INSTITUTO FEDERAL
Paraná
Campus Paranaguá

Ministério da Educação

ENSINO MÉDIO INTEGRADO: INFORMÁTICA
Disciplina de Desenvolvimento Web
Aula 04: CSS / Conceitos Avançados

Gil Eduardo de Andrade

Conceitos Preliminares

CSS - Cascading Style Sheets (https://www.w3schools.com/html/)

CSS é a sigla para o termo em inglês Cascading Style Sheets (Folha de Estilo em Cascatas), sendo utilizado para estilizar elementos escritos na linguagem HTML. O CSS foi desenvolvido pelo W3C (World Wide Web Consortium - https://www.w3.org/Style/CSS/) em 1996, com intuito de dar suporte a definição de estilos dentro do HTML, que não foi projetado com tags voltadas a formatar e estilizar uma página.

Transformação 2D

Os efeitos de transformação 2D, propriedade "*transform*", possibilitam que elementos HTML sejam movidos, girados, inclinados e redimensionados. Juntamente com a propriedade "*transform*" podem ser utilizados os seguintes métodos de transformação 2D:

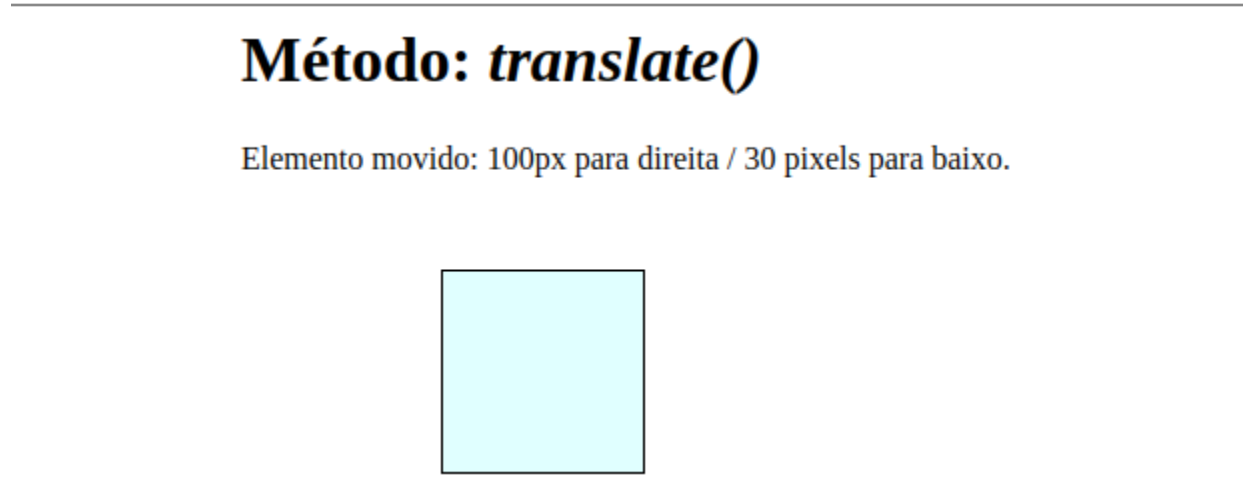
- *translate()*
- *rotate()*
- *scaleX()*
- *scaleY()*
- *scale()*
- *skewX()*
- *skewY()*
- *skew()*
- *matrix()*

Método "*translate()*"

O método “*translate()*” permite mover um elemento, a partir da sua posição atual, nos eixos X e Y.

Arquivo: ./css/exemplos/transform2D/ex_2Dtransform01.html

```
<!DOCTYPE html>
<html lang="pt-br">
   <head>
       <meta charset="UTF-8">
       <style>
           div {
               width: 100px;
               height: 100px;
               background-color: lightcyan;
               border: 1px solid black;
               transform: translate(100px, 30px);
           }
       </style>
   </head>
   <body>
       <h1>Método: <i>translate()</i></h1>
       <p>Elemento movido: 100px para direita / 30 pixels para baixo.</p>
       <div> </div>
   </body>
</html>
```



Método "*rotate()*"

O método "*rotate()*" permite girar um elemento, no sentido horário ou anti-horário, a partir de um determinado grau especificado.

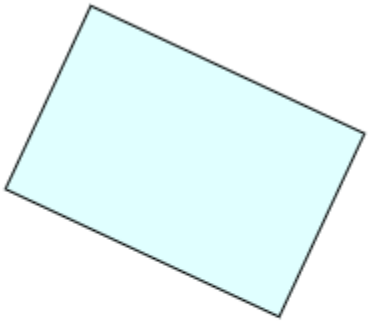
Arquivo: ./css/exemplos/transform2D/ex_2Dtransform02.html

```
<!DOCTYPE html>
<html lang="pt-br">
   <head>
       <meta charset="UTF-8">
       <style>
           p {
               margin-bottom: 30px;
           }
           p.marg {
               margin-top: 40px;
           }
           div {
               margin-left: 30px;
               width: 150px;
               height: 100px;
               background-color: lightcyan;
               border: 1px solid black;
           }
           div.rotate01 {
               transform: rotate(25deg);
           }
           div.rotate02 {
               transform: rotate(-25deg);
           }
       </style>
   </head>
   <body>
       <h1>Método: <i>rotate()</i></h1>
       <p>Elemento rotacionado: 30° horário</p>
       <div class="rotate01"></div>
       <p class="marg">Elemento rotacionado: 30° anti-horário</p>
       <div class="rotate02"></div>
```

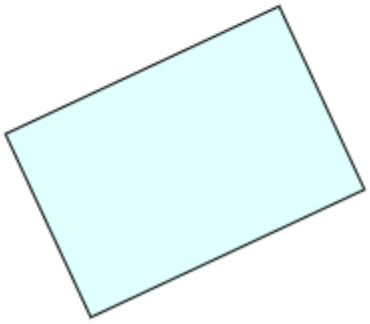
```
    </body>
</html>
```

Método: *rotate()*

Elemento rotacionado: 30º horário



Elemento rotacionado: 30º anti-horário



Método “*scale()*”

O método “*scale()*” permite redimensionar (aumentar / diminuir) o tamanho de um elemento HTML, de acordo com os parâmetros fornecidos de largura e altura.

Arquivo: ./css/exemplos/transform2D/ex_2Dtransform03.html

```
<!DOCTYPE html>
<html lang="pt-br">
    <head>
        <meta charset="UTF-8">
        <style>
            div {
                margin-left: 60px;
```

```
            margin-top: 60px;
            width: 100px;
            height: 50px;
            background-color:lightblue;
            border: 1px solid black;
        }
        div.scale {
            transform: scale(2, 3);
        }
        div.scale01 {
            margin-top: -20px;
            transform: scale(0.5, 0.5);
        }
    </style>
</head>
<body>
    <h1>Método: <i>scale()</i></h1>
    <p>Aumentou 2x a largura e 3x a altura do elemento</p>
    <div class="scale"></div>
        <p style="margin-top: 70px;">Reduziu a largura e a altura pela
metade</p>
    <div class="scale01"></div>
</body>
</html>
```

Método: *scale()*

Aumentou 2x a largura e 3x a altura do elemento

Reduziu a largura e a altura pela metade

Transições

Os efeitos de transição possibilitam que os valores de determinadas propriedades de um elemento HTML sejam suavemente alterados. Neste documento iremos ver as seguintes propriedades:

- *transition*
- *transition-timing-function*
- *transition-delay*

Para criarmos uma transição é necessário especificar duas coisas:

- A propriedade CSS que iremos aplicar um efeito;
- A duração do efeito;

No exemplo a seguir a propriedade CSS "*width*", do elemento , é especificada para aplicação do efeito, com duração de 3 segundos.

Arquivo: ./css/exemplos/transition/ex_transition01.html

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
   <meta charset="UTF-8">
   <style>
       div {
           width: 100px;
           height: 100px;
           background: red;
           background-image: linear-gradient(
                   to right, red, orange, yellow, green, blue, indigo, violet);
           transition: width 3s;
       }
       div:hover {
           width: 300px;
       }
```

```
   </style>
</head>
<body>
   <h1>Propriedade <i>transition</i></h1>
   <p>Passe o mouse para verificar o efeito de transição!</p>
   <div></div>
</body>
</html>
```

Propriedade *transition*

Passe o mouse para verificar o efeito de transição!

O exemplo a seguir mostra o efeito de transição para ambas as propriedades: *width* e *height*, com duração de 3 e 4 segundos, respectivamente.

Arquivo: ./css/exemplos/transition/ex_transition02.html

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
   <meta charset="UTF-8">
   <style>
       div {
           width: 100px;
           height: 100px;
           background: red;
           background-image: linear-gradient(
                   to right, red, orange, yellow, green, blue, indigo, violet);
           transition: width 3s, height 4s;
       }
```

```
        div:hover {
            width: 350px;
            height: 350px;
        }
    </style>
</head>
<body>
    <h1>Propriedade <i>transition</i></h1>
    <p>Passe o mouse para verificar o efeito de transição!</p>
    <div></div>
</body>
</html>
```

Propriedade *transition*

Passe o mouse para verificar o efeito de transição!

Especificando a Curva de Velocidade

A propriedade "*transition-timing-function*" permite especificar a curva de velocidade do efeito de transição. Ela pode receber os seguintes valores:

- **ease** - especifica um efeito de transição com um início lento, depois rápido e termina lentamente (padrão);
- **linear** - especifica um efeito de transição com a mesma velocidade do início ao fim;
- **ease-in** - especifica um efeito de transição com um início lento;
- **ease-out** - especifica um efeito de transição com um fim lento;
- **ease-in-out** - especifica um efeito de transição com um início e fim lentos;
- **cubic-bezier(n,n,n,n)** - permite que você defina seus próprios valores em uma função cubic-bezier

Arquivo: ./css/exemplos/transition/ex_transition03.html

```
<!DOCTYPE html>
<html lang="pt-br">
   <head>
       <meta charset="UTF-8">
       <style>
           div {
               width: 100px;
               height: 100px;
               background: red;
               background-image: linear-gradient(
                   to right, red, orange, yellow, green, blue, indigo, violet);
               transition: width 2s;
           }

           #div1 {transition-timing-function: linear;}
           #div2 {transition-timing-function: ease;}
           #div3 {transition-timing-function: ease-in;}
           #div4 {transition-timing-function: ease-out;}
           #div5 {transition-timing-function: ease-in-out;}

           div:hover {
               width: 300px;
           }
       </style>
   </head>
   <body>
       <h1>Propriedade <i>transition-timing-function</i></h1>
       <p>Passe o mouse para verificar as diferentes velocidades!</p>
       <div id="div1">linear</div><br>
       <div id="div2">ease</div><br>
       <div id="div3">ease-in</div><br>
       <div id="div4">ease-out</div><br>
       <div id="div5">ease-in-out</div><br>
   </body>
</html>
```

Propriedade *transition-timing-function*

Passe o mouse para verificar as diferentes velocidades!

Atraso no Efeito de Transição

A propriedade "*transition-delay*" permite especificar um tempo de atraso, em segundos, para que o efeito seja iniciado.

Arquivo: ./css/exemplos/transition/ex_transition04.html

```
<!DOCTYPE html>
<html lang="pt-br">
   <head>
       <meta charset="UTF-8">
       <style>
           div {
               width: 100px;
               height: 100px;
               background: red;
                background-image: linear-gradient(to right, red, orange, yellow,
green, blue, indigo, violet);
               transition: width 3s;
               transition-delay: 1s;
           }

           div:hover {
               width: 300px;
           }
       </style>
   </head>
   <body>
```

```
        <h1>Propriedade <i>transition-delay</i></h1>
        <p>Passe o mouse para verificar o atraso no início do efeito!</p>
        <div></div>
    </body>
</html>
```

Propriedade *transition-delay*

Passe o mouse para verificar o atraso no início do efeito!

Transição + Transformação

O CSS permite que os efeitos de transição (*transition*) e transformação (*transformation*) sejam utilizados em conjunto.

Arquivo: ./css/exemplos/transition/ex_transition05.html

```
<!DOCTYPE html>
<html lang="pt-br">
    <head>
        <meta charset="UTF-8">
        <style>
            div {
                width: 100px;
                height: 100px;
                background: red;
                background-image: linear-gradient(
                    to right, red, orange, yellow, green, blue, indigo, violet);
                transition: width 1s, height 1s, transform 2s;
            }
            div:hover {
                width: 300px;
```

```
            height: 300px;
            transform: rotate(180deg);
        }
    </style>
</head>
<body>
    <h1>Propriedades: <i>transition</i> + <i>transform</i></h1>
    <p>Passe o mouse para verificar a combinação de efeitos!</p>
    <div></div>
</body>
</html>
```

Propriedades: *transition* + *transform*

Passe o mouse para verificar a combinação de efeitos!

Especificando Propriedades de Transição

O CSS permite também, que as propriedades de transição sejam especificadas uma a uma, como no exemplo apresentado a seguir:

Arquivo: ./css/exemplos/transition/ex_transition06.html

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <style>
        div {
            width: 100px;
            height: 100px;
            background: red;
```

```
            background-image: linear-gradient(
                    to right, red, orange, yellow, green, blue, indigo, violet);
            transition-property: width;
            transition-duration: 2s;
            transition-timing-function: linear;
            transition-delay: 1s;
            /* OU -> transition: width 2s linear 1s;*/
        }
        div:hover {
            width: 350px;
        }
    </style>
</head>
<body>
    <h1>Propriedades Detalhadas </h1>
    <p>Passe o mouse para verificar o efeito de transição!</p>
    <div></div>
</body>
</html>
```

Propriedades Detalhadas

Passe o mouse para verificar o efeito de transição!

Animações

O CSS permite que animações sejam feitas, para elementos HTML, sem que o uso da linguagem JavaScript seja necessário. Neste documento iremos trabalhar com as seguintes propriedades:

- *@keyframes*

- *animation-name*
- *animation-duration*
- *animation-delay*
- *animation-iteration-count*
- *animation-direction*
- *animation-timing-function*
- *animation-fill-mode*
- *animation*

As animações, basicamente, permitem que os elementos mudem, gradualmente, de um estilo para outro, possibilitando que qualquer propriedade CSS seja mudada a qualquer momento. Para utilizar as animações CSS é necessário, primeiramente, alguns quadros chaves (*keyframes*) da animação. Os quadros chave especificam estilos que um elemento deverá possuir em determinados momentos.

Regra *@Keyframes*

Ao especificarmos um estilo CSS dentro de uma regra "*@keyframes*", a animação muda seu estilo atual para o novo estilo em determinados momentos. Para que a animação funcione, ela deve ser vinculada a um elemento HTML.

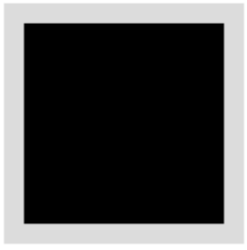
No exemplo a seguir, a animação "cores" dura 4 segundos, está vinculada ao elemento , que muda de cor, gradualmente, do preto para o cinza internamente, e do cinza para o preto nas bordas.

Arquivo: ./css/exemplos/animation/ex_animation01.html

```
<!DOCTYPE html>
<html>
   <head>
       <meta charset="UTF-8">
       <style>
           div {
               width: 100px;
               height: 100px;
               border: 10px solid;
               border-color: gainsboro;
               background-color: black;
```

```
            animation-name: cores;
            animation-duration: 4s;
        }
        @keyframes cores {
            from {
                background-color: black;
                border-color: gainsboro;
            }
            to {
                background-color: gainsboro;
                border-color: black;
            }
        }
    </style>
</head>
<body>
    <h1>Animação CSS</h1>
        <div></div>
        <p>Ao final da animação, volta ao estilo original.</p>
</body>
</html>
```

Animação CSS



Ao final da animação, volta ao estilo original.

Nota: A propriedade "*animation-duration*" define o tempo de duração da animação. Caso ela não seja definida, nenhuma animação será efetuada, porque o valor padrão é 0 segundo.

No exemplo especificamos quando o estilo mudará usando as palavras-chave "***from***" e "***to***", que representam 0% (início) e 100% (término). Também é possível usar percent, que permite adicionar quantas alterações de estilo desejar.

O exemplo a seguir mudará o "*background-color*" do elemento quando a animação estiver 25% concluída, 50% concluída e novamente quando a animação estiver 100% concluída:

Arquivo: ./css/exemplos/animation/ex_animation02.html

```
<!DOCTYPE html>
<html>
   <head>
       <meta charset="UTF-8">
       <style>
           div {
               width: 100px;
               height: 100px;
               border: 5px solid;
               border-color: black;
               background-color: black;
               animation-name: cores;
               animation-duration: 6s;
           }
           @keyframes cores {
               0%   {background-color: black;}
               25%  {background-color: gray;}
               50%  {background-color: burlywood;}
               100% {background-color: beige;}
           }
       </style>
   </head>
   <body>
       <h1>Animação CSS</h1>
           <div></div>
           <p>Ao final da animação, volta ao estilo original.</p>
   </body>
</html>
```

Animação CSS

Ao final da animação, volta ao estilo original.

No exemplo abaixo a animação irá alterar a cor de fundo e a posição do elemento , quando estiver 25%, 50% e 100% concluída:

Arquivo: ./css/exemplos/animation/ex_animation03.html

```
<!DOCTYPE html>
<html>
   <head>
       <meta charset="UTF-8">
       <style>
           div {
               width: 100px;
               height: 100px;
               background-color: black;
               position: relative;
               animation-name: cores;
               animation-duration: 6s;
           }
           @keyframes cores {
               0%   {background-color: black; left:0px; top:0px;}
               25%  {background-color: gray; left:200px; top:0px;}
               50%  {background-color: brown; left:200px; top:200px;}
               75%  {background-color: burlywood; left:0px; top:200px;}
               100% {background-color: beige; left:0px; top:0px;}
           }
       </style>
```

```
    </head>
    <body>
        <h1>Animação CSS</h1>
            <div></div>
    </body>
</html>
```

Animação CSS

Animação com Atraso (*delay*)

A propriedade "*animation-delay*" permite especificar um atraso para que a animação seja iniciada. No exemplo abaixo temos uma atraso de 2 segundos para o início da animação:

Arquivo: ./css/exemplos/animation/ex_animation04.html

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <style>
            div {
                width: 100px;
                height: 100px;
                background-color: black;
                position: relative;
                animation-name: cores;
                animation-duration: 6s;
                animation-delay: 2s;
            }
```

```
            @keyframes cores {
                0%   {background-color: black; left:0px; top:0px;}
                25%  {background-color: gray; left:200px; top:0px;}
                50%  {background-color: brown; left:200px; top:200px;}
                75%  {background-color: burlywood; left:0px; top:200px;}
                100% {background-color: beige; left:0px; top:0px;}
            }
        </style>
    </head>
    <body>
        <h1>Animação CSS</h1>
            <div></div>
    </body>
</html>
```

Animação CSS

A propriedade "*animation-delay*" também permite o uso de valores negativos, como, por exemplo, -2 segundos. Neste caso a animação se inicia como se 2 segundos já tivessem sido passados.

Arquivo: ./css/exemplos/animation/ex_animation05.html

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <style>
            div {
                width: 100px;
                height: 100px;
```

```
            background-color: black;
            position: relative;
            animation-name: cores;
            animation-duration: 6s;
            animation-delay: -1.5s;
        }
        @keyframes cores {
            0%   {background-color: black; left:0px; top:0px;}
            25%  {background-color: gray; left:200px; top:0px;}
            50%  {background-color: brown; left:200px; top:200px;}
            75%  {background-color: burlywood; left:0px; top:200px;}
            100% {background-color: beige; left:0px; top:0px;}
        }
    </style>
</head>
<body>
    <h1>Animação CSS</h1>
        <div></div>
</body>
</html>
```

Animação CSS

Animação Executada Várias Vezes

A propriedade "*animation-iteration-count*" possibilita especificar o número de vezes que uma animação deve ser executada. No exemplo abaixo a animação será executada 2 vezes antes de parar.

Arquivo: ./css/exemplos/animation/ex_animation06.html

```
<!DOCTYPE html>
```

```
<html>
   <head>
       <meta charset="UTF-8">
       <style>
           div {
               width: 100px;
               height: 100px;
               background-color: black;
               position: relative;
               animation-name: cores;
               animation-duration: 4s;
               animation-delay: 1s;
               animation-iteration-count: 2;
                /*animation-iteration-count: infinite;*/
           }
           @keyframes cores {
               0%   {background-color: black; left:0px; top:0px;}
               25%  {background-color: gray; left:200px; top:0px;}
               50%  {background-color: brown; left:200px; top:200px;}
               75%  {background-color: burlywood; left:0px; top:200px;}
               100% {background-color: beige; left:0px; top:0px;}
           }
       </style>
   </head>
   <body>
       <h1>Animação CSS</h1>
           <div></div>
   </body>
</html>
```

Animação CSS

OBS.: podemos especificar que a animação deve ser executada permanentemente, para isso utilizamos o valor "*infinite*", ou seja, `animation-iteration-count: infinite;`

Animação Executada em Várias Direções

A propriedade "*animation-direction*" possibilita especificar se uma animação deve ser reproduzida para frente, para trás, ou em ciclos alternados. Ela pode conter os seguintes valores:

- ***normal*** - A animação é reproduzida normalmente, para frente (padrão);
- ***reverse*** - A animação é reproduzida na direção reversa, para trás;
- ***alternate*** - A animação é reproduzida primeiro para frente, depois para trás;
- ***alternate-reverse*** - A animação é reproduzida primeiro para trás primeiro, depois para frente;

O exemplo abaixo executa a animação na direção reversa.

Arquivo: ./css/exemplos/animation/ex_animation07.html

```
<!DOCTYPE html>
<html>
   <head>
       <meta charset="UTF-8">
       <style>
           div {
               width: 100px;
               height: 100px;
               background-color: black;
               position: relative;
               animation-name: cores;
               animation-duration: 4s;
               animation-direction: reverse;
           }
           @keyframes cores {
               0%   {background-color: black; left:0px; top:0px;}
               25%  {background-color: gray; left:200px; top:0px;}
               50%  {background-color: brown; left:200px; top:200px;}
```

```
            75%  {background-color: burlywood; left:0px; top:200px;}
            100% {background-color: beige; left:0px; top:0px;}
        }
    </style>
</head>
<body>
    <h1>Animação CSS</h1>
        <div></div>
</body>
</html>
```

Animação CSS

O exemplo a seguir utiliza o valor “alternate”, fazendo com que a animação primeiro execute para frente, e posteriormente, para trás.

Arquivo: ./css/exemplos/animation/ex_animation08.html

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <style>
            div {
                width: 100px;
                height: 100px;
                background-color: black;
                position: relative;
                animation-name: cores;
                animation-duration: 4s;
                animation-iteration-count: 2;
```

```
            animation-direction: alternate;
            /*animation-direction: alternate-reverse;*/

        }
        @keyframes cores {
            0%   {background-color: black; left:0px; top:0px;}
            25%  {background-color: gray; left:200px; top:0px;}
            50%  {background-color: brown; left:200px; top:200px;}
            75%  {background-color: burlywood; left:0px; top:200px;}
            100% {background-color: beige; left:0px; top:0px;}
        }
    </style>
</head>
<body>
    <h1>Animação CSS</h1>
        <div></div>
</body>
</html>
```

Animação CSS

OBS.: utilizando o valor `animation-direction: alternate-reverse;` a animação será executada de modo contrária ao apresentado no exemplo acima, ou seja, primeiro para trás, e depois para frente.

Curva de Velocidade da Animação

A propriedade "*animation-timing-function*" permite especificar a curva de velocidade da animação. Ela pode ser configurada com os seguintes valores:

- ***ease*** - especifica uma animação com o seguinte perfil: lento, rápido e lento (padrão);
- ***linear*** - especifica uma animação com a mesma velocidade do início ao fim;
- ***ease-in*** - Especifica uma animação com um início lento
- ***ease-out*** - Especifica uma animação com um final lento;
- ***ease-in-out*** - Especifica uma animação com um início e fim lentos
- ***cubic-bezier(n,n,n,n)*** - Permite que você defina seus próprios valores em uma função cubic-bezier;

Veja o exemplo abaixo para cada uma das configurações:

Arquivo: ./css/exemplos/animation/ex_animation09.html

```
<!DOCTYPE html>
<html>
   <head>
       <meta charset="UTF-8">
       <style>
           div {
               width: 80px;
               height: 40px;
               background-color: gainsboro;
               font-weight: bold;
               position: relative;
               animation: movimento 4s;
               animation-fill-mode: forwards; /*será explicada a seguir*/
               margin-bottom: 3px;
           }
           #div1 {animation-timing-function: linear;}
           #div2 {animation-timing-function: ease;}
           #div3 {animation-timing-function: ease-in;}
           #div4 {animation-timing-function: ease-out;}
           #div5 {animation-timing-function: ease-in-out;}
           @keyframes movimento {
               from {left: 0px;}
               to {left: 300px;}
           }
       </style>
   </head>
```

```
    <body>
        <h1>Animação CSS</h1>
        <div id="div1">linear</div>
        <div id="div2">ease</div>
        <div id="div3">ease-in</div>
        <div id="div4">ease-out</div>
        <div id="div5">ease-in-out</div>
    </body>
</html>
```

Animação CSS

linear

ease

ease-in

ease-out

ease-in-out

Modo de Preenchimento da Animação

As animações CSS não afetam um elemento antes que o primeiro quadro-chave (*keyframe*) seja reproduzido, ou depois que o último quadro-chave (*keyframe*) seja reproduzido. Contudo, a propriedade "*animation-fill-mode*" pode modificar esse comportamento. Ela possibilita especificar um estilo, ao elemento, quando a animação não está sendo reproduzida, ou seja, antes de começar e depois de terminar.

A propriedade "*animation-fill-mode*" pode ter os seguintes valores:

- ***none*** - a animação não aplica nenhum estilo ao elemento antes ou depois de sua execução;

- ***forwards*** - o elemento manterá os valores de estilo definidos pelo último quadro-chave (dependendendo das propriedades *animation-direction* e *animation-iteration-count*);
- ***backwards*** - o elemento terá os valores de estilo definidos pelo primeiro quadro-chave (dependendo da propriedade *animation-direction*) e os manterá durante o período de atraso da animação;
- ***both*** - a animação seguirá as regras para frente e para trás, estendendo as propriedades da animação em ambas as direções;

O exemplo abaixo faz com que o elemento retenha os valores de estilo do último quadro-chave quando a animação termina:

Arquivo: ./css/exemplos/animation/ex_animation10.html

```
<!DOCTYPE html>
<html>
   <head>
       <meta charset="UTF-8">
       <style>
           div {
               width: 120px;
               height: 120px;
               background-color: gainsboro;
               position: relative;
               animation: movimento 4s;
               animation-delay: 1s;
               animation-fill-mode: forwards;
               /* animation-fill-mode: backwards; */
               /* animation-fill-mode: both; */
           }
           @keyframes movimento {
               from {top: 0px; background-color: orange;}
               to {top: 120px; background-color: salmon;}
           }
       </style>
   </head>
   <body>
       <h1>Animação CSS</h1>
```

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

Animação CSS

Propriedades de Animação - Abreviação

O CSS permite que todas as seis propriedades de animação sejam utilizadas e configuradas com uma única linha de código. Veja o exemplo abaixo:

Arquivo: ./css/exemplos/animation/ex_animation11.html

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <style>
            div {
                width: 120px;
                height: 120px;
                background-color: gainsboro;
                position: relative;
                animation: movimento 4s linear 1s infinite alternate;
                /*
                    animation-name: movimento;
                    animation-duration: 4s;
                    animation-timing-function: linear;
                    animation-delay: 1s;
                    animation-iteration-count: infinite;
                    animation-direction: alternate;
```

```
            */
        }
        @keyframes movimento {
            from {top: 0px; background-color: orange;}
            to {top: 200px; background-color: salmon;}
        }
    </style>
</head>
<body>
    <h1>Animação CSS</h1>
    <div></div>
</body>
</html>
```

Animação CSS

EXEMPLO MARIO BROS

Link: https://gileduardo.com.br/ifpr/dw/downloads/mario-bros

Diretório: ./css/exemplos/animation/mario-bros

Media Queries

A regra ***@media***, introduzida no CSS2, possibilita definir diferentes regras de estilo para diferentes tipos de mídia (computador, tablet, smartphone, etc.) ou tamanhos de tela. As consultas de mídia no CSS3 ampliaram a ideia dos tipos de mídia do CSS2: em vez de procurar um tipo de dispositivo, elas analisam a capacidade do mesmo.
As consultas de mídia (*Media Queries*) podem ser usadas para verificar um conjunto variado de informações, tais como:

- largura e altura da janela de visualização;
- orientação da janela de visualização (paisagem ou retrato)
- resolução

O uso das "*Media Queries*" é uma técnica popular para fornecer uma folha de estilo personalizada para desktops, laptops, tablets e celulares (como iPhones e celulares Android).

Tipos de Mídia CSS (*Media Types*)

VALOR	**DESCRIÇÃO**
all	Usada para todos os tipos de dispositivos de mídia
print	Usado para o modo de visualização de impressão
screen	Usado para telas de computadores, tablets e smartphones

Recursos Comuns de Mídias CSS (*Media Features*)

VALOR	**DESCRIÇÃO**
orientation	Orientação da ***viewport*** [1] - Retrato (Portrait) ou Paisagem (Landscape).
max-height	Altura máxima da *viewport*.
min-height	Altura mínima da *viewport*.

height	Altura da *viewport* (com barra de rolagem)
max-width	Largura máxima da *viewport*.
min-width	Largura mínima da *viewport*.
width	Largura da *viewport* (com barra de rolagem)

[1] área que está sendo exibida no momento (janela de visualização).

Sintaxe - Media Queries

Uma consulta de mídia baseia-se num tipo de mídia (*media type*) e pode conter um ou mais recursos de mídia (*media features*), que são resolvidos como verdadeiro ou falso.

```
@media not|only mediatype and (media feature) and (media feature) {
  CSS-Code;
}
```

O tipo de mídia é opcional (se omitido, será definido como "*all*"). Contudo, se utilizarmos "*not*" ou "*only*", será necessário especificar um tipo de mídia.

O resultado da consulta é "***true***" se o tipo de mídia especificado corresponder ao tipo de dispositivo no qual o documento está sendo exibido e todos os recursos de mídia na consulta de mídia forem "***true***". Quando uma consulta de mídia for "***true"***, a folha de estilo ou regras de estilo correspondentes serão aplicadas, seguindo as regras normais de cascata.
Significado das palavras "***not***", "***only***" e "***and***":

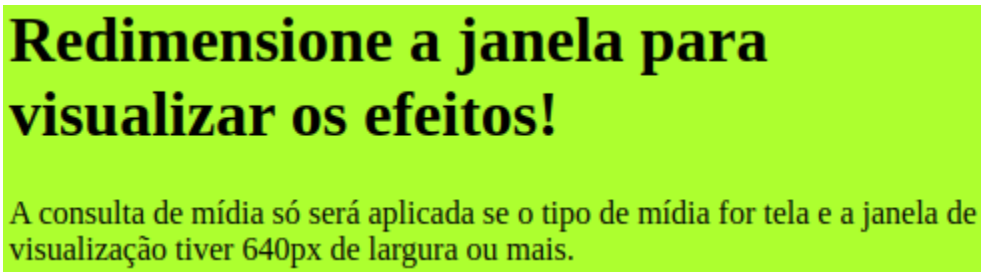
- "***not***" - palavra-chave que inverte o significado de uma consulta de mídia inteira;
- "***only***" - palavra-chave que impede que os navegadores mais antigos (não suportam media queries) apliquem os estilos especificados. Ela não tem efeito em navegadores modernos.
- "***and***" - palavra-chave que combina um tipo de mídia com um ou mais recursos de mídia.

Exemplos - Media Queries

Uma maneira de uso das "*Media Queries*" é ter uma seção CSS alternativa dentro da sua folha de estilo. O exemplo a seguir altera a cor de fundo para laranja se a janela de visualização tiver 640 pixels de largura ou mais (se a janela de visualização tiver menos de 640 pixels, a cor de fundo será verde).

Arquivo: ./css/exemplos/media-queries/ex_mediaqueries01.html

```
<!DOCTYPE html>
<html>
   <head>
       <style>
           body {
               background-color:greenyellow;
           }
           /* screen: se aplica às telas - computador, tablet e smartphone */
           /* min-width: largura mínima da tela do dispositivo */
           @media screen and (min-width: 640px) {
               body {
                   background-color: orangered;
               }
           }
       </style>
   </head>
   <body>
       <h1>Redimensione a janela para visualizar os efeitos!</h1>
        <p>A consulta de mídia só será aplicada se o tipo de mídia for tela e a janela de visualização tiver 640px de largura ou mais.</p>
   </body>
</html>
```

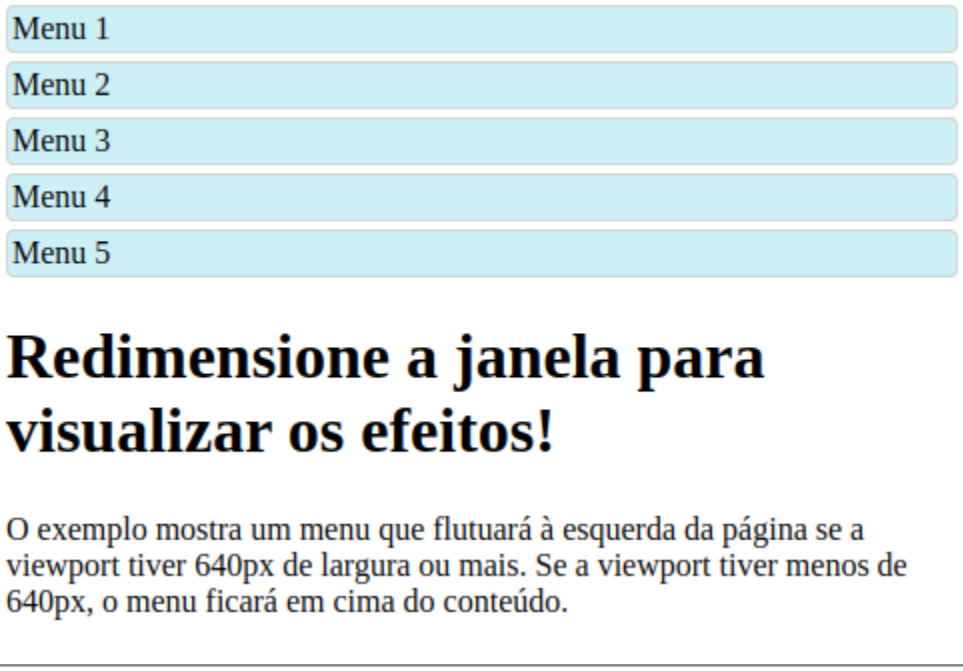


O exemplo a seguir mostra um menu que flutuará à esquerda da página se a janela de visualização tiver 640 pixels de largura ou mais (se a janela de visualização tiver menos de 640 pixels, o menu ficará sobre o conteúdo):

Arquivo: ./css/exemplos/media-queries/ex_mediaqueries02.html

```
<!DOCTYPE html>
<html>
   <head>
       <meta name="viewport" content="width=device-width, initial-scale=1.0">
       <style>
           .wrapper {
               overflow: auto;
           }
           #main {
               margin-left: 4px;
           }
           #leftsidebar {
               float: none;
               width: auto;
           }
           #menulist {
               margin: 0;
               padding: 0;
           }
           .menuitem {
               background: #CDF0F6;
               border: 1px solid #d4d4d4;
               border-radius: 4px;
               list-style-type: none;
               margin: 4px;
               padding: 2px;
           }
           @media screen and (min-width: 640px) {
               #leftsidebar { width: 200px; float: left; }
               #main { margin-left: 216px; }
           }
       </style>
```

```
    </head>
    <body>
        <div class="wrapper">
            <div id="leftsidebar">
                <ul id="menulist">
                <li class="menuitem">Menu 1</li>
                <li class="menuitem">Menu 2</li>
                <li class="menuitem">Menu 3</li>
                <li class="menuitem">Menu 4</li>
                <li class="menuitem">Menu 5</li>
                </ul>
            </div>
            <div id="main">
                <h1>Redimensione a janela para visualizar os efeitos!</h1>
                 <p>O exemplo mostra um menu que flutuará à esquerda da página se a viewport tiver 640px de largura ou mais. Se a viewport tiver menos de 640px, o menu ficará em cima do conteúdo.</p>
            </div>
        </div>
    </body>
</html>
```



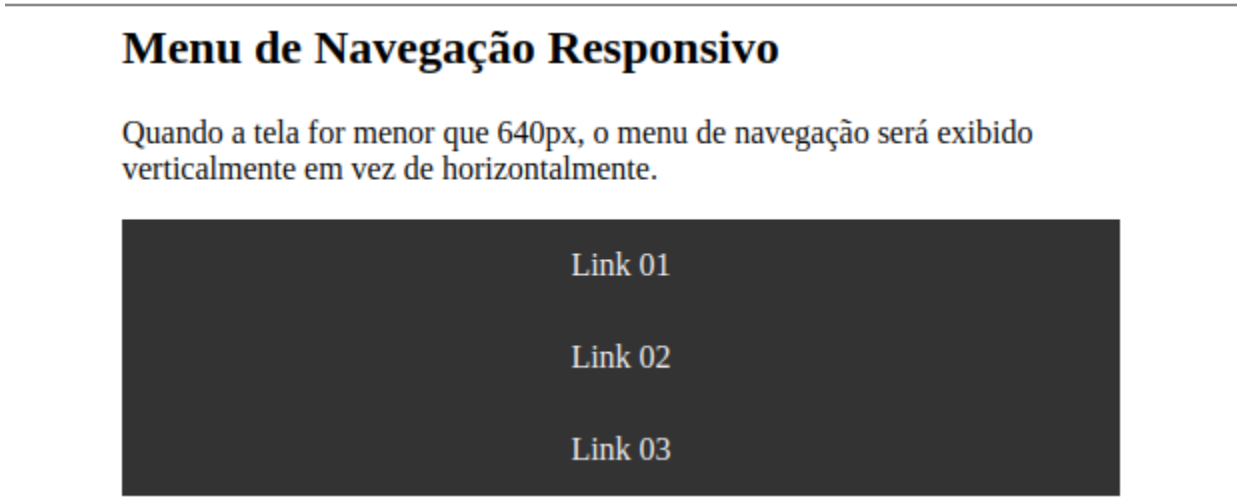
Exemplos - Media Queries para Menus

No exemplo a seguir usamos consultas de mídia para criar um menu de navegação responsivo, que varia em design em diferentes tamanhos de tela.

Arquivo: ./css/exemplos/media-queries/ex_mediaqueries03.html

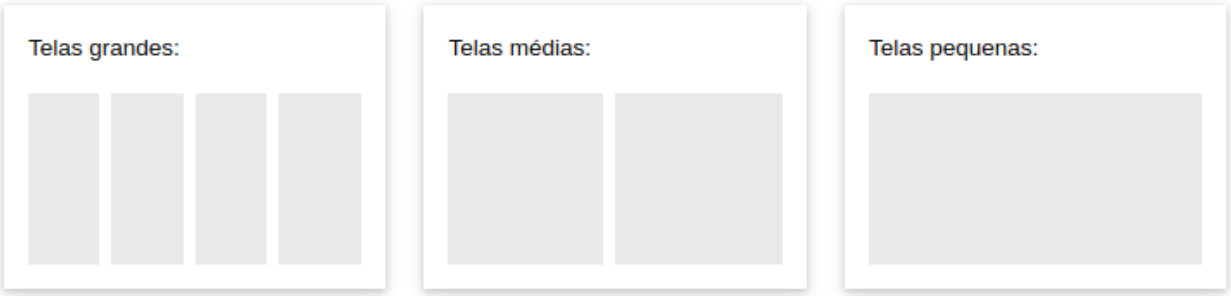
```
<!DOCTYPE html>
<html lang="pt-br">
   <head>
       <meta charset="utf-8">
       <meta name="viewport" content="width=device-width, initial-scale=1">
       <style>
           * {
               box-sizing: border-box;
           }
           /* Etilo - Barra de Navegação do Topo */
           .topnav {
               overflow: hidden;
               background-color: #333;
           }
           /* Estilo - Links da Barra */
               .topnav a {
               float: left;
               display: block;
               color: #f2f2f2;
               text-align: center;
               padding: 14px 16px;
               text-decoration: none;
           }
           /* Modofica a Cor - Passar o Mouse */
           .topnav a:hover {
               background-color: #ddd;
               color: black;
           }
               /* Telas com 640px de largura ou menos, faz os links do menu
empilharem, ao inveś de ficar lado a lado */
           @media screen and (max-width: 640px) {
               .topnav a {
                   float: none;
```

```
                width: 100%;
            }
        }
    </style>
</head>
<body>
    <h2>Menu de Navegação Responsivo</h2>
     <p>Quando a tela for menor que 640px, o menu de navegação será exibido verticalmente em vez de horizontalmente.</p>
    <div class="topnav">
        <a href="#">Link 01</a>
        <a href="#">Link 02</a>
        <a href="#">Link 03</a>
    </div>
</body>
</html>
```



Exemplos - Media Queries para Colunas

Um uso comum de *media queries* é criar um layout flexível. No exemplo a seguir é criado um layout que varia entre quatro, duas e colunas de largura total, dependendo dos diferentes tamanhos de tela.



Arquivo: ./css/exemplos/media-queries/ex_mediaqueries04.html

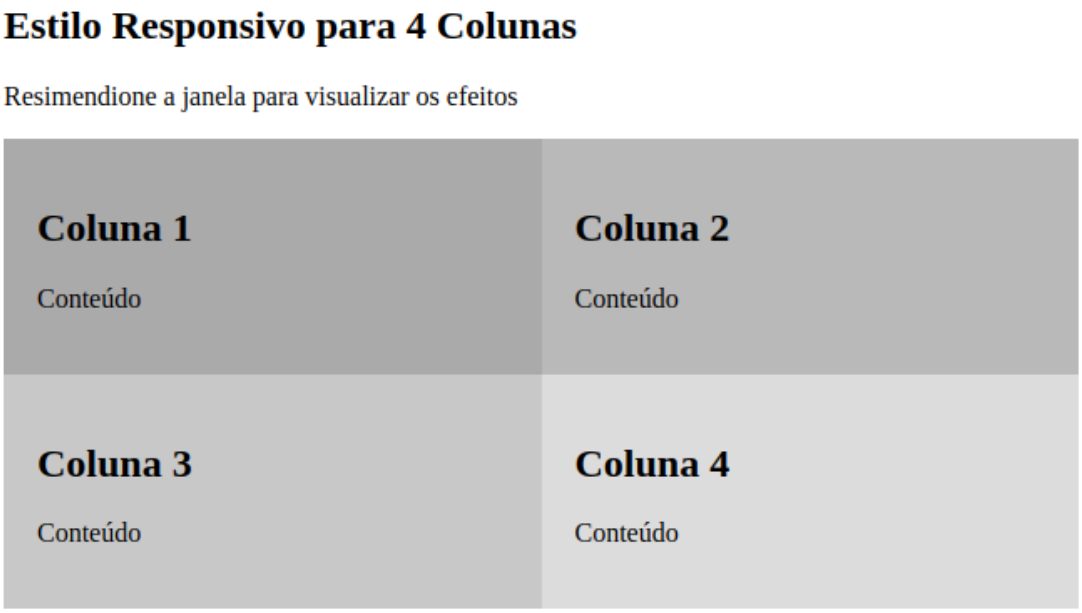
```
<!DOCTYPE html>
<html>
   <head>
       <meta charset="utf-8">
       <meta name="viewport" content="width=device-width, initial-scale=1">
       <style>
           * {
               box-sizing: border-box;
           }
           /* 4 Colunas iguais que flutuam lado a lado */
           .column {
               float: left;
               width: 25%;
               padding: 20px;
           }
           /* Limpa após as colunas */
           .row:after {
               content: "";
               display: table;
               clear: both;
           }
            /* Telas com 992px de largura ou menos, altera de quatro para duas
colunas */
           @media screen and (max-width: 992px) {
               .column {
                   width: 50%;
               }
           }
```

```
            /* Telas com 600px de largura ou menos, empilhe as colunas, ao invés
de colocar uma ao lado da outra */
            @media screen and (max-width: 600px) {
                .column {
                    width: 100%;
                }
            }
        </style>
    </head>
    <body>
        <h2>Estilo Responsivo para 4 Colunas</h2>
        <p>Resimendione a janela para visualizar os efeitos</p>
        <div class="row">
            <div class="column" style="background-color:#aaa;">
                <h2>Coluna 1</h2>
                <p>Conteúdo</p>
            </div>

            <div class="column" style="background-color:#bbb;">
                <h2>Coluna 2</h2>
                <p>Conteúdo</p>
            </div>

            <div class="column" style="background-color:#ccc;">
                <h2>Coluna 3</h2>
                <p>Conteúdo</p>
            </div>

            <div class="column" style="background-color:#ddd;">
                <h2>Coluna 4</h2>
                <p>Conteúdo</p>
            </div>
        </div>
    </body>
</html>
```



Dica: Uma maneira mais moderna de criar layouts de coluna é usar CSS Flexbox, que será abordado na sequência (veja o exemplo abaixo). No entanto, ele não é suportado no Internet Explorer 10 e versões anteriores. Se você precisar de suporte ao IE6-10, use floats (como mostrado acima).

Arquivo: ./css/exemplos/media-queries/ex_flexbox05.html

```
            /* Contêiner para flexboxes */
            .row {
                display: flex;
                flex-wrap: wrap;
            }
            /* 4 Colunas Iguais */
            .column {
                flex: 25%;
                padding: 20px;
            }
             /* Telas com 992px de largura ou menos, altera de quatro para duas
colunas */
            @media screen and (max-width: 992px) {
                .column {
                    flex: 50%;
```

```
            }
        }
        /* Telas com 600px de largura ou menos, empilhe as colunas, ao invés de colocar uma ao lado da outra  */
        @media screen and (max-width: 600px) {
            .row {
                flex-direction: column;
            }
        }
```

Exemplos - Media Queries - Ocultar Elementos

Mais um uso comum de "*Media Queries*" é ocultar elementos considerando diferentes tamanhos de tela.

Arquivo: ./css/exemplos/media-queries/ex_mediaqueries06.html

```
<!DOCTYPE html>
<html>
   <head>
       <meta name="viewport" content="width=device-width, initial-scale=1">
       <style>
           table.example {
               width: 100%;
               text-align: center;
           }
           td, th {
               border: 2px solid black;
           }
           .hide { }

           @media screen and (max-width: 800px) {
               .hide {
                   display: none;
               }
           }
       </style>
   </head>
   <body>
```

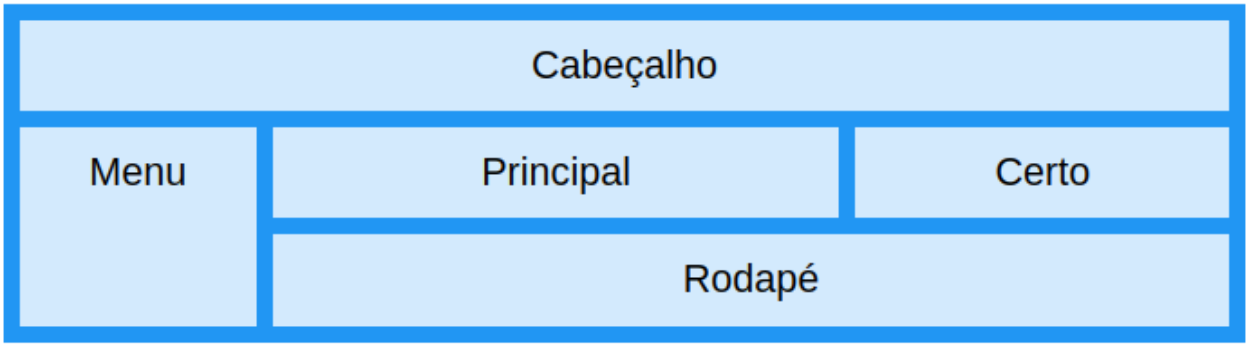
```
    <h2>Oculta Colunas de Acordo com o Tamanho da Tela</h2>
    <table class="example">
        <thead>
            <th>ID</th>
            <th>NOME</th>
            <th class="hide">CPF</th>
            <th class="hide">ENDEREÇO</th>
        </thead>
        <tbody>
            <tr>
                <td>1</td>
                <td>GIL EDUARDO DE ANDRADE</td>
                <td class="hide">000.000.000.01</td>
                <td class="hide">Av. Oswald de Andrade, 411</td>
            </tr>
        </tbody>
    </table>
  </body>
</html>
```

Oculta Colunas de Acordo com o Tamanho de Tela

ID	NOME
1	GIL EDUARDO DE ANDRADE

Grid Layout

O módulo *Grid Layout* (Grade), do CSS, oferece um sistema de layout baseado em grade, com linhas e colunas, facilitando o design de páginas da web sem precisar usar flutuação e posicionamento.



Elementos do Grid

Um *Grid Layout* (Estilo de Grade) consiste num elemento pai, com um ou mais elementos filhos. Um elemento HTML se torna um contêiner de grade quando sua propriedade "***display***" é definida como "***grid***" ou "***inline-grid***". Todos os filhos diretos do contêiner de grade tornam-se, automaticamente, itens de grade.

Arquivo: ./css/exemplos/grid/ex_grid01.html

```
<!DOCTYPE html>
<html>
   <head>
       <style>
           .grid-container {
            /* Transforma o elemento em um contêiner de grade, onde seus
            elementos filhos ("itens de grade") serão organizados em uma grade
            bidimensional
               display: grid;
            /* Especifica o número e a largura das colunas da grade, onde a
            largura da coluna será determinada automaticamente pelo conteúdo
            dos itens, com um total de três colunas */
               grid-template-columns: auto auto auto;
               background-color: orange;
               padding: 10px;
           }
           .grid-item {
               background-color: white;
               border: 1px solid orangered;
               padding: 20px;
               font-size: 30px;
```

```
            text-align: center;
        }
    </style>
</head>
<body>
    <h1>Elementos em Grade</h1>
        <p>Um layout em grade deve ter um elemento pai com a propriedade <em>display</em> definida como <em>grid</em> ou <em>inline-grid</em>.</p>
        <p>Os elementos que são filhos diretos do container grid, tornam-se, automaticamente, itens do grid.</p>
    <div class="grid-container">
        <div class="grid-item">1</div>
        <div class="grid-item">2</div>
        <div class="grid-item">3</div>
        <div class="grid-item">4</div>
        <div class="grid-item">5</div>
        <div class="grid-item">6</div>
        <div class="grid-item">7</div>
        <div class="grid-item">8</div>
        <div class="grid-item">9</div>
    </div>
</body>
</html>
```

Elementos em Grade

Um layout em grade deve ter um elemento pai com a propriedade *display* definida como *grid* ou *inline-grid*.

Os elementos que são filhos diretos do container grid, tornam-se, automaticamente, itens do grid.

1	2	3
4	5	6
7	8	9

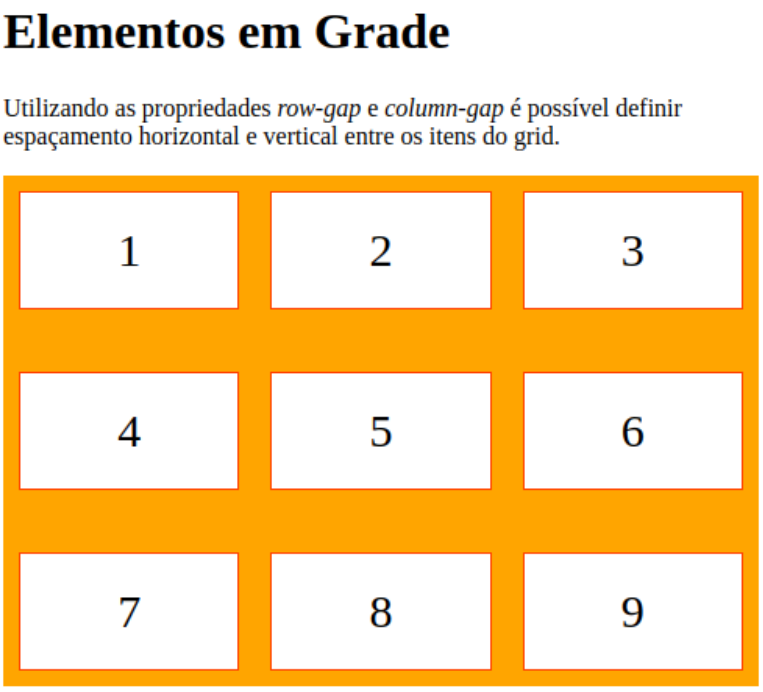
As linhas verticais dos itens da grade são chamadas de colunas. As linhas horizontais dos itens da grade são chamadas de linhas. Os espaços entre cada coluna/linha são chamados de lacunas (*gaps*).

Arquivo: ./css/exemplos/grid/ex_grid02.html

```
<!DOCTYPE html>
<html>
   <head>
       <style>
           .grid-container {
               display: grid;
               grid-template-columns: auto auto auto;
               background-color: orange;
               padding: 10px;
               row-gap: 40px;
               column-gap: 20px;
                /* Abreviação da Escrita */
                /* gap: 40px 20px */
                /* Único Espaçamento para Linha e Coluna */
                /* gap: 40px */
           }
           .grid-item {
               background-color: white;
               border: 1px solid orangered;
               padding: 20px;
               font-size: 30px;
               text-align: center;
           }
       </style>
   </head>
   <body>

       <h1>Elementos em Grade</h1>
         <p>Utilizando as propriedades <em>row-gap</em> e <em>column-gap</em> é possível definir espaçamento horizontal e vertical entre os itens do grid.</p>
       <div class="grid-container">
           <div class="grid-item">1</div>
```

```
            <div class="grid-item">2</div>
            <div class="grid-item">3</div>
            <div class="grid-item">4</div>
            <div class="grid-item">5</div>
            <div class="grid-item">6</div>
            <div class="grid-item">7</div>
            <div class="grid-item">8</div>
            <div class="grid-item">9</div>
        </div>
    </body>
</html>
```

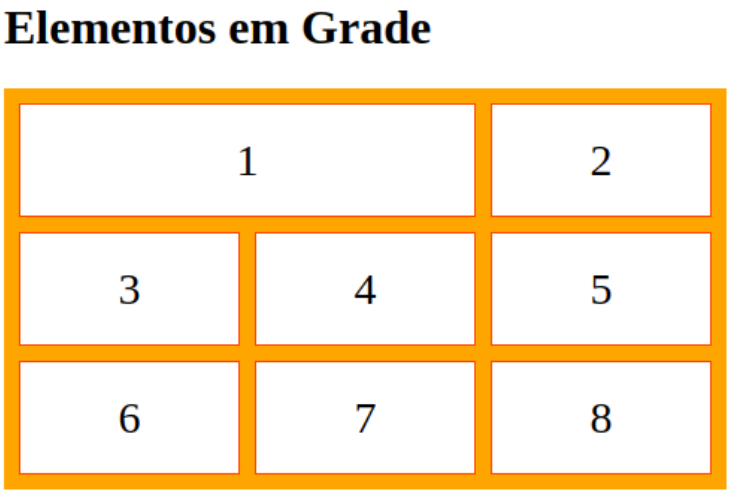


O Grid Layout permite, também, que um item de grade, para uma determinada linha, comece na coluna 1 e se prolongue até a coluna 3, por exemplo. O mesmo pode ser feito para determinada coluna, começando na linha 1 e se prolongando até a linha 2, por exemplo.

Arquivo: ./css/exemplos/grid/ex_grid03.html

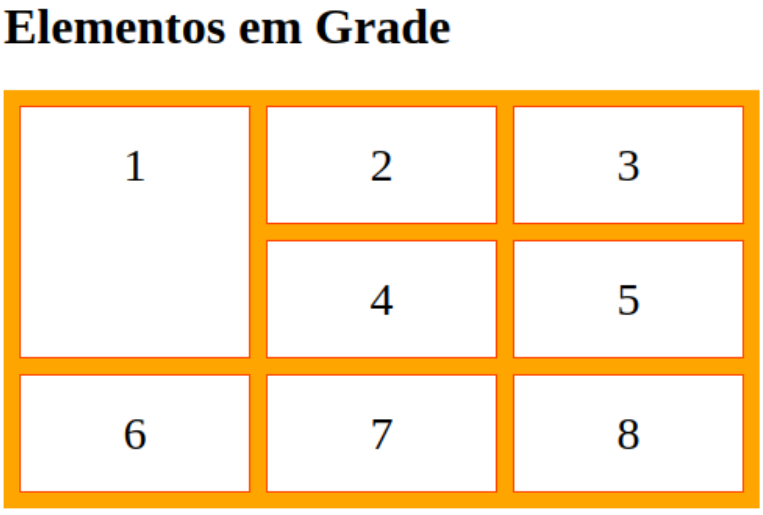
```
<!DOCTYPE html>
```

```
<html>
   <head>
       <style>
           .grid-container {
               display: grid;
               grid-template-columns: auto auto auto;
               background-color: orange;
               padding: 10px;
               gap: 10px;
           }
            /* Seleciona todos os elementos div que são filhos diretos de um
            elemento com a classe "grid-container" */
           .grid-container > div {
               background-color: white;
               border: 1px solid orangered;
               padding: 20px;
               font-size: 30px;
               text-align: center;
           }
           .item1 {
               grid-column-start: 1;
               grid-column-end: 3;
           }
       </style>
   </head>
   <body>
       <h1>Elementos em Grade</h1>
       <div class="grid-container">
           <div class="item1">1</div>
           <div>2</div>
           <div>3</div>
           <div>4</div>
           <div>5</div>
           <div>6</div>
           <div>7</div>
           <div>8</div>
       </div>
   </body>
</html>
```



Arquivo: ./css/exemplos/grid/ex_grid04.html

```
.item1 {
    grid-row-start: 1;
    grid-row-end: 3;
}
```



Todas as Propriedades do Grid Layout

PROPRIEDADE	DESCRIÇÃO
column-gap	Especifica o espaço entre as colunas da grid.
gap	Propriedade de abreviação para as propriedades *row-gap* e *column-gap.*
grid	Propriedade de abreviação para as propriedades de *grid-template-rows*, *grid-template-columns*, *grid-template-areas*, *grid-auto-rows*, *grid-auto-columns* e *grid-auto-flow properties.*
grid-area	Pode ser usada para nomear o item da grid ou como propriedade de abreviação para as propriedades *grid-row-start*, *grid-column-start*, *grid-row-end* e *grid-column-end.*
grid-auto-columns	Especifica um tamanho de coluna padrão.
grid-auto-flow	Especifica como os itens colocados automaticamente devem ser inseridos na grid.
grid-auto-rows	Especifica um tamanho de linha padrão.
grid-column	Propriedade de abreviação para as propriedades *grid-column-start* e *grid-column-end.*
grid-column-end	Especifica onde termina o item da grid.
grid-column-gap	Especifica o tamanho do espaço entre as colunas.
grid-column-start	Especifica onde começa o item da grid
grid-gap	Propriedade de abreviação para as propriedades *grid-row-gap* e *grid-column-gap.*
grid-row	Propriedade de abreviação para as propriedades *grid-row-start* e *grid-row-end.*
grid-row-end	Especifica onde termina o item da grid.
grid-row-gap	Especifica o tamanho do espaço entre as linhas.
grid-row-start	Especifica onde começa o item da grid.
grid-template	Propriedade de abreviação para as propriedades

	grid-template-rows, *grid-template-columns* e *grid-areas.*
grid-template-areas	Especifica como as colunas e linhas devem ser exibidas, usando itens da grid nomeados.
grid-template-columns	Especifica o tamanho e a quantidade de colunas para um *grid layout*.
grid-template-rows	Especifica o tamanho das linhas para um *grid layout*.
row-gap	Especifica o espaço entre as linhas da grid.

Mais alguns exemplos para as propriedades listadas na tabela anterior:

Arquivo: ./css/exemplos/grid/ex_grid05.html

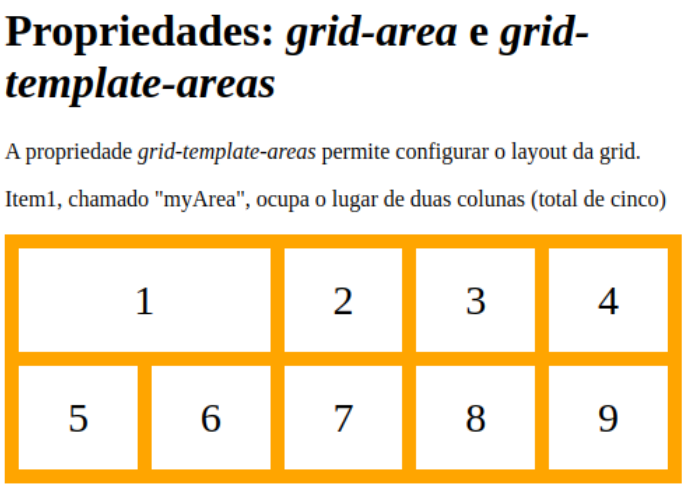
---

grid-area / *grid-template-areas*

```
<!DOCTYPE html>
<html>
   <head>
       <style>
           .item1 {
               grid-area: myArea;
           }
           .grid-container {
               display: grid;
               grid-template-areas: 'myArea myArea . . .';
               grid-gap: 10px;
               background-color: orange;
               padding: 10px;
           }
           .grid-container > div {
               background-color: white;
               text-align: center;
               padding: 20px 0;
               font-size: 30px;
           }
       </style>
   </head>
```

```
    <body>
        <h1>Propriedades: <i>grid-area</i> e <i>grid-template-areas</i></h1>
            <p>A propriedade <em>grid-template-areas</em> permite configurar o
layout da grid.</p>
          <p>Item1, chamado "myArea", ocupa o lugar de duas colunas (total de
cinco)</p>

        <div class="grid-container">
            <div class="item1">1</div>
            <div class="item2">2</div>
            <div class="item3">3</div>
            <div class="item4">4</div>
            <div class="item5">5</div>
            <div class="item6">6</div>
            <div class="item7">7</div>
            <div class="item8">8</div>
            <div class="item9">9</div>
        </div>
    </body>
</html>
```



OUTRO EXEMPLO - grid-area / grid-template-areas

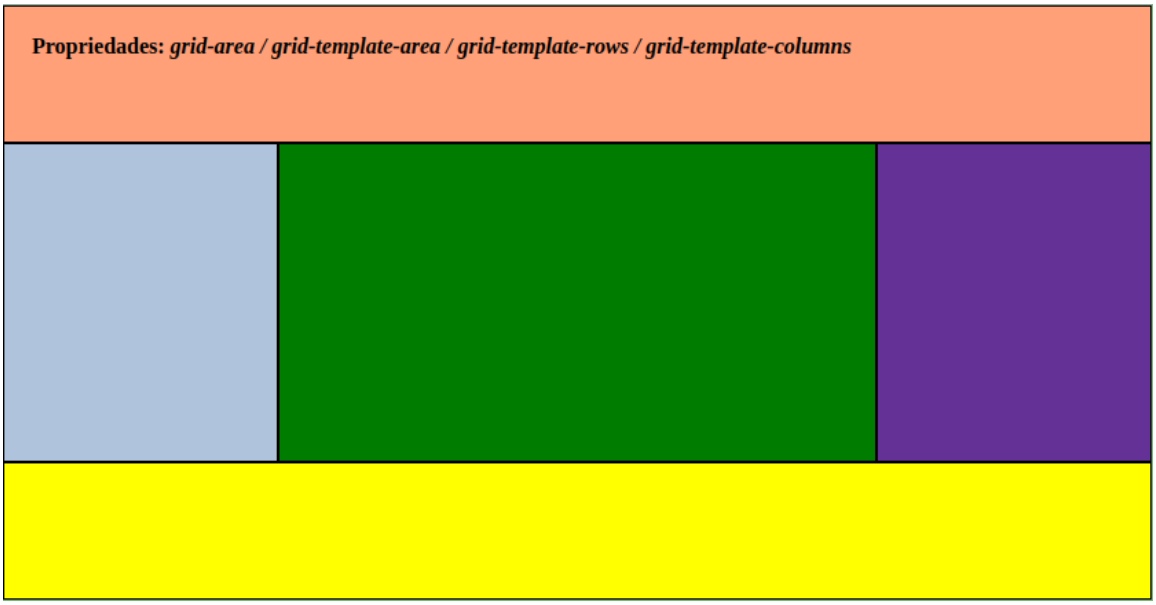
Arquivo: ./css/exemplos/grid/ex_grid08.html

```
<!DOCTYPE html>
```

```
<html>
   <head>
       <style>
           * {
               padding: 0;
               margin: 0;
           }
           .border {
               border: 1px solid black;
               padding: 20px;
           }
           body {
               display: grid;
               /* tamanho mínimo da altura - 100% da tela*/
               min-height: 100vh;
                 /* 2 linhas de 100 pixels e outra ocupando todo o espaço
restante*/
               grid-template-rows: 100px 1fr 100px;
               /* 3 colunas com tamanhos iguais*/
               grid-template-columns: 200px 1fr 200px;
               /* Áreas que preenchem o grid */
               grid-template-areas:
                   'topo topo topo'            /* 1ª linha */
                   'menu conteudo info'        /* 2ª linha */
                   'rodape rodape rodape';     /* 3ª linha */
           }
           header {
               grid-area: topo;
               background-color: lightsalmon;
           }
           nav {
               grid-area: menu;
               background-color: lightsteelblue;
           }
           main {
               grid-area: conteudo;
               background-color: green;
           }
```

```
            aside {
                grid-area: info;
                background-color: rebeccapurple;
            }
            footer {
                grid-area: rodape;
                background-color: yellow;
            }

        </style>
    </head>
    <body>
        <header class="border">
            <h4>Propriedades:
                <i>grid-area / grid-template-area / grid-template-rows /
grid-template-columns</i>
            </h4>
        </header>
        <nav class="border"></nav>
        <main class="border"></main>
        <aside class="border"></aside>
        <footer class="border"></footer>
    </body>
</html>
```



Arquivo: ./css/exemplos/grid/ex_grid06.html

---

grid-row-start / grid-column-end

```
<!DOCTYPE html>
<html>
   <head>
       <style>
           .item1 {
               grid-row-start: 2;
           }
           .item4 {
               grid-column-end: span 3;
           }
           .grid-container {
               display: grid;
               grid-template-areas: '. . . . .';
               grid-gap: 10px;
               background-color: orange;
               padding: 10px;
           }
           .grid-container > div {
               background-color: white;
```

```
            text-align: center;
            padding: 20px 0;
            font-size: 30px;
        }
    </style>
</head>
<body>
    <h1>Propriedades: <i>grid-row-start</i> e <i>grid-column-end</i></h1>
      <p>A propriedade <em>grid-row-start</em> indica em qual linha o item1 deve começar.</p>
     <p>A propriedade <em>grid-column-end</em> indica em qual coluna o item4 deve terminar.</p>
    <div class="grid-container">
        <div class="item1">1</div>
        <div class="item2">2</div>
        <div class="item3">3</div>
        <div class="item4">4</div>
        <div class="item5">5</div>
        <div class="item6">6</div>
    </div>
</body>
</html>
```

Propriedades: *grid-row-start* e *grid-column-end*

A propriedade *grid-row-start* indica em qual linha o item1 deve começar.

A propriedade *grid-column-end* indica em qual coluna o item4 deve terminar.