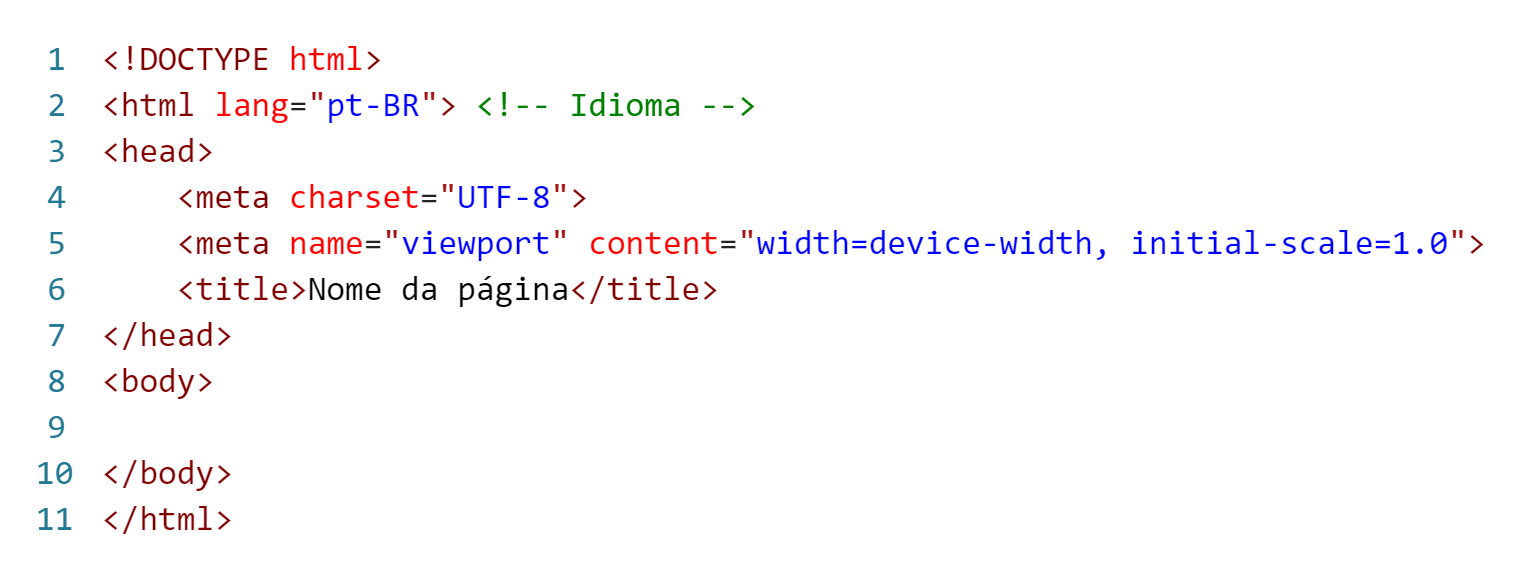
# HTML

## Código Base HTML

Todo documento HTML deve possuir as estruturas **‘<head>’** e **‘<body>’**, e claro, **‘<html>’**.



No VSCode é possível escrever todo o código acima digitando uma exclamação, aparecendo assim uma sugestão e deve-se clicar no ENTER ou na exclamação que apareceu.

## Elementos HTML

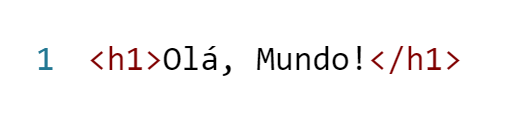
Os elementos HTML são fundamentais na construção de uma página.

### HTML Headings

Existem seis tipos de títulos, ou melhor, seis níveis.

1. **‘<h1>’:** Título de nível 1
2. **‘<h2>’:** Título de nível 2
3. **‘<h3>’:** Título de nível 3
4. **‘<h4>’:** Título de nível 4
5. **‘<h5>’:** Título de nível 5
6. **‘<h6>’:** Título de nível 6

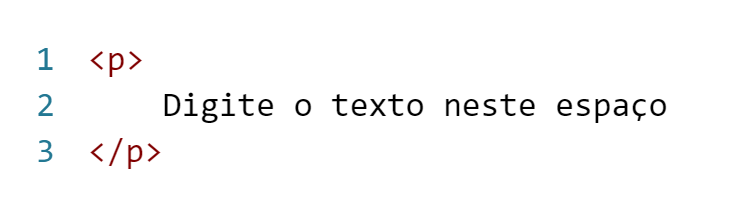
Essas tags devem ser fechadas, por exemplo:



Perceba que ao fim da linha há uma outra tag **<h1>**, porém com uma **‘/’** antes. Saiba que nem todas as tags em HTML possuem fechamento.

### Parágrafos

Para criar um parágrafo em HTML, use a tag **<p>**.

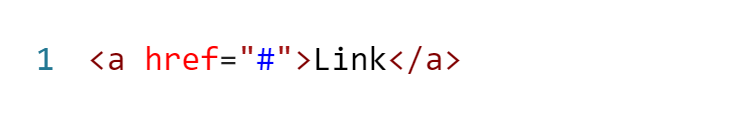


### Quebra de linha

A tag **<br>**, que significa “break row”, e é usada para quebrar de linha.

### *Hyperlinks*

Para criar links em HTML, é usada a tag **<a>**.



Chamamos de atributo o **‘href’**, seu valor deve ser uma url local ou web, ou seja, um link.

Caso não queira redirecionamento nenhum, basta colocar uma ‘#’ no lugar do link.

Entre as tags de abertura e fechamento, deve ficar o texto que, se caso o usuário clicar, será redirecionado ao local do link.

#### Abas

Por padrão, ao clicar em um link, o usuário é redirecionado a uma página que sobrepõe a atual. Para mudar isso, use o atributo **‘target’**. Seus valores são os seguintes:

* **‘\_self’:** Irá abrir o link na mesma aba. É o valor padrão.
* **‘\_blank’:** Irá abrir o link em uma nova aba.
* **‘\_parent’:** Irá abrir o link no quadro principal. Este valor é usado para quando a página está sendo exibida em um frame.
* **‘\_top’:** Irá abrir o link na janela inteira, substituindo qualquer estrutura de quadro existente.

#### Links para Download

Para download em um link, basta adicionar mais dois atributos ao elemento ‘<a>’: ‘download’ (que vai receber o nome do arquivo a ser baixado) e ‘type’ (que vai receber o tipo de arquivo que vai ser baixado).

##### Media Types

São os tipos de arquivos para download. Abaixo está uma lista com os mais comuns.

* **aplication/pdf**
* **aplication/zip**
* **text/html**
* **text/css**
* **text/javascript**
* **video/mp4**
* **video/JPEG**
* **audio/aac**
* **audio/mpeg**
* **font/ttf**
* **image/jpeg**
* **image/png**

**Mais Media Types**: <https://www.iana.org/assignments/media-types/media-types.xhtml>

## Comentários

Tudo que estiver entre **‘<!--Texto aqui!-->’** é considerado comentário em HTML.

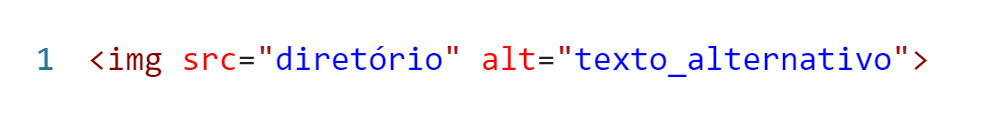


## Imagens

A tag ‘<img>’ tem como função a importação de uma imagem.

Há dois atributos principais para essa tag:

* **‘src’**: Especifica o diretório da imagem (local ou web).
* **‘alt’:** Determina um texto alternativo para a imagem. Esse texto aparece caso a imagem não possa ser carregada.



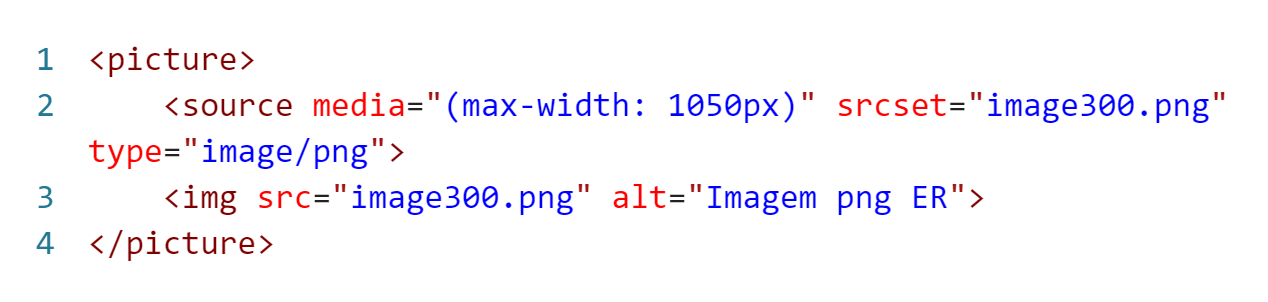
### Altura e largura da imagem

Há dois atributos para definir a altura e a largura da imagem.

* **‘width’:** Define a largura da imagem. O valor deve ser em pixels.
* **‘height’:** Define a altura da imagem. O valor também deve ser em pixels.

### Imagens Dinâmicas

A tag **‘<picture>’** é usada para concentrar as outras fontes da imagem.



#### <source>

Essa tag com seus atributos (vistos acima) servem para personalizar o tamanho visível da imagem de acordo com a largura da tela.

Isso faz com que uma imagem com vários tamanhos, ou seja, uma imagem que a partir dela foram geradas novas cópias com tamanhos diferentes, sejam exibidas com um tamanho adequado de acordo com o tamanho da tela.

##### Atributos

* **‘type’:** Deve especificar o Media Type.
* **‘srcset’:** Determina qual imagem que será carregada quando o tamanho máximo for atingido
* **‘media’:** Indica o tamanho máximo a ser considerado para carregar a imagem indicada no atributo ‘scrset’.

No código visto anteriormente é usado todos os três atributos.

## Áudios

A tag ‘<audio>’ é usada para carregar áudios em uma página.

Essa tag possui alguns atributos:

1. **‘type’:** Especifica o Media Type do áudio.
2. **‘preload’:** Indica se o áudio será carregado ou não. Esse atributo possui três valores:

* **‘metadata’:** Vai carregar apenas informações sobre o arquivo (tempo, tamanho, etc).
* **‘none’:** Não vai iniciar absolutamente nada a não ser que o usuário ou um script inicie a reprodução.
* **‘auto’:** Vai carregar o áudio inteiro assim que a página for carregada, mesmo que o usuário nunca inicie a reprodução.

1. **‘controls’:** Se não escrito, não aparecerá o player para controle na tela.
2. **‘autoplay’:** Se escrito, o áudio irá iniciar assim que a página for carregada.
3. **‘loop’:** Se escrito, a reprodução se reinicia ao chegar no final.

## Vídeos

Pode ser hospedado em seu site ou incorporado de outro, assim como as imagens e áudios. Os sites mais usados para incorporação são o YouTube e o Vimeo.

Os vídeos são incorporados por meio de iframes.

### Vídeos Hospedados no Site

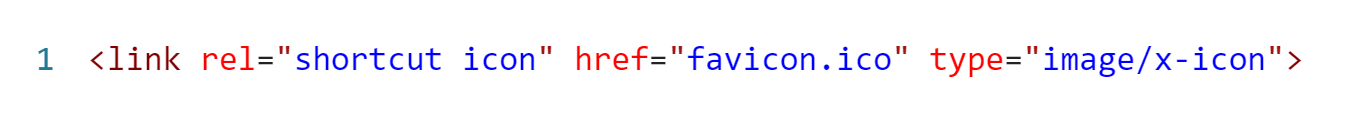
A tag ‘<video>’ é usada para mostrar um vídeo em uma página. Seus atributos são:

1. **‘width’:** Determina o tamanho do vídeo na tela. Considere o tamanho em pixels.
2. **‘poster’:** É a capa do vídeo que vai aparecer antes do usuário clicar. O valor deve ser o diretório ou url de uma imagem.
3. **‘controls’:** Se não escrito, não aparecerá o player para controle na tela.
4. **‘autoplay’:** Se escrito, o áudio irá iniciar assim que a página for carregada.
5. **‘loop’:** Se escrito, a reprodução se reinicia ao chegar no final.

### *Favicons*

Os *favicons* são os ícones ao lado do nome da página. É recomendado o uso de arquivos **.ico**, que são arquivos de ícones.

Para adicionar o ícone:



**ATENÇÃO:** Essa linha de código deve ser adicionada na área **<head>** do documento.

## Elementos de Formatação

Mudam o texto, tornando-o em negrito, itálico, riscando-o, etc.

### Negrito / Destaque

A palavra **"negrito"** está escrita em negrito! Foi utilizada a tag **<b>** (não semântico)

Já aqui a palavra **"negrito"** está utilizando a tag **<strong>** (semântico)

### Itálico / Ênfase

Escrito de forma *genérica*, com a tag **<i>** (não semântico)

Já nesse parágrafo, o *itálico* é por conta da tag **<em>** (semântico)

### Marcador

A palavra está marcada! Utilizado a tag **<mark>**.

### Sublinhar

Texto sublinhado. Foi utilizado a tag **<ins>**.

### Riscar texto

Este texto está ~~riscado~~, utilizado a tag **‘<del>’**.

### Caracteres Pequenos

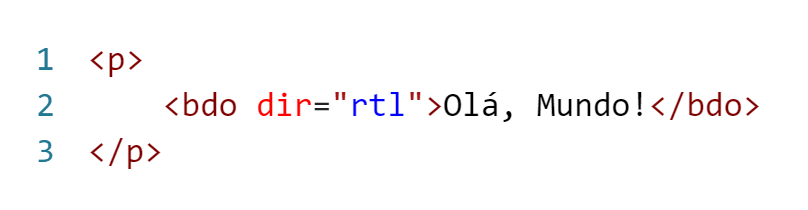
H2O, o caractere pequeno foi escrito com a tag **<sub>**.

105, escrito com a tag **<sup>**.

### Texto Invertido

Use a tag **<bdo>** (*bi-directional override*) para inverter um texto. Nele, há o atributo **‘dir’,** coloque a direção:

* **‘rtl’:** right to left.
* **‘lrt’:** left to right**.**



## Emojis

Para usar um emoji, basta saber seu *Codepoints*, que pode ser encontrado no site [www.emojipedia.org](http://www.emojipedia.org).

Transcrevendo o emoji referente ao *Codepoints* U+1F600 (carinha feliz) para ser escrito em HTML, basta trocar U+ por &#x (minúsculo mesmo) e ao final acrescentar um ponto e vírgula ‘**;**’.

## Citações

### Citação no meio do parágrafo

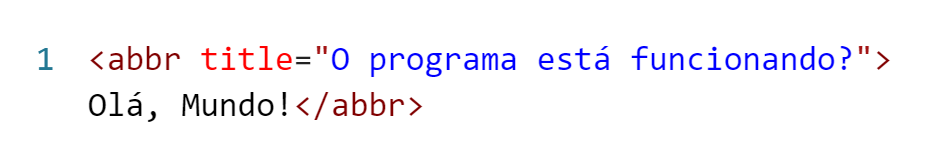
Para fazer uma citação, use a tag **<q>**, que significa “quote”. Pode ser usado para fazer uma citação no meio de um parágrafo.

### Citar um Bloco Inteiro

A tag **‘<blockquote>’** permite citar um bloco inteiro. Ela possui um atributo chamado **‘cite’**, e seu valor pode ser uma URL de um site.

## Abreviações

A tag **‘<abbr>’** permite mostrar um pequeno texto ao passar o mause sobre a palavra. Ela possui um atributo **‘title’**, e seu valor deve ser a mensagem para ser mostrada como abreviação.



## Endereço

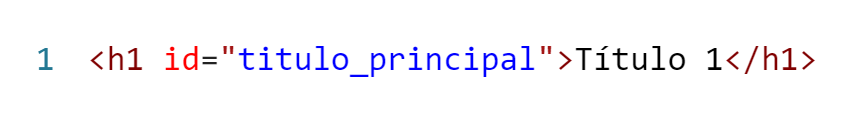
A tag **‘<address>’** define um campo para informações de contato para o autor do documento ou artigo. Todas as informações devem ser escritas dentro da tag de abertura e fechamento.

## ‘id’, ‘class’ e ‘name’

São identificadores, e no caso do **‘id’** e do **‘class’** são usados principalmente para a aplicação de estilos em CSS.

### ‘id’

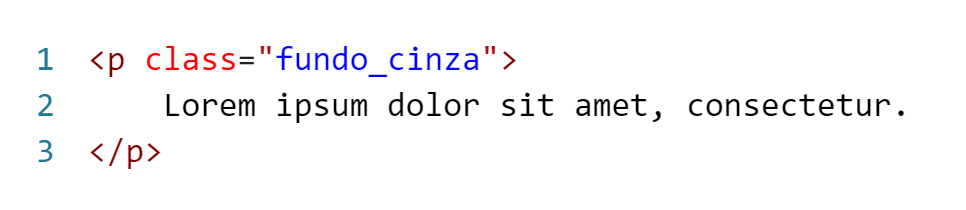
O **‘id’** é um atributo que pode ser adicionado a qualquer elemento HTML. Seu valor é um nome para se referir a esse elemento específico.



Em um único projeto, deve haver apenas um elemento com um **‘id’** específico, ou seja, nada de dois elementos como o mesmo **‘id’**.

### ‘class’

O atributo **‘class’** similar ao ‘id’, mas podendo ser adicionado a diversos elementos com um mesmo nome de referência, permitindo criar estilos em CSS para vários elementos ao mesmo tempo.



### ‘name’

O atributo **‘name’** especifica um nome para um elemento HTML.

Esse nome pode ser usado para referenciar o elemento em JavaScript, ou para referenciar dados enviados em formulários. Além disso, é possível trabalhar com esse atributo junto aos elementos ‘<map>’, ‘<iframe>’ e ‘<meta>’.

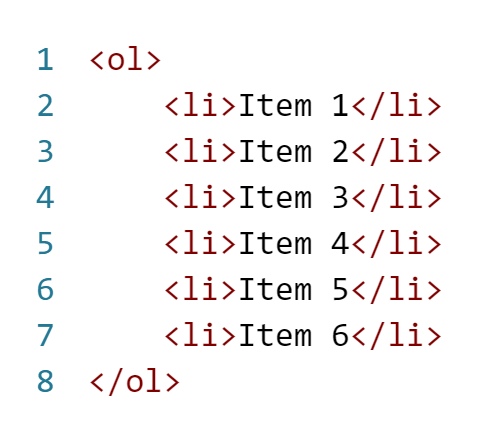
## Listas em HTML

Listas são bem úteis para organização e para criar menus em um site.

### Listas Ordenadas (*Ordered Lists*)

Nesse tipo de lista, a ordem dos itens importa, e por isso cada item é numerado.

A tag **‘<ol>’** cria uma lista ordenada, e dentro das tags de abertura e fechamento vão existir os itens, identificados pela tag **‘<li>’**.



O atributo **‘type’**, aplicado à tag **‘<ol>’** determina o tipo de marcador da lista.

Os tipos de marcadores são:

1. Valor padrão, cada item é numerado.
2. Lista alfabética maiúscula.
3. Lista alfabética minúscula.
4. Lista em números romanos em maiúsculo.
5. Lista em números romanos em minúsculo.

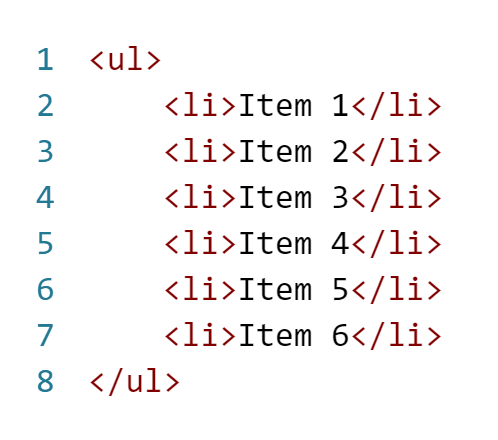
Usando o atributo **‘start’**, é possível definir o valor do primeiro item de uma lista, fazendo assim ela ser contada a partir desse valor. Esse valor deve ser numérico, mesmo que a forma de contagem da lista seja alfabética.

### Listas Não Ordenadas (*Unordered Lists*)

Também são chamadas de Listas com Marcadores, que são aquelas em que a ordem dos itens não tem importância.

Ao invés de ‘<ol>’, é usada a tag **‘<ul>’** para criar as listas não ordenadas.

Para criar itens para a lista, é exatamente igual as listas ordenadas, utilizando a tag **‘<li>’**.



No atributo ‘type’ da tag ‘<ul>’, é possível especificar o tipo de marcador. Os valores são:

* **‘disc’:** Um círculo preto preenchido.
* **‘circle’:** Um círculo preto sem preenchimento.
* **‘square’:** Um quadrado preto.

É importante ressaltar que é possível mesclar as duas listas, colocando uma dentro da outra, por exemplo.

### Listas de Definição (*Definition List*)

É como um dicionário, primeiro um termo e abaixo um texto. Toda definition list está dentro de uma tag **‘<dl>’**e cada termo é definido por uma tag **‘<dt>’** (definition term), para uma descrição é usada a tag **‘<dd>’** (definition description).

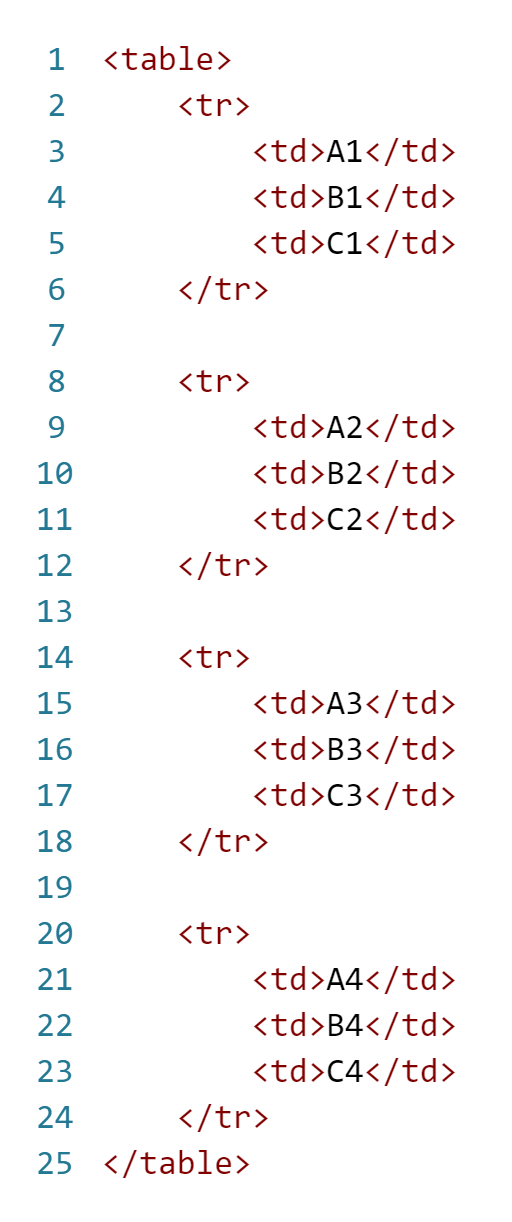


## Tabelas em HTML

A anatomia de uma tabela é a seguinte: **Table**, **Table Row**, **Table Header**, **Table Data**.

1. **‘table’:** Contém todos os elementos de uma tabela dentro de si.
2. **‘table row’:** É relativo a linha da tabela. Sua tag é **‘<tr>’**.
3. **‘table header’:** É relativo a um cabeçalho de uma tabela. Sua tag é **‘<th>’**.
4. **‘table data’:** É relativo a um dado de uma tabela. Sua tag é **‘<td>’**.

Exemplo de uma estrutura de tabela:

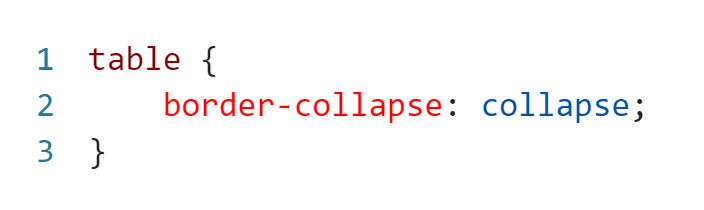


### Borda na Tabela

Com a propriedade **‘border’** em CSS, é possível criar uma borda para a tabela.

Para as linhas ficarem coladas uma às outras, adicione a propriedade **‘border-collapse’**.

Seus valores são **‘separate’** e **‘collapse’**, para se separar e juntar, respectivamente.



Lembre-se que essa propriedade deve ser adicionado à tabela diretamente, como no exemplo acima.

### Alinhamento de Células

#### Alinhamento Vertical

Para alinhar verticalmente um dado de uma tabela, use a propriedade **‘vertical-align’** em CSS.

Seus valores são:

* **‘top’:** Em cima.
* **‘middle’:** No meio.
* **‘bottom’:** Em baixo.

#### Alinhamento Horizontal

Para alinhar horizontalmente um dado, use a propriedade **‘text-align’**.

Seus valores são:

* **‘left’:** Esquerda.
* **‘center’:** Centro.
* **‘right’:** Direita.

### Tabelas Semânticas

Pode-se utilizar três tags de agrupamento de tabela diferentes:

1. **‘thead’:** É a “cabeça” da tabela. Geralmente onde ficam os cabeçalhos de uma tabela.
2. **‘tbody’:** É o “corpo” da tabela. É onde ficam as linhas e os dados de uma tabela.
3. **‘tfoot’:** É o “pé” da tabela. É onde fica o rodapé da tabela.

Em todas elas é possível usar as tags **‘<th>’**, **‘<tr>’** e **‘<td>’**.

#### Legenda

Com a tag **‘<caption>’** é possível criar uma legenda acima da tabela. No escopo desta tag, deve ser escrito apenas o texto da legenda.

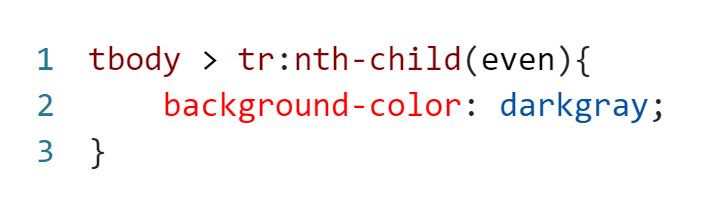
### Escopo do ‘<th>’

É importante especificar aos mecanismos de pesquisas se os valores relativos aos cabeçalhos estão abaixo ou ao lado. Para isso, existe um atributo chamado **‘scope’**, que possui os seguintes valores.

1. **‘col’:** Quando os valores relativos à linha estão abaixo.
2. **‘row’:** Quando os valores relativos à linha estão lado.
3. **‘colgroup’:** Quando os valores relativos à linha estão em um grupo abaixo.
4. **‘rowgroup’:** Quando os valores relativos à linha estão em um grupo ao lado.

### Efeito Zebrado

O efeito zebrado é útil para organizar melhor uma tabela. Para criá-lo, é necessário a adição de um seletor com uma pseudo-classe:



A pseudo-classe **‘:nth-child’** pode receber como valor **‘2n’**, sendo o “2” o número de linhas a ser puladas. Também pode receber os valores **‘odd’** (ímpar) e **‘even’** (par).

### Cabeçalho Fixado

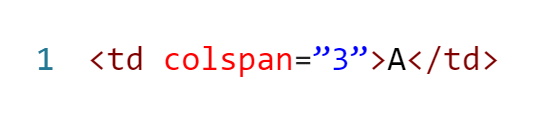
Para manter o cabeçalho de uma tabela fixado, mesmo ao rolar a tela, dê a tabela a posição relativo (**‘position: relative’**) e ao cabeçalho a posição “colável” (**‘position: sticky’**). Depois adicione a propriedade: **‘top: 0px;’**.

### Espaçamento das Células

Cada dado de uma tabela é uma célula, e podemos fazer uma célula ocupar mais de uma coluna e/ou linha utilizando os atributos **‘colspan’** e **‘rowspan’**.

#### ‘colspan’

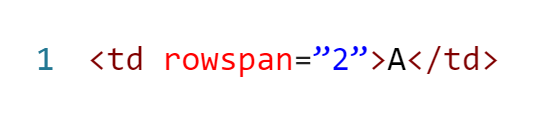
O valor dado ao atributo **‘colspan’** definirá quantas colunas uma célula vai ocupar. O valor deve ser numérico.



No código acima, a célula contendo a letra “A” será redimensionada para ocupar três linhas.

#### ‘rowspan’

O valor dado ao atributo **‘rowspan’** definirá quantas colunas uma célula vai ocupar. O valor também deve ser numérico.

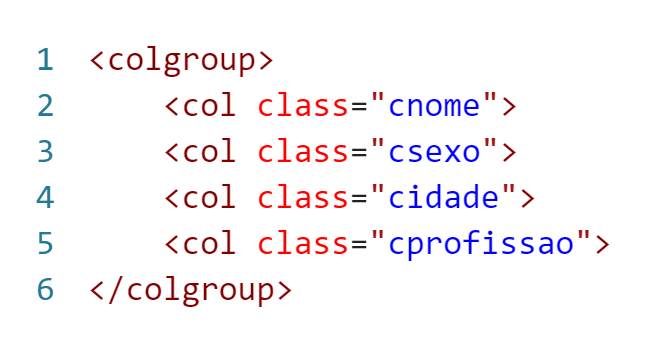


No exemplo acima, a célula que contém a letra “A” será redimensionada para ocupar o lugar de duas linhas.

### Isolamento por Coluna

Para fazer uma alteração pontual em uma coluna específica, existe a tag **‘<colgroup>’** em HTML. Essa tag ajuda a denominar cada coluna como única.

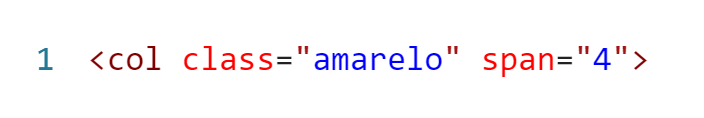
Dentro dessa tag, existem as tags **‘<col>’**, que são cada coluna.



Note que a tag **‘<col>’** não precisa de fechamento.

Com o atributo ‘span’, é possível definir mais de uma coluna, dependendo do valor especificado.

Por exemplo:



Quando for alterar o estilo da classe “amarelo” em CSS, quatro colunas serão modificadas.

### Tabelas Responsivas

Quando uma tabela é muito grande, é criada uma barra de rolagem lateral na página. É possível fazer com que essa rolagem seja dentro da tabela, não afetando o resto do conteúdo da página.

Para isso, a tabela precisa estar dentro de uma **‘<div>’**, por exemplo, que receberá a propriedade **‘overflow-x’**, com o valor **‘scroll’** ou **‘auto’**.

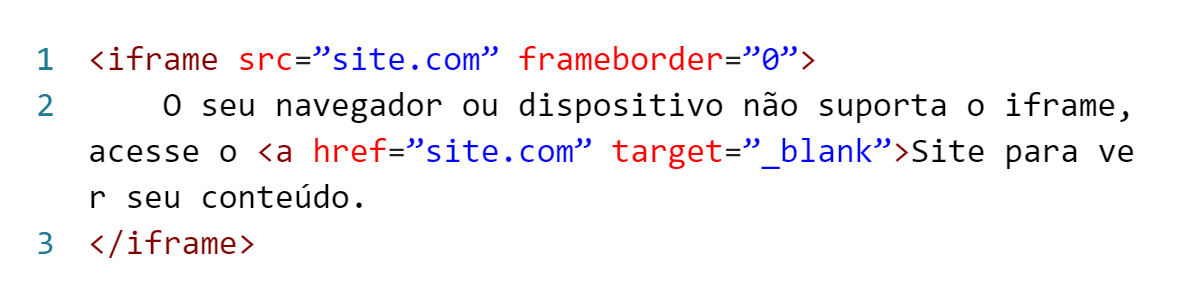
## Iframes

Os **‘iframes’** permitem incorporar um documento ou página web dentro de outra página. A sigla “iframe” significa “**inline frame**”.

### Compatibilidade com Navegadores

É importante ressaltar que os iframes não funcionam em todos os navegadores, porém, é possível exibir uma mensagem no lugar do iframe caso o navegador não consiga interpretá-lo.

Entre as tags do iframe (**‘<iframe>’**) pode-se deixar uma mensagem, que aparecerá caso o navegador não carregue o iframe.



No exemplo acima, o texto junto ao link que está dentro do texto aparecerá caso o iframe não possa ser carregado ou suportado.

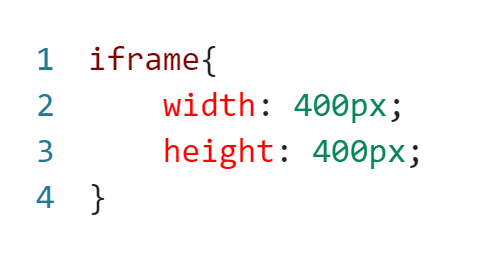
### Altura e Largura

Por padrão, os iframes possuem uma largura de 300 pixels e uma altura de 150 pixels (300x150). É possível mudar isso identificando os valores dos atributos **‘width’** e **‘height’**, que são largura e altura, respectivamente.



No exemplo acima, os valores foram alterados do padrão 300x150 pixels para 500x300 pixels.

É possível também especificar os valores de largura e altura em CSS, configurando o seletor **‘iframe’**.



A prioridade são as folhas de estilo, caso sejam configurados também em HTML a largura e altura do iframe.

### Borda do iframe

O atributo **‘frameborder’** aceita o valor **‘0’** e qualquer outro valor numérico. Quando o valor for **‘0’**, ele não exibirá uma borda, já se for qualquer outro número, irá ser exibida uma borda.

#### Borda personalizada

Para usar uma borda personalizada, mantenha o valor do **‘frameborder’** para **‘0’** e configure uma borda em CSS normalmente.

### Conteúdo local no iframe

Além de páginas web, é possível colocar página locais em um iframe. Para isso, é necessário indicar o diretório da página no atributo **‘src’**.

### Abrindo uma página por link

É possível abrir uma página no iframe por meio de links. Para isso, é necessário um link ligado à página e um target com o **‘name’** do iframe.

Não é necessário que esse iframe tenha um diretório já especificado, podendo deixar o ‘src’ em branco.

### Elementos HTML dentro do iframe

Ao invés de usar o atributo ‘src’, pode-se usar o atributo **‘srcdoc’**, que permite usar elementos HTML, como parágrafos, títulos, imagens, etc.

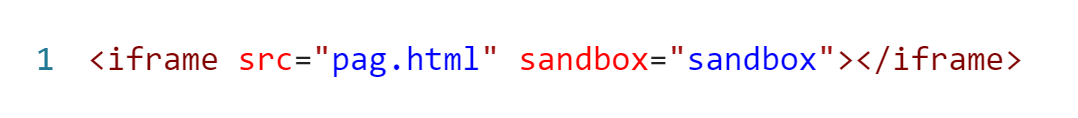
### Iframes mais seguros

Os iframes podem permitir ataques ao site, e por isso é importante torná-los seguros, limitando suas ações até o máximo possível. Isso, claro, é bem desnecessário quando a página carregada pelo iframe é do próprio servidor e tem sua total confiança.

Existem alguns atributos que limitam as ações do iframe, o que ajuda a torná-lo mais seguro.

#### ‘sandbox’

O atributo **‘sandbox’** pode receber a si mesmo como valor, ou seja:



O valor ‘sandbox’ é a restrição máxima, mas o valor desse parâmetro também pode ser: **‘allow-forms’**, **‘allow-modals’**, **‘allow-scripts’**, entre outros. Cada um permitindo alguma ação em específico.

Também é possível permitir mais de uma coisa, colocando mais de um valor, separando-os com espaço.

#### ‘referrerpolicy’

Esse atributo é a política de referência. Seu valor padrão é ‘no-referrer-when-downgrade’, mas para tornar o iframe um pouco mais limitado, o valor é apenas **‘no-refferrer’**.



Assim, a página carregada dentro do iframe não terá permissão para coletar informações do usuário.

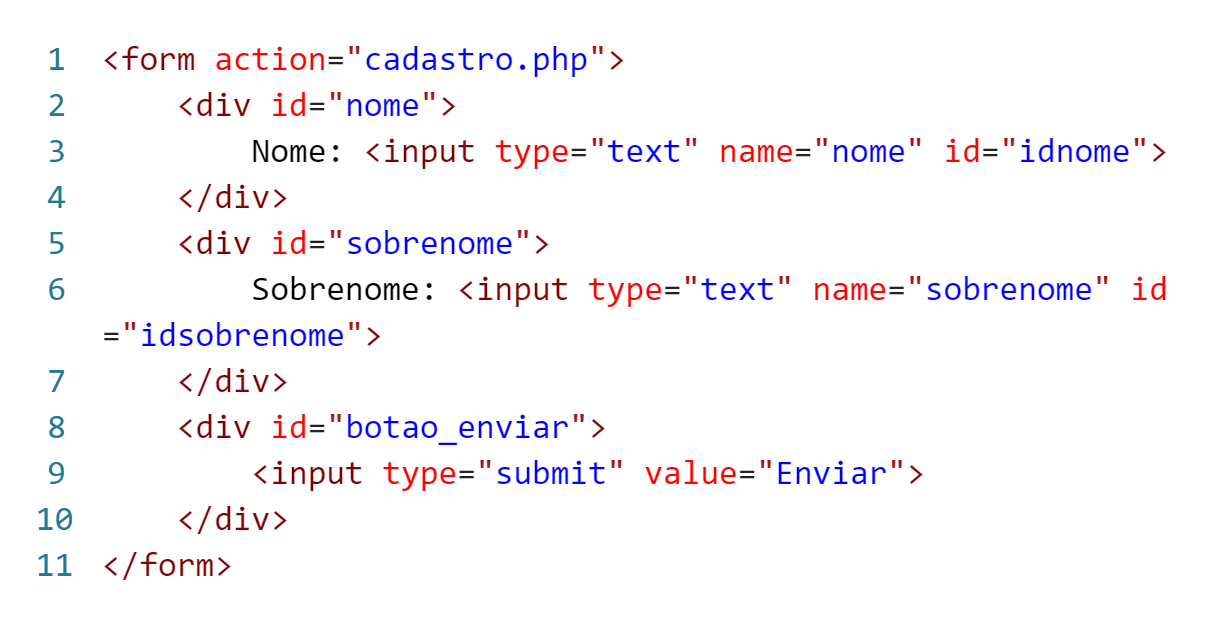
## Formulários em HTML

Os formulários são usados para que o servidor receba informações enviadas pelo usuário por meio de um campo de texto.

Para criar um formulário em HTML, use a tag **‘<form>’**.

### Aplicando o ‘<form>’

A tag **‘<form>’** precisa ser fechada, ou seja, o formulário fica entre **‘<form>’** e **‘</form>’**.

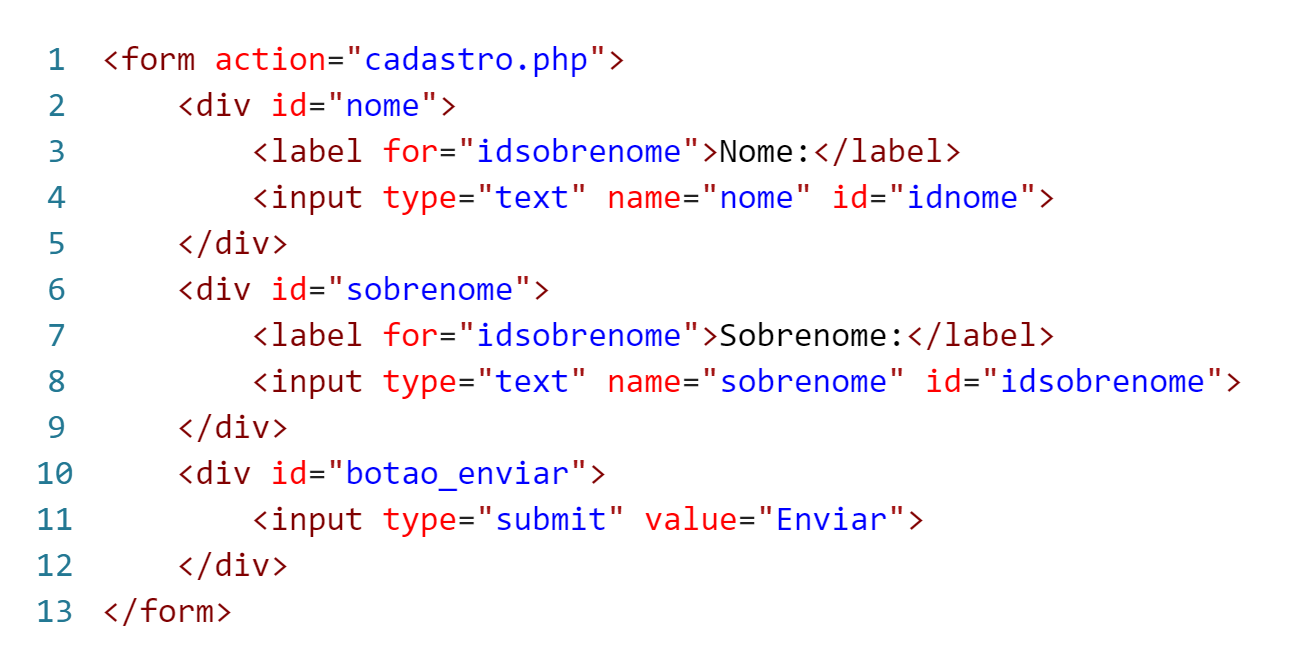


No exemplo acima, é pedido o nome e o sobrenome do usuário, e no fim, há um botão de enviar.

Ao enviar, os dados serão enviados ao documento “cadastro.php”, especificado no atributo **‘action’**, no início do código. Lembrando que não precisa ser PHP, especificamente.

### ‘<label>’

A tag **‘<label>’** é fundamental em um formulário. Ela cria uma relação entre um texto e seu ‘<input>’. No código anterior, não há nenhuma relação entre o texto “nome” e o ‘<input>’, não contribuindo assim, com os mecanismos de busca.



### Métodos de envio

Existem dois métodos de envio no HTML, o método **GET** e o método **POST**.

#### Método get

O método **get** é o método padrão caso não especificado o método post. Com ele, os dados serão visíveis na URL.

O método **get** é recomendado para quando os dados não forem sensíveis ou privados, pois o mesmo não é seguro para ocultar tais informações.

Tem o limite de 3000 bytes, o que seria mais ou menos 3000 caracteres.

Ótimo para compartilhamento de URL, sendo um ponto muito positivo.

Por outro lado, o mesmo não permite o envio de arquivos como imagens e vídeos.

#### Método post

O método **post** oculta os dados enviados na URL, mas ainda assim é possível ver esses dados de outras maneiras.

Ao contrário do get, permite o envio de arquivos, como imagens e vídeos.

### Configurações do ‘<input>’

Existem diversos atributos de configuração do **‘<input>’**.

* **‘type’:** Define o tipo de entrada. Pode ser ‘text’, ‘password’, ‘email’, ‘number’, entre outros.
* **‘value’:** Define o valor padrão para aquele campo de entrada
* **‘placeholder’:** Exibe um texto dentro do campo de entrada.
* **‘required’:** Torna o preenchimento do campo de entrada obrigatório.
* **‘disabled’:** Desativa qualquer interação com o campo de entrada.
* **‘readonly’:** Torna o campo de entrada somente leitura, assim o usuário só pode ver o valor, mas não pode modificá-lo.
* **‘size’:** Define a largura do campo de entrada em caracteres.
* **‘maxlength’, ‘minlength’:** Limitam a o máximo e o mínimo de caracteres permitidos no campo de entrada.
* **‘max’ e ‘min’:** Estabelecem os valores mínimo e máximo para campos numéricos.
* **‘step’:** Especifica o quanto aumenta ou diminui (incremento ou decremento) dos botões em campos numéricos.
* **‘autocomplete’:** Define se o navegador vai ou não completar automaticamente ou sugerir os valores do campo de entrada.
* **‘pattern’:** Define um padrão de entrada, como por exemplo, definir “x xxxx-xxxx” para um campo numérico onde o valor deve ser um número de telefone.
* **‘autofocus’:** Determina se o campo de entrada deve ou não receber o foco automaticamente quando a página é carregada.
* **‘multiple’:** Permite a seleção múltipla em campos de arquivo.
* **‘accept’:** Especifica os tipos de arquivo que podem ser enviados através de campos de arquivo.
* **‘form’:** Especifica a qual formulário o campo de entrada pertence.

### Tipos de ‘type’

O atributo ‘type’ suporta diversos valores. Abaixo uma lista com os mais utilizados.

* **‘text’:** Cria um campo de entrada para inserção de texto, é completamente livre.
* **‘password’:** Cria um campo de entrada de texto onde os caracteres digitados são mascarados, geralmente exibidos como asteriscos ou pontos, para ocultar a senha.
* **‘number’:** Cria um campo de entrada numérico. Atributos como **‘min’**, **‘max’** e **‘step’** controlam os valores aceitos.
* **‘radio’:** Cria um campo de entrada de seleção que permite o usuário selecionar apenas uma opção. Nesse caso, para que apenas uma opção seja marcada por vez, o nome de todas as alternativas deverão ser o mesmo.
* **‘checkbox’:** Cria um campo de entrada de seleção que permite ao usuário selecionar uma ou várias opções entre um conjunto de opções.
* **‘file’:** Cria um botão que permite aos usuários selecionar e enviar arquivos de seu dispositivo local. Lembre-se: quando utilizar o tipo file, use como método de formulário o **post**. O motivo já foi dito antes.
* **‘submit’:** Cria um botão de envio de formulário que envia os dados do formulário para o servidor.
* **‘reset’:** Cria um botão que redefine os valores do formulário para seus valores padrão.
* **‘time’:** Cria um campo de entrada para seleção de horário.
* **‘date’:** Cria um campo de entrada para seleção de data.
* **‘range’:** Cria uma barra de deslizamento horizontal. Valores como **‘max’**, ‘min’ e **‘step’** ajudam a controlar seus valores.
* **‘color’:** Cria um campo de entrada para seleção de cor.
* **‘url’:** Cria um campo de entrada de texto, feito especificamente para URLs, validando a entrada do usuário de acordo com as regras de formato de URL.
* **‘tel’:** Cria um campo de entrada para números de telefone.
* **‘search’:** Cria um campo de pesquisa. Geralmente usado para pesquisas em mecanismos de busca.
* **‘hidden’:** Cria um campo de entrada oculto que não é visível para o usuário, mas que pode ser usado para armazenar dados no formulário.
* **‘week’:** Cria um campo de entrada para seleção de semana e ano.
* **‘month’:** Cria um campo de entrada para mês e ano.
* **‘datetime’:** Cria um campo de entrada para seleção de data e horário.
* **‘datetime-local’:** Cria um campo de entrada para seleção de data e horário, sem considerar o fuso horário.
* **‘image’:** Cria um botão de imagem que, quando clicado, envia as coordenadas do ponto onde o usuário clicou no botão.
* **‘button’:** Cria um botão de ação genérico que pode executar uma função JavaScript ou realizar uma ação específica quando clicado.

### Agrupamento de Campos

A tag **‘<fieldset’>** agrupa vários campos de entrada em uma área, criando uma borda em volta.

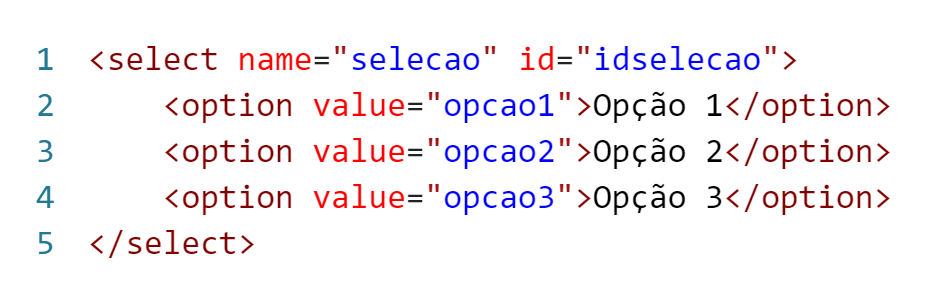
Todos os inputs de labels devem estar envelopados dentro desta tag para compartilharem uma única borda externa.

#### Legenda

Dentro do ‘<fieldset>’ pode ser colocada uma outra tag, a tag **‘<legend>’**. Ela cria uma legenda no topo da área.

### ‘<select>’

A tag **‘<select>’** permite ao usuário escolher uma das opções criadas pelo desenvolvedor. Cada opção é definida pela tag **‘<option>’**, que deve conter a opção dentro de si.

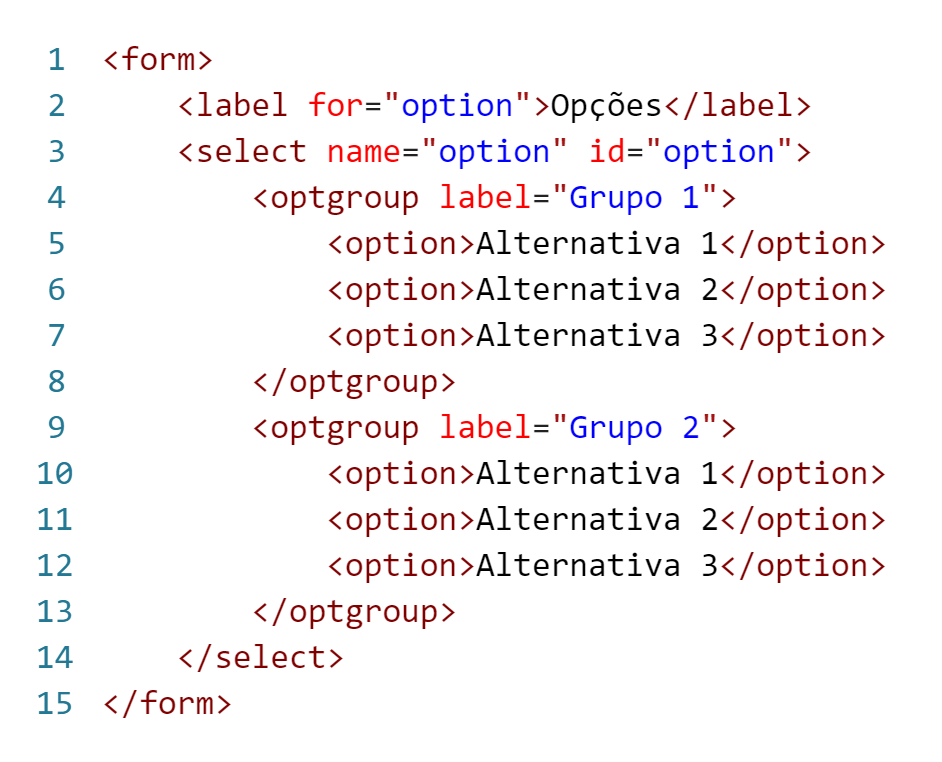


O atributo **‘value’** recebe o valor que vai ser enviado quando o usuário escolher a opção.

#### Grupo de opções

A tag **‘<optgroup>’** pode conter várias opções (**‘<option>’**) dentro de si, formando um grupo de opções.

Essa tag deve conter o atributo **‘label’**, que tem como valor o nome do grupo de opções.



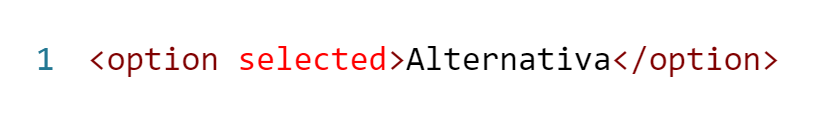
#### Seleção Múltipla

Caso adicionado o atributo **‘multiple’** à tag ‘<select>’, torna possível que o usuário escolha mais de uma opção.



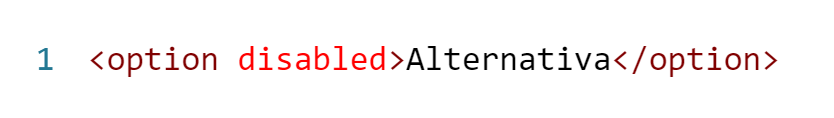
#### Seleção padrão

O atributo ‘selected’ adicionado à tag ‘<option>’ fará com que a opção relativa a essa opção seja selecionada automaticamente quando o formulário for carregado.



#### Desabilitar opção

Ao adicionar o atributo **‘disabled’** em um **‘<option>’**, a opção passa a ser não selecionável, se tornando inativa.

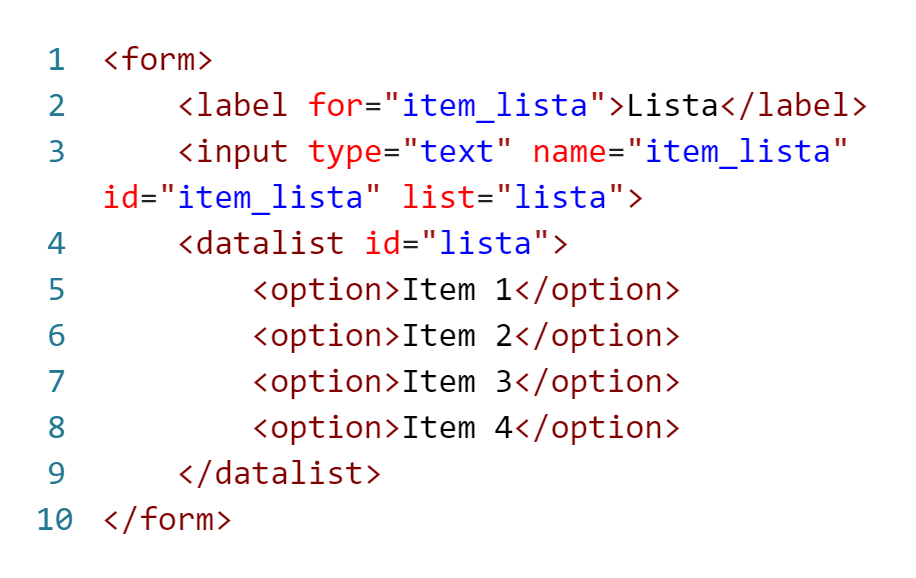


### ‘<datalist>’

A função da tag **‘<datalist>’** é parecida com a ‘<select>’, porém, ela permite o usuário digitar valores diferentes dos determinados pelo desenvolvedor ao ser combinada com um ‘<input>’.

Dentro dessa tag também pode ser usado a tag **‘<option>’**, assim como no ‘<select>’.

Para criar uma relação entre as opções e a caixa de texto em que o usuário pode digitar, é necessário usar o atributo **‘list’** dentro da tag **‘<input>’**.



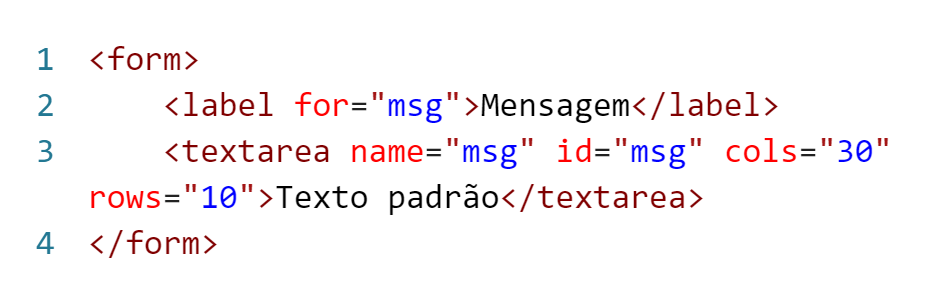
No exemplo acima, o id **“lista”**, determinado na tag ‘<datalist>’, cria uma relação entre as opções e o campo de texto por meio do atributo **‘list’**, especificado no ‘<input>’.

A tag **‘<datalist>’** ajuda o usuário dando a ele opções que podem até mesmo ajudá-lo a escrever uma não especificada, mas relacionada de qualquer forma. Tudo isso por meio de uma lista que é suspensa sobre o campo de entrada.

### ‘textarea’

A tag **‘<textarea>’** cria um campo de entrada de texto personalizado, permitindo aos usuários inserirem dados em um formato de várias linhas.

Os atributos para personalizar a área do campo de entrada são **‘cols’**(colunas) e **‘rows’**(linhas). Em ambos os atributos os valores devem ser numéricos, especificando uma quantidade de colunas e linhas.



No exemplo acima, é determinado o valor 30 para o atributo de colunas e o valor de 10 para as linhas, ou seja, a área do campo de entrada será de 30 colunas e 10 linhas.

Também veja que essa tag trabalha junto com a tag **‘<label>’**, criando uma relação entre o texto “Mensagem” e o campo de entrada de texto por meio do atributo **‘name’**.

O valor digitado entre as tags de abertura e fechamento da tag **‘<textarea>’** definem um texto padrão ao campo de entrada.

### ‘<output>’

A tag **‘<output>’** é usada para exibir o resultado de um cálculo ou script. Ela é geralmente usada em conjunto com a tag ‘<input>’ para criar interações entre o usuário e a página.

Não terá detalhes sobre JavaScript aqui, por isso, não terá detalhes sobre essa tag.

# CSS

## Tipos Aplicações de CSS

Existem três formas de aplicar as CSS em sua página HTML: **inline**, **internal** e **external**.

### CSS *Inline*

Adiciona o estilo na própria linha do elemento a ser estilizado em HTML por meio do atributo **‘style’**.



No exemplo acima, tag ‘<h1>’ recebeu o atributo **‘style’**, que tem como valor um código CSS, que altera a cor do texto para azul-escuro.

As CSS Inline devem ser evitadas, e o motivo é dificuldade para fazer alterações futuras, tendo que modicar elemento por elemento. Além disso, polui muito o código HTML.

### CSS *Internal*

As configurações de estilos são definidas pela tag **‘<style>’** que deve ser aplicada na área ‘<head>’ do documento HTML.



Todos os estilos devem estar no escopo da tag **‘<style>’**, que, por sua vez, precisa de fechamento.

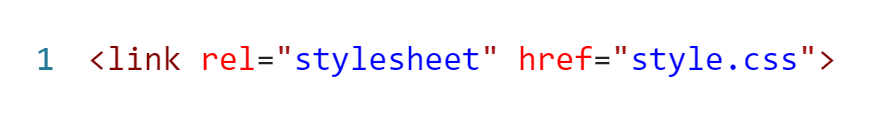
No exemplo acima, todos os títulos de nível 1 (‘h1’) serão da cor azul-escuro, e todos os títulos de nível 2 (‘h2’) serão da cor azul.

Além disso, o que foi configurado para o ‘body’ é a cor dos textos para vermelho-escuro e alterado a cor de fundo, além de uma tipografia(fonte) diferente.

### CSS *External*

Esse é o método de aplicação das CSS mais recomendado, e o melhor em caso de grandes projetos, pois o objetivo é separar os estilos do código HTML em um novo arquivo.

A tag **‘<link>’** em HTML com o atributo **‘stylesheet’** cria uma ligação do arquivo HTML com o arquivo CSS, onde está armazenado as linhas de código dos estilos.



O valor do atributo **‘href’** é o nome do arquivo CSS.

#### Estrutura de Arquivo CSS



Na primeira linha do código acima, existe uma regra, que determina o formato dos caracteres, no caso, definido para “UTF-8”.

Na segunda linha, há um comentário, explicando o uso da primeira linha.

## Comentários em CSS

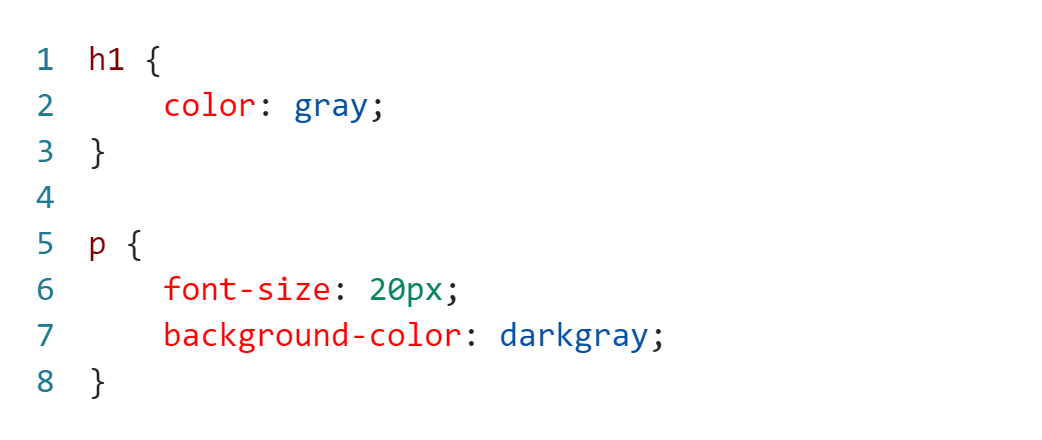
Tudo que estiver entre **‘/\* \*/’** será considerado comentário em CSS.



## Seletores em CSS

Os seletores são usados para especificar quais os elementos serão estilizados.

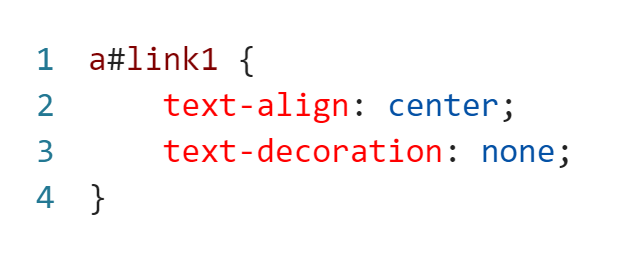
### Estilizar por ‘id’ e ‘class’



No exemplo acima, os elementos **‘h1’** e **‘p’** são também seletores, e todos os títulos nível 1 terão o texto de cor cinza e todos os parágrafos terão um texto de 20px de tamanho e uma cor de fundo cinza-escuro.

#### ‘id’

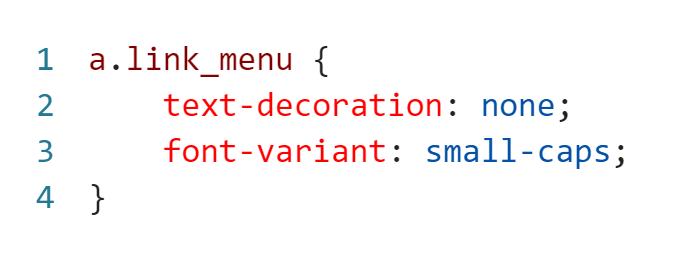
Para estilizar um elemento vinculado a um ‘id’, deve-se criar um seletor, escrevendo **‘#’** + **nome do ‘id’**.



Isso também poderia ser escrito sem a especificação do elemento ‘a’.

#### ‘class’

Com as classes é possível estilizar vários elementos ao mesmo tempo. Para isso, ao invés de utilizar ‘#’ como no ‘id’, deve-se usar ‘.’. Ficando **‘.’** + **nome da classe**.

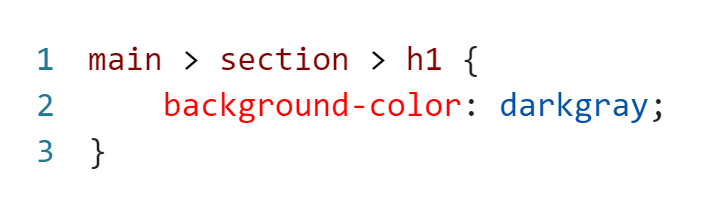


Assim como o ‘id’, não é necessário que o elemento seja especificado antes do nome da classe.

### Combinadores

São usados para especificar os seletores de forma mais precisa.

**‘>’**



No exemplo acima, o sinal de ‘**>**’ serve para especificar que o próximo elemento está dentro do anterior.

Poderia também ser escrito sem especificar todo o caminho até o elemento:



Não é necessário especificar todo o caminho até o elemento, mas nesse caso não deve haver o sinal **‘>’**.

**‘+’**

Faz referência ao seu “irmão adjacente”, ou seja, o próximo seletor que compartilha o mesmo elemento pai, será influenciado.



No código acima, o próximo elemento ‘p’ que compartilha o mesmo elemento pai do elemento ‘div’ ganhará uma cor de fundo cinza. Porém, o próprio elemento ‘div’ não **será influenciado**.

**‘~’**

É parecido com o modificador ‘+’, porém ao invés de ser apenas o seletor adjacente, são todos os seletores.



No código acima, todos os elementos ‘p’ após o elemento ‘div’ ganharão uma cor de fundo cinza. Porém, o próprio elemento ‘div’ **não será afetado**.

## Medidas em CSS

É recomendado o uso de **em** e **px**, além de **%** e **vh**, que são para situações específicas.

* **‘px’ (pixels)**: É a mais comum e representa um pixel na tela.
* **‘em’**: É relativo à altura do elemento pai. Por exemplo, se o tamanho da fonte do elemento pai for de 16 pixels por padrão, ‘font-size: 1.5em;’ definirá o tamanho como 24 pixels(1.5 vezes o tamanho da fonte do pai).
* **‘rem’**:É relativo ao tamanho da fonte do elemento raiz, definido em ‘html’.
* **% (porcentagem)**: É relativo ao valor de referência.
* **‘vw’ (viewport width)**: É relativo à largura da janela de visualização (viewport).
* **‘vh’ (viewport width)**: é relativo à altura da janela de visualização (viewport).

## Fontes

Para a aplicação e configuração de fontes em CSS, existem as seguintes propriedades:

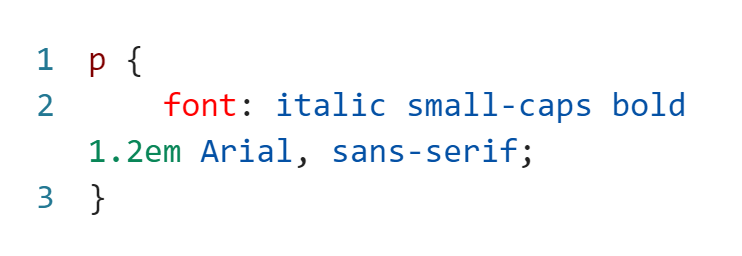
* **‘font-style’:** É o estilo da fonte. Pode ser **‘italic’**, **‘oblique’** ou **‘normal’**.
* **‘font-variant’:** O valor **‘small-caps’** torna o texto inteiro em maiúsculo, porém as letras que seriam minúsculas têm um tamanho um reduzido. O valor **‘normal’** é o valor padrão.
* **‘font-weight’:** Define peso da fonte. Pode ser um **‘lighter’**, **‘bold’**, **‘bolder’** e **‘normal’**. Além de valores como 100, 200, ..., 800, 900.
* **‘font-size’:** Define o tamanho da fonte.
* **‘line-height’:** Define a altura da linha do texto.
* **‘font-family’:** Define a família tipográfica.

### Shorthand

A propriedade **‘font’** permite especificar várias configurações de fontes em apenas uma propriedade.

A ordem é a seguinte:

1. **‘font-style’**
2. **‘font-variant’**
3. **‘font-weight’**
4. **‘font-size’**
5. **‘font-family’**



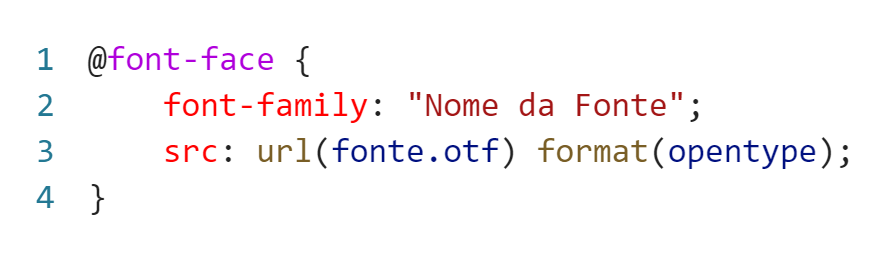
### Fontes Externas

É possível usar fontes externas por meio de arquivos ou importações via URL.

Os arquivos de fonte podem ser baixados por meio de sites, o [www.dafont.com](http://www.dafont.com) é um exemplo.

Existem dois tipos de arquivo de fonte: **otf** e **ttf**. A diferença entre eles é a compatibilidade com os navegadores.

Para aplicar a fonte em elementos é necessário criar uma regra:



Na área ‘font-family’ deve estar escrito o nome que deseja para a fonte (não precisa ser o nome real da fonte). É por esse nome que ela será referenciada.

Na **‘url’**, deve ser especificado o diretório do arquivo de fonte.

No **‘format’** deve ser especificado o formato da fonte. Os formatos são:

* **‘opentype’:** Para fontes **otf** (algumas fontes opentype são ttf)
* **‘opentype’:** Para fontes **ttf**
* **‘embedded-opentype’**
* **‘truetype-att’:** Apple Advanced Typography
* **‘svg’**

Procure saber o tipo da fonte para saber qual o valor ideal.

#### Importação de Fontes

O Google Fonts é um exemplo de sites em que se pode importar fontes.

Nesses sites para importação, existe uma área que especifica alguma regra em CSS ou ‘<link>’ em HTML para a importação.

## Transformações de Texto

### ‘text-transform’

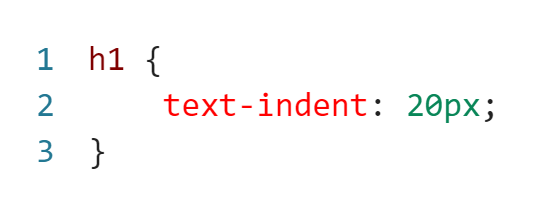
A propriedade **text-transform** altera um texto, seja para que todas as letras se transformem em maiúsculas, minúsculas, entre outras opções.

Valores:

1. **‘capitalize’:** Transforma a primeira letra de cada palavra em maiúscula.
2. **‘lowercase’:** Converte o texto inteiro para minúsculo.
3. **‘uppercase’:** Converte o texto inteiro para maiúsculo.

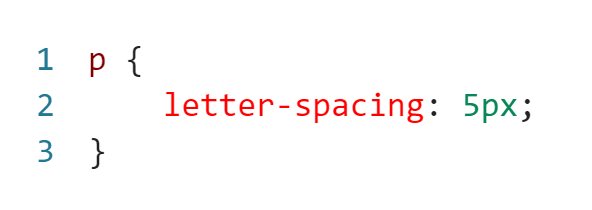
### ‘text-indent’

Adiciona um espaço de indentação ao começo do texto.



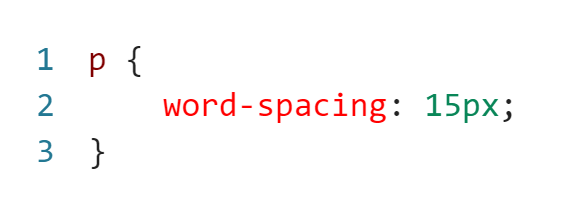
### ‘letter-spacing’

Essa propriedade define um espaçamento entre os caracteres.



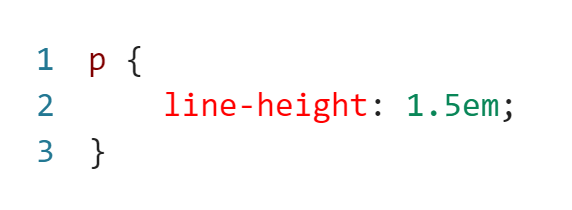
### ‘word-spacing’

Essa propriedade é usada para definir o espaçamento entre as palavras.



### ‘line-height’

Essa propriedade determina o tamanho do espaçamento entre as linhas.



### ‘white-space’

Determina como o espaço em branco dentro do elemento deve ser comportar.

Valores:

1. **‘normal’:** O texto será quebrado quando necessário.
2. **‘nowrap’:** Não haverá quebra de linha.
3. **‘pre’:** O texto será mantida da forma que foi digitada. Função igual a tag ‘<pre>’ em HTML.
4. **‘pre-line’:** Parecido com o valor ‘pre’, porém excluindo os espaços de indentação.
5. **‘pre-wrap’:** Esse valor é como o valor ‘pre’, mas ele quebra de linha quando necessário (pelo menos é isso que está na documentação).

## Estilização de *Links*

Como você provavelmente já sabe, um link pode ser estilizado com propriedades simples, como ‘color’, ‘font-size’, ‘font-family’, ‘text-decoration’, ‘background’, etc, e é por que vamos focar mais em estilos focadas em seus estados.

Existem quatro estados de link:

1. **‘a:link’:** Quando não visitado pelo usuário.
2. **‘a:visited’:** Quando já visitado pelo usuário.
3. **‘a:hover’:** Quando passado o mause sobre o link.
4. **‘a:active’:** No momento em que for clicado.

## Caixas

A grande maioria dos elementos HTML são caixas. Elas armazenam conteúdos dentro delas, até mesmo outras caixas.

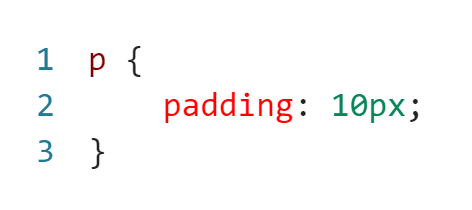
### Estrutura de uma Caixa

Uma caixa é composta por:

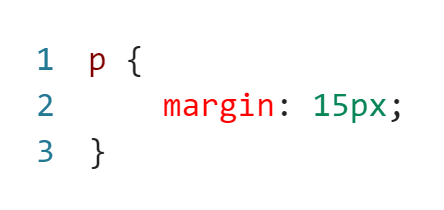
* ***Content*:** É o conteúdo, e é por padrão, o centro da caixa.
* **‘padding’:** É o preenchimento que fica entre o conteúdo e a borda.
* **‘border’:** É a borda da caixa.
* **‘outline’:** É um contorno após a borda.
* **‘margin’:** Cria uma margem ao lado de fora da caixa.

#### ‘padding’ e ‘margin’

O **‘padding’** é uma propriedade, e seu valor deve alguma medida para indicar o espaçamento entre o conteúdo e a borda.

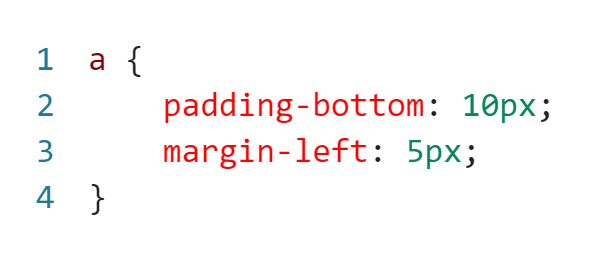


Já a propriedade **‘margin’**, assim como o ‘padding’, cria um espaçamento, porém no lado de fora da caixa.

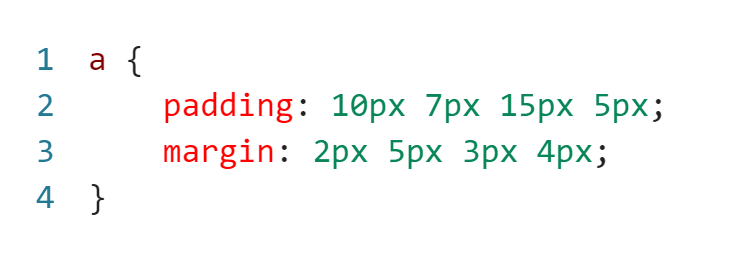


Caso o valor de ‘margin’ seja **‘auto’**, a caixa será centralizada.

Para ambos, **‘padding’** e **‘margin’**, existe a possibilidade de especificar o valor de espaçamento apenas para um lado específico:



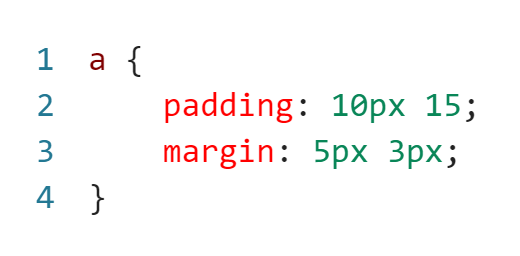
Especificando quatro valores na propriedade em ambas as propriedades **‘padding’** e **‘margin’**, é possível definir o valor dos quatro lados em uma única propriedade.



A ordem é a seguinte:

1. **Top** (Cima)
2. **Right** (Direita)
3. **Bottom** (Baixo)
4. **Left** (Esquerda)

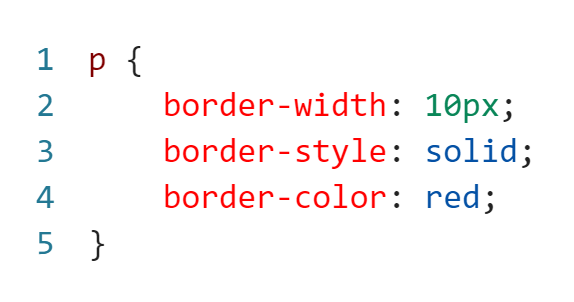
Especificando apenas dois valores em ambas as propriedades **‘padding’** e **‘margin’**, será definido os valores de espaçamento para cima e para baixo no primeiro valor, e da direita e esquerda no segundo valor.



#### ‘border’ (Borda)

Existem três configurações para borda:

1. **‘border-width’:** Largura da borda.
2. **‘border-style’:** Estilo da borda.  
   Seus valores são:
   * **‘solid’:** Cria uma borda sólida.
   * **‘dashed’:** Cria uma borda tracejada.
   * **‘dotted’:** Cria uma borda pontilhada.
   * **‘double’:** A borda será preenchida, se tornando uma borda dupla.
   * **‘groove’:** Cria um efeito 3D na borda. O efeito depende da cor da borda.
   * **‘ridge’:** Têm o efeito do ‘groove’, porém invertido.
   * **‘inset’:** Cria um efeito 3D com a cor escolhida, aplicando a cor escolhida na borda de baixo e um tom mais escuro da mesma cor na borda de cima.
   * **‘outset’:** Cria um efeito 3D com a cor escolhida, aplicando a cor escolhida na borda de cima e um tom mais escuro da mesma cor na borda de baixo.
   * **‘none’:** Não define uma borda.
   * **‘hidden’:** Cria uma borda oculta.
3. **‘border-color’:** Define uma cor para a borda.



##### Shorhand

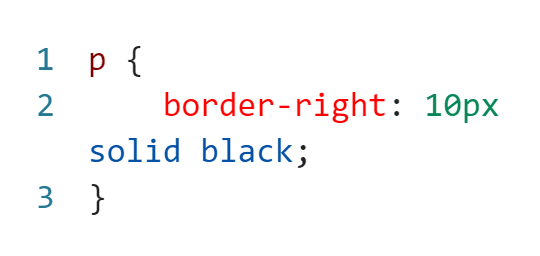
É possível especificar ‘border-width’, **‘border-style’** e **‘border-color’** usando uma única propriedade: **‘border’**.

Os valores devem ser separados por espaço e seguir a seguinte ordem:

1. **‘border-width’**
2. **‘border-style’**
3. **‘border-color’**

##### Estilo para um lado específico da borda

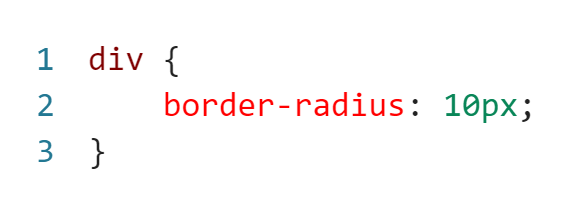
É possível também modificar um lado específico da borda, com propriedades como:



No exemplo acima, apenas o lado direito obterá esses estilos.

##### Cantos Arredondados

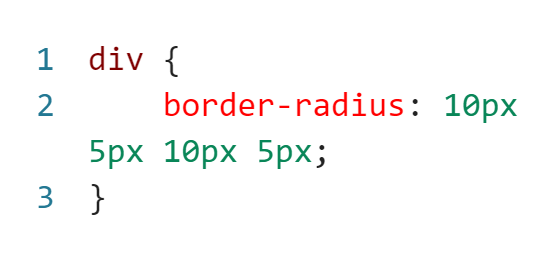
O valor da propriedade **‘border-radius’** define o quão arredondado será os cantos das bordas.



É possível indicar quatro valores, relativo aos quatro campos do elemento.

A ordem é a seguinte:

1. Superior esquerdo
2. Superior direito
3. Inferior direito
4. Inferior esquerdo

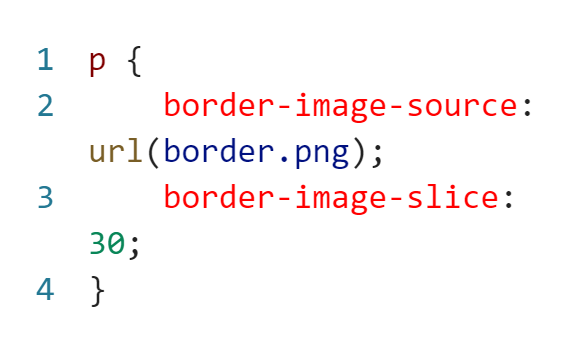


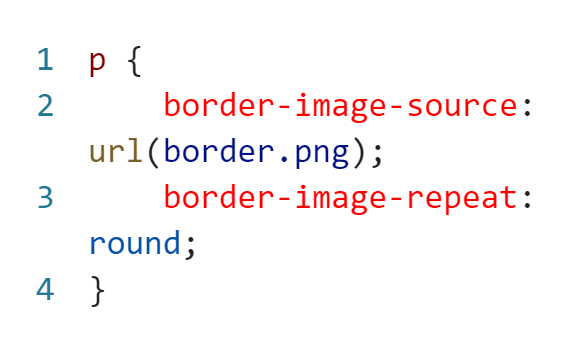
##### Imagem como Borda

Primeiramente, uma imagem precisa ser feita especialmente para ser uma borda.

O recomendado é que a imagem tenha 80x80 pixels de tamanho.

A propriedade **‘border-image’** recebe três valores:

1. **‘border-image-source’:** Define a imagem a ser usada como borda por meio da função **‘url()’**.  
   
2. **‘border-image-slice’:** Seu valor deve ser numérico, relativo ao fatiamento da borda.  
   
3. **‘border-image-repeat’:** Indica como a borda vai aparecer na tela. Existem três valores:
   * **‘repeat’:** A borda se repetirá para ocupar a área.
   * **‘stretch’:** A borda irá se esticar para ocupar a área. Esse é o valor padrão.
   * **‘round’:** A imagem será redimensionada para que caiba dentro do espaço da borda.
   * **‘space’:** O espaço da imagem é distribuído para caber dentro do espaço da borda.

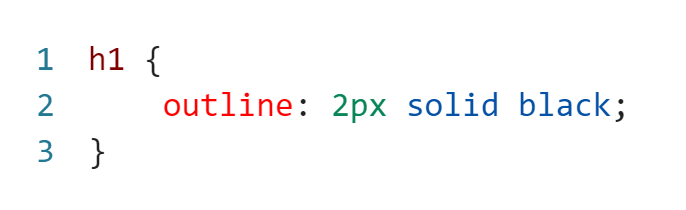


#### ‘outline’

Como já dito anteriormente, é um contorno desenhado fora das bordas. É bem parecido com o border, pois possui as mesmas propriedades, e uma a mais.

* **outline-style**
* **outline-color**
* **outline-width**
* **outline-offset**
* **outline** **(shorthand)**

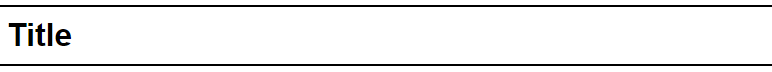
Falando apenas da propriedade não vista até agora, a propriedade **outline-offset** adiciona um espaço transparente entre o contorno e a borda do elemento.



O resultado sem o outline-offset é o seguinte:



Agora com uma a propriedade outline-offset definida com 15px.



Perceba que a área interna agora possui um pouco mais de espaço.

### Tags de Agrupamento

Existem duas tags de agrupamento genéricas, **‘<div>’**, que é *block-level*, e **‘<span>’**, que é *inline-level*.

#### Agrupamento Semântico

As tags de agrupamento semânticas servem organizar melhor o conteúdo, além de dar mais sentido a eles.

As tags de agrupamento semânticas são:

##### ‘<header>’

Cria áreas relativas a cabeçalhos. Pode ser o cabeçalho principal de um site ou até mesmo o cabeçalho de uma seção ou artigo. Normalmente inclui títulos <h1> - <h6> e subtítulos. Podem também conter menus de navegação.

##### ‘<nav>’

Define uma área que possui os links de navegação pela estrutura de páginas que vão compor o website.

##### ‘<main>’

É um agrupamento usado para delimitar o conteúdo principal de um site. Geralmente concentra as seções, artigos e conteúdos periféricos.

##### ‘<section>’

Cria seções para sua página. Ela pode conter o conteúdo diretamente no seu corpo ou dividir os conteúdos em artigos com conteúdos específicos. “Uma seção é um agrupamento temático de conteúdos, tipicamente com um cabeçalho”.

##### ‘<article>’

Um artigo é um elemento que vai conter um conteúdo que pode ser lido de forma independente e dizem respeito a um mesmo assunto. Podemos usar um <article> para delimitar um post de blog ou fórum, uma notícia, etc.

##### ‘<aside>’

É usado para definir um bloco de conteúdo que complementa o conteúdo principal, mas pode ser removido sem afetar significativamente o significado do conteúdo principal.

##### ‘<footer>’

Cria um rodapé para o site inteiro, seção ou artigo. É um conteúdo que não faz parte diretamente do conteúdo nem é um conteúdo periférico, mas possui informações sobre autoria do conteúdo, links adicionais, mapa do site, documentos relacionados.

#### Múltiplos Níveis

É possível ter um ou mais ‘<article>’ dentro de uma ‘<section>’ ou até mesmo criar ‘<section>’ dentro de um ‘<article>’. Não existem limitações quanto a isso.

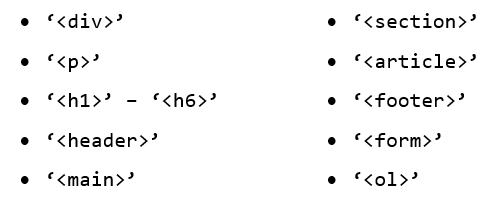
### Tipos de Caixas

Existem duas categorias de caixas, classificadas de acordo com seu comportamento: ***block-level*** e ***inline-level***.

#### *Block-level*

Um elemento ***block-level*** sempre se inicia em uma nova linha e vai ocupar a largura total do elemento onde está contido. Se não estiver contido em nenhuma outra caixa, vai ocupar a largura total do body.

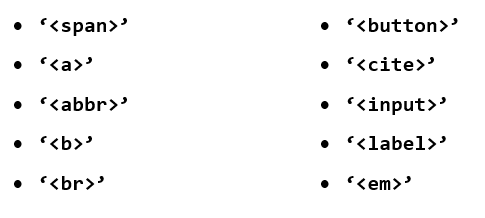
Alguns elementos ***block-level***.



#### *Inline-level*

O elemento irá começar no ponto exato de onde estava definido, ao contrário do *block-level*. Além disso, sua largura irá ocupar o espaço relativo ao seu conteúdo.

Alguns elementos ***inline-level***.



## ‘z-index’

A propriedade **‘z-index’** especifica a ordem em que um elemento fica em um plano de fundo, se ele vai ficar na frente ou atrás de um outro elemento.

Essa propriedade pode ter valores positivos ou negativos.

É importante lembrar que o menor valor ficará mais atrás, e o menor ficará mais a frente.

## ‘overflow’

A propriedade **‘overflow’** controla o que acontece com o conteúdo muito grande.

Valores:

1. **‘visible’:** O conteúdo dentro da caixa é cortado, e o conteúdo renderiza mesmo fora da caixa. É o valor padrão.
2. **‘hidden’:** O conteúdo dentro da caixa é cortado, e o conteúdo fora da caixa se torna invisível.
3. **‘scroll’:** O conteúdo é cortado e é adicionado uma barra de rolagem para possibilitar a vista do resto do conteúdo.
4. **‘auto’:** Parecido com o valor ‘scroll’, porém adiciona uma barra de rolagem apenas se necessário.

## ‘float’ e ‘clear’

São propriedades usadas para posicionar e formatar elementos.

#### ‘float’

A propriedade **‘float’** é usada para posicionar e formatar o conteúdo, podendo posicioná-lo em alguma direção.

Valores:

1. **‘none’:** O elemento não irá flutuar. É o valor padrão.
2. **‘left’:** O elemento será posicionado mais à esquerda no container.
3. **‘right’:** O elemento será posicionado mais à direita no container.

#### ‘clear’

A propriedade **‘clear’** determina como um elemento se comporta em relação a elementos flutuantes ao redor.

Valores:

1. **‘none’:** O elemento não será colocado abaixo dos elementos flutuantes. É o valor padrão.
2. **‘left’:** O elemento será empurrado para baixo dos elementos flutuantes à esquerda.
3. **‘right’:** O elemento será empurrado para baixo dos elementos flutuantes à direita.
4. **‘both’:** O elemento será empurrado para baixo dos elementos flutuantes à esquerda e à direita.

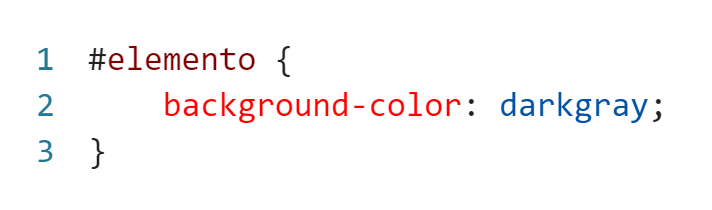
## Fundo (‘background’)

Define o fundo da caixa ou da página, caso aplicado ao ‘body’. Suas propriedades são:

* **‘background-color’**
* **‘background-image’**
* **‘background-position’**
* **‘background-repeat’**
* **‘background-size’**
* **‘background-attachment’**

### ‘background-color’

O valor desta propriedade deve ser uma cor.



### ‘background-image’

A propriedade ‘background-image’ permite definir uma imagem de fundo para a página ou caixa.



Por meio da função **‘url()’** é especificado o diretório da imagem no servidor local ou endereço da imagem na web.

#### Múltiplas imagens de fundo

É possível adicionar múltiplas imagens de fundo com a propriedade **‘background-image’**, separando as imagens com vírgula.



Na segunda linha do exemplo acima, foram definidas duas imagens de fundo. E nas próximas linhas foram especificados dois valores também separados por vírgula, cada valor relativo a uma imagem.

#### Gradientes

Os gradientes são considerados imagens.

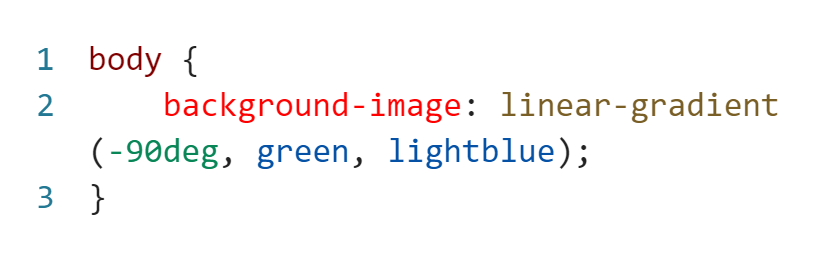
##### Gradiente Linear

Para criar um gradiente linear, deve se definir duas ou mais cores e a direção do gradiente na função **‘linear-gradient()’**.

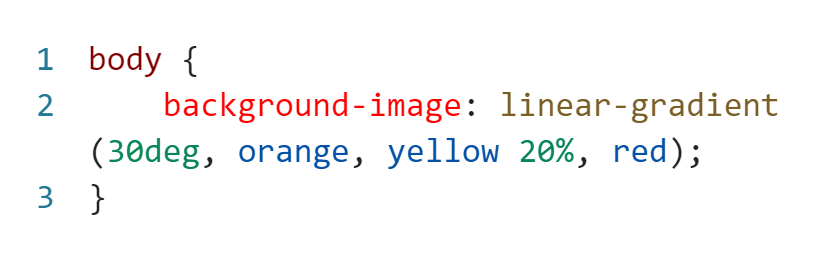


Deve se definir primeiro a direção e em seguida as cores, como visto no exemplo acima.

Além disso, é possível usar também os ângulos em graus para denifir as direções.



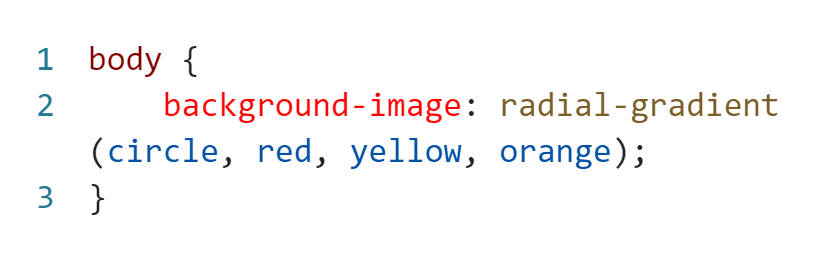
É possível também definir o espaço que cada cor ocupará no gradiente, para isso deve se especificar logo após a cor, um número com porcentagem.



##### Gradiente Radial

Para criar um gradiente radial (circular) é usada a função **‘radial-gradient()’**.

A sintaxe é bem parecida com o gradiente linear, porém deve-se definir **‘circle’** no que seria a direção.

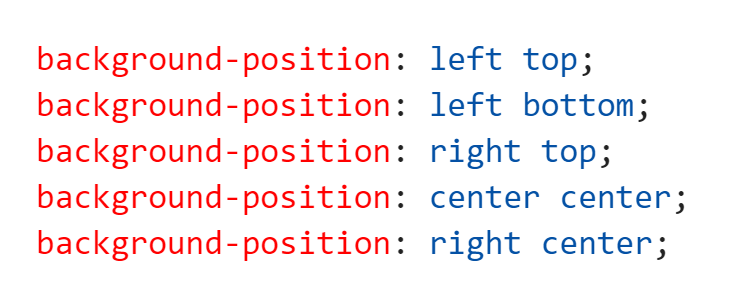


### ‘background-position’

A propriedade ‘background-position’ define a posição do ponto de ancoragem da imagem. Devem ser especificados dois valores escritos, o primeiro da coluna e o segundo da linha.

A coluna é definida pelos valores: **‘left’**, **‘center’** ou **‘right’**. Já a linha é definida pelos valores: **‘top’**, **‘center’** ou **‘bottom’**.

Os valores dependem da necessidade.



### ‘background-size’

A propriedade **‘background-size’** define o tamanho da imagem de fundo, aceitando valores com medidas ou valores escriots, sendo eles:

* **‘auto’:** É o valor padrão, onde a imagem é exibida em seu tamanho original.
* **‘cover’:** A imagem é redimensionada para que sua largura ou altura cubra completamente o espaço, mantendo a proporção e podendo haver recortes na imagem para se adequar ao tamanho do elemento.
* **‘contain’:** A imagem é redimensionada proporcionalmente dentro do elemento, mantendo a proporção e não havendo corte da imagem, mas podendo haver espaços vazios se a imagem não preencher todo o elemento.

### ‘background-attachment’

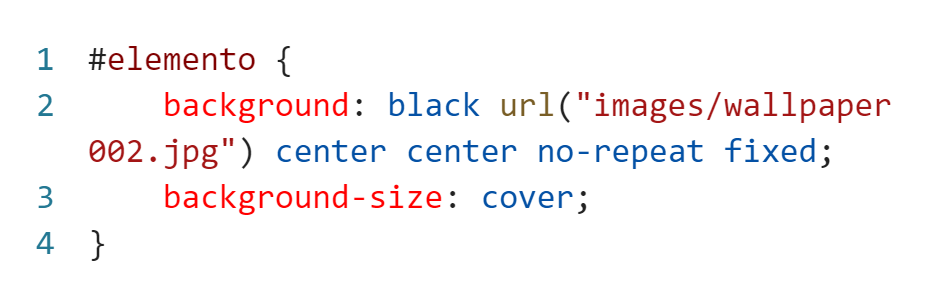
A propriedade ‘background-attachment’ define se a imagem deve rolar junto com o conteúdo ou se mesma fica fixa em relação à janela de visualização.

* **‘scroll’:** É o valor padrão. A imagem rola junto ao conteúdo.
* **‘auto’:** A imagem permanece fixa mesmo quando o usuário rola a página.
* **‘local’:** A imagem rola com o conteúdo do elemento em que está aplicada.

### Shorhand

Os valores para a propriedade **‘background’**, que é uma shorthand, devem ser escritos na seguinte ordem:

1. **‘background-color’**
2. **‘background-image’**
3. **‘background-position’**
4. **‘background-repeat’**
5. **‘background-attachment’**

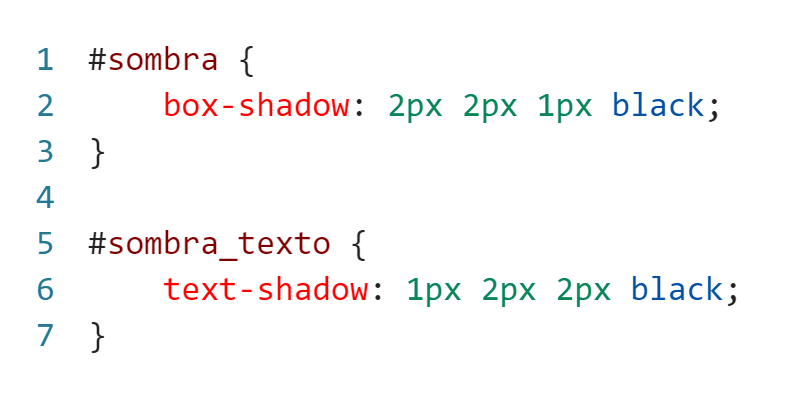


**Tamanho da imagem:** O **‘background-size’**, por algum motivo, não faz parte da shorthand e precisa ser definido separadamente.

## Sombras em CSS

Podem ser em caixas ou em textos, e em ambos seus valores são:

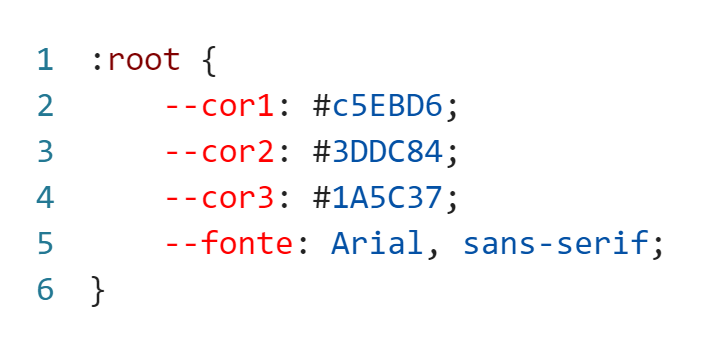
1. Posição horizontal da sombra.
2. Posição vertical da sombra.
3. Desfoque da sombra.
4. Alargamento da sombra (opcional)
5. Cor da sombra.



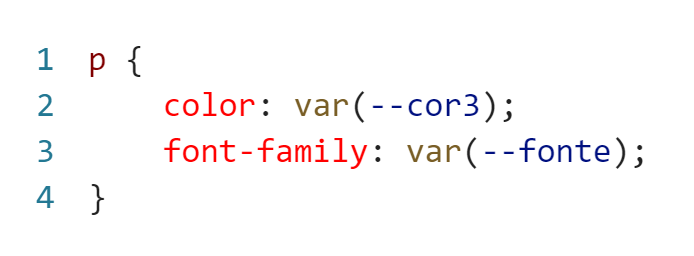
## Variáveis em CSS

Para criar variáveis em CSS, é necessário criar uma seletor especial chamado **‘:root’**, e entre as chaves devemos criar todas as variáveis que vamos utilizar no documento.

Toda variável possui dois hífens em sequência **--** no começo, após os hífens vêm o nome da variável, o valor após os dois pontos será o valor dessa variável, que pode ser uma cor, uma fonte, etc.



Para usar alguma dessas variáveis, é necessário usar a função **‘var()’**.



## Posições em CSS

O posicionamento de um elemento em CSS é definido pela propriedade **‘position’**, que pode ter cinco valores diferentes. São eles:

### ‘static’

O posicionamento padrão de todo elemento HTML é **‘static’**, que é o alinhamento ao canto superior esquerdo do corpo do documento ou do container em que está dentro.

### ‘fixed’

O valor **‘fixed’** faz com que o elemento não se mova na tela.

### ‘sticky’

Sua tradução seria “colado” ou “pegajoso”. Esse valor é parecido com o ‘fixed’, porém ele fica fixo em relação ao rolamento da página, e não da tela.

### ‘relative’

É usado para alterar a posição de um elemento tendo como base a posição inicial dele. Apenas aplicar esse valor a um elemento não o modificará em nada a sua posição, para isso é necessário aplicar o **‘top’** e **‘left’** para alterar sua posição de cima para baixo e esquerda para direita, tendo como referência a sua posição inicial.

### ‘absolute’

O posicionamento absoluto possui dois comportamentos diferentes. O primeiro é quando o elemento com esse valor possui um elemento pai de valor diferente de ‘static’, usando esse elemento pai como referência para ser posicionado.

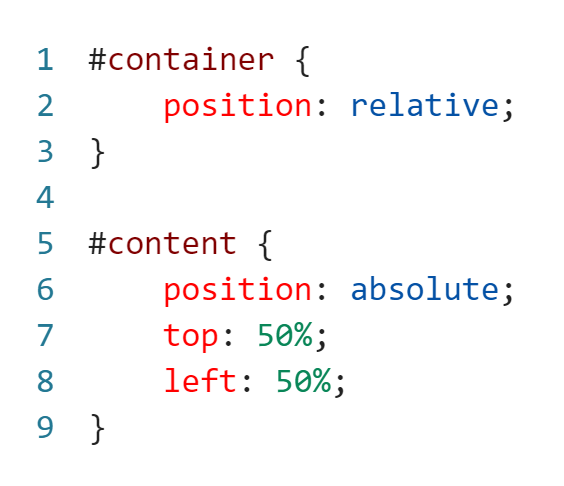
O segundo comportamento é quando o elemento absoluto não possui pai, ou o elemento pai possui a posição ‘static’. Nesse caso ele irá ser posicionado no canto superior esquerdo do documento, podendo até mesmo sobrepor a eles.

Em resumo, no primeiro comportamento o elemento absoluto tem um pai e o usa como base para ser posicionado. Já no segundo comportamento ele não possui um elemento pai ou o elemento pai possui valor ‘static’, e dessa forma ele usa o corpo do documento como base para seu posicionamento.

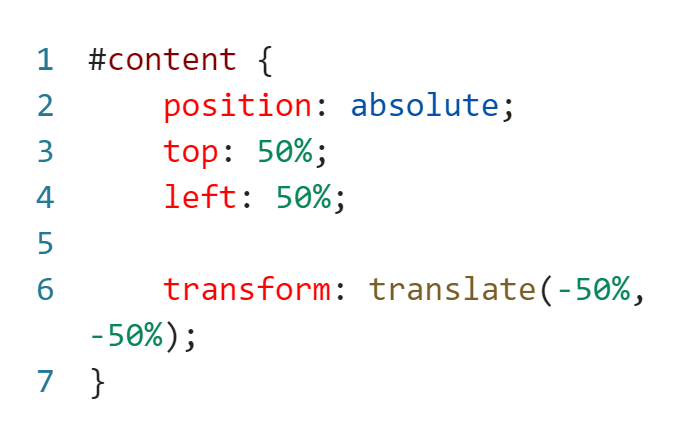
## Alinhamento Vertical

Para alinhar um elemento verticalmente, o elemento pai precisa ter a posição relativa e o elemento a ser alinhado deve ter a posição absoluta.

Após isso, deve-se alterar o **‘top’** e o **‘left’**, ambos para 50%.



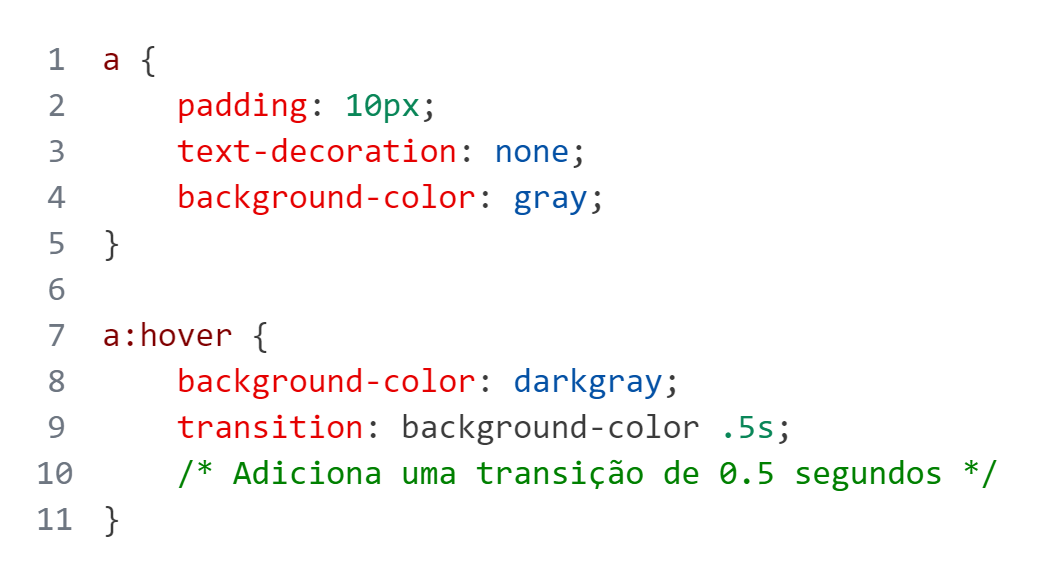
Por padrão, o ponto de ancoragem dos elementos é o canto superior esquerdo deles. Para alinhá-los, é necessário movê-los com a propriedade **‘transform’**.



## Transições

A propriedade **‘transition’** permite controlar a velocidade de uma animação quando há mudanças em determinada propriedade CSS.

Por exemplo, é possível definir em que velocidade um link obterá uma cor de fundo ao passar o mause sobre ele.



### Propriedades de Transição

A propriedade **‘transition’** é uma shorthand das propriedades:

1. **‘transition-property’**
2. **‘transition-duration’**
3. **‘transition-timing-function’**
4. **‘transition-delay’**

#### ‘transition-property’

Define a qual propriedade o efeito de transição deve ser aplicado.



Como pode ver, é possível definir mais de uma propriedade por vez.

#### ‘transition-duration’

Define a duração da transição. O seu valor pode ser em segundos e em milisegundos.

#### ‘transition-timing-function’

Define qual será o efeito da transição. Seus valores podem ser:

* **‘ease’:** É o valor padrão, onde a velocidade aumenta no meio da transição e diminui novamente no final
* **‘linear’:** A velocidade é igual durante toda a transição.
* **‘ease-in’:** A velocidade vai aumentando até o fim da transição.
* **‘ease-out’:** A transição se inicia rapidamente e vai diminuindo conforme a transição continua.
* **‘ease-in-out’:** Começo e fim lentos e meio acelerado.
* **‘cubic-bezier()’:** São quatro valores que definem a velocidade da transição de forma personalizada. Seus valores de 1 a 3 devem estar entre 0 e 1.
* **‘steps()’:** Exibe a transição ao longo de n paradas ao longo da transição. São dois valores, as n paradas e os termos de salto. São eles:
  + **‘jump-start’:** O primeiro salto ocorre assim que a transição começa.
  + **‘jump-none’:** Não há salto nenhum.
  + **‘jump-both’:** Inclui pausas.

## Animações

É possível criar uma sequência de animações em CSS. Para isso é necessário utilizar a regra **‘@keyframes’**.

## Transformações

Com a propriedade **‘transform’** é possível alterar a posição, escala, rotação e perspectiva de elementos, possibilitando inclusive manipular elementos no espaço 2D e 3D.

A propriedade **‘transform’** suporta vários valores:

* **‘translate()’**: Move um elemento em uma determinada direção. Devem ser especificadas as coordenadas X e Y. Por exemplo:



Com isso, moverá o elemento 50 pixels para a direita e 100 pixels para baixo. Para girar para a esquerda e para cima use valores negativos.

* **‘rotate()’**: Gira um elemento em torno de um ponto de referência específico, definindo o ângulo de rotação em graus. Por exemplo:



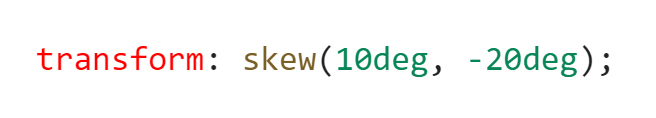
Com isso, irá girar o elemento em 45 graus no sentido horário.

* **‘scale()’**: Altera o tamanho de um elemento, especificando valores para escala horizontal e vertical. Por exemplo:

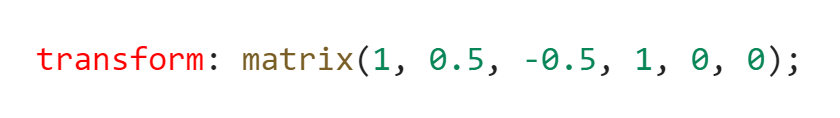


Com isso, aumentará o elemento em 1,5 vezes no eixo X e 2 vezes no eixo Y.

* **‘skew()’**: Aplica uma inclinação a um elemento, especificando valores para os eixos X e/ou Y. Ela permite distorcer a forma do elemento em um ângulo específico. Por exemplo:

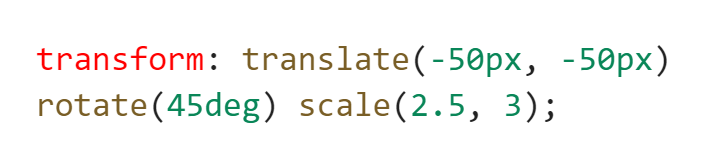


* **‘matrix()’**: Define transformações 2D personalizadas usando uma matriz 4x4. Os valores da matriz definem as transformações de translação, rotação, escala e inclinação.Por exemplo:



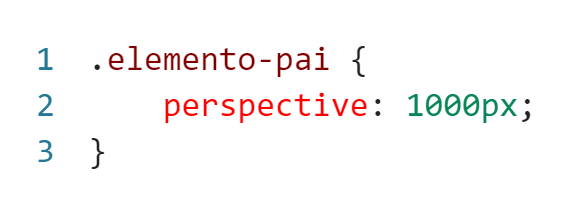
Isso irá aplicar uma transformação personalizada ao elemento.

É possível aplicar vários dos valores citados acima em apenas uma declaração. Por exemplo:



### Perspectiva

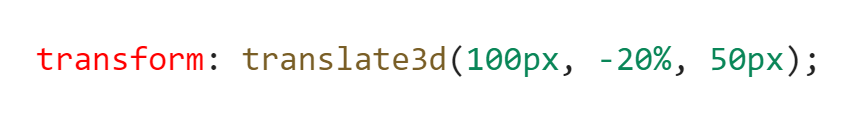
A propriedade **‘perspective’** deve ser aplicada no elemento pai e trabalha em conjunto com as transformações 3D aplicadas ao elemento filho para criar a ilusão de profundidade e tridimensionalidade em elementos.



O valor pode ser especificado em pixels ou em outras unidades de comprimento. Além disso, o valor também pode ser negativo, mas é pouco comum.

A combinação é feita com as funções usadas na propriedade **‘transform’**, aplicadas no elemento filho.

* **‘translate3d()’**: É como o ‘translate’, mas deve ser especificado também o **valor de Z**, além dos valores X e Y.

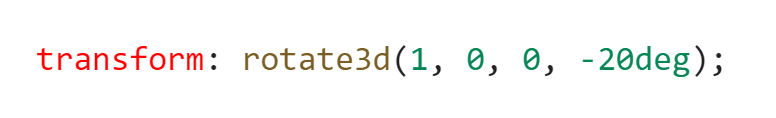


No exemplo acima, o elemento será movido 100 pixels para a direita ao longo do eixo X, 20% para cima ao longo do eixo Y e 50 pixels para frente ao longo do eixo Z.

* **‘rotate3d()’**: Permite realizar rotações em três dimensões. Devem ser especificados os seguintes valores em ordem: <eixoX>, <eixoY>, <eixoZ>, <ângulo>.

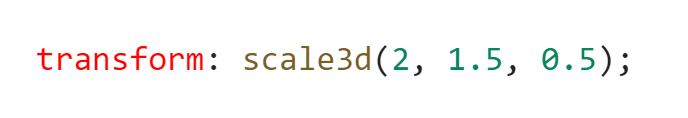
Os eixos representam os componentes do vetor de eixo em torno do qual a rotação será aplicada. Esses valores podem variar de -1 a 1 e definem a direção do eixo.

O ângulo representa o ângulo de rotação em graus.



Nesse exemplo, o elemento será rotacionado em torno do eixo X em 45 graus. O valor ‘1’ indica que o eixo de rotação está alinhado com o eixo X.

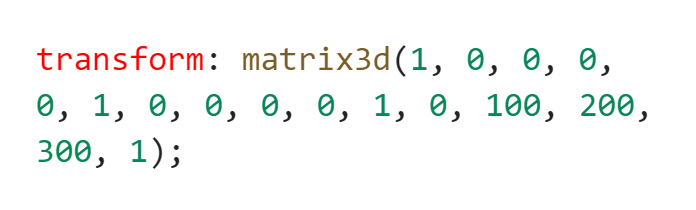
* **‘scale3d()’**: É como o scale, mas devem ser especificados o **valor de Z**, além dos valores X e Y.



No exemplo acima, o elemento será escalado para o dobro do tamanho ao longo do eixo X, 1.5 vezes o tamanho original ao longo do eixo Y e reduzido pela metado do tamanho original ao longo do eixo Z.

* **‘matrix3d()’**: Essa função permite especificar diretamente os 16 valores da matrix para obter o efeito de transformação desejado.

Uma matriz 4x4 é uma representação matemática que combina diferentes tipos de transformações, como translação, rotação, escala e cisalhamento em um único valor.

Os valores de 1 a 16 representam os elementos da matriz 4x4.

A **‘matrix3d()’** é como uma shorthand, porém muito complexa.

## *Media Queries*

As Media Queries são uma parte fundamental do design responsivo na web. Elas permitem adaptar a aparência e o layout de um site ou aplicativo de acordo com as características do dispositivo e das janelas visualização em que estão sendo exibidos.

### *Media Types*

As Media Types, ou tipos de mídia, permitem que os desenvolvedores criem um estilo diferente para cada tipo de mídia, como telas de computadores, de celulares, de TV e até para layout de impressão. As Media Types permitem que os estilos sejam adaptados para cada tipo de mídia, garantindo uma apresentação adequada e otimizada para diferentes contextos.

As Media Types mais comuns são:

* **‘all’:** É a Media Type padrão e se aplica a todos os dispositivos e mídias.
* **‘screen’:** Para todos os dispositivos de tela.
* **‘print’:** Para impressoras.
* **‘speech’:** Para dispositivos de sintetização de voz, como leitores de tela ou sistemas de reconhecimento de voz.
* **‘projection’:** Para projetores.
* **‘tv’:** Para dispositivos de televisão.

#### Aplição das Media Types

Existem duas formas de aplicar as Media Types:

##### Documento de estilo único

Nesse caso, um único documento CSS irá conter dentro de si todos os estilos de cada tipo de mídia.

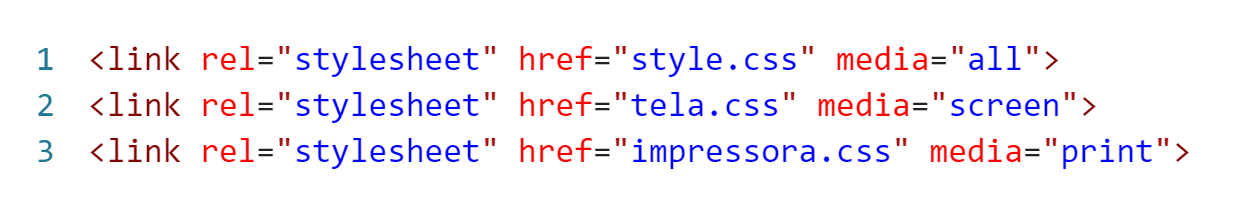
Após a regra **‘@media’**, escreva o tipo de mídia relativo a o estilo a ser trabalhado.



##### Separação por tipo de mídia

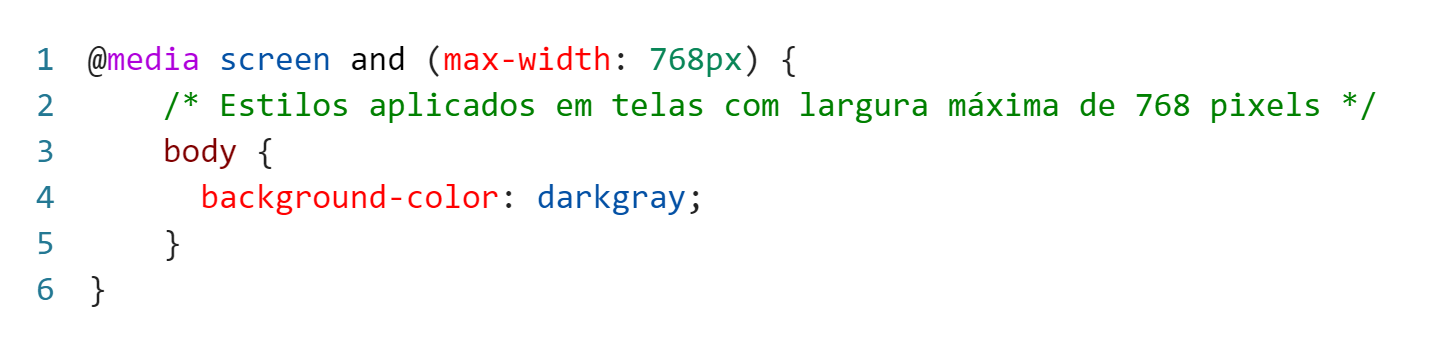
Nesse caso, será um documento para cada tipo de mídia.

Ao chamar o arquivo CSS no documento HTML, será adicionado o atributo **‘media’** na tag ‘<link>’.



### *Media Features*

As **Media Features**(características de mídia) são usadas em conjunto com as Media Types, e são utilizadas para especificar as características ou condições específicas de um dispositivo ou ambiente de visualização para criar estilos personalizados para cada situação.



No caso do exemplo acima, todos os dispositivos que rodarem a página e tiverem a tela com 768 pixels ou menos, irá possuir a cor de fundo cinza escuro em seu corpo.

As Media Features **são aplicadas entre parênteses**, e ligadas junto a Media Type pela palavra **‘and’** (‘e’, em português).

#### Tipos de *Media Features*

Existem vários tipos de Media Features, e as mais comuns são:

* **‘width’:** Especifica a largura do viewport do dispositivo.
* **‘height’:** Especifica a altura do viewport do dispositivo.
* **‘min-width’ – ‘max-width’:** Especificam a largura mínima e máxima do viewport do dispositivo.
* **‘min-height’ – ‘max-height’:** Especificam a altura mínima e máxima do viewport do dispositivo.
* **‘device-width’:** Especifica a largura real do dispositivo físico.
* **‘device-height’:** Especifica a altura real do dispositivo físico.
* **‘orientation’:** Especifica a orientação do dispositivo (retrato/portrait ou paisagem/landscape). O valor **‘portrait’** ou **‘landscape’** deve ser especificado após dois pontos depois da palavra ‘orientation’.
* **‘aspect-ratio’:** Especifica a relação entre a largura e a altura do viewport.
* **‘color’:** Especifica o número de bits de cores suportados pelo dispositivo.
* **‘resolution’:** Especifica a resolução de exibição do dispositivo em DPI (pontos por polegada).
* **‘pointer’:** Especifica o tipo de dispositivo de entrada de ponteiro disponível (mause, touchscreen, etc.)
* **‘hover’:** Especifica se o dispositivo suporta interações.
* **‘grid’:** Especifica se o dispositivo suporta recursos de layout em grade.

### *Mobile First*

***Mobile First*** é priorizar a experiência do usuário em dispositivos móveis. O conceito surge da necessidade de adaptar o conteúdo e o layout para as limitações de espaço e recursos desses dispositivos, que possuem tela menores e capacidades de processamento diferentes dos computadores.

A estratégia ***Mobile First*** envolve projetar e construir começando pela versão móvel e, em seguida, expandindo-o para dispositivos maiores.

***Typical Device Breakpoints***

Refere-se às larguras de tela específicas em que o layout de um site ou aplicativo se adapta para se ajustar a diferentes dispositivos. São fundamentais para um design responsivo de páginas web.

1. Dispositivos extra pequenos (Telefones):
   * Largura de tela: Menos de 576 pixels
2. Dispositivos pequenos (Tablets):
   * Largura de tela: 576 pixels a 767 pixels
3. Dispositivos médios (Laptops e desktops pequenos):
   * Largura de tela; 768 pixels a 991 pixels
4. Dispositivos grandes (Desktops grandes e telas de alta resolução):
   * Largura da tela: 1200 pixels ou mais

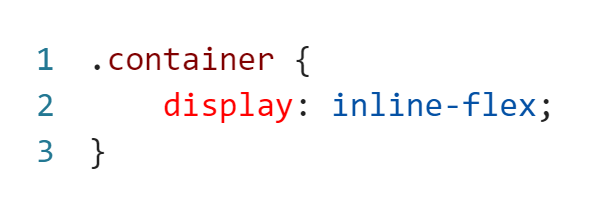
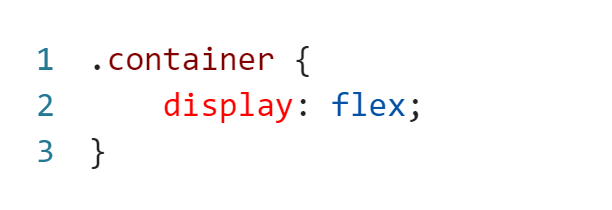
## *Flexbox*

***Flexbox*** é uma ferramenta poderosa para criar layouts flexíveis e responsivos. Foi uma novidade do CSS3 e é especialmente útil quando a necessidade é alinhar ou distribuir elemento em um container.

Tudo de Flexbox gira em torno de dois conceitos: os elementos flexíveis (**flex items**) e o container flexível (**flex** **container**). Os elementos flexíveis são filhos de um container flexível.

### ‘display: flex;’

Primeiramente é necessário definir um container flexível, para assim atribuir a ele a propriedade **‘display: flex’** ou **‘display: inline-flex’**.



### *Containers*

Um container deve possuir filhos dentro de si para que faça sentido o uso das Flexbox, por exemplo:



No exemplo acima, temos uma div que é um container, e o mesmo engloba outras três divs, que são os elementos chamados de filhos (flex items).

#### Propriedades do *Flexbox*

* **‘flex-direction’:** Define a direção dos items dentro do container.

**Valores:**

* + **‘row’:** Os flex items são dispostos em uma linha da esquerda para a direita.
  + **‘row-reversed’:** Os flex items são dispostos em uma linha da direita para a esquerda.
  + **‘column’:** Os flex items são dispostos em uma coluna de cima para baixo.
  + **‘column-reverse’:** Os flex items são dispostos em uma coluna de baixo para cima.
* **‘flex-wrap’:** Define se os flex items devem permanecer em uma única linha ou quebrar em várias linhas.

**Valores:**

* **‘nowrap’:** Os flex items são dispostos em uma única linha (padrão).
* **‘wrap’:** Os flex items são dispostos em várias linhas.
* **‘wrap-reverse’:** Os flex items são dispostos em várias linhas, em ordem reversa.
* **‘justify-content’:** Define o alinhamento horizontal dos flex items.

**Valores:**

* **‘center’:** Os flex items são centralizados horizontalmente no container.
* **‘flex-start’:** Os flex items são alinhados à esquerda do container.
* **‘flex-end’:** Os flex items são alinhados à direita do container.
* **‘space-between’:** Os flex items são distribuídos igualmente ao longo do eixo principal, com espaçamento entre eles.
* **‘space-around’:** Os flex items são distribuídos uniformemente ao longo do eixo principal, com espaçamento igual em ambos os lados.
* **‘space-evenly’:** Os flex items são distribuídos uniformemente ao longo do eixo principal, com espaçamento igual em todos os lados.
* **‘align-items:** Define o alinhamento vertical dos flex items.

**Valores:**

* **‘center’:** Os flex items são centralizados verticalmente no container.
* **‘flex-start’:** Os flex items são alinhados ao topo do container.
* **‘flex-end’:** Os flex items são alinhados à base do container.
* **‘baseline’:** Os flex items são alinhados na linha de base do container.
* **‘stretch’:** Os flex items são redimensionados para preencher a altura total do container.
* **‘align-content’:** Define o alinhamento vertical das linhas de flex items quando há espaço extra no container e as linhas não preenchem totalmente o espaço disponível.

**Valores:**

* **‘center’:** As linhas de flex items são centralizados verticalmente no container.
* **‘flex-start’:** As linhas de flex items são alinhados ao topo do container.
* **‘flex-end’:** As linhas de flex items são alinhados na base do container.
* **‘space-between’:** As linhas de flex items são distribuídos igualmente ao longo do eixo principal, com espaçamento entre elas.
* **‘space-around’:** As linhas de flex items são distribuídos uniformemente ao longo do eixo principal, com espaçamento igual em ambos os lados.
* **‘stretch’:** As linhas de flex items são redimensionadas para preencher o espaço vertical disponível.
* **‘align-self’:** Define o alinhamento vertical de um flex item individual, substituindo o valor definido pela propriedade ‘align-items’ no container. Os valores desta propriedade são os mesmos da propriedade ‘align-items’.
* **‘order’:** Define a ordem em que os flex items são exibidos dentro do container. O valor deve ser algum número inteiro para determinar a ordem.
* **‘flex-grow’:** Define a capacidade de crescimento de um flex item em relação aos outros flex items quando há espaço extra disponível. O valor deve ser algum número inteiro que representa a proporção de crescimento.
* **‘flex-shrink’:** Ao contrário do ‘flex-grow’, define a capacidade de encolhimento de um flex item em relação aos outros flex items quando o espaço é limitado. O valor deve ser um número inteiro que representa a proporção de encolhimento.
* **‘flex-basis’:** Define o tamanho inicial de um flex item antes de ser distribuído dentro do container. O valor deve ser algum número que representa o tamanho inicial em alguma unidade de medida, ou o valor **‘auto’** para usar o tamanho padrão do conteúdo.
* **‘row-gap’:** Define o espaçamento vertical entre os flex items em um layout de linha. É usado junto à propriedade ‘flex-wrap’.
* **‘column-gap’:** Define o espaçamento horizontal entre os flex items em um layout de coluna. É usado junto à propriedade ‘flex-wrap’.

#### Shorthands de *Flexbox*

1. **‘flex’:** Shorthand para **‘flex-grow’**, **‘flex-shrink’** e **‘flex-basis’**, respectivamente.
2. **‘flex-flow’:** Shorthand para **‘flex-direction’** e **‘flex-wrap’**, respectivamente.
3. **‘gap’:** Shorthand para **‘row-gap’** e **‘column-gap’**, respectivamente.
4. **‘place-content’:** Shorthand para **‘align-content’** e **‘justify-content’**, respectivamente.