



Programação Web Front-End

Aula 2 - CSS

Profa. Rosangela de Fátima Pereira Marquesone
romarquesone@utfpr.edu.br

Proposta: apresentar os conceitos referentes ao modelo de caixa do CSS e aspectos relacionados ao alinhamento dos elementos.

Objetivos: espera-se que após essa aula, você tenha habilidade para compreender os seguintes tópicos:

1. [Aprender o conceito de modelo de caixa do CSS](#)
2. [Aprender a trabalhar com a margem](#)
3. [Aprender a trabalhar com a borda](#)
4. [Aprender a trabalhar com o preenchimento \(*padding*\)](#)
5. [Aprender a usar a propriedade display](#)
6. [Aprender a utilizar a propriedade float](#)
7. [Aprender a utilizar a propriedade overflow](#)

Dicas de aprendizado:

- Execute todos os passos com atenção, compreendendo o que está sendo realizado;
- Procure não copiar código, para ter a prática de digitar o código desenvolvido;
- Pergunte quando tiver alguma dúvida;
- Mantenha um histórico dos códigos desenvolvidos, seja no github ou em algum outro meio de armazenamento (e-mails, google drive, etc.);
- Tenha curiosidade e explore os recursos apresentados.

Tópicos anteriores:

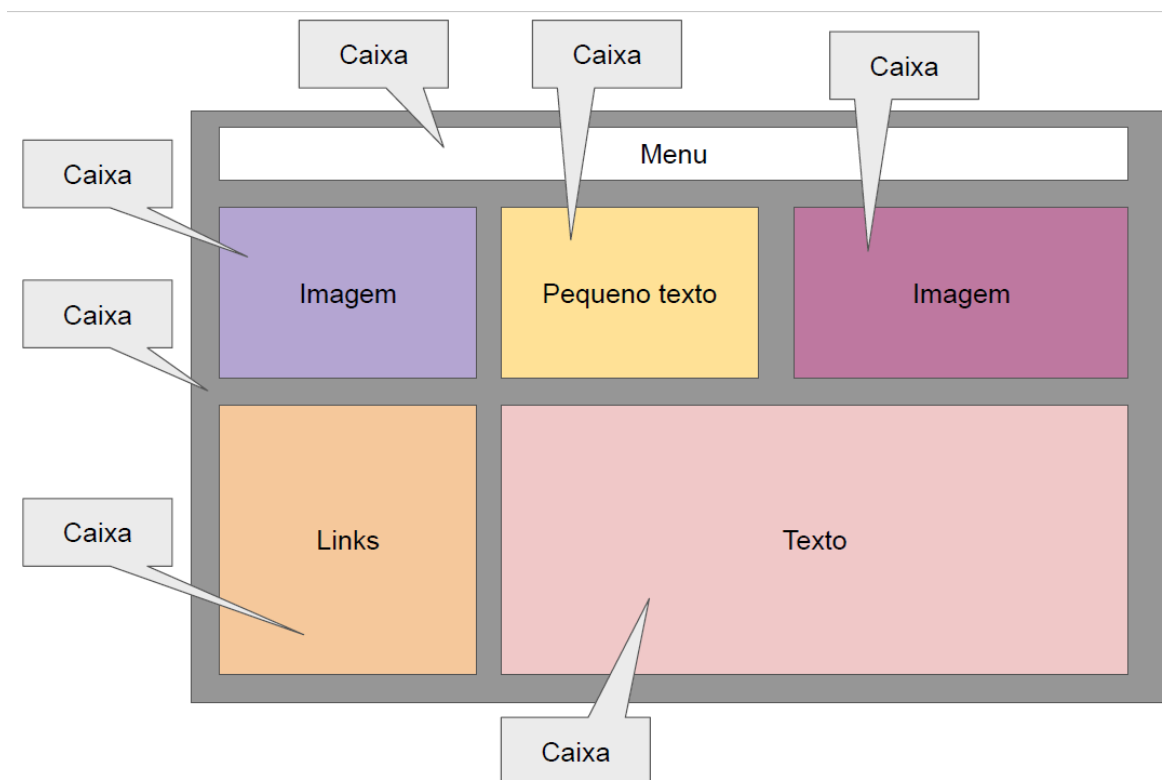
- Compreender o que é HTML
- Compreender o que são tags HTML básicas
- Criar um arquivo .html no Visual Studio (VS) Code
- Abrir o arquivo .html em um navegador
- Visualizar o código-fonte de uma página em um navegador
- Inspecionar a página em um navegador
- Utilizar o Live Server no VS Code
- Aprender a utilizar tags semânticas
- Aprender a inserir links
- Aprender a inserir listas
- Aprender a criar uma página com seu Curriculum Vitae (CV) (atividade prática)
- Aprender a inserir figuras
- Aprender a utilizar a tag semântica <figure>
- Inserir figuras em seu Curriculum Vitae (CV) (atividade prática)

- Aprender a criar formulários
- Criar um formulário (atividade prática)
- Descobrir o que é CSS
- Aprender a sintaxe do CSS
- Aprender os tipos de seletores CSS
- Aprender as formas de inclusão de CSS
- Aprender a definir cores
- Aprender a alterar as propriedades de texto

Passo 1 - Aprender o conceito de modelo de caixa do CSS

Na primeira aula sobre CSS foram apresentados conteúdos referentes ao conceito de CSS, bem como sua sintaxe e meios para lidar com as propriedades de texto e de cores em uma página Web. Além desses conceitos, outro muito importante no desenvolvimento de uma página ou um produto digital refere-se ao **modelo de caixa**.

Em uma página Web, cada elemento é identificado como sendo uma caixa retangular, conforme a figura a seguir.



Por esse motivo, em CSS, o termo "modelo de caixa" é usado quando se fala em design e layout, dado que ele nos permite adicionar uma borda ao redor dos elementos e definir o espaço entre eles. Mais especificamente, o modelo de caixa consiste em: margens, bordas, preenchimento e o conteúdo, conforme ilustrado na figura a seguir.



Nesse contexto, cada propriedade refere-se aos seguintes fatores:

- **Conteúdo** (*content box*): onde são inseridos os conteúdos (como textos, botões, formulários, imagens). Seu tamanho é definido pelas propriedades *width* e *height*.
- **Preenchimento** (*padding box*): área ao redor do conteúdo. Está situado no espaço entre o conteúdo e a borda de um elemento.
- **Borda** (*border box*): borda que contorna o preenchimento e o conteúdo. Ou seja, ela existe como um limite entre as propriedades de margem e preenchimento de uma caixa. Pode ter diferentes cores.
- **Margem** (*margin box*): área fora da borda, ou seja, do espaço exterior. Consiste em um espaço em branco entre uma caixa e outros elementos.

É importante compreender que os atributos de altura (*height*) e largura (*width*) são especificados em um modelo de caixa CSS com foco em determinar o tamanho da área de conteúdo (*content*). Dessa forma, o tamanho total de uma caixa é formado pelos seguintes valores:

- Largura: $width + padding-left + padding-right + border-left + border-right$
- Altura: $height + padding-top + padding-bottom + border-top + border-bottom$

Perceba que a margem não afeta o tamanho da caixa em si, porém, ela afeta outros conteúdos que interagem com a caixa, sendo uma parte importante do modelo de caixa CSS.

Para complementar o conhecimento sobre esse assunto, veremos como atuar com cada um desses atributos nos passos a seguir.

Passo 2 - Aprender a trabalhar com a margem

A propriedade ***margin*** define de forma abreviada a área de margem nos quatro lados de um elemento: superior, direita, inferior e esquerda. Ela é uma **área transparente**, na qual não se pode definir uma cor.

As seguintes propriedades podem ser utilizadas para sua definição, nessa ordem:

- *margin-top*
- *margin-right*
- *margin-bottom*
- *margin-left*

Além disso, veja exemplos de possíveis valores para essas propriedades:

- *auto*: o navegador calcula a margem. O navegador faz a seleção de uma margem adequada para utilizar. Por exemplo, este valor pode ser utilizado para centralizar um elemento horizontalmente.
- *length*: especifica uma margem em px, cm, etc., com um valor fixo, podendo esse valor ser negativo.
- *%*: especifica uma margem em % em relação à largura do elemento que a contém.

Veja o exemplo a seguir:

```
.box1 {  
  margin-top: 20px;  
  margin-right: 10px;  
  margin-bottom: 5px;  
  margin-left: 10px;  
}
```

Além da definição de cada um dos lados da margem (superior, esquerda, inferior, direita), é possível utilizar a propriedade ***margin*** e definir os valores para os quatro lados de uma só vez. Veja alguns exemplos a seguir:

```
/* Aplica para todos os quatro lados (sentido horário) */  
/* topo | direita | inferior | esquerda */  
margin: 10%;  
  
/* vertical | horizontal */  
margin: 5% auto;  
  
/* topo | horizontal | inferior */  
margin: 10px auto 20px;  
  
/* topo | esquerda | inferior | direita */  
margin: 2px 5px 0 auto;
```

Curiosidade

Caso você queira centralizar um elemento em CSS, uma técnica a ser utilizada é definir as margens à esquerda e à direita como auto, fazendo com que o elemento seja centralizado horizontalmente.

/* centraliza um elemento dentro de seu pai*/

margin: 0 auto;

Como alternativa, você também pode utilizar display: flex; justify-content: center;

Você pode encontrar outros meios de lidar com a margem por esse [link](#).

Veja um outro exemplo, no qual a margem é aplicada a um elemento div e a um elemento p.

| css | html |
|---|--|
| <pre>div { border: 1px solid black; margin: 25px 150px 75px 100px; background-color: lightblue; } p{ margin-bottom: inherit; }</pre> | <pre><!DOCTYPE html> <html> <head> <meta charset="utf-8" /> <title>Exemplo</title> <link rel="stylesheet" href="css/style.css" /> </head> <body> <h2>Exemplo de margem</h2> <div> <p>Laboratório de CSS - Aprendendo o uso de margem.</p> </div> </body> </html></pre> |
| Resultado | |
| <p>Exemplo de margem</p> <div><p>Laboratório de CSS - Aprendendo o uso de margem.</p></div> | |

PRATICANDO:

- Faça o download do arquivo compactado “**aula2-css.zip**”, disponível no moodle da disciplina.
- Descompacte o arquivo e abra-o no VS Code.
- Abra o arquivo “**ex-margem.html**” no VS Code, visualize o resultado via Live Server e após isso faça alterações nas propriedades para visualizar novas opções de layout.

Perceba que a margem corresponde ao espaço fora da área da borda, enquanto o *padding* (que será visto no [Passo 4](#)) está dentro da área da borda.

Perceba também que os planos de fundo das margens são sempre transparentes, enquanto o plano de fundo da área de *padding* de uma caixa é especificado pela propriedade *background* do elemento.

Passo 3 - Aprender a trabalhar com a borda

A propriedade abreviada ***border*** possibilita especificar o estilo, a largura e a cor da borda de um elemento, contendo as seguintes opções de propriedades:

- border-style
- border-width
- border-color

Propriedade ***border-style***

A propriedade *border-style* especifica que tipo de borda deve ser exibido. Veja alguns exemplos a seguir:

```
border-style: dotted; /*Define uma borda pontilhada */
border-style: dashed; /*Define uma borda tracejada */
border-style: solid; /*Define uma borda sólida */
border-style: double; /*Define uma borda dupla */
```

É importante também saber que você pode adotar um estilo de borda para cada um dos cantos de um elemento, conforme exemplos a seguir:

| css | html |
|--|--|
| <pre>.dotted { border-style: dotted; } .dashed { border-style: dashed; } .solid { border-style: solid; } .double { border-style: double; } .hidden { border-style: hidden; } .mix { border-style: dotted dashed solid double; }</pre> | <pre><!DOCTYPE html> <html> <head> <meta charset="utf-8" /> <title>Exemplo</title> <link rel="stylesheet" href="css/style.css" /> </head> <body> <h2>Exemplo de bordas</h2> <p class="dotted">Borda pontilhada.</p> <p class="dashed">Borda tracejada.</p> <p class="solid">Borda sólida.</p> <p class="double">Borda dupla.</p> <p class="hidden">Borda oculta.</p> <p class="mix">Bordas mistas.</p> </body> </html></pre> |

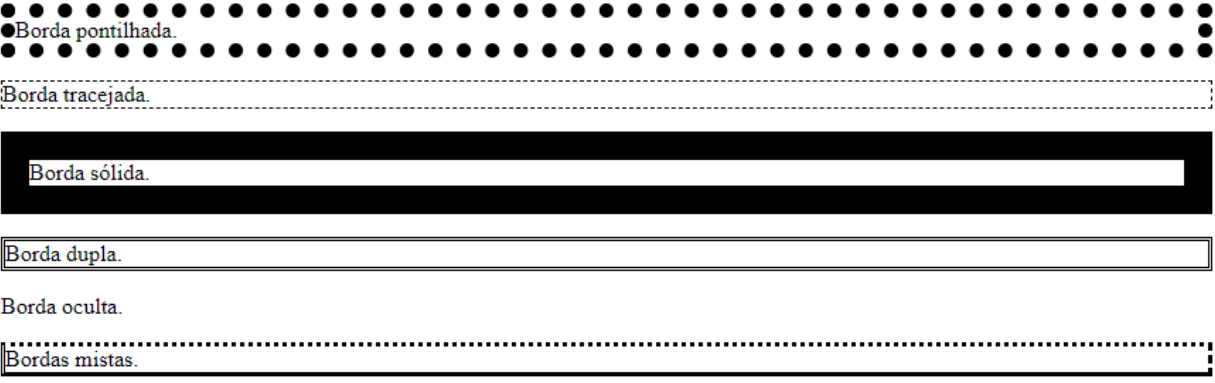
| Resultado |
|---|
| <h2>Exemplo de bordas</h2> <div>Borda pontilhada.</div> <div>Borda tracejada.</div> <div>Borda sólida.</div> <div>Borda dupla.</div> <div>Borda oculta.</div> <div>Bordas mistas.</div> |

PRATICANDO: Abra o arquivo “**ex-borda-estilo.html**” no VS Code, visualize o resultado via Live Server e após isso faça alterações nas propriedades de estilo para visualizar novas opções de layout.

Propriedade *border-width*

A propriedade *border-width* especifica a largura das quatro bordas. A maioria dos elementos HTML tem um valor de borda zero por padrão. Porém, essa largura pode ser definida como um tamanho específico (em px, cm, etc) ou usando um dos três valores: *thin*, *medium*, ou *thick*, conforme exemplo a seguir.

| css | html |
|--|--|
| <pre>.dotted { border-style: dotted; border-width: 10px; } .dashed { border-style: dashed; border-width: thin; } .solid { border-style: solid; border-width: 20px; } .double { border-style: double; border-width: medium; } .hidden { border-style: hidden; }</pre> | <pre><!DOCTYPE html> <html> <head> <meta charset="utf-8" /> <title>Exemplo</title> <link rel="stylesheet" href="css/style.css" /> </head> <body> <h2>Exemplo de bordas</h2> <p class="dotted">Borda pontilhada.</p> <p class="dashed">Borda tracejada.</p> <p class="solid">Borda sólida.</p> <p class="double">Borda dupla.</p> <p class="hidden">Borda oculta.</p> <p class="mix">Bordas mistas.</p> </body> </html></pre> |

| p.mix { border-style: dotted dashed solid double; } | |
|---|--|
| Resultado | |
| Exemplo de bordas  <p>Borda pontilhada.</p> <p>Borda tracejada.</p> <p>Borda sólida.</p> <p>Borda dupla.</p> <p>Borda oculta.</p> <p>Bordas mistas.</p> | |

PRATICANDO: Abra o arquivo “**ex-borda-largura.html**” no VS Code, visualize o resultado via Live Server e após isso faça alterações nas propriedades de estilo para visualizar novas opções de layout.

Propriedade *border-color*

A propriedade *border-color* é usada para definir a cor das quatro bordas. Pode ter de um a quatro valores (para a borda superior, borda direita, borda inferior e borda esquerda), seguindo as possibilidades de valores descritas no tutorial anterior (palavra-chave, hexadecimal, RGB, ...).

Veja um exemplo no código a seguir.

| .css | html |
|--|--|
| <pre>.dotted { border-style: dotted; border-width: 10px; border-color: red; } .dashed { border-style: dashed; border-width: thin; } .solid { border-style: solid; border-width: 20px; border-radius: 30px; border-color: red green blue yellow; } .double { border-style: double; border-width: medium;</pre> | <pre><!DOCTYPE html> <html> <head> <meta charset="utf-8" /> <title>Exemplo</title> <link rel="stylesheet" href="css/style.css" /> </head> <body> <h2>Exemplo de bordas</h2> <p class="dotted">Borda pontilhada.</p> <p class="dashed">Borda tracejada.</p> <p class="solid">Borda sólida.</p> <p class="double">Borda dupla.</p> <p class="mix">Bordas mistas.</p> </body> </html></pre> |

```
border-color: hsl(145, 89%, 26%);  
}  
  
.mix {  
border-style: dotted dashed solid double;  
border-color: #c2247e;  
}
```

Resultado

Exemplo de bordas



PRATICANDO: Abra o arquivo “**ex-borda-cor.html**” no VS Code, visualize o resultado via Live Server e após isso faça alterações nas propriedades de estilo para visualizar novas opções de layout.

É importante saber também que, para encurtar o código, também é possível especificar todas as propriedades de borda individuais em uma propriedade.

A propriedade border é uma propriedade abreviada para as seguintes propriedades de borda individuais:

- Largura da borda
- Estilo de borda (obrigatório)
- Cor da borda

Ou seja, ao invés de definir as propriedades da forma a seguir:

```
p.dotted {  
border-style: dotted;  
border-width: 10px;  
border-color: red;  
}
```

Você pode definir os valores das propriedades da seguinte forma abreviada:

```
p.dotted {  
  border: 10px dotted red;  
}
```

Por fim, é importante saber também que você pode definir a largura, cor e estilo de **cada um dos cantos da borda**, conforme exemplos a seguir:

- border-top-width
- border-top-style
- border-top-color
- border-right-width
- border-right-style
- border-right-color
- border-bottom-width
- border-bottom-style
- border-bottom-color
- border-left-width
- border-left-style
- border-left-color

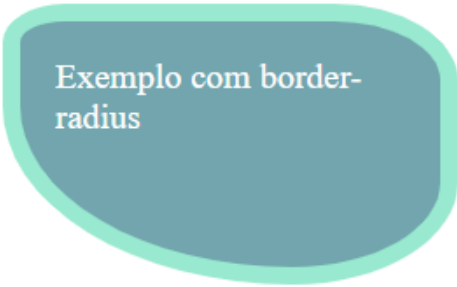
Propriedade ***border-radius***

Outra propriedade que pode ser útil para você é a border-radius. Ela é utilizada para arredondar os cantos da borda externa de um determinado elemento. Confira alguns exemplos a seguir, com as propriedades de border-radius aplicada.

Você também pode encontrar mais exemplos por esse [link](#).

| CSS | Resultado |
|--|-----------|
| <i>/* aplicado para os 4 lados */</i> border-radius: 30px; | |
| <i>/* top-left-bottom-right top-right-bottom-left */</i> border-radius: 25% 5%; | |

```
/* top-left | top-right | bottom-right | bottom-left */  
border-radius: 15% 30% 40% 70%;
```



Exemplo com border-radius

Passo 4 - Aprender a trabalhar com o preenchimento (*padding*)

A propriedade ***padding*** é usada para criar um espaço ao redor do conteúdo de um elemento, dentro de qualquer borda definida. Suas opções de propriedades são:

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

Assim como vimos com a opção border, a propriedade *padding* é uma propriedade abreviada para essas propriedades individuais (*top*, *right*, *bottom*, *left*). Quando se utiliza a propriedade abreviada *padding* e dois valores estão presentes, o primeiro valor determina o preenchimento superior e inferior e o segundo valor determina o preenchimento esquerdo e direito, conforme vimos no caso da propriedade *margin*.

Confira alguns exemplos de definição do *padding*:

```
.ex0 {  
  padding-top: 20px;  
  padding-right: 10px;  
  padding-bottom: 5px;  
  padding-left: 10px;  
}  
  
/* 4 lados */  
.ex1 {  
  padding: 10px;  
}  
  
/* vertical | horizontal */  
.ex2 {  
  padding: 20px 10px;  
}  
  
/* top | horizontal | bottom */  
.ex3 {  
  padding: 1em 10px 5px;  
}  
  
/* top | right | bottom | left */  
.ex4 {  
  padding: 20px 10% 2em 10px;  
}
```

Veja a seguir um exemplo de código **SEM** a definição do tamanho do espaço ao redor do conteúdo de um elemento.

| CSS | .html |
|---|--|
| <pre>div { border: 10px solid red; margin: 20px; background-color: lightblue; }</pre> | <pre><!DOCTYPE html> <html> <head> <meta charset="utf-8" /> <title>Exemplo</title> <link rel="stylesheet" href="css/style.css" /> </head> <body> <h2>Exemplo de padding</h2> <div>Aula de CSS - Aprendendo a usar a propriedade padding.</div> </body> </html></pre> |
| Resultado | |
| <div><h2>Exemplo de padding</h2><div>Aula de CSS - Aprendendo a usar a propriedade padding.</div></div> | |

Agora veja o resultado COM a especificação do tamanho do espaço ao redor do conteúdo de um elemento.

| .CSS | html |
|---|--|
| <pre>div { border: 10px solid red; margin: 20px; background-color: lightblue; padding: 10px 50px 50px 10px; }</pre> | <pre><!DOCTYPE html> <html> <head> <meta charset="utf-8" /> <title>Exemplo</title> <link rel="stylesheet" href="css/style.css" /> </head> <body> <h2>Exemplo de padding</h2> <div>Aula de CSS - Aprendendo a usar a propriedade padding.</div> </body> </html></pre> |

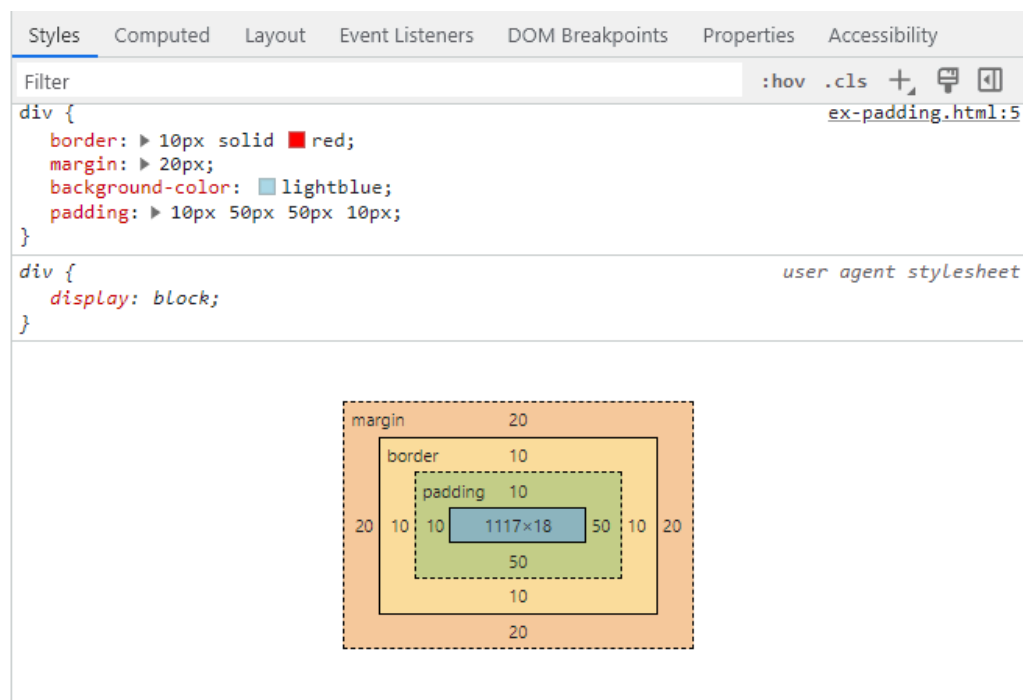
Resultado

Exemplo de padding

Aula de CSS - Aprendendo a usar a propriedade padding.

PRATICANDO: Abra o arquivo “**ex-padding.html**” no VS Code, visualize o resultado via Live Server e após isso faça alterações nas propriedades de estilo para visualizar novas opções de layout.

Caso queira verificar o tamanho de todos os valores de borda, margem e preenchimento definidos para cada elemento, você pode utilizar o recurso de inspecionar a página e acessar o recurso que apresenta os valores do modelo de caixa, conforme exemplificado na figura a seguir.



OBSERVAÇÃO

É importante que você saiba também sobre a propriedade **box-sizing**. Ela define como a largura e a altura totais de um elemento serão calculadas.

Por padrão, a largura e a altura de um elemento são aplicadas apenas à caixa de conteúdo do elemento. Por exemplo, imagine que você tenha dois elementos lado a lado. Caso ambos tenham a largura de 50%, podemos pensar que irão caber em um container, porém, se além de definirmos uma largura a um elemento, também adicionarmos valores ao padding e ao border, a largura geral do elemento aumentará. Ou seja, o *padding* e *border* são adicionados à largura total do elemento.

Portanto, se os elementos definidos com largura 50% tiverem *padding* e/ou *border* esquerda/direita, eles terão um tamanho maior que 50%.

Veja alguns valores possíveis:

- ***box-sizing: content-box***: valor padrão dos elementos. As propriedades de largura e altura incluem o conteúdo (*content*), mas **não incluem o *padding*, *border* e *margin***.
- ***box-sizing: border-box***: as propriedades de largura e altura passam a incluir os valores de *content*, *padding* e *border*, mas não incluem *margin*.

PRATICANDO: Para exemplificar, abra o arquivo “**ex-box-sizing.html**” e inclua os campos destacados em vermelho. Após isso, verifique o resultado via Live Server, comparando com a versão original.

```
<style>

#div1 {
  padding: 15px;
  border-style: solid;
  border-color: blue;
  margin: 20px;
  width: 100px;
  height: 100px;
  box-sizing: content-box;
}

#div2 {
  padding: 15px;
  border-style: solid;
  border-color: red;
  margin: 20px;
  width: 100px;
  height: 100px;
  box-sizing: border-box;
}

</style>
```

O que aconteceu? O tamanho dos elementos permaneceu igual?

Passo 5 - Aprender a usar a propriedade display

Quando falamos sobre a propriedade *display*, devemos ter em mente que o modo como um elemento é exibido afeta os elementos vizinhos em uma página. Nesse contexto, é importante compreendermos que a propriedade *display* especifica o comportamento de exibição internos e externos de um elemento.

São exemplos de valores possíveis para a propriedade *display*:

- display: inline
- display: block
- display: inline-block
- display: none
- display: flex
- display: grid

display: inline

Voltando um pouco ao conteúdo de HTML, é importante sabermos que existem dois tipos de elementos HTML; *block-level* e *inline-level*.

Por padrão, **elementos inline** ocupam apenas a largura e altura necessárias. Eles não precisam começar em uma nova linha, podendo ficar lado a lado de outros elementos. São exemplos de elementos do tipo *inline*:

-
- <a>
-

Veja um exemplo de código a seguir, com o elemento span no modo *inline*. Perceba que o texto não foi para a linha de baixo, permanecendo no mesmo local onde foi iniciado o parágrafo.

| css | html |
|---|--|
| <pre>span { background-color: yellow; }</pre> | <pre><!DOCTYPE html> <html> <head> <meta charset="utf-8" /> <title>Exemplo</title> <link rel="stylesheet" href="css/style.css" /> </head> <body> <p>Exemplo de parágrafo com elemento span.</p> </body> </html></pre> |

| Resultado |
|---|
| Exemplo de parágrafo com elemento span. |

PRATICANDO: Abra o arquivo “**ex-inline.html**” no VS Code, visualize o resultado via Live Server e após isso, faça alterações nas propriedades de estilo para visualizar novas opções de layout.

Conforme apresentado pela especificação MDN, os elementos *inline* possuem as seguintes características:

- A caixa (box) não irá quebrar em uma nova linha.
- As propriedades de largura e altura não serão aplicadas.
- No preenchimento superior e inferior, as margens e as bordas serão aplicados, mas não farão com que outras caixas embutidas se afastem da caixa.
- No preenchimento esquerdo e direito, as margens e as bordas serão aplicados e farão com que outras caixas embutidas se afastem da caixa.

display: block

Diferente dos elementos *inline*, **elementos block** forçam uma quebra de linha entre seus elementos. Ou seja, ocupam toda a largura horizontal disponível.

São exemplos de elementos do tipo *block*:

- <div>
- <h1> - <h6>
- <p>
- <form>
-
-
- <header>
- <footer>

| css | html |
|--|---|
| <pre>p { background-color: yellow; }</pre> | <pre><!DOCTYPE html> <html> <head> <meta charset="utf-8" /> <title>Exemplo</title> <link rel="stylesheet" href="css/style.css" /> </head> <body> <p>Primeiro parágrafo</p> <p>Segundo parágrafo</p></pre> |

| | </body> </html> |
|--|--------------------|
| Resultado | |
| <div>Primeiro parágrafo</div> <div>Segundo parágrafo</div> | |

PRATICANDO: Abra o arquivo “**ex-block.html**” no VS Code, visualize o resultado via Live Server e após isso faça alterações nas propriedades de estilo para visualizar novas opções de layout.

Em resumo, para elementos desse tipo, as seguintes características são então observadas:

- A caixa (box) irá quebrar em uma nova linha.
- As propriedades de largura e altura são respeitadas.
- Preenchimento, margem e borda farão com que outros elementos sejam afastados da caixa.
- Se a largura não for especificada, a caixa se estenderá na direção *inline* para preencher o espaço disponível.

Ou seja, é importante compreender que os elementos com *display:block* **consumirão todo o espaço na direção horizontal**. Como foi visto no exemplo, os parágrafos se espalham e ficam o maior possível no bloco que os contém. Caso seja especificado a largura (*width*) desses, ainda assim eles continuarão um abaixo do outro, mesmo havendo espaço para ficarem lado a lado.

display: inline-block

A grande diferença desse valor, comparado com o *display:inline*, é que esse permite definir a altura e a largura do elemento. Já, em relação ao *display:block*, a grande diferença é que o ***inline-block*** não adiciona uma quebra de linha após o elemento, podendo o elemento ficar próximo a outros elementos.

display: none

Esse valor é utilizado para ocultar um elemento, de forma que a página será exibida como se o elemento não estivesse lá. Ou seja, o documento HTML é renderizado como se o elemento não existisse.

Veja um exemplo a seguir.

| css | html |
|---------------------------------------|--|
| <pre>#oculto{ display:none; }</pre> | <pre><!DOCTYPE html> <html> <head></pre> |

| <pre>li { display: inline; }</pre> | <pre><meta charset="utf-8" /> <title>Exemplo</title> <link rel="stylesheet" href="css/style.css" /> </head> <body> <p>Exemplo de alteração da exibição de uma lista:</p> Item 1 <li id="oculto">Item 2 Item 3 </body> </html></pre> |
|--|---|
| Resultado | |
| <p>Exemplo de alteração da exibição de uma lista:</p> <p>Item 1 Item 3</p> | |

PRATICANDO: Abra o arquivo “**ex-none.html**” no VS Code, visualize o resultado via Live Server e após isso faça alterações nas propriedades de estilo para visualizar novas opções de layout.

Perceba que o navegador renderizou o documento como se o item 2 não estivesse nele. Caso o objetivo seja manter o espaço do elemento oculto, a opção a ser utilizada é a propriedade *visibility*, com valor hidden, conforme exemplo a seguir.

| .css | html |
|---|---|
| <pre>#oculto{ visibility:hidden; } li { display: inline; }</pre> | <pre><!DOCTYPE html> <html> <head> <meta charset="utf-8" /> <title>Exemplo</title> <link rel="stylesheet" href="css/style.css" /> </head> <body> <p>Exemplo de alteração da exibição de uma lista:</p> Item 1 <li id="oculto">Item 2 Item 3 </body> </html></pre> |
| Resultado | |

Exemplo de alteração da exibição de uma lista:

[Item 1](#)

[Item 3](#)

PRATICANDO: Abra o arquivo “**ex-hidden.html**” no VS Code, visualize o resultado via Live Server e após isso faça alterações nas propriedades de estilo para visualizar novas opções de layout.


Perceba que nesse caso, embora o item 2 esteja oculto, o espaço reservado para ele é mantido.

Passo 6 - Aprender a utilizar a propriedade float

A propriedade float especifica como um elemento deve flutuar, indicando como posicionar o conteúdo. Essa propriedade pode conter os seguintes valores:

- *left* - O elemento flutua à esquerda de seu container
- *right* - O elemento flutua à direita de seu container
- *none* - O elemento não flutua (será exibido apenas onde ocorre no texto) (**Valor padrão**)
- *inherit* - O elemento herda o valor de seu pai

Confira um exemplo a seguir, no qual o texto de uma imagem será posicionada à esquerda de um parágrafo.

| css | html |
|--|---|
| <pre>img { float: left; width: 80px; height: 80px; margin: 0 10px 0 0; }</pre> | <pre><!DOCTYPE html> <html> <head> <meta charset="utf-8" /> <title>Exemplo</title> <link rel="stylesheet" href="css/style.css" /> </head> <body> <h2>Exemplo de float</h2> <p> Os Objetivos de Desenvolvimento Sustentável são um apelo global à ação para acabar com a pobreza, proteger o meio ambiente e o clima e garantir que as pessoas, em todos os lugares, possam desfrutar de paz e de prosperidade. </p> </body> </html></pre> |
| Resultado | |
| <h2>Exemplo de float</h2> <div><p>Os Objetivos de Desenvolvimento Sustentável são um apelo global à ação para acabar com a pobreza, proteger o meio ambiente e o clima e garantir que as pessoas, em todos os lugares, possam desfrutar de paz e de prosperidade.</p></div> | |

PRATICANDO: Abra o arquivo “**ex-float.html**” no VS Code, visualize o resultado via Live Server e após isso faça alterações nas propriedades de estilo para visualizar novas opções de layout.

Curiosidade!

Em cenários em que a propriedade *float* é usada e queremos que o próximo elemento fique posicionado abaixo (e não à direita ou à esquerda), devemos usar a propriedade *clear*, que especifica o que deve acontecer com o elemento próximo a um elemento flutuante. Seus valores possíveis são: *none*, *left*, *right*, *both* e *inherit*.

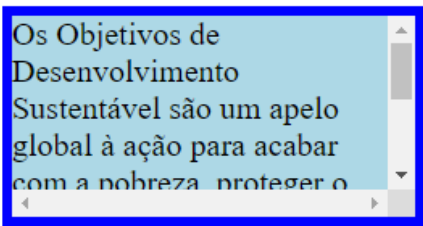
Passo 7 - Aprender a utilizar a propriedade overflow

A propriedade *overflow* controla o que acontece com o conteúdo ao se tornar grande demais para uma determinada área. Nesse contexto, as propriedades *overflow-x* e *overflow-y* especificam se o conteúdo deve ser alterado apenas horizontalmente ou verticalmente. Se apenas uma palavra-chave for especificada, *overflow-x* e *overflow-y* serão definidos com o mesmo valor.

Os seguintes valores podem ser utilizados nesta propriedade:

- *visible*: Propriedade padrão. O conteúdo é renderizado fora da caixa do elemento.
- *hidden*: O restante do conteúdo fica invisível.
- *scroll*: Uma barra de rolagem é adicionada para ver o restante do conteúdo, independente se algum conteúdo está transbordando ou não um elemento.
- *auto*: Semelhante ao *scroll*, mas adiciona barras de rolagem somente quando necessário.

Veja um exemplo a seguir no qual é apresentada uma barra de rolagem no caso em que o conteúdo excede o tamanho de uma área.

| css | html |
|--|--|
| <pre>div { background-color: lightblue; width: 200px; height: 100px; border: 5px solid blue; overflow: scroll; }</pre> | <pre><!DOCTYPE html> <html> <head> <meta charset="utf-8" /> <title>Exemplo</title> <link rel="stylesheet" href="css/style.css" /> </head> <body> <h2>Exemplo de overflow</h2> <div>Os Objetivos de Desenvolvimento Sustentável são um apelo global à ação para acabar com a pobreza, proteger o meio ambiente e o clima e garantir que as pessoas, em todos os lugares, possam desfrutar de paz e de prosperidade.</div> </body> </html></pre> |
| Resultado | |
| <div><h3>Exemplo de overflow</h3></div> | |

PRATICANDO: Abra o arquivo “**ex-overflow.html**” no VS Code, visualize o resultado via Live Server e após isso faça alterações nas propriedades de estilo para visualizar novas opções de layout.

Considerações finais

Caso tenha chegado até aqui, você conseguiu completar o conteúdo do segundo tutorial sobre CSS. A partir desses recursos, você passa a ser capaz de tornar suas páginas mais bonitas e agradáveis. Mas há muito mais para ser aprendido sobre CSS ainda. Continue explorando as opções disponíveis, utilizando sua criatividade.

Bom estudo!