

CSS

Sumário

CSS INLINE.....	1
VANTAGENS DO CSS INLINE:	1
DESVANTAGENS DO INLINE CSS:	1
CSS INTERNO	1
VANTAGENS DE CSS INTERNO:	1
DESVANTAGENS DE CSS INTERNO:	2
CSS EXTERNO	2
VANTAGENS DE CSS EXTERNO:.....	2
DESVANTAGENS DE CSS EXTERNO:.....	2
SELETORES SIMPLES DE ELEMENTOS CSS	2
O SELETOR DE ELEMENTOS CSS.....	3
O SELETOR DE ID CSS.....	3
O SELETOR DE CLASSE CSS	3
O SELETOR UNIVERSAL CSS	3
O SELETOR DE AGRUPAMENTO CSS.....	3
CORES EM CSS	4
NOMES DE CORES CSS	4
CORES RGB	4
VALOR RGBA	4
CORES HEX (HEXADECIMAL).....	4
CORES HSL.....	5
VALOR HSLA	5
BACKGROUND	5
BACKGROUND-COLOR	5
BACKGROUND-IMAGE	5
BACKGROUND-REPEAT	6
BACKGROUND-POSITION.....	6
BACKGROUND-ATTACHMENT.....	6
BACKGROUND SHORTHAND.....	7
UNIDADE DE MEDIDAS - ABSOLUTAS	7
PIXELS (PX).....	8
POINTS (PT)	8

POLEGADAS IN (INCHES)	8
CENTÍMETRO E MILÍMETRO (CM/MM)	8
PAICA (PC)	8
UNIDADES DE MEDIDAS - RELATIVAS	9
EMS (EM).....	9
REMS (REM, "ROOT EM").....	10
PORCENTAGEM (%).....	10
EX E CH.....	11
VW (VIEWPORT WIDTH)	11
VH (VIEWPORT HEIGHT).....	11
VMIN (VIEWPORT MINIMUM).....	11
VMAX (VIEWPORT MAXIMUM)	12
MARGIN E PADDING	12
MARGIN	12
PADDING.....	12
MARGIN / PADDING SHORTHAND	13
BORDER	14
BORDER-STYLE.....	14
BORDER-WIDTH	15
BORDER-COLOR.....	15
BORDER-RADIUS	15
BOX MODEL	15
OUTLINE	16
OUTLINE-STYLE.....	16
OUTLINE-WIDTH	17
OUTLINE-COLOR	17
OUTLINE-OFFSET	17
AJUSTES DE TEXTO	17
COLOR	17
BACKGROUND-COLOR	17
ALINHAMENTO E DIREÇÃO DE TEXTO	18
TEXT-DECORATION.....	18
TEXT-TRANSFORM.....	19
ESPAÇAMENTO DE TEXTO.....	19
TEXT-SHADOW	19
FONTES	20

FONT-FAMILY	20
FONT-SIZE	21
FONT-STYLE	21
FONT-VARIANT	22
FONT-WEIGHT	22
FONT SHORTHAND	22
FONTES CUSTOMIZADAS	23
FONTES DO GOOGLE	23
@FACE-FONT	23
LINK	24
LISTAS	25
LIST-STYLE-TYPE	25
LIST-STYLE-IMAGE	26
LIST-STYLE-POSITION	26
LIST-STYLE	26
TABELAS	27
PROPRIEDADES DA TABELA CSS	27
DISPLAY	28
ELEMENTOS BLOCK	28
ELEMENTOS INLINE	28
ELEMENTOS INLINE-BLOCK	28
DISPLAY NONE	29
ALTERAR DISPLAY	29
OVERFLOW	30
OVERFLOW: VISIBLE;	30
OVERFLOW: HIDDEN;	30
OVERFLOW: SCROLL;	30
OVERFLOW: AUTO;	30
ALINHAMENTO HORIZONTAL	31
ELEMENTOS BLOCK	31
ELEMENTOS INLINE OU INLINE-BLOCK	31
POSITION	32
POSITION: STATIC;	32
POSITION: RELATIVE;	32
POSITION: ABSOLUTE;	32

POSITION: FIXED;	33
POSITION: STICKY;	33
SOBREPOSIÇÃO Z-INDEX.....	34
FLOAT E CLEAR	34
FLOAT	34
CLEAR.....	34
ALINHAMENTO VERTICAL	35
USANDO PADDING.	35
USANDO LINE-HEIGHT.	35
USANDO TRANSFORM E POSITION	35
VIEWPORT.....	35
CONFIGURANDO VIEWPORT	36
RESPONSIVIDADE	37
MEDIA QUERY	37
SINTAXE.....	38
COMBINADORES CSS	39
DESCENDANT SELECTOR (SPACE).....	39
CHILD SELECTOR (>)	40
ADJACENT SIBLING SELECTOR (+).....	40
GENERAL SIBLING SELECTOR (~)	40
SELETORES DE ATRIBUTOS.....	41
CSS [ATTRIBUTE] SELECTOR	41
CSS [ATTRIBUTE="VALUE"] SELECTOR	41
CSS [ATTRIBUTE~="VALUE"] SELECTOR	42
CSS [ATTRIBUTE = "VALUE"] SELECTOR	42
CSS [ATTRIBUTE^="VALUE"] SELECTOR	42
CSS [ATTRIBUTE\$="VALUE"] SELECTOR	43
CSS [ATTRIBUTE*="VALUE"] SELECTOR.....	43
PSEUDO-ELEMENTOS CSS	43
PSEUDO-ELEMENTOS REFERÊNCIA	44
PSEUDO-CLASSES CSS.....	44
PSEUDO-CLASSES REFERÊNCIA.....	45
FUNÇÃO CALC(_).....	46

TRANSFORMAÇÕES	46
MÉTODO TRANSLATE()	46
MÉTODO ROTATE()	47
MÉTODO SCALE()	47
MÉTODO SKEW()	47
MÉTODO MATRIX()	48
REFERÊNCIA PROPRIEDADES DE TRANSFORMAÇÃO	48
REFERÊNCIA MÉTODOS DE TRANSFORMAÇÃO	48
COMPATIBILIDADE DE NAVEGADORES	49
VERIFICAR SUPORTE DOS NAVEGADORES	49
VARIÁVEIS CSS	49
VARIÁVEL DE ESCOPO GLOBAL	49
VARIÁVEL DE ESCOPO LOCAL	50
ALTERAR VARIÁVEIS CSS COM JAVASCRIPT	51
FLEXBOX	52
FLEX-DIRECTION	52
FLEX-WRAP	53
FLEX-FLOW	54
JUSTIFY-CONTENT	54
ALIGN-ITEMS	55
ALIGN-CONTENT	56
CENTRALIZAÇÃO PERFEITA COM FLEXBOX	58
FLEX-ITEMS	58
FLEX-GROW	58
FLEX-SHRINK	59
ORDER	59
ALIGN-SELF	59
CSS GRID	60
DISPLAY GRID	60
GRID COLUMNS	60
GRID ROWS	60
GRID GAP	61
GRID LINES	61
GRID-TEMPLATE	62
ALINHAMENTO CSS GRID	63
JUSTIFY-CONTENT	63
ALIGN-CONTENT	64

PLACE-CONTENT	65
JUSTIFY-ITEMS	65
ALIGN-ITEMS	66
PLACE-ITEMS	67
JUSTIFY-SELF	67
ALIGN-SELF	67
PLACE-SELF	68
POSICIONAMENTO CSS GRID	68
GRID-COLUMN	68
GRID-ROW	69
GRID-AREA	69
GRID-TEMPLATE-AREAS	69
TIPOS DE ALINHAMENTOS CSS	70
CSS ANIMATIONS	71
@KEYFRAMES	71
ANIMATION-DELAY	72
ANIMATION-INTERATION-COUNT	72
ANIMATION-DIRECTION	72
ANIMATION-FILL-MODE	73
ANIMATION-TIMING-FUNCION	73
ANIMATION SHORTHAND	74
CSS TRANSITIONS	74
TRANSITION-DELAY	75
TRANSITION-TIMING-FUNCTION	75
TRANSITION SHORTHAND	75

CSS INLINE

O CSS inline é usado para dar estilo a um elemento HTML específico. Para este estilo de CSS você somente vai precisar adicionar o atributo style para cada tag HTML, sem usar os seletores. Aqui, nós adicionamos um CSS inline para as tags <p> e <h1>:

```
<h1 style="color:white; padding:30px;"> Título Exemplo </h1>
```

```
<p style="color:white;">Parágrafo exemplo</p>
```

Vantagens do CSS Inline:

- É possível inserir elementos CSS de maneira fácil e rápida numa página HTML.
- É por isso que esse método é útil para testar e pré-visualizar mudanças, assim como executar correções rápidas no seu site. Você não precisa criar e fazer upload de um documento separado como no estilo externo.

Desvantagens do Inline CSS:

- Adicionar regras CSS para cada elemento HTML consome muito tempo e faz a sua estrutura HTML ficar bagunçada.
- Estilizar múltiplos elementos pode afetar o tamanho da sua página o tempo para download.

CSS INTERNO

O CSS interno ou incorporado requer que você adicione a tags <style> na seção <head> do seu documento HTML. Este estilo de CSS é um método efetivo de estilizar uma única página. <style> body { background-color: blue; } h1 { color: red; padding : 60px; } </style> </head>.

Vantagens de CSS Interno:

- Classes e seletores de IDs podem ser usados por stylesheet interno. Confira um exemplo abaixo:
- Não há necessidade de carregar vários arquivos. HTML e CSS podem estar no mesmo arquivo.

Desvantagens de CSS Interno:

- Adicionar o código para o documento HTML pode aumentar o tamanho da página e o tempo de carregamento.

CSS EXTERNO

O CSS externo vai linkar as páginas da internet com um arquivo “.css” externo. Este tipo de CSS é um método eficiente quando se está estilizando um site grande. Ao editar um arquivo .css, você pode modificar um site inteiro de uma só vez. O CSS externo acontece quando é criado um novo arquivo .css com um editor de texto e então adicione regras de estilo, logo após isso na seção <head> da planilha HTML, é adicionada uma referência para o arquivo .css logo depois da tag <title>: <link rel="stylesheet" type="text/css" href="style.css" /> (style.css é o nome do arquivo de estilo css, caso seja outro, o mesmo deve ser alterado).

Vantagens de CSS Externo:

- Como o código CSS está num documento separado, os seus arquivos HTML terão uma estrutura mais limpa e serão menores
- O mesmo arquivo .css pode ser usado em várias páginas.

Desvantagens de CSS Externo:

- Até que o CSS externo seja carregado, a página pode não ser processada corretamente.
- Fazer o upload ou links para múltiplos arquivos CSS pode aumentar o tempo de download do seu site.

SELETORES SIMPLES DE ELEMENTOS CSS

Seletores CSS são usados para "encontrar" (ou selecionar) os elementos HTML que você deseja estilizar. Podemos dividir os seletores CSS em cinco categorias: **Seletores simples** (selecione elementos com base no nome, id, classe); **Seletores de combinação** (selecione elementos com base em um relacionamento específico entre eles); **Seletores de pseudo-classe** (selecionar elementos com base em um determinado estado); **Seletores de pseudo-**

elementos (selecione e estilize uma parte de um elemento); **Seletores de atributo** (selecione elementos com base em um atributo ou valor de atributo). Os seletores simples estudados nesse tópico são os mais básicos dos seletores.

O seletor de elementos CSS

O seletor de elemento seleciona elementos HTML com base no tipo do elemento. Aqui, todos os elementos <p> na página serão alinhados ao centro, com uma cor de texto vermelha: `p { text-align: center; color: red; }`

O seletor de id CSS

O seletor id usa o atributo id de um elemento HTML para selecionar um elemento específico. O id de um elemento é único dentro de uma página, então o seletor de id é usado para selecionar um elemento único. Para selecionar um elemento com um id específico, escreva um caractere hashtag (#), seguido do id do elemento. A regra CSS abaixo será aplicada ao elemento HTML com id="para1": `#para1 { text-align: center; color: red }.`

O seletor de classe CSS

O seletor de classe seleciona elementos HTML com um atributo de classe específico. Para selecionar elementos com uma classe específica, escreva um caractere ponto (.), seguido do nome da classe. Neste exemplo, todos os elementos HTML com class="center" serão vermelhos e alinhados ao centro: `center {text-align: center; color: red;}`

O seletor universal CSS

O seletor universal (*) seleciona todos os elementos HTML na página. A regra CSS abaixo afetará todos os elementos HTML na página: `* { text-align: center; color: blue;}`

O seletor de agrupamento CSS

O seletor de agrupamento seleciona vários elementos HTML e aplica as mesmas definições de estilo. Veja o seguinte código CSS (os elementos h1, h2 e p têm as mesmas definições de estilo): Será melhor agrupar os seletores, para minimizar o código. Para agrupar seletores, separe cada seletor com uma vírgula: `h1, h2, p { text-align: center; color: red;}`

CORES EM CSS

As cores em CSS são especificadas usando nomes de cores predefinidos ou valores RGB, HEX, HSL, RGBA, HSLA.

Nomes de cores CSS

Em CSS, uma cor pode ser especificada usando um nome de cor predefinido. É possível encontrar esses nomes no site https://www.w3schools.com/colors/colors_names.asp

Ex: `<h1 style="background-color:DodgerBlue;">Hello World</h1>` **HELLO WORLD**

Cores RGB

Um valor de cor RGB representa fontes de luz VERMELHA (red), VERDE (green) e AZUL (blue). Em CSS, uma cor pode ser especificada como um valor RGB, usando esta fórmula: **rgb (vermelho, verde, azul)**. Cada parâmetro (vermelho, verde e azul) define a intensidade da cor entre 0 e 255.

Ex: `<h1 style="background-color: rgb(0, 0, 255);">Hello World</h1>` **HELLO WORLD**

Valor RGBA

Os valores de cor RGBA são uma extensão dos valores de cor RGB com um canal alfa - que especifica a opacidade de uma cor. Um valor de cor RGBA é especificado com: **rgba(vermelho, verde, azul, alfa)**. O parâmetro alfa é um número entre 0,0 (totalmente transparente) e 1,0 (nada transparente):

Cores HEX (Hexadecimal)

Uma cor hexadecimal é especificada com: #RRGGBB, onde os inteiros hexadecimais RR (vermelho), GG (verde) e BB (azul) especificam os componentes da cor. Em CSS, uma cor pode ser especificada usando um valor hexadecimal na forma: **#rrggbb**, onde rr (vermelho), gg (verde) e bb (azul) são valores hexadecimais entre 00 e ff (o mesmo que decimal 0-255). Por exemplo, #ff0000 é exibido como vermelho, porque o vermelho é definido com o valor mais alto (ff) e os outros são definidos com o valor mais baixo (00). Para exibir preto, defina todos os valores para 00, assim: #000000. Para exibir branco, defina todos

os valores para ff, assim: #ffffff. Ex `<h1 style="background-color:#0000ff;">HELLO WORLD</h1>` **HELLO WORLD.**

Cores HSL

HSL significa tonalidade(hue), saturação(saturation), luminosidade(lightness). Em CSS, uma cor pode ser especificada usando (HSL) na forma: **hsl (tonalidade, saturação, luminosidade)**. Tonalidade é um grau na roda de cores de 0 a 360. 0 é vermelho, 120 é verde e 240 é azul. A saturação é um valor percentual, 0% significa um tom de cinza e 100% é a cor completa. A luminosidade também é uma porcentagem, 0% é preto, 50% não é claro nem escuro, 100% é branco. EX `<h1 style="background-color:hsl(240, 100%, 50%);">HELLO WORLD</h1>` **HELLO WORLD.**

Valor HSLA

Os valores de cor HSLA são uma extensão dos valores de cor HSL com um canal alfa - que especifica a opacidade de uma cor. Um valor de cor HSLA é especificado com: **hsla (tonalidade, saturação, luminosidade, alfa)**. O parâmetro alfa é um número entre 0,0 (totalmente transparente) e 1,0 (nada transparente):

BACKGROUND

As propriedades de background CSS são usadas para adicionar efeitos de plano de fundo para elementos.

background-color

A propriedade background-color especifica a cor de fundo de um elemento. Com CSS, uma cor é mais frequentemente especificada por: um nome de cor válido - como "red"; um valor HEX - como "#ff0000"; um valor RGB - como "rgb(255,0,0)". Ex: `body {background-color: lightblue;}`.

background-image

A propriedade background-image especifica uma imagem a ser usada como plano de fundo de um elemento. Por padrão, a imagem é repetida para cobrir todo o elemento.

background-repeat

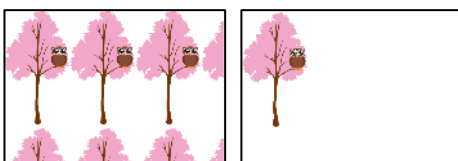
Por padrão, a propriedade background-image repete uma imagem na horizontal e na vertical, em alguns casos essa repetição fica desproporcional como no exemplo: `body { background-image: url("gradient_bg.png"); }`



Se a imagem acima for repetida apenas horizontalmente (`background-repeat: repeat-x;`), o plano de fundo ficará melhor: `body { background-image: url("gradient_bg.png"); background-repeat: repeat-x; }`



Para mostrar a imagem de fundo apenas uma vez também é especificado pela propriedade background-repeat (`no-repeat`): `body { background-image: url("img_tree.png"); background-repeat: no-repeat; }`



background-position

A propriedade background-position é usada para especificar a posição da imagem de fundo. Exemplo: Posicione a imagem de fundo no canto superior direito: `body { background-image: url("img_tree.png"); background-repeat: no-repeat; background-position: right top; }`



background-attachment

A propriedade background-attachment especifica se a imagem de fundo deve rolar ou ser fixa (não rolará com o resto da página).

background shorthand

Para encurtar o código, também é possível especificar todas as propriedades do plano de fundo em uma única propriedade. Isso é chamado de propriedade abreviada (shorthand). Para fazer isso deve-se usar a propriedade abreviada `background`. Exemplo: Use a propriedade abreviada para definir as propriedades do plano de fundo em uma declaração: `body { background: #ffffff url("img_tree.png") no-repeat right top;}`

Ao usar a propriedade abreviada, a ordem dos valores da propriedade é:

1. `background-color`
2. `background-image`
3. `background-repeat`
4. `background-attachment`
5. `background-position`

Não importa se um dos valores da propriedade está faltando, desde que os outros estejam nesta ordem.

UNIDADE DE MEDIDAS - ABSOLUTAS

São medidas que não estão referenciadas a qualquer outra unidade, ou seja, não dependem de um valor de referência. São unidades de medidas definidas pela física, como o pixel, centímetro, metro, etc. Essas medidas são fixas e não mudam de acordo com as especificações do dispositivo. Esse tipo de medida é indicada para quando conhecemos perfeitamente as características físicas e as configurações das mídias onde serão exibidos nossos projetos.

unit	name	equivalence
<i>'cm'</i>	centimeters	1cm = 96px/2.54
<i>'mm'</i>	millimeters	1mm = 1/10th of 1cm
<i>'Q'</i>	quarter-millimeters	1Q = 1/40th of 1cm
<i>'in'</i>	inches	1in = 2.54cm = 96px
<i>'pc'</i>	picas	1pc = 1/6th of 1in
<i>'pt'</i>	points	1pt = 1/72th of 1in
<i>'px'</i>	pixels	1px = 1/96th of 1in

Pixels (px)

Essa é uma medida bastante famosa para os web designers, grande parte dos desenvolvedores web usam o pixel como unidade principal de seus projetos. A definição de pixel de referência no CSS é o ângulo visual(0.0213deg) de um pixel em um dispositivo com a densidade de 96dpi a uma distância de um braço do leitor (28 polegadas).

Points (pt)

Essa medida é geralmente utilizada em propriedades relacionadas a fonte do seu projeto. Sua abreviação se dá com a marcação de pt e seu uso não é tão comum. Geralmente espera-se que essa medida seja utilizada em folhas de estilo para impressões, quando se precisa ter certeza do tamanho da fonte utilizada. Não é recomendada para a estilização em tela!

Polegadas in (inches)

Polegada ou inch em inglês é mais uma unidade de medida que conhecemos do mundo das medidas absolutas - geralmente vemos elas quando queremos comprar uma nova TV ou monitor, mas essa unidade também existe no mundo Web.

Centímetro e Milímetro (cm/mm)

Nós brasileiros, que adotamos o sistema métrico, conhecemos bem essas duas medidas, que são bastante utilizadas no dia a dia. Apesar de bastante comuns, tanto centímetro e milímetro são pouco usadas no CSS. Assim como o pt, o uso dessas duas é esperado para folhas de estilo para impressões (medidas mais precisas), evitando que elas sejam aplicadas para exibições em tela.

Paica (pc)

Também uma unidade pouco usada no mundo web, a Paica também vem para o CSS sendo herdada da tipografia. Por não ser uma unidade amplamente conhecida, ela acaba sendo fadada ao esquecimento, mas é sempre importante conhecermos todas as ferramentas que estão à nossa disposição. A relação entre as unidades absolutas é:

UNIDADES DE MEDIDAS - RELATIVAS

Essas medidas são calculadas tendo como base uma outra unidade de medida definida, como por exemplo “em” e o “rem”. O uso delas é mais apropriado para que possamos fazer ajustes em diferentes dispositivos garantindo um layout consistente e fluido em diversas mídias. Devido ao fato de que essas medidas são calculadas pelo browser baseando-se em outra unidade, elas tendem a ser bastante flexíveis. Ou seja, podemos ter resultados diferentes de acordo com o ambiente.

ems (em)

Uma técnica bastante utilizada consiste justamente em fazer uso desse poder do “em” flexibilizando nossos elementos. A ideia é que a alteração do tamanho da fonte do elemento pai faça com que todo o componente se modifique e redimensione baseando-se nesse novo valor. Quando expressamos tamanhos como margin, padding utilizando “em”, isso significa que eles serão relativos ao tamanho da fonte do elemento pai. Portanto, de acordo com o tamanho da fonte utilizada em determinado elemento, os elementos filhos serão redimensionados de forma a obedecer a referência a esse tamanho de fonte!

```
<style>
  #div{
    font-size: 16px;
  }

  #filho{
    font-size: 2em;
  }
</style>

<div id="pai">
  div pai
  <div id="filho">
    div filho
  </div>
</div>
```

Nesse exemplo o elemento filho terá um tamanho de 32px que é 2em do tamanho do elemento pai, ou seja, 16px *2.

Essa é uma unidade de medida utilizada para facilitar a manutenção do componente como um todo, sem ter que sofrer alterando valores de todas as partes do componente, como por exemplo pra ícones e elementos que precisam escalar junto com o texto, como botões e inputs.

Porém um ponto que se deve atentar ao usar o “em”, é que quando usamos essa medida, nós temos que considerar o font-size de todos os elementos pai. Isso tende a se complicar quando estamos falando de 5, 6, 7 divs aninhadas, provavelmente não será muito divertido calcular isso. O lado bom é que a unidade de medida rem resolve esse problema.

rems (rem, "root em")

O REM vem como sucessor do EM e ambos compartilham a mesma lógica de funcionamento, porém a forma de implementação é diferente. Enquanto o em está diretamente relacionado ao tamanho da fonte do elemento pai, o rem está relacionado com o tamanho da fonte do elemento root (raiz), no caso, a tag. O fato de que o rem se relaciona com o elemento raiz resolve aquele problema que tínhamos com diversas divs (elementos) aninhados, uma vez que não haverá essa "herança" de tamanhos, lembra?! Ou seja, não precisaremos ter dor de cabeça tendo que realizar cálculos, uma vez que nos baseamos na tag raiz(root), ou seja, o arquivo html.

```
html{
  font-size: 62,5%;
}

h1{
  font-size: 1.2rem; /*equivalente a 12px*/
}

p{
  font-size: 2.4rem; /*equivalente a 24px*/
}
```

Normalmente os browsers especificam o tamanho default da fonte do elemento root (raiz) sendo 16px, nesse exemplo foi definido o tamanho da fonte para 62,5% desse valor, ou seja, 10px. Logo após foi definido o tamanho da fonte do elemento h1 para 1.2rem ou seja 120% do valor do elemento raiz, 12px. Outro exemplo também foi definido o elemento p para equivaler a 2.4rem, ou seja, 24px.

Porcentagem (%)

Apesar de não ser uma unidade de medida, a porcentagem costuma ser bastante utilizada quando falamos de layout responsivo e fluido, por isso, não poderia deixá-la passar. A porcentagem permite que criemos módulos que sempre vão se readaptar para ocupar a quantidade especificada. Por exemplo,

se definirmos um elemento tendo um tamanho de 50%, independente do dispositivo em questão, esse módulo sempre ocupará metade do espaço que lhe cabe (caso esteja dentro de algum outro elemento).

A porcentagem tem um comportamento parecido com o “em”, já que ele se relaciona diretamente com o tamanho da propriedade do elemento pai. Portanto, ao trabalharmos com a porcentagem, temos o mesmo problema que tínhamos com o em, quanto mais elementos aninhados, mais complicado será de definirmos exatamente o tamanho, por isso, deve-se ter cuidado ao utiliza-la.

ex e ch

Diferentemente da forma como a EM e a REM funcionavam, essa unidade não se relaciona com o tamanho da fonte (font-size), mas com qual fonte está sendo utilizada naquele momento (font-family), mais especificamente ao tamanho do caractere x dessa fonte em questão (x-height). Esse valor pode vir diretamente com a fonte, o browser pode medir o caractere em caixa baixa (lower case) e se esses dois não funcionarem, o browser estipula um valor de 0.5em para 1ex. Diferente do “ex” mas também relacionado ao estilo da fonte, o ch (character unit) é definida na documentação como sendo a "medida avançada" da largura do caractere zero ("0").

vw (viewport width)

A unidade vw se relaciona diretamente com a largura da viewport, onde 1vw representa 1% do tamanho da largura dessa área visível. A diferença entre vw e a % é bem semelhante a diferença entre em e rem, onde a % é relativa ao contexto local do elemento e o vw é relativo ao tamanho total da largura da viewport do usuário.

Vh (viewport height)

Essa unidade funciona da mesma forma que o vw, porém dessa vez, a referência será a altura e não a largura

Vmin (viewport minimum)

Essa unidade também se relaciona com as dimensões da viewport, porém, diferente das anteriores, o vmin utilizará como base a menor dimensão da viewport (altura x largura). Imagine que estamos trabalhando com uma

viewport de 1600px de altura e 900px de largura. Nesse caso, 1vmin terá o valor de 9px (1% da menor dimensão!), caso tenhamos 100vmin, esse será igual a 900px.

Vmax (viewport maximum)

Seguindo a mesma base lógica da unidade anterior, o vmax terá como valor de referência a maior dimensão da viewport. Ou seja, utilizando o mesmo exemplo, se tivermos 1600px de altura e 900px de largura, 1vmax será equivalente a 16px.

MARGIN E PADDING

Margin

A propriedade CSS `margin` é usada para criar espaço ao redor dos elementos, **fora** de quaisquer bordas definidas. Com CSS, você tem controle total sobre as margens. Existem propriedades para definir a margem para cada lado de um elemento (superior, direito, inferior e esquerdo).

- `margin-top`
- `margin-right`
- `margin-bottom`
- `margin-left`

Todas as propriedades de margem podem ter os seguintes valores:

- `auto` - o navegador calcula a margem
- `length` - especifica uma margem em px, pt, cm, etc.
- `%` - especifica uma margem em % da largura do elemento que o contém
- `inherit` - especifica que a margem deve ser herdada do elemento pai

Padding

O padding (preenchimento) é usado para criar espaço ao redor do conteúdo de um elemento, **dentro** de qualquer borda definida. Assim como margin o padding possui propriedades para definir cada lado do elemento (superior, direito, inferior e esquerdo).

- padding-top
- padding-right
- padding-bottom
- padding-left

Todas as propriedades de preenchimento podem ter os seguintes valores:

- length - especifica um preenchimento em px, pt, cm, etc.
- % - especifica um preenchimento em % da largura do elemento que o contém
- inherit - especifica que o preenchimento deve ser herdado do elemento pai

Margin / Padding shorthand

Tanto o Margin quanto o Padding aceitam o formato abreviado, onde você pode indicar mais de um valor na mesma linha.

Exemplos:

- Com 4 valores:
 - margin: top right bottom left;
 - margin: 25px 30px 35px 40px;
- Com apenas 3 valores:
 - padding: top right|left bottom
 - padding: 25px 50px 20px (25px de margem para cima, 50px tanto para direita quanto para esquerda e 20px para baixo)
- Com dois valores:
 - margin: top|bottom right|left
 - margin: 25px 50px (25px de margem para cima e para baixo, 50px tanto para direita quanto para esquerda)
- Com apenas 1 valor:
 - padding: top|bottom|right|left
 - padding: 25px; (25px de margem para todos os lados);

BORDER

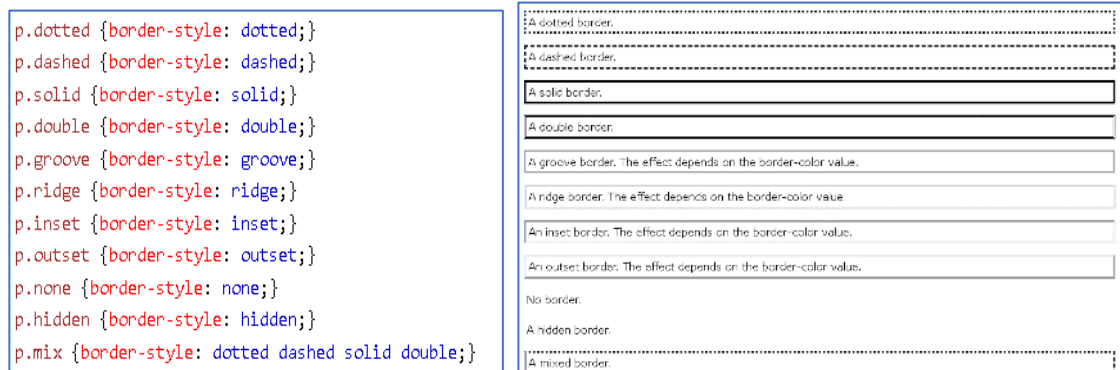
As propriedades da borda CSS permitem que você especifique o estilo, a largura e a cor da borda de um elemento.

border-style

A propriedade border-style especifica que tipo de borda exibir. Os seguintes valores são permitidos:

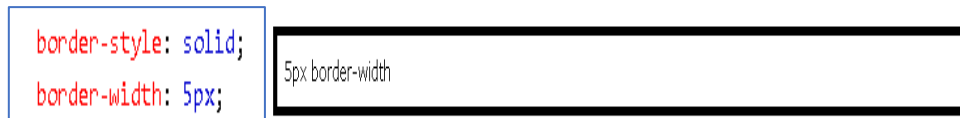
- dotted- Define uma borda pontilhada
- dashed- Define uma borda tracejada
- solid- Define uma borda sólida
- double- Define uma borda dupla
- groove- Define uma borda ranhurada 3D. O efeito depende do valor da cor da borda
- ridge- Define uma borda 3D estriada. O efeito depende do valor da cor da borda
- inset- Define uma borda de inserção 3D. O efeito depende do valor da cor da borda
- outset- Define uma borda inicial 3D. O efeito depende do valor da cor da borda
- none- Não define borda
- hidden- Define uma borda oculta

A propriedade border-style pode ter de um a quatro valores (para a borda superior, borda direita, borda inferior e borda esquerda).



border-width

A propriedade `border-width` especifica a largura das quatro bordas. A largura pode ser definida como um tamanho específico (em px, pt, cm, em, etc) ou usando um dos três valores predefinidos: `thin`, `medium`, ou `thick` (fino, médio ou grosso).



border-color

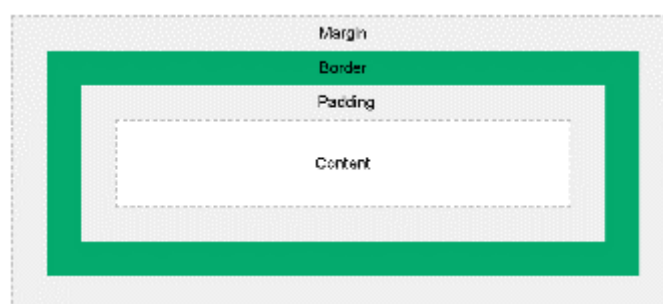
A propriedade `border-color` é usada para definir a cor das quatro bordas.

border-radius

A propriedade `border-radius` é usada para adicionar bordas arredondadas a um elemento

BOX MODEL

Todos os elementos HTML podem ser considerados como caixas. Em CSS, o termo “box model” (modelo de caixa) é usado quando se fala de design e layout. O modelo de caixa CSS é essencialmente uma caixa que envolve cada elemento HTML. Ele consiste em: margens, bordas, preenchimento e o conteúdo real. A imagem abaixo ilustra o modelo da caixa:



Explicação das diferentes partes:

- Content - O conteúdo da caixa, onde o texto e as imagens aparecem
- Padding - Limpa uma área ao redor do conteúdo. O forro é transparente
- Border - Uma borda que contorna o preenchimento e o conteúdo

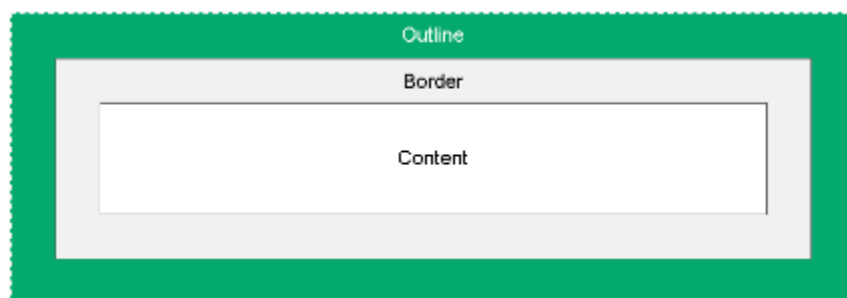
- Margin - Limpa uma área fora da fronteira. A margem é transparente

O modelo de caixa nos permite adicionar uma borda ao redor dos elementos e definir o espaço entre os elementos.

Para definir a largura e a altura de um elemento corretamente em todos os navegadores, você precisa saber como funciona o modelo de caixa. Ao definir as propriedades de largura e altura de um elemento com CSS, basta definir a largura e a altura da área de conteúdo. Para calcular o tamanho total de um elemento, você também deve adicionar preenchimento, bordas e margens. Ex: 320px (largura) + 20px (preenchimento esquerdo + direito) + 10px (borda esquerda + direita) + 0px (margem esquerda + direita) = largura total (width) 350px.

OUTLINE

A propriedade outline é um contorno, uma linha desenhada fora da borda do elemento. A outline é diferente das bordas. Ao contrário da borda, a outline é desenhado fora da borda do elemento e pode sobrepor outro conteúdo. Além disso, a outline não faz parte das dimensões do elemento; a largura e a altura totais do elemento não são afetadas pela largura da outline.



outline-style

A propriedade outline-style especifica o estilo do contorno e pode ter um dos seguintes valores:

- dotted- Define um contorno pontilhado
- dashed- Define um contorno tracejado
- solid- Define um contorno sólido
- double- Define um contorno duplo
- groove- Define um contorno ranhurado 3D

- ridge- Define um contorno 3D estriado
- inset- Define um contorno de inserção 3D
- outset- Define um esboço inicial 3D
- none- Não define contorno
- hidden- Define um contorno oculto

Nenhuma das outras propriedades de contorno terá qualquer efeito a menos que a propriedade `outline-style` seja definida.

outline-width

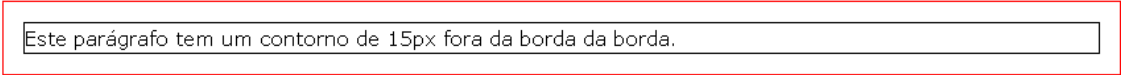
A propriedade `outline-width` especifica a largura do contorno.

outline-color

A propriedade `outline-color` é usada para definir a cor do contorno.

outline-offset

A propriedade `outline-offset` adiciona espaço entre um contorno e a borda/borda de um elemento. O espaço entre um elemento e seu contorno é transparente. O exemplo a seguir especifica um contorno de 15px fora da borda da borda:



Este parágrafo tem um contorno de 15px fora da borda da borda.

AJUSTES DE TEXTO

color

A propriedade `color` é usada para definir a cor do texto.

```
h1 {
  color: green;
}
```



This is heading 1

background-color

A propriedade `background-color` define o fundo do texto. O alto contraste é muito importante para pessoas com problemas de visão. Portanto, certifique-se sempre de que o contraste entre a cor do texto e a cor de fundo (ou imagem de fundo) seja bom.

```
h1 {
  background-color: black;
  color: white;
}
```

This is a Heading

Alinhamento e direção de texto

As propriedades que alteram o direcionamento e o alinhamento de texto são:

- **text-align** - Especifica o alinhamento horizontal do texto (center, left, right, justify)
- **text-align-last** - Especifica como alinhar apenas a última linha de um texto
- **vertical-align** - Define o alinhamento vertical de um elemento
- **direction** - Especifica a direção do texto/direção de escrita (da esquerda para direita ou direita para esquerda)
- **unicode-bidi** - Usado junto com a propriedade direction para definir ou retornar se o texto deve ser substituído para suportar vários idiomas no mesmo documento.

text-decoration

A propriedade text-decoration-line é usada para adicionar uma linha de decoração ao texto.

```
h1 {
  text-decoration: overline;
}
h2 {
  text-decoration: line-through;
}
h3 {
  text-decoration: underline;
}
p.ex {
  text-decoration: overline underline;
}
```

Overline text decoration

~~Line-through text decoration~~

Underline text decoration

Overline and underline text decoration

As propriedades de decoração de texto são:

- **text-decoration** - Define todas as propriedades de decoração de texto em uma declaração
- **text-decoration-color** - Especifica a cor da decoração de texto

- **text-decoration-line** - Especifica o tipo de decoração de texto a ser usado (sublinhado, overline, etc.)
- **text-decoration-style** - Especifica o estilo da decoração do texto (solid, dotted, wave, etc.)
- **text-decoration-thickness** - Especifica a espessura da linha de decoração de texto

text-transform

A propriedade text-transform é usada para especificar letras maiúsculas e minúsculas em um texto. Ele pode ser usado para transformar tudo em letras maiúsculas ou minúsculas, ou colocar em maiúscula a primeira letra de cada palavra: através dos valores uppercase, lowercase e capitalize.

Espaçamento de texto

Existem propriedades em CSS que tornam possível alterar o recuo de texto, espaçamento entre letras, altura da linha, espaçamento entre palavras e espaço em branco. Essas propriedades são:

- **text-indent** - Especifica o recuo da primeira linha em um bloco de texto
- **letter-spacing** - Especifica o espaço entre os caracteres em um texto
- **line-height** - Especifica a altura da linha
- **word-spacing** - Especifica o espaço entre as palavras em um texto
- **white-space** - Especifica como lidar com o espaço em branco dentro de um elemento

text-shadow

A text-shadow propriedade adiciona sombra ao texto. Ela aceita uma lista de sombras separadas por vírgula que serão aplicados ao texto e ao text-decoration do elemento. Cada sombra é especificada como um deslocamento do texto, juntamente com valores opcionais de cor e raio de desfoque. Múltiplas sombras são aplicadas de frente para trás, com a primeira sombra especificada no topo.

```
h1 {
  text-shadow: 2px 2px 5px red;
}
```

Text-shadow effect!

```
h1 {
  color: white;
  text-shadow: 1px 1px 2px black, 0 0 25px blue, 0 0 5px darkblue;
}
```

Text-shadow effect!

FONTES

Escolher a fonte certa tem um enorme impacto em como os leitores experimentam um site. A fonte certa pode criar uma identidade forte para sua marca. Usar uma fonte que seja fácil de ler é importante. A fonte agrega valor ao seu texto. Também é importante escolher a cor e o tamanho do texto corretos para a fonte.

font-family

Especifica a família da fonte para o texto. A propriedade font-family deve conter vários nomes de fontes como um sistema "fallback", para garantir a máxima compatibilidade entre navegadores/sistemas operacionais. Comece com a fonte desejada e termine com uma família genérica (para permitir que o navegador escolha uma fonte semelhante na família genérica, se não houver outras fontes disponíveis). Os nomes das fontes devem ser separados por vírgula. Se o nome da fonte for mais de uma palavra, deve estar entre aspas, como: "Times New Roman". Em CSS existem cinco famílias de fontes genéricas:

- **Serif** - As fontes com serifa têm um pequeno traço nas bordas de cada letra.
- **Sans-serif** - As fontes sem serifa têm linhas limpas (sem pequenos traços anexados).
- **Monospace** - Fontes monoespaçadas todas as letras têm a mesma largura fixa. Eles criam uma aparência mecânica.
- **Cursive** - As fontes cursivas imitam a caligrafia humana.
- **Fantasy** - Fontes de fantasia são fontes decorativas/lúdicas.

font-size

Especifica o tamanho da fonte do texto. Ser capaz de gerenciar o tamanho do texto é importante no design da web. No entanto, você não deve usar ajustes de tamanho de fonte para fazer os parágrafos parecerem títulos ou os títulos parecerem parágrafos. O valor do tamanho da fonte pode ser um tamanho absoluto ou relativo.

- Tamanho absoluto: Define o texto para um tamanho especificado. Não permite que um usuário altere o tamanho do texto em todos os navegadores (ruim por motivos de acessibilidade). O tamanho absoluto é útil quando o tamanho físico da saída é conhecido.
- Tamanho relativo: Define o tamanho em relação aos elementos circundantes. Permite que um usuário altere o tamanho do texto nos navegadores.

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

h2 {
  font-size: 1.875em; /* 30px/16=1.875em */
}

p {
  font-size: 0.875em; /* 14px/16=0.875em */
}
```

This is heading 1

This is heading 2

This is a paragraph.

font-style

Especifica o estilo da fonte para o texto. É usada principalmente para especificar texto em itálico. Esta propriedade tem três valores:

- normal - O texto é mostrado normalmente
- italic - O texto é mostrado em itálico
- oblique - O texto está "inclinado" (o oblíquo é muito semelhante ao itálico, mas menos suportado)

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

p.italic {
  font-style: italic;
}

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

This is a paragraph in normal style.

This is a paragraph in italic style.

This is a paragraph in oblique style.

font-variant

Especifica se um texto deve ou não ser exibido em um fontes small-caps. Em uma fonte de letras minúsculas, todas as letras minúsculas são convertidas em letras maiúsculas. No entanto, as letras maiúsculas convertidas aparecem em um tamanho de fonte menor do que as letras maiúsculas originais no texto.

<pre>p.normal { font-variant: normal; } p.small { font-variant: small-caps; }</pre>	<p>My name is Hege Refsnes.</p> <p>My NAME IS HEGE REFSNES.</p>
--	---

font-weight

Especifica o peso de uma fonte

<pre>p.normal { font-weight: normal; } p.light { font-weight: lighter; } p.thick { font-weight: bold; } p.thicker { font-weight: 900; }</pre>	<p>This is a paragraph.</p> <p>This is a paragraph.</p> <p>This is a paragraph.</p> <p>This is a paragraph.</p>
--	---

font shorthand

As propriedades de fonte podem ser alteradas de maneira abreviada para definir todas as propriedades da fonte em uma única declaração. Os valores font-size e font-family são obrigatórios. Se um dos outros valores estiver ausente, seu valor padrão será usado.

```
p.a {
  font: 20px Arial, sans-serif;
}

p.b {
  font: italic bold 12px/30px Georgia, serif;
}
```

Este é um parágrafo. O tamanho da fonte é definido como 20 pixels e a família de fontes é Arial.

Este é um parágrafo. A fonte é definida como itálico e negrito, o tamanho da fonte é definido como 12 pixels, a altura da linha é definida como 30 pixels e a família de fontes é Georgia

FONTES CUSTOMIZADAS

Se você não quiser usar nenhuma das fontes padrão em HTML, poderá usar o Google Fonts, ou baixar um arquivo de fonte separado (formato TTF/OTF recomendado).

Fontes do Google

As fontes do Google são gratuitas e têm mais de 1.000 fontes para escolher. Para usar as fontes do Google basta adicionar um link de folha de estilo especial na seção <head> e, em seguida, consultar a fonte no CSS.

```
<head>
<link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Sofia">
<style>
body {
  font-family: "Sofia", sans-serif;
}
</style>
</head>
```



@face-font

As fontes da Web permitem que os designers da Web usem fontes que não estão instaladas no computador do usuário. Quando você encontrar/comprar a fonte que deseja usar, basta incluir o arquivo de fonte em seu servidor web, e ele será baixado automaticamente para o usuário quando necessário. Suas fontes "próprias" são definidas dentro da regra CSS @font-face. Para usar a fonte em um elemento HTML, consulte o nome da fonte por meio da propriedade font-family.

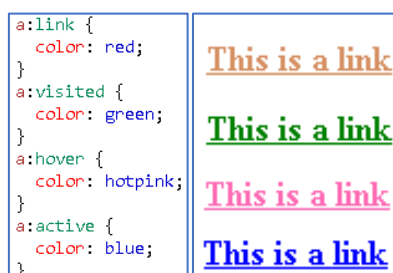
```
@font-face {
  font-family: myFirstFont;
  src: url(sansation_light.woff);
}

div {
  font-family: myFirstFont;
}
```

LINK

Com CSS, os links podem ser estilizados de muitas maneiras diferentes. Além disso, os links podem ter estilos diferentes dependendo do estado em que estão. Os quatro estados de links são:

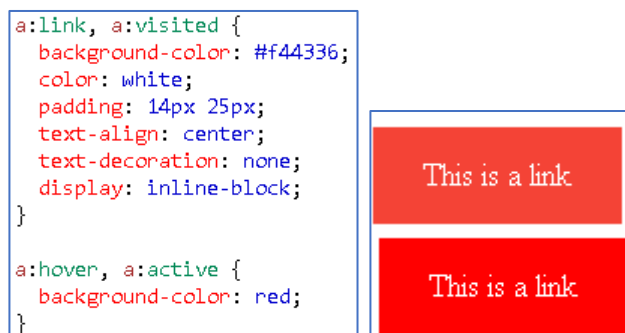
- **a:link**- um link normal e não visitado
- **a:visited**- um link que o usuário visitou
- **a:hover**- um link quando o usuário passa o mouse sobre ele
- **a:active**- um link no momento em que é clicado



Ao definir o estilo para vários estados de link, existem algumas regras de ordem:

- a:hover DEVE vir depois de a:link e a:visited
- a:active DEVE vir depois de a:hover

A propriedade text-decoration é usada principalmente para remover sublinhados de links. A propriedade background-color pode ser usada para especificar uma cor de fundo para links.



Nesse exemplo um link foi estilizado para se parecer com um botão, foram definidos dois estilos para o botão, sendo um para os estados link e visited, e outro estilo para os estados hover e active.

LISTAS

Em HTML, existem dois tipos principais de listas:

- listas não ordenadas () - os itens da lista são marcados com marcadores
- listas ordenadas () - os itens da lista são marcados com números ou letras

As propriedades da lista CSS permitem:

- Definir marcadores de itens de lista diferentes para listas ordenadas
- Definir marcadores de itens de lista diferentes para listas não ordenadas
- Definir uma imagem como marcador de item da lista
- Adicionar cores de fundo a listas e itens de lista

list-style-type

A propriedade list-style-type especifica o tipo de marcador de item de lista.

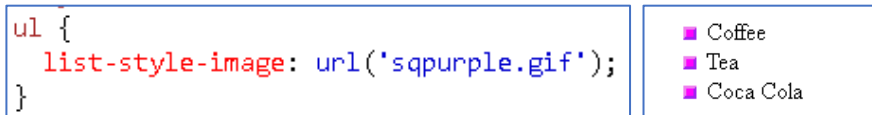
O exemplo a seguir mostra alguns dos marcadores de itens de lista disponíveis:

<pre>ul.a { list-style-type: circle; } ul.b { list-style-type: square; } ol.c { list-style-type: upper-roman; } ol.d { list-style-type: lower-alpha; }</pre>	<p>Example of unordered lists:</p> <ul style="list-style-type: none"> • Coffee • Tea • Coca Cola <p>▪ Coffee</p> <p>▪ Tea</p> <p>▪ Coca Cola</p> <p>Example of ordered lists:</p> <p>I Coffee</p> <p>II Tea</p> <p>III Coca Cola</p> <p>a. Coffee</p> <p>b. Tea</p> <p>c. Coca Cola</p>
---	--

<pre>ul.demo { list-style-type: none; margin: 0; padding: 0; }</pre>	<p>Coffee</p> <p>Tea</p> <p>Coca Cola</p>
--	---

list-style-image

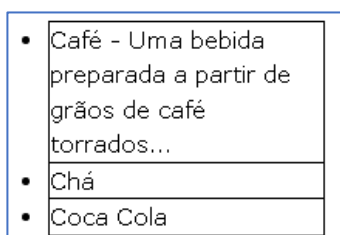
A propriedade list-style-image especifica uma imagem como o marcador de item da lista:



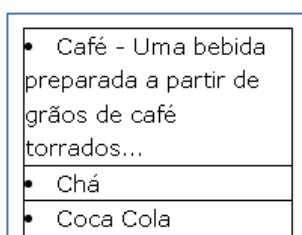
list-style-position

A propriedade list-style-position especifica a posição dos marcadores de item de lista (pontos de bala).

"list-style-position: outside;" significa que os marcadores estarão fora do item da lista. O início de cada linha de um item de lista será alinhado verticalmente. Este é o padrão:



"list-style-position: inside;" significa que os marcadores estarão dentro do item da lista. Como faz parte do item da lista, fará parte do texto e empurrará o texto no início:



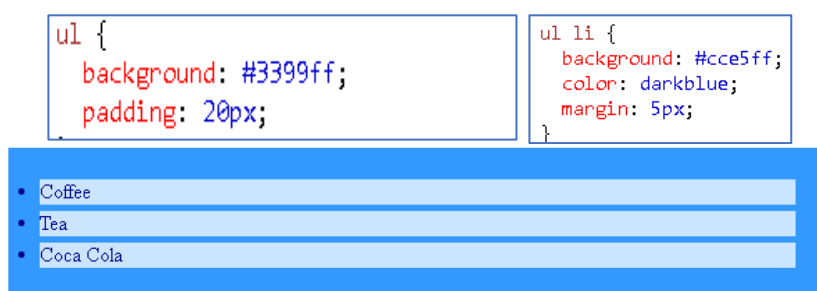
list-style

A propriedade list-style é uma propriedade abreviada. Ele é usado para definir todas as propriedades da lista em uma declaração. Ao usar a propriedade abreviada, a ordem dos valores da propriedade é:

1. **list-style-type**(se uma imagem de estilo de lista for especificada, o valor desta propriedade será exibido se a imagem por algum motivo não puder ser exibida).

2. **list-style-position**(especifica se os marcadores de item de lista devem aparecer dentro ou fora do fluxo de conteúdo).
3. **list-style-image**(especifica uma imagem como marcador de item da lista).

Se um dos valores de propriedade acima estiver ausente, o valor padrão para a propriedade ausente será inserido, se houver.



TABELAS

A aparência de uma tabela HTML pode ser muito melhorada com CSS.

Propriedades da tabela CSS

- **border** - Define todas as propriedades da borda em uma declaração
- **border-collapse** - Especifica se as bordas da tabela devem ou não ser recolhidas
- **border-spacing** - Especifica a distância entre as bordas das células adjacentes
- **caption-side** - Especifica o posicionamento de uma legenda de tabela
- **empty-cells** - Especifica se as bordas e o plano de fundo devem ou não ser exibidos em células vazias em uma tabela
- **table-layout** - Define o algoritmo de layout a ser usado para uma tabela

DISPLAY

Display é a propriedade CSS mais importante para controlar o layout, ela especifica se/como um elemento é exibido. Cada elemento HTML tem um valor de exibição padrão dependendo do tipo de elemento que é. O valor de exibição padrão para a maioria dos elementos é block ou inline.

Elementos Block

Um elemento block sempre começa em uma nova linha e ocupa toda a largura disponível (se estende para a esquerda e para a direita o máximo possível).

O elemento <div> é um elemento de nível de bloco.

Exemplos de elementos block:

- <div>
- <h1> - <h6>
- <p>
- <formulário>
- <cabeçalho>
- <rodapé>
- <seção>

Elementos Inline

Um elemento inline não inicia em uma nova linha e só ocupa a largura necessária.

Este é um elemento embutido dentro de um parágrafo.

Exemplos de elementos inline:

-
- <a>
-

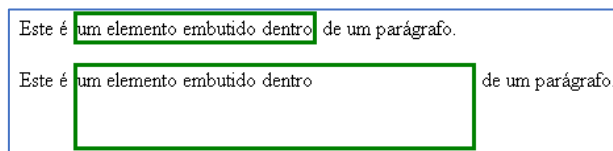
Elementos Inline-Block

Um elemento display: inline-block; mantém as características de um elemento inline, porém, propriedades de um display do tipo block. No próximo exemplo é demonstrado a alteração do display de um elemento inline para um block-inline:

```
<p>Este é <span class="inline">um elemento embutido dentro</span> de um
parágrafo.</p>

<p>Este é <span class="bloco">um elemento embutido dentro</span> de um
parágrafo.</p>
```

```
.inline {
  border:solid 3px green;
  width: 200px;
  height: 50px;
}
.bloco {
  display: inline-block;
  border:solid 3px green;
  width: 300px;
  height: 60px;
}
```



É possível observar que o elemento `<p>`, por padrão, é definido como um elemento do tipo `block`, portanto, ele só aceita elementos do tipo `inline` dentro dele, assim como o elemento ``. Porém é possível alterar como o elemento é exibido (`display`), mostrando-o “como se fosse” um elemento `block`. No exemplo acima houve uma tentativa de acrescentar estilos e alterar valores de altura e largura do elemento ``, mas por ele se tratar de um elemento `inline`, essa alteração não será possível como demonstra o primeiro exemplo `span`. Porém como é feita a alteração do modo de exibição para `block-inline` no segundo elemento `span`, o elemento passa a ser exibido como um bloco, porém embutido dentro de outro bloco aceitando então tais alterações.

Display none

O valor de `display: none;` é comumente usado com JavaScript para ocultar e mostrar elementos sem excluí-los e recriá-los. O elemento `<script>` usa `display: none;` como padrão.

Alterar Display

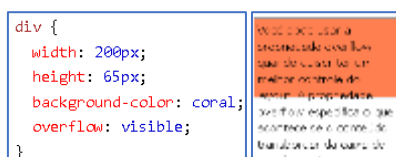
Como mencionado, cada elemento tem um valor de exibição padrão. No entanto, você pode substituir isso. Alterar um elemento `inline` para um elemento `block`, ou vice-versa, pode ser útil para fazer a página parecer de uma maneira específica e ainda seguir os padrões da web. Definir a propriedade `display` de um elemento altera apenas a forma como o elemento é exibido (`display`), não o tipo de elemento que ele é. Portanto, um elemento `inline` com `display: block;` não pode ter outros elementos de `block` dentro dele.

OVERFLOW

A propriedade overflow controla o que acontece com o conteúdo grande demais para caber em uma área. A propriedade especifica se o conteúdo deve ser recortado ou adicionado barras de rolagem quando o conteúdo de um elemento for muito grande para caber na área especificada. A propriedade overflow só funciona para elementos de bloco com uma altura especificada.

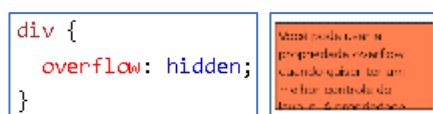
Overflow: visible;

Predefinição. O estouro não é cortado. O conteúdo é renderizado fora da caixa do elemento



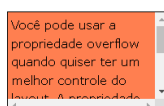
Overflow: hidden;

O estouro é cortado e o restante do conteúdo ficará invisível



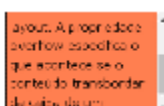
Overflow: scroll;

O estouro é cortado e uma barra de rolagem é adicionada para ver o restante do conteúdo. Esse valor adicionará uma barra de rolagem horizontal e verticalmente (mesmo que você não precise). Ainda podemos editar a barra; com overflow-x apenas a barra horizontal é exibida; com overflow-y apenas a parte vertical é exibida.



Overflow: auto;

Semelhante a scroll, mas adiciona barras de rolagem apenas quando necessário.



ALINHAMENTO HORIZONTAL

elementos block

Para centralizar horizontalmente um elemento do tipo block (como <div>), use **margin: auto;** definir a largura do elemento impedirá que ele se estenda até as bordas de seu contêiner. O elemento então ocupará a largura especificada e o espaço restante será dividido igualmente entre as duas margens:

```
.center {
  margin: auto;
  width: 60%;
  border: 3px solid #73AD21;
  padding: 10px;
}
```

Este elemento div é centralizado.

O alinhamento ao centro com uso de `margin:auto` não tem efeito se a propriedade `width` não estiver definida.

elementos inline ou inline-block

Elementos inline podem ser considerados como um “texto” dentro de um elemento block, seguindo por esse princípio, é possível então alinhar os elementos inline e inline-block com a propriedade **text-align: center;**. Segue um exemplo:

```
<div>

</div>
```

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



Através então da alteração da propriedade `display` é possível acrescentar a propriedade `margin` em elementos inline/inline-block para centralizá-los de outra maneira. Segue um exemplo:

```
img {
  display: block;
  margin-left: auto;
  margin-right: auto;
}
```



POSITION

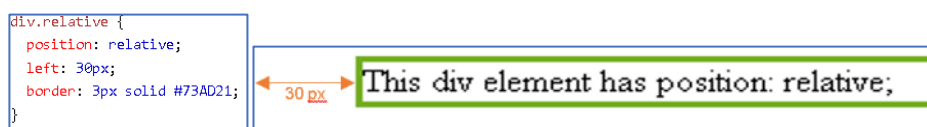
A propriedade position especifica o tipo de método de posicionamento usado para um elemento static, relative, fixed, absolute or sticky (estático, relativo, fixo, absoluto ou grudento). Os elementos são então posicionados usando as propriedades top, bottom, left e right. No entanto, essas propriedades não funcionarão a menos que a position propriedade seja definida primeiro. Eles também funcionam de forma diferente dependendo do valor da propriedade position.

position: static;

A propriedade position define por default o valor static para todos os elementos. Elementos posicionados em static não são afetados pelas propriedades top, bottom, left e right. Um elemento com position: static; não está posicionado de forma especial; é sempre posicionado de acordo com o fluxo normal da página.

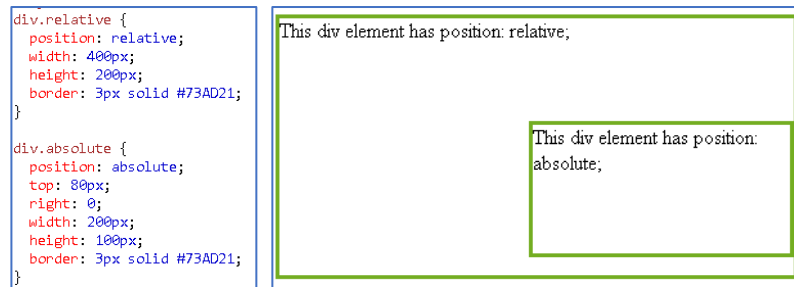
position: relative;

Um elemento com position: relative; é posicionado em relação à sua posição normal. Definir as propriedades top, bottom, left e right de um elemento relativamente posicionado fará com que ele seja ajustado para fora de sua posição normal. Outros conteúdos não serão ajustados para caber em qualquer lacuna deixada pelo elemento.

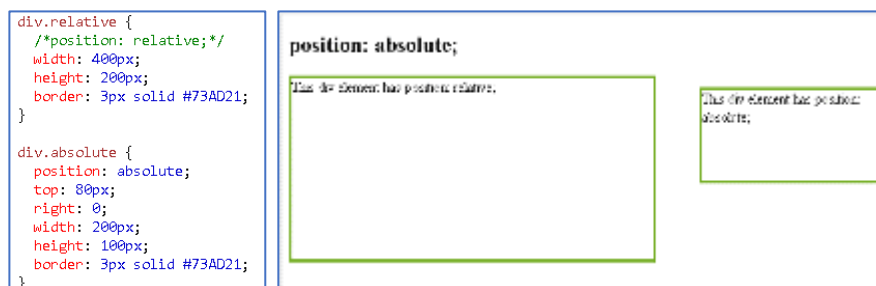


position: absolute;

Um elemento com position: absolute; é posicionado em relação ao ancestral posicionado mais próximo (em vez de posicionado em relação à janela de visualização, como fixo). No entanto; se um elemento posicionado absoluto não tiver parents posicionados, ele usará o corpo do documento. Os elementos posicionados absolutos são removidos do fluxo normal e podem sobrepor elementos.



No exemplo decima foram determinadas as posições para o elemento “div.absolute” através das propriedades right 0px e top 80px, mas isso só foi possível devido a propriedade position deste elemento ter o valor absolute e o elemento parent “pai” deste elemento possuí valor atribuído como relative.



Nesse exemplo o elemento pai do elemento com position absolute, não possui valor de position atribuído (ou position static). Por isso ele vai buscar em outro elemento com relação superior o posicionamento correto. Logo as propriedades right 0px e top 80px foram aplicadas ao corpo da página(elemento body).

position: fixed;

Um elemento com position: fixed; é posicionado em relação à janela de visualização, o que significa que ele sempre permanece no mesmo lugar, mesmo que a página seja rolada(scroll). As propriedades top, right, bottom e left são usadas para posicionar o elemento. Um elemento fixo não deixa uma lacuna na página onde normalmente estaria localizado.

position: sticky;

Um elemento com position: sticky; é posicionado com base na posição de rolagem do usuário. Um elemento sticky se comporta de maneira parecida com elementos relative e fixed, dependendo da posição de rolagem. Ele tem uma posição relativa até que uma determinada posição de deslocamento seja encontrada na viewport, então ele "gruda" no lugar (como position:fixed).

SOBREPOSIÇÃO Z-INDEX

Quando os elementos são posicionados, eles podem se sobrepor a outros elementos. A propriedade z-index especifica a ordem de pilha de um elemento (qual elemento deve ser colocado na frente ou atrás dos outros) com valores numéricos. Um elemento pode ter uma ordem de pilha positiva ou negativa. A propriedade z-index só funciona em elementos propriedade position definida e flex items (flexbox).



FLOAT e CLEAR

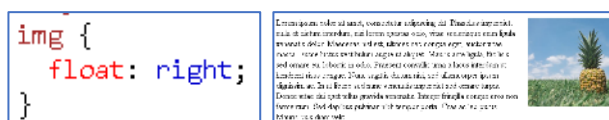
A propriedade float especifica como um elemento deve flutuar. A propriedade clear especifica quais elementos podem flutuar ao lado do elemento limpo e de que lado.

float

A propriedade float é usada para posicionar e formatar o conteúdo, por exemplo, deixar uma imagem flutuar à esquerda do texto em um contêiner. A float pode ter um dos seguintes valores:

- left- O elemento flutua à esquerda de seu contêiner
- right- O elemento flutua à direita de seu contêiner
- none- O elemento não flutua (será exibido exatamente onde ocorre no texto). Isso é padrão
- inherit- O elemento herda o valor float de seu pai

Em seu uso mais simples, a propriedade float pode ser usada para quebrar o texto em torno das imagens



clear

Quando usamos a propriedade float, e queremos o próximo elemento abaixo (não à direita ou à esquerda), teremos que usar a propriedade clear. A

propriedade `clear` especifica o que deve acontecer com o elemento que está próximo a um elemento flutuante. Ela pode ter um dos seguintes valores:

- `none`- O elemento não é empurrado abaixo dos elementos flutuantes à esquerda ou à direita. Isso é padrão
- `left`- O elemento é empurrado abaixo dos elementos flutuantes à esquerda
- `right`- O elemento é empurrado abaixo dos elementos flutuantes à direita
- `both`- O elemento é empurrado abaixo dos elementos flutuantes esquerdo e direito
- `inherit`- O elemento herda o valor `clear` de seu pai

ALINHAMENTO VERTICAL

Usando Padding.

Existem muitas maneiras de centralizar um elemento verticalmente em CSS. Uma solução simples é usar **top e bottom padding**:



Usando line-height.

Outro truque é usar a propriedade `line-height` com um valor igual à propriedade `height` do elemento.

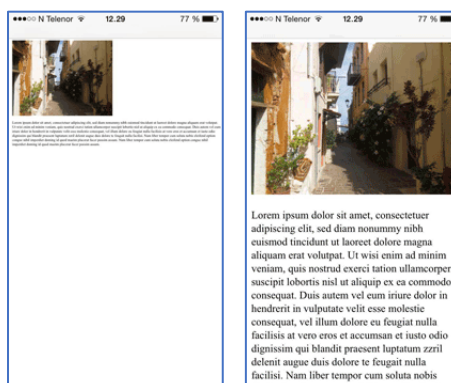
Usando transform e position

Se `padding` e `line-height` não forem opções, outra solução é usar o posicionamento e a propriedade `transform`.

VIEWPORT

Viewport ou janela de visualização é a área visível do usuário de uma página da web. Ela varia de acordo com o dispositivo e será menor em um celular do que em uma tela de computador. Antes dos tablets e celulares, as páginas da web eram projetadas apenas para telas de computador, e era comum que as páginas da web tivessem um design estático e um tamanho fixo. Então, quando começamos a navegar na internet usando tablets e telefones celulares, as

páginas da web de tamanho fixo eram muito grandes para caber na janela de visualização. Para corrigir isso, os navegadores desses dispositivos reduziram toda a página da Web para caber na tela.



Configurando Viewport

O HTML5 introduziu um método para permitir que os web designers assumam o controle da viewport, por meio da <meta>tag. Você deve incluir o seguinte <meta>elemento de janela de visualização em todas as suas páginas da web:

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

Isso fornece ao navegador instruções sobre como controlar as dimensões e o dimensionamento da página.

- A parte **width=device-width** define a largura da página para seguir a largura da tela do dispositivo (que varia de acordo com o dispositivo).
- A parte **initial-scale=1.0** define o nível de zoom inicial quando a página é carregada pela primeira vez pelo navegador.

RESPONSIVIDADE

Um web-design responsivo faz com que sua página da web fique bem em todos os dispositivos.



É possível fazer essa responsividade alterando valores de atributos como width e height para corresponder a um valor em porcentagem de elementos parente.

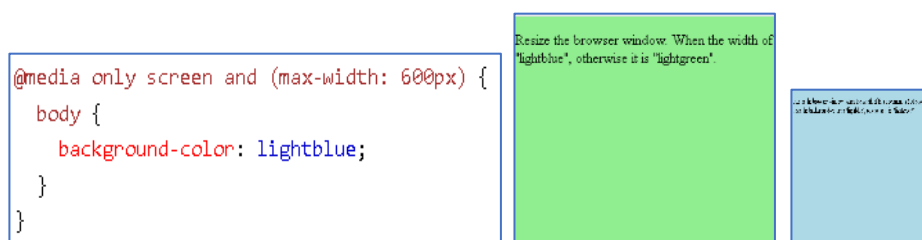


No exemplo acima um elemento parágrafo representado pela cor azul está dentro de um elemento div representado pela cor cinza, o elemento de cor azul sempre vai ficar com width proporcional a 80% da width do elemento cinza. Já o elemento cinza está com propriedades width e height diretamente relacionadas ao tamanho da viewport já que foram atribuídos valores em porcentagem para essas propriedades.

MEDIA QUERY

Media query é uma técnica CSS introduzida no CSS3. Ele usa a regra @media para incluir um bloco de propriedades CSS somente se uma determinada condição for validada. Ou seja, ela cria uma condição para certos atributos de estilos serem atribuídos.

Nesse exemplo a condição observa se o tamanho da janela do navegador corresponde a 600px ou menor, caso isso aconteça a cor de fundo será azul claro:



Media query é útil quando você deseja modificar seu site ou aplicativo dependendo do tipo de dispositivo (como tela de impressão x tela) ou características e parâmetros específicos (como resolução de tela ou largura da janela de visualização do navegador).

Sintaxe

Uma media query pode ser composta por **media types** (opcional) e expressões **media feature**, que podem ser combinadas de acordo com a necessidade e de varias maneiras através do uso de **operadores lógicos**. Expressões media query não diferenciam letras minúsculas e letras maiúsculas.

Media types – Definem qual a tipo de dispositivo a media-query está sendo aplicada: all (Adequado para todos os dispositivos), print (Destinado a material paginado e documentos visualizados em uma tela no modo de visualização de impressão), screen (Destinado para telas). Se não for especificado o media type, será então atribuído o valor “all” (exceto quando os operadores not e only são utilizados, pois os mesmos exigem os media types).

```
@media print { ... }
```

Media features – Descrevem e especificam características do agente de usuário (página web, browser, viewport, etc): any-hover, any-pointer, aspect-ratio, color, color-gamut, color-index, device-aspect-ratio, device-height, device-width, display-mode, forced-colors, grid, height, hover, inverted-colors, monochrome, orientation, overflow-block, overflow-inline, pointer, prefers-color-scheme, prefers-contrast, prefers-reduced-motion, resolution, scripting, update, width. As expressões de media features devem ser especificadas entre parentes.

```
@media (hover: hover) { ... }
```

Por exemplo, a feature (recurso) **hover** permite que a media-query teste se o usuário está passando o mouse em cima do elemento e essa condição for atendida estilos podem ser atribuídos, etc.

Operadores Lógicos – Podem ser usadas para compor media queries mais complexas. São eles: “not”, “Only”, “and” e “,(vírgula)”.

Exemplos:

Este exemplo combina dois media features para restringir estilos a dispositivos orientados em paisagem e com uma largura de pelo menos 30 em:

```
@media (min-width: 30em) and (orientation: landscape) {  
... }
```

A regra a seguir aplicará seus estilos se o dispositivo do usuário tiver uma altura mínima de 680px ou for um dispositivo de tela no modo retrato:

```
@media (min-height: 680px), screen and (orientation:  
portrait) { ... }
```

COMBINADORES CSS

Um combinador é algo que determina especifica uma relação entre dois seletores. Um seletor CSS pode conter mais de um seletor simples, entre eles pode ser incluso um combinador.

Descendant Selector (Space)

O seletor de descendentes corresponde a todos os elementos que são descendentes de um elemento específico. O seguinte exemplo seleciona todos os elementos <p> dentro de um elemento <div>.

<code><head></code>	
<code><style></code>	
<code>div p {</code>	
<code>background-color: yellow;</code>	
<code>}</code>	
<code></style></code>	
<code></head></code>	
<code><body></code>	
<code><div></code>	Paragraph 1 in the div.
<code><p>Paragraph 1 in the div.</p></code>	Paragraph 2 in the div.
<code><p>Paragraph 2 in the div.</p></code>	
<code><section><p>Paragraph 3 in the div.</p></section></code>	Paragraph 3 in the div.
<code></div></code>	
<code><p>Paragraph 4. Not in a div.</p></code>	Paragraph 4. Not in a div.
<code><p>Paragraph 5. Not in a div.</p></code>	
<code></body></code>	Paragraph 5. Not in a div.

Child Selector (>)

O seletor de filho seleciona todos os elementos que são filhos de um elemento específico. O seguinte exemplo seleciona todos os elementos `<p>` que são filhos de um elemento `<div>`.

<code><head></code>	
<code><style></code>	
<code>div > p {</code>	
<code>background-color: yellow;</code>	
<code>}</code>	
<code></style></code>	
<code></head></code>	
<code><body></code>	
<code><div></code>	
<code><p>Paragraph 1 in the div.</p></code>	Paragraph 1 in the div.
<code><p>Paragraph 2 in the div.</p></code>	
<code><section></code>	Paragraph 2 in the div.
<code><!-- not Child but Descendant --></code>	
<code><p>Paragraph 3 in the div (inside a section element).</p></code>	Paragraph 3 in the div (inside a section element).
<code></section></code>	
<code><p>Paragraph 4 in the div.</p></code>	Paragraph 4 in the div.
<code></div></code>	
<code><p>Paragraph 5. Not in a div.</p></code>	Paragraph 5. Not in a div.
<code><p>Paragraph 6. Not in a div.</p></code>	
<code></body></code>	Paragraph 6. Not in a div.

Adjacent Sibling Selector (+)

O seletor de irmãos adjacentes é usado para selecionar um elemento que está imediatamente após um elemento específico. Sibling (irmãos), pois, eles têm o mesmo parent (pai), e adjacente (adjacente) para significar “imediatamente a seguir”. O seguinte exemplo seleciona o primeiro elemento `<p>` que está localizado imediatamente após o elemento `<div>`.

<code><head></code>	
<code><style></code>	
<code>div + p {</code>	
<code>background-color: yellow;</code>	
<code>}</code>	
<code></style></code>	
<code></head></code>	
<code><body></code>	
<code><div></code>	Paragraph 1 in the div.
<code><p>Paragraph 1 in the div.</p></code>	
<code><p>Paragraph 2 in the div.</p></code>	Paragraph 2 in the div.
<code></div></code>	
<code><p>Paragraph 3. After a div.</p></code>	Paragraph 3. After a div.
<code><p>Paragraph 4. After a div.</p></code>	
<code></body></code>	Paragraph 4. After a div.

General Sibling Selector (~)

O seletor geral de irmãos seleciona todos os próximos elementos irmãos de um elemento específico. Sibling (irmãos), pois, eles têm o mesmo parent (pai).

O seguinte exemplo seleciona todos os elementos <p> que são os próximos irmãos de um elemento <div>.

```
<head>
<style>
div ~ p {
  background-color: yellow;
}
</style>
</head>
<body>
<p>Paragraph 1.</p>
<div>
  <p>Paragraph 2.</p>
</div>
<p>Paragraph 3.</p>
<code>Some code.</code>
<p>Paragraph 4.</p>
<p>Paragraph 5.</p>
<p>Paragraph 6.</p>
</body>
```

Paragraph 1.

Paragraph 2.

Paragraph 3.

Some code.

Paragraph 4.

Paragraph 5.

Paragraph 6.

SELETORES DE ATRIBUTOS

É possível alterar o estilo de um elemento que tenha um atributo específico ou valor de atributo específico através dos seletores de atributos.

CSS [attribute] Selector

O seletor **[atributo]** é usado para selecionar elementos com um atributo específico. O seguinte exemplo seleciona todos os elementos <a> com o atributo "target".

```
<head>
<style>
a[target] {
  background-color: yellow;
}
</style>
</head>
<body>
<a href="https://www.w3schools.com">w3schools.com</a>
<a href="http://www.disney.com" target="_blank">disney.com</a>
<a href="http://www.wikipedia.org" target="_top">wikipedia.org</a>
w3schools.com disney.com wikipedia.org
</body>
```

CSS [attribute="value"] Selector

O seletor **[atributo="valor"]** é usado para selecionar um elemento com um valor específico de atributo.

```
<head>
<style>
a[target=_blank] {
  background-color: yellow;
}
</style>
</head>
<body>
<a href="https://www.w3schools.com">w3schools.com</a>
<a href="http://www.disney.com" target="_blank">disney.com</a>
<a href="http://www.wikipedia.org" target="_top">wikipedia.org</a>
w3schools.com disney.com wikipedia.org
</body>
```

CSS [attribute~="value"] Selector

O seletor **[atributo~="valor"]** é usado para selecionar elementos com um valor específico de um atributo contendo uma palavra específica.



CSS [attribute|= "value"] Selector

O seletor **[atributo|= "valor "]** é usado para selecionar elementos com um atributo específico e um valor específico ou esse valor seguido de um hífen (-). O valor deve ser uma palavra inteira, sozinha, como `class="top"`, ou seguida por um hífen(-), como `class="top-text"`.



CSS [attribute^="value"] Selector

O seletor **[atributo^="valor"]** é usado para selecionar um elemento com atributo específico do qual o valor começa com um valor específico. O valor não precisa ser uma palavra inteira.



CSS [attribute\$="value"] Selector

O seletor **[atributo\$="valor"]** é usado para selecionar elementos do qual o valor de atributo terminar com um valor específico. O valor não precisa ser uma palavra inteira.

```
<head>
<style>
[class$="test"] {
  background: yellow;
}
</style>
</head>
<body>
<div class="first_test">The first div element.</div>
<div class="second">The second div element.</div>
<div class="my-test">The third div element.</div>
<p class="mytest">This is some text in a paragraph.</p>
</body>
```

The first div element.
The second div element.
The third div element.
This is some text in a paragraph.

CSS [attribute*="value"] Selector

O seletor **[atributo*="valor"]** é usado para selecionar elementos do qual o valor de atributo contém um valor específico em qualquer parte. O valor não precisa ser uma palavra inteira.

```
<head>
<style>
[class*="te"] {
  background: yellow;
}
</style>
</head>
<body>
<div class="first_test">The first div element.</div>
<div class="second">The second div element.</div>
<div class="aateaa">The third div element.</div>
<p class="mytest">This is some text in a paragraph.</p>
</body>
```

The first div element.
The second div element.
The third div element.
This is some text in a paragraph.

PSEUDO-ELEMENTOS CSS

No CSS, pseudo-elementos são utilizados para estilizar partes específicas de um elemento. A sintaxe dos pseudo-elementos é:

```
selector::pseudo-element {
  property: value;
}
```

O seguinte exemplo exibirá a primeira letra dos parágrafos com class="intro", em vermelho e em tamanho maior.

```

<head>
<style>
p.intro::first-letter {
  color: #ff0000;
  font-size: 200%;
}
</style>
</head>
<body>
<p class="intro">This is an introduction.</p>
<p>This is a paragraph with some text. A bit more text
even.</p>
</body>

```

This is an introduction.

This is a paragraph with some text. A bit more text even.

Pseudo-Elementos Referência

SELETOR	EXEMPLO	DESCRIÇÃO DO EXEMPLO
<u>::after</u>	p::after	Inserir algo após o conteúdo de cada elemento <p>
<u>::before</u>	p::before	Inserir algo após o conteúdo de cada parágrafo <p>
<u>::first-letter</u>	p::first-letter	Seleciona a primeira letra de cada elemento <p>
<u>::first-line</u>	p::first-line	Seleciona a primeira linha de cada elemento <p>
<u>::marker</u>	::marker	Seleciona o marcador dos itens de uma lista
<u>::selection</u>	p::selection	Seleciona a porção de um elemento que está sendo selecionada pelo usuário

PSEUDO-CLASSES CSS

A pseudoclasse em CSS é utilizada para definir um estado especial de um elemento. Como por exemplo estilizar um elemento quando o mouse está sobre ele, estilizar um link que ainda não foi visitado, estilizar um elemento que está em foco, etc.

A sintaxe das pseudoclasses é:

```

selector:pseudo-class {
  property: value;
}

```

No exemplo a seguir, o seletor corresponde a qualquer elemento <p> que seja o primeiro filho de qualquer elemento.

```

<head>
<style>
p:first-child {
  color: blue;
}
</style>
</head>
<body>
<p>This is some text.</p>
<p>This is some text.</p>
<div>
  <p>This is some text.</p>
  <p>This is some text.</p>
</div>
</body>

```

This is some text.

This is some text.

This is some text.

This is some text.

Pseudo-Classes Referência

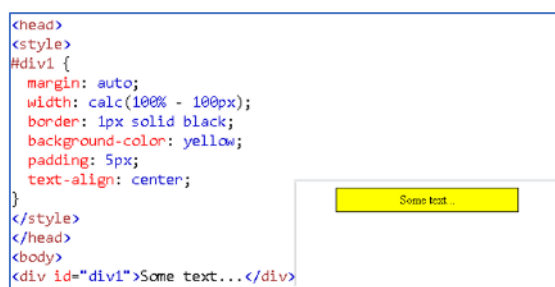
SELETOR	EXEMPLO	DESCRIÇÃO DO EXEMPLO
<u>:active</u>	a:active	Seleciona um link que já foi clicado (ativo)
<u>:checked</u>	input:checked	Seleciona todos os elementos input checked (preenchido / marcado)
<u>:disabled</u>	input:disabled	Seleciona todos os elementos input que estão desabilitados
<u>:empty</u>	p:empty	Seleciona todos os elementos <p> que não possuem filhos ou vazios
<u>:enabled</u>	input:enabled	Seleciona todos os elementos input que estão habilitados
<u>:first-child</u>	p:first-child	Seleciona todos os elementos <p> que são os primeiros filhos de qualquer elemento
<u>:first-of-type</u>	p:first-of-type	Seleciona todos os elementos <p> que são os primeiros de seu tipo
<u>:focus</u>	input:focus	Seleciona o elemento <input> que está em foco
<u>:hover</u>	a:hover	Seleciona o link em que o mouse está sobre ele
<u>:in-range</u>	input:in-range	Seleciona um elemento <input> com um valor dentro de um intervalo determinado
<u>:invalid</u>	input:invalid	Seleciona todos os elementos <input> com um valor inválido
<u>:lang(language)</u>	p:lang(it)	Seleciona todos elementos <p> com um atributo de linguagem começado com "it"
<u>:last-child</u>	p:last-child	Seleciona todos os elementos <p> que são os últimos filhos de um pai
<u>:last-of-type</u>	p:last-of-type	Seleciona todos elementos <p> que são os últimos do seu tipo
<u>:link</u>	a:link	Seleciona todos os links não visitados
<u>:not(selector)</u>	:not(p)	Seleciona todos os elementos que não são um elemento <p>
<u>:nth-child(n)</u>	p:nth-child(2)	Seleciona todos os elementos <p> que são os segundos filhos de um pai
<u>:nth-last-child(n)</u>	p:nth-last-child(2)	Seleciona todos os elementos <p> que são os segundos filhos de um pai, contando a partir do ultimo
<u>:nth-last-of-type(n)</u>	p:nth-last-of-type(2)	Seleciona todos os elementos <p> que são os segundos do seu tipo, contando a partir do ultimo
<u>:nth-of-type(n)</u>	p:nth-of-type(2)	Seleciona todos os elementos <p> que são os segundos do seu tipo
<u>:only-of-type</u>	p:only-of-type	Seleciona todos os elementos <p> que são os únicos do seu tipo de um elemento pai
<u>:only-child</u>	p:only-child	Seleciona todos os elementos <p> que são os únicos filhos de um elemento pai
<u>:optional</u>	input:optional	Seleciona os elementos <input> que não tenham o atributo "required" especificado

<u>:out-of-range</u>	input:out-of-range	Seleciona todos os elementos <input> com o valor fora de um intervalo determinado
<u>:read-only</u>	input:read-only	Seleciona todos os elementos <input> com o atributo "readonly" especificado
<u>:read-write</u>	input:read-write	Seleciona todos os elementos <input>
<u>:required</u>	input:required	Seleciona os elementos <input> que tenham o atributo "required" especificado
<u>:root</u>	root	Seleciona o elemento raiz do documento
<u>:target</u>	#news:target	Seleciona o elemento #news que está ativo atualmente (clicado em um URL que contém esse nome de âncora)
<u>:valid</u>	input:valid	Seleciona todos os elementos <input> com um valor válido
<u>:visited</u>	a:visited	Seleciona todos os links que já foram visitados/acessados

FUNÇÃO CALC(_)

A função calc() executa um calculo que pode ser usado como valor de uma propriedade. É necessária uma expressão matemática em que o resultado será usado como o valor. Os seguintes operadores podem ser usados: + - * /.

No seguinte exemplo a largura de um elemento é determinada com a função calc() que especifica que a largura de um elemento <div> deve ser 100% - 100pixels, ou seja, um espaço de 50px entre as duas bordas da janela.



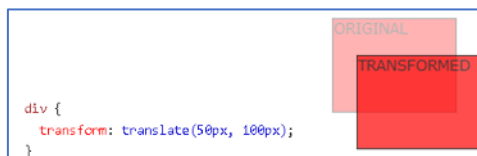
TRANSFORMAÇÕES

Transformações CSS permitem mover, rotacionar, dimensionar e torcer elementos.

Método translate()

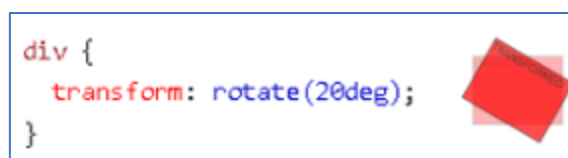
O método translate() move um elemento de sua posição atual de acordo com os parâmetros fornecidos para o eixo X e o eixo Y. O seguinte exemplo

move um elemento 50 pixels para direita e 100 pixels para baixo de sua posição atual.



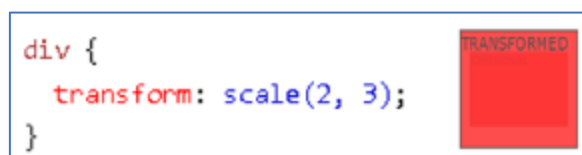
Método rotate()

O método rotate() rotaciona um elemento no sentido horário ou anti-horário de acordo com um determinado grau. O uso de valores negativos girará o elemento no sentido anti-horário. O exemplo a seguir gira o elemento <div> no sentido horário com 20 graus.



Método scale()

O método scale() aumenta ou diminui o tamanho de um elemento de acordo com parâmetros fornecidos como largura e altura. O seguinte exemplo aumenta o elemento <div> para ser duas vezes maior que o tamanho original de largura e três vezes maior que o tamanho original de altura.



Método skew()

O método skew() “torce” um elemento ao longo do eixo X e Y de acordo com um angulo fornecido. Se o segundo parâmetro não for especificado o valor do eixo Y vai ser 0. O seguinte exemplo torce um elemento <div> 20 graus ao longo do eixo X e 10 graus ao longo do eixo Y.



Método matrix()

O matrix() método combina todos os métodos de transformação em um. O método matrix() aceita seis parâmetros, contendo funções matemáticas, que permitem girar, dimensionar, mover (traduzir) e inclinar elementos. Os parâmetros são os seguintes:

matrix(scaleX(),skewY(),skewX(),scaleY(),translateX(),translateY())

```
div {
  transform: matrix(1, -0.3, 0, 1, 0, 0);
}
```

Referência Propriedades de Transformação

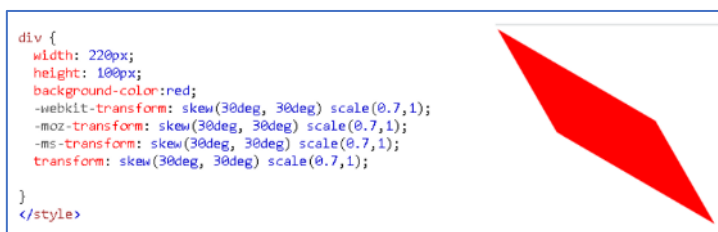
Propriedade	Descrição
transform	Aplica uma transformação 2D ou 3D em um elemento
transform-origin	Permite alterar a posição de um elemento transformado

Referência Métodos de Transformação

Função	Descrição
matrix(n,n,n,n,n,n)	Define uma transformação 2D usando uma matriz de 6 valores
translate(x,y)	Define uma interpretação 2D do elemento, movendo-o ao longo do eixo X e eixo Y
translateX(n)	Define uma interpretação 2D do elemento, movendo-o ao longo do eixo X
translateY(n)	Define uma interpretação 2D do elemento, movendo-o ao longo do eixo Y
scale(x,y)	Define uma transformação de dimensão alterando a largura e altura de um elemento
scaleX(n)	Define uma transformação de dimensão alterando a largura de um elemento
scaleY(n)	Define uma transformação de dimensão alterando a altura de um elemento
rotate(angle)	Define uma rotação 2D, o ângulo deve ser especificado nos parâmetros
skew(x-angle,y-angle)	Define uma transformação de torção 2D ao longo do eixo X e do eixo Y
skewX(angle)	Define uma transformação de torção 2D ao longo do eixo X
skewY(angle)	Define uma transformação de torção 2D ao longo do eixo Y

COMPATIBILIDADE DE NAVEGADORES

Antes de começar a implementar o HTML5 e o CSS3 em seus projetos, é importante você ter consciência das marcações e propriedades suportadas pelas diferentes versões dos diferentes navegadores. No seguinte exemplo o uso da propriedade transform não apresenta suporte a todos os navegadores ou a versões anteriores, por isso o código foi adaptado para funcionar corretamente em, versões inferiores do Google Chrome (“-webkit-”), versões inferiores do internet explore (“-ms-”) e com versões do navegador Mozilla Firefox (“-moz-”).



Verificar Suporte dos Navegadores

- <https://caniuse.com/> - informações completas sobre o suporte
- <https://shouldiprefix.com/> - informações sobre prefixos
- <https://html5readiness.com/> - infográfico interativo.
- <https://modernizr.com/> - Técnicas para compatibilidade com biblioteca JS.

VARIÁVEIS CSS

A função `var()` é usada para inserir o valor de uma variável CSS. As variáveis CSS podem ser acessadas pelo DOM, o que significa que é possível criar e alterar variáveis CSS com JavaScript. A sintaxe da função das variáveis CSS é:

```
var(--name, value)
```

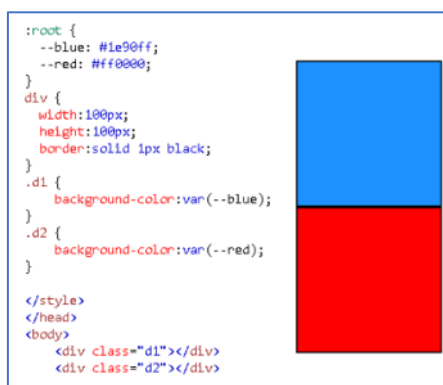
O nome da variável deve sempre começar com dois dashes (--) quando for declarada.

Variável de Escopo Global

Para criar primeiro é preciso declarar o seletor “:root”, esse seletor corresponde ao elemento raiz do documento, ou seja, ao “HTML”. Após isso as variáveis podem ser declaradas dentro desse seletor, os seus nomes devem

começar com dois dashes (--) e são case sensitive. Após isso essas variáveis podem ser usadas como valores de propriedades para alterar estilos de elementos como uso da função var().

No seguinte exemplo foram declaradas duas variáveis (--blue e --red), e então utilizada a função var() para utilizar o valor dessas variáveis em alguns elementos.



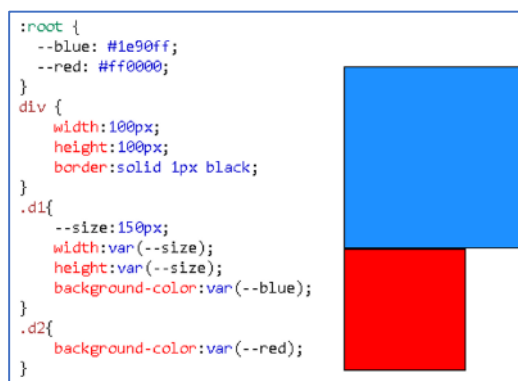
As vantagens de usar var() é tornar o código mais fácil de ler (mais compreensível) e tornar muito mais fácil alterar os valores de cor. Para alterar a cor "--blue" e "--red" para um azul e vermelho mais suave, basta alterar os dois valores das variáveis.

Variável de Escopo Local

Às vezes queremos que as variáveis mudem apenas em uma seção específica da página. No seguinte exemplo suponha que queremos uma cor diferente de azul para o elemento <div class="d1">. Então, podemos declarar novamente a variável --blue dentro do seletor da <div>. Quando for utilizada a função var(--blue) dentro deste seletor, ela usará o valor da variável local --blue.

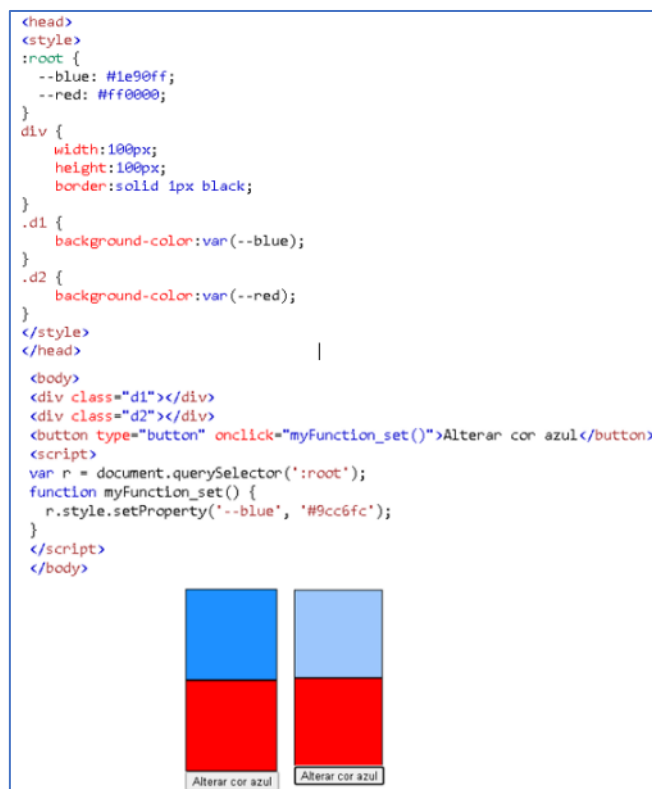


As variáveis de escopo local são declaradas no próprio seletor e não necessitam do seletor `:root`. No seguinte exemplo foi alterado o tamanho do elemento utilizando uma variável de escopo local (`--size`).



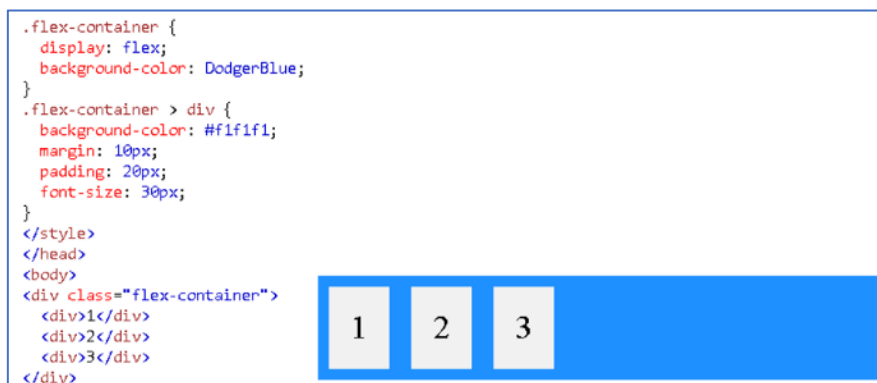
Alterar Variáveis CSS Com Javascript

As variáveis CSS têm acesso ao DOM, o que significa que você pode alterá-las com JavaScript. É possível alterar e acessar as variáveis CSS através dos métodos **getComputedStyle()**, para pegar todos as propriedades e valores de estilo de um elemento, **getPropertyValue()**, para pegar o valor de uma propriedade e o método **setProperty()** para definir uma nova variável CSS ou alterar uma variável existente. No exemplo a seguir é demonstrado como criar um script para alterar a variável `--blue` dos exemplos usados anteriormente.



FLEXBOX

Flexbox é um módulo de layout de caixa flexível, através dele é possível deixar o design mais fácil, responsivo e estruturar o layout sem o uso das técnicas de posicionamento. **Para utilizar o modelo Flexbox é preciso definir um “flex container”** (recipiente flexível), esse container se torna flexível quando é atribuído a ele a propriedade de “display:flex”.



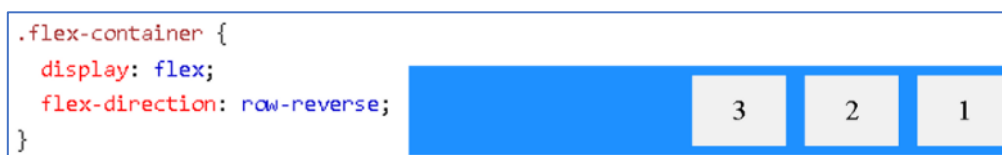
flex-direction

A propriedade **flex-direction** define em qual direção o container vai empilhar/organizar seus flex-items.

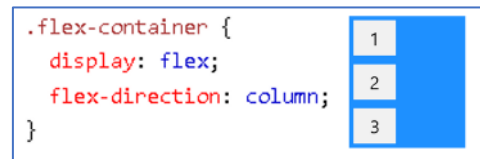
Por padrão a propriedade é definida como **flex-direction:row**, ou seja, os flex-items dentro do container serão organizados horizontalmente em uma “linha”, começando da esquerda para direita.



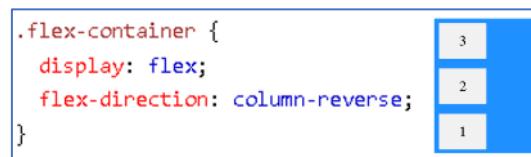
A propriedade definida como **flex-direction:row-reverse** organiza os flex-items de maneira semelhante, porém começando direita para esquerda.



A propriedade definida como **flex-direction:column** organiza os flex-items verticalmente como uma coluna de cima para baixo.



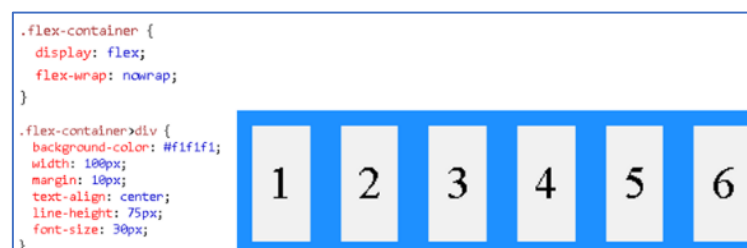
A propriedade definida como **flex-direction:column-reverse** organiza os flex-items de maneira semelhante, porém começando de baixo para cima.



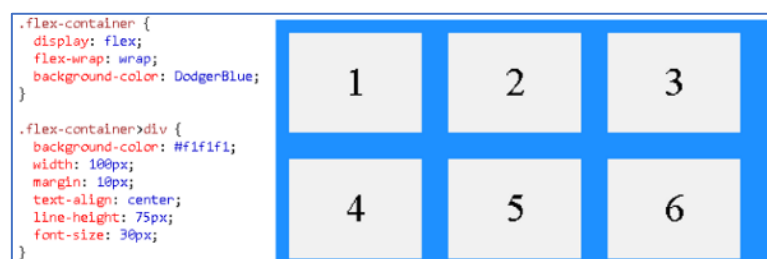
flex-wrap

A propriedade **flex-wrap** especifica se os flex-items devem ser “wrapped” (embrulhados, agrupados), ou seja, essa propriedade define se os tamanhos podem ou não ser alterados para caber dentro do container em uma linha ou coluna.

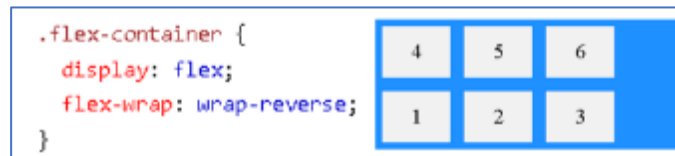
Por padrão o valor da propriedade é **flex-wrap:nowrap**, ou seja, os flex-items serão “espremidos” conforme o tamanho da janela é diminuído para caber dentro do container.



A propriedade definida como **flex-wrap:wrap** vai “espremer” os flex-items somente se necessário (quando o tamanho da janela for menor que o tamanho do item).

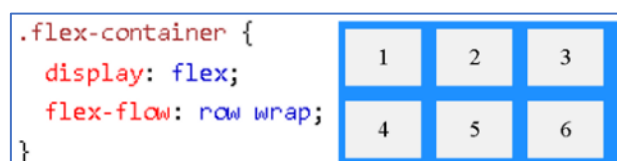


A propriedade definida como **flex-wrap:wrap-reverse** especifica que os itens flexíveis serão agrupados na ordem inversa.



flex-flow

A propriedade **flex-flow** é uma maneira encurtada para definir as propriedades flex-direction e flex-wrap em uma única linha.



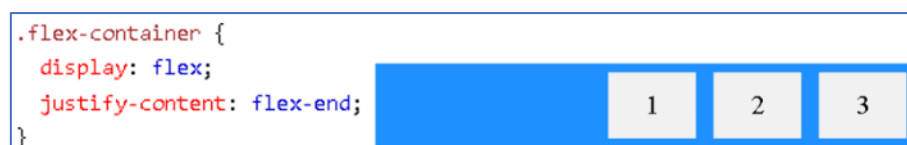
justify-content

A propriedade **justify-content** é usada para alinhar os flex-items de maneira **horizontal** de acordo com a largura dos itens e do container.

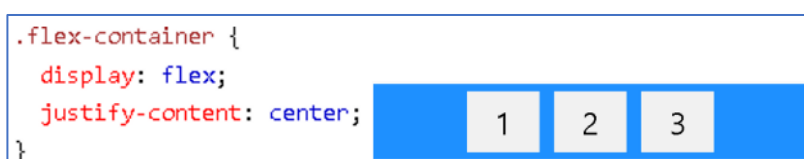
Por padrão a propriedade é definida como **justify-content :flex-start**, os seja, os flex-items vão ser alinhados no começo do container.



A propriedade definida com o valor **justify-content :flex-end** alinha os flex-items no final do container.



A propriedade definida com o valor **justify-content :center** alinha os flex-items no centro do container.



A propriedade definida com o valor **justify-content :space-around** exibe os flex-items com espaços antes, entre e depois das linhas ou colunas.



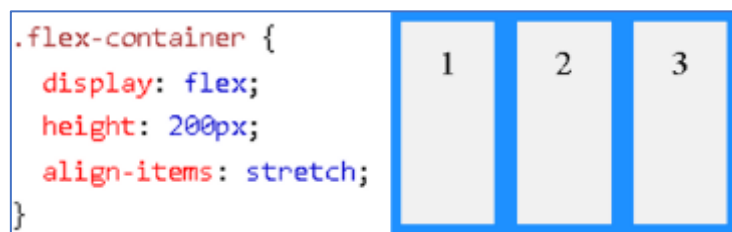
A propriedade definida com o valor **justify-content:space-between** exibe os flex-items com espaços entre eles.



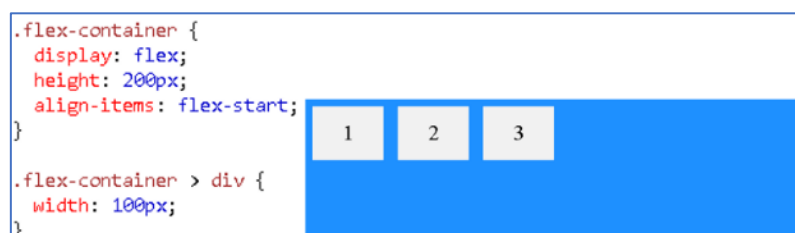
align-items

A propriedade **align-items** também é usada para alinhar os flex-items, porém de maneira **vertical** de acordo com o tamanho dos itens e do container.

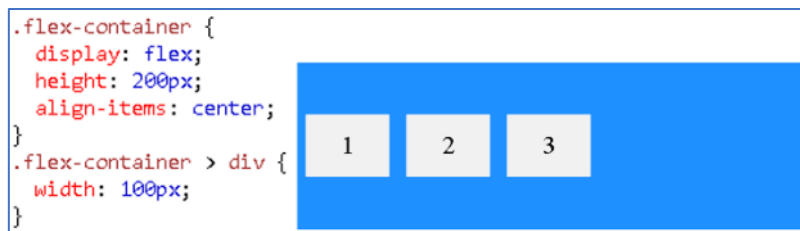
Por padrão a propriedade é definida com o valor **align-items:stretch**. Os flex-items serão “esticados” para preencher o container se o mesmo tiver uma altura definida.



A propriedade definida com o valor **align-items:flex-start** alinha os flex-items na parte de cima do container.



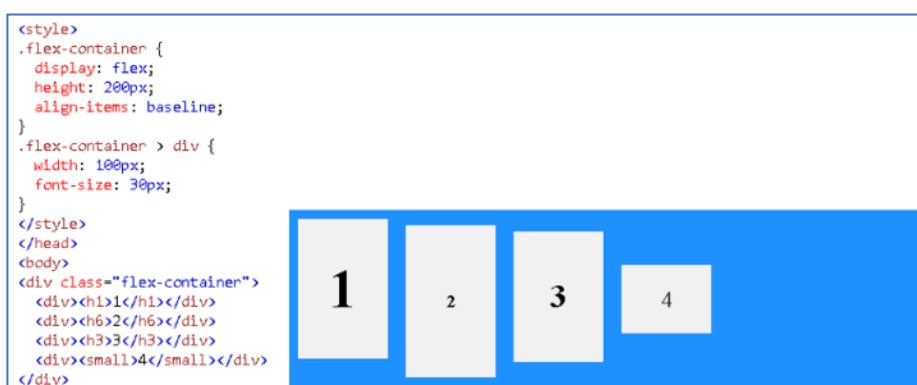
A propriedade definida com o valor **align-items:center** alinha os flex-items no centro do container.



A propriedade definida com o valor **align-items:flex-end** alinha os flex-items na parte de baixo do container.



A propriedade definida com o valor **align-items:baseline** alinha os flex-items de acordo com as suas bases de linha. Tamanhos de fonte diferentes definem linhas de base diferentes, a propriedade com o valor baseline alinha essas bases de linha e por fim altera o tamanho dos flex-items de acordo com o tamanho do container e a linha de base.

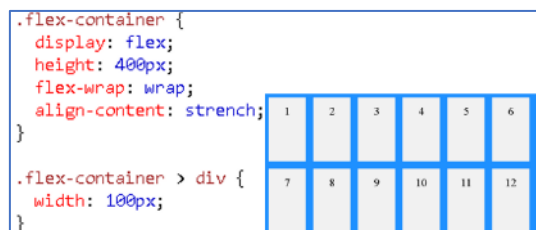


align-content

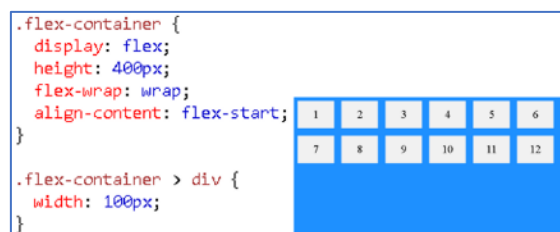
Como já foi mencionado, por padrão os flex-items possuem a propriedade **flex-wrap:nowrap**, ou seja, o tamanho do item vai ser modificado para todos os itens caberem dentro de uma única linha ou coluna do container. Porém caso a propriedade for definida como **flex-wrap:wrap** o container vai criar mais linhas ou colunas para que caber todos os flex-items tentando manter seus tamanhos previamente definidos. A propriedade **align-content** **alinha as linhas e colunas dos flex-items**.



Por padrão a propriedade é definida como **align-content: stretch**, as linhas vão ser esticadas para preencher o espaço do container.



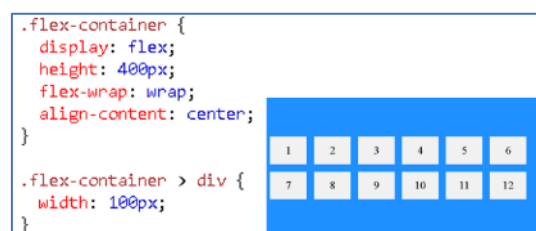
A propriedade definida com o valor **align-content: flex-start** vai e exibir as linhas no começo do container.



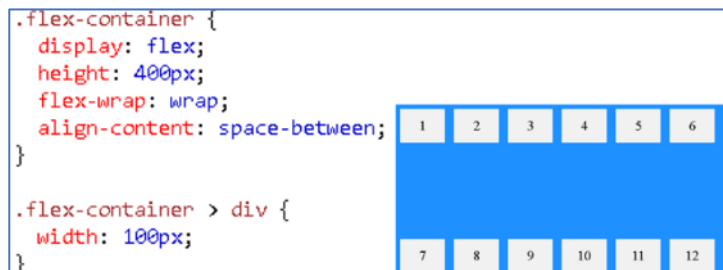
A propriedade definida com o valor **align-content: flex-end** vai e exibir as linhas no final do container.



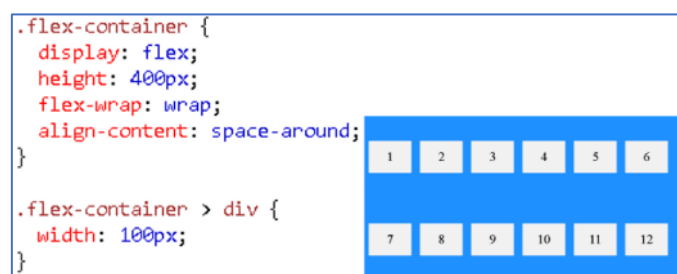
A propriedade definida com o valor **align-content: center** vai e exibir as linhas no centro do container.



A propriedade definida com o valor **align-content:space-between**,
exibe as linhas flexíveis com um espaço igual entre elas.

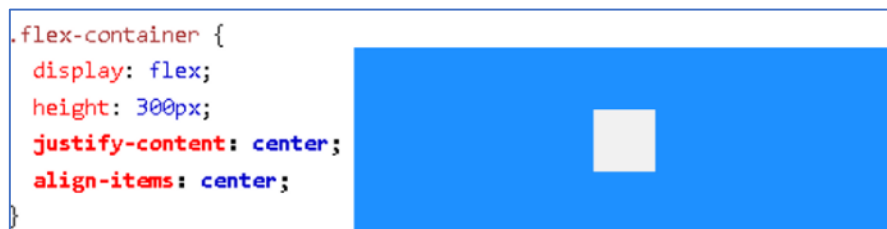


A propriedade definida com o valor **align-content:space-around**,
exibe as linhas flexíveis com espaço antes, entre e depois delas.



Centralização Perfeita Com Flexbox

Definindo as propriedades **justify-content** e **align-items** com o valor **center** o item flexível ficará perfeitamente centralizado no container.



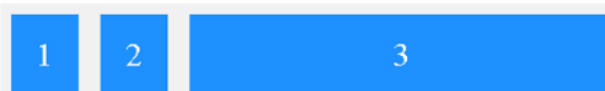
Flex-Items

Os elementos “filhos” de um container com `display:flex` são automaticamente flex-items (flex-items). As propriedades dos flex-items podem ser alteradas.

flex-grow

A propriedade **flex-grow** especifica o quanto um flex-item vai aumentar em relação ao restante dos outros flex-items para preencher o container. No seguinte exemplo o terceiro elemento vai aumentar oito vezes mais que o restante.


```
<div class="flex-container">
  <div style="flex-grow: 1">1</div>
  <div style="flex-grow: 1">2</div>
  <div style="flex-grow: 8">3</div>
</div>
```



flex-shrink

A propriedade **flex-shrink** define se um flex-item vai encolher quando o tamanho da janela for diminuído. A propriedade aceita dois valores, sendo “1” o seu valor padrão e “0” que especifica que o item não vai ser encolhido.

```
<div class="flex-container">
  <div>1</div>
  <div>2</div>
  <div style="flex-shrink: 0">3</div>
</div>
```



order

A propriedade order especifica a ordem que os flex-items devem aparecer dentro do container.

```
<div class="flex-container">
  <div style="order: 3">1</div>
  <div style="order: 2">2</div>
  <div style="order: 4">3</div>
  <div style="order: 1">4</div>
</div>
```



align-self

A propriedade **align-self** especifica o alinhamento de um item selecionado dentro de um container flexível. A propriedade align-self substitui o alinhamento padrão definido pela propriedade align-items do contêiner.

```
.flex-container {
  display: flex;
  height: 200px;
  align-items: stretch;
}

<div class="flex-container">
  <div>1</div>
  <div>2</div>
  <div style="align-self: center">3</div>
  <div>4</div>
</div>
```

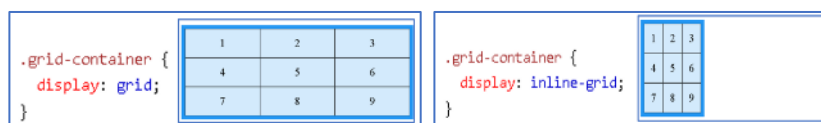


CSS GRID

O módulo de layout GRID CSS oferece um sistema de layout baseado em grade, com linhas e colunas, tornando mais fácil projetar o design de web pages sem utilizar técnicas de objetos flutuantes e de posicionamento.

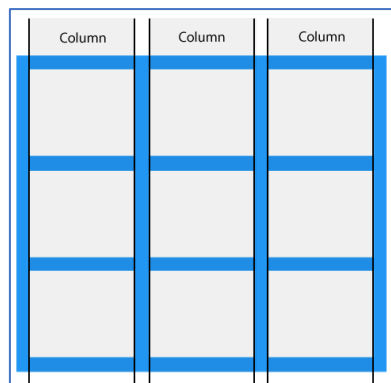
Display Grid

Um elemento HTML se torna um **grid container** quando a propriedade **display** é definida como **grid** ou **inline-grid**. Todos os filhos diretos de um grid container se tornam automaticamente **grid-items**.



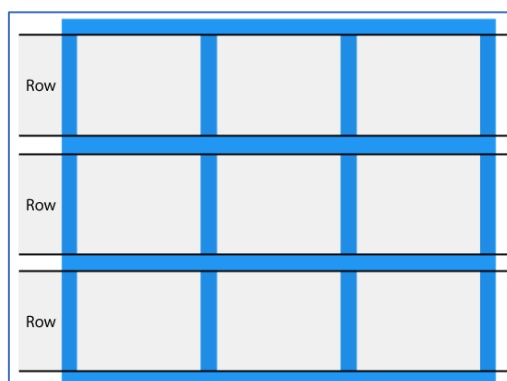
Grid Columns

As fileiras verticais formadas por grid items são chamadas de columns (colunas).



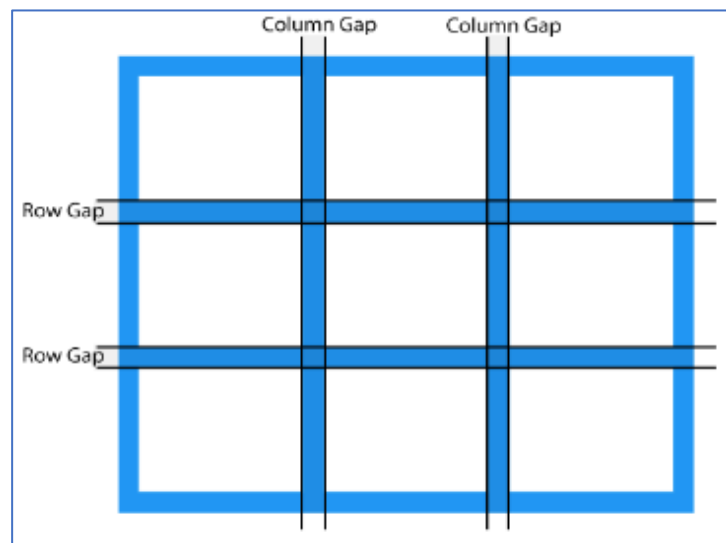
Grid Rows

As fileiras horizontais formadas por grid items são chamadas de rows (linhas).



Grid Gap

Os espaços entre cada column/row são chamados de gaps (vão).

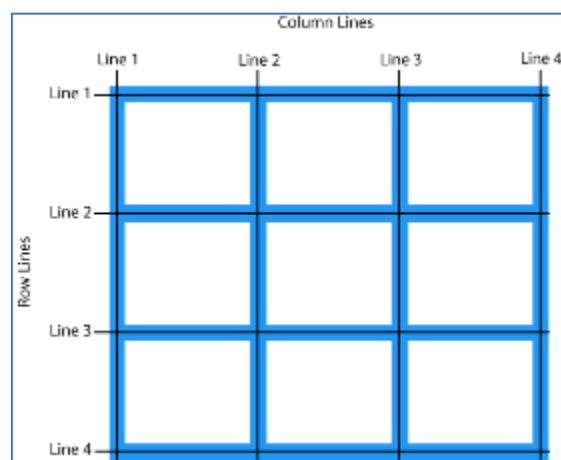


É possível ajustar o tamanho do gap utilizando as seguintes propriedades: **column-gap**, **row-gap** e **gap**.

<pre><code>.grid-container { display: grid; column-gap: 50px; }</code></pre>	<pre><code>.grid-container { display: grid; row-gap: 50px; }</code></pre>	<pre><code>.grid-container { display: grid; gap: 50px 100px; }</code></pre>
--	---	---

Grid Lines

As linhas entre as columns são chamadas de **column-lines**. As linhas entre as rows são chamadas de **row-lines**.



Cada linha possui um número correspondente que será utilizado como referência para posicionar os grid items em um grid container.

No seguinte exemplo o grid item “item1” vai ser posicionado de uma maneira que ele comece na column line 1 e termine na column line 3.

<code>.item1 {</code>	1	2
<code>grid-column-start: 1;</code>	3	4
<code>grid-column-end: 3;</code>	6	7
<code>}</code>	8	

No seguinte exemplo o grid item “item1” vai ser posicionado de uma maneira que ele comece na row line 1 e termine na row line 3.

<code>.item1 {</code>	1	2	3
<code>grid-row-start: 1;</code>		4	5
<code>grid-row-end: 3;</code>	6	7	8
<code>}</code>			

grid-template

A propriedade **grid-template-columns** define o número de colunas no grid layout e permite definir o tamanho de cada coluna. No seguinte exemplo foram definidas 4 colunas no grid layout, os tamanhos (largura) das colunas foram definidos como auto, ou seja, os tamanhos vão ser iguais de acordo com o dimensionamento da janela.


<code>.grid-container {</code>	1	2	3	4
<code>display: grid;</code>	5	6	7	8
<code>grid-template-columns: auto auto auto auto;</code>				
<code>}</code>				

No seguinte exemplo foram definidas 4 colunas, porém, as larguras de algumas colunas foram especificadas, ou seja, o tamanho dessas colunas será mantido independente da alteração do tamanho da janela.

<code>.grid-container {</code>	1	2	3	4
<code>display: grid;</code>	5	6	7	8
<code>grid-template-columns: 80px 200px auto 40px;</code>				
<code>}</code>				

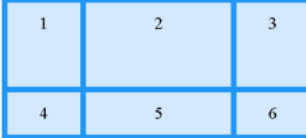
O mesmo se aplica a propriedade **grid-template-rows**, que define o número de rows no grid layout e permite definir o tamanho de cada row. No seguinte exemplo foi definido um grid layout com duas linhas com tamanhos(altura) específicos.

<code>.grid-container {</code>			
<code>display: grid;</code>			
<code>grid-template-rows: 80px 200px;</code>			
<code>}</code>			



Um maneira abreviada para definir a quantidade de rows e columns é através da propriedade **grid-template** utilizando o sinal gráfico “/” para demonstrar a separação dos parâmetros de definição . A definição de grid row vem deve vir antes da definição de grid column.

<code>.grid-container {</code>			
<code>display: grid;</code>			
<code>grid-template: 150px auto / 1fr 2fr 1fr;</code>			
<code>}</code>			



Alinhamento CSS Grid

justify-content

Caso a tamanho total da grid for menor que o container essa propriedade **alinha a grid** dentro de um container (basicamente ajuste do padding). Alinha a grid referente ao eixo inline (**row ou eixo horizontal**).

Valores:

- **start** – alinha a grid para nivelar com o começo do grid container;

<code>justify-content: start;</code>			
1	2	3	
4	5	6	

- **end** – alinha a grid para nivelar com o final do grid container;

<code>justify-content: end;</code>			
			1
			2
			3
			4
			5
			6

- **center** – alinha a grid no centro do grid container;

<code>justify-content: center;</code>			
			1
			2
			3
			4
			5
			6

- **stretch** – redimensiona os grid items para permitir que a grid preencha toda a largura do grid container;

<code>justify-content: stretch;</code>			
1	2	3	
4	5	6	

- **space-around** – posiciona igualmente espaços entre cada grid item (na horizontal) com metade desses espaços nas extremidades;

justify-content: space-around;

1	2	3
4	5	6

- **space-between** – posiciona igualmente espaços entre cada grid item (na horizontal) sem espaços nas extremidades;

justify-content: space-between;

1	2	3
4	5	6

- **space-evenly** – posiciona igualmente espaços entre cada grid item (na horizontal) e nas extremidades;

justify-content: space-evenly;

1	2	3
4	5	6

align-content

Caso a tamanho total da grid for menor que o container essa propriedade **alinha a grid** dentro de um container (basicamente ajuste do padding). Alinha a grid referente ao eixo block (**column ou eixo vertical**).

Valores:

- **start** – alinha a grid para nivelar com o começo do grid container;

align-content: start;

1	2	3
4	5	6

- **end** – alinha a grid para nivelar com o final do grid container;

align-content: end;

1	2	3
4	5	6

- **center** – alinha a grid no centro do grid container;

align-content: center;

1	2	3
4	5	6

- **stretch** – redimensiona os grid items para permitir que a grid preencha toda a largura do grid container;

align-content: stretch;

1	2	3
4	5	6

- **space-around** – posiciona igualmente espaços entre cada grid item (na vertical) com metade desses espaços nas extremidades;

align-content: space-around;

1	2	3
4	5	6

- **space-between** – posiciona igualmente espaços entre cada grid item (na vertical) sem espaços nas extremidades;

align-content: space-between;

1	2	3
4	5	6


- **space-evenly** – posiciona igualmente espaços entre cada grid item (na vertical) e nas extremidades;

align-content: space-evenly;

1	2	3
4	5	6

place-content

A propriedade **place-content** define as propriedades **align-items** e **justify-items** em uma única declaração.

<pre><code>.grid-container { display: grid; place-content: center; }</code></pre>	
---	---

justify-items

Alinha **todos os itens** dentro de uma célula. Alinha os grid items referente ao eixo inline (**row ou eixo horizontal**).

Valores:

- **start** – alinha os itens para nivelar com o começo da célula;

justify-items: start;

1	2	3
4	5	6

- **end** – alinha os itens para nivelar com o final da célula;

justify-items: end;

	1		2		3
	4		5		6

- **center** – alinha a grid no centro da célula;

justify-items: center;

	1		2		3
	4		5		6

- **stretch** –(default) redimensiona os grid items para preencher todo espaço da célula;

justify-items: stretch;

1	2	3
4	5	6

align-items

Alinha **todos os itens** dentro de uma célula (basicamente ajuste do margin). Alinha os grid-items referente ao eixo block (**column ou eixo vertical**).

Valores:

- **start** – alinha os itens para nivelar com o começo da célula;

align-items: start;

1	2	3
4	5	6

- **end** – alinha os itens para nivelar com o final da célula;

align-items: end;

1	2	3
4	5	6

- **center** – alinha a grid no centro da célula;

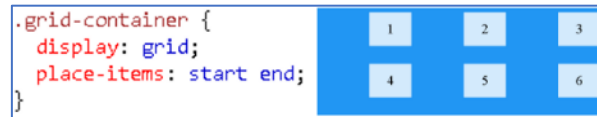
align-items: center;

1	2	3
4	5	6

- **stretch** – (default) redimensiona os grid items para preencher todo espaço da célula;
- **baseline** – alinhar itens ao longo da linha de base do texto.

place-items

A propriedade **place-items** define as propriedades **align-items** e **justify-items** em uma única declaração.

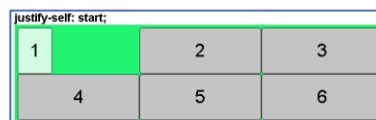


justify-self

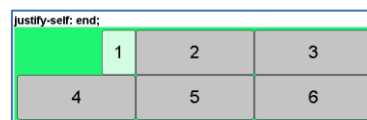
Alinha **um item** dentro de uma célula (basicamente ajuste do margin).
Alinha o grid item referente ao eixo inline (**row ou eixo horizontal**).

Valores:

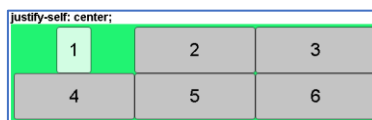
- **start** – alinha o grid item para nivelar com o começo da célula;



- **end** – alinha o grid item para nivelar com o final da célula;



- **center** – alinha o grid item no centro da célula;



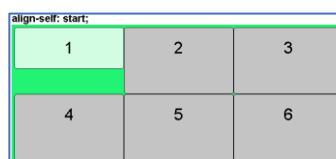
- **stretch** – (default) redimensiona o grid item para preencher todo espaço da célula;

align-self

Alinha **um item** dentro de uma célula (basicamente ajuste do margin).
Alinha o grid item referente ao eixo block (**column ou eixo vertical**).

Valores:

start – alinha o grid item para nivelar com o começo da célula;



end – alinha o grid item para nivelar com o final da célula;

align-self: end;

1	2	3
4	5	6

center – alinha o grid item no centro da célula;

align-self: center;

1	2	3
4	5	6

stretch – (default) redimensiona o grid item para preencher todo espaço da célula;

place-self

A propriedade **place-self** define as propriedades **align-self** e **justify-self** em uma única declaração.

1	2	3
4	5	6

place-self: start end;

Posicionamento CSS Grid

grid-column

Os grid-items podem ser posicionados verticalmente através de duas propriedades, **grid-column-start** e **grid-column-end**. A propriedade **grid-column** é uma maneira encurtada de definir essas propriedades. Primeiro é definido a column-line onde o item começará e após o uso do sinal de “/” é definido a column-line onde o item irá terminar. O posicionamento do grid-item é feito utilizando o número correspondente a column-line desejada, ou utilizando “span” que define quantas columns o item vai expandir.

<code>.item1 {</code>	1	2	3
<code> grid-column: 1 / 5;</code>	4	5	6
<code>}</code>	7	8	9
	10	11	12
	13	14	15

<code>.item1 {</code>	1	2	3	4
<code> grid-column: 1 / span 3;</code>	5	6	7	8
<code>}</code>	9	10	11	12
	13	14	15	16

grid-row

Os grid-items são posicionados horizontalmente através de duas propriedades, **grid-row-start** e **grid-row-end**. A propriedade **grid-row** é uma maneira encurtada de definir essas propriedades. Primeiro é definido a row-line onde o item começará e após o uso do sinal de “/” é definido a row-line onde o item irá terminar. O posicionamento do grid-item é feito utilizando o número correspondente a row-line desejada, ou utilizando “span” que define quantas rows o item vai expandir. No seguinte exemplo o “item1” começa na row-line 1 e vai até a row-line 4.

```

.item1 {
  grid-row: 1 / 4;
}

```

1	2	3	4	5	6
	7	8	9	10	11
	12	13	14	15	16

```

.item1 {
  grid-row: 1 / span 2;
}

```

1	2	3	4	5	6
	7	8	9	10	11
	12	13	14	15	16
					17

grid-area

A propriedade **grid-area** é uma maneira encurtada de definir de as propriedades **grid-row-start**, **grid-column-start**, **grid-row-end** e **grid-column-end**. Primeiro é definido a row-line e column-line onde o item começará e depois é definido a row-line e column-line onde o item irá terminar. O posicionamento do grid-item é feito utilizando o número correspondente a line desejada, ou utilizando “span” que define quantas columns ou rows o item vai expandir. No seguinte exemplo o “item8” começa na row-line 1 e column-line 2, e então vai até a row-line 5 e column-line 6.

```
.item8 {  
  grid-area: 1 / 2 / 5 / 6;  
}
```

1	8				2
3					4
5					6
7					9
10	11	12	13	14	15

No seguinte exemplo o “item8” começa na row-line 2 e column-line 1, e expande 2 rows e 3 columns.

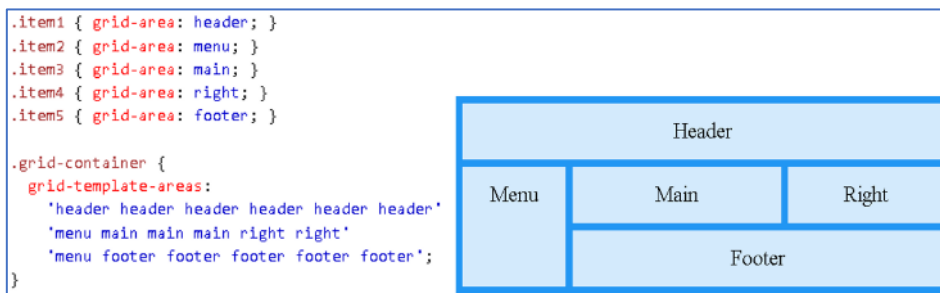
```
.item8 {  
  grid-area: 2 / 1 / span 2 / span 3;  
}
```

1	2	3	4	5	6
8			7	9	10
			11	12	13

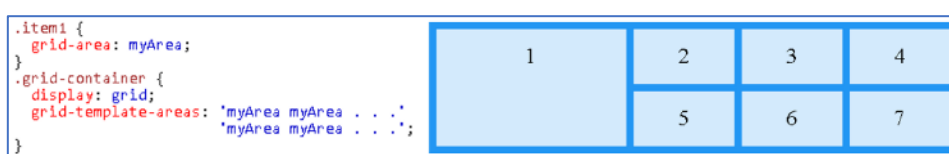
grid-template-areas

A propriedade **grid-template-areas** também pode ser usada para atribuir nomes aos grid-items, os itens nomeados podem então ser referenciados através da propriedade **grid-template-areas** no grid container. Cada row é definida por

apóstrofes (' '). As columns em cada row são definidas dentro dos apóstrofos, separadas por um espaço.



Um sinal de ponto “.” representa um grid item sem nome



TIPOS DE ALINHAMENTOS CSS

As propriedades de alinhamento de caixa no CSS são definidas como 6 propriedades que controlam o alinhamento de caixas dentro de outras caixas (containers). Elas podem ser classificadas como: Em qual dimensão/eixo a propriedade se refere (main/inline ou cross/block); O controle do posicionamento está relacionado a caixa dentro do parent, ou relacionado ao conteúdo dentro da própria caixa. A seguinte imagem explica isso.

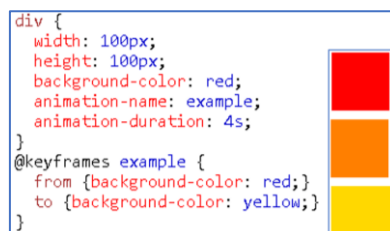
Common	Axis	Aligns	Applies to
‘justify-content’	main/inline	content within element	block containers , flex containers , and grid containers
‘align-content’	cross/block	element (effectively adjusts padding)	
‘justify-self’	inline	element within parent	block-level boxes, absolutely-positioned boxes, and grid items
‘align-self’	cross/block	parent (effectively adjusts margins)	
‘justify-items’	inline	items inside box	block containers and grid containers
‘align-items’	cross/block	child items (controls child items’ ‘justify-self: auto’)	

CSS ANIMATIONS

CSS permite animar elemento HTML sem usar JavaScript ou flash, um elemento é animado quando ele gradualmente vai de um estilo para outro, é possível alterar muitas propriedades CSS de muitas maneiras. Para usar animações CSS é preciso de alguns keyframes específicos. Os keyframes contém os estilos que um elemento vai possuir em um certo momento.

@keyframes

Quando um estilo é especificado dentro de uma regra @keyframes, a animação vai mudar gradualmente para do estilo atual para o novo estilo em um determinado momento. Para fazer a animação funcionar é preciso vincular a animação a um elemento. No seguinte exemplo a animação “example” é vinculada ao elemento <div> através da propriedade **animation-name**. A animação vai terminar após 4 segundos, e vai gradualmente alterar a propriedade background-color do elemento <div> de “red” para “yellow”.



A propriedade **animation-duration** define **quanto tempo vai durar a animação** até que ela esteja completa. Se essa propriedade não for especificada, nenhuma animação vai acontecer, já que o valor padrão é 0s. Para representar os estágios da animação é possível utilizar “from” e “to” dentro do keyframes. Porém o uso de valores em porcentagem permite adicionar mais estágios a animação. No seguinte exemplo o background-color vai ser alterado do elemento <div> quando a animação estiver 25% completa, 50% completa, e novamente quando a animação for 100% completa.

```
div {
  width: 100px;
  height: 100px;
  background-color: red;
  animation-name: example;
  animation-duration: 4s;
}
@keyframes example {
  0% {background-color: red;}
  25% {background-color: yellow;}
  50% {background-color: blue;}
  100% {background-color: green;}
}
```

animation-delay

A propriedade animation-delay especifica o tempo de atraso para a animação ser iniciada. É possível ainda acrescentar valores negativos, esses valores fazem a animação iniciar como se já estivesse sendo reproduzida por N segundos.

animation-iteration-count

A propriedade animation-iteration-count especifica o número de vezes em que a animação vai ser executada. No seguinte exemplo a animação vai ser executada 3 vezes, e após isso parar. No outro exemplo é utilizado o valor "infinite" para fazer a animação continuar para sempre:

<pre>div { width: 100px; height: 100px; position: relative; background-color: red; animation-name: example; animation-duration: 4s; animation-iteration-count: 3; }</pre>	<pre>div { width: 100px; height: 100px; position: relative; background-color: red; animation-name: example; animation-duration: 4s; animation-iteration-count: infinite; }</pre>
---	--

animation-direction

A propriedade animation-direction especifica se a animação vai ser executada de forma progressiva (para frente), regressiva (para trás) ou em ciclos alternados. Para o valor "alternate" funcionar é preciso que a propriedade animation-iteration-count esteja definida com um valor maior ou igual a dois. O exemplo a seguir usa o valor "alternate" fazendo com que a animação seja executada e depois executada novamente de forma inversa.

```
div {
  width: 100px;
  height: 100px;
  position: relative;
  background-color: red;
  animation-name: example;
  animation-duration: 4s;
  animation-iteration-count: 2;
  animation-direction: alternate;
}
```

A propriedade animation-direction pode ter os seguintes valores:

- **normal**
- **reverse**
- **alternate**
- **alternate-reverse**

animation-fill-mode

A propriedade **animation-fill-mode** especifica o estilo do elemento quando a animação não está sendo executada (antes de começar, após ela terminar e ambos). Possui os seguintes valores:

- **none - default.** A animação não aplicará nenhum estilo ao elemento antes ou depois de ser executado
- **forwards** - O elemento manterá os valores de estilo definidos pelo **último keyframe** (dependendo das propriedades animation-direction e da animation-iteration-count)
- **backwards** - O elemento receberá os valores de estilo definidos pelo **primeiro keyframe** (dependendo da animation-direction) manterá esse estilo durante o tempo de delay.
- **both** - A animação fará as duas regras anteriores.

Sem o uso do animation-fill-mode o elemento vai começar e terminar a animação de maneira brusca, com os estilos padrões sendo atribuídos ao elemento. No seguinte exemplo a animação vai manter a última alteração de estilo definida no keyframe.

```
div {
  width: 100px;
  height: 100px;
  background: red;
  position: relative;
  animation-name: example;
  animation-duration: 3s;
  animation-delay: 2s;
  animation-fill-mode: backwards;
}
```

animation-timing-function

A propriedade **animation-timing-function** especifica a **curva de velocidade da animação**. Possui os seguintes valores:

- **ease** – default. Especifica uma animação que vai começar devagar, ficar rápida, e terminar devagar.
- **linear** – Especifica uma animação com a mesma velocidade do começo até o fim.
- **ease-in** – Especifica uma animação com uma velocidade devagar no começo.
- **ease-out** - Especifica uma animação com uma velocidade devagar no final.
- **ease-in-out** - Especifica uma animação com uma velocidade devagar no começo e no final.
- **cubic-bezier(n,n,n,n)** – Permite especificar os valores da velocidade em uma função.

animation shorthand

A propriedade **animation** é uma **maneira encurtada de usar todas as propriedades anteriores** em uma única linha de código. No seguinte exemplo utiliza 6 propriedades de animação diferentes.

```
div {
  animation-name: example;
  animation-duration: 5s;
  animation-timing-function: linear;
  animation-delay: 2s;
  animation-iteration-count: infinite;
  animation-direction: alternate;
}
```

O mesmo efeito de animação acima pode ser obtido usando a propriedade de animation abreviada.

```
div {
  animation: example 5s linear 2s infinite alternate;
}
```

```
div {
  width: 100px;
  height: 100px;
  background-color: red;
  position: relative;
  animation: myfirst 5s linear 2s infinite alternate;
}

@keyframes myfirst {
  0% {background-color:red; left:0px; top:0px;}
  25% {background-color:yellow; left:200px; top:0px;}
  50% {background-color:blue; left:200px; top:200px;}
  75% {background-color:green; left:0px; top:200px;}
  100% {background-color:red; left:0px; top:0px;}
}
```

CSS TRANSITIONS

As transições CSS permitem alterar os valores de propriedade de forma suave durante um determinado período de duração. Se o tempo de duração não for especificado a transição não vai ter nenhum efeito pois o valor default é 0. No seguinte exemplo um elemento <div> com dimensões de 100px x 100px recebe uma alteração no valor da propriedade width quando a pseudoclassee (“hover”) do elemento é ativada. A transição de desse valor é feita de forma gradual e suave devido a propriedade transition.

```
div {
  width: 100px;
  height: 100px;
  background: red;
  transition: width 2s;
}
div:hover {
  width: 300px;
}
```


transition-delay

A propriedade **transition-delay** especifica um atraso (em segundos) no efeito de transição.

```
div {
  width: 100px;
  height: 100px;
  background: red;
  transition: width 3s;
  transition-delay: 1s;
}

div:hover {
  width: 300px;
}
```

transition-timing-function

A propriedade **transition-timing-function** especifica a curva de velocidade em que a transição vai acontecer, assim como nas animações.

- **ease** – default. Especifica uma transição que vai começar devagar, ficar rápida, e terminar devagar.
- **linear** – Especifica uma transição com a mesma velocidade do começo até o fim.
- **ease-in** – Especifica uma transição com uma velocidade devagar no começo.
- **ease-out** - Especifica uma transição com uma velocidade devagar no final.
- **ease-in-out** - Especifica uma transição com uma velocidade devagar no começo e no final.
- **cubic-bezier(n,n,n,n)** – Permite especificar os valores da velocidade em uma função.

transition shorthand

No seguinte exemplo a transição de estilo foi especificada em 4 diferentes propriedades:

```
div {
  transition-property: width;
  transition-duration: 2s;
  transition-timing-function: linear;
  transition-delay: 1s;
}
```

É possível simplificar e determinar todas essas propriedades em apenas uma:

```
div {
  transition: width 2s linear 1s;
}
```

(As propriedades devem ser definidas nessa sequência: transition-property, transition-duration, transition-timing-function e transition-delay)

É possível ainda especificar a transição de alteração de diversas propriedades em tempos diferentes, adicionando o sinal gráfico de vírgula “,” após especificada o nome cada propriedade e seus devidos tempos de duração, como no seguinte exemplo.

```
div {  
  width: 100px;  
  height: 100px;  
  background: red;  
  transition: width 2s, height 2s, transform 2s;  
}  
  
div:hover {  
  width: 300px;  
  height: 300px;  
  transform: rotate(180deg);  
}
```

```
div {  
  width: 100px;  
  height: 100px;  
  background: red;  
  transition: width 2s linear 1s, height 4s ease 2s;  
}  
  
div:hover {  
  width: 300px;  
  height: 300px;  
}
```

REFERÊNCIAS:

- <https://www.w3schools.com>
- <https://developer.mozilla.org>
- <https://www.w3.org>
- <https://www.alura.com.br>
- <https://www.hostinger.com.br>
- <https://www.maujor.com>
- <https://alistapart.com>