inferior esquerdo.

```
<style rel="stylesheet">

    p{
        border: solid 1px #000;
        border-radius: 0px 30px 0px 30px;
    }

</style>
```



Podemos especificar o arredondamento das bordas de forma individual basta usar as propriedades a seguir.

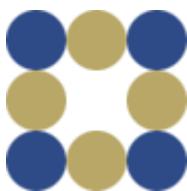
border-top-left-radius = Canto superior esquerdo.
border-top-right-radius = Canto superior direito.
border-bottom-right-radius = Canto inferior direito.
border-bottom-left-radius = Canto inferior esquerdo.

border-image – CSS3

Podemos definir uma imagem para servir de borda, caso queira uma borda com um visual personalizado, podemos adicionar uma imagem como borda e ainda configurar alguns aspectos desta imagem.

A imagem que iremos usar na borda é a imagem a seguir.

borda1.png

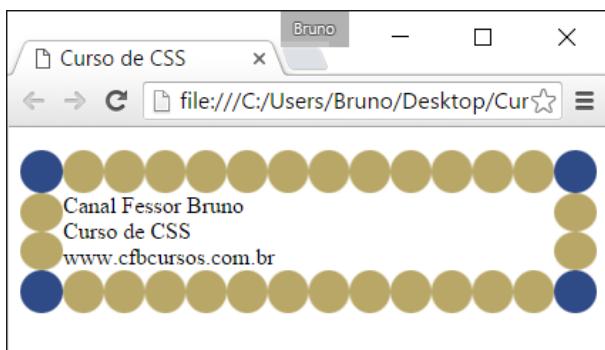


```
<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de CSS</title>
        <meta charset="UTF-8">
        <style rel="stylesheet">
            p{
                border: 30px;
                border-image: url(borda1.png) 30 round;
            }
        </style>
    </head>
    <body>

        <p>Canal Fessor Bruno<br>Curso de CSS<br>www.cfbcursos.com.br</p>

    </body>
</html>
```

Veja o resultado.

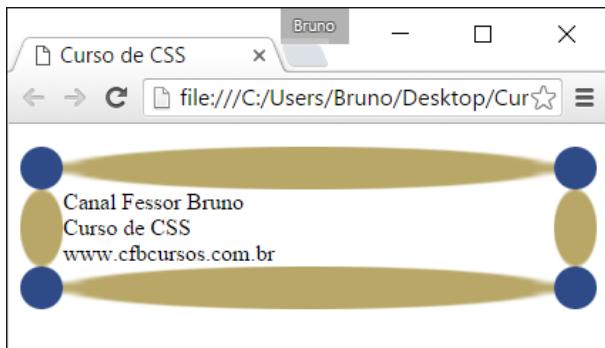


A sintaxe para border-image é: border-image: imagem corte contorno;

A imagem de fundo é cortada em 9 partes, quatro cantos, quatro bordas e o centro, então no parâmetro corte usamos o valor 30 que é a porcentagem da imagem que desejamos cortar, se usare 50 o corte será pela metade e 100 não haverá corte.

O último parâmetro configura como será mostrada as imagens dos meios, round indica que sórá contorno simples se configurar em stretch a imagem do meio será esticada, veja a seguir.

```
<style rel="stylesheet">
p{
    border: 30px;
    border-image: url(borda1.png) 30 stretch;
}
</style>
```



O parâmetro border: 30px; define o tamanho da borda.

Cores RGB decimal

Podemos usar o padrão rgb com valores decimais, isso pode facilitar um pouco o uso das cores, a regra é parecida, devemos informar um valor de intensidade para cada canal red, green e blue, também com profundidade 8 bits, mas neste caso vamos informar o valor com base decimal e não hexadecimal, ou seja, de 0 a 255.

A sintaxe para esta forma de trabalho é a seguinte: rgb(valor para red, valor para green, valor para blue).

Vamos criar 12 divs e aplicar uma cor de fundo usando este padrão a cada uma delas.

```
<!doctype html>
<html lang="pt-br">
<head>
    <title>Curso de CSS</title>
    <meta charset="UTF-8">
    <style rel="stylesheet">
        div{
            width:100px;
            height:100px;
            border:#000 solid 1px;
            display:inline-table;
            color:#000;
        }
        #gr1{
            background: rgb(255,255,255);
        }
        #gr2{
            background: rgb(0,0,0);
        }
        #gr3{
            background: rgb(255,0,0);
        }
        #gr4{
            background: rgb(0,255,0);
        }
        #gr5{
            background: rgb(0,0,255);
        }
        #gr6{
            background: rgb(0,255,255);
        }
        #gr7{
            background: rgb(255,0,255);
        }
    </style>
</head>
<body>
</body>
</html>
```

```
#gr8{ background: rgb(255,255,0); }
#gr9{ background: rgb(255,128,0); }
#gr10{ background: rgb(128,0,255); }
#gr11{ background: rgb(128,128,128); }
#gr12{ background: rgb(128,64,0); }

```

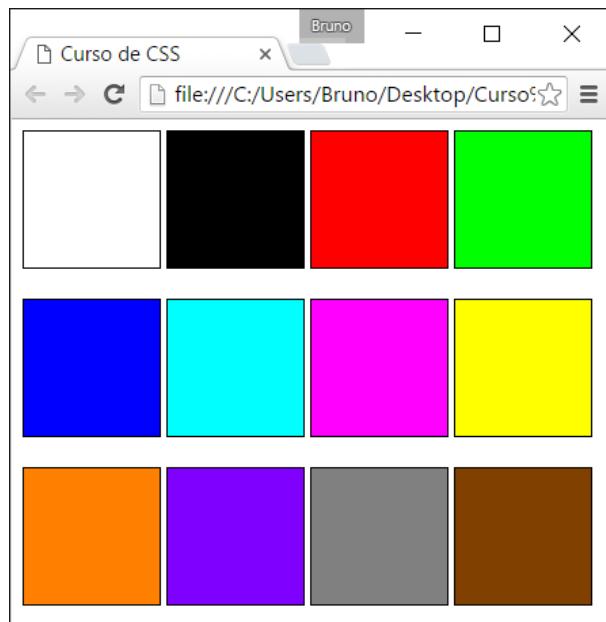
</style>

```
</head>
<body>
    <div id="gr1"></div>
    <div id="gr2"></div>
    <div id="gr3"></div>
    <div id="gr4"></div>
    <br><br>
    <div id="gr5"></div>
    <div id="gr6"></div>
    <div id="gr7"></div>
    <div id="gr8"></div>
    <br><br>
    <div id="gr9"></div>
    <div id="gr10"></div>
    <div id="gr11"></div>
    <div id="gr12"></div>

```

</body>

```
</html>
```



rgba – Transparência – CSS3

Um dos recursos introduzidos no CSS3 é a possibilidade trabalhar com transparência nas cores, a sintaxe é parecida com o modo rgb anterior, com a adição de um valor para indicar a opacidade, quanto menor o valor, menor opaco ou mais transparente.

A sintaxe é: `rgba(valor para red, valor para green, valor para blue, valor para opacidade)`.

No valor para opacidade inserimos um valor entre 0.0 (totalmente transparente) e 1.0 (totalmente opaco, sem transparência).

Em nosso exemplo, vamos criar um texto em `<h1>` e uma `<div>` por cima do texto, nesta div vamos aplicar a cor de fundo vermelho com 50% de opacidade, ou 50% transparente.

```
<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de CSS</title>
        <meta charset="UTF-8">
        <style rel="stylesheet">
            div{
                width:100px;
                height:100px;
                border:#000 solid 1px;
                display:inline-table;
                color:#000;
                position:absolute;
                top:0px;
                left:60px;
                background-color:rgba(255,0,0,0.5);
            }
        </style>
    </head>
    <body>
        <h1>Canal Fessor Bruno</h1>
        <div></div>
    </body>
</html>
```



Veja que a div está transparente.

margin – margens

Os elementos HTML possuem margens externas que podem ser controladas para configurar em certo espaço entre os elementos.

Em nosso código de exemplo, adicionamos margens de 50 pixels nos quatro lados.

```
<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de CSS</title>
        <meta charset="UTF-8">
        <style rel="stylesheet">
            body{
                margin:0px;
            }
            div{
                margin:20px;
                border:#000 solid 1px;
                display:inline-block;
            }
        </style>
    </head>
    <body>

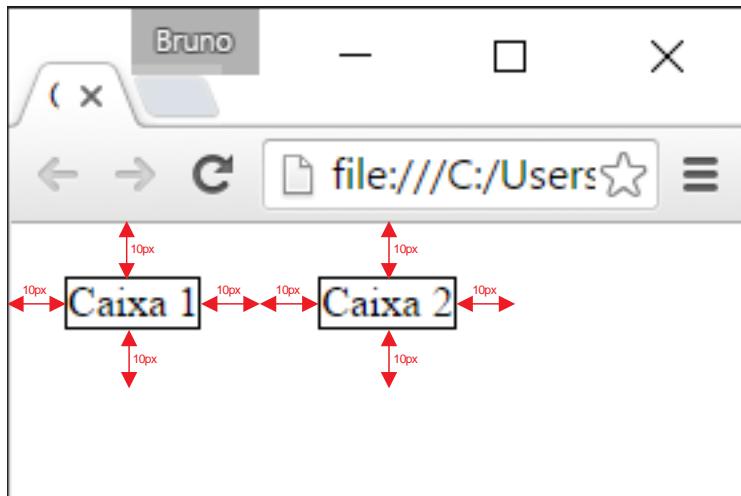
        <div>Caixa 1</div>
        <div>Caixa 2</div>

    </body>

```

```
</body>  
</html>
```

Veja o resultado do código.



Configuramos as margens do documento para zero pixels e adicionamos margens de 10 pixels às divs.

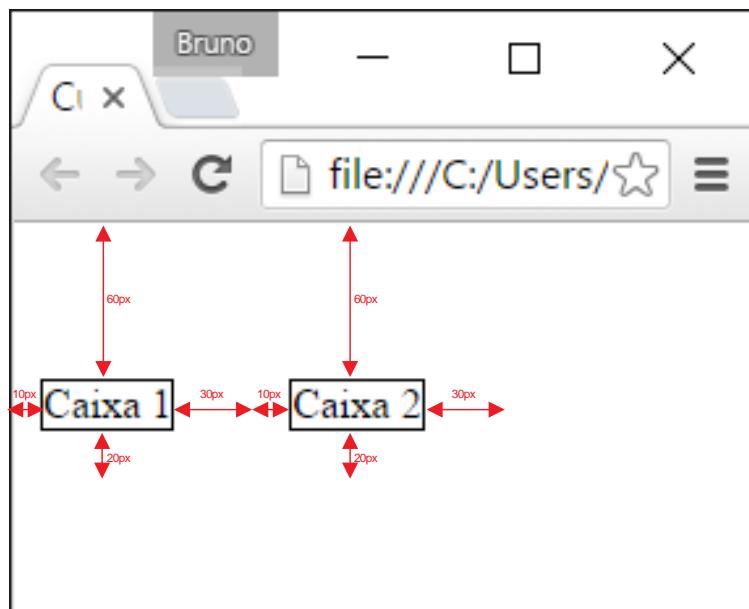
Não se preocupe com a propriedade “display” neste momento, veremos sobre ela mais adiante.

Podemos especificar valores diferentes para as margens, veja a sintaxe.

margin: topo direita inferior esquerda.

Veja o código.

```
<style rel="stylesheet">  
    body{  
        margin:0px;  
    }  
    div{  
        margin:60px 30px 20px 10px;  
        border:#000 solid 1px;  
        display:inline-block;  
    }  
</style>
```



Também podemos usar os parâmetros separadamente.

margin-top = Margem superior

margin-right = Margem direita

margin-bottom = Margem inferior

margin-left = Margem esquerda

Centralizar a página na janela e seus elementos

Podemos centralizar qualquer elemento na horizontal utilizando o parâmetro margin de forma simples, vamos ver as opções.

Nosso código de exemplo irá centralizar a <div> com a página.

```
<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de CSS</title>
        <meta charset="UTF-8">
        <style rel="stylesheet">
            div {
                width:300px;
                margin:auto;
                border:1px solid #000;
            }
        </style>
    </head>
    <body>

        <div>
            <p>Canal Fessor Bruno</p>
        </div>

    </body>
</html>
```



Mas o que acontece de inserirmos outro elemento que não seja um div em nossa página? Vamos inserir uma imagem e um parágrafo.

```
<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de CSS</title>
        <meta charset="UTF-8">
        <style rel="stylesheet">
            div {
                width:300px;
                margin: auto;
                border: 1px solid #000;
            }
        </style>
    </head>
    <body>

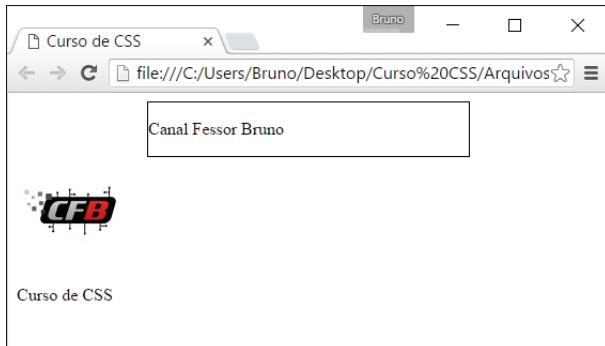
        <div>
            <p>Canal Fessor Bruno</p>
        </div>

        
        <p>Curso de CSS</p>

    </body>
</html>
```

```
</body>  
</html>
```

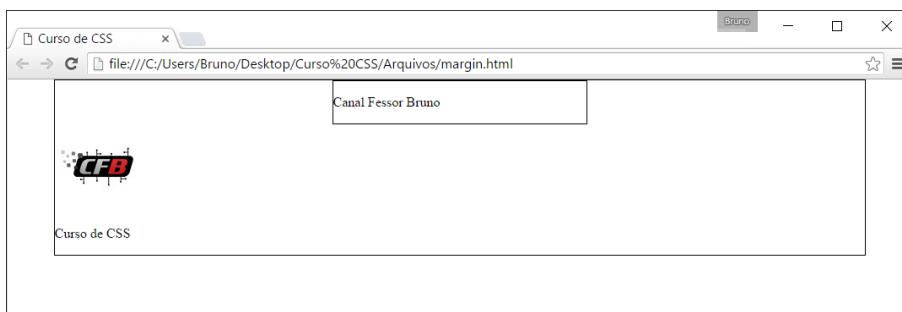
Vamos ver o resultado e notar que estes dois novos itens não foram centralizados.



É até óbvio porque isso acontece, porque os dois novos itens não foram centralizados, a resposta é porque eles não foram configurados para ficarem no centro igual a div anterior, correto? Correto! Mas imagina se tivermos que configurar a centralização de todos os elementos individualmente? Seria um pouco trabalhoso, além de não dar certo com todos os elementos.

Então ao invés de centralizar um a um, vamos configurar o `<body>`, isso mesmo, definir um tamanho e centralizar o `<body>`, assim todos os elementos que estiverem dentro dele serão centralizados, vejamos.

```
<style rel="stylesheet">  
    body {  
        width: 960px;  
        margin: auto;  
        border: #000 solid 1px;  
    }  
    div {  
        width: 300px;  
        margin: auto;  
        border: 1px solid #000;  
    }  
</style>
```



Note que o `body` foi centralizado com tamanho 960 pixels, todos os elementos dentro dele o acompanharam e a `div`, que já tinha configuração para ficar centralizada permaneceu no centro do `body`.

Para centralizar a imagem, basta coloca-la dentro de uma `div`, vamos realizar algumas alterações em nosso código para que todos os itens sejam centralizados, vou destacar as alterações em vermelho.

```
<!doctype html>  
<html lang="pt-br">  
    <head>  
        <title>Curso de CSS</title>  
        <meta charset="UTF-8">  
        <style rel="stylesheet">  
            body {  
                width: 960px;  
                margin: auto;  
                border: #000 solid 1px;  
            }  
            div {  
                width: 300px;  
                margin: auto;  
                border: 1px solid #000;  
            }  
            img {  
                width: 100%;  
                height: auto;  
            }  
        </style>  
    </head>  
    <body>  
        <div>  
            <img alt="Logo do Canal Fessor Bruno" data-bbox="95 635 145 655"/>  
            <div>Canal Fessor Bruno</div>  
        </div>  
        <div>Curso de CSS</div>  
    </body>  
</html>
```

```

        }
        #dv1 {
            width:300px;
            margin: auto;
            border: 1px solid #000;
        }
        #dv2 {
            width:100px;
            margin: auto;
        }
        p{
            text-align:center;
        }
    
```

</style>

</head>

<body>

<div id="dv1">

<p>Canal Fessor Bruno</p>

</div>

<div id="dv2">

</div>

<p>Curso de CSS</p>

</body>

</html>

Note que agora passamos a referenciar os ids das divs, pois, temos divs com tamanhos diferentes, a primeira tem largura 200 pixels e a segunda tem tamanho 100 pixels.

Note também que para centralizar o elemento <p> usamos uma configuração específica de alinhamento de texto, o parâmetro text-align com valor center.

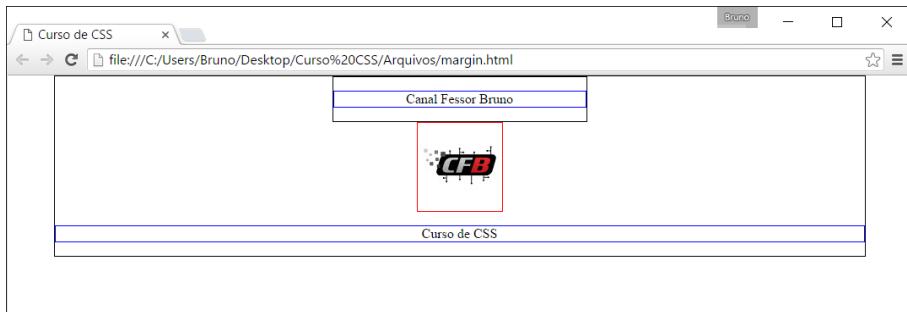


Vamos adicionar bordas nos outros dois elementos para visualizarmos melhor.

```

<style rel="stylesheet">
    body{
        width:960px;
        margin:auto;
        border:#000 solid 1px;
    }
    #dv1 {
        width:300px;
        margin: auto;
        border: 1px solid #000;
    }
    #dv2 {
        width:100px;
        margin: auto;
        border: 1px solid #F00;
    }
    p{
        text-align:center;
        border: 1px solid #00F;
    }
</style>

```



padding - margens internas

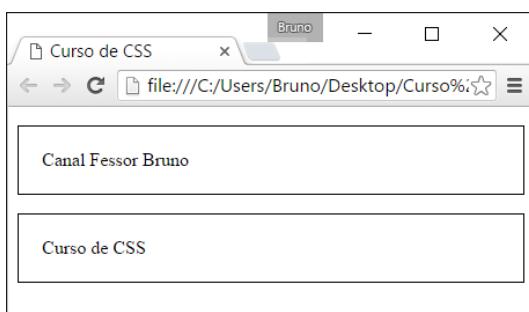
Padding são as margens internas dos elementos, usamos o padding da mesma forma que usamos margin, porém, na parte interna dos elementos, veja a ilustração.



Note que a margem “cresce” para o lado de fora da borda e padding para o lado de dentro.

Em nosso código de exemplo vamos configurar o padding para o elemento `<p>`, mas podemos usar em outros elementos como `<div>` `<section>` `<article>` `<h1>` a `<h6>` e vários outros elementos, vamos ver no nosso código onde configuraremos os elementos `<p>` com padding de 20 pixels.

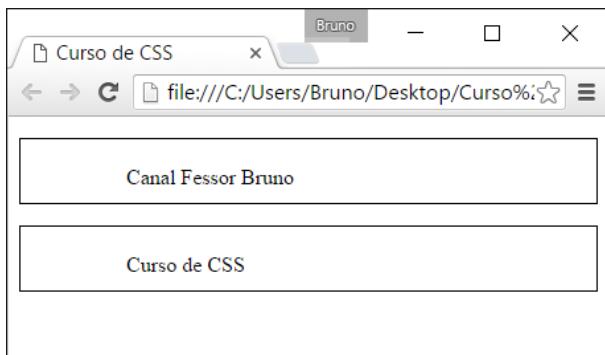
```
<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de CSS</title>
        <meta charset="UTF-8">
        <style rel="stylesheet">
            p{
                border:#000 solid 1px;
                padding:20px;
            }
        </style>
    </head>
    <body>
        <div>
            <p>Canal Fessor Bruno</p>
            <p>Curso de CSS</p>
        </div>
    </body>
</html>
```



Essa configuração faz com que seja adicionada um espaçamento interno de 20 pixels nos quatro lados.

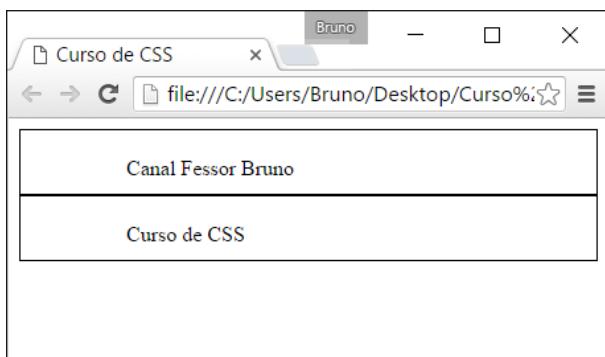
Vamos configurar valores direntes para cada canto, começando pelo topo 20px, direito 0px, inferior 10px, esquerdo 80px.

```
<style rel="stylesheet">
  p{
    border:#000 solid 1px;
    padding:20px 0px 10px 80px;
  }
</style>
```



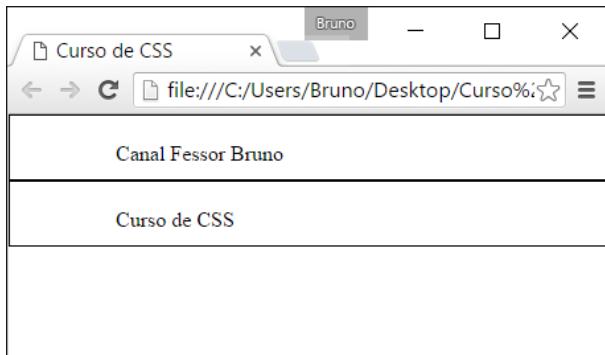
Naturalmente os elementos <p> tem uma margem externa, para eliminar esta margem basta configurar margin em 0 pixels.

```
<style rel="stylesheet">
  p{
    border:#000 solid 1px;
    padding:20px 0px 10px 80px;
    margin:0px;
  }
</style>
```



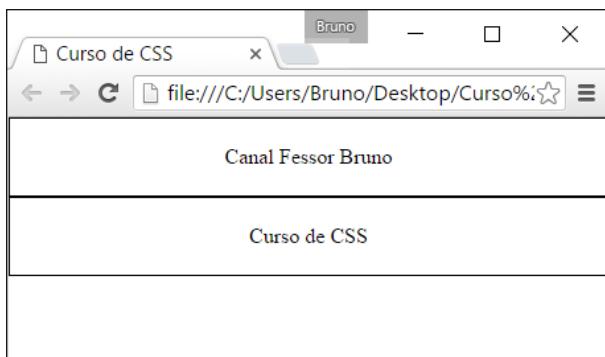
Vamos eliminar também a margem padrão do <body> configurando em zero.

```
<style rel="stylesheet">
  body{
    margin:0px;
  }
  p{
    border:#000 solid 1px;
    padding:20px 0px 10px 80px;
    margin:0px;
  }
</style>
```



Agora vamos reajustar os paddings dos elementos <p> e centralizar o texto.

```
<style rel="stylesheet">
    body{
        margin:0px;
    }
    p{
        border:#000 solid 1px;
        padding:20px;
        margin:0px;
        text-align:center;
    }
</style>
```



Para finalizar o assunto de padding, também podemos configurar de forma individual, veja os parâmetros.

padding-top = Espaçamento interno supedior
padding-right = Espaçamento interno direito
padding-bottom = Espaçamento interno inferior
padding-left = Espaçamento interno esquerdo

Gradientes – CSS3

Esta é uma nova e incrível possibilidade implementada em CSS3, trabalhar com cores gradientes diretamente pelo código sem a possibilidade de precisar inserir uma imagem é muito interessante.

Vamos ver como funcionam os gradientes em CSS.

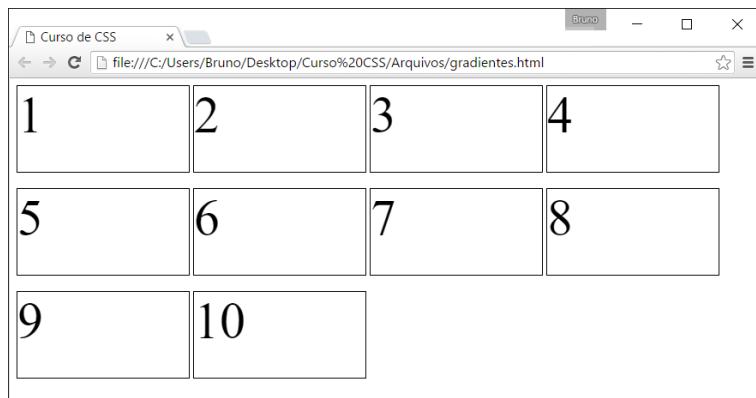
A sintaxe de utilização dos gradientes é: tipo-de-gradiente(direção, cor1, cor2, ...);

São dois tipos de gradiente que podemos trabalhar, linear e gradiente, com uma pequena variação cada que é repeating-linear-gradient e repeating-radial-gradient.

Vamos ao nosso código de exemplo, primeiro vamos adicionar 10 divs e depois vamos aplicar os gradientes ao background de cada div.

```
<!doctype html>
```

```
<html lang="pt-br">
  <head>
    <title>Curso de CSS</title>
    <meta charset="UTF-8">
    <style rel="stylesheet">
      div{
        width:200px;
        height:100px;
        border:#000 solid 1px;
        display:inline-table;
        color:#000;
        font-size:60px;
      }
    </style>
  </head>
  <body>
    <div id="gr1">1</div>
    <div id="gr2">2</div>
    <div id="gr3">3</div>
    <div id="gr4">4</div>
    <br><br>
    <div id="gr5">5</div>
    <div id="gr6">6</div>
    <div id="gr7">7</div>
    <div id="gr8">8</div>
    <br><br>
    <div id="gr9">9</div>
    <div id="gr10">10</div>
  </body>
</html>
```



Gradativamente vamos aplicando os gradientes nas divs.

linear-gradient

O gradiente linear possui uma variação linear de um ponto inicial A até um ponto final B, para configurar este gradiente basta informar o sentido da variação linear e as cores.

Vamos a algumas configurações.

`linear-gradient(to bottom, #F00, #FF0);` → Variação linear na vertical de cima para baixo com duas cores, vermelho e amarelo.

`linear-gradient(45deg, #F00, #00F, #FF0);` → Variação linear em 45 graus com três cores, vermelho, azul e amarelo.

`linear-gradient(to bottom, #F00 0%, #00F 25%, #FF0 50%);` → Variação linear na vertical de cima para baixo com três cores em posições definidas, vermelho no início, azul em 25% e amarelo em 50%.

`linear-gradient(to right, #F00, #FF0);` → Variação linear na horizontal da esquerda para direita com duas cores, vermelho e amarelo.

radial-gradient

O gradiente radial possui uma variação circular/radial do centro para fora, para configurar este gradiente informamos simplesmente as cores. Vamos a algumas configurações.

`radial-gradient(#F00, #FF0);` → Variação radial do centro para fora com duas cores, vermelho e amarelo.

`radial-gradient(#F00, #00F, #FF0);` → Variação radial do centro para fora com três cores, vermelho, azul e amarelo.

`radial-gradient(#F00 0%, #00F 25%, #FF0 50%);` → Variação radial do centro para fora com três cores em posições definidas, vermelho no início, azul em 25% e amarelo iniciando no meio.

`radial-gradient(circle, #F00, #FF0);` → Variação radial circular, independente do formato retangular, a variação do gradiente não será oval e sim circular, do centro para fora com duas cores, vermelho e amarelo.

repeating-linear-gradient & repeat-radial-gradient

Estes gradientes são uma variação dos gradientes linear e radial, eles funcionam repetindo o gradiente configurado até preencher

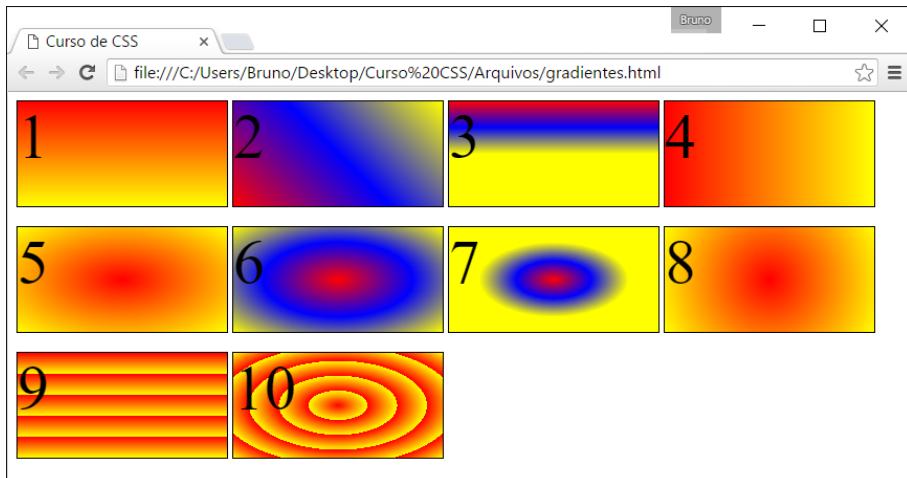
`repeating-linear-gradient(#F00, #FF0 20%);` → Repete o gradiente linear

`repeating-radial-gradient(#F00, #FF0 20%);` → Repete o gradiente radial

A seguir vamos aplicar estes gradientes no background das divs e conferir os resultados.

Aplicando os gradientes

```
<style rel="stylesheet">
div{
    width:200px;
    height:100px;
    border:#000 solid 1px;
    display:inline-table;
    color:#000;
    font-size:60px;
}
#gr1{
    background: linear-gradient(to bottom, #F00, #FF0);
}
#gr2{
    background: linear-gradient(45deg, #F00, #00F, #FF0);
}
#gr3{
    background: linear-gradient(to bottom, #F00 0%, #00F 25%, #FF0 50%);
}
#gr4{
    background: linear-gradient(to right, #F00, #FF0);
}
#gr5{
    background: radial-gradient(#F00, #FF0);
}
#gr6{
    background: radial-gradient(#F00, #00F, #FF0);
}
#gr7{
    background: radial-gradient(#F00 0%, #00F 25%, #FF0 50%);
}
#gr8{
    background: radial-gradient(circle, #F00, #FF0);
}
#gr9{
    background: repeating-linear-gradient(#F00, #FF0 20%);
}
#gr10{
    background: repeating-radial-gradient(#F00, #FF0 20%);
}
</style>
```



Alguns browsers tem uma sintaxe específica para uso do gradiente, configura a seguir

Safari 5.1 até 6.0 ➔ -webkit-linear-gradient(#F00, #FF0);

Opera 11.1 até 12.0 ➔ -o-linear-gradient(#F00, #FF0);

Firefox 3.6 até 15 ➔ -moz-linear-gradient(#F00, #FF0);

Sintaxe padrão ➔ linear-gradient(#F00, #FF0);

shadow – Efeito de sombra – CSS3

Existem duas propriedades para trabalharmos com sombras, são elas, text-shadow e box-shadow, obviamente a primeira para aplicar sombras em textos e a segunda nos demais elementos com <div> por exemplo.

text-shadow

Vamos começar com um código para sombra básica, simplesmente adicionando uma sombra a uma distância de 5 pixels na horizontal e 5 pixels na vertical.

```
<!doctype html>
<html lang="pt-br">
  <head>
    <title>Curso de CSS</title>
    <meta charset="UTF-8">
    <style rel="stylesheet">
      h1{
        text-shadow:5px 5px;
      }
    </style>
  </head>
  <body>
    <h1>Canal Fessor Bruno</h1>
  </body>
</html>
```



Simples, text-shadow:distânciaHorizontal distânciasVertical;

Mas o resultado final foi uma sombra muito sólida e pesada, vamos a outras configurações para a sombra.

`text-shadow:distânciaHorizontal distânciaVertical blur cor;`

A propriedade blur consiste no enevoamento/embaçamento da sombra.

Vamos alterar nosso código e melhorar a sombra.

```
<style rel="stylesheet">
    h1{
        color:#F00;
        text-shadow:3px 3px 6px #000;
    }
</style>
```



Um aspecto interessante é que podemos adicionar mais de uma sombra no mesmo texto, vamos adicionar duas sombras.

```
<style rel="stylesheet">
    h1{
        color:#000;
        text-shadow:3px 3px 6px #F00, -3px -3px 6px #0F0;
    }
</style>
```



Podemos adicionar mais sombras, três, quatro, cinco, quantas forem preciso, mas lembre-se que exagerar nos efeitos torna o elemento mais difícil de ser renderizado.

Podemos usar sombras como se fosse brilho, basta posicionar em distância 0 pixels, veja.

```
<style rel="stylesheet">
    h1{
        font-size:50px;
        color:#800;
        text-shadow:0px 0px 5px #FFF, 0px 0px 15px #F00;
    }
</style>
```



Adicionei duas sombras como brilho, uma branca bem curtinha e uma vermelha maior.

E assim podemos “brincar” criando vários efeitos, basta usar a criatividade.

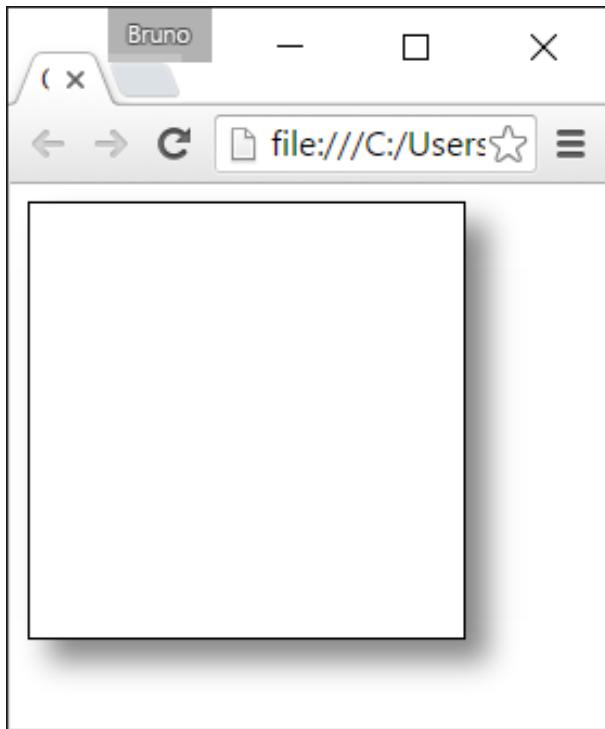
```
<style rel="stylesheet">
    h1{
        font-size:50px;
        color:#FFF;
        text-shadow:1px 1px 2px #000, 0px 0px 7px #800, 0px 0px 20px #F00;
    }
</style>
```



box-shadow

A propriedade box-shadow tem a mesma sintaxe de text-shadow, porém aplica sombras aos outros elementos diferentes de texto, como `<div>` por exemplo.

```
<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de CSS</title>
        <meta charset="UTF-8">
        <style rel="stylesheet">
            div{
                width:200px;
                height:200px;
                background-color:#FFF;
                border:#000 solid 1px;
                box-shadow:10px 10px 20px #888;
            }
        </style>
    </head>
    <body>
        <div></div>
    </body>
</html>
```



Lembre-se, podemos aplicar box-shadow em qualquer elemento que não seja um texto, para textos devemos usar text-shadow.

Comentários

É muito útil podermos adicionar textos em nosso código que não serão interpretados como parte do código, em CSS temos uma forma bem simples de adicionar comentários, basta usar /* para iniciar e */ para finalizar, todo conteúdo que for inserido como comentário não será interpretado como código de formatação.

```
<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de CSS</title>
        <meta charset="UTF-8">
        <style rel="stylesheet">
            div{
                width:200px; /* Largura */
                height:200px; /* Altura */
                background-color:#FFF; /* Cor de fundo */
                border:#000 solid 1px; /* Bordas */
            }
        </style>
    </head>
    <body>
        <div></div>
    </body>
</html>
```

OBS: Podemos adicionar comentários em qualquer conteúdo do código CSS, mas lembre-se, se um comando for comentado ele não será interpretado.

width e height / largura e altura

As propriedade width e height são respectivamente para definir largura e altura do elemento, são propriedade de dimensionamento.

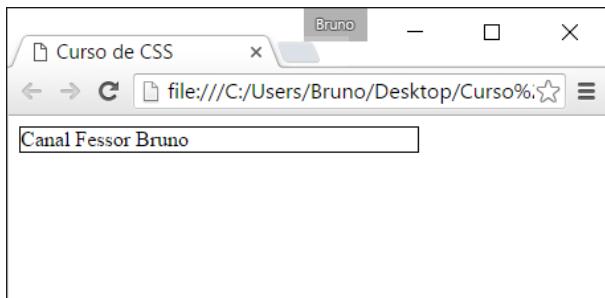
Já vimos algumas vezes, mas vamos aprender um pouco mais destas propriedades.

width – largura

A propriedade width define a largura de um elemento, pode usar várias unidades de medida, px, %, em, cm, etc.

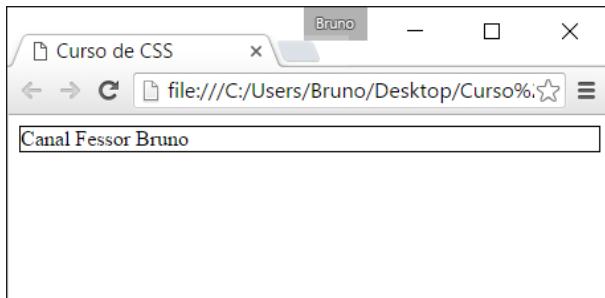
```
<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de CSS</title>
        <meta charset="UTF-8">
        <style rel="stylesheet">
            #dv1{
                width:300px;
                border:#000 solid 1px;
            }
        </style>
    </head>
    <body>
        <div id="dv1">Canal Fessor Bruno</div>
    </body>
</html>
```

No código acima definimos a largura da nossa div como 300 pixels.



Vamos mudar para porcentagem.

```
<style rel="stylesheet">
    #dv1{
        width:100%;
        border:#000 solid 1px;
    }
</style>
```



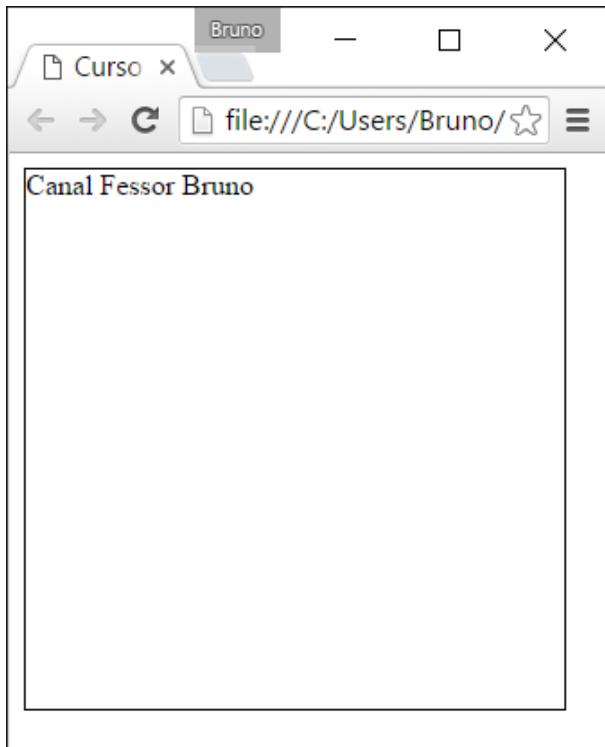
Veja que agora <div> sempre terá a largura total da tela.

height – Altura

Já a propriedade height define a altura de um determinado elemento, vamos usar o mesmo código anterior.

```
<style rel="stylesheet">
    #dv1{
        width:300px;
        height:300px;
        border:#000 solid 1px;
    }
</style>
```

Aqui definimos tanto a largura como a altura da div para 300 pixels.



max-width & min-width

Podemos configurar um valor para largura máxima e mínima, em caso de uso de unidades de medida relativas como %, o elemento será aumentado ou diminuído de acordo com a largura da tela, mas nunca será menor que min-width ou maior que max-width.

```
<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de CSS</title>
        <meta charset="UTF-8">
        <style rel="stylesheet">
            #dv1{
                width:100%;
                max-width:500px;
                min-width:200px;
                border:#000 solid 1px;
            }
        </style>
    </head>
    <body>
        <div id="dv1">CFB</div>
    </body>
</html>
```

Com o código acima a <div> será redimensionada de acordo com o tamanho da janela, mas, nunca será menor que 200 pixels e quando a tela for maior que 500 pixels a <div> não acompanhará mais o tamanho da tela, ou seja, entre 200 e 500 pixels o elemento segue a largura da tela.

OBS: Também podemos definir os valores mínimo e máximo para altura basta usar as propriedades max-height e min-height.

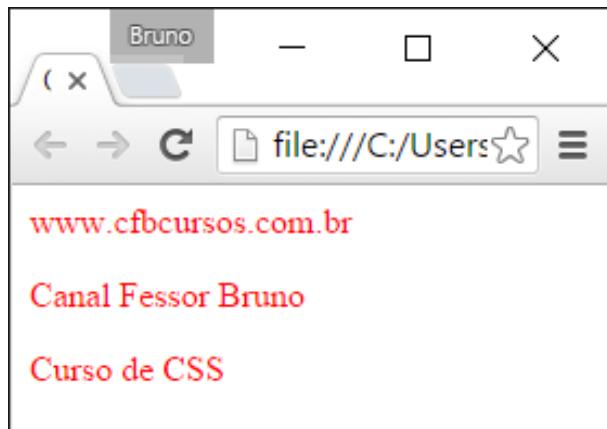
Formatações para textos

Vamos ver neste capítulo os parâmetros disponíveis para formatação de textos, são várias configurações disponíveis, vamos ver a seguir.

color – cor

A propriedade color configura a cor do texto, podemos usar qualquer uma das formas de criação de cores que aprendemos.

```
<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de CSS</title>
        <meta charset="UTF-8">
        <style rel="stylesheet">
            #dv1{
                color:#FF0000;
            }
            #p1{
                color:#F00;
            }
            #p2{
                color:rgb(255,0,0);
            }
        </style>
    </head>
    <body>
        <div id="dv1">www.cfbcursos.com.br</div>
        <p id="p1">Canal Fessor Bruno</p>
        <p id="p2">Curso de CSS</p>
    </body>
</html>
```



text-align - alinhamento

Esta propriedade define o alinhamento do texto.

```
<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de CSS</title>
        <meta charset="UTF-8">
        <style rel="stylesheet">
            #dv1{
                text-align:center;
            }
            #p1{
                text-align:center;
            }
        </style>
    </head>
    <body>
        <div id="dv1">www.cfbcursos.com.br</div>
        <p id="p1">Canal Fessor Bruno</p>
    </body>
</html>
```



Os valores possíveis para text-align são:

left → Esquerda

right → Direita

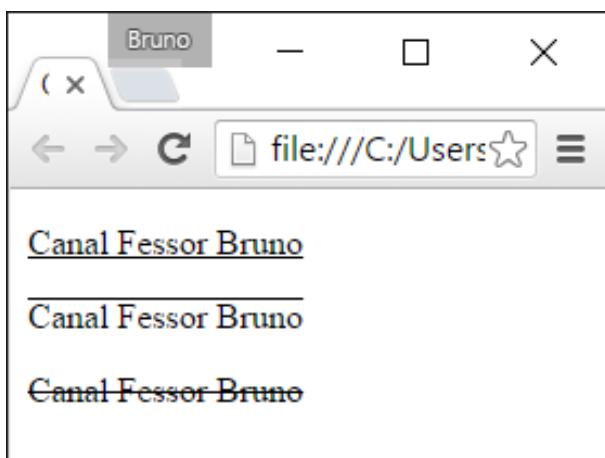
center → Centro

justify → Justificado

text-decoration – Decoração

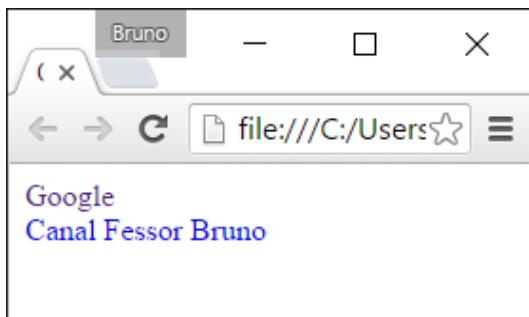
Configura o método de decoração do texto, a decoração são tipos de linhas como abaixo do texto, acima do texto ou riscado.

```
<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de CSS</title>
        <meta charset="UTF-8">
        <style rel="stylesheet">
            #p1{
                text-decoration:underline;
            }
            #p2{
                text-decoration:overline;
            }
            #p3{
                text-decoration:line-through;
            }
        </style>
    </head>
    <body>
        <p id="p1">Canal Fessor Bruno</p>
        <p id="p2">Canal Fessor Bruno</p>
        <p id="p3">Canal Fessor Bruno</p>
    </body>
</html>
```



Ainda podemos usar text-decoration para remover o sublinhado natural dos links.

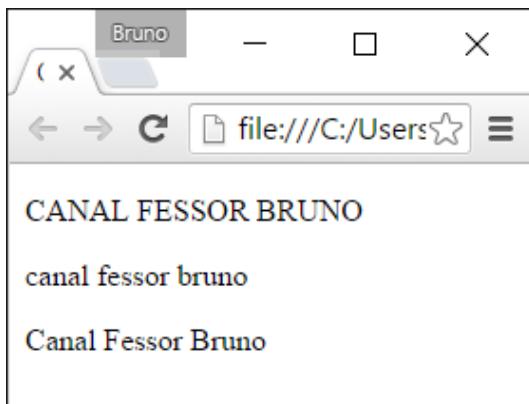
```
<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de CSS</title>
        <meta charset="UTF-8">
        <style rel="stylesheet">
            a{
                text-decoration:none;
            }
        </style>
    </head>
    <body>
        <a href="http://www.google.com.br">Google</a>
        <br>
        <a href="http://www.cfbcursos.com.br">Canal Fessor Bruno</a>
    </body>
</html>
```



text-transform – Transformações

Com esta propriedade podemos configurar letras em maiúsculas, minúsculas e primeiras letras em maiúsculo.

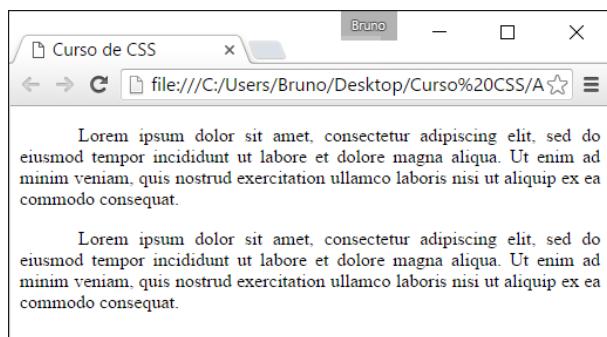
```
<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de CSS</title>
        <meta charset="UTF-8">
        <style rel="stylesheet">
            #p1{
                text-transform:uppercase; /* Converte em maiúsculo */
            }
            #p2{
                text-transform:lowercase; /* Converte em minúsculo */
            }
            #p3{
                text-transform:capitalize; /* Converte as primeiras letras em maiúsculo */
            }
        </style>
    </head>
    <body>
        <p id="p1">canal fessor bruno</p>
        <p id="p2">CANAL FESSOR BRUNO</p>
        <p id="p3">canal fessor bruno<p>
    </body>
</html>
```



text-indent – Endentação

Este parâmetro configura a endentação da primeira linha, definindo o tamanho do espaçamento do parágrafo.

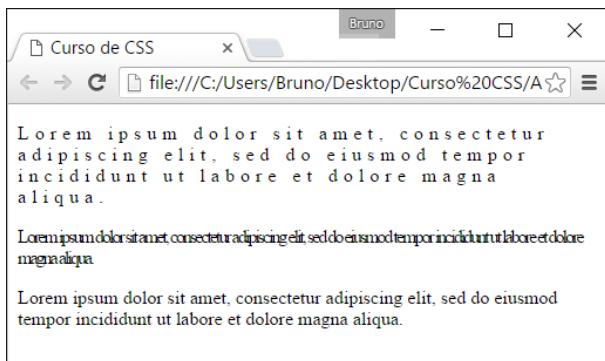
```
<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de CSS</title>
        <meta charset="UTF-8">
        <style rel="stylesheet">
            p {
                text-indent: 50px;
                text-align: justify;
            }
        </style>
    </head>
    <body>
        <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.</p>
        <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.</p>
    </body>
</html>
```



letter-spacing – Espaçamento entre as letras

Com letter-spacing podemos configurar o espaçamento entre as letras.

```
<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de CSS</title>
        <meta charset="UTF-8">
        <style rel="stylesheet">
            #p1 {
                letter-spacing: 5px;
            }
            #p2 {
                letter-spacing: -2px;
            }
        </style>
    </head>
    <body>
        <p id="p1">Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor
incididunt ut labore et dolore magna aliqua.</p>
        <p id="p2">Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor
incididunt ut labore et dolore magna aliqua.</p>
        <p id="p3">Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor
incididunt ut labore et dolore magna aliqua.</p>
    </body>
</html>
```



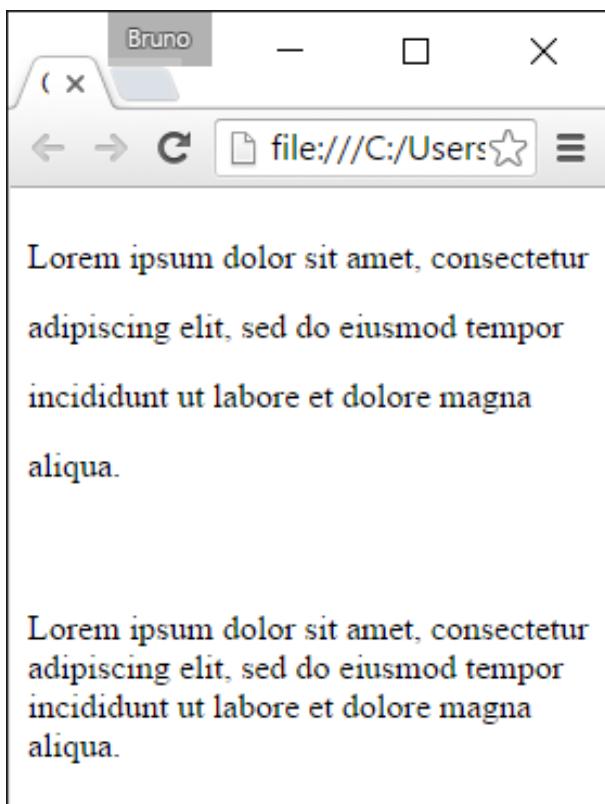
line-height – Espaçamento entre as linhas

A propriedade line-height especifica distância entre as linhas de um texto.

```
<!doctype html>
<html lang="pt-br">
  <head>
    <title>Curso de CSS</title>
    <meta charset="UTF-8">
    <style rel="stylesheet">
      #p1{
        line-height: 2;
      }
    </style>
  </head>
  <body>
    <p id="p1">Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor  

    incididunt ut labore et dolore magna aliqua.</p>
    <br>
    <p id="p2">Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor  

    incididunt ut labore et dolore magna aliqua.</p>
  </body>
</html>
```



direction – Direção do texto

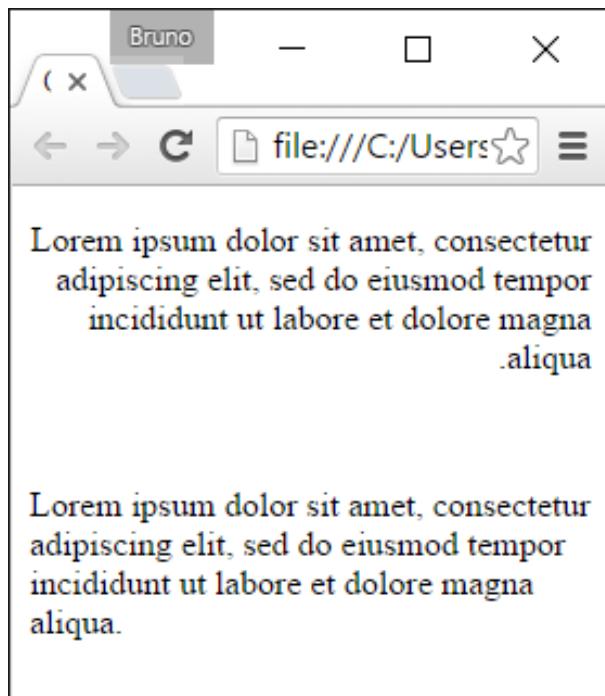
Usando a propriedade direction podemos definir a direção do texto da esquerda para direita ou da direita para esquerda.

Podemos usar os valores:

rtl → Right To Left (Direita para Esquerda)

ltr → Left To Right (Esquerda para Direita)

```
<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de CSS</title>
        <meta charset="UTF-8">
        <style rel="stylesheet">
            #p1{
                direction: rtl;
            }
            #p2{
                direction: ltr;
            }
        </style>
    </head>
    <body>
        <p id="p1">Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor
incididunt ut labore et dolore magna aliqua.</p>
        <br>
        <p id="p2">Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor
incididunt ut labore et dolore magna aliqua.</p>
    </body>
</html>
```



word-spacing – Espaçamento entre as palavras

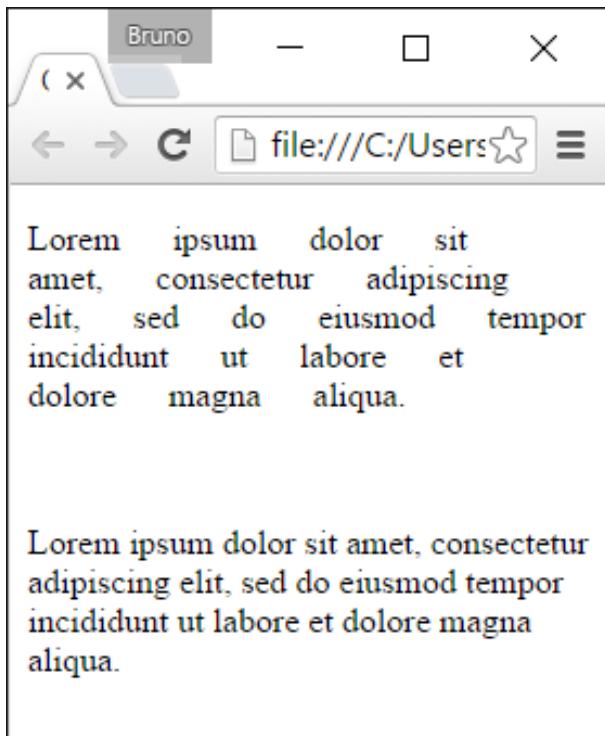
Configuração do espaçamento entre as palavras do texto.

```
<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de CSS</title>
        <meta charset="UTF-8">
        <style rel="stylesheet">
```

```

        #p1{
            word-spacing: 20px;
        }
    </style>
</head>
<body>
    <p id="p1">Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor
incididunt ut labore et dolore magna aliqua.</p>
    <br>
    <p id="p2">Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor
incididunt ut labore et dolore magna aliqua.</p>
</body>
</html>

```



white-space

A propriedade white-space especifica como o texto dos elementos <p> vão se comportar.

Veja o código de exemplo.

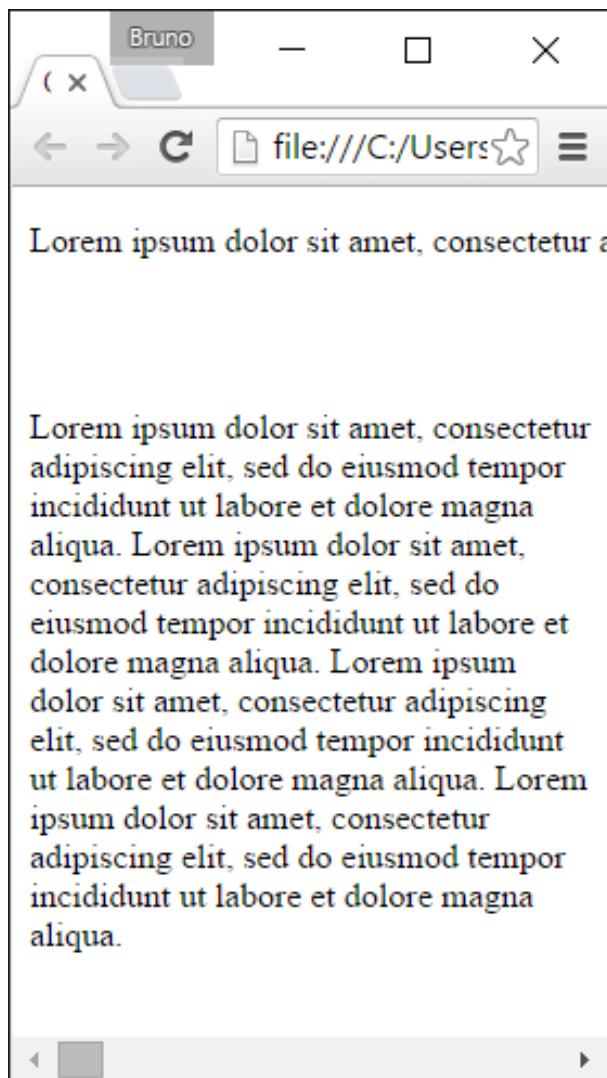
```

<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de CSS</title>
        <meta charset="UTF-8">
        <style rel="stylesheet">
            #p1{
                white-space: nowrap;
            }
        </style>
    </head>
    <body>
        <p id="p1">
            Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt
            ut labore et dolore magna aliqua. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod
            tempor incididunt ut labore et dolore magna aliqua. Lorem ipsum dolor sit amet, consectetur adipiscing elit,
            sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Lorem ipsum dolor sit amet, consectetur
            adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
        </p>
        <br><br>
        <p id="p2">
            Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt
            ut labore et dolore magna aliqua. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod
            tempor incididunt ut labore et dolore magna aliqua. Lorem ipsum dolor sit amet, consectetur adipiscing elit,
            sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
        </p>
    </body>
</html>

```

```
sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
```

```
</p>  
</body>  
</html>
```



Note que no primeiro parágrafo não houve quebra de linha no texto, isso por causa do valor “nowrap”.

overflow – CSS3 - Texto transbordado

A propriedade overflow configura como será o comportamento em casos de texto transbordado para fora dos limites especificados, em nosso exemplo definimos o tamanho do parágrafo em 20 pixels, como o texto é grande obviamente que 200 pixels não serão suficientes para comportar todo o texto.

```
<!doctype html>  
<html lang="pt-br">  
  <head>  
    <title>Curso de CSS</title>  
    <meta charset="UTF-8">  
    <style rel="stylesheet">  
      #p1{  
        white-space: nowrap;  
        width: 200px;  
        border: 1px solid #000000;  
        overflow: hidden;  
      }  
      #p2{  
        white-space: nowrap;
```

```

        width: 200px;
        border: 1px solid #000000;
    }

```

```
</style>
```

```
</head>
```

```
<body>
```

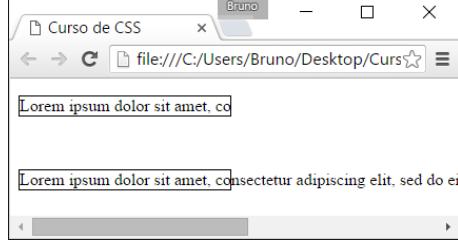
```
    <p id="p1">Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor  
incididunt ut labore et dolore magna aliqua.</p>
```

```
    <br>
```

```
    <p id="p2">Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor  
incididunt ut labore et dolore magna aliqua.</p>
```

```
</body>
```

```
</html>
```



No primeiro parágrafo usamos o valor “hidden” que diz para esconder o texto transbordado.

Veja os valores possíveis.

visible → O texto transbordado permanece visível, é a propriedade padrão.

hidden → O texto transbordado fica oculto.

scroll → Disponibiliza uma barra de rolagem para rolar o texto transbordado.

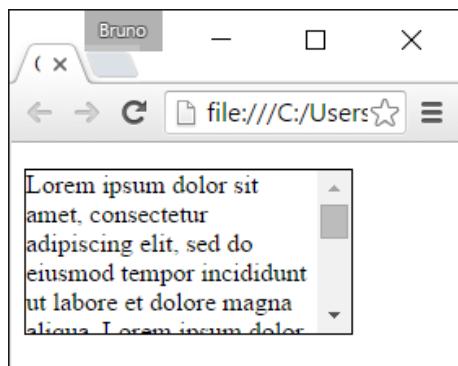
auto → Automaticamente define se será mostrada ou não a barra de rolagem.

Veja um exemplo com o valor auto.

```

<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de CSS</title>
        <meta charset="UTF-8">
        <style rel="stylesheet">
            #p1{
                width:200px;
                height:100px;
                border:1px solid #000000;
                overflow:auto;
            }
        </style>
    </head>
    <body>
        <p id="p1">Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor  
incididunt ut labore et dolore magna aliqua. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do  
eiusmod tempor incididunt ut labore et dolore magna aliqua. Lorem ipsum dolor sit amet, consectetur  
adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.</p>
    </body>
</html>

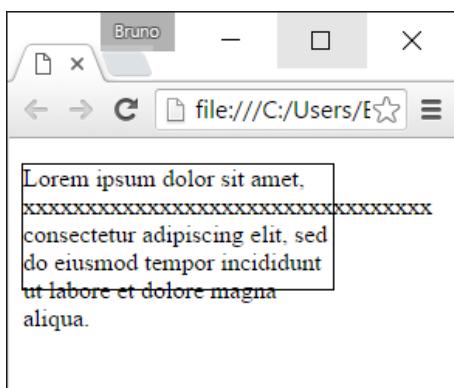
```



word-wrap – CSS3 – Quebra de texto longo

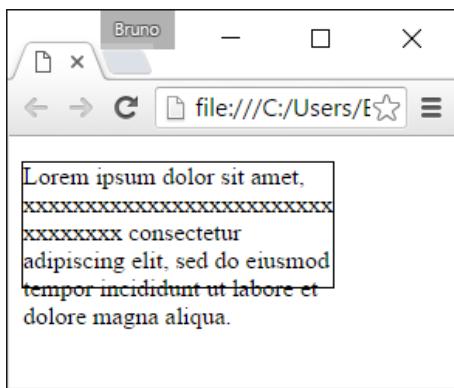
Define se alguma palavra muito grande será quebrada caso não caiba no seu container, veja o exemplo

```
<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de CSS</title>
        <meta charset="UTF-8">
        <style rel="stylesheet">
            #p1{
                width:200px;
                height:80px;
                border:1px solid #000000;
            }
        </style>
    </head>
    <body>
        <p id="p1">Lorem ipsum dolor sit amet, xxxxxxxxxxxxxxxxxxxxxxxx consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.</p>
    </body>
</html>
```



Note que a “palavra” xxx... é muito grande e transborda para fora dos limites do <p>, vamos adicionar o parâmetro word-wrap: break-word e ver o resultado.

```
<style rel="stylesheet">
    #p1{
        width:200px;
        height:80px;
        border:1px solid #000000;
        word-wrap: break-word;
    }
</style>
```

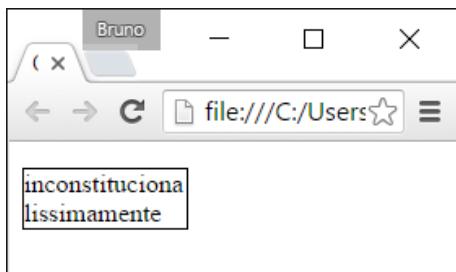


Note que agora o texto xxx é quebrado e não excede os limites do <p>.

word-break – CSS3 – Quebra a palavra

Quebra a palavra caso ela não caiba no seu container, usando a separação de sílabas correta.

```
<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de CSS</title>
        <meta charset="UTF-8">
        <style rel="stylesheet">
            #p1{
                width:100px;
                border:1px solid #000000;
                word-break: break-all;
            }
        </style>
    </head>
    <body>
        <p id="p1">inconstitucionalissimamente</p>
    </body>
</html>
```



text-align-last – CSS3 – Alinhamento da última linha do texto

Define como será o alinhamento da última linha do parágrafo, independente do alinhamento ao texto.

```
p{
    text-align:justify;
    text-align-last:center;
}
```

Formatações para fontes

A seguir vamos conferir as propriedades disponíveis para formatação da fonte, anteriormente vimos as formatações do texto de forma geral, agora vamos ver da fonte em si.

font-family

Com parâmetro font-family podemos configurar grupos de fontes para serem usadas em um determinado texto, a vantagem de especificar grupos de fontes é tentar garantir que sejam usadas somente fontes definidas pelo programador, o que não é 100% garantido.

No parâmetro font-family especificamos várias fontes, o browser tentará usar a primeira especificada, caso não seja encontrada buscará pela segunda, se não for encontrada buscará pela terceira e assim sucessivamente, se não encontrar nenhuma das opções será usada a fonte padrão do browser.

```
<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de CSS</title>
        <meta charset="UTF-8">
        <style rel="stylesheet">
            #p1{
                font-family:Arial, "Times New Roman", sans-serif;
            }
        </style>
    </head>
    <body>
        <p id="p1">Canal Fessor Bruno</p>
    </body>
</html>
```

Em nosso código estamos usando as fontes Arial, Times New Roman e sans-serif, como arial foi encontrada está é a fonte que será usada.

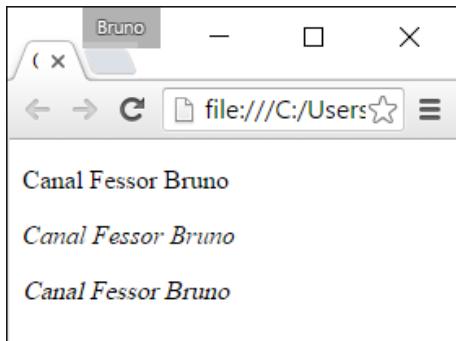


Podemos especificar quantas fontes forem necessárias.

font-style

Com o parâmetro `font-style` podemos definir texto em itálico, oblíquo ou normal que é o padrão sem estilos.

```
<!doctype html>
<html lang="pt-br">
  <head>
    <title>Curso de CSS</title>
    <meta charset="UTF-8">
    <style rel="stylesheet">
      #p1{
        font-style:normal; /*Valor padrão */
      }
      #p2{
        font-style:italic;
      }
      #p3{
        font-style:oblique;
      }
    </style>
  </head>
  <body>
    <p id="p1">Canal Fessor Bruno</p>
    <p id="p2">Canal Fessor Bruno</p>
    <p id="p3">Canal Fessor Bruno</p>
  </body>
</html>
```



font-size

Com este parâmetro podemos definir o tamanho da fonte.

```
<!doctype html>
<html lang="pt-br">
  <head>
    <title>Curso de CSS</title>
    <meta charset="UTF-8">
    <style rel="stylesheet">
      #p1{
        font-size:30px;
      }
      #p2{
        font-size:40px;
      }
    </style>
  </head>
  <body>
    <p id="p1">Canal Fessor Bruno</p>
    <p id="p2">Canal Fessor Bruno</p>
  </body>
</html>
```

```

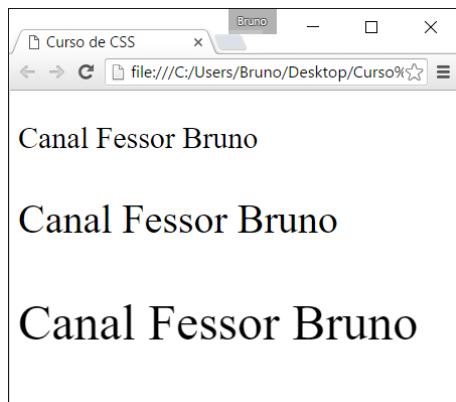
        }
    #p3{
        font-size:50px;
    }

```

```

</style>
</head>
<body>
    <p id="p1">Canal Fessor Bruno</p>
    <p id="p2">Canal Fessor Bruno</p>
    <p id="p3">Canal Fessor Bruno</p>
</body>
</html>

```



No código acima nós usamos a unidade pixels, mas podemos usar várias outras unidades em, pt e %.

font-weight

Configuração do peso da fonte, de forma mais simples é o tipo do negrito.

```

<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de CSS</title>
        <meta charset="UTF-8">
        <style rel="stylesheet">
            #p1{
                font-weight:bold;
            }
            #p2{
                font-weight:lighter;
            }
        </style>
    </head>
    <body>
        <p id="p1">Canal Fessor Bruno</p>
        <p id="p2">Canal Fessor Bruno</p>
    </body>
</html>

```



font-variant

Com o parâmetro font-variant todos os caracteres são convertidos para maiúsculos, porém ficam em tamanho menor do que a primeira letra de cada palavra.

```
<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de CSS</title>
        <meta charset="UTF-8">
        <style rel="stylesheet">
            #p1{
                font-size:30px;
                font-variant: small-caps;
            }
        </style>
    </head>
    <body>
        <p id="p1">Canal Fessor Bruno</p>
    </body>
</html>
```



@font-face – CSS3

Com o uso de @font-face podemos usar praticamente qualquer fonte em nossa página, sem se preocupar se no computador de quem estará vendo o site terá a fonte ou não, isso porque podemos enviar o arquivo da fonte para nosso servidor ou usar o link de fonte disponível na web.

Em nosso exemplo, vou usar uma fonte diferente que não tem como garantir que todos os sistemas operacionais tenham esta fonte, “Magneto Negrito” o nome do arquivo é “MAGNETOB.TTF”.

```
<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de CSS</title>
        <meta charset="UTF-8">
        <style rel="stylesheet">
            @font-face {
                font-family:fonteCurso;
                src:url(MAGNETOB.TTF) ;
            }

            #p1{
                font-family:fonteCurso;
                font-size:30px;
            }
        </style>
    </head>
    <body>
        <p id="p1">Canal Fessor Bruno</p>
    </body>
</html>
```



Podemos usar fontes on-line, o google disponibiliza uma vasta biblioteca de fontes, acesse www.google.com/fonts.

The screenshot shows the Google Fonts interface with a search term entered. Below the search bar, there are filter options for 'Thickness', 'Slant', 'Width', and 'Script' (set to Latin). The results are displayed in a grid format:

- Open Sans**, 10 Styles by Steve Matteson: Preview text "Grumpy wizards make toxic brew for the evil Queen and Jack." with a 'Normal 400' style.
- Roboto**, 12 Styles by Christian Robertson: Preview text "Grumpy wizards make toxic brew for the evil Queen and Jack." with a 'Normal 400' style.
- Lato**, 10 Styles by Lukasz Dziedzic: Preview text "Grumpy wizards make toxic brew for the evil Queen and Jack." with a 'Normal 400' style.
- Slabo 27px**, 1 Style by John Hudson: Preview text "Grumpy wizards make toxic brew for the evil Queen and Jack." with a 'Normal 400' style.
- Normal 400**: Preview text "Grumpy wizards make toxic brew for the evil Queen and Jack." with a 'Normal 400' style.

At the bottom left, there's a link to 'Collection (0 font families)'. At the bottom right, there are buttons for 'Choose', 'Review', and 'Use'.

Você pode baixar as fontes ou incorporar diretamente no seu código.

Formatação de links

Formatar os links é a mesma coisa de formatar um texto normal, a diferença é que podemos especificar uma formatação para cada estado do link.

No primeiro código de exemplo vamos formatar somente a tag `<a>`, desta forma, todos os estados do link recebem a mesma formatação.

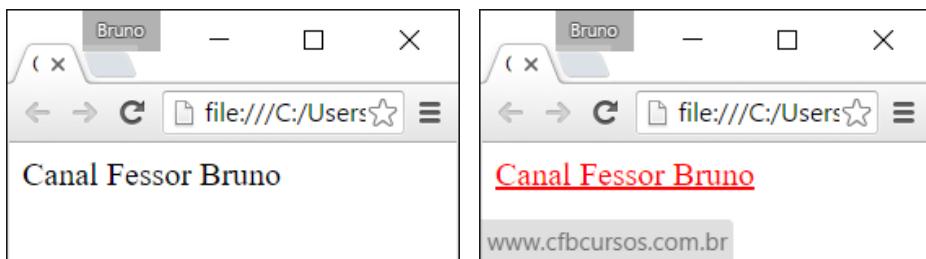
```
<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de CSS</title>
        <meta charset="UTF-8">
        <style rel="stylesheet">
            a{
                text-decoration:none;
                font-size:20px;
                color:#F00;
            }
        </style>
    </head>
    <body>
        <a href="http://www.cfbcursos.com.br" target="_blank">Canal Fessor Bruno</a>
    </body>
</html>
```



Vamos formatar os estados dos links separadamente.

```
a:link, a:visited, a:active{
    text-decoration:none;
    font-size:20px;
    color:#000;
}
a:hover{
    text-decoration:underline;
    font-size:20px;
    color:#F00;
}
```

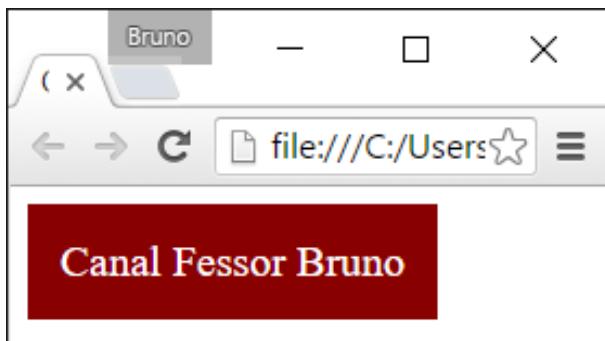
Confira o resultado a seguir, na esquerda o link normal e na direita o link com o mouse sobre.



Formatando o link como um botão

Podemos usar as propriedades CSS para transformar o link na tag normal como se fosse um botão.

```
<style rel="stylesheet">
    a{
        text-decoration:none;
        font-size:20px;
        color:#FFF;
        background-color:#800;
        padding: 15px 15px 15px 15px;
        display:inline-block;
    }
    a:hover{
        background-color:#F00;
    }
</style>
```



Quando o mouse estiver sobre o “botão” este ficará mais claro.

Transformações 2D – CSS3

Existem várias transformações que podemos aplicar aos elementos, translação, rotação, escala, distorção, etc.

Denominamos transformações 2D quanto trabalhamos com os eixos X e Y, quando trabalhamos com o eixo Z denominamos de transformação 3D.

Neste capítulo vamos aprender sobre estas transformações 2D que foram adicionadas em CSS3.

translate()

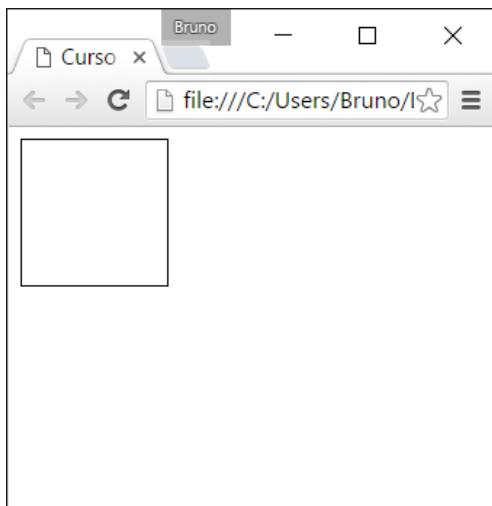
Esta transformação desloca o elemento para o ponto especificado dentro dos parênteses.

`translate(deslocamento X, deslocamento Y);`

Vamos ao nosso código de exemplo.

```
<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de CSS</title>
        <meta charset="UTF-8">
        <style rel="stylesheet">
            #dv1{
                width:100px;
                height:100px;
                border:#000 solid 1px;
            }
        </style>
    </head>
    <body>
        <div id="dv1"></div>
    </body>
</html>
```

Vamos usar este mesmo código para todas as transformações. Inserimos uma div sem especificar a posição, então ela é inserida na posição topo esquerda da tela.



Agora vamos deslocar esta div em 100 pixels para esquerda e 50 pixels para baixo.

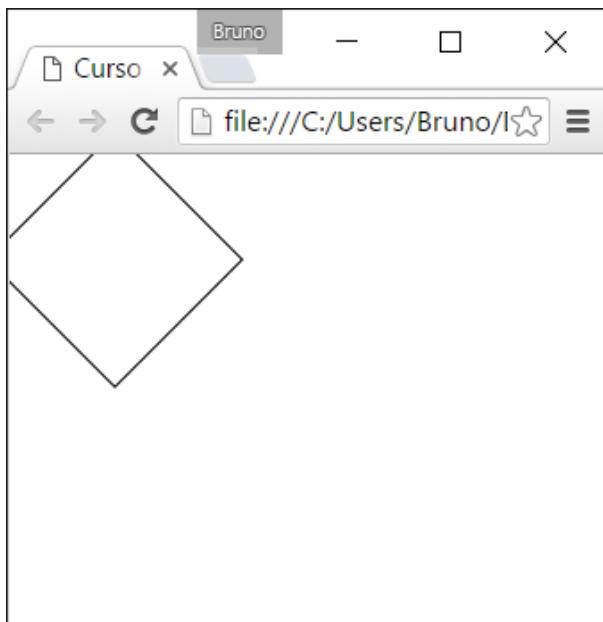
```
<style rel="stylesheet">
    #dv1{
        width:100px;
        height:100px;
        border:#000 solid 1px;
        transform: translate(100px, 50px);
    }
</style>
```



rotate()

Transformação que rotaciona o elemento, basta informar o grau da rotação.

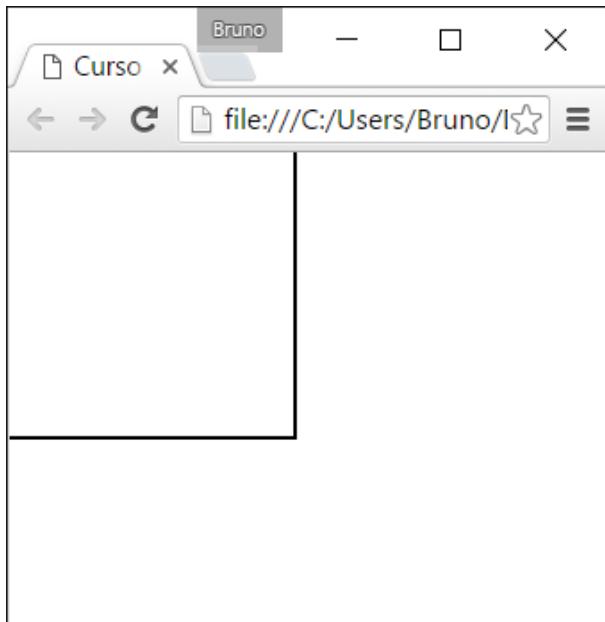
```
<style rel="stylesheet">
    #dv1{
        width:100px;
        height:100px;
        border:#000 solid 1px;
        transform: rotate(45deg);
    }
</style>
```



scale()

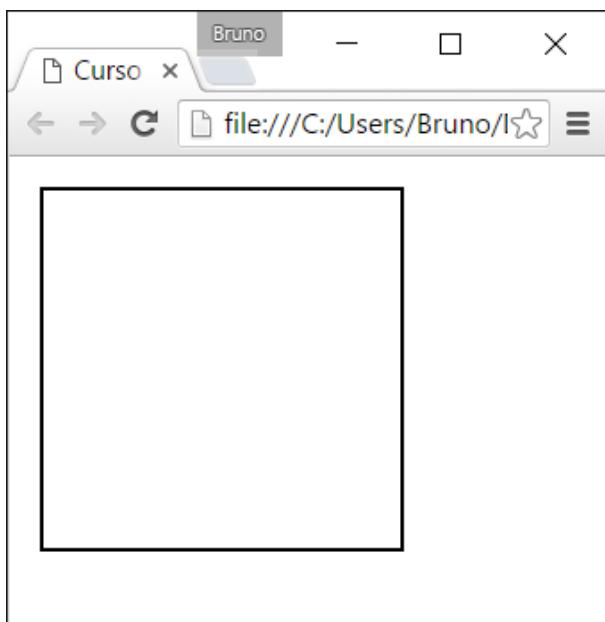
Transformação relativa ao tamanho, aumenta ou diminui.

```
<style rel="stylesheet">
    #dv1{
        width:100px;
        height:100px;
        border:#000 solid 1px;
        transform:scale(2);
    }
</style>
```



Note que a <div> teve seu tamanho dobrado, vamos deslocar para que apareça por inteiro.

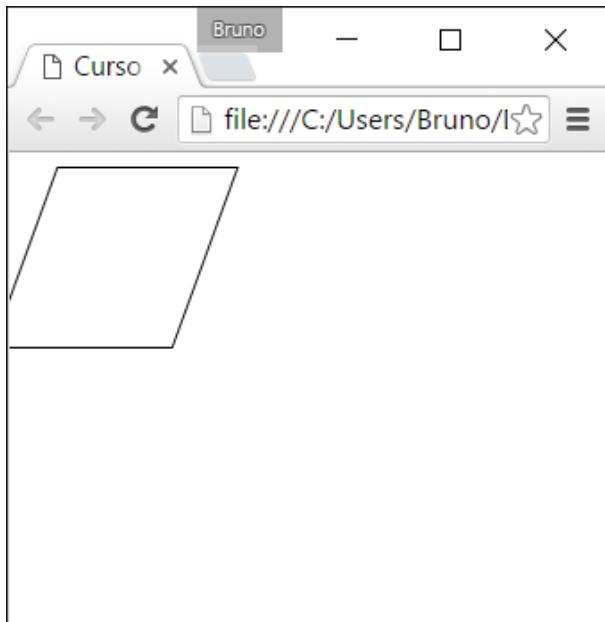
```
<style rel="stylesheet">
    #dv1{
        width:100px;
        height:100px;
        border:#000 solid 1px;
        transform:scale(2) translate(30px, 30px);
    }
</style>
```



skewX()

Permite inclinar o elemento na horizontal.

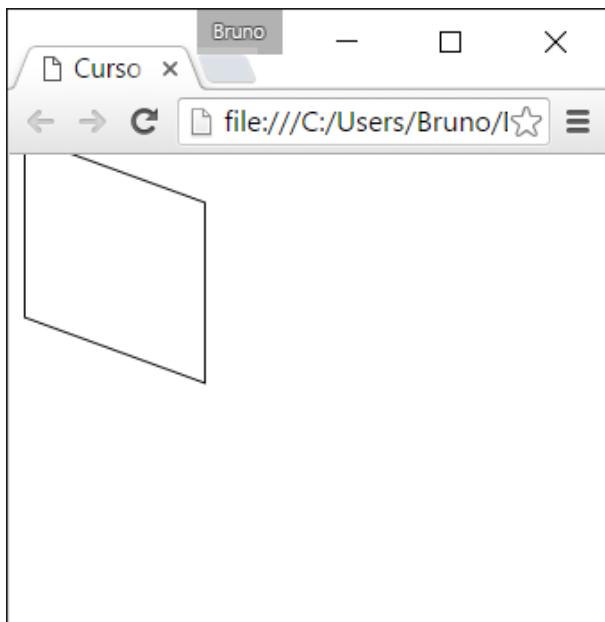
```
<style rel="stylesheet">
    #dv1{
        width:100px;
        height:100px;
        border:#000 solid 1px;
        transform: skewX(-20deg);
    }
</style>
```



skewY()

Permite inclinar o elemento na vertical.

```
<style rel="stylesheet">
    #dv1{
        width:100px;
        height:100px;
        border:#000 solid 1px;
        transform: skewY(-20deg);
    }
</style>
```

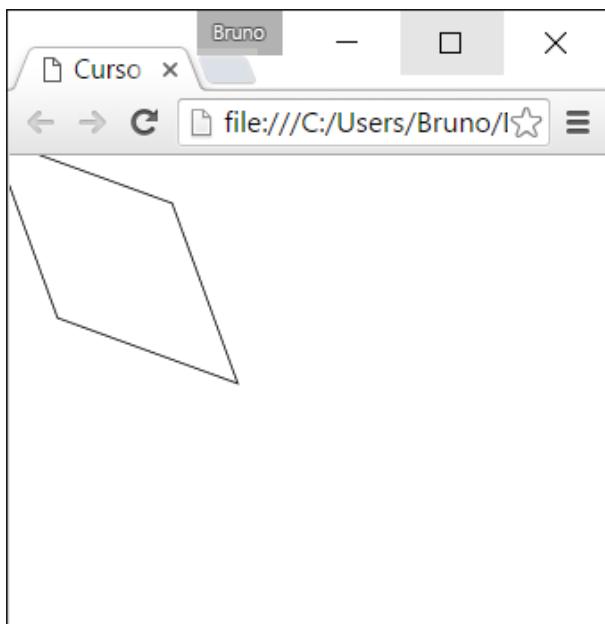


skew()

Permite inclinar o elemento na horizontal e na vertical.

```
<style rel="stylesheet">
    #dv1{
        width:100px;
        height:100px;
        border:#000 solid 1px;
```

```
        transform: skew(20deg, 20deg);  
    }  
</style>
```



matrix()

Combina todas as transformações 2D em um só comando.

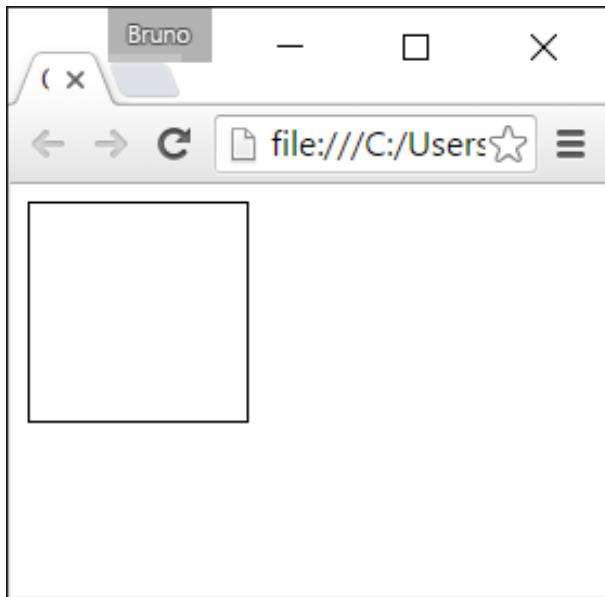
A sintaxe é: `transform: matrix(scaleX(),skewY(),skewX(),scaleY(),translateX(),translateY());`

transform-origin

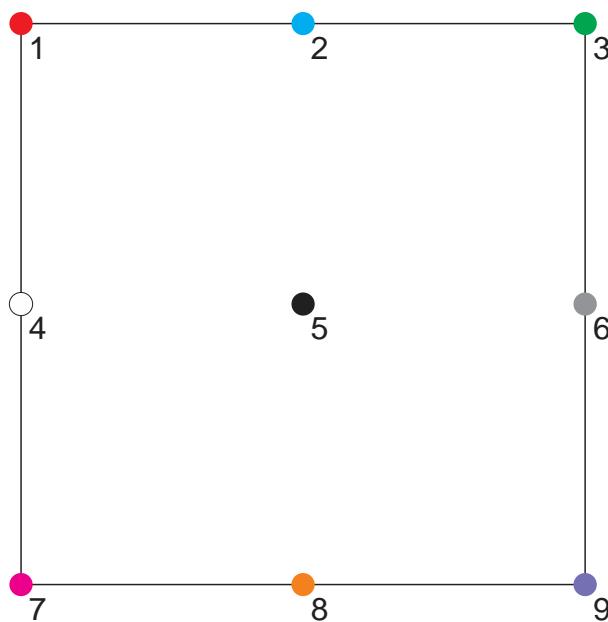
Com esta propriedade podemos especificar o ponto de origem ou de fixação para as transformações, vamos ver o código de exemplo com rotações.

```
<!doctype html>  
<html lang="pt-br">  
    <head>  
        <title>Curso de CSS</title>  
        <meta charset="UTF-8">  
    <style rel="stylesheet">  
        #dv1{  
            width:100px;  
            height:100px;  
            border:#000 solid 1px;  
            transform: rotate(0deg);  
            transform-origin: 0% 0%;  
        }  
    </style>  
    </head>  
    <body>  
        <div id="dv1"></div>  
    </body>  
</html>
```

Neste código inicialmente configuramos rotação 0 graus e origem em 0% em X e 0% em Y.



Antes de alterarmos a rotação, vamos entender um pouco sobre “transform-origin”, a ilustração a seguir tem nove pontos destacados por cor e número, estes pontos estão indicando pontos de origem específicos.



Vamos às configurações dos pontos acima.

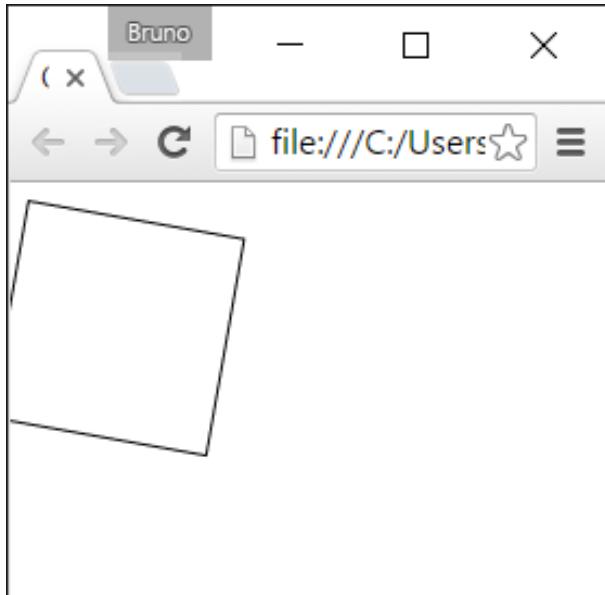
- 1 vermelho = transform-origin(0%, 0%);
- 2 azul = transform-origin(50%, 0%);
- 3 verde = transform-origin(100%, 0%);
- 4 branco = transform-origin(0%, 50%);
- 5 preto = transform-origin(50%, 50%);
- 6 cinza = transform-origin(100%, 50%);
- 7 magenta = transform-origin(0%, 100%);
- 8 laranjado = transform-origin(50%, 100%);
- 9 lilás = transform-origin(100%, 100%);

O ponto de origem padrão para todos os elementos é 50% em X e 50% em Y, ou seja, no centro dos elementos.

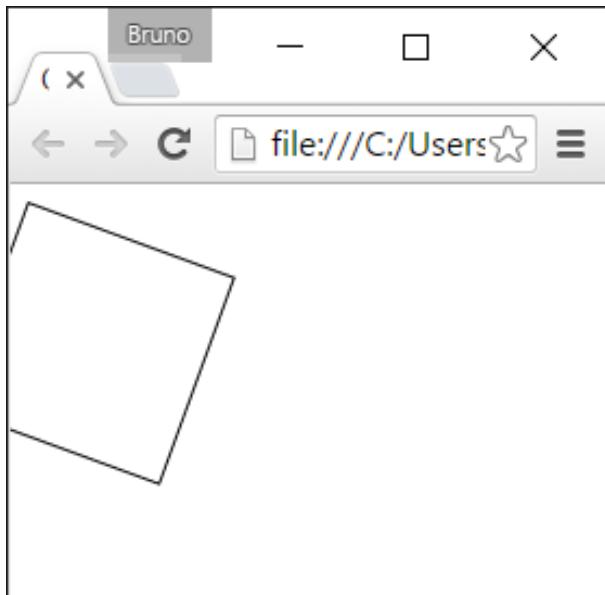
Em nosso código configuramos a origem em 0% e 0%, então de acordo com nossa ilustração, é o ponto 1 vermelho.

Vamos rotacionar a div.

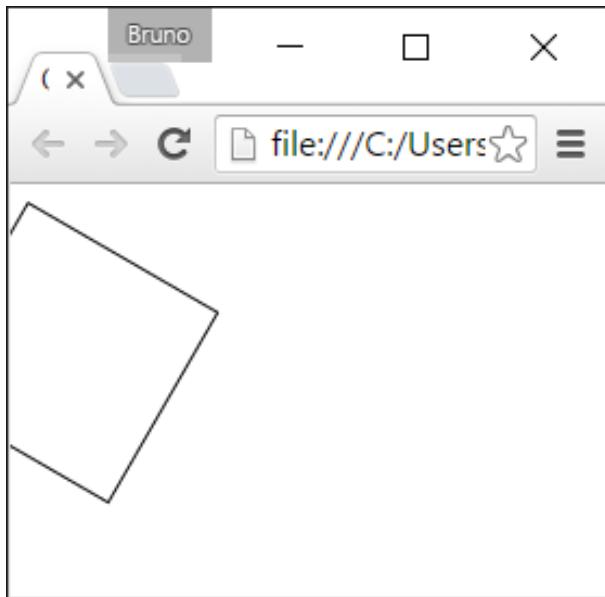
```
<style rel="stylesheet">
    #dv1{
        width:100px;
        height:100px;
        border:#000 solid 1px;
        transform: rotate(10deg);
        transform-origin: 0% 0%;
    }
</style>
```



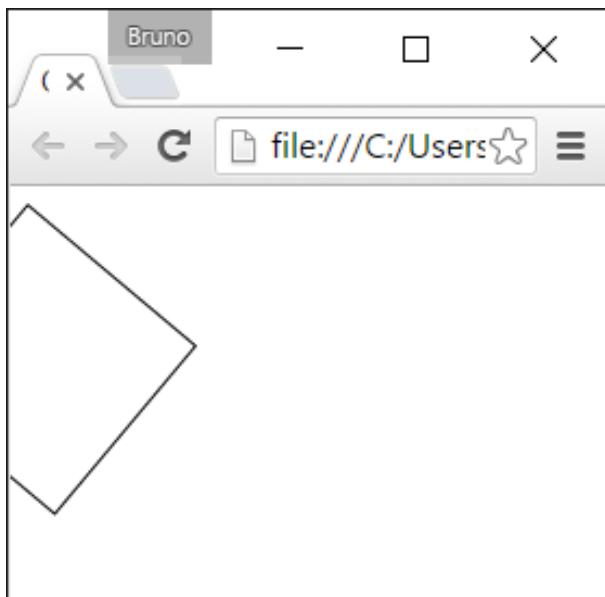
```
transform: rotate(20deg);
```



```
transform: rotate(30deg);
```



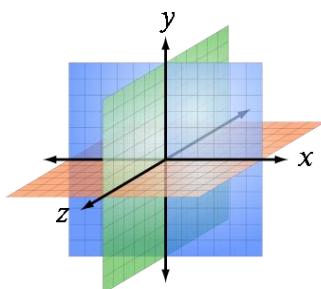
transform: rotate(40deg);



Transformações 3D – CSS3

As propriedades são as mesmas que vimos anteriormente em 2D onde trabalhamos com os eixos X e Y, a diferença é que também podemos usar o eixo Z, por isso especificamos como 3D.

A ilustração a seguir demonstra os eixos X, Y e Z, para que facilite o entendimento.



A forma de uso das funções 3D é exatamente igual às funções 2D, a única diferença como foi dito é o uso de um terceiro valor referente ao eixo Z.

Veja a seguir as funções 3D que podemos utilizar.

```
transform: translate3d(x,y,z);
transform: translateX(x);
transform: translateY(y);
transform: translateZ(z);
transform: scale3d(x,y,z);
transform: scaleX(x);
transform: scaleY(y);
transform: scaleZ(z);
transform: rotate3d(x,y,z,ângulo);
transform: rotateX(x);
transform: rotateY(y);
transform: rotateZ(z);
perspective(n);
```

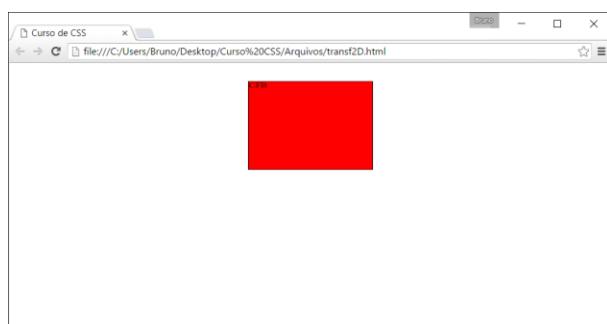
perspective()

Vou falar da função `perspective()` porque diferente das outras seu uso não é tão óbvio, esta configuração não é aplicada no elemento em si e sim no elemento pai, em nosso código de exemplo vamos aplicar `perspective` no `<body>` e adicionar uma `<div>`, em seguida vamos rotacionar esta div e ver a perspectiva funcionando.

```
<!doctype html>
<html lang="pt-br">
<head>
    <title>Curso de CSS</title>
    <meta charset="UTF-8">
    <style rel="stylesheet">
        body {
            perspective:0px;
            width:960px;
            margin:auto;
        }

        #d1 {
            width:200px;
            height:200px;
            margin:auto;
            border: 1px solid black;
            background-color: red;
            transform: rotateX(45deg);
        }
    </style>
</head>
<body>
    <div id="d1">CFB</div>
</body>
</html>
```

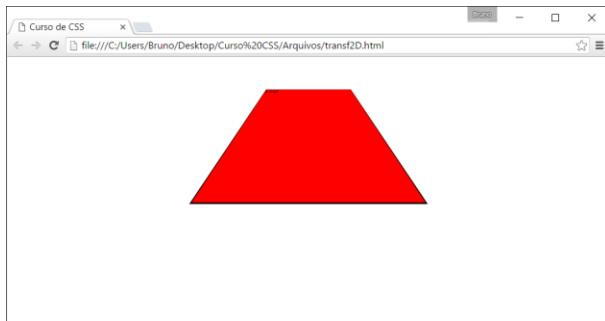
Observe que aplicamos `perspective` no `<body>` mas está configurada como 0 pixels, embora a div esteja rotacionada em X 45 graus, como não existe `perspective` não conseguimos perceber muito bem esta rotação.



Agora vamos aplicar um pouco de perspectiva.

```
body {  
    perspective: 150px;  
    width: 960px;  
    margin: auto;  
}
```

Veja o resultado.



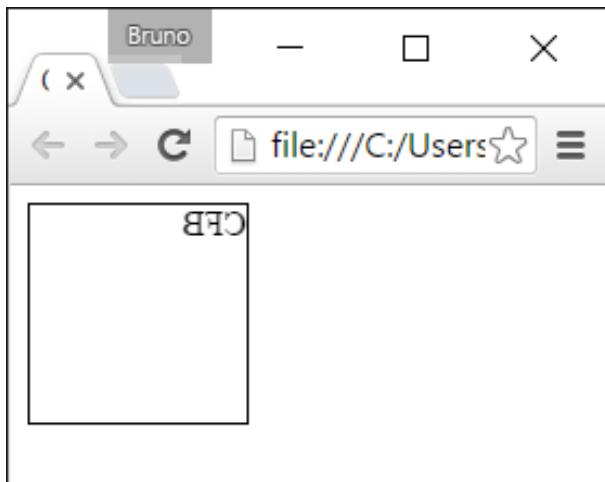
Agora conseguimos notar claramente a perspectiva aplicada.

backface-visibility

Esta propriedade faz com que a parte de trás de um elemento rotacionado mais de 90 graus, seja visível.

Vejamos o código de exemplo.

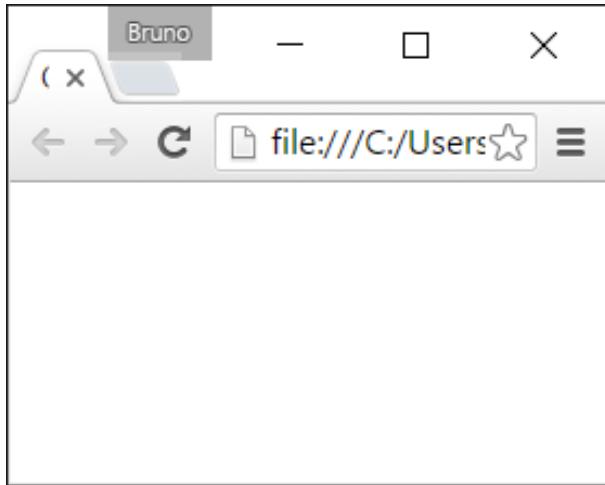
```
<!doctype html>  
<html lang="pt-br">  
    <head>  
        <title>Curso de CSS</title>  
        <meta charset="UTF-8">  
        <style rel="stylesheet">  
            #dv1{  
                width:100px;  
                height:100px;  
                border:#000 solid 1px;  
                transform: rotateY(180deg);  
                backface-visibility:visible;  
            }  
        </style>  
    </head>  
    <body>  
        <div id="dv1">CFB</div>  
    </body>  
</html>
```



Note que o elemento está rotacionado em Y 180 graus, então estamos vendo a parte de trás do elemento. Vamos alterar `backface-visibility` para `hidden`.

```
<style rel="stylesheet">
#dv1{
    width:100px;
    height:100px;
    border:#000 solid 1px;
    transform: rotateY(180deg);
    backface-visibility:hidden;
}
</style>
```

Desta maneira a parte de trás do elemento não será mostrada.



Listas

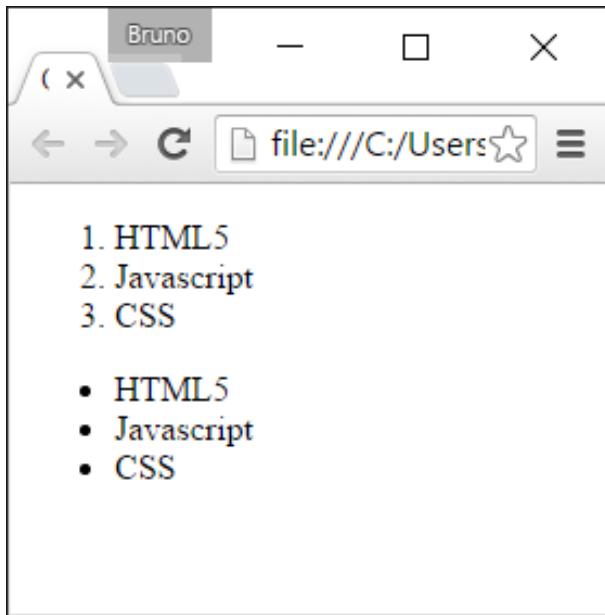
Nesta capítulo vamos aprender como usar os parâmetros CSS para formatação de listas.

Nosso código de exemplo conterá duas listas uma ordenada `` e outra não ordenada ``.

```
<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de CSS</title>
        <meta charset="UTF-8">
        <style rel="stylesheet">

        </style>
    </head>
    <body>
        <ol>
            <li>HTML5</li>
            <li>Javascript</li>
            <li>CSS</li>
        </ol>

        <ul>
            <li>HTML5</li>
            <li>Javascript</li>
            <li>CSS</li>
        </ul>
    </body>
</html>
```

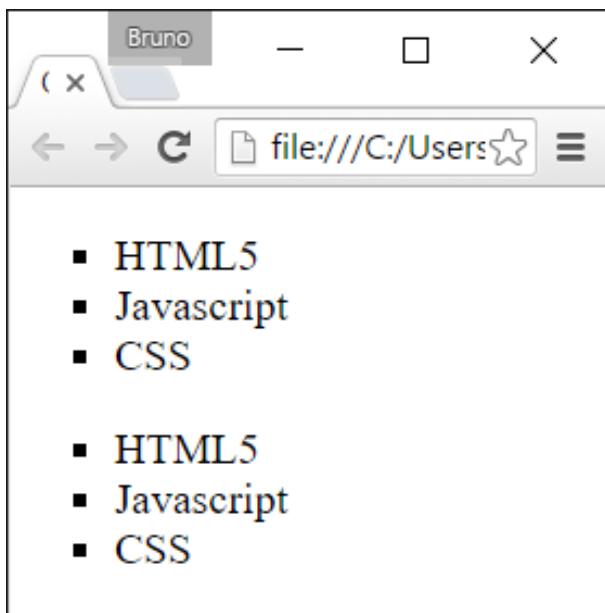


Vamos utilizar estas listas nas propriedades a seguir.

list-style-type

Esta propriedade formata o tipo do marcador da lista, pode ser aplicada em listas ordenadas ou não ordenadas, mas alguns valores são específicos de um tipo ou de outro.

```
<style rel="stylesheet">
    ol,ul{
        list-style-type:square;
        font-size:20px;
    }
</style>
```



Note que mesmo as duas listas sendo diferentes e a propriedade funciona para ambas, veja que mudamos os marcadores para quadrado.

A seguir acompanhe a tabela com os tipos de marcadores possíveis.

Tipo do marcador	Descrição
armenian	Númeral tradicional americano
circle	Círculo não preenchido
cjk-ideographic	Números ideográficos
decimal	Numerais decimais (valor padrão para)
decimal-leading-zero	Numerais decimais com zero na frente (01, 02, 03, ...)
georgian	Numeração georgiana
hebrew	Numeração hebraica
hiragana	Numeração hiragana
hiragana-iroha	Numeração iroha Hiragana
katakana	Numeração katakana
katakana-iroha	Numeração iroha katakana
lower-alpha	Maracor com letras minúsculas (a, b, c, ...)
lower-greek	Marcador grego minúsculo
lower-latin	Marcador com caracteres latinos minúsculos (a, b, c, ...)
lower-roman	Numerais romanos minúsculos (i, ii, iii, iv, v, ...)
none	SEM MARCADORES
square	Quadrado
upper-alpha	Maracor com letras maiúsculas (A, B, C, ...)
upper-latin	Marcador com caracteres latinos maiúsculos (A, B, C, ...)
upper-roman	Numerais romanos maiúsculos (I, II, III, IV, V, ...)
disc	Círculo preenchido (valor padrão para)

list-style-image

Nós podemos definir uma imagem qualquer para ser um marcador, caso queira um marcador personalizado, basta usar a propriedade *list-style-image*.

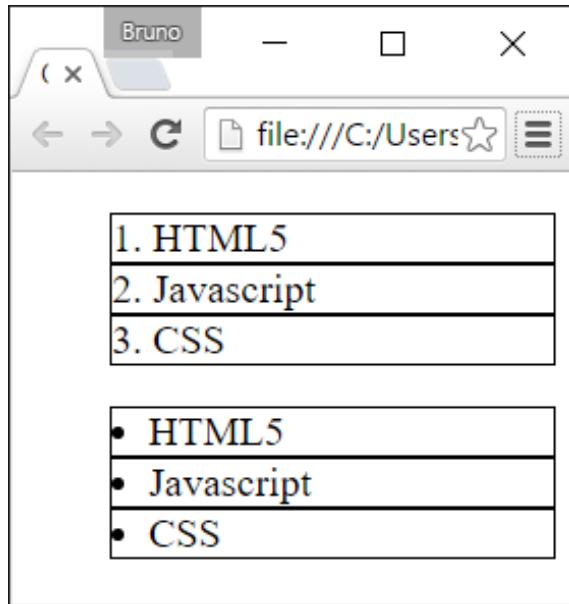
```
<style rel="stylesheet">
    ol,ul{
        list-style-image:url(marcador.png);
        font-size:30px;
    }
</style>
```



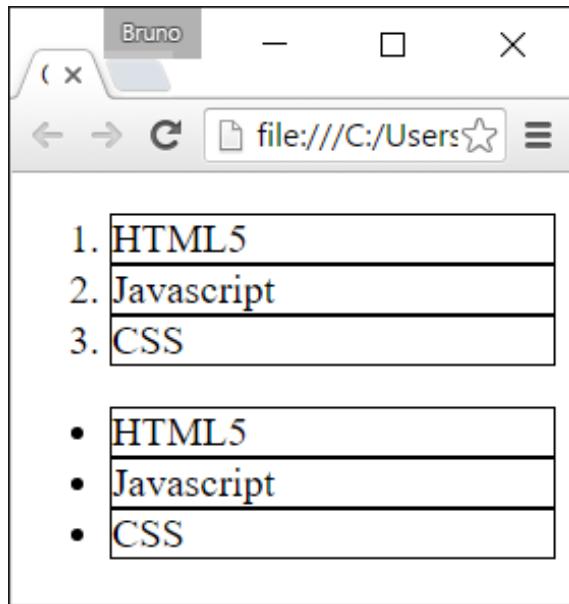
list-style-position

Esta propriedade configura se os marcadores estão no fluxo do texto (inside) ou não (outside), para visualizar melhor esta propriedade configuramos bordas nos elementos , veja os exemplos.

```
<style rel="stylesheet">
    ol,ul{
        list-style-position: inside;
        font-size:20px;
    }
    li{
        border:#000 solid 1px;
    }
</style>
```



```
<style rel="stylesheet">
    ol,ul{
        list-style-position: outside;
        font-size:20px;
    }
    li{
        border:#000 solid 1px;
    }
</style>
```

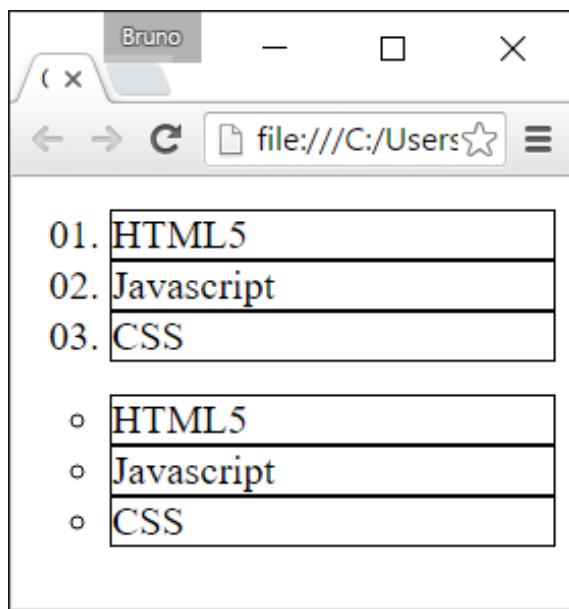


list-style

A metapropriedade `list-style` incorpora todas as propriedades de lista em um só comando, podemos configurar `type`, `position` e `image`.

A sintaxe é: `list-style: estilos posição imagem;`

```
<style rel="stylesheet">
    ol{
        list-style: decimal-leading-zero outside;
    }
    ul{
        list-style: circle outside;
    }
    li{
        border:#000 solid 1px;
        font-size:20px;
    }
</style>
```



Menu horizontal com listas

É muito comum usar o elemento lista para configuração de menu, vou mostrar a seguir um menu construído com listas.

```
<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de CSS</title>
        <meta charset="UTF-8">

    <style rel="stylesheet">
        ol{
            list-style:none;
            padding:0px;
        }
        li{
            width:200px;
            height:40px;
            border-right:#FFF solid 1px;
            font-size:20px;
            display:table-cell;
            vertical-align:middle;
            text-align:center;
            background-color:#800;
        }
    </style>

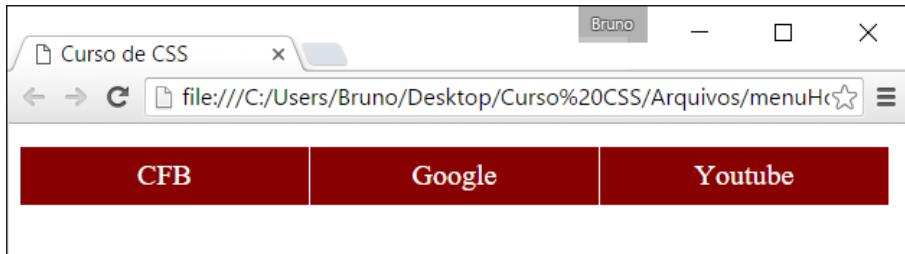
```

```

a{
    text-decoration:none;
    color:#FFF;
}
li:hover{
    background-color:#F00;
}
</style>

</head>
<body>
<ol>
    <li><a href="" target="_black">CFB</a></li>
    <li><a href="" target="_black">Google</a></li>
    <li><a href="" target="_black">Youtube</a></li>
</ol>
</body>
</html>

```



Tabelas

Vamos ver neste capítulo os parâmetros disponíveis para formatação em tabelas, vamos usar o código de exemplo a seguir que monta uma tabela com 3 colunas e 4 linhas.

```

<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de CSS</title>
        <meta charset="UTF-8">

        <style rel="stylesheet">
        </style>
    </head>
    <body>
        <table>
            <tr>
                <td>Peça</td><td>Valor</td><td>Estoque</td>
            </tr>
            <tr>
                <td>Processador</td><td>R$800,00</td><td>30</td>
            </tr>
            <tr>
                <td>Memória</td><td>R$150,00</td><td>55</td>
            </tr>
            <tr>
                <td>HD</td><td>R$320,00</td><td>20</td>
            </tr>
        </table>
    </body>
</html>

```

Peça	Valor	Estoque
Processador	R\$800,00	30
Memória	R\$150,00	55
HD	R\$320,00	20

Formatações comuns

```
<style rel="stylesheet">
    table, tr, td, th{
        border:#000 solid 1px;
        width:450px;
        padding:0px;
        font-size:20px;
    }
    td{
        width:150px;
        height:30px;
    }
    th{
        background-color:#060;
        color:#FFF;
    }
</style>
```

Peça	Valor	Estoque
Processador	R\$800,00	30
Memória	R\$150,00	55
HD	R\$320,00	20

vertical-align

Esta propriedade formata o alinhamento vertical do conteúdo da célula.

vertical-align: top; → Alinha o texto na parte de cima da célula

vertical-align: middle; → Alinha o texto no meio vertical da célula (valor padrão)

vertical-align: bottom; → Alinha o texto na parte de baixo da célula

border-collapse

Formata o distanciamento entre as células da tabela, os valores possíveis para a propriedade são separate (células separadas) e collapse (células sem separação).

```
<style rel="stylesheet">
    table, tr, td, th{
        border:#000 solid 1px;
        width:450px;
        padding:0px;
        font-size:20px;
        border-collapse:collapse;
    }
</style>
```

```

}
td{
    width:150px;
    height:30px;
}
th{
    background-color:#060;
    color:#FFF;
}
</style>

```

Peça	Valor	Estoque
Processador	R\$800,00	30
Memória	R\$150,00	55
HD	R\$320,00	20

border-spacing

Formata o espaçamento entre as células.

```

table {
    border-collapse: separate;
    border-spacing: 10px 50px;
}

```

empty-cells

Esta propriedade permite seconder as células que estão vazias, caso queira mostrar basta usar o valor show.

```

table {
    border-collapse: separate;
    empty-cells: hide;
}

```

table-layout

Permite configurar o layout em fixo, ou seja, não redimensiona a célula de acordo com o conteúdo.

```

table {
    table-layout: fixed;
}

```

nth-child() – CSS3

Especifica o a cor de fundo de um determinado elemento ou de uma sequencia de elementos em estilo “zebra”. Veja a aplicação em nossa tabela, anterior.

```

<style rel="stylesheet">
    table, tr, td, th{
        border:#000 solid 1px;
        width:450px;
        padding:0px;
        font-size:20px;
        border-collapse:collapse;
    }
    td{
        width:150px;
        height:30px;
    }
    tr:nth-child(even) {
        background: #DDDDDD;
    }
</style>

```

```
th{
    background-color:#060;
    color:#FFF;
}
</style>
```

Neste caso o parâmetro preenche a cor de fundo dos elementos <tr> de forma “zebra”, ou seja, preenche 1 e pula outro, até o final de todos os elementos.

Peça	Valor	Estoque
Processador	R\$800,00	30
Memória	R\$150,00	55
HD	R\$320,00	20

Vamos usar com elementos <p>.

```
<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de CSS</title>
        <meta charset="UTF-8">

        <style rel="stylesheet">
            p:nth-child(even) {
                background: #DDDDDD;
            }
        </style>
    </head>
    <body>
        <p>Canal Fessor Bruno</p>
        <p>Canal Fessor Bruno</p>
    </body>
</html>
```

Canal Fessor Bruno

outline

A propriedade `outline` controla uma linha presente após a borda do elemento, é importante ressaltar que `outline` não atua como margem, a ilustração a seguir mostra a linha externa `outline`.

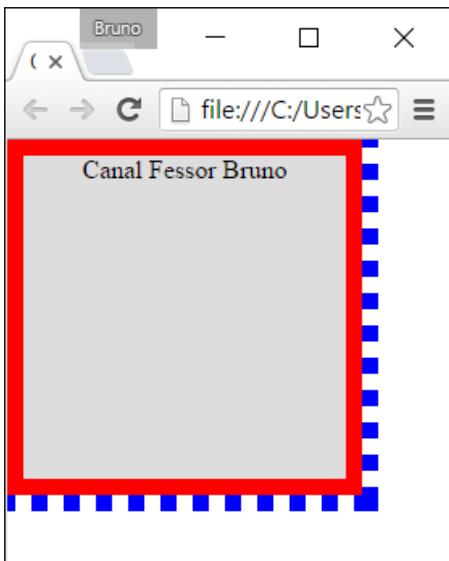


Podemos usar a metapropriedade `outline` onde configuramos de uma só vez cor, tipo de linha e espessura, ou podemos usar as propriedades separadas.

```
<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de CSS</title>
        <meta charset="UTF-8">

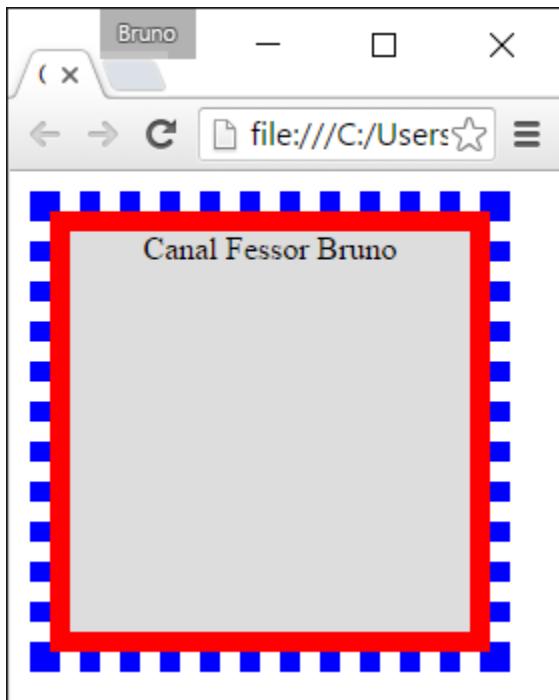
        <style rel="stylesheet">
            body{
                margin:0px;
                padding:0px;
            }
            div{
                width:200px;
                height:200px;
                text-align:center;
                border:#F00 solid 10px;
                outline:#00F dotted 10px;
                background:#DDDDDD;
            }
        </style>
    </head>
    <body>
        <div>Canal Fessor Bruno</div>
    </body>
</html>
```

Em nosso código de exemplo configuramos a linha externa com cor azul, pontilhada e largura de 5 pixels.



Note que outline por padrão não ocupa espaço na página, pois as partes esquerda e superior estão para fora da tela, vamos então adicionar uma margem ao elemento.

```
div{  
    width:200px;  
    height:200px;  
    text-align:center;  
    border:#F00 solid 10px;  
    outline:#00F dotted 10px;  
    background:#DDDDDD;  
    margin:20px;  
}
```



Da mesma forma que várias propriedades anteriores, podemos controlar as propriedades outline separadamente.

outline-color

Cor da linha externa.

```
p{  
    outline-color:#00F;  
}
```

outline-offset

Distância da linha externa para a borda, quanto maior o valor mais distante a linha estará.

```
p{  
    outline-offset:#15px;  
}
```

outline-style

Estilo da linha externa, usamos os mesmos estilos vistos no parâmetro border

```
p{  
    outline-style:solid;  
}
```

outline-width

Espessura da linha externa.

```
p{  
    outline-width:20px;  
}
```

display

Basicamente esta propriedade indica como um elemento será exibido, é muito importante entender o uso da propriedade display para construção dos nossos layouts, já até usamos algumas vezes anteriormente, mas neste capítulo vou mostrar como usar display.

Todos os elementos tem um valor padrão para sua propriedade display, a maioria dos elementos tem seu display configurado em “block” ou “inline-block”.

block

Um elemento que tem o display configurado em block, sempre inicia em uma nova linha, não aceita elementos na mesma linha que ele, ou seja, quebra a linha após o elemento, e sua área acupa toda a linha onde ele é inserido.

Podemos citar como exemplo alguns elementos que tem display:block configurados por padrão: <div>, <h1> até <h6>, <p>, <form>, <header>, <footer>, <section>, <table>.

inline

Os elementos configurados com display:inline não iniciam em uma nova linha nem quebram de linha após o elemento, outro detalhe é que eles não ocupam a linha inteira, somente ocupam o espaço de seu conteúdo.

Podemos citar alguns exemplos como: , <a>, .

none

Este valor para propriedade display pode ser usada para ocultar o elemento pelo javascript.

inline-block

Parecido com inline, só que preserva o tamanho configurado do elemento.

inline-table

Elemento é exibido como uma tabela, não iniciam em uma nova linha nem quebram de linha após o elemento, outro detalhe é que eles não ocupam a linha inteira, somente ocupam o espaço de seu conteúdo, podendo inclusive usar as propriedades de formatação do elemento <table>.

list-item

Elemento se comporta como um inclusive passar a mostrar o marcador.

run-in

Exibe um elemento como block ou inline, dependendo do contexto onde está inserido.

table

O element se comporta exatamente igual a uma tabela, podendo inclusive usar as propriedades de formatação do elemento <table>.

table-caption

O elemento se comporta como se fosse um caption de uma tabela <th>.

table-header-group

O elemento se comporta como se fosse um <thead> da tabela.

table-footer-group

O elemento se comporta como se fosse um <tfoot> da tabela.

table-row-group

O elemento se comporta como se fosse um <tbody> da tabela.

table-cell

O elemento se comporta como se fosse uma célula de uma tabela, podendo receber todas as propriedades que usamos para <td>.

table-column

Elemento se comporta como o elemento <col>.

table-row

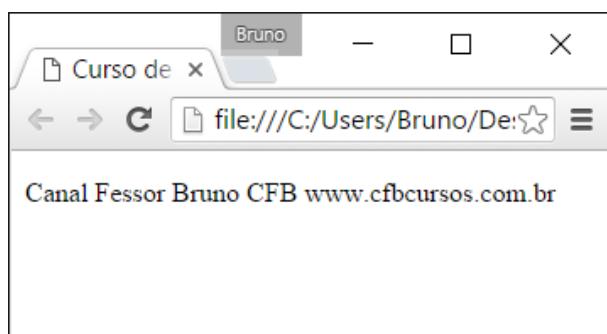
O elemento se comporta como se fosse uma linha de uma tabela, podendo receber todas as propriedades que usamos para <tr>.

Vejamos agora alguns exemplos.

Por padrão a tag tem seu display configurado como inline, veja o código a seguir.

```
<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de CSS</title>
        <meta charset="UTF-8">

        <style rel="stylesheet">
        </style>
    </head>
    <body>
        <p>Canal Fessor Bruno <span>CFB</span> www.cfbcursos.com.br</p>
    </body>
</html>
```



Vamos alterar o display da tag para block e ver o resultado.

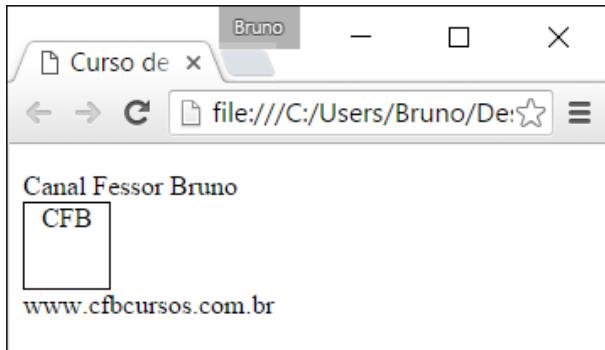
```
<style rel="stylesheet">
    span{
        display:block;
    }
</style>
```



Note que agora foram inseridas quebra de linha antes e depois do .

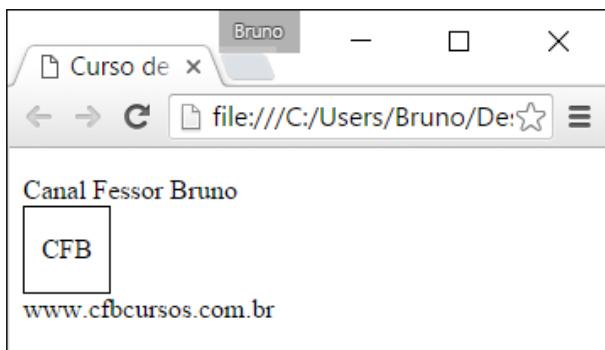
Vamos configurar mais alguns detalhes do , acompanhe.

```
<style rel="stylesheet">
    span{
        display:block;
        width:50px;
        height:50px;
        border:#000 solid 1px;
        text-align:center;
        vertical-align:middle;
    }
</style>
```



Se você prestar atenção, verá que a propriedade “vertical-align:middle” não está funcionando, isso porque esta é uma propriedade para tabelas, então, para funcionar, vamos mudar o display do nosso elemento para table-cell.

```
<style rel="stylesheet">
    span{
        display:table-cell;
        width:50px;
        height:50px;
        border:#000 solid 1px;
        text-align:center;
        vertical-align:middle;
    }
</style>
```

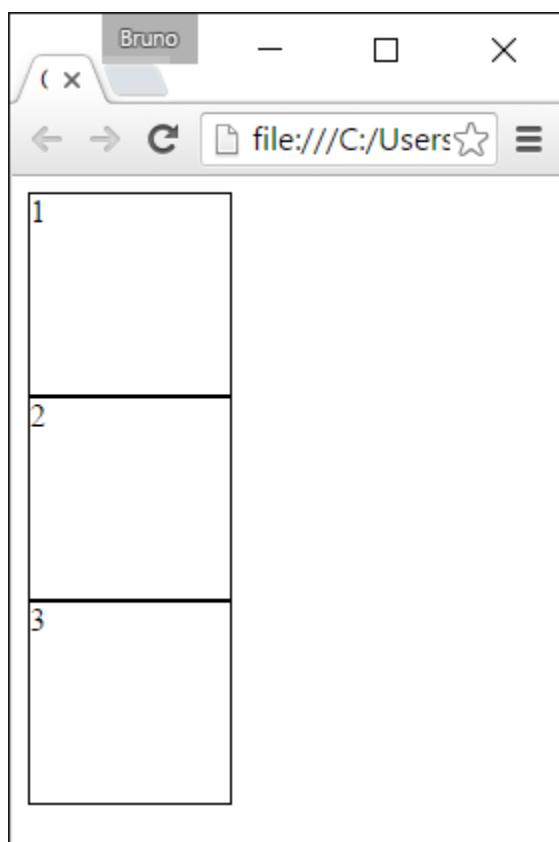


Vamos ver outro exemplo usando divs.

```
<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de CSS</title>
        <meta charset="UTF-8">

        <style rel="stylesheet">
            div{
                width:100px;
                height:100px;
                border:#000 solid 1px;
            }
        </style>

    </head>
    <body>
        <div id="d1">1</div>
        <div id="d2">2</div>
        <div id="d3">3</div>
    </body>
</html>
```



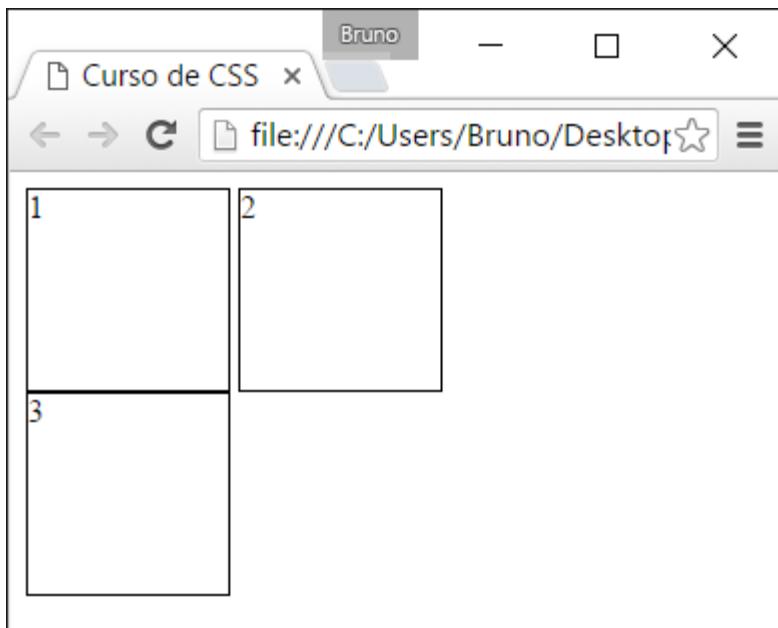
Note que o display padrão do elemento `<div>` é “block”, mas se precisarmos que todas estas divs sejam posicionadas uma ao lado da outra? Simple, basta mudar o display para “`inline-block`”.

```
<style rel="stylesheet">
    div{
        width:100px;
        height:100px;
        border:#000 solid 1px;
        display:inline-block;
    }
</style>
```



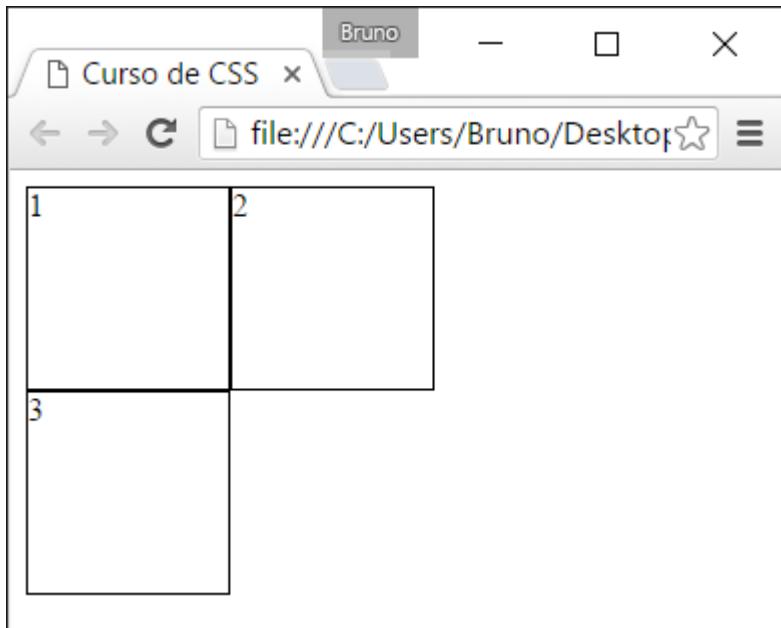
Ótimo, mas se só precisar que as duas primeiras fiquem uma ao lado da outra e a terceira fique normal? Nesta caso vamos configurar as divs pelos ids.

```
<style rel="stylesheet">
    div{
        width:100px;
        height:100px;
        border:#000 solid 1px;
    }
    #d1, #d2{
        display:inline-block;
    }
</style>
```



Um detalhe importante, estamos vendo um espaço entre as divs que estão na mesma linha, como podemos remover este espaço? É mais simples do que você imagina, basta posicionar, no código, uma div ao lado da outra, veja.

```
<body>
    <div id="d1">1</div><div id="d2">2</div>
    <div id="d3">3</div>
</body>
```



position

A propriedade position especifica como um elemento será posicionado na tela, podemos até posicionar em um ponto específico controlando os parâmetros left e top.

Muitos programadores associam a propriedade position somente ao elemento <div> mas a verdade é que podemos utilizar em muitos outros elementos como e <p>.

São quatro tipos de posicionamento disponíveis, static, relative, fixed e absolute, vamos entender.

A única configuração que não permite escolher um posicionamento para o elemento é static, os demais podemos usar as propriedades top, left, right ou bottom.

top = Desloca o elemento na vertical (Y), o valor é a distância do elemento com o topo, ou seja, 0 pixels o elemento fica totalmente acima. Descola no sentido de cima para baixo.

left = Desloca o elemento na horizontal (X), o valor é a distância do elemento com a borda esquerda, ou seja, 0 pixels o elemento fica totalmente a esquerda. Descola no sentido esquerda para direita.

right = Desloca o elemento na horizontal (X), o valor é a distância do elemento com a borda direita, ou seja, 0 pixels o elemento fica totalmente a direita. Descola no sentido direita para esquerda.

bottom = Desloca o elemento na vertical (Y), o valor é a distância do elemento com a borda inferior, ou seja, 0 pixels o elemento fica totalmente abaixo. Descola no sentido de baixo para cima.

Em nossos exemplos iremos usar divs simplesmente para facilitar o entendimento.

static

Este é o valor padrão de todos os elementos HTML, neste posicionamento os elementos não podem ser controlados por top e left e não tem seu posicionamento afetado pelo posicionamento de outros elementos.

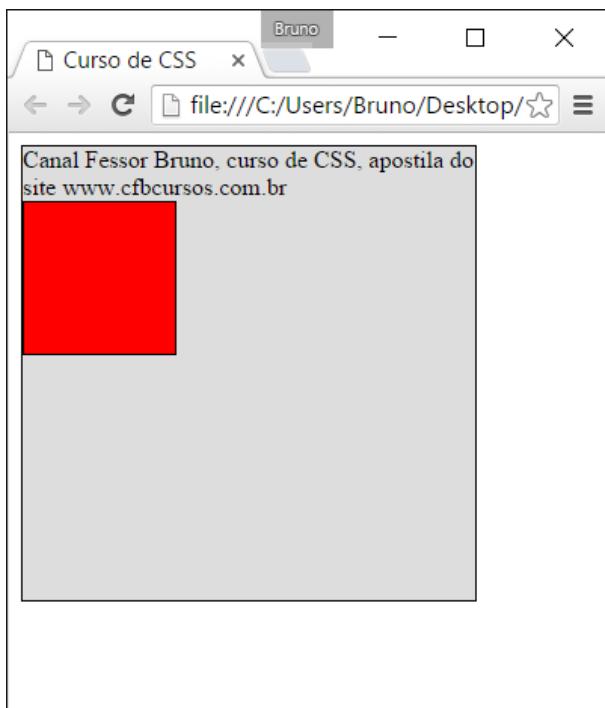
A seguir acompanhe nosso código de exemplo para trabalhar com position.

```
<!doctype html>
<html lang="pt-br">
  <head>
    <title>Curso de CSS</title>
```

```
<meta charset="UTF-8">

<style rel="stylesheet">
    div{
        width:100px;
        height:100px;
        border:#000 solid 1px;
    }
    #dvPai{
        width:300px;
        height:300px;
        background-color:#DDD;
    }
    #d1{
        position:static;
        background-color:#F00;
    }
</style>

</head>
<body>
    <div id="dvPai">
        Canal Fessor Bruno, curso de CSS, apostila do site www.cfbcursos.com.br
        <div id="d1"></div>
    </div>
</body>
</html>
```

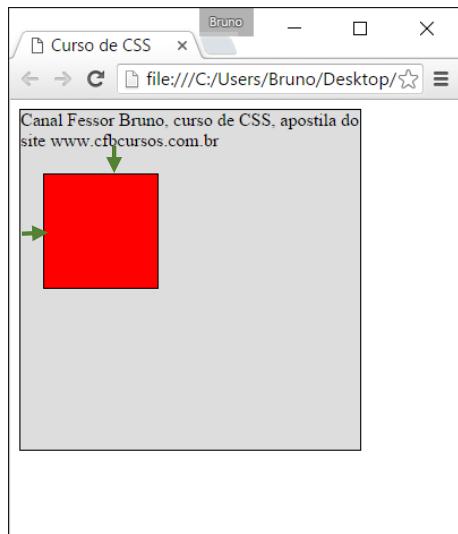


relative

Um elemento com `position:relative` tem seu posicionamento `left` e `top` relacionado com o elemento anterior, veja o código.

```
<style rel="stylesheet">
    div{
        width:100px;
        height:100px;
        border:#000 solid 1px;
    }
    #dvPai{
        width:300px;
        height:300px;
        background-color:#DDD;
    }
    #d1{
        position:relative;
```

```
    top:20px;
    left:20px;
    background-color:#F00;
}
</style>
```

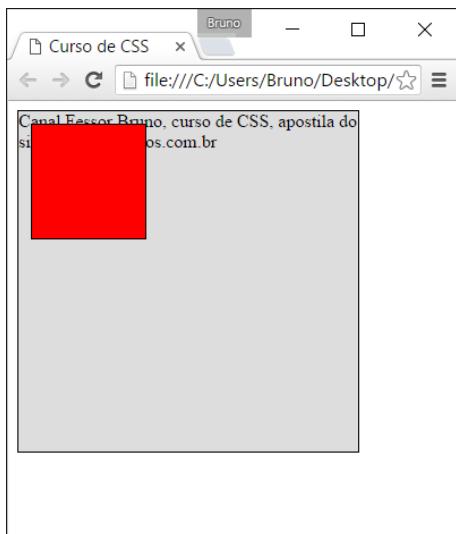


Note pelas setas verdes ilustrativas, que d1 foi posicionado em left:20px e top:20px em relação ao texto anterior e não ao início do elemento pai dvPai.

absolute

Um elemento com position:absolute tem seu posicionamento left e top relacionado com o elemento pai e não com o elemento anterior, desta maneira elementos anteriores não irão afetar seu posicionamento.

```
<style rel="stylesheet">
    div{
        width:100px;
        height:100px;
        border:#000 solid 1px;
    }
    #dvPai{
        width:300px;
        height:300px;
        background-color:#DDD;
    }
    #d1{
        position:absolute;
        top:20px;
        left:20px;
        background-color:#F00;
    }
</style>
```

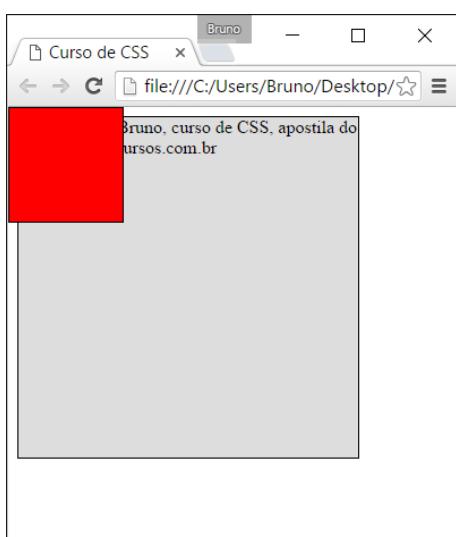


Veja que agora o elemento d1 tem seu posicionamento em relação ao elemento pai dvPai e não ao(s) elemento(s) que estão antes dele.

fixed

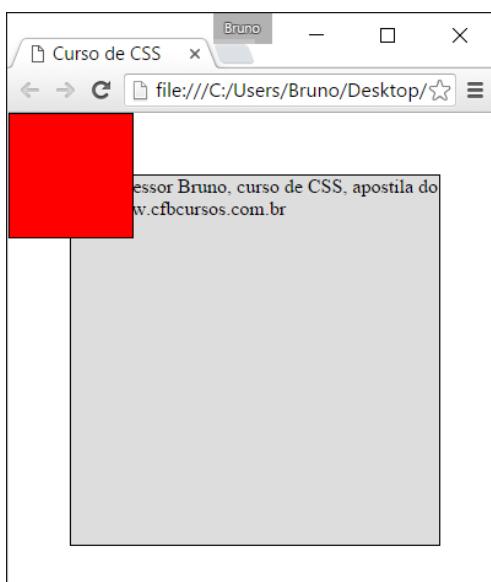
Neste posicionamento o elemento é posicionado em relação à janela, independente se esteja dentro de outro ou não, outro detalhe importante sobre fixed é que o elemento é independente da rolagem do conteúdo da tela, a tela pode rolar que o elemento que estiver configurado como fixed não rola junto com a tela.

```
<style rel="stylesheet">
    div{
        width:100px;
        height:100px;
        border:#000 solid 1px;
    }
    #dvPai{
        width:300px;
        height:300px;
        background-color:#DDD;
    }
    #d1{
        position:fixed;
        top:0px;
        left:0px;
        background-color:#F00;
    }
</style>
```



Mesmo que o elemento pai dvPai seja movido d1 não será afetado por ele.

```
<style rel="stylesheet">
    div{
        width:100px;
        height:100px;
        border:#000 solid 1px;
    }
    #dvPai{
        position:absolute;
        top:50px;
        left:50px;
        width:300px;
        height:300px;
        background-color:#DDD;
    }
    #d1{
        position:fixed;
        top:0px;
        left:0px;
        background-color:#F00;
    }
</style>
```



Agora, vamos adicionar um elemento `<p>` após a `div` e formata-lo com uma margem superior enorme para que fique bem embaixo da página.

```
<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de CSS</title>
        <meta charset="UTF-8">

        <style rel="stylesheet">
            div{
                width:100px;
                height:100px;
                border:#000 solid 1px;
            }
            #dvPai{
                position:absolute;
                top:50px;
                left:50px;
                width:300px;
                height:300px;
                background-color:#DDD;
            }
            #d1{
                position:fixed;
                top:0px;
                left:0px;
                background-color:#F00;
            }
            p{
                margin-top: 100px;
            }
        </style>
    </head>
    <body>
        <div></div>
        <div id="dvPai"></div>
        <div id="d1"></div>
        <p>Professor Bruno, curso de CSS, apostila do w.cfbcursos.com.br</p>
    </body>
</html>
```

```

        margin-top:2000px;
    }

```

```

</style>

</head>
<body>
    <div id="dvPai">
        Canal Fessor Bruno, curso de CSS, apostila do site www.cfbcursos.com.br
        <div id="d1"></div>
    </div>
    <p>CFB</p>
</body>
</html>

```



Agora, note que temos uma barra de rolagem, role a tela e veja que a div não vai sair do lugar.

z-index

A propriedade z-index controla a ordem de empilhamento dos elementos, o padrão é a ordem de inserção, ou seja, um elemento posicionado no código primeiro mas podemos mudar, vamos ao exemplo prático.

```

<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de CSS</title>
        <meta charset="UTF-8">

        <style rel="stylesheet">
            div{
                width:100px;
                height:100px;
                border:#000 solid 1px;
                position:fixed;
            }

            #d1{
                top:20px;
                left:20px;
                background-color:#F00;
            }
            #d2{
                top:40px;
                left:40px;
                background-color:#0F0;
            }
            #d3{
                top:60px;
                left:60px;
                background-color:#00F;
            }
            #d4{
                top:80px;
            }
        </style>
    </head>
    <body>
        <div id="d1"></div>
        <div id="d2"></div>
        <div id="d3"></div>
        <div id="d4"></div>
    </body>
</html>

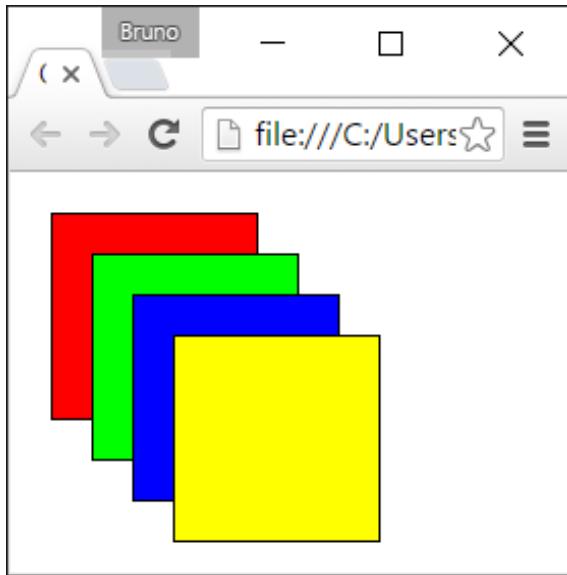
```

```
        left:80px;
        background-color:#FF0;
    }

```

```
</style>

</head>
<body>
    <div id="d1"></div>
    <div id="d2"></div>
    <div id="d3"></div>
    <div id="d4"></div>
</body>
</html>
```



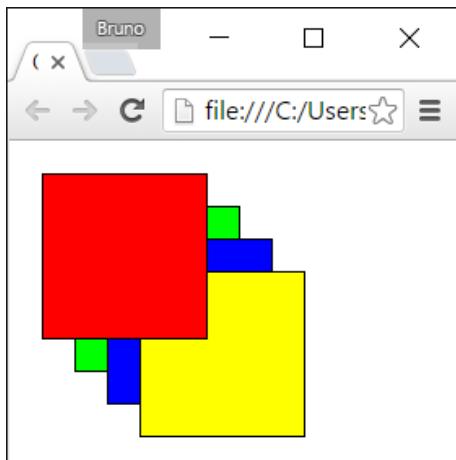
Note que a ordem de empilhamento é a normal, o primeiro elemento inserido ficou abaixo de todos e último elemento ficou acima de todos.

Vamos adicionar o parâmetro z-index em todos os elementos e configurar o empilhamento.

```
<style rel="stylesheet">
    div{
        width:100px;
        height:100px;
        border:#000 solid 1px;
        position:fixed;
    }
    #d1{
        top:20px;
        left:20px;
        background-color:#F00;
        z-index:0;
    }
    #d2{
        top:40px;
        left:40px;
        background-color:#0F0;
        z-index:1;
    }
    #d3{
        top:60px;
        left:60px;
        background-color:#00F;
        z-index:2;
    }
    #d4{
        top:80px;
        left:80px;
        background-color:#FF0;
        z-index:3;
    }
</style>
```

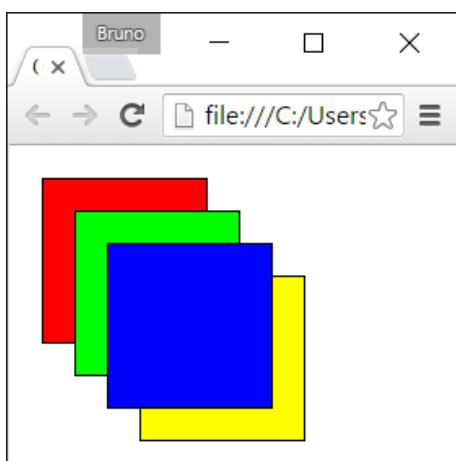
Agora vamos posicionar o primeiro elemento d1 acima de todos os outros, basta usar um z-index maior que o último.

```
#d1{  
    top:20px;  
    left:20px;  
    background-color:#F00;  
    z-index:4;  
}
```



Vamos voltar d1 para seu empilhamento padrão e posicionar d4 em baixo de todos os outros.

```
#d1{  
    top:20px;  
    left:20px;  
    background-color:#F00;  
    z-index:1;  
}  
#d2{  
    top:40px;  
    left:40px;  
    background-color:#0F0;  
    z-index:2;  
}  
#d3{  
    top:60px;  
    left:60px;  
    background-color:#00F;  
    z-index:3;  
}  
#d4{  
    top:80px;  
    left:80px;  
    background-color:#FF0;  
    z-index:0;  
}
```



float

Esta propriedade especifica se o elemento vai flutuar ou não, vamos entender na prática. Em nosso código de exemplo, inserimos um element `` e um `<p>`.

```
<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de CSS</title>
        <meta charset="UTF-8">

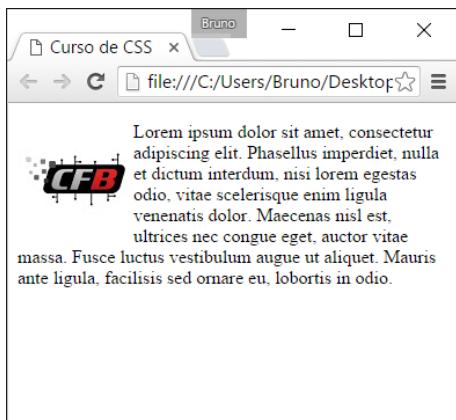
        <style rel="stylesheet">
        </style>
    </head>
    <body>
        
        <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum, nisi lorem egestas odio, vitae scelerisque enim ligula venenatis dolor. Maecenas nisl est, ultrices nec congue eget, auctor vitae massa. Fusce luctus vestibulum augue ut aliquet. Mauris ante ligula, facilisis sed ornare eu, lobortis in odio.</p>
    </body>
</html>
```

Na ilustração podemos conferir como é o comportamento normal deste código.

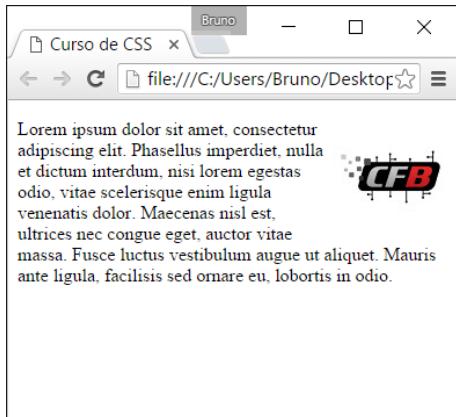


Vamos usar a propriedade float para posicionar a imagem da esquerda e na direita do texto.

```
<style rel="stylesheet">
    img{
        float:left;
    }
</style>
```



```
<style rel="stylesheet">
    img{
        float:right;
    }
</style>
```



Vamos voltar a imagem para esquerda e mudar um pouco o texto.

```
<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de CSS</title>
        <meta charset="UTF-8">

        <style rel="stylesheet">
            img{
                float:left;
                width:150px;
                border:#000 solid 1px;
                margin:0px 10px 10px 0px;
            }
        </style>
    </head>
    <body>
        
        <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.</p>
        <p>Phasellus imperdiet, nulla et dictum interdum, nisi lorem egestas odio.</p>
        <p>Maecenas nisl est, ultrices nec congue eget, auctor vitae massa.</p>
        <p>Mauris ante ligula, facilisis sed ornare eu, lobortis in odio.</p>
    </body>
</html>
```



E se precisarmos cancelar o efeito do float em um certo ponto, por exemplo, no terceiro parágrafo, vamos deixar o float:left atuar somente nos dois primeiros parágrafos, para isso, precisamos usar a propriedade clear.

clear

A propriedade clear é usada para cancelar o comportamento da propriedade float. Em nosso primeiro exemplo vamos utilizar a propriedade clear dentro de uma classe, vamos ao código.

```
<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de CSS</title>
        <meta charset="UTF-8">

        <style rel="stylesheet">
            .fix{
                clear:left;
            }

            img{
                float:left;
                width:150px;
                border:#000 solid 1px;
                margin:0px 10px 10px 0px;
            }
        </style>
    </head>
    <body>
        
        <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.</p>
        <p>Phasellus imperdiet, nulla et dictum interdum, nisi lorem egestas odio.</p>
        <p class="fix">Maecenas nisl est, ultrices nec congue eget, auctor vitae massa.</p>
        <p>Mauris ante ligula, facilisis sed ornare eu, lobortis in odio.</p>
    </body>
</html>
```



Veja que agora o terceiro parágrafo não está mais influenciado pelo float.

Vamos a mais algumas mudanças em nosso código, iremos retirar a classe fix do terceiro parágrafo, adicionar mais uma imagem e preparar a formatação de forma que uma imagem flutue à esquerda e outra à direita.

```
<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de CSS</title>
        <meta charset="UTF-8">

        <style rel="stylesheet">
            .fix{
                clear:left;
            }

            img{
                width:150px;
                border:#000 solid 1px;
                margin:0px 10px 10px 0px;
            }
        </style>
    </head>
    <body>
        
        <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.
        Phasellus imperdiet, nulla et dictum interdum, nisi lorem egestas odio.
        Maecenas nisl est, ultrices nec congue eget, auctor vitae massa.
        Mauris ante ligula, facilisis sed ornare eu, lobortis in odio.
        </p>
    </body>
</html>
```

```

    #img1{
        float:left;
    }
    #img2{
        float:right;
    }

```

</style>

</head>

<body>

<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.</p>

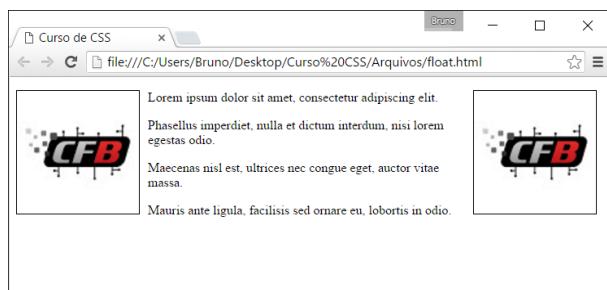
<p>Phasellus imperdiet, nulla et dictum interdum, nisi lorem egestas odio.</p>

<p>Maecenas nisl est, ultrices nec congue eget, auctor vitae massa.</p>

<p>Mauris ante ligula, facilisis sed ornare eu, lobortis in odio.</p>

</body>

</html>

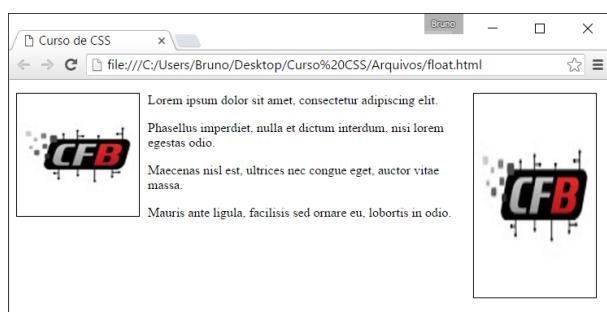


Propositalmente vamos distorcer a imagem da direita para que fique com altura de 250 pixels.

```

#img2{
    float:right;
    height:250px;
}

```

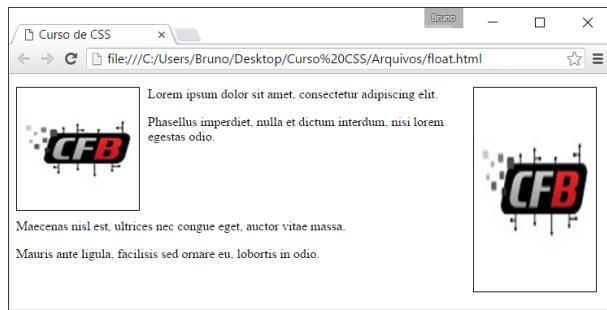


Agora vamos aplicar novamente a classe “fix” no terceiro parágrafo.

```

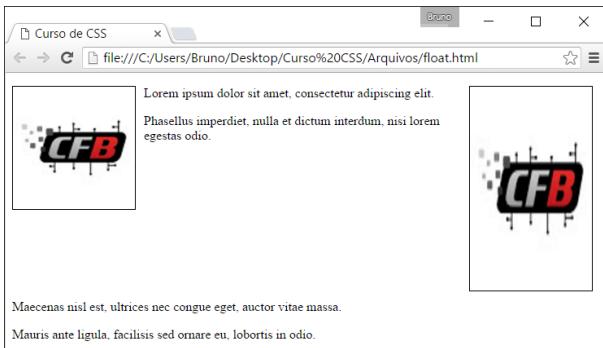
<body>
    
    <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.</p>
    <p>Phasellus imperdiet, nulla et dictum interdum, nisi lorem egestas odio.</p>
    <p class="fix">Maecenas nisl est, ultrices nec congue eget, auctor vitae massa.</p>
    <p>Mauris ante ligula, facilisis sed ornare eu, lobortis in odio.</p>
</body>

```



Como configuramos a propriedade clear com valor left, note que o float:right não foi anulado. Como resolvemos isto? Neste caso como as duas imagens estão antes do <p>, basta configura o clear com valor right, veja a alteração.

```
.fix{
    clear:right;
}
```



Pronto, problema resolvido.

Então os valores possíveis para o clear são:

left → Não permite elementos flutuantes do lado esquerdo.

right → Não permite elementos flutuantes do lado direito.

both → Não permite elementos flutuantes no lado esquerdo e no lado direito.

Vamos ver uma configuração simples de layout usando float.

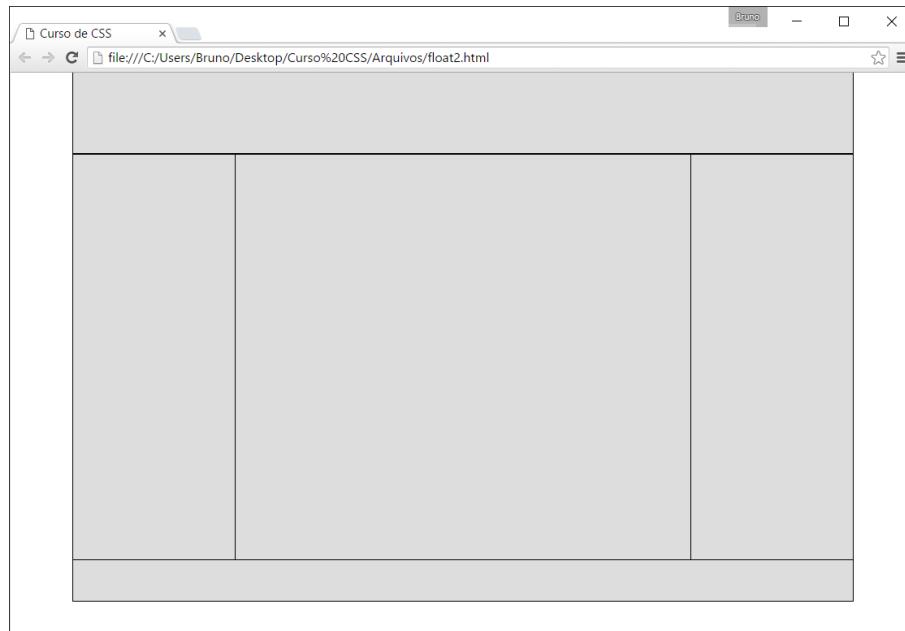
```
<!doctype html>
<html lang="pt-br">
<head>
    <title>Curso de CSS</title>
    <meta charset="UTF-8">

    <style rel="stylesheet">
        body{
            width:960px;
            margin:auto;
            padding:0px;
        }
        nav, header, aside, section, footer{
            outline:#000 solid 1px;
            padding:0px;
            background-color:#DDD;
        }
        nav{
            width:960px;
            height:100px;
        }
        aside{
            width:200px;
            height:500px;
        }
        #as1{
            float:left;
        }
        #as2{
            float:right;
        }
        section{
            float:left;
            width:560px;
            height:500px;
        }
        footer{
            float:left;
            width:960px;
            height:50px;
        }
    </style>

```

```
</style>

</head>
<body>
    <nav></nav>
    <header></header>
    <aside id="as1"></aside>
    <aside id="as2"></aside>
    <section></section>
    <footer></footer>
</body>
</html>
```

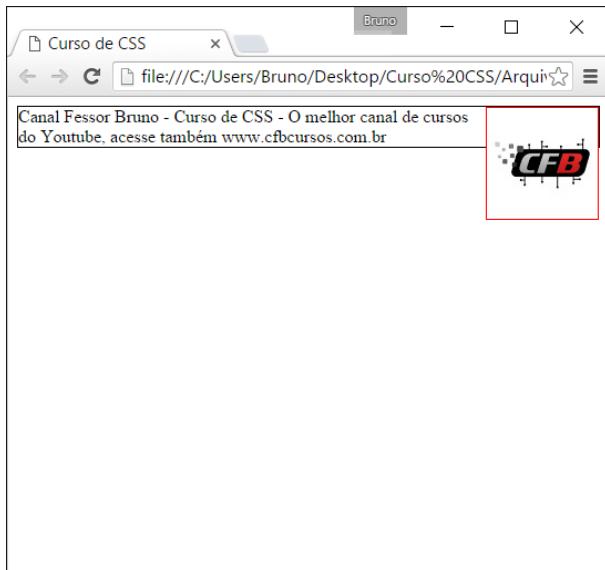


overflow

A propriedade **overflow** especifica qual será o comportamento do elemento pai, caso seu conteúdo extrapole seus limites, em nosso código de exemplo temos uma imagem que está flutuando à direita de um texto.

```
<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de CSS</title>
        <meta charset="UTF-8">

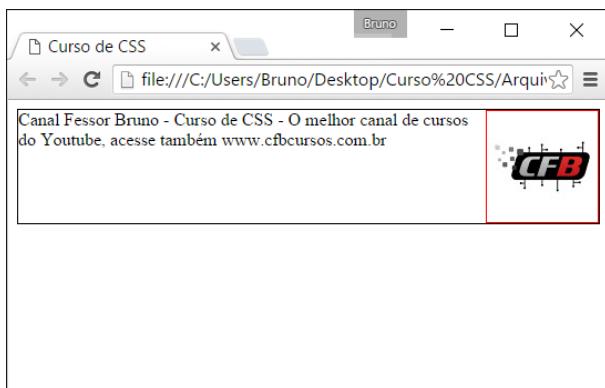
        <style rel="stylesheet">
            div{
                border:#000 solid 1px;
            }
            #img1{
                float:right;
                border:#F00 solid 1px;
            }
        </style>
    </head>
    <body>
        <div>
            
            <span>Canal Fessor Bruno - Curso de CSS - O melhor canal de cursos do Youtube, acesse também www.cfbcursos.com.br<span>
        </div>
    </body>
</html>
```



Adicionei bordas para ficar mais fácil a visualização, note que a imagem extrapolou os limites da div, isso porque está flutuando.

Como resolver isto? Neste caso temos que preparar a div para este caso, vamos configurar o overflow da div.

```
<style rel="stylesheet">
    div{
        border:#000 solid 1px;
        overflow:auto;
    }
    #img1{
        float:right;
        border:#F00 solid 1px;
    }
</style>
```



Veja que com o overflow configurado em auto a div se ajusta ao seu conteúdo, independente se este conteúdo esteja flutuando ou não.

Vamos ver outros valores possíveis para overflow.

visible = Valor padrão.

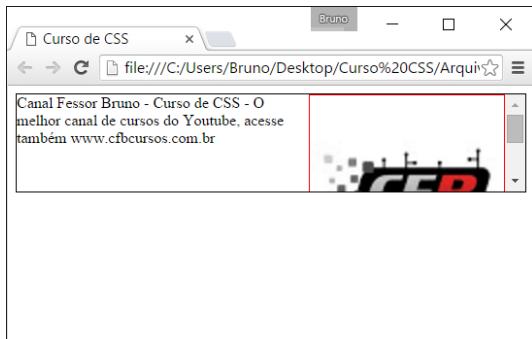
hidden = Esconde a parte que saiu pra fora dos limites.

scroll = Mostra uma barra de rolagem para rolar o conteúdo independente se o conteúdo está extrapolando ou não, não ajusta o tamanho do elemento.

auto = Ajusta o tamanho do elemento de acordo com o conteúdo que esteja extrapolando os limites do elemento, caso o elemento tenha um tamanho fixo, será mostrada uma barra de rolagem.

Vamos realizar algumas alterações em nosso código para aumentar a imagem a div e adicionar barra de rolagem.

```
<style rel="stylesheet">
    div{
        border:#000 solid 1px;
        overflow:auto;
        height:100px;
    }
    #img1{
        float:right;
        border:#F00 solid 1px;
        width:200px;
    }
</style>
```



Note que agora temos uma barra de rolagem disponível na div.

Seletores after e before

No início da apostila aprendemos sobre seletores, até disponibilizei uma tabela com vários seletores que podemos usar, neste capítulo vamos tratar de dois seletores interessantes, after (depois) e before (antes).

Com estes seletores podemos configurar o conteúdo que será mostrado antes ou depois do elemento, vamos aos exemplos.

```
<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de CSS</title>
        <meta charset="UTF-8">

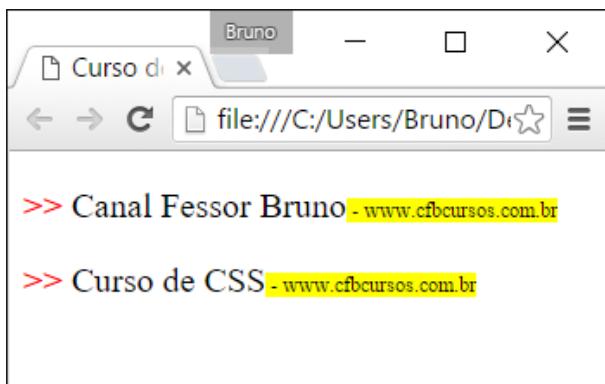
        <style rel="stylesheet">
            p::before{
                content:">> ";
            }
            p::after{
                content:" - www.cfbcursos.com.br";
            }
        </style>
    </head>
    <body>
        <p>Canal Fessor Bruno</p>
        <p>Curso de CSS</p>
    </body>
</html>
```



Note que em todos os elementos <p> foram adicionados ">>" no início e " – www.cfbcursos.com.br" no final.

Vamos adicionar mais algumas formatações.

```
<style rel="stylesheet">
    p::before{
        content:>> ";
        color:#F00;
        font-weight:bold;
    }
    p::after{
        content:" – www.cfbcursos.com.br";
        background-color:#FF0;
        font-size:0.6em;
    }
    p{
        font-size:20px;
    }
</style>
```



OBS: em é uma unidade de medida relativa, veremos sobre isso mais adiante.

Menu dropDown sem script

Vamos usar o que aprendemos até agora para construir um menu estilo dropDown, sem nada se javascript, usando apenas propriedades CSS, parece impossível? Mas não é, na verdade é bem simples.

```
<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de CSS</title>
        <meta charset="UTF-8">

        <style rel="stylesheet">
            .menuDD{
                position:relative;
                display:inline-block;
            }

            .menuBtn{
                background-color:#800;
                color:white;
                padding:20px;
                font-size:20px;
                border:none;
                cursor:pointer;
            }

            .menuConteudo{
                display:none;
                position:absolute;
                background-color:#EEE;
                min-width:160px;
                box-shadow:0px 8px 16px 0px rgba(0,0,0,0.2);
            }
        </style>
    </head>
    <body>
        <div class="menuDD">
            <span>Menu</span>
            <ul class="menuConteudo">
                <li>Item 1</li>
                <li>Item 2</li>
                <li>Item 3</li>
            </ul>
        </div>
    </body>
</html>
```

```

.menuConteudo a{
    color:black;
    padding:12px 16px;
    text-decoration:none;
    display:block;
    font-size:20px;
}

.menuConteudo a:hover{
    background-color:#FDD;
    color:#800;
}

.menudd:hover .menuConteudo{
    display: block;
}

.menudd:hover .menuBtn{
    background-color:#F00;
}

```

</style>

</head>

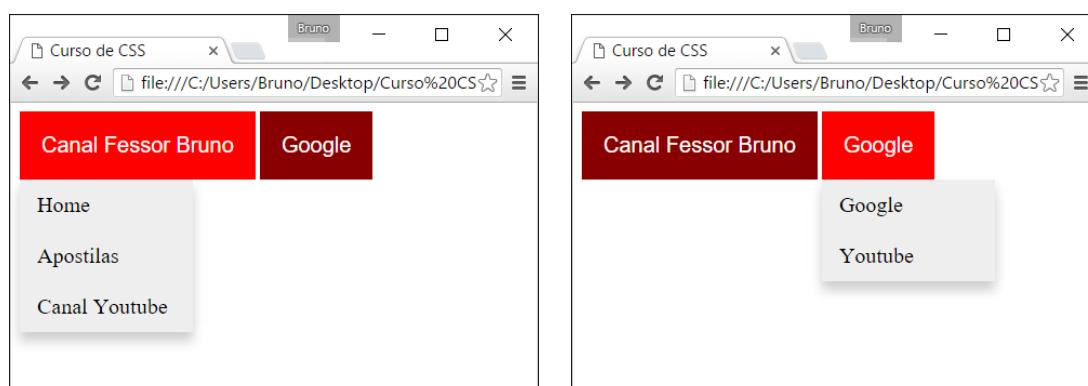
<body>

<div class="menudd">

<button class="menuBtn">Canal Fessor Bruno</button>

<button class="menuBtn">Google</button>

Quando posicionamos o mouse sobre o menu ele mostra seus itens.



transition – Transições – CSS3

Podemos implementar efeitos de transição em nossos elementos usando a propriedade transition, basicamente precisamos informar a propriedade que receberá a transição, width, height, color, visibility, etc e o tempo de duração da transição.

Vamos ao primeiro exemplo.

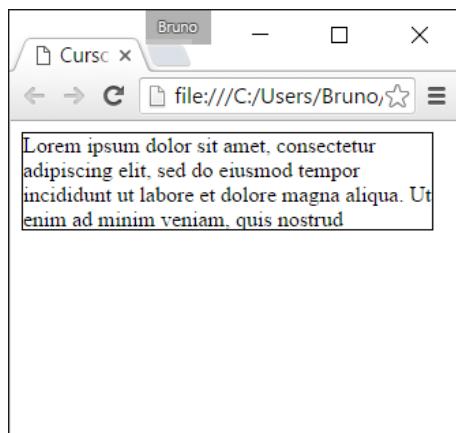
```
<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de CSS</title>
        <meta charset="UTF-8">

        <style rel="stylesheet">

            div{
                width:300px;
                height:70px;
                border:#000 solid 1px;
                overflow:hidden;
                transition:height 2s;
            }

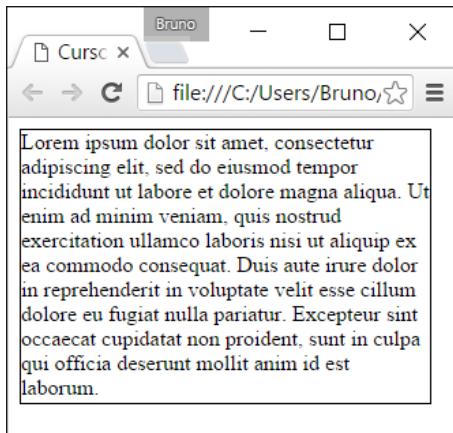
        </style>

    </head>
    <body>
        <div>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.</div>
    </body>
</html>
```



Note que configuramos a transição para a altura em um tempo de 2 segundos, mas você viu o efeito? Não, isso porque não configuramos um disparador para o efeito, a transição vai acontecer quando a propriedade for configurada for alterada, então precisamos configurar um evento interativo para que a transição ocorra, então vamos adicionar hover em nossa div e alterar a altura.

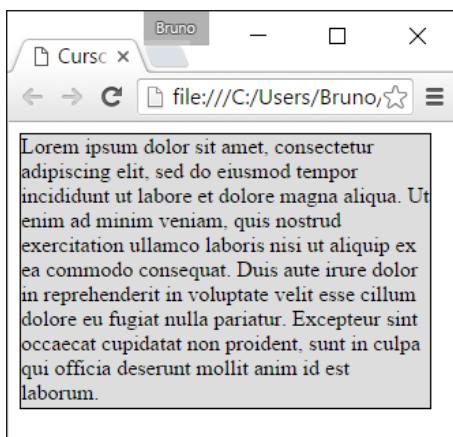
```
<style rel="stylesheet">
    div{
        width:300px;
        height:70px;
        border:#000 solid 1px;
        overflow:hidden;
        transition:height 2s;
    }
    div:hover{
        height:200px;
    }
</style>
```



Pronto, agora sempre que posicionarmos o mouse sobre esta div, ela será aumentada para o altura 200 pixels, suavemente.

Podemos adicionar mais de uma transformação, veja o código, vamos adicionar transição para cor também.

```
<style rel="stylesheet">
    div{
        width:300px;
        height:70px;
        border:#000 solid 1px;
        overflow:hidden;
        transition:height 2s,background-color 2s;
    }
    div:hover{
        height:200px;
        background-color:#DDD;
    }
</style>
```



Agora além da altura a cor também é trocada em transição.

transition-timing-function

Com esta propriedade podemos configurar a curva de velocidade da transição, se será mais acelerada no início, no fim ou no meio, se a velocidade será linear ou podemos até especificar os momentos em que será mais rápida ou devagar.

Os valores possíveis são:

ease = (padrão) Configura a velocidade para começar devagar, acelerar até a metade do efeito e desacelerar no final.

linear = Sem alteração de velocidade do início ao fim com a mesma velocidade.

ease-in = Inicia devagar.

ease-out = Termiona devagar.

ease-in-out – Inicia e termina devagar.

cubic-bezier(n,n,n,n) – Permite definirmos a velocidade em pontos diferentes.

Adicione esta propriedade em nosso código de exemplo para ver o resultado final.

```
div{
    width:300px;
    height:70px;
    border:#000 solid 1px;
    overflow:hidden;
    transition:height 2s,background-color 2s;
    transition-timing-function:linear;
}
```

Podemos inserir o valor de transition-timing-function usando a metapropriedade transition, por exemplo, transition:height 2s linear 2s, background-color 2s linear 2s;

transition-delay

Permite configurar um tempo de atraso para início da animação, ou seja, com este delay configurado a animação não começa de imediato.

```
div{
    width:300px;
    height:70px;
    border:#000 solid 1px;
    overflow:hidden;
    transition:height 2s,background-color 2s;
    transition-delay:1s;
}
```

Com esta propriedade em nosso código a animação irá demorar 1 segundo para começar a reproduzir, tanto no início como no fim.

transition + transform

Podemos juntar os dois recursos para criar uma transição mais interessante, em nosso exemplo, vamos adicionar um efeito de rotação em nossa transição.

```
<style rel="stylesheet">

div{
    width:300px;
    height:70px;
    border:#000 solid 1px;
    overflow:hidden;
    transition:height 2s,background-color 2s,transform 2s;
}
div:hover{
    height:200px;
    background-color:#DDD;
    transform:rotate(360deg);
}

</style>
```

animation – CSS3

Outro recurso fantástico disponível em CSS3 é a possibilidade de criar animações, tudo via código sem a necessidade de um programa de terceiro ou arquivo com a animação.

O primeiro passo é definir o nome e os keyframes da animação, definimos estes parâmetros da seguinte forma:

```
@keyframes nomeDaAnimação{
    /* keyframes */
}
```

Depois basta usar a animação com a propriedade `animation-name`.

Vamos iniciar com uma animação simples.

```
<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de CSS</title>
        <meta charset="UTF-8">

        <style rel="stylesheet">

            @keyframes cfb{
                from{
                    left:0px;
                }
                to{
                    left:400px;
                }
            }

            #dv1{
                position:relative;
                width:100px;
                height:100px;
                border:#000 solid 1px;
                background-color:#00F;
                animation-name:cfb;
                animation-duration:3s;
            }

        </style>

    </head>
    <body>
        <div id="dv1"></div>
    </body>
</html>
```

Esta animação simplesmente move a div iniciando em `left:0px` e terminando em `left:400px`.

Neste primeiro caso especificamos somente dois keyframes, o inicial “from” e o final “to”.

Podemos especificar vários keyframes, veja.

```
<style rel="stylesheet">

    @keyframes cfb{
        0%   {left:0px;}
        20%  {left:500px;}
        40%  {left:250px;}
        60%  {left:500px;}
        80%  {left:250px;}
        100% {left:0px;}
    }

    #dv1{
        position:relative;
        width:100px;
        height:100px;
        border:#000 solid 1px;
        background-color:#00F;
        animation-name:cfb;
        animation-duration:3s;
    }

</style>
```

Nesta caso a animação tem seis keyframes onde modificamos somente a posição LEFT iniciamos em 0px, deslocamos para 500px, depois para 250px, 500px, 250px e por último voltamos em 0px.

Vamos alterar mais um pouco, vamos incluir outras modificações nos keyframes.

```
@keyframes cfb{
    0%   {left:0px; width:100px; height:100px; }
    25%  {left:500px; width:100px; height:100px; }
    50%  {left:500px; width:200px; height:200px; }
    75%  {left:500px; width:100px; height:100px; }
    100% {left:0px; width:100px; height:100px; }
}

#dv1{
    position:relative;
    width:100px;
    height:100px;
    border:#000 solid 1px;
    background-color:#00F;
    animation-name:cfb;
    animation-duration:3s;
}
```

Nesta animação, deslocamos para left:500px, aumentamos, diminuimos e voltamos ao left:0px;

Podemos aplicar transformações também, veja.

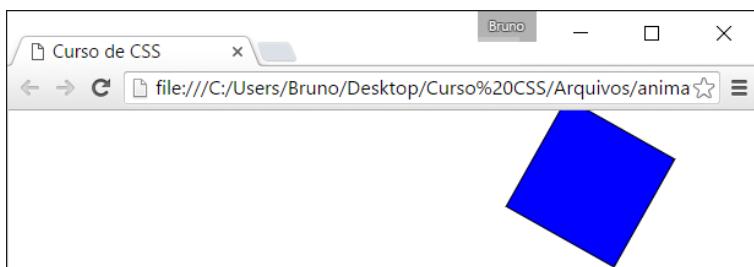
```
<style rel="stylesheet">

    @keyframes cfb{
        0%   {left:0px; }
        25%  {left:500px; transform:rotate(360deg); }
        100% {left:0px; }
    }

    #dv1{
        position:relative;
        width:100px;
        height:100px;
        border:#000 solid 1px;
        background-color:#00F;
        animation-name:cfb;
        animation-duration:3s;
    }

</style>
```

Neste caso a div desloca-se para left:500px girando e volta girando para left:0px.



animation-duration

Para controlar o tempo da animação podemos usar a propriedade `animation-duration`, veja o exemplo da animação anterior com tempo de 5 segundos.

```
@keyframes cfb{
    0%   {left:0px; }
    25%  {left:500px; transform:rotate(360deg); }
    100% {left:0px; }
}

#dv1{
```

```
position: relative;
width: 100px;
height: 100px;
border: #000 solid 1px;
background-color: #00F;
animation-name: cfb;
animation-duration: 5s;
}
```

animation-direction

Direção da animação, no código de exemplo vamos reverter a direção.

Os valores possíveis são:

reverse = Reverte a direção

alternate = Alterna a direção

animation-timing-function

Configura a curva de velocidade da animação, os valores são:

ease = (padrão) Configura a velocidade para começar devagar, acelerar até a metade do efeito e desacelerar no final.

linear = Sem alteração de velocidade do início ao fim com a mesma velocidade.

ease-in = Inicia devagar.

ease-out = Termina devagar.

ease-in-out – Inicia e termina devagar.

cubic-bezier(n,n,n,n) – Permite definirmos a velocidade em pontos diferentes.

animation-iteration-count

Configura quantas vezes a animação irá se repetir.

animation-iteration-count:3; → Animação é executada três vezes

animation-iteration-count:infinite → Animação é reproduzida continuamente, sem parar.

Metapropriedade transition

Podemos juntar todos os parâmetros somente na propriedade transition, veja o exemplo

```
animation: nomeAnimação 8s linear 4s infinite reverse;
```

animation-play-state

Configura o estado da animação, se está rodando ou está em pausa, os valores são:

paused → Pausa a animação.

running → Reproduz a animação.

Colunas – CSS3

Para dividirmos um elemento em colunas podemos usar a metapropriedade columns e ou as demais propriedades para trabalhar com colunas, vamos entender como funciona este recurso.

Usando a propriedade columns informamos a largura da coluna e o número de colunas.

Em nosso primeiro código de exemplo, vamos dividir em 3 colunas o conteúdo da div.

```
<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de CSS</title>
        <meta charset="UTF-8">

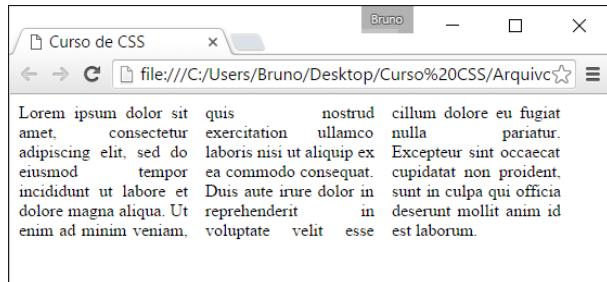
        <style rel="stylesheet">

            #colunas{
                width:500px;
                -webkit-columns:100px 3; /* Chrome, Safari, Opera */
                columns:100px 3; /* Padrão */
            }

        </style>
    </head>
    <body>

        <div id="colunas">
            Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
        </div>

    </body>
</html>
```



Podemos usar outras propriedades para formatar, vamos vê-las a seguir.

column-count

Especifica o número de colunas.

```
.colunas {
    -webkit-column-count: 3; /* Chrome, Safari, Opera */
    -moz-column-count: 3; /* Firefox */
    column-count: 3;
}
```

column-gap

Especifica a distância entre as colunas.

```
.colunas {
    -webkit-column-count: 3; /* Chrome, Safari, Opera */
    -moz-column-count: 3; /* Firefox */
    column-count: 3;

    -webkit-column-gap: 40px; /* Chrome, Safari, Opera */
    -moz-column-gap: 40px; /* Firefox */
    column-gap: 40px;
}
```

column-rule

Configura a linha divisória entre as colunas.

```
.colunas {
    -webkit-column-count: 3; /* Chrome, Safari, Opera */
    -moz-column-count: 3; /* Firefox */
    column-count: 3;

    -webkit-column-gap: 40px; /* Chrome, Safari, Opera */
    -moz-column-gap: 40px; /* Firefox */
    column-gap: 40px;

    -webkit-column-rule: 4px outset #ff00ff; /* Chrome, Safari, Opera */
    -moz-column-rule: 4px outset #ff00ff; /* Firefox */
    column-rule: 4px outset #ff00ff;
}
```

column-span

Especifica que um elemento vai “mesclar” as colunas, como se ignorasse a formação de colunas.

```
.colunas {
    -webkit-column-count: 3; /* Chrome, Safari, Opera */
    -moz-column-count: 3; /* Firefox */
    column-count: 3;
}

h1 { /*Elemento deve estar dentro da div que tem a formatação de colunas*/
    -webkit-column-span: all; /* Chrome, Safari, Opera */
    column-span: all;
}
```

column-width

Configura a largura de cada coluna

```
.colunas {
    -webkit-column-width: 100px; /* Chrome, Safari, Opera */
    -moz-column-width: 100px; /* Firefox */
    column-width: 100px;
}
```

resize

Este parâmetro especifica se a <div> pode ser redimensionada.

```
<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de CSS</title>
        <meta charset="UTF-8">

        <style rel="stylesheet">

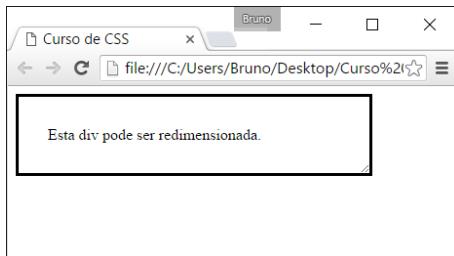
            div{
                border:#000 solid 3px;
                padding:30px;
                width: 300px;
                resize:both;
                overflow:auto;
            }

        </style>

    </head>
    <body>

        <div>Esta div pode ser redimensionada.</div>

    </body>
</html>
```



Os valores que podemos usar são:

both → Podemos redimensionar na horizontal e na vertical

horizontal → Podemos redimensionar na horizontal

vertical → Podemos redimensionar na vertical

box-sizing – CSS3

A propriedade box-sizing é muito interessante, ela permite incluir ao tamanho total de um elemento o padding e a borda.

Isso resolve muitos problemas, na hora de adicionar um padding por exemplo o tamanho original do elemento não é alterado.

Vamos ao exemplo.

```
<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de CSS</title>
        <meta charset="UTF-8">

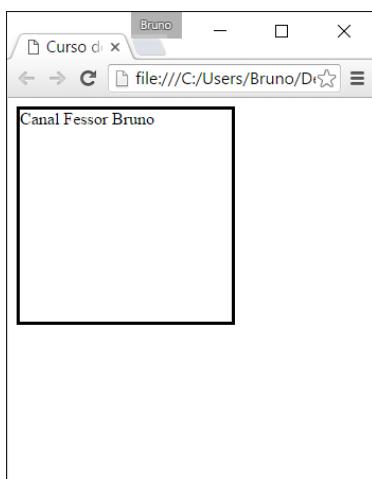
        <style rel="stylesheet">

            div{
                border:#000 solid 3px;
                width: 200px;
                height:200px;
            }

        </style>
    </head>
    <body>

        <div>Canal Fessor Bruno</div>

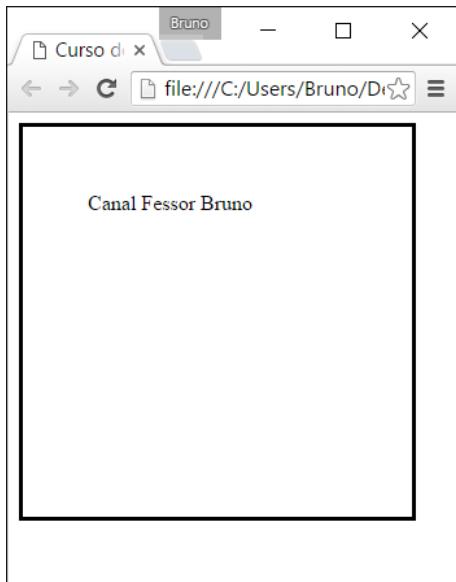
    </body>
</html>
```



Vamos adicionar 50 pixels de padding na div.

```
div{  
    border:#000 solid 3px;  
    width: 200px;  
    height:200px;  
    padding:50px;  
}
```

Neja que o tamanho original da div foi alterado.



Isso acontece porque o tamanho total de um elemento é calculado da seguinte forma.

$\text{width} + \text{padding} + \text{border} = \text{Largura atual do elemento}$.

$\text{height} + \text{padding} + \text{border} = \text{Altura atual do elemento}$.

Ao adiciona a propriedade box-sizing podemos mudar isto.

```
div{  
    border:#000 solid 3px;  
    width: 200px;  
    height:200px;  
    padding:50px;  
    box-sizing:border-box;  
}
```



Note que agora nem as bordas de 3 pixels nem os paddings de 50 pixels foram adicionado ao tamanho do elemento, neste caso, a div possui exatamente o tamanho de 200 pixels de largura por 200 pixels de altura originais.

Os valores possíveis são:

content-box → (Padrão) Os tamanho do elemento é calculado junto com o padding e o border.

border-box → O tamanho do elemento é calculado pelo width e height, sem somar os valores de padding e border.

Filter - CSS3

A propriedade filter adiciona uma série de efeitos visuais ao elemento, efeitos como brilho, embaçamento, contraste, transparência, etc.

Confira a sintaxe da propriedade filter:

```
filter: none | blur() | brightness() | contrast() | drop-shadow() | grayscale() | hue-rotate() | invert() | opacity() |  
saturate() | sepia() | url();
```

No Chrome devemos usar:

```
-webkit-filter: none | blur() | brightness() | contrast() | drop-shadow() | grayscale() | hue-rotate() | invert() |  
opacity() | saturate() | sepia() | url();
```

Vamos aos exemplo.

```
<!doctype html>  
<html lang="pt-br">  
  <head>  
    <title>Curso de CSS</title>  
    <meta charset="UTF-8">  
  
    <style rel="stylesheet">  
      img{width:300px; }  
  
    </style>  
  
  </head>  
  <body>  
      
  </body>  
</html>
```



Vou aplicar os filtros no evento hover para que fique mais fácil visualizar, assim, quando o mouse estiver fora da imagem será mostrada normalmente e quando o mouse estiver sobre a imagem será mostrada com o filtro aplicado.

grayscale

Filtro que configura o elemento em escala de cinza, com valor 100% fica totalmente em escala de cinza.

```
<style rel="stylesheet">
    img{width:300px;}

    img:hover{
        -webkit-filter: grayscale(100%); /* Chrome, Safari, Opera */
        filter: grayscale(100%);
    }
</style>
```



blur

Embaçamento, quanto maior o valor maior será o embaçamento.

```
<style rel="stylesheet">
    img{width:300px;}

    img:hover{
        -webkit-filter: blur(5px); /* Chrome, Safari, Opera */
        filter: blur(5px);
    }
</style>
```



brightness

Intensidade do brilho, 100% é o valor padrão.

```
<style rel="stylesheet">
  img{width:300px;}

  img:hover{
    -webkit-filter: brightness(200%); /* Chrome, Safari, Opera */
    filter: brightness(200%);
  }
</style>
```



contrast

Ajusta o contraste do elemento, 100% é o valor padrão.

```
<style rel="stylesheet">
  img{width:300px;}

  img:hover{
    -webkit-filter: contrast(200%); /* Chrome, Safari, Opera */
    filter: contrast(200%);
  }
</style>
```



drop-shadow

Adicione sombra projetada.

drop-shadow(distânciaX distânciaY tamanhoEmbaçamento cor);

```
<style rel="stylesheet">
    img{width:300px;}

    img:hover{
        -webkit-filter: drop-shadow(8px 8px 10px #F00); /* Chrome, Safari, Opera */
        filter: drop-shadow(8px 8px 10px #F00);
    }
</style>
```



hue-rotate

Matiz

```
<style rel="stylesheet">
    img{width:300px;}

    img:hover{
        -webkit-filter: hue-rotate(90deg); /* Chrome, Safari, Opera */
        filter: hue-rotate(90deg);
    }
</style>
```



invert

inverte as cores.

```
<style rel="stylesheet">
    img{width:300px;}
```

```
img:hover{  
    -webkit-filter: invert(100%); /* Chrome, Safari, Opera */  
    filter: invert(100%);  
}
```



opacity

Opacidade, controle a transparência, quanto maior o valor, menos opaco, mais transparente.

```
<style rel="stylesheet">  
img{width:300px;}  
  
img:hover{  
    -webkit-filter: opacity(50%); /* Chrome, Safari, Opera */  
    filter: opacity(50%);  
}  
</style>
```



saturate

Saturação, intensidade das cores.

```
<style rel="stylesheet">  
img{width:300px;}  
  
img:hover{  
    -webkit-filter: saturate(800%); /* Chrome, Safari, Opera */  
    filter: saturate(800%);  
}
```

```
</style>
```



sepia

Efeito de coloração, pode ficar semelhante a envelhecimento.

```
<style rel="stylesheet">
    img{width:300px;}

    img:hover{
        -webkit-filter: sepia(100%); /* Chrome, Safari, Opera */
        filter: sepia(100%);
    }
</style>
```



Aplicando vários filtros.

```
<style rel="stylesheet">
    img{width:300px;}

    img:hover{
        -webkit-filter: contrast(200%) brightness(150%); /* Chrome, Safari, Opera */
        filter: contrast(200%) brightness(150%);
    }
</style>
```

opacity

O parâmetro opacity funciona exatamente como o filtro opacity, controla a opacidade do elemento, o valor é entre 1 mais opaco sem transparência e 0 menos opaco ou transparência total, 0.5 meia opacidade ou meia transparência.

```
img {  
    opacity: 0.5;  
}
```

No exemplo acima configuramos o elemento img com transparência 50%.

@media – layouts responsivos

As regras media permitem configurar diferentes regras CSS para diferentes tipos de mídia, diante da necessidade de facilitar a configuração do site para se adaptar bem em dispositivos de diferentes tamanhos, dai vem o conceito de design responsivo ou layout responsivo.

Pode ter várias configurações de estilo para um mesmo site, para que se adapte aos diversos dispositivos, podemos ter estilos para computadores, celulares, impressão, televisão, tablets, etc.

As regras media queries podem verificas várias características dos dispositivos como, largura e altura da janela, largura e altura do dispositivo, orientação (se o dispositivo está na horizontal/paisagem ou vertical/retrato), resolução.

A sintaxe é a seguinte:

```
@media not|only mediatype and (expressões) {  
    Código CSS/  
}
```

O resultado de um comando @media é do tipo boolean (true/false), verdadeiro ou falso, se o tipo de dispositivo corresponde ao tipo de mídia terá um retorno true/verdadeiro e as regras CSS estabelecidas para esta @media serão aplicadas.

Também podemos definir diferentes folhas de estilo para diferentes mídias.

```
<link rel="stylesheet" media="print()" href="impressora.css">  
<link rel="stylesheet" media="screen and (max-width: 500px)" href="celular.css">  
<link rel="stylesheet" media="screen and (min-width: 500px)" href="pc.css">  
<link rel="stylesheet" media="screen and (min-width: 1024px)" href="telasGrandes.css">
```

Os valores que podemos usar para os tipos de mídia são.

all → Todos os tipos de dispositivos

print → Usado para impressoras

screen → Usado para telas, computador, celulares, tablets, etc.

speech → Usado para telas de leitores de contúdo, leitores de ebook por exemplo, em voz alta.

Vamos a um primeiro código de exemplo onde iremos configurar a cor de fundo da tela para vermelho, mas iremos adicionar uma regra @media que irá muda a cor da tela para amarelo quando a largura for maior que 500 pixels, iremos definir a largura mínima para 500 pixels.

```
<!doctype html>  
<html lang="pt-br">  
    <head>  
        <title>Curso de CSS</title>  
        <meta charset="UTF-8">  
  
        <style rel="stylesheet">  
            body {  
                background-color:#F00;
```

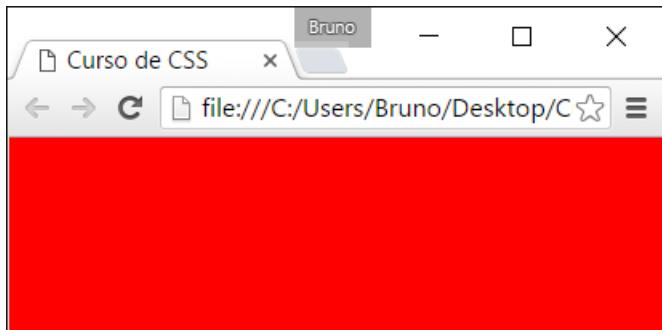
```
        }

        @media screen and (min-width: 500px) {
            body {
                background-color:#FF0;
            }
        }
    </style>

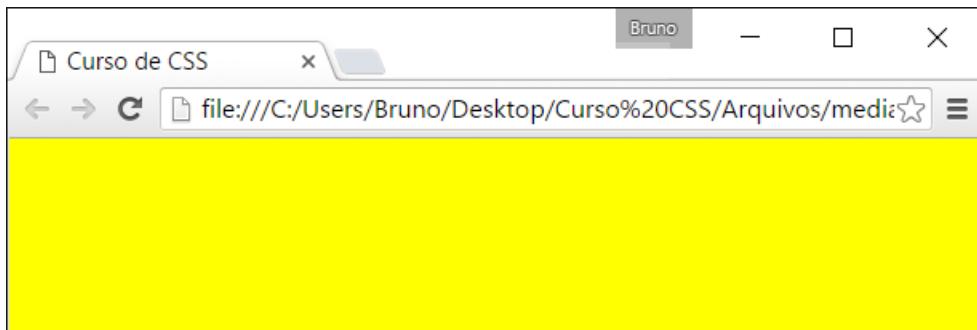
</head>
<body>

</body>
</html>
```

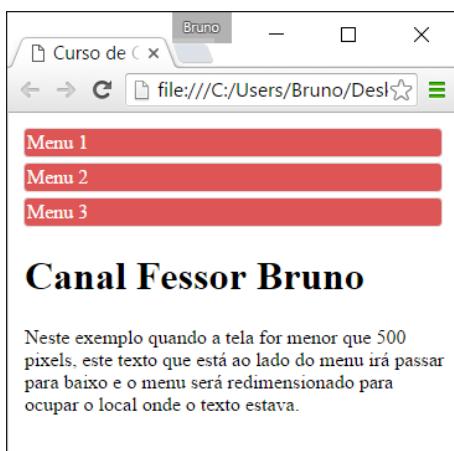
Tela com largura menor que 500 pixels.



Tela com largura maior que 500 pixels.



Vamos a um segundo exemplo, onde nosso layout com um menu e um texto se ajustará ao tamanho da tela, quando a janela tiver largura menor que 500 pixels se ajustará com o menu em cima do texto.



Já quando a janela tiver largura maior que 500 pixels o menu ficará ao lado esquerdo do texto.



Veja o código.

```
<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de CSS</title>
        <meta charset="UTF-8">

        <style rel="stylesheet">
            .container{overflow:auto;}

            #txtPrincipal{margin-left:216px;}
            #cxMenu{width:200px; float:left;}

            #menu{
                margin:0;
                padding:0;
            }

            .menuItem{
                background:#D55;
                color:#FFF;
                border:#ddd solid 1px;
                border-radius:4px;
                list-style-type:none;
                margin:4px;
                padding:2px;
            }

        </style>

    </head>
    <body>

        <div class="container">
            <div id="cxMenu">
                <ul id="menu">
                    <li class="menuItem">Menu 1</li>
                    <li class="menuItem">Menu 2</li>
                    <li class="menuItem">Menu 3</li>
                </ul>
            </div>
            <div id="txtPrincipal">
                <h1>Canal Fessor Bruno</h1>
                <p>Neste exemplo quando a tela for menor que 500 pixels, este texto que está ao lado do menu irá passar para baixo e o menu será redimensionado para ocupar o local onde o texto estava.</p>
            </div>
        </div>
    </body>
</html>
```

Este é o código sem a responsividade, independente da largura da janela, o layout sempre será o mesmo.



Agora vamos adicionar a rotina @media.

```
<style rel="stylesheet">
    .container{overflow:auto;}
    #txtPrincipal{margin-left:216px;}
    #cxMenu{width:200px; float:left;}
    #menu{
        margin:0;
        padding:0;
    }
    .menuItem{
        background:#D55;
        color:#FFF;
        border:#ddd solid 1px;
        border-radius:4px;
        list-style-type:none;
        margin:4px;
        padding:2px;
    }
    @media screen and (max-width:500px){
        #cxMenu{float:none; width:auto;}
        #txtPrincipal{margin-left:5px;}
    }
</style>
```

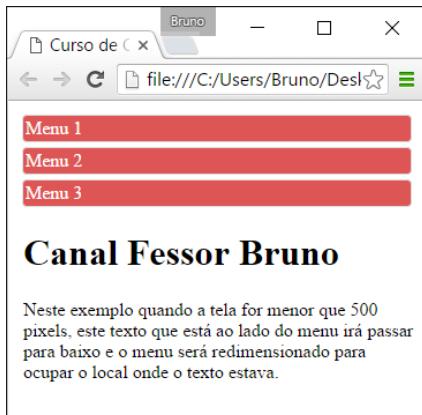
Neste instrução @media indicamos:

`@media screen and (max-width:500px){` → quando a largura máxima da tela for 500 pixels, ou seja, enquanto a tela for menor que 500 pixels, os comandos a seguir estarão atuando.

`#cxMenu{float:none; width:auto;}` → Formata o menu sem flutuar e com a largura em auto para se ajustar ao tamanho atual da janela.

`#txtPrincipal{margin-left:5px;}` → O texto principal recebe uma margem esquerda de 5 pixels para se diatânciar um pouco da borda esquera da janela.

Com esta nova implementação de @media, quando a janela estiver menor que 500 pixels se apresentará como na ilustração a seguir.



As propriedades possíveis de utilização como valor para expressão de verificação em @media estão mostradas na tabela a seguir.

Value	Description
aspect-ratio	A proporção entre a largura e a altura da janela de visualização
color	O número de bits por componente de cor para o dispositivo de saída
color-index	Quantidade de cores que o dispositivo pode exibir
device-aspect-ratio	A proporção entre a largura e a altura que o dispositivo pode exibir
device-height	Medida da altura do dispositivo, por exemplo a tela do computador
device-width	Medida da largura do dispositivo, por exemplo a tela do computador
grid	Se o dispositivo é um grid ou bitmap
height	Altura da janela
max-aspect-ratio	A proporção máxima entre largura e altura da área da tela
max-color	O número de bits máximo por componente de cor para o dispositivo de saída
max-color-index	Quantidade máxima de cores que o dispositivo pode exibir
max-device-aspect-ratio	A proporção máxima entre a largura e a altura que o dispositivo pode exibir
max-device-height	Medida máxima da altura do dispositivo, por exemplo a tela do computador
max-device-width	Medida máxima da largura do dispositivo, por exemplo a tela do computador
max-height	Altura máxima da janela
max-monochrome	O número de bits máximo por componente de cor para o dispositivo de saída monocromático
max-resolution	Resolução máxima do dispositivo, em dpi ou dpcm
max-width	Largura máxima da janela
min-aspect-ratio	A proporção mínima entre largura e altura da área da tela
min-color	O número de bits mínimo por componente de cor para o dispositivo de saída
min-color-index	Quantidade mínima de cores que o dispositivo pode exibir
min-device-aspect-ratio	A proporção mínima entre a largura e a altura que o dispositivo pode exibir
min-device-width	Medida mínima da largura do dispositivo, por exemplo a tela do computador
min-device-height	Medida mínima da altura do dispositivo, por exemplo a tela do computador
min-height	Altura mínima da janela
min-monochrome	O número de bits mínimo por componente de cor para o dispositivo de saída monocromático
min-resolution	Resolução mínima do dispositivo, em dpi ou dpcm
min-width	Largura mínima da janela
monochrome	O número de bits por componente de cor para o dispositivo de saída em dispositivo monocromático
orientation	Orientação do dispositivo, retrato ou paisagem
overflow-block	Conteúdo transborda para fora do bloco principal
overflow-inline	Quando o conteúdo que transborda a janela ao longo da linha
resolution	Resolução do dispositivo, em dpi ou dpcm
scan	Processo de escanamento/varredura da tela
update-frequency	Quão rápido o dispositivo pode atualizar o conteúdo mostrado, frequência de atualização da tela
width	Largura da janela

Vamos ver mais um layout responsivo, desta vez com 3 possibilidades de apresentação, telas com largura menor que 450 pixels, entre 451 e 800 pixels e maior que 800 pixels que vai ser a apresentação normal, então, precisamos criar duas regras @media.

Apresentação acima de 800 pixels de largura.



Layout responsivo - CFB - Curso de CSS

Menu 1
Menu 2
Menu 3
Menu 4

Canal Fessor Bruno

No canal do youtube www.youtube.com/canalfessorbruno ou o site www.cfbcursos.com.br você encontra os melhores cursos de informática.

Curso

Curso de CSS3, conteúdo exclusivo do canalfessorbruno no youtube e do site www.cfbcursos.com.br

Custo

Conteúdo 100% gratuito

Layout responsivo do curso de CSS - www.cfbcursos.com.br - youtube canalfessorbruno

Apresentação entre 451 e 800 pixels de largura.

Layout responsivo - CFB - Curso de CSS

Menu 1
Menu 2
Menu 3
Menu 4

Canal Fessor Bruno

No canal do youtube www.youtube.com/canalfessorbruno ou o site www.cfbcursos.com.br você encontra os melhores cursos de informática.

Curso

Curso de CSS3, conteúdo exclusivo do canalfessorbruno no youtube e do site www.cfbcursos.com.br

Custo

Conteúdo 100% gratuito

Layout responsivo do curso de CSS - www.cfbcursos.com.br - youtube canalfessorbruno

Apresentação menor ou igual a 450 pixels.

Layout responsivo - CFB - Curso de CSS

Menu 1
Menu 2
Menu 3
Menu 4

Canal Fessor Bruno

No canal do youtube www.youtube.com/canalfessorbruno ou o site www.cfbcursos.com.br você encontra os melhores cursos de informática.

Curso

Curso de CSS3, conteúdo exclusivo do canalfessorbruno no youtube e do site www.cfbcursos.com.br

Custo

Conteúdo 100% gratuito

Layout responsivo do curso de CSS - www.cfbcursos.com.br - youtube canalfessorbruno

Vamos ao código.

```
<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de CSS</title>
        <meta charset="UTF-8">

        <style rel="stylesheet">
            body {
                font-family: Arial;
            }

            h1{
                font-size:1.5em;
            }

            h2{
                font-size:1em;
            }

            p{
                font-size:1em;
            }

            header, aside, section, footer, nav{
                padding:1%;
            }

            header{
                background-color:#eee;
                width:98%;
            }

            nav{
                width:13%;
                float:left;
            }

            nav a{
                margin:1%;
                margin-left:0;
                margin-top:0;
                border-bottom:#ddd solid 1px;
                cursor:pointer;
                display:block;
            }

            section{
                width:53%;
                float:left;
            }

            aside{
                background-color:#FDD;
                float:right;
                width:28%;
            }

            footer{
                width:98%;
                text-align:center;
                background-color:#ddd;
                clear:both;
            }
        </style>
    </head>
    <body>
        <header>
            <h1>Layout responsivo - CFB - Curso de CSS<h1>
        </header>
        <nav>
            <a href="#">Menu 1</a>
            <a href="#">Menu 2</a>
            <a href="#">Menu 3</a>
            <a href="#">Menu 4</a>
```

```

</nav>
<section>
    <h1>Canal Fessor Bruno</h1>
    <p>No canal do youtube www.youtube.com/canalfessorbruno ou o site www.cfbcursos.com.br você
encontra os melhores cursos de informática.</p>
    
</section>
<aside>
    <h1>Curso</h1>
    <p>Curso de CSS3, conteúdo exclusivo do canalfessorbruno no youtube e do site
www.cfbcursos.com.br</p>
    <h1>Custo</h1>
    <p>Conteúdo 100% gratuito</p>
</aside>

<footer>
    <p>Layout responsivo do curso de CSS - www.cfbcursos.com.br - youtube:canalfessorbruno</p>
</footer>
</body>
</html>

```

Este é o código sem as técnicas de responsividade, ou seja o layout ainda não se ajusta à tela.

Vamos adicionar as regras de responsividade.

```

<style rel="stylesheet">
    body{
        font-family: Arial;
    }

    h1{
        font-size:1.5em;
    }

    h2{
        font-size:1em;
    }

    p{
        font-size:1em;
    }

    header, aside, section, footer, nav{
        padding:1%;
    }

    header{
        background-color:#eee;
        width:98%;
    }

    nav{
        width:13%;
        float:left;
    }

    nav a{
        margin:1%;
        margin-left:0;
        margin-top:0;
        border-bottom:#ddd solid 1px;
        cursor:pointer;
        display:block;
    }

    section{
        width:53%;
        float:left;
    }

    aside{
        background-color:#FDD;
        float:right;
        width:28%;
    }

    footer{

```

```
width:98%;  
text-align:center;  
background-color:#ddd;  
clear:both;  
}  
  
@media screen and (max-width:800px) and (min-width:451px) {  
    section{  
        width:83%;  
        padding:1%;  
        margin:0%;  
    }  
    aside{  
        width:98%;  
        clear:both;  
        padding:1%;  
        margin:0%;  
    }  
}  
  
@media screen and (max-width:450px){  
    nav, section, aside, footer, nav a{  
        width:98%;  
        clear:both;  
        padding:1%;  
        margin:0%;  
    }  
}  
}  
</style>
```

A primeira regra @media formata o layout quando a tela tem largura máxima de 800 pixels e mínima de 451 pixels.

@media screen and (max-width:800px) and (min-width:451px){

Quando esta regra for verdadeira iremos alterar a configuração do tamanho do elemento section e vamos descer o elemento aside.

```
section{  
    width:83%;  
    padding:1%;  
    margin:0%;  
}  
  
aside{  
    width:98%;  
    clear:both;  
    padding:1%;  
    margin:0%;  
}
```

A segunda regra @media formata o layout quando a janela for menor ou igual a 450 pixels, esta regra apresenta os elementos um abaixo do outro.

```
@media screen and (max-width:450px){  
    nav, section, aside, footer, nav a{  
        width:98%;  
        clear:both;  
        padding:1%;  
        margin:0%;  
    }  
}
```

Unidades de medida

Veja a tabela com as unidades de medida e a descrição.

Unidade	Descrição
em	Unidade relativa ao tamanho do font-size anterior, aumenta ou diminui a fonte em relação a fonte-size, se elemento A tem font-size:15px e elemento B tem font-size:2em, então o tamanho do elemento B é igual a 30px. ($15\text{px} \times 2\text{em} = 30\text{px}$) ($5\text{px} \times 4\text{em} = 20\text{px}$) ($2\text{px} \times 1.5\text{em} = 3\text{px}$)
ex	Unidade relativa à altura da letra x em font-size anterior.
ch	Unidade relativa à largura do numeral zero do font-size anterior.
rem	Unidade relativa ao font-size do elemento raiz.
vw	Unidade relativa a 1% da largura da janela.
vh	Unidade relativa a 1% da altura da janela.
vmin	Unidade relativa a 1% da altura ou largura da janela, o que for menor.
vmax	Unidade relativa a 1% da altura ou largura da janela, o que for maior.
%	Porcentagem.
cm	Unidade absoluta relacionada a centímetros.
mm	Unidade absoluta relacionada a milímetros.
in	Unidade absoluta relacionada a polegadas. ($1\text{in} = 96\text{px} = 2.54\text{cm}$)
px	Unidade absoluta relacionada a pixels. ($1\text{px} = 1/96\text{th of 1in}$)
pt	Unidade absoluta relacionada a pontos. ($1\text{pt} = 1/72\text{ of 1in}$)
pc	Unidade absoluta relacionada a picas, pronuncia-se “paicas”. ($1\text{pc} = 12\text{pt}$)

inherit, initial

Todas as propriedade possuem os valores inherit e initial, vou explicar de forma simples estes valores.

initial: Uma propriedade configurada com valor “initial”, define que o valor para esta propriedade será o valor inicial, todas as propriedades tem um valor inicial, primeiro valor, veja o exemplo.

```
.caixa{
    color: initial;
}
```

O código acima define a cor do texto da classe caixa como preto, se essa for a cor padrão do navegador para esta propriedade.

inherit: Uma propriedade configurada com valor inherit tem o mesmo valor desta propriedade do seu elemento pai

```
<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de CSS</title>
        <meta charset="UTF-8">

        <style rel="stylesheet">
            body {
                color:#F00;
            }
            h1{
                color:inherit;
            }
        </style>
    </head>
    <body>
        <h1>Canal Fessor Bruno</h1>
    </body>
</html>
```

No código de exemplo acima a cor do elemento `<h1>` será vermelho, pois é a cor definida com elemento pai o `<body>`.

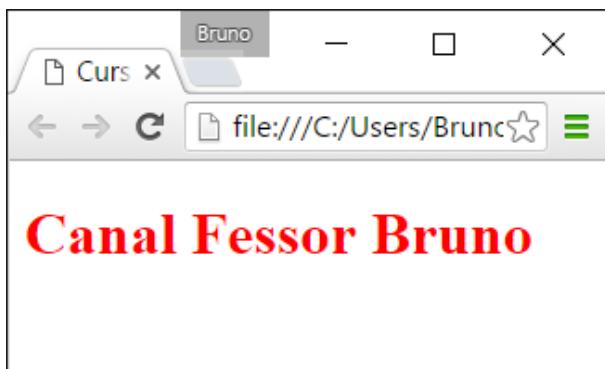


Tabela geral dos parâmetros CSS

CSS	JavaScript	Descrição
		*Os valores sublinhados são os padrões. *& indica que as propriedades anteriores se repetem após o valor indicado.
<td aligncontent<="" td=""><td>Alinhamento entre as linhas dentro de um container flexível quando os itens não usam todo o espaço disponível. Valores: <u>stretch</u>, center, flex-start, flex-end, space-between, space-around, initial, inherit.</td></td>	<td>Alinhamento entre as linhas dentro de um container flexível quando os itens não usam todo o espaço disponível. Valores: <u>stretch</u>, center, flex-start, flex-end, space-between, space-around, initial, inherit.</td>	Alinhamento entre as linhas dentro de um container flexível quando os itens não usam todo o espaço disponível. Valores: <u>stretch</u> , center, flex-start, flex-end, space-between, space-around, initial, inherit.
<td alignitems<="" td=""><td>Alinhamento para itens dentro de um container flexível. Valores: <u>stretch</u>, center, flex-start, flex-end, space-between, space-around, baseline, initial, inherit.</td></td>	<td>Alinhamento para itens dentro de um container flexível. Valores: <u>stretch</u>, center, flex-start, flex-end, space-between, space-around, baseline, initial, inherit.</td>	Alinhamento para itens dentro de um container flexível. Valores: <u>stretch</u> , center, flex-start, flex-end, space-between, space-around, baseline, initial, inherit.
<td alignself<="" td=""><td>Definir o alinhamento de um dos itens no interior de um elemento flexível para se ajustar ao container. Valores: <u>auto</u>, stretch, center, flex-start, flex-end, space-between, space-around, baseline, initial, inherit.</td></td>	<td>Definir o alinhamento de um dos itens no interior de um elemento flexível para se ajustar ao container. Valores: <u>auto</u>, stretch, center, flex-start, flex-end, space-between, space-around, baseline, initial, inherit.</td>	Definir o alinhamento de um dos itens no interior de um elemento flexível para se ajustar ao container. Valores: <u>auto</u> , stretch, center, flex-start, flex-end, space-between, space-around, baseline, initial, inherit.
animation	animation	Metapropriedade. Define todas as propriedades de animação a seguir, menos animationPlayState.
animation-delay	animationDelay	Define quando a animação vai começar. Valores: time, initial, inherit. (padrão time com valor 0)
animation-direction	animationDirection	Direção da animação. Valores: <u>normal</u> , reverse, alternate, alternate-reverse, initial, inherit
animation-duration	animationDuration	Tempo de duração da animação em milissegundos. Valores: time, initial, inherit. (padrão time com valor 0)
animation-fill-mode	animationFillMode	Quais valores serão aplicadas pela animação fora do período que ele está executando. Valores: <u>none</u> , forwards, backwards, both, initial, inherit
animation-iteration-count	animationIterationCount	Número de vezes que uma animação deve ser executada. Valores: number, infinite, initial, inherit (padrão number com valor 1)
animation-name	animationName	Nome da animação ou para os @keyframes da animação. Valores: <u>none</u> , keyframename, initial, inherit
animation-timing-function	animationTimingFunction	Curva de velocidade da animação. Valores: linear, <u>ease</u> , ease-in, ease-out, cubic-bezier(n,n,n,n), initial, inherit
animation-play-state	animationPlayState	Define se a animação está em execução ou em pausa. Valores: <u>running</u> , paused, initial, inherit
azimuth	azimuth	Posicionamento espacial do áudio gerado pelo objeto. Valores: ângulo, left-side, far-left, left, center-left, <u>center</u> , center-right, right, far-right, right-side, [& behind], leftwards, rightwards, inherit
background	background	Metapropriedade. Define backgroundColor, backgroundImage, backgroundRepeat, backgroundAttachment e backgroundPosition.
background-attachment	backgroundAttachment	Indica se a imagem de fundo vai ser fixa ou vai rolar. Valores: <u>scroll</u> , fixed, inherit

background-color	backgroundColor	Cor de fundo. Valores: cor, transparente, inherit
background-image	backgroundImage	Imagem de fundo. Valores: URL, none, inherit
background-position	backgroundPosition	Cordenadas X e Y da imagem de fundo. Valores: porcentagem, tamanho (exemplo: x,y), top, center, bottom, left, center, right, inherit
background-repeat	backgroundRepeat	Configura a repetição da imagem de fundo. Valores: repeat, repeat-x, repeat-y, no-repeat, inherit
background-clip	backgroundClip	Especifica a área de exibição da imagem de fundo. Valores: border-box, padding-box, content-box, initial, inherit
background-origin	backgroundOrigin	Posicionamento de origem da imagem. Valores: padding-box, border-box, content-box, initial, inherit
background-size	backgroundSize	Tamanho da imagem de fundo. Valores: auto, length, cover, contain, initial, inherit
backface-visibility	backfaceVisibility	A propriedade backfaceVisibility define se ou não um elemento deve ser visível quando não está rotacionada de frente para a tela. Esta propriedade é útil quando um elemento é rodado, e você não quer ver a parte de trás. Valores: visible, hidden, initial, inherit
border	border	Metapropriedade. Define todas as bordas ao mesmo tempo.
border-bottom	borderBottom	Metapropriedade. Define as propriedades borderBottomWidth, borderBottomStyle e borderBottomColor.
border-bottom-color	borderBottomColor	Cor da borda inferior. Valores: cor, inherit
border-radius	borderRadius	Define o arredondamento de todas as bordas igualmente Valores: length%, initial, inherit (valor padrão 0)
border-bottom-left-radius	borderBottomLeftRadius	Arredonda a borda inferior esquerda Valores: 0 (% porcentagem), initial, inherit (padrão valor = 0)
border-bottom-right-radius	borderBottomRightRadius	Arredonda a borda inferior direita Valores: 0 (% porcentagem), initial, inherit (padrão valor = 0)
border-top-left-radius	borderTopLeftRadius	Arredonda a borda superior esquerda Valores: 0 (% porcentagem), initial, inherit (padrão valor = 0)
border-top-right-radius	borderTopRightRadius	Arredonda a borda superior direita Valores: 0 (% porcentagem), initial, inherit (padrão valor = 0)
border-bottom-style	borderBottomStyle	Estilo da borda inferior. Valores: none, hidden, dotted, dashed, solid, double, groove, ridge, inset, outset, inherit
border-bottom-width	borderBottomWidth	Largura da borda inferior. Valores: thin, medium, thick, tamanho, inherit
Borde-collapse	borderCollapse	Reduzir a borda da tabela Valores: separate, collapse, initial, inherit
border-color	borderColor	Cores de todas as bordas.
border-image	borderImage	Metapropriedade, especifica uma imagem como borda em torno de um elemento <div> Valores: source slice width outset repeat, initial, inherit
border-image-outset	borderImageOutset	Configura a imagem de borda para fora ou para dentro das bordas originais da <div> Valores: length, number, initial, inherit (valor padrão 0), para ajustar internamente use valores menores que 1, por exemplo 0.1
border-image-repeat	borderImageRepeat	Configura a repetição da imagem usada como borda. Valores: stretch, repeat, round, initial, inherit
border-image-slice	borderImageSlice	Especifica o deslocamento/distância das imagens da borda em caso de repetição Valores: number %, fill, initial, inherit (valor padrão 100%)
border-image-source	borderImageSource	Especifica qual imagem será usada como borda em torno de um elemento div: Valores: none, image, initial, inherit
border-image-width	borderImageWidth	Largura da imagem de borda Valores: number %, auto, initial, inherit (valor padrão 1)

border-left	borderLeft	Semelhante a borderBottom aplicado à borda esquerda.
border-left-color	borderLeftColor	Semelhante a borderBottomColor aplicado à borda esquerda.
border-left-style	borderLeftStyle	Semelhante a borderBottomStyle aplicado à borda esquerda.
border-left-width	borderLeftWidth	Semelhante a borderBottomWidth aplicado à borda esquerda.
border-right	borderRight	Semelhante a borderBottom aplicado à borda direita.
border-right-color	borderRightColor	Semelhante a borderBottomColor aplicado à borda direita.
border-right-style	borderRightStyle	Semelhante a borderBottomStyle aplicado à borda direita.
border-right-width	borderRightWidth	Semelhante a borderBottomWidth aplicado à borda direita.
border-spacing	borderSpacing	Espaçamento entre as bordas de duas células adjacentes, somente para tabelas. Valores: tamanho, inherit.
border-style	borderStyle	Estilo de todas as bordas. Valores: <u>none</u> , hidden, dotted, dashed, solid, double, groove, ridge, inset, outset, inherit
border-top	borderTop	Semelhante a borderBottom aplicado à borda esquerda.
border-top-color	borderTopColor	Semelhante a borderBottom aplicado à borda esquerda.
border-top-style	borderTopStyle	Semelhante a borderBottom aplicado à borda esquerda.
border-top-width	borderTopWidth	Semelhante a borderBottom aplicado à borda esquerda.
border-width	borderWidth	Largura de todas as bordas. Valores: thin, <u>medium</u> , thick, tamanho
bottom	bottom	Distância entre a base de um objeto e a base do bloco onde está contido. Valores: tamanho, porcentagem, <u>auto</u> , inherit
box-shadow	boxShadow	Adiciona sombra a um elemento div. Valores: <u>none</u> , h-shadow v-shadow blur spread color, inset, initial, inherit
box-sizing	boxSizing	Permite definir certos elementos para caber dentro de em um outro elemento. Valores: <u>content-box</u> , border-box, initial, inherit
caption-side	captionSide	Define ou retorna a posição do table caption. Valores: <u>top</u> , bottom, initial, inherit
clear	clear	Com o valor configurado diferente de none, será posicionado abaixo dos elementos flutuantes que estiverem alinhados da maneira especificada aqui. Valores: <u>none</u> , left, right, both, inherit
clip	clip	Área visível de um objeto. Valores: forma, <u>auto</u> , inherit.
color	color	Cor do texto. Valores: cor, inherit
column-count	columnCount	Especifica o número de colunas que um elemento pode conter Valores: number, auto, initial, inherit
column-gap	columnGap	Especifica a lacuna, separação entre as colunas Valores: length, <u>normal</u> , initial, inherit
column-rule	columnRule	Metapropriedade para definir todas as configurações de columnRule
column-rule-color	columnRuleColor	Especifica a cor da barra divisória entre as colunas Valores: color, initial, inherit
column-rule-style	columnRuleStyle	Define o estilo entre as colunas Valores: <u>none</u> , hidden, dotted, dashed, solid, double, groove, ridge, inset, outset, initial, inherit
column-rule-width	columnRuleWidth	Largura da separação entre as colunas Valores: <u>medium</u> , thin, thick, length, initial, inherit
columns	columns	Número de colunas Valores: <u>auto</u> , column-width column-count, initial, inherit

column-span	columnSpan	Especifica quantas colunas um elemento deve abranger Valores: 1, all, initial, inherit
column-width	columnWidth	Largura da coluna Valores: <u>auto</u> , length, initial, inherit
cursor	cursor	Estilo do ponteiro do Mouse. Valores: url, <u>auto</u> , crosshair, default, pointer, move, e-resize, ne-resize, n-resize, se-resize, sw-resize, s-resize, w-resize, text, wait, help, inherit.
direction	direction	Direção do texto de um objeto e a ordem de construção das colunas de uma tabela. Valores: <u>ltr</u> , rtl, inherit
display	display	Forma de visualização do objeto. Valores: <u>inline</u> , block, list-item, run-in, compact, marker, table, inline-table, table-row-group, table-header-group, table-footer-group, table-row, table-column-group, table-column, table-cell, table-caption, none, inherit.
empty-cells	emptyCells	Em tabelas que usam conteúdos separador por células, configura se as células sem conteúdo terão bordas ou não. Valores: <u>show</u> , hide, inherit
filter	filter	Aplica filtros de sombra, iluminação, etc Valores: <u>none</u> blur() brightness() contrast() drop-shadow() grayscale() hue-rotate() invert() opacity() saturate() sepia()
flex	flex	Configura todos os itens flexíveis com a mesma largura Valores: flex-grow flex-shrink flex-basis, auto, initial, inherit
flex-basis	flexBasis	Configura a largura inicial de item flexível Valores: number, <u>auto</u> , initial, inherit
flex-direction	flexDirection	Configura a direção de itens flexíveis Valores: row, row-reverse, column, column-reverse, initial, inherit
flex-flow	flexFlow	Fluxo dos elementos flexíveis Valores: flex-direction, flex-wrap, initial, inherit
flex-grow	flexGrow	Especifica quanto o item irá crescer em relação ao resto dos elementos flexíveis no interior do mesmo container Valores: number, initial, inherit
flex-shrink	flexShrink	Especifica quanto o item irá diminuir em relação ao resto dos elementos flexíveis no interior do mesmo container Valores: number, initial, inherit
flex-wrap	flexWrap	Especifica se os itens flexíveis deve envolver ou não Valores: nowrap, wrap, wrap-reverse, initial, inherit
float	cssFloat	Especifica se o elemento vai flutuar a direita ou a esquerda Valores: left, right, <u>none</u> , initial, inherit
font	font	Metapropriedade. Configura todas as propriedades da fonte. Valores: fontStyle fontVariant fontWeight fontSize lineHeight fontFamily, caption, icon, menu, message-box, small-caption, status-bar, inherit
font-family	fontFamily	Define uma ou mais famílias de fonte. Valores: [fonte1, fonte2], inherit
font-size	fontSize	Tamanho da fonte. Valores: tamanho, porcentagem, xx-small, x-small, small, <u>medium</u> , large, x-large, xx-large, larger, smaller, inherit.
font-style	fontStyle	Estilo da fonte Valores: <u>normal</u> , italic, oblique, initial, inherit
font-variant	fontVariant	Como as letras da fonte vão ser exibidas Valores: <u>normal</u> , small-caps, initial, inherit
font-stretch	fontStretch	Largura dos caracteres da fonte, wider estica e narrower achata a fonte. Valores: normal, wider, narrower, ultra-condensed, extra-condensed, condensed, semi-condensed, semi-expanded, expanded, extra-expanded, ultra-expanded, inherit.
font-weight	fontWeight	Espessura da fonte. Valores: <u>normal</u> , bold, bolder, lighter, 100, 200, 300, 400, 500, 600, 700, 800, 900, inherit
font-stretch	fontStretch	Especifica como será o ajuste da fonte, se irá esticar ou encolher por exemplo Valores: <u>normal</u> , condensed, expanded
height	height	Altura do objeto. Valores: tamanho, porcentagem, <u>auto</u> , inherit.

image-orientation	imageOrientation	Especifica a orientação/direção da imagem
left	left	Semelhante à propriedade bottom, porém, aplicado à esquerda. Valores: tamanho, porcentagem, <u>auto</u> , inherit
letter-spacing	letterSpacing	Distância entre os caracteres do texto. Valores: <u>normal</u> , tamanho, inherit
line-height	lineHeight	Altura da linha. Valores: <u>normal</u> , fator, tamanho, porcentagem, inherit.
list-style	listStyle	Metapropriedade. Define as propriedades listStyleType, listStylePosition e listStyleImage.
list-style-image	listStyleImage	Imagem que será usada como marcador. Valores: url, <u>none</u> , inherit
list-style-position	listStylePosition	Posição do marcador. Valores: inside, <u>outside</u> , inherit
list-style-type	listStyleType	Tipo do marcador. Valores: <u>disc</u> , circle, square, decimal, decimal-leading-zero, lower-roman, upper-roman, lower-greek, llower-alpha, lower-latin, upper-alpha, upper-latin, hebrew, armenian, georgian, cjk-ideographic, hiragana, katakana, hiragana-iroha, katakana-iroha, none, inherit
margin	margin	Largura das quatro margens. Valores: tamanho (exemplo: 1,2,3,4), inherit
margin-bottom	marginBottom	Largura da margem inferior. Valores: tamanho, inherit
margin-left	marginLeft	Largura da margem esquerda. Valores: tamanho, inherit
margin-right	marginRight	Largura da margem direita. Valores: tamanho, inherit
margin-top	marginTop	Largura da margem superior. Valores: tamanho, inherit
max-height	maxHeight	Altura máxima do elemento Valores: none, px, %, initial, inherit
max-width	maxWidth	Largura máxima do elemento Valores: none, px, %, initial, inherit
min-height	minHeight	Altura mínima do elemento Valores: none, px, %, initial, inherit
min-width	minWidth	Largura mínima do elemento Valores: none, px, %, initial, inherit
nav-down	navDown	Define ou retorna para onde navegar ao usar a tecla de navegação seta-para baixo
nav-index	navIndex	Ordem de tabulação de um elemento
nav-left	navLeft	Comportamento ao usar a tecla de navegação de seta-esquerda
nav-right	navRight	Comportamento ao usar a tecla de navegação de seta-direita
nav-up	navUp	Comportamento ao usar a tecla de navegação de seta-para cima
opacity	opacity	Transparência do elemento <div> Valores: number, initial, inherit
order	order	Ordem de itens flexíveis Valores: number, initial, inherit
orphans	orphans	Especifica o número mínimo de linhas que serão mostradas no final da página Valores: number, initial, inherit
outline	outline	Metapropriedade. Define as propriedades de contorno, outlineColor, outlineStyle, outlineWidth, se difere da borda por não ocupar espaço e não precisa ser retangular.
outline-color	outlineColor	Cor do contorno. Valores: cor, <u>invert</u> , inherit
outline-style	outlineStyle	Estilo do contorno.

		Valores: <u>none</u> , hidden, dotted, dashed, solid, double, groove, ridge, inset, outset, inherit
outline-width	outlineWidth	Largura do contorno. Valores: thin, <u>medium</u> , thick, largura, inherit.
overflow	overflow	Define o comportamento visual do objeto caso ele extrapole a área definida pelo bloco pai. Valores: <u>visible</u> , hidden, scroll, auto, inherit
overflow-x	overflowX	Define o comportamento visual do objeto caso ele extrapole a área definida pelo bloco pai pela horizontal Valores: <u>visible</u> , hidden, scroll, auto, initial, inherit
overflow-y	overflowY	Define o comportamento visual do objeto caso ele extrapole a área definida pelo bloco pai pela vertical Valores: <u>visible</u> , hidden, scroll, auto, initial, inherit
padding	padding	Todos os lados do espaçamento interno. Valores: tamanho (ex: 1,2,3,4), porcentagem (ex. 1,2,3,4), inherit
padding-bottom	paddingBottom	Espaçamento interno inferior. Valores: tamanho, porcentagem, inherit.
padding-left	paddingLeft	Espaçamento interno esquerdo. Valores: tamanho, porcentagem, inherit.
padding-right	paddingRight	Espaçamento interno direito. Valores: tamanho, porcentagem, inherit.
padding-top	paddingTop	Espaçamento interno superior. Valores: tamanho, porcentagem, inherit.
page-break-after	pageBreakAfter	Define ou retorna o comportamento page-break depois de um elemento (para impressão ou visualização de impressão). Valores: <u>auto</u> , always, avoid, emptystring, left, right, initial, inherit
page-break-before	pageBreakBefore	Define ou retorna o comportamento page-break antes de um elemento (para impressão ou visualização de impressão). Valores: <u>auto</u> , always, avoid, emptystring, left, right, initial, inherit
page-break-inside	pageBreakInside	Define ou retorna o comportamento page-break dentro de um elemento (para impressão ou visualização de impressão). Valores: <u>auto</u> , avoid, initial, inherit
perspective	perspective	Perspectiva de visualização do elemento Valores: length, <u>none</u>
perspective-origin	perspectiveOrigin	Base 3D inicial da perspectiva Valores: x-axis y-axis, initial, inherit
position	position	Posicionamento do objeto. Valores: <u>static</u> , relative, absolute, fixed, inherit
quotes	quotes	Configura os elementos de marcação Valores: <u>none</u> , string string string string, initial, inherit
resize	resize	Configura um element <div> como reimencionável Valores: none, both, horizontal, vertical, initial, inherit
right	right	Semelhante à propriedade bottom, porém, aplicado à direita. Valores: tamanho, porcentagem, <u>auto</u> , inherit
table-layout	tableLayout	Configura em fixo o tamanho das linhas e colunas da tabela Valores: automatic, fixed, initial, inherit
tab-size	tabSize	Configura o espaçamento do elemento <pre> Valores: number, length, initial, inherit
text-align	textAlign	Alinhamento do texto. Valores: left, right, center, justify, stringAlinhamento, inherit
text-align-last	textAlignLast	Alinha a última linha do parágrafo Valores: auto, left, right, center, justify, start, end, initial, inherit
text-decoration	textDecoration	Decoração do texto. Valores: <u>none</u> , underline, overline, line-through, blink, inherit
text-justify	textJustify	Configura o alinhamento do texto em justificado.
text-indent	textIndent	Indentação da primeira linha do texto. Valores: tamanho, porcentagem, inherit.

text-overflow	textOverflow	Configura o texto excedente ao container Valores: clip, ellipsis, string, initial, inherit
text-shadow	textShadow	Lista de sombras aplicadas ao texto. Valores: <u>none</u> , {cor, [distanciaX & distanciaY & desfoque}, inherit
text-transform	textTransform	Define o fluxo de caracteres maiúsculos e minúsculos do texto. Valores: capitalize, uppercase, lowercase, <u>none</u> , inherit
top	top	Semelhante à propriedade bottom, porém, aplicado ao topo. Valores: tamanho, porcentagem, <u>auto</u> , inherit
transform	transform	Aplica uma transformação como rotação e escala por exemplo a uma <div> Valores: <u>none</u> , transform-functions, initial, inherit
transform-origin	transformOrigin	Ponto de origem, inicial para aplicar a transformação Valores: x-axis, y-axis, z-axis, initial, inherit
transform-style	transformStyle	Define ou retorna como elementos aninhados são renderizados no espaço 3D. Valores: flat, preserve-3d, initial, inherit
transition	transition	Define as configurações da transição. Valores: property duration timing-function delay, initial, inherit
transition-property	transitionProperty	Propriedade de configuração da transição Valores: <u>none</u> , all[property, initial, inherit
transition-duration	transitionDuration	Tempo de duração da transição Valores: time, initial, inherit
transition-timing-function	transitionTimingFunction	Configura curva de velocidade da transição, velocidade do andamento Valores: ease linear, ease-in, ease-out, ease-in-out, cubic-bezier(), initial, inherit
transition-delay	transitionDelay	Tempo para iniciar a transição Valores: time, initial, inherit
vertical-align	verticalAlign	Alinhamento vertical. Valores: baseline, sub, super, top, text-top, middle, bottom, text-bottom, porcentagem, tamanho, inherit.
visibility	visibility	Visibilidade do objeto. Valores: visible, hidden, colapse, <u>inherit</u>
word-break	wordBreak	Configura a regra de quebra de linha do texto Valores: normal, break-all, keep-all, initial, inherit
word-wrap	wordWrap	Configura se uma palavra poderá ser quebrada para próxima linha Valores: normal, break-word, initial, inherit
width	width	Largura do objeto. Valores: tamanho, porcentagem, <u>auto</u> , inherit
word-spacing	wordSpacing	Diatânciá entre as palavras. Valores: <u>normal</u> , tamanho, inherit
z-index	zIndex	Profundidade do objeto, empilhamento, camadas. Valores: <u>auto</u> , profundidade, inherit

Tabela de compatibilidade, CSS x Browser

Propriedade	IE	Firefox	Chrome	Safari	Opera
align-content	11	28	21	9	12.1
align-items	11	20	21	9	12.1
align-self	11	20	21	9	12.1
@keyframes	10	16	43	9	30
animation	10	16	43	9	30
animation-name	10	16	43	9	30
animation-duration	10	16	43	9	30
animation-timing-function	10	16	43	9	30

animation-delay	10	16	43	9	30
animation-iteration-count	10	16	43	9	30
animation-direction	10	16	43	9	30
animation-play-state	10	16	43	9	30
backface-visibility	10	16	36	9	23
background-clip	9	4	4	3	10.5
background-origin	9	4	4	3	10.5
background-size	9	4	4	4.1	10
border-bottom-left-radius	9	4	5	5	10.5
border-bottom-right-radius	9	4	5	5	10.5
border-image	11	15	16	6	15
border-image-outset	11	15	15	6	15
border-image-repeat	11	15	15	6	15
border-image-slice	11	15	15	6	15
border-image-source	11	15	15	6	15
border-image-width	11	13	15	6	15
border-radius	9	4	5	5	10.5
border-top-left-radius	9	4	5	5	10.5
border-top-right-radius	9	4	5	5	10.5
box-decoration-break					
box-shadow	9	4	10	5.1	10.5
box-sizing	8	29	10	5.1	9.5
break-after	10				11.1
break-before	10				11.1
break-inside					
column-count	10	2	4	3	15
column-fill		13			
column-gap	10	2	4	3	15
column-rule	10	2	4	3	15
column-rule-color	10	2	4	3	15
column-rule-style	10	2	4	3	15
column-rule-width	10	2	4		15
column-span	10		4		15
column-width	10	2	4	3	15
columns	10	9	4	3	15
filter		35	18	6	15
fit					
flex	11	28	29	9	17
flex-basis	11	28	29	9	17
flex-direction	11	28	29	9	17
flex-flow	11	28	29	9	17
flex-grow	11	28	29	9	17
flex-shrink	11	28	29	9	17
flex-wrap	11	28	29	9	17
@font-face	9	3.6	4	3	10
@font-feature-values					
font-feature-settings	10	34	16		25
font-kerning					

font-language-override					
font-size-adjust		3			
font-stretch					
font-synthesis					
font-variant					
font-variant-alternates					
font-variant-caps					
font-variant-east-asian					
font-variant-ligatures					
font-variant-numeric					
font-variant-position					
hanging-punctuation					
hyphens	10	6	13	5.1	
image-orientation		26			
image-rendering					11.6
image-resolution					
justify-content	11	28	29	9	17
line-break					
mark					
mark-after					
mark-before					
marks					
marquee-direction					
marquee-play-count					
marquee-speed					
marquee-style					
nav-down					11.5
nav-index					11.5
nav-left					11.5
nav-right					11.5
nav-up					11.5
object-fit					
object-position					
opacity	9	2	4	3.1	9
order	11	28	29	9	17
outline-offset		3.5	4	3	10.5
overflow-wrap	9	3.5		3	9.5
overflow-x	9	3.5	4	3	9.5
overflow-y	9		4	3	9.5
@page	8	19		5	6
perspective	10	16	36	9	23
perspective-origin	10	16	36	9	23
resize		5	4	4	15
rest					
rest-after					
rest-before					
ruby-align					
tab-size		4	21	6.1	15

text-align-last	5.5	12	35		
text-combine-upright					
text-decoration-color		6			
text-decoration-line		6			
text-decoration-style		6			
text-justify	5.5				
text-orientation					
text-overflow	6	7	4	3.1	11
text-shadow	10	3.5	4	4	9.6
text-underline-position					
transform	10	16	36	9	23
transform-origin	10	16	36	9	23
transform-style	11	16	36	9	23
transition	10	16	26	6.1	12.1
transition-delay	10	16	26	6.1	12.1
transition-duration	10	16	26	6.1	12.1
transition-property	10	16	26	6.1	12.1
transition-timing-function	10	16	26	6.1	12.1
word-break	5.5	15	4	3.1	15
word-wrap	5.5	3.5	4	3.1	10.5
writing-mode					

Considerações finais

Neste material você aprendeu sobre CSS, existem inúmeras possibilidade de utilização deste conteúdo, assim como em qualquer linguagem de programação a criatividade é fundamental, pratique bastante, seja ousado e misture todo esse conhecimento.