U Positivo - UTFPR - PUCPr – 03/05/2024 - 17:27:26.0
Prof Dr P Kantek (pkantek@gmail.com)
Tutoriais HTML/CSS/JS VIVXh52a, V: 1.05
76001 CARLOS EDUARDO ANTAO 24FRO106 - 1 entregar ate 23/maio/24 _____ / _____ /

CSS 2

Para aplicar estilos a links, pode-se usar qualquer propriedade CSS (color, font-family, background etc). Além disso, os links podem ser estilizados dependendo do estado em que estão. Existem 4 estados

a:link	link ainda não visitado
a:visited	link já visitado
a:hover	quando o mouse passa sobre o link
a:active	link no momento em que é clicado

Veja no exemplo:

```
a:link { color: #FF0000; }
a:visited { color: #00FF00; }
a:hover { color: #FF00FF; }
a:active { color: #0000FF; }
```

Exercício 1

Aproveite os arquivos brasil.html e brasil.css e garanta que o html tenha pelo menos 5 links. Aplique no arquivo brasil.css os links acima descritos com as cores aqua, violet, sienna e chartreuse pela ordem. Acrescente nos 4 o amarelo como cor de fundo (`background-color:yellow`) Veja o funcionamento disso tudo num browser.

Para aplicar estilos a listas, pode-se usar a propriedade `list-style-type:` que pode ser circle, square, upper-roman, lower-alpha. Pode-se usar também `list-style-image` que indica qual imagem usar como o marcador de itens em listas não ordenadas. Eis o formato:

```
ul { list-style-image: url('sqpurple.gif');}
```

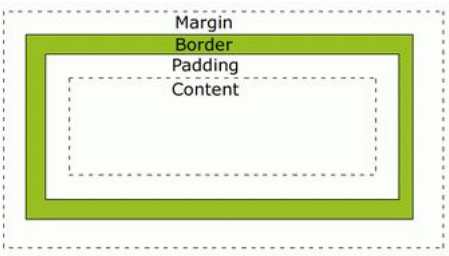
Para as tabelas, as propriedades são: `border:`, `border-collapse:`, `width:`, `height:`, `text-align:`, `vertical-align:`, `padding:`, `color:` e `background-color:`. Veja um exemplo de algumas dessas opções:

```
<style>
table, td, th { border: 1px solid green; }
th { background-color: green; color: white;}
</style>
...
<table> <tr> <th>Nome</th><th>Cidade</th>
<th>Contribuição</th> </tr>
<tr> <td>Pedro</td> <td>Curitiba</td>
<td>100</td>   </tr>
<tr> <td>Maria</td> <td>Joinville</td>
<td>300</td>   </tr>  </table>
```

Exercício 2

Defina arquivos tabela.html e tabela.css e escreva o exemplo acima. Sobre ele, aplique pelo menos 3 propriedades de tabelas diferentes e veja em cada caso o resultado em um browser.

Agora vamos estudar o modelo caixa do CSS (*CSS box model*). Todos os elementos HTML podem ser considerados como uma caixa. Este conceito é muito importante quando se pensa em layout. Pense no modelo da caixa como uma cebola quadrada composta de 4 camadas. De fora para dentro tem-se: margin, border, padding e finalmente o conteúdo que é o elemento HTML. Veja o desenho a seguir para entender o conceito



Eis um exemplo disso tudo junto

```
div { width: 300px;  padding: 25px;
   border: 25px solid navy; margin: 25px;}
```

Note que quando você define largura e altura de alguma coisa, essas medidas se referem à área de conteúdo. Para saber o tamanho total do elementos as outras medidas (se existentes) devem ser somadas. Por exemplo, o elemento abaixo ocupa 350px:

```
div { width: 320px;  padding: 10px;
    border: 5px solid gray; margin: 0;}
```

As propriedades da borda são: `border-style:` com os valores none, dotted, dashed, solid, double, groove, ridge, inset e outset. Todas as demais propriedades só atuam se a propriedade border-style for assinalada.

Depois vem `border-width:`, `border-color:` etc. Para definir a borda de maneira abreviada, a sequência é `border-width:`, `border-style:` (obrigatória), `border-color:`.

A seguir existem diversas propriedades para tratar o outline, elas são muito parecidas com as propriedades da borda e ficam como dever de casa para serem pesquisadas.

A margem clareia completamente o entorno de um elemento (fora da borda). A margem não tem cor de background e sempre é completamente transparente. As margens superior, direita, inferior e esquerda podem ser estabelecidas separadamente, ou pode-se usar a maneira abreviada e fazer tudo junto. Os valores possíveis de `margin:` são: auto (o browser calcula), tamanho (medida em px, pt, cm, ... - default é 0px), % medida em percentagem do tamanho do elemento contido e inherit quando a margem é herdada do elemento pai. Note que em tamanho pode-se usar valores negativos para sobrepujar algum conteúdo. As margens podem ser especificadas como `margin-top:`, `margin-bottom:`,`margin-right:` e `margin-left:` ou se preferir escrever apenas `margin:` e depois 4, 3 2 ou 1 valor de tamanho. A regra é: 3 tamanhos → top,laterais,bottom; 2 tamanhos → cima-baixo, laterais, e 1 tamanho → os 4 iguais.

O padding limpa o entorno do elemento (dentro da borda). Ele é afetado pela cor de fundo do elemento. De maneira muito parecida à margem, ele pode ser estabelecido com 4 propriedades ou de maneira abreviada com uma única propriedade.

Os elementos HTML também podem ter suas dimensões estabelecidas pelas propriedades: `height:`, `max-height:`, `min-height:`, `width:`, `max-width:` e `min-width:`. Veja um exemplo:

```
p { max-width: 20%;
    background-color: yellow;  }
```

A propriedade `display:` determina se e como um elemento é mostrado e a propriedade `visibility` especifica se um elemento deve ser visível ou escondido. Um elemento pode ser apagado fazendo `display: none` ou `visibility:hidden`. Este último apaga o elemento mas preserva o espaço ocupado por ele (em branco). Ou seja o layout permanece inalterado. Já `display: none` apaga o elemento e consome com o espaço ocupado por ele. O lay-out é refeito.

Exercício 3

Escreva um html chamado image.html e um css chamado image.css. Fale sobre o litoral paranaense. Apresente 3 fotos de mesma dimensão. Use 3 classes diferentes. No CSS alterne as propriedades vistas acima. Veja o resultado no browser.

Relembrando elementos em bloco (como `<h1>`, `<p>`, `<li>` e `<div>` por exemplo) são aqueles que forçam quebra de linha antes e depois dele. Já os elementos inline não quebram linha nem antes nem depois. Para transformar um elemento inline em bloco e vice-versa, usa-se a propriedade `display:`. Para inline o valor é `inline` e para bloco o valor é `block`. Veja no exemplo como transformar itens de lista em elementos inline (todos os itens serão mostrados na mesma linha).

```
li { display: inline; }
```

As propriedades de posicionamento permitem posicionar um elemento. Também permitem por um na frente do outro e especificam o que acontece quando o conteúdo de um elemento é muito grande. A primeira propriedade é `position:`. Ela tem 4 valores:

static é o valor default. O elemento é posicionado de acordo com o fluxo normal de preenchimento do browser. O elemento não é afetado pelas propriedades `top`, `bottom`, `left` e `right`.

fixed posição fixa em relação à janela do browser. Ele não é movido mesmo que a janela seja *scrolled*. Veja um exemplo
`p.fixo { position: fixed; top: 30px; right: 5px;}`
Neste caso o elemento é removido do fluxo normal. Os outros elementos são posicionados como se o elemento fixo não existisse.

relative modifica a posição em relação à sua posição normal. Aqui por exemplo, os cabeçalhos h2, serão trazidos para fora da janela:
`h2 { position: relative; left: -20px;}`

absolute posiciona relativo ao primeiro elemento pai que tenha outra posição que não static. Se não existir, será o `<html>`.

Quando elementos são posicionados fora do fluxo normal eles podem sobrepujar outros elementos. A propriedade `z-index:` indica qual a ordem de apresentação. (podem ser atribuídos valores positivos ou negativos). A regra é que um número maior sempre vai aparecer na frente de um número menor. Veja o exemplo:

```
<style>
img { position: absolute; left: 0px;
    top: 0px; z-index: -2; }
</style> </head> <body>
<h1>Um cabeçalho</h1>
<img src="qualquer.gif" width="100" height="140">
</body>
```

Exercício 4

Copie o exemplo acima, usando uma figura qualquer. Varie a propriedade `z-index` e veja o que acontece.

☞ Para você fazer

1. Imprima os arquivos pedidos (brasil htlm e css, tabela htlm e css e image html e css) grampeie-os nesta folha e devolva tudo ou então gere-os no micro na sala de aula e mostre-os ao professor, durante a aula.

2. Descreva o que faz o código a seguir

```
<!DOCTYPE html> <html> <head> <style>
p {background-color: yellow; }
p.padding { padding-top: 25px; padding-right: 50px;
  padding-bottom: 25px; padding-left: 50px; }
</style> </head> <body>
<p>This is a paragraph with no specified padding.</p>
<p class="padding">This is a paragraph with
specified paddings.</p> </body> </html>
```

3. Descreva o que faz o código a seguir

```
<!DOCTYPE html> <html> <head> <style>
table, td, th {border: 1px solid black; }
td {height: 50px;  vertical-align: bottom; }
</style> </head> <body> <table>  <tr>
<th>Firstname</th> <th>Lastname</th> <th>Savings</th>
</tr> <tr> <td>Peter</td> <td>Griffin</td> <td>$100</td>
</tr> <tr> <td>Lois</td> <td>Griffin</td> <td>$150</td>
</tr> </table> </body> </html>
```

.

106-76001 - ga/ a

U Positivo - UTFPR - PUCPr – 03/05/2024 - 17:27:26.0
Prof Dr P Kantek (pkantek@gmail.com)
Tutoriais HTML/CSS/JS VIVXh52a, V: 1.05
76199 DANILO DEPETRIS SOARES 24FRO106 - 2 entregar ate 23/maio/24 _____ / _____ /

CSS 2

Para aplicar estilos a links, pode-se usar qualquer propriedade CSS (color, font-family, background etc). Além disso, os links podem ser estilizados dependendo do estado em que estão. Existem 4 estados

a:link	link ainda não visitado
a:visited	link já visitado
a:hover	quando o mouse passa sobre o link
a:active	link no momento em que é clicado

Veja no exemplo:

```
a:link { color: #FF0000; }
a:visited { color: #00FF00; }
a:hover { color: #FF00FF; }
a:active { color: #0000FF; }
```

Exercício 1

Aproveite os arquivos brasil.html e brasil.css e garanta que o html tenha pelo menos 5 links. Aplique no arquivo brasil.css os links acima descritos com as cores aqua, violet, sienna e chartreuse pela ordem. Acrescente nos 4 o amarelo como cor de fundo (`background-color:yellow`) Veja o funcionamento disso tudo num browser.

Para aplicar estilos a listas, pode-se usar a propriedade `list-style-type:` que pode ser circle, square, upper-roman, lower-alpha. Pode-se usar também `list-style-image` que indica qual imagem usar como o marcador de itens em listas não ordenadas. Eis o formato:

```
ul { list-style-image: url('sqpurple.gif');}
```

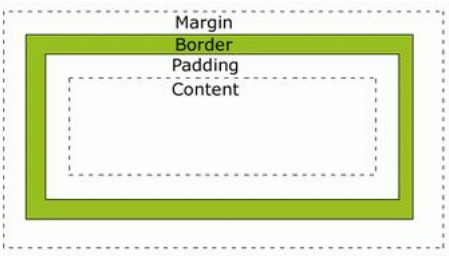
Para as tabelas, as propriedades são: `border:`, `border-collapse:`, `width:`, `height:`, `text-align:`, `vertical-align:`, `padding:`, `color:` e `background-color:`. Veja um exemplo de algumas dessas opções:

```
<style>
table, td, th { border: 1px solid green; }
th { background-color: green; color: white;}
</style>
...
<table> <tr> <th>Nome</th><th>Cidade</th>
<th>Contribuição</th> </tr>
<tr> <td>Pedro</td> <td>Curitiba</td>
<td>100</td>   </tr>
<tr> <td>Maria</td> <td>Joinville</td>
<td>300</td>   </tr>  </table>
```

Exercício 2

Defina arquivos tabela.html e tabela.css e escreva o exemplo acima. Sobre ele, aplique pelo menos 3 propriedades de tabelas diferentes e veja em cada caso o resultado em um browser.

Agora vamos estudar o modelo caixa do CSS (*CSS box model*). Todos os elementos HTML podem ser considerados como uma caixa. Este conceito é muito importante quando se pensa em layout. Pense no modelo da caixa como uma cebola quadrada composta de 4 camadas. De fora para dentro tem-se: margin, border, padding e finalmente o conteúdo que é o elemento HTML. Veja o desenho a seguir para entender o conceito



Eis um exemplo disso tudo junto

```
div { width: 300px;  padding: 25px;
   border: 25px solid navy; margin: 25px;}
```

Note que quando você define largura e altura de alguma coisa, essas medidas se referem à área de conteúdo. Para saber o tamanho total do elementos as outras medidas (se existentes) devem ser somadas. Por exemplo, o elemento abaixo ocupa 350px:

```
div { width: 320px;  padding: 10px;
    border: 5px solid gray; margin: 0;}
```

As propriedades da borda são: `border-style:` com os valores none, dotted, dashed, solid, double, groove, ridge, inset e outset. Todas as demais propriedades só atuam se a propriedade border-style for assinalada.

Depois vem `border-width:`, `border-color:` etc. Para definir a borda de maneira abreviada, a sequência é `border-width:`, `border-style:` (obrigatória), `border-color:`.

A seguir existem diversas propriedades para tratar o outline, elas são muito parecidas com as propriedades da borda e ficam como dever de casa para serem pesquisadas.

A margem clareia completamente o entorno de um elemento (fora da borda). A margem não tem cor de background e sempre é completamente transparente. As margens superior, direita, inferior e esquerda podem ser estabelecidas separadamente, ou pode-se usar a maneira abreviada e fazer tudo junto. Os valores possíveis de `margin:` são: auto (o browser calcula), tamanho (medida em px, pt, cm, ... - default é 0px), % medida em percentagem do tamanho do elemento contido e inherit quando a margem é herdada do elemento pai. Note que em tamanho pode-se usar valores negativos para sobrepujar algum conteúdo. As margens podem ser especificadas como `margin-top:`, `margin-bottom:`,`margin-right:` e `margin-left:` ou se preferir escrever apenas `margin:` e depois 4, 3 2 ou 1 valor de tamanho. A regra é: 3 tamanhos → top,laterais,bottom; 2 tamanhos → cima-baixo, laterais, e 1 tamanho → os 4 iguais.

O padding limpa o entorno do elemento (dentro da borda). Ele é afetado pela cor de fundo do elemento. De maneira muito parecida à margem, ele pode ser estabelecido com 4 propriedades ou de maneira abreviada com uma única propriedade.

Os elementos HTML também podem ter suas dimensões estabelecidas pelas propriedades: `height:`, `max-height:`, `min-height:`, `width:`, `max-width:` e `min-width:`. Veja um exemplo:

```
p { max-width: 20%;
    background-color: yellow;  }
```

A propriedade `display:` determina se e como um elemento é mostrado e a propriedade `visibility` especifica se um elemento deve ser visível ou escondido. Um elemento pode ser apagado fazendo `display: none` ou `visibility:hidden`. Este último apaga o elemento mas preserva o espaço ocupado por ele (em branco). Ou seja o layout permanece inalterado. Já `display: none` apaga o elemento e consome com o espaço ocupado por ele. O lay-out é refeito.

Exercício 3

Escreva um html chamado image.html e um css chamado image.css. Fale sobre o litoral paranaense. Apresente 3 fotos de mesma dimensão. Use 3 classes diferentes. No CSS alterne as propriedades vistas acima. Veja o resultado no browser.

Relembrando elementos em bloco (como `<h1>`, `<p>`, `<li>` e `<div>` por exemplo) são aqueles que forçam quebra de linha antes e depois dele. Já os elementos inline não quebram linha nem antes nem depois. Para transformar um elemento inline em bloco e vice-versa, usa-se a propriedade `display:`. Para inline o valor é `inline` e para bloco o valor é `block`. Veja no exemplo como transformar itens de lista em elementos inline (todos os itens serão mostrados na mesma linha).

```
li { display: inline; }
```

As propriedades de posicionamento permitem posicionar um elemento. Também permitem por um na frente do outro e especificam o que acontece quando o conteúdo de um elemento é muito grande. A primeira propriedade é `position:`. Ela tem 4 valores:

static é o valor default. O elemento é posicionado de acordo com o fluxo normal de preenchimento do browser. O elemento não é afetado pelas propriedades `top`, `bottom`, `left` e `right`.

fixed posição fixa em relação à janela do browser. Ele não é movido mesmo que a janela seja *scrolled*. Veja um exemplo
`p.fixo { position: fixed; top: 30px; right: 5px;}`
Neste caso o elemento é removido do fluxo normal. Os outros elementos são posicionados como se o elemento fixo não existisse.

relative modifica a posição em relação à sua posição normal. Aqui por exemplo, os cabeçalhos h2, serão trazidos para fora da janela:
`h2 { position: relative; left: -20px;}`

absolute posiciona relativo ao primeiro elemento pai que tenha outra posição que não static. Se não existir, será o `<html>`.

Quando elementos são posicionados fora do fluxo normal eles podem sobrepujar outros elementos. A propriedade `z-index:` indica qual a ordem de apresentação. (podem ser atribuídos valores positivos ou negativos). A regra é que um número maior sempre vai aparecer na frente de um número menor. Veja o exemplo:

```
<style>
img { position: absolute; left: 0px;
    top: 0px; z-index: -2; }
</style> </head> <body>
<h1>Um cabeçalho</h1>
<img src="qualquer.gif" width="100" height="140">
</body>
```

Exercício 4

Copie o exemplo acima, usando uma figura qualquer. Varie a propriedade `z-index` e veja o que acontece.

☞ Para você fazer

1. Imprima os arquivos pedidos (brasil htlm e css, tabela htlm e css e image html e css) grampeie-os nesta folha e devolva tudo ou então gere-os no micro na sala de aula e mostre-os ao professor, durante a aula.

2. Descreva o que faz o código a seguir

```
<!DOCTYPE html> <html> <head> <style>
a:link, a:visited {     display: block;
 font-weight: bold;     color: #ffffff;
 background-color: #98bf21;      width: 120px;
 text-align: center;      padding: 4px;
 text-decoration: none;  }
a:hover, a:active {    background-color: #7A991A;  }
</style> </head> <body>
<a href="default.asp" target="_blank">This is a link</a>
</body> </html>
```

3. Descreva o que faz o código a seguir

```
<!DOCTYPE html> <html> <head> <style>
a.three:link {color:#ff0000;}
a.three:visited {color:#0000ff;}
a.three:hover {background:#66ff66;}
</style> </head> <body>
<p>Mouse over the links and watch them change layout:</p>
<p><b><a class="three" href="default.asp"
target="_blank">This link changes background-color</a></b></p>
</body> </html>
```

.

106-76199 - ga/ a

U Positivo - UTFPR - PUCPr – 03/05/2024 - 17:27:26.0
Prof Dr P Kantek (pkantek@gmail.com)
Tutoriais HTML/CSS/JS VIVXh52a, V: 1.05
76018 EDUARDO CAVALHEIRO DIAS
24FRO106 - 3 entregar ate 23/maio/24 _____ / / _____

CSS 2

Para aplicar estilos a links, pode-se usar qualquer propriedade CSS (color, font-family, background etc). Além disso, os links podem ser estilizados dependendo do estado em que estão. Existem 4 estados

a:link	link ainda não visitado
a:visited	link já visitado
a:hover	quando o mouse passa sobre o link
a:active	link no momento em que é clicado

Veja no exemplo:

```
a:link { color: #FF0000; }
a:visited { color: #00FF00; }
a:hover { color: #FF00FF; }
a:active { color: #0000FF; }
```

Exercício 1

Aproveite os arquivos brasil.html e brasil.css e garanta que o html tenha pelo menos 5 links. Aplique no arquivo brasil.css os links acima descritos com as cores aqua, violet, sienna e chartreuse pela ordem. Acrescente nos 4 o amarelo como cor de fundo (`background-color:yellow`) Veja o funcionamento disso tudo num browser.

Para aplicar estilos a listas, pode-se usar a propriedade `list-style-type:` que pode ser circle, square, upper-roman, lower-alpha. Pode-se usar também `list-style-image` que indica qual imagem usar como o marcador de itens em listas não ordenadas. Eis o formato:

```
ul { list-style-image: url('sqpurple.gif');}
```

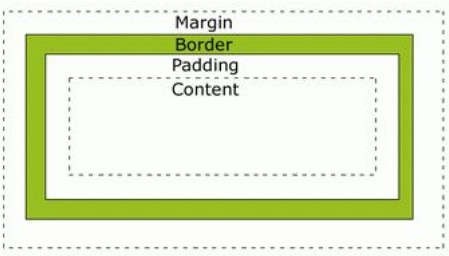
Para as tabelas, as propriedades são: `border:`, `border-collapse:`, `width:`, `height:`, `text-align:`, `vertical-align:`, `padding:`, `color:` e `background-color:`. Veja um exemplo de algumas dessas opções:

```
<style>
table, td, th { border: 1px solid green; }
th { background-color: green; color: white;}
</style>
...
<table> <tr> <th>Nome</th><th>Cidade</th>
<th>Contribuição</th> </tr>
<tr> <td>Pedro</td> <td>Curitiba</td>
<td>100</td>   </tr>
<tr> <td>Maria</td> <td>Joinville</td>
<td>300</td>   </tr>  </table>
```

Exercício 2

Defina arquivos tabela.html e tabela.css e escreva o exemplo acima. Sobre ele, aplique pelo menos 3 propriedades de tabelas diferentes e veja em cada caso o resultado em um browser.

Agora vamos estudar o modelo caixa do CSS (*CSS box model*). Todos os elementos HTML podem ser considerados como uma caixa. Este conceito é muito importante quando se pensa em layout. Pense no modelo da caixa como uma cebola quadrada composta de 4 camadas. De fora para dentro tem-se: margin, border, padding e finalmente o conteúdo que é o elemento HTML. Veja o desenho a seguir para entender o conceito



Eis um exemplo disso tudo junto

```
div { width: 300px;  padding: 25px;
   border: 25px solid navy; margin: 25px;}
```

Note que quando você define largura e altura de alguma coisa, essas medidas se referem à área de conteúdo. Para saber o tamanho total do elementos as outras medidas (se existentes) devem ser somadas. Por exemplo, o elemento abaixo ocupa 350px:

```
div { width: 320px;  padding: 10px;
    border: 5px solid gray; margin: 0;}
```

As propriedades da borda são: `border-style:` com os valores none, dotted, dashed, solid, double, groove, ridge, inset e outset. Todas as demais propriedades só atuam se a propriedade border-style for assinalada.

Depois vem `border-width:`, `border-color:` etc. Para definir a borda de maneira abreviada, a sequência é `border-width:`, `border-style:` (obrigatória), `border-color:`.

A seguir existem diversas propriedades para tratar o outline, elas são muito parecidas com as propriedades da borda e ficam como dever de casa para serem pesquisadas.

A margem clareia completamente o entorno de um elemento (fora da borda). A margem não tem cor de background e sempre é completamente transparente. As margens superior, direita, inferior e esquerda podem ser estabelecidas separadamente, ou pode-se usar a maneira abreviada e fazer tudo junto. Os valores possíveis de `margin:` são: auto (o browser calcula), tamanho (medida em px, pt, cm, ... - default é 0px), % medida em percentagem do tamanho do elemento contido e inherit quando a margem é herdada do elemento pai. Note que em tamanho pode-se usar valores negativos para sobrepujar algum conteúdo. As margens podem ser especificadas como `margin-top:`, `margin-bottom:`,`margin-right:` e `margin-left:` ou se preferir escrever apenas `margin:` e depois 4, 3 2 ou 1 valor de tamanho. A regra é: 3 tamanhos → top,laterais,bottom; 2 tamanhos → cima-baixo, laterais, e 1 tamanho → os 4 iguais.

O padding limpa o entorno do elemento (dentro da borda). Ele é afetado pela cor de fundo do elemento. De maneira muito parecida à margem, ele pode ser estabelecido com 4 propriedades ou de maneira abreviada com uma única propriedade.

Os elementos HTML também podem ter suas dimensões estabelecidas pelas propriedades: `height:`, `max-height:`, `min-height:`, `width:`, `max-width:` e `min-width:`. Veja um exemplo:

```
p { max-width: 20%;
    background-color: yellow;  }
```

A propriedade `display:` determina se e como um elemento é mostrado e a propriedade `visibility` especifica se um elemento deve ser visível ou escondido. Um elemento pode ser apagado fazendo `display: none` ou `visibility:hidden`. Este último apaga o elemento mas preserva o espaço ocupado por ele (em branco). Ou seja o layout permanece inalterado. Já `display: none` apaga o elemento e consome com o espaço ocupado por ele. O lay-out é refeito.

Exercício 3

Escreva um html chamado image.html e um css chamado image.css. Fale sobre o litoral paranaense. Apresente 3 fotos de mesma dimensão. Use 3 classes diferentes. No CSS alterne as propriedades vistas acima. Veja o resultado no browser.

Relembrando elementos em bloco (como `<h1>`, `<p>`, `<li>` e `<div>` por exemplo) são aqueles que forçam quebra de linha antes e depois dele. Já os elementos inline não quebram linha nem antes nem depois. Para transformar um elemento inline em bloco e vice-versa, usa-se a propriedade `display:`. Para inline o valor é `inline` e para bloco o valor é `block`. Veja no exemplo como transformar itens de lista em elementos inline (todos os itens serão mostrados na mesma linha).

```
li { display: inline; }
```

As propriedades de posicionamento permitem posicionar um elemento. Também permitem por um na frente do outro e especificam o que acontece quando o conteúdo de um elemento é muito grande. A primeira propriedade é `position:`. Ela tem 4 valores:

static é o valor default. O elemento é posicionado de acordo com o fluxo normal de preenchimento do browser. O elemento não é afetado pelas propriedades `top`, `bottom`, `left` e `right`.

fixed posição fixa em relação à janela do browser. Ele não é movido mesmo que a janela seja *scrolled*. Veja um exemplo
`p.fixo { position: fixed; top: 30px; right: 5px;}`
Neste caso o elemento é removido do fluxo normal. Os outros elementos são posicionados como se o elemento fixo não existisse.

relative modifica a posição em relação à sua posição normal. Aqui por exemplo, os cabeçalhos h2, serão trazidos para fora da janela:
`h2 { position: relative; left: -20px;}`

absolute posiciona relativo ao primeiro elemento pai que tenha outra posição que não static. Se não existir, será o `<html>`.

Quando elementos são posicionados fora do fluxo normal eles podem sobrepujar outros elementos. A propriedade `z-index:` indica qual a ordem de apresentação. (podem ser atribuídos valores positivos ou negativos). A regra é que um número maior sempre vai aparecer na frente de um número menor. Veja o exemplo:

```
<style>
img { position: absolute; left: 0px;
    top: 0px; z-index: -2; }
</style> </head> <body>
<h1>Um cabeçalho</h1>
<img src="qualquer.gif" width="100" height="140">
</body>
```

Exercício 4

Copie o exemplo acima, usando uma figura qualquer. Varie a propriedade `z-index` e veja o que acontece.

☞ Para você fazer

1. Imprima os arquivos pedidos (brasil htlm e css, tabela htlm e css e image html e css) grampeie-os nesta folha e devolva tudo ou então gere-os no micro na sala de aula e mostre-os ao professor, durante a aula.

2. Descreva o que faz o código a seguir

```
<!DOCTYPE html> <html> <head> <style>
p {background-color: yellow;}
p.padding { padding: 25px 50px; }
</style> </head> <body>
<p>This is a paragraph with no specified padding.</p>
<p class="padding">This with specified paddings.</p>
</body> </html>
```

3. Descreva o que faz o código a seguir

```
<!DOCTYPE html> <html> <head> <style>
a.four:link {color:#ff0000;}
a.four:visited {color:#0000ff;}
a.four:hover {font-family:monospace;}
</style>  </head> <body>
<p>Mouse over the links and watch them change layout:</p>
<p><b><a class="four" href="default.asp"
target="_blank">This link changes font-family</a></b></p>
</body> </html>
```

.

106-76018 - ga/ a

U Positivo - UTFPR - PUCPr – 03/05/2024 - 17:27:26.0
Prof Dr P Kantek (pkantek@gmail.com)
Tutoriais HTML/CSS/JS VIVXh52a, V: 1.05
76025 ERIC CORDEIRO 24FRO106 - 4 entregar ate 23/maio/24 _____ / _____ / _____

CSS 2

Para aplicar estilos a links, pode-se usar qualquer propriedade CSS (color, font-family, background etc). Além disso, os links podem ser estilizados dependendo do estado em que estão. Existem 4 estados

a:link	link ainda não visitado
a:visited	link já visitado
a:hover	quando o mouse passa sobre o link
a:active	link no momento em que é clicado

Veja no exemplo:

```
a:link { color: #FF0000; }
a:visited { color: #00FF00; }
a:hover { color: #FF00FF; }
a:active { color: #0000FF; }
```

Exercício 1

Aproveite os arquivos brasil.html e brasil.css e garanta que o html tenha pelo menos 5 links. Aplique no arquivo brasil.css os links acima descritos com as cores aqua, violet, sienna e chartreuse pela ordem. Acrescente nos 4 o amarelo como cor de fundo (`background-color:yellow`) Veja o funcionamento disso tudo num browser.

Para aplicar estilos a listas, pode-se usar a propriedade `list-style-type:` que pode ser circle, square, upper-roman, lower-alpha. Pode-se usar também `list-style-image` que indica qual imagem usar como o marcador de itens em listas não ordenadas. Eis o formato:

```
ul { list-style-image: url('sqpurple.gif');}
```

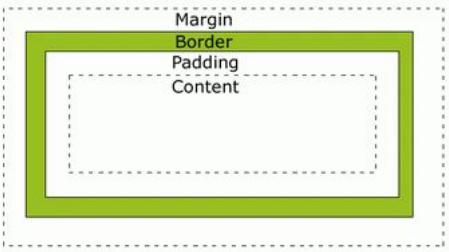
Para as tabelas, as propriedades são: `border:`, `border-collapse:`, `width:`, `height:`, `text-align:`, `vertical-align:`, `padding:`, `color:` e `background-color:`. Veja um exemplo de algumas dessas opções:

```
<style>
table, td, th { border: 1px solid green; }
th { background-color: green; color: white;}
</style>
...
<table> <tr> <th>Nome</th><th>Cidade</th>
<th>Contribuição</th> </tr>
<tr> <td>Pedro</td> <td>Curitiba</td>
<td>100</td>   </tr>
<tr> <td>Maria</td> <td>Joinville</td>
<td>300</td>   </tr>  </table>
```

Exercício 2

Defina arquivos tabela.html e tabela.css e escreva o exemplo acima. Sobre ele, aplique pelo menos 3 propriedades de tabelas diferentes e veja em cada caso o resultado em um browser.

Agora vamos estudar o modelo caixa do CSS (*CSS box model*). Todos os elementos HTML podem ser considerados como uma caixa. Este conceito é muito importante quando se pensa em layout. Pense no modelo da caixa como uma cebola quadrada composta de 4 camadas. De fora para dentro tem-se: margin, border, padding e finalmente o conteúdo que é o elemento HTML. Veja o desenho a seguir para entender o conceito



Eis um exemplo disso tudo junto

```
div { width: 300px;  padding: 25px;
   border: 25px solid navy; margin: 25px;}
```

Note que quando você define largura e altura de alguma coisa, essas medidas se referem à área de conteúdo. Para saber o tamanho total do elementos as outras medidas (se existentes) devem ser somadas. Por exemplo, o elemento abaixo ocupa 350px:

```
div { width: 320px;  padding: 10px;
    border: 5px solid gray; margin: 0;}
```

As propriedades da borda são: `border-style:` com os valores none, dotted, dashed, solid, double, groove, ridge, inset e outset. Todas as demais propriedades só atuam se a propriedade border-style for assinalada.

Depois vem `border-width:`, `border-color:` etc. Para definir a borda de maneira abreviada, a sequência é `border-width:`, `border-style:` (obrigatória), `border-color:`.

A seguir existem diversas propriedades para tratar o outline, elas são muito parecidas com as propriedades da borda e ficam como dever de casa para serem pesquisadas.

A margem clareia completamente o entorno de um elemento (fora da borda). A margem não tem cor de background e sempre é completamente transparente. As margens superior, direita, inferior e esquerda podem ser estabelecidas separadamente, ou pode-se usar a maneira abreviada e fazer tudo junto. Os valores possíveis de `margin:` são: auto (o browser calcula), tamanho (medida em px, pt, cm, ... - default é 0px), % medida em percentagem do tamanho do elemento contido e inherit quando a margem é herdada do elemento pai. Note que em tamanho pode-se usar valores negativos para sobrepujar algum conteúdo. As margens podem ser especificadas como `margin-top:`, `margin-bottom:`,`margin-right:` e `margin-left:` ou se preferir escrever apenas `margin:` e depois 4, 3 2 ou 1 valor de tamanho. A regra é: 3 tamanhos → top,laterais,bottom; 2 tamanhos → cima-baixo, laterais, e 1 tamanho → os 4 iguais.

O padding limpa o entorno do elemento (dentro da borda). Ele é afetado pela cor de fundo do elemento. De maneira muito parecida à margem, ele pode ser estabelecido com 4 propriedades ou de maneira abreviada com uma única propriedade.

Os elementos HTML também podem ter suas dimensões estabelecidas pelas propriedades: `height:`, `max-height:`, `min-height:`, `width:`, `max-width:` e `min-width:`. Veja um exemplo:

```
p { max-width: 20%;
    background-color: yellow;  }
```

A propriedade `display:` determina se e como um elemento é mostrado e a propriedade `visibility` especifica se um elemento deve ser visível ou escondido. Um elemento pode ser apagado fazendo `display: none` ou `visibility:hidden`. Este último apaga o elemento mas preserva o espaço ocupado por ele (em branco). Ou seja o layout permanece inalterado. Já `display: none` apaga o elemento e consome com o espaço ocupado por ele. O lay-out é refeito.

Exercício 3

Escreva um html chamado image.html e um css chamado image.css. Fale sobre o litoral paranaense. Apresente 3 fotos de mesma dimensão. Use 3 classes diferentes. No CSS alterne as propriedades vistas acima. Veja o resultado no browser.

Relembrando elementos em bloco (como `<h1>`, `<p>`, `<li>` e `<div>` por exemplo) são aqueles que forçam quebra de linha antes e depois dele. Já os elementos inline não quebram linha nem antes nem depois. Para transformar um elemento inline em bloco e vice-versa, usa-se a propriedade `display:`. Para inline o valor é `inline` e para bloco o valor é `block`. Veja no exemplo como transformar itens de lista em elementos inline (todos os itens serão mostrados na mesma linha).

```
li { display: inline; }
```

As propriedades de posicionamento permitem posicionar um elemento. Também permitem por um na frente do outro e especificam o que acontece quando o conteúdo de um elemento é muito grande. A primeira propriedade é `position:`. Ela tem 4 valores:

static é o valor default. O elemento é posicionado de acordo com o fluxo normal de preenchimento do browser. O elemento não é afetado pelas propriedades `top`, `bottom`, `left` e `right`.

fixed posição fixa em relação à janela do browser. Ele não é movido mesmo que a janela seja *scrolled*. Veja um exemplo
`p.fixo { position: fixed;`
`top: 30px; right: 5px;}`
Neste caso o elemento é removido do fluxo normal. Os outros elementos são posicionados como se o elemento fixo não existisse.

relative modifica a posição em relação à sua posição normal. Aqui por exemplo, os cabeçalhos h2, serão trazidos para fora da janela:
`h2 { position: relative; left: -20px;}`

absolute posiciona relativo ao primeiro elemento pai que tenha outra posição que não static. Se não existir, será o `<html>`.

Quando elementos são posicionados fora do fluxo normal eles podem sobrepujar outros elementos. A propriedade `z-index:` indica qual a ordem de apresentação. (podem ser atribuídos valores positivos ou negativos). A regra é que um número maior sempre vai aparecer na frente de um número menor. Veja o exemplo:

```
<style>
img { position: absolute; left: 0px;
    top: 0px; z-index: -2; }
</style> </head> <body>
<h1>Um cabeçalho</h1>
<img src="qualquer.gif" width="100" height="140">
</body>
```

Exercício 4

Copie o exemplo acima, usando uma figura qualquer. Varie a propriedade `z-index` e veja o que acontece.

☞ Para você fazer

1. Imprima os arquivos pedidos (brasil htlm e css, tabela htlm e css e image html e css) grampeie-os nesta folha e devolva tudo ou então gere-os no micro na sala de aula e mostre-os ao professor, durante a aula.

2. Descreva o que faz o código a seguir

```
<!DOCTYPE html> <html> <head> <style>
table, td, th {border: 1px solid black;  }
td {padding: 15px; }
</style> </head> <body>
<table>  <tr> <th>Firstname</th> <th>Lastname</th>
 <th>Savings</th> </tr>  <tr> <td>Peter</td>
 <td>Griffin</td> <td>$100</td> </tr> <tr>
 <td>Lois</td>  <td>Griffin</td> <td>$150</td> </tr>
</table> </body> </html>
```

3. Descreva o que faz o código a seguir

```
<!DOCTYPE html> <html> <head> <style>
a:link, a:visited {     display: block;
 font-weight: bold;     color: #ffffff;
 background-color: #98bf21;      width: 120px;
 text-align: center;      padding: 4px;
 text-decoration: none;  }
a:hover, a:active {    background-color: #7A991A;  }
</style> </head> <body>
<a href="default.asp" target="_blank">This is a link</a>
</body> </html>
```

.

106-76025 - ga/ a

U Positivo - UTFPR - PUCPr – 03/05/2024 - 17:27:26.0
Prof Dr P Kantek (pkantek@gmail.com)
Tutoriais HTML/CSS/JS VIVXh52a, V: 1.05
76032 FELIPE EIJI KANO 24FRO106 - 5
entregar ate 23/maio/24 _____ / _____ / _____

CSS 2

Para aplicar estilos a links, pode-se usar qualquer propriedade CSS (color, font-family, background etc). Além disso, os links podem ser estilizados dependendo do estado em que estão. Existem 4 estados

a:link	link ainda não visitado
a:visited	link já visitado
a:hover	quando o mouse passa sobre o link
a:active	link no momento em que é clicado

Veja no exemplo:

```
a:link { color: #FF0000; }
a:visited { color: #00FF00; }
a:hover { color: #FF00FF; }
a:active { color: #0000FF; }
```

Exercício 1

Aproveite os arquivos brasil.html e brasil.css e garanta que o html tenha pelo menos 5 links. Aplique no arquivo brasil.css os links acima descritos com as cores aqua, violet, sienna e chartreuse pela ordem. Acrescente nos 4 o amarelo como cor de fundo (`background-color:yellow`) Veja o funcionamento disso tudo num browser.

Para aplicar estilos a listas, pode-se usar a propriedade `list-style-type:` que pode ser circle, square, upper-roman, lower-alpha. Pode-se usar também `list-style-image` que indica qual imagem usar como o marcador de itens em listas não ordenadas. Eis o formato:

```
ul { list-style-image: url('sqpurple.gif');}
```

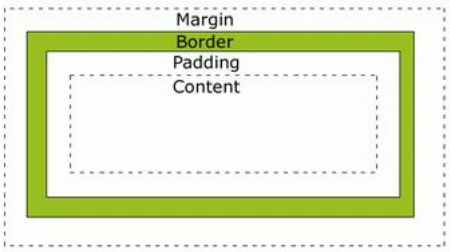
Para as tabelas, as propriedades são: `border:`, `border-collapse:`, `width:`, `height:`, `text-align:`, `vertical-align:`, `padding:`, `color:` e `background-color:`. Veja um exemplo de algumas dessas opções:

```
<style>
table, td, th { border: 1px solid green; }
th { background-color: green; color: white;}
</style>
...
<table> <tr> <th>Nome</th><th>Cidade</th>
<th>Contribuição</th> </tr>
<tr> <td>Pedro</td> <td>Curitiba</td>
<td>100</td>   </tr>
<tr> <td>Maria</td> <td>Joinville</td>
<td>300</td>   </tr>  </table>
```

Exercício 2

Defina arquivos tabela.html e tabela.css e escreva o exemplo acima. Sobre ele, aplique pelo menos 3 propriedades de tabelas diferentes e veja em cada caso o resultado em um browser.

Agora vamos estudar o modelo caixa do CSS (*CSS box model*). Todos os elementos HTML podem ser considerados como uma caixa. Este conceito é muito importante quando se pensa em layout. Pense no modelo da caixa como uma cebola quadrada composta de 4 camadas. De fora para dentro tem-se: margin, border, padding e finalmente o conteúdo que é o elemento HTML. Veja o desenho a seguir para entender o conceito



Eis um exemplo disso tudo junto

```
div { width: 300px;  padding: 25px;
   border: 25px solid navy; margin: 25px;}
```

Note que quando você define largura e altura de alguma coisa, essas medidas se referem à área de conteúdo. Para saber o tamanho total do elementos as outras medidas (se existentes) devem ser somadas. Por exemplo, o elemento abaixo ocupa 350px:

```
div { width: 320px;  padding: 10px;
    border: 5px solid gray; margin: 0;}
```

As propriedades da borda são: `border-style:` com os valores none, dotted, dashed, solid, double, groove, ridge, inset e outset. Todas as demais propriedades só atuam se a propriedade border-style for assinalada.

Depois vem `border-width:`, `border-color:` etc. Para definir a borda de maneira abreviada, a sequência é `border-width:`, `border-style:` (obrigatória), `border-color:`.

A seguir existem diversas propriedades para tratar o outline, elas são muito parecidas com as propriedades da borda e ficam como dever de casa para serem pesquisadas.

A margem clareia completamente o entorno de um elemento (fora da borda). A margem não tem cor de background e sempre é completamente transparente. As margens superior, direita, inferior e esquerda podem ser estabelecidas separadamente, ou pode-se usar a maneira abreviada e fazer tudo junto. Os valores possíveis de `margin:` são: auto (o browser calcula), tamanho (medida em px, pt, cm, ... - default é 0px), % medida em percentagem do tamanho do elemento contido e inherit quando a margem é herdada do elemento pai. Note que em tamanho pode-se usar valores negativos para sobrepujar algum conteúdo. As margens podem ser especificadas como `margin-top:`, `margin-bottom:`,`margin-right:` e `margin-left:` ou se preferir escrever apenas `margin:` e depois 4, 3 2 ou 1 valor de tamanho. A regra é: 3 tamanhos → top,laterais,bottom; 2 tamanhos → cima-baixo, laterais, e 1 tamanho → os 4 iguais.

O padding limpa o entorno do elemento (dentro da borda). Ele é afetado pela cor de fundo do elemento. De maneira muito parecida à margem, ele pode ser estabelecido com 4 propriedades ou de maneira abreviada com uma única propriedade.

Os elementos HTML também podem ter suas dimensões estabelecidas pelas propriedades: `height:`, `max-height:`, `min-height:`, `width:`, `max-width:` e `min-width:`. Veja um exemplo:

```
p { max-width: 20%;
    background-color: yellow;  }
```

A propriedade `display:` determina se e como um elemento é mostrado e a propriedade `visibility` especifica se um elemento deve ser visível ou escondido. Um elemento pode ser apagado fazendo `display: none` ou `visibility:hidden`. Este último apaga o elemento mas preserva o espaço ocupado por ele (em branco). Ou seja o layout permanece inalterado. Já `display: none` apaga o elemento e consome com o espaço ocupado por ele. O lay-out é refeito.

Exercício 3

Escreva um html chamado image.html e um css chamado image.css. Fale sobre o litoral paranaense. Apresente 3 fotos de mesma dimensão. Use 3 classes diferentes. No CSS alterne as propriedades vistas acima. Veja o resultado no browser.

Relembrando elementos em bloco (como `<h1>`, `<p>`, `<li>` e `<div>` por exemplo) são aqueles que forçam quebra de linha antes e depois dele. Já os elementos inline não quebram linha nem antes nem depois. Para transformar um elemento inline em bloco e vice-versa, usa-se a propriedade `display:`. Para inline o valor é `inline` e para bloco o valor é `block`. Veja no exemplo como transformar itens de lista em elementos inline (todos os itens serão mostrados na mesma linha).

```
li { display: inline; }
```

As propriedades de posicionamento permitem posicionar um elemento. Também permitem por um na frente do outro e especificam o que acontece quando o conteúdo de um elemento é muito grande. A primeira propriedade é `position:`. Ela tem 4 valores:

static é o valor default. O elemento é posicionado de acordo com o fluxo normal de preenchimento do browser. O elemento não é afetado pelas propriedades `top`, `bottom`, `left` e `right`.

fixed posição fixa em relação à janela do browser. Ele não é movido mesmo que a janela seja *scrolled*. Veja um exemplo
`p.fixo { position: fixed; top: 30px; right: 5px;}`
Neste caso o elemento é removido do fluxo normal. Os outros elementos são posicionados como se o elemento fixo não existisse.

relative modifica a posição em relação à sua posição normal. Aqui por exemplo, os cabeçalhos h2, serão trazidos para fora da janela:
`h2 { position: relative; left: -20px;}`

absolute posiciona relativo ao primeiro elemento pai que tenha outra posição que não static. Se não existir, será o `<html>`.

Quando elementos são posicionados fora do fluxo normal eles podem sobrepujar outros elementos. A propriedade `z-index:` indica qual a ordem de apresentação. (podem ser atribuídos valores positivos ou negativos). A regra é que um número maior sempre vai aparecer na frente de um número menor. Veja o exemplo:

```
<style>
img { position: absolute; left: 0px;
    top: 0px; z-index: -2; }
</style> </head> <body>
<h1>Um cabeçalho</h1>
<img src="qualquer.gif" width="100" height="140">
</body>
```

Exercício 4

Copie o exemplo acima, usando uma figura qualquer. Varie a propriedade `z-index` e veja o que acontece.

☞ Para você fazer

1. Imprima os arquivos pedidos (brasil htlm e css, tabela htlm e css e image html e css) grampeie-os nesta folha e devolva tudo ou então gere-os no micro na sala de aula e mostre-os ao professor, durante a aula.

2. Descreva o que faz o código a seguir

```
<!DOCTYPE html> <html> <head> <style>
div {background-color: lightblue;width: 200px;}
</style> </head> <body>
<div>Lorem ipsum dolor
adipiscing elit, sed
ut labore et dolore. </div>
</body> </html>
```

3. Descreva o que faz o código a seguir

```
<!DOCTYPE html> <html> <head> <style>
table, td, th { border: 1px solid green; }
th { background-color: green; color: white; }
</style> </head> <body> <table>  <tr>
<th>Firstname</th>  <th>Lastname</th> <th>Savings</th> </tr>
<tr><td>Peter</td><td>Griffin</td><td>$100</td></tr>
<tr><td>Lois</td><td>Griffin</td><td>$150</td></tr>
<tr><td>Cleveland</td><td>Brown</td><td>$250</td></tr>
</table> </body> </html>
```

.

106-76032 - ga/ a

U Positivo - UTFPR - PUCPr – 03/05/2024 - 17:27:26.0
Prof Dr P Kantek (pkantek@gmail.com)
Tutoriais HTML/CSS/JS VIVXh52a, V: 1.05
76049 FELIPE JESMIEL LEITE 24FRO106 - 6
entregar ate 23/maio/24 _____ / _____ / _____

CSS 2

Para aplicar estilos a links, pode-se usar qualquer propriedade CSS (color, font-family, background etc). Além disso, os links podem ser estilizados dependendo do estado em que estão. Existem 4 estados

a:link	link ainda não visitado
a:visited	link já visitado
a:hover	quando o mouse passa sobre o link
a:active	link no momento em que é clicado

Veja no exemplo:

```
a:link { color: #FF0000; }
a:visited { color: #00FF00; }
a:hover { color: #FF00FF; }
a:active { color: #0000FF; }
```

Exercício 1

Aproveite os arquivos brasil.html e brasil.css e garanta que o html tenha pelo menos 5 links. Aplique no arquivo brasil.css os links acima descritos com as cores aqua, violet, sienna e chartreuse pela ordem. Acrescente nos 4 o amarelo como cor de fundo (`background-color:yellow`) Veja o funcionamento disso tudo num browser.

Para aplicar estilos a listas, pode-se usar a propriedade `list-style-type:` que pode ser circle, square, upper-roman, lower-alpha. Pode-se usar também `list-style-image` que indica qual imagem usar como o marcador de itens em listas não ordenadas. Eis o formato:

```
ul { list-style-image: url('sqpurple.gif');}
```

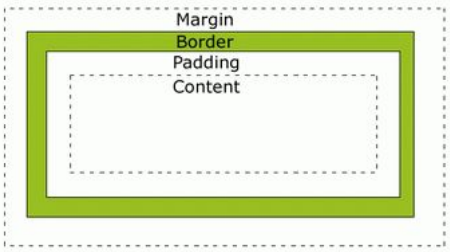
Para as tabelas, as propriedades são: `border:`, `border-collapse:`, `width:`, `height:`, `text-align:`, `vertical-align:`, `padding:`, `color:` e `background-color:`. Veja um exemplo de algumas dessas opções:

```
<style>
table, td, th { border: 1px solid green; }
th { background-color: green; color: white;}
</style>
...
<table> <tr> <th>Nome</th><th>Cidade</th>
<th>Contribuição</th> </tr>
<tr> <td>Pedro</td> <td>Curitiba</td>
<td>100</td>   </tr>
<tr> <td>Maria</td> <td>Joinville</td>
<td>300</td>   </tr>  </table>
```

Exercício 2

Defina arquivos tabela.html e tabela.css e escreva o exemplo acima. Sobre ele, aplique pelo menos 3 propriedades de tabelas diferentes e veja em cada caso o resultado em um browser.

Agora vamos estudar o modelo caixa do CSS (*CSS box model*). Todos os elementos HTML podem ser considerados como uma caixa. Este conceito é muito importante quando se pensa em layout. Pense no modelo da caixa como uma cebola quadrada composta de 4 camadas. De fora para dentro tem-se: margin, border, padding e finalmente o conteúdo que é o elemento HTML. Veja o desenho a seguir para entender o conceito



Eis um exemplo disso tudo junto

```
div { width: 300px;  padding: 25px;
   border: 25px solid navy; margin: 25px;}
```

Note que quando você define largura e altura de alguma coisa, essas medidas se referem à área de conteúdo. Para saber o tamanho total do elementos as outras medidas (se existentes) devem ser somadas. Por exemplo, o elemento abaixo ocupa 350px:

```
div { width: 320px;  padding: 10px;
    border: 5px solid gray; margin: 0;}
```

As propriedades da borda são: `border-style:` com os valores none, dotted, dashed, solid, double, groove, ridge, inset e outset. Todas as demais propriedades só atuam se a propriedade border-style for assinalada.

Depois vem `border-width:`, `border-color:` etc. Para definir a borda de maneira abreviada, a sequência é `border-width:`, `border-style:` (obrigatória), `border-color:`.

A seguir existem diversas propriedades para tratar o outline, elas são muito parecidas com as propriedades da borda e ficam como dever de casa para serem pesquisadas.

A margem clareia completamente o entorno de um elemento (fora da borda). A margem não tem cor de background e sempre é completamente transparente. As margens superior, direita, inferior e esquerda podem ser estabelecidas separadamente, ou pode-se usar a maneira abreviada e fazer tudo junto. Os valores possíveis de `margin:` são: auto (o browser calcula), tamanho (medida em px, pt, cm, ... - default é 0px), % medida em percentagem do tamanho do elemento contido e inherit quando a margem é herdada do elemento pai. Note que em tamanho pode-se usar valores negativos para sobrepujar algum conteúdo. As margens podem ser especificadas como `margin-top:`, `margin-bottom:`,`margin-right:` e `margin-left:` ou se preferir escrever apenas `margin:` e depois 4, 3 2 ou 1 valor de tamanho. A regra é: 3 tamanhos → top,laterais,bottom; 2 tamanhos → cima-baixo, laterais, e 1 tamanho → os 4 iguais.

O padding limpa o entorno do elemento (dentro da borda). Ele é afetado pela cor de fundo do elemento. De maneira muito parecida à margem, ele pode ser estabelecido com 4 propriedades ou de maneira abreviada com uma única propriedade.

Os elementos HTML também podem ter suas dimensões estabelecidas pelas propriedades: `height:`, `max-height:`, `min-height:`, `width:`, `max-width:` e `min-width:`. Veja um exemplo:

```
p { max-width: 20%;
    background-color: yellow;  }
```

A propriedade `display:` determina se e como um elemento é mostrado e a propriedade `visibility` especifica se um elemento deve ser visível ou escondido. Um elemento pode ser apagado fazendo `display: none` ou `visibility:hidden`. Este último apaga o elemento mas preserva o espaço ocupado por ele (em branco). Ou seja o layout permanece inalterado. Já `display: none` apaga o elemento e consome com o espaço ocupado por ele. O lay-out é refeito.

Exercício 3

Escreva um html chamado image.html e um css chamado image.css. Fale sobre o litoral paranaense. Apresente 3 fotos de mesma dimensão. Use 3 classes diferentes. No CSS alterne as propriedades vistas acima. Veja o resultado no browser.

Relembrando elementos em bloco (como `<h1>`, `<p>`, `<li>` e `<div>` por exemplo) são aqueles que forçam quebra de linha antes e depois dele. Já os elementos inline não quebram linha nem antes nem depois. Para transformar um elemento inline em bloco e vice-versa, usa-se a propriedade `display:`. Para inline o valor é `inline` e para bloco o valor é `block`. Veja no exemplo como transformar itens de lista em elementos inline (todos os itens serão mostrados na mesma linha).

```
li { display: inline; }
```

As propriedades de posicionamento permitem posicionar um elemento. Também permitem por um na frente do outro e especificam o que acontece quando o conteúdo de um elemento é muito grande. A primeira propriedade é `position:`. Ela tem 4 valores:

static é o valor default. O elemento é posicionado de acordo com o fluxo normal de preenchimento do browser. O elemento não é afetado pelas propriedades `top`, `bottom`, `left` e `right`.

fixed posição fixa em relação à janela do browser. Ele não é movido mesmo que a janela seja *scrolled*. Veja um exemplo
```
p.fixo { position: fixed;
     top: 30px; right: 5px;}
```
Neste caso o elemento é removido do fluxo normal. Os outros elementos são posicionados como se o elemento fixo não existisse.

relative modifica a posição em relação à sua posição normal. Aqui por exemplo, os cabeçalhos h2, serão trazidos para fora da janela:
`h2 { position: relative; left: -20px;}`

absolute posiciona relativo ao primeiro elemento pai que tenha outra posição que não static. Se não existir, será o `<html>`.

Quando elementos são posicionados fora do fluxo normal eles podem sobrepujar outros elementos. A propriedade `z-index:` indica qual a ordem de apresentação. (podem ser atribuídos valores positivos ou negativos). A regra é que um número maior sempre vai aparecer na frente de um número menor. Veja o exemplo:

```
<style>
img { position: absolute; left: 0px;
    top: 0px; z-index: -2; }
</style> </head> <body>
<h1>Um cabeçalho</h1>
<img src="qualquer.gif" width="100" height="140">
</body>
```

Exercício 4

Copie o exemplo acima, usando uma figura qualquer. Varie a propriedade `z-index` e veja o que acontece.

☞ Para você fazer

1. Imprima os arquivos pedidos (brasil htlm e css, tabela htlm e css e image html e css) grampeie-os nesta folha e devolva tudo ou então gere-os no micro na sala de aula e mostre-os ao professor, durante a aula.

2. Descreva o que faz o código a seguir

```
<!DOCTYPE html> <html> <head> <style>
div {background-color: lightblue;width: 200px;}
</style> </head> <body>
<div>Lorem ipsum dolor
adipiscing elit, sed
ut labore et dolore. </div>
</body> </html>
```

3. Descreva o que faz o código a seguir

```
<!DOCTYPE html> <html> <head> <style>
a.three:link {color:#ff0000;}
a.three:visited {color:#0000ff;}
a.three:hover {background:#66ff66;}
</style> </head> <body>
<p>Mouse over the links and watch them change layout:</p>
<p><b><a class="three" href="default.asp"
target="_blank">This link changes background-color</a></b></p>
</body> </html>
```

.

106-76049 - ga/ a

U Positivo - UTFPR - PUCPr – 03/05/2024 - 17:27:26.0
Prof Dr P Kantek (pkantek@gmail.com)
Tutoriais HTML/CSS/JS VIVXh52a, V: 1.05
76056 GABRIELA BERNARDO PASQUALOTT
24FRO106 - 7 entregar ate 23/maio/24 _____ / / _____

CSS 2

Para aplicar estilos a links, pode-se usar qualquer propriedade CSS (color, font-family, background etc). Além disso, os links podem ser estilizados dependendo do estado em que estão. Existem 4 estados

a:link	link ainda não visitado
a:visited	link já visitado
a:hover	quando o mouse passa sobre o link
a:active	link no momento em que é clicado

Veja no exemplo:

```
a:link { color: #FF0000; }
a:visited { color: #00FF00; }
a:hover { color: #FF00FF; }
a:active { color: #0000FF; }
```

Exercício 1

Aproveite os arquivos brasil.html e brasil.css e garanta que o html tenha pelo menos 5 links. Aplique no arquivo brasil.css os links acima descritos com as cores aqua, violet, sienna e chartreuse pela ordem. Acrescente nos 4 o amarelo como cor de fundo (`background-color:yellow`) Veja o funcionamento disso tudo num browser.

Para aplicar estilos a listas, pode-se usar a propriedade `list-style-type:` que pode ser circle, square, upper-roman, lower-alpha. Pode-se usar também `list-style-image` que indica qual imagem usar como o marcador de itens em listas não ordenadas. Eis o formato:

```
ul { list-style-image: url('sqpurple.gif');}
```

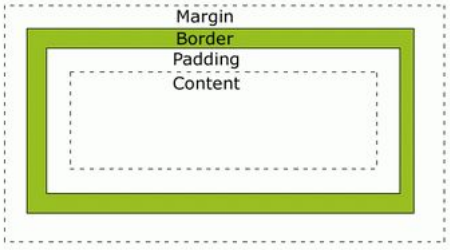
Para as tabelas, as propriedades são: `border:`, `border-collapse:`, `width:`, `height:`, `text-align:`, `vertical-align:`, `padding:`, `color:` e `background-color:`. Veja um exemplo de algumas dessas opções:

```
<style>
table, td, th { border: 1px solid green; }
th { background-color: green; color: white;}
</style>
...
<table> <tr> <th>Nome</th><th>Cidade</th>
<th>Contribuição</th> </tr>
<tr> <td>Pedro</td> <td>Curitiba</td>
<td>100</td>   </tr>
<tr> <td>Maria</td> <td>Joinville</td>
<td>300</td>   </tr>  </table>
```

Exercício 2

Defina arquivos tabela.html e tabela.css e escreva o exemplo acima. Sobre ele, aplique pelo menos 3 propriedades de tabelas diferentes e veja em cada caso o resultado em um browser.

Agora vamos estudar o modelo caixa do CSS (*CSS box model*). Todos os elementos HTML podem ser considerados como uma caixa. Este conceito é muito importante quando se pensa em layout. Pense no modelo da caixa como uma cebola quadrada composta de 4 camadas. De fora para dentro tem-se: margin, border, padding e finalmente o conteúdo que é o elemento HTML. Veja o desenho a seguir para entender o conceito



Eis um exemplo disso tudo junto

```
div { width: 300px;  padding: 25px;
   border: 25px solid navy; margin: 25px;}
```

Note que quando você define largura e altura de alguma coisa, essas medidas se referem à área de conteúdo. Para saber o tamanho total do elementos as outras medidas (se existentes) devem ser somadas. Por exemplo, o elemento abaixo ocupa 350px:

```
div { width: 320px;  padding: 10px;
    border: 5px solid gray; margin: 0;}
```

As propriedades da borda são: `border-style:` com os valores none, dotted, dashed, solid, double, groove, ridge, inset e outset. Todas as demais propriedades só atuam se a propriedade border-style for assinalada.

Depois vem `border-width:`, `border-color:` etc. Para definir a borda de maneira abreviada, a sequência é `border-width:`, `border-style:` (obrigatória), `border-color:`.

A seguir existem diversas propriedades para tratar o outline, elas são muito parecidas com as propriedades da borda e ficam como dever de casa para serem pesquisadas.

A margem clareia completamente o entorno de um elemento (fora da borda). A margem não tem cor de background e sempre é completamente transparente. As margens superior, direita, inferior e esquerda podem ser estabelecidas separadamente, ou pode-se usar a maneira abreviada e fazer tudo junto. Os valores possíveis de `margin:` são: auto (o browser calcula), tamanho (medida em px, pt, cm, ... - default é 0px), % medida em percentagem do tamanho do elemento contido e inherit quando a margem é herdada do elemento pai. Note que em tamanho pode-se usar valores negativos para sobrepujar algum conteúdo. As margens podem ser especificadas como `margin-top:`, `margin-bottom:`,`margin-right:` e `margin-left:` ou se preferir escrever apenas `margin:` e depois 4, 3 2 ou 1 valor de tamanho. A regra é: 3 tamanhos → top,laterais,bottom; 2 tamanhos → cima-baixo, laterais, e 1 tamanho → os 4 iguais.

O padding limpa o entorno do elemento (dentro da borda). Ele é afetado pela cor de fundo do elemento. De maneira muito parecida à margem, ele pode ser estabelecido com 4 propriedades ou de maneira abreviada com uma única propriedade.

Os elementos HTML também podem ter suas dimensões estabelecidas pelas propriedades: `height:`, `max-height:`, `min-height:`, `width:`, `max-width:` e `min-width:`. Veja um exemplo:

```
p { max-width: 20%;
    background-color: yellow;  }
```

A propriedade `display:` determina se e como um elemento é mostrado e a propriedade `visibility` especifica se um elemento deve ser visível ou escondido. Um elemento pode ser apagado fazendo `display: none` ou `visibility:hidden`. Este último apaga o elemento mas preserva o espaço ocupado por ele (em branco). Ou seja o layout permanece inalterado. Já `display: none` apaga o elemento e consome com o espaço ocupado por ele. O lay-out é refeito.

Exercício 3

Escreva um html chamado image.html e um css chamado image.css. Fale sobre o litoral paranaense. Apresente 3 fotos de mesma dimensão. Use 3 classes diferentes. No CSS alterne as propriedades vistas acima. Veja o resultado no browser.

Relembrando elementos em bloco (como `<h1>`, `<p>`, `<li>` e `<div>` por exemplo) são aqueles que forçam quebra de linha antes e depois dele. Já os elementos inline não quebram linha nem antes nem depois. Para transformar um elemento inline em bloco e vice-versa, usa-se a propriedade `display:`. Para inline o valor é `inline` e para bloco o valor é `block`. Veja no exemplo como transformar itens de lista em elementos inline (todos os itens serão mostrados na mesma linha).

```
li { display: inline; }
```

As propriedades de posicionamento permitem posicionar um elemento. Também permitem por um na frente do outro e especificam o que acontece quando o conteúdo de um elemento é muito grande. A primeira propriedade é `position:`. Ela tem 4 valores:

static é o valor default. O elemento é posicionado de acordo com o fluxo normal de preenchimento do browser. O elemento não é afetado pelas propriedades `top`, `bottom`, `left` e `right`.

fixed posição fixa em relação à janela do browser. Ele não é movido mesmo que a janela seja *scrolled*. Veja um exemplo
`p.fixo { position: fixed; top: 30px; right: 5px;}`
Neste caso o elemento é removido do fluxo normal. Os outros elementos são posicionados como se o elemento fixo não existisse.

relative modifica a posição em relação à sua posição normal. Aqui por exemplo, os cabeçalhos h2, serão trazidos para fora da janela:
`h2 { position: relative; left: -20px;}`

absolute posiciona relativo ao primeiro elemento pai que tenha outra posição que não static. Se não existir, será o `<html>`.

Quando elementos são posicionados fora do fluxo normal eles podem sobrepujar outros elementos. A propriedade `z-index:` indica qual a ordem de apresentação. (podem ser atribuídos valores positivos ou negativos). A regra é que um número maior sempre vai aparecer na frente de um número menor. Veja o exemplo:

```
<style>
img { position: absolute; left: 0px;
    top: 0px; z-index: -2; }
</style> </head> <body>
<h1>Um cabeçalho</h1>
<img src="qualquer.gif" width="100" height="140">
</body>
```

Exercício 4

Copie o exemplo acima, usando uma figura qualquer. Varie a propriedade `z-index` e veja o que acontece.

☞ Para você fazer

1. Imprima os arquivos pedidos (brasil htlm e css, tabela htlm e css e image html e css) grampeie-os nesta folha e devolva tudo ou então gere-os no micro na sala de aula e mostre-os ao professor, durante a aula.

2. Descreva o que faz o código a seguir

```
<!DOCTYPE html> <html> <head> <style>
div {background-color: lightblue;
 width: 200px; padding: 25px; }
</style> </head> <body>
<div>Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod.</div>
</body> </html>
```

3. Descreva o que faz o código a seguir

```
<!DOCTYPE html> <html> <head> <style>
a.three:link {color:#ff0000;}
a.three:visited {color:#0000ff;}
a.three:hover {background:#66ff66;}
</style> </head> <body>
<p>Mouse over the links and watch them change layout:</p>
<p><b><a class="three" href="default.asp"
target="_blank">This link changes background-color</a></b></p>
</body> </html>
```

.

106-76056 - ga/ a

U Positivo - UTFPR - PUCPr – 03/05/2024 - 17:27:26.0
Prof Dr P Kantek (pkantek@gmail.com)
Tutoriais HTML/CSS/JS VIVXh52a, V: 1.05
76063 GUILHERME VITOR DA SILVA
24FRO106 - 8 entregar ate 23/maio/24 _____ / / _____

CSS 2

Para aplicar estilos a links, pode-se usar qualquer propriedade CSS (color, font-family, background etc). Além disso, os links podem ser estilizados dependendo do estado em que estão. Existem 4 estados

a:link	link ainda não visitado
a:visited	link já visitado
a:hover	quando o mouse passa sobre o link
a:active	link no momento em que é clicado

Veja no exemplo:

```
a:link { color: #FF0000; }
a:visited { color: #00FF00; }
a:hover { color: #FF00FF; }
a:active { color: #0000FF; }
```

Exercício 1

Aproveite os arquivos brasil.html e brasil.css e garanta que o html tenha pelo menos 5 links. Aplique no arquivo brasil.css os links acima descritos com as cores aqua, violet, sienna e chartreuse pela ordem. Acrescente nos 4 o amarelo como cor de fundo (`background-color:yellow`) Veja o funcionamento disso tudo num browser.

Para aplicar estilos a listas, pode-se usar a propriedade `list-style-type:` que pode ser circle, square, upper-roman, lower-alpha. Pode-se usar também `list-style-image` que indica qual imagem usar como o marcador de itens em listas não ordenadas. Eis o formato:

```
ul { list-style-image: url('sqpurple.gif');}
```

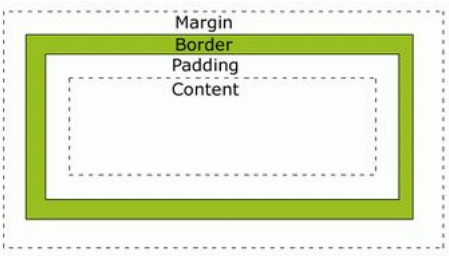
Para as tabelas, as propriedades são: `border:`, `border-collapse:`, `width:`, `height:`, `text-align:`, `vertical-align:`, `padding:`, `color:` e `background-color:`. Veja um exemplo de algumas dessas opções:

```
<style>
table, td, th { border: 1px solid green; }
th { background-color: green; color: white;}
</style>
...
<table> <tr> <th>Nome</th><th>Cidade</th>
<th>Contribuição</th> </tr>
<tr> <td>Pedro</td> <td>Curitiba</td>
<td>100</td>   </tr>
<tr> <td>Maria</td> <td>Joinville</td>
<td>300</td>   </tr>  </table>
```

Exercício 2

Defina arquivos tabela.html e tabela.css e escreva o exemplo acima. Sobre ele, aplique pelo menos 3 propriedades de tabelas diferentes e veja em cada caso o resultado em um browser.

Agora vamos estudar o modelo caixa do CSS (*CSS box model*). Todos os elementos HTML podem ser considerados como uma caixa. Este conceito é muito importante quando se pensa em layout. Pense no modelo da caixa como uma cebola quadrada composta de 4 camadas. De fora para dentro tem-se: margin, border, padding e finalmente o conteúdo que é o elemento HTML. Veja o desenho a seguir para entender o conceito



Eis um exemplo disso tudo junto

```
div { width: 300px;  padding: 25px;
   border: 25px solid navy; margin: 25px;}
```

Note que quando você define largura e altura de alguma coisa, essas medidas se referem à área de conteúdo. Para saber o tamanho total do elementos as outras medidas (se existentes) devem ser somadas. Por exemplo, o elemento abaixo ocupa 350px:

```
div { width: 320px;  padding: 10px;
    border: 5px solid gray; margin: 0;}
```

As propriedades da borda são: `border-style:` com os valores none, dotted, dashed, solid, double, groove, ridge, inset e outset. Todas as demais propriedades só atuam se a propriedade border-style for assinalada.

Depois vem `border-width:`, `border-color:` etc. Para definir a borda de maneira abreviada, a sequência é `border-width:`, `border-style:` (obrigatória), `border-color:`.

A seguir existem diversas propriedades para tratar o outline, elas são muito parecidas com as propriedades da borda e ficam como dever de casa para serem pesquisadas.

A margem clareia completamente o entorno de um elemento (fora da borda). A margem não tem cor de background e sempre é completamente transparente. As margens superior, direita, inferior e esquerda podem ser estabelecidas separadamente, ou pode-se usar a maneira abreviada e fazer tudo junto. Os valores possíveis de `margin:` são: auto (o browser calcula), tamanho (medida em px, pt, cm, ... - default é 0px), % medida em percentagem do tamanho do elemento contido e inherit quando a margem é herdada do elemento pai. Note que em tamanho pode-se usar valores negativos para sobrepujar algum conteúdo. As margens podem ser especificadas como `margin-top:`, `margin-bottom:`,`margin-right:` e `margin-left:` ou se preferir escrever apenas `margin:` e depois 4, 3 2 ou 1 valor de tamanho. A regra é: 3 tamanhos → top,laterais,bottom; 2 tamanhos → cima-baixo, laterais, e 1 tamanho → os 4 iguais.

O padding limpa o entorno do elemento (dentro da borda). Ele é afetado pela cor de fundo do elemento. De maneira muito parecida à margem, ele pode ser estabelecido com 4 propriedades ou de maneira abreviada com uma única propriedade.

Os elementos HTML também podem ter suas dimensões estabelecidas pelas propriedades: `height:`, `max-height:`, `min-height:`, `width:`, `max-width:` e `min-width:`. Veja um exemplo:

```
p { max-width: 20%;
    background-color: yellow;  }
```

A propriedade `display:` determina se e como um elemento é mostrado e a propriedade `visibility` especifica se um elemento deve ser visível ou escondido. Um elemento pode ser apagado fazendo `display: none` ou `visibility:hidden`. Este último apaga o elemento mas preserva o espaço ocupado por ele (em branco). Ou seja o layout permanece inalterado. Já `display: none` apaga o elemento e consome com o espaço ocupado por ele. O lay-out é refeito.

Exercício 3

Escreva um html chamado image.html e um css chamado image.css. Fale sobre o litoral paranaense. Apresente 3 fotos de mesma dimensão. Use 3 classes diferentes. No CSS alterne as propriedades vistas acima. Veja o resultado no browser.

Relembrando elementos em bloco (como `<h1>`, `<p>`, `<li>` e `<div>` por exemplo) são aqueles que forçam quebra de linha antes e depois dele. Já os elementos inline não quebram linha nem antes nem depois. Para transformar um elemento inline em bloco e vice-versa, usa-se a propriedade `display:`. Para inline o valor é `inline` e para bloco o valor é `block`. Veja no exemplo como transformar itens de lista em elementos inline (todos os itens serão mostrados na mesma linha).

```
li { display: inline; }
```

As propriedades de posicionamento permitem posicionar um elemento. Também permitem por um na frente do outro e especificam o que acontece quando o conteúdo de um elemento é muito grande. A primeira propriedade é `position:`. Ela tem 4 valores:

static é o valor default. O elemento é posicionado de acordo com o fluxo normal de preenchimento do browser. O elemento não é afetado pelas propriedades `top`, `bottom`, `left` e `right`.

fixed posição fixa em relação à janela do browser. Ele não é movido mesmo que a janela seja *scrolled*. Veja um exemplo
`p.fixo { position: fixed;`
`top: 30px; right: 5px;}`
Neste caso o elemento é removido do fluxo normal. Os outros elementos são posicionados como se o elemento fixo não existisse.

relative modifica a posição em relação à sua posição normal. Aqui por exemplo, os cabeçalhos h2, serão trazidos para fora da janela:
`h2 { position: relative; left: -20px;}`

absolute posiciona relativo ao primeiro elemento pai que tenha outra posição que não static. Se não existir, será o `<html>`.

Quando elementos são posicionados fora do fluxo normal eles podem sobrepujar outros elementos. A propriedade `z-index:` indica qual a ordem de apresentação. (podem ser atribuídos valores positivos ou negativos). A regra é que um número maior sempre vai aparecer na frente de um número menor. Veja o exemplo:

```
<style>
img { position: absolute; left: 0px;
    top: 0px; z-index: -2; }
</style> </head> <body>
<h1>Um cabeçalho</h1>
<img src="qualquer.gif" width="100" height="140">
</body>
```

Exercício 4

Copie o exemplo acima, usando uma figura qualquer. Varie a propriedade `z-index` e veja o que acontece.

☞ Para você fazer

1. Imprima os arquivos pedidos (brasil htlm e css, tabela htlm e css e image html e css) grampeie-os nesta folha e devolva tudo ou então gere-os no micro na sala de aula e mostre-os ao professor, durante a aula.

2. Descreva o que faz o código a seguir

```
<!DOCTYPE html> <html> <head> <style>
a.five:link {color:#ff0000;text-decoration:none;}
a.five:visited {color:#0000ff;text-decoration:none;}
a.five:hover {text-decoration:underline;}
</style> </head> <body>
<p>Mouse over the links and watch them change layout:</p>
<p><b><a class="five" href="default.asp"
target="_blank">This link changes text-decoration</a></b></p>
</body> </html>
```

3. Descreva o que faz o código a seguir

```
<!DOCTYPE html> <html> <head> <style>
table, td, th {border: 1px solid black;  }
td {padding: 15px; }
</style> </head> <body>
<table>  <tr> <th>Firstname</th> <th>Lastname</th>
 <th>Savings</th> </tr>  <tr> <td>Peter</td>
 <td>Griffin</td> <td>$100</td> </tr> <tr>
 <td>Lois</td>  <td>Griffin</td> <td>$150</td> </tr>
</table> </body> </html>
```

.

106-76063 - ga/ a

U Positivo - UTFPR - PUCPr – 03/05/2024 - 17:27:26.0
Prof Dr P Kantek (pkantek@gmail.com)
Tutoriais HTML/CSS/JS VIVXh52a, V: 1.05
76087 LAURA PACHECO DELATTRE
24FRO106 - 9 entregar ate 23/maio/24 _____ / / _____

CSS 2

Para aplicar estilos a links, pode-se usar qualquer propriedade CSS (color, font-family, background etc). Além disso, os links podem ser estilizados dependendo do estado em que estão. Existem 4 estados

a:link	link ainda não visitado
a:visited	link já visitado
a:hover	quando o mouse passa sobre o link
a:active	link no momento em que é clicado

Veja no exemplo:

```
a:link { color: #FF0000; }
a:visited { color: #00FF00; }
a:hover { color: #FF00FF; }
a:active { color: #0000FF; }
```

Exercício 1

Aproveite os arquivos brasil.html e brasil.css e garanta que o html tenha pelo menos 5 links. Aplique no arquivo brasil.css os links acima descritos com as cores aqua, violet, sienna e chartreuse pela ordem. Acrescente nos 4 o amarelo como cor de fundo (`background-color:yellow`) Veja o funcionamento disso tudo num browser.

Para aplicar estilos a listas, pode-se usar a propriedade `list-style-type:` que pode ser circle, square, upper-roman, lower-alpha. Pode-se usar também `list-style-image` que indica qual imagem usar como o marcador de itens em listas não ordenadas. Eis o formato:

```
ul { list-style-image: url('sqpurple.gif');}
```

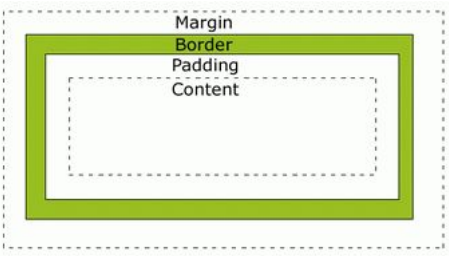
Para as tabelas, as propriedades são: `border:`, `border-collapse:`, `width:`, `height:`, `text-align:`, `vertical-align:`, `padding:`, `color:` e `background-color:`. Veja um exemplo de algumas dessas opções:

```
<style>
table, td, th { border: 1px solid green; }
th { background-color: green; color: white;}
</style>
...
<table> <tr> <th>Nome</th><th>Cidade</th>
<th>Contribuição</th> </tr>
<tr> <td>Pedro</td> <td>Curitiba</td>
<td>100</td>   </tr>
<tr> <td>Maria</td> <td>Joinville</td>
<td>300</td>   </tr>  </table>
```

Exercício 2

Defina arquivos tabela.html e tabela.css e escreva o exemplo acima. Sobre ele, aplique pelo menos 3 propriedades de tabelas diferentes e veja em cada caso o resultado em um browser.

Agora vamos estudar o modelo caixa do CSS (*CSS box model*). Todos os elementos HTML podem ser considerados como uma caixa. Este conceito é muito importante quando se pensa em layout. Pense no modelo da caixa como uma cebola quadrada composta de 4 camadas. De fora para dentro tem-se: margin, border, padding e finalmente o conteúdo que é o elemento HTML. Veja o desenho a seguir para entender o conceito



Eis um exemplo disso tudo junto

```
div { width: 300px;  padding: 25px;
   border: 25px solid navy; margin: 25px;}
```

Note que quando você define largura e altura de alguma coisa, essas medidas se referem à área de conteúdo. Para saber o tamanho total do elementos as outras medidas (se existentes) devem ser somadas. Por exemplo, o elemento abaixo ocupa 350px:

```
div { width: 320px;  padding: 10px;
    border: 5px solid gray; margin: 0;}
```

As propriedades da borda são: `border-style:` com os valores none, dotted, dashed, solid, double, groove, ridge, inset e outset. Todas as demais propriedades só atuam se a propriedade border-style for assinalada.

Depois vem `border-width:`, `border-color:` etc. Para definir a borda de maneira abreviada, a sequência é `border-width:`, `border-style:` (obrigatória), `border-color:`.

A seguir existem diversas propriedades para tratar o outline, elas são muito parecidas com as propriedades da borda e ficam como dever de casa para serem pesquisadas.

A margem clareia completamente o entorno de um elemento (fora da borda). A margem não tem cor de background e sempre é completamente transparente. As margens superior, direita, inferior e esquerda podem ser estabelecidas separadamente, ou pode-se usar a maneira abreviada e fazer tudo junto. Os valores possíveis de `margin:` são: auto (o browser calcula), tamanho (medida em px, pt, cm, ... - default é 0px), % medida em percentagem do tamanho do elemento contido e inherit quando a margem é herdada do elemento pai. Note que em tamanho pode-se usar valores negativos para sobrepujar algum conteúdo. As margens podem ser especificadas como `margin-top:`, `margin-bottom:`,`margin-right:` e `margin-left:` ou se preferir escrever apenas `margin:` e depois 4, 3 2 ou 1 valor de tamanho. A regra é: 3 tamanhos → top,laterais,bottom; 2 tamanhos → cima-baixo, laterais, e 1 tamanho → os 4 iguais.

O padding limpa o entorno do elemento (dentro da borda). Ele é afetado pela cor de fundo do elemento. De maneira muito parecida à margem, ele pode ser estabelecido com 4 propriedades ou de maneira abreviada com uma única propriedade.

Os elementos HTML também podem ter suas dimensões estabelecidas pelas propriedades: `height:`, `max-height:`, `min-height:`, `width:`, `max-width:` e `min-width:`. Veja um exemplo:

```
p { max-width: 20%;
    background-color: yellow;  }
```

A propriedade `display:` determina se e como um elemento é mostrado e a propriedade `visibility` especifica se um elemento deve ser visível ou escondido. Um elemento pode ser apagado fazendo `display: none` ou `visibility:hidden`. Este último apaga o elemento mas preserva o espaço ocupado por ele (em branco). Ou seja o layout permanece inalterado. Já `display: none` apaga o elemento e consome com o espaço ocupado por ele. O lay-out é refeito.

Exercício 3

Escreva um html chamado image.html e um css chamado image.css. Fale sobre o litoral paranaense. Apresente 3 fotos de mesma dimensão. Use 3 classes diferentes. No CSS alterne as propriedades vistas acima. Veja o resultado no browser.

Relembrando elementos em bloco (como `<h1>`, `<p>`, `<li>` e `<div>` por exemplo) são aqueles que forçam quebra de linha antes e depois dele. Já os elementos inline não quebram linha nem antes nem depois. Para transformar um elemento inline em bloco e vice-versa, usa-se a propriedade `display:`. Para inline o valor é `inline` e para bloco o valor é `block`. Veja no exemplo como transformar itens de lista em elementos inline (todos os itens serão mostrados na mesma linha).

```
li { display: inline; }
```

As propriedades de posicionamento permitem posicionar um elemento. Também permitem por um na frente do outro e especificam o que acontece quando o conteúdo de um elemento é muito grande. A primeira propriedade é `position:`. Ela tem 4 valores:

static é o valor default. O elemento é posicionado de acordo com o fluxo normal de preenchimento do browser. O elemento não é afetado pelas propriedades `top`, `bottom`, `left` e `right`.

fixed posição fixa em relação à janela do browser. Ele não é movido mesmo que a janela seja *scrolled*. Veja um exemplo
`p.fixo { position: fixed; top: 30px; right: 5px;}`
Neste caso o elemento é removido do fluxo normal. Os outros elementos são posicionados como se o elemento fixo não existisse.

relative modifica a posição em relação à sua posição normal. Aqui por exemplo, os cabeçalhos h2, serão trazidos para fora da janela:
`h2 { position: relative; left: -20px;}`

absolute posiciona relativo ao primeiro elemento pai que tenha outra posição que não static. Se não existir, será o `<html>`.

Quando elementos são posicionados fora do fluxo normal eles podem sobrepujar outros elementos. A propriedade `z-index:` indica qual a ordem de apresentação. (podem ser atribuídos valores positivos ou negativos). A regra é que um número maior sempre vai aparecer na frente de um número menor. Veja o exemplo:

```
<style>
img { position: absolute; left: 0px;
    top: 0px; z-index: -2; }
</style> </head> <body>
<h1>Um cabeçalho</h1>
<img src="qualquer.gif" width="100" height="140">
</body>
```

Exercício 4

Copie o exemplo acima, usando uma figura qualquer. Varie a propriedade `z-index` e veja o que acontece.

☞ Para você fazer

1. Imprima os arquivos pedidos (brasil htlm e css, tabela htlm e css e image html e css) grampeie-os nesta folha e devolva tudo ou então gere-os no micro na sala de aula e mostre-os ao professor, durante a aula.

2. Descreva o que faz o código a seguir

```
<!DOCTYPE html> <html> <head> <style>
div {background-color: lightblue;
 width: 200px; padding: 25px;
 border: 25px solid navy; }
</style> </head> <body>
<div>Lorem ipsum dolor sit
amet, consectetur adipiscing
elit, sed.</div>
</body> </html>
```

3. Descreva o que faz o código a seguir

```
<!DOCTYPE html> <html> <head> <style>
p {background-color: yellow; }
p.padding { padding-top: 25px; padding-right: 50px;
  padding-bottom: 25px; padding-left: 50px; }
</style> </head> <body>
<p>This is a paragraph with no specified padding.</p>
<p class="padding">This is a paragraph with
specified paddings.</p> </body> </html>
```

.

106-76087 - ga/ a

U Positivo - UTFPR - PUCPr – 03/05/2024 - 17:27:26.0
Prof Dr P Kantek (pkantek@gmail.com)
Tutoriais HTML/CSS/JS VIVXh52a, V: 1.05
76106 LUIZ MIGUEL OLINEK 24FRO106 - 10
entregar ate 23/maio/24 _____ / _____ / _____

CSS 2

Para aplicar estilos a links, pode-se usar qualquer propriedade CSS (color, font-family, background etc). Além disso, os links podem ser estilizados dependendo do estado em que estão. Existem 4 estados

a:link	link ainda não visitado
a:visited	link já visitado
a:hover	quando o mouse passa sobre o link
a:active	link no momento em que é clicado

Veja no exemplo:

```
a:link { color: #FF0000; }
a:visited { color: #00FF00; }
a:hover { color: #FF00FF; }
a:active { color: #0000FF; }
```

Exercício 1

Aproveite os arquivos brasil.html e brasil.css e garanta que o html tenha pelo menos 5 links. Aplique no arquivo brasil.css os links acima descritos com as cores aqua, violet, sienna e chartreuse pela ordem. Acrescente nos 4 o amarelo como cor de fundo (`background-color:yellow`) Veja o funcionamento disso tudo num browser.

Para aplicar estilos a listas, pode-se usar a propriedade `list-style-type:` que pode ser circle, square, upper-roman, lower-alpha. Pode-se usar também `list-style-image` que indica qual imagem usar como o marcador de itens em listas não ordenadas. Eis o formato:

```
ul { list-style-image: url('sqpurple.gif');}
```

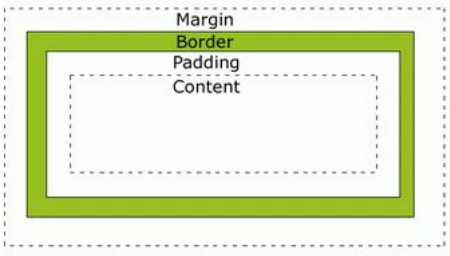
Para as tabelas, as propriedades são: `border:`, `border-collapse:`, `width:`, `height:`, `text-align:`, `vertical-align:`, `padding:`, `color:` e `background-color:`. Veja um exemplo de algumas dessas opções:

```
<style>
table, td, th { border: 1px solid green; }
th { background-color: green; color: white;}
</style>
...
<table> <tr> <th>Nome</th><th>Cidade</th>
<th>Contribuição</th> </tr>
<tr> <td>Pedro</td> <td>Curitiba</td>
<td>100</td>   </tr>
<tr> <td>Maria</td> <td>Joinville</td>
<td>300</td>   </tr>  </table>
```

Exercício 2

Defina arquivos tabela.html e tabela.css e escreva o exemplo acima. Sobre ele, aplique pelo menos 3 propriedades de tabelas diferentes e veja em cada caso o resultado em um browser.

Agora vamos estudar o modelo caixa do CSS (*CSS box model*). Todos os elementos HTML podem ser considerados como uma caixa. Este conceito é muito importante quando se pensa em layout. Pense no modelo da caixa como uma cebola quadrada composta de 4 camadas. De fora para dentro tem-se: margin, border, padding e finalmente o conteúdo que é o elemento HTML. Veja o desenho a seguir para entender o conceito



Eis um exemplo disso tudo junto

```
div { width: 300px;  padding: 25px;
   border: 25px solid navy; margin: 25px;}
```

Note que quando você define largura e altura de alguma coisa, essas medidas se referem à área de conteúdo. Para saber o tamanho total do elementos as outras medidas (se existentes) devem ser somadas. Por exemplo, o elemento abaixo ocupa 350px:

```
div { width: 320px;  padding: 10px;
    border: 5px solid gray; margin: 0;}
```

As propriedades da borda são: `border-style:` com os valores none, dotted, dashed, solid, double, groove, ridge, inset e outset. Todas as demais propriedades só atuam se a propriedade border-style for assinalada.

Depois vem `border-width:`, `border-color:` etc. Para definir a borda de maneira abreviada, a sequência é `border-width:`, `border-style:` (obrigatória), `border-color:`.

A seguir existem diversas propriedades para tratar o outline, elas são muito parecidas com as propriedades da borda e ficam como dever de casa para serem pesquisadas.

A margem clareia completamente o entorno de um elemento (fora da borda). A margem não tem cor de background e sempre é completamente transparente. As margens superior, direita, inferior e esquerda podem ser estabelecidas separadamente, ou pode-se usar a maneira abreviada e fazer tudo junto. Os valores possíveis de `margin:` são: auto (o browser calcula), tamanho (medida em px, pt, cm, ... - default é 0px), % medida em percentagem do tamanho do elemento contido e inherit quando a margem é herdada do elemento pai. Note que em tamanho pode-se usar valores negativos para sobrepujar algum conteúdo. As margens podem ser especificadas como `margin-top:`, `margin-bottom:`,`margin-right:` e `margin-left:` ou se preferir escrever apenas `margin:` e depois 4, 3 2 ou 1 valor de tamanho. A regra é: 3 tamanhos → top,laterais,bottom; 2 tamanhos → cima-baixo, laterais, e 1 tamanho → os 4 iguais.

O padding limpa o entorno do elemento (dentro da borda). Ele é afetado pela cor de fundo do elemento. De maneira muito parecida à margem, ele pode ser estabelecido com 4 propriedades ou de maneira abreviada com uma única propriedade.

Os elementos HTML também podem ter suas dimensões estabelecidas pelas propriedades: `height:`, `max-height:`, `min-height:`, `width:`, `max-width:` e `min-width:`. Veja um exemplo:

```
p { max-width: 20%;
    background-color: yellow;  }
```

A propriedade `display:` determina se e como um elemento é mostrado e a propriedade `visibility` especifica se um elemento deve ser visível ou escondido. Um elemento pode ser apagado fazendo `display: none` ou `visibility:hidden`. Este último apaga o elemento mas preserva o espaço ocupado por ele (em branco). Ou seja o layout permanece inalterado. Já `display: none` apaga o elemento e consome com o espaço ocupado por ele. O lay-out é refeito.

Exercício 3

Escreva um html chamado image.html e um css chamado image.css. Fale sobre o litoral paranaense. Apresente 3 fotos de mesma dimensão. Use 3 classes diferentes. No CSS alterne as propriedades vistas acima. Veja o resultado no browser.

Relembrando elementos em bloco (como `<h1>`, `<p>`, `<li>` e `<div>` por exemplo) são aqueles que forçam quebra de linha antes e depois dele. Já os elementos inline não quebram linha nem antes nem depois. Para transformar um elemento inline em bloco e vice-versa, usa-se a propriedade `display:`. Para inline o valor é `inline` e para bloco o valor é `block`. Veja no exemplo como transformar itens de lista em elementos inline (todos os itens serão mostrados na mesma linha).

```
li { display: inline; }
```

As propriedades de posicionamento permitem posicionar um elemento. Também permitem por um na frente do outro e especificam o que acontece quando o conteúdo de um elemento é muito grande. A primeira propriedade é `position:`. Ela tem 4 valores:

static é o valor default. O elemento é posicionado de acordo com o fluxo normal de preenchimento do browser. O elemento não é afetado pelas propriedades `top`, `bottom`, `left` e `right`.

fixed posição fixa em relação à janela do browser. Ele não é movido mesmo que a janela seja *scrolled*. Veja um exemplo
`p.fixo { position: fixed;`
`top: 30px; right: 5px;}`
Neste caso o elemento é removido do fluxo normal. Os outros elementos são posicionados como se o elemento fixo não existisse.

relative modifica a posição em relação à sua posição normal. Aqui por exemplo, os cabeçalhos h2, serão trazidos para fora da janela:
`h2 { position: relative; left: -20px;}`

absolute posiciona relativo ao primeiro elemento pai que tenha outra posição que não static. Se não existir, será o `<html>`.

Quando elementos são posicionados fora do fluxo normal eles podem sobrepujar outros elementos. A propriedade `z-index:` indica qual a ordem de apresentação. (podem ser atribuídos valores positivos ou negativos). A regra é que um número maior sempre vai aparecer na frente de um número menor. Veja o exemplo:

```
<style>
img { position: absolute; left: 0px;
    top: 0px; z-index: -2; }
</style> </head> <body>
<h1>Um cabeçalho</h1>
<img src="qualquer.gif" width="100" height="140">
</body>
```

Exercício 4

Copie o exemplo acima, usando uma figura qualquer. Varie a propriedade `z-index` e veja o que acontece.

☞ Para você fazer

1. Imprima os arquivos pedidos (brasil htlm e css, tabela htlm e css e image html e css) grampeie-os nesta folha e devolva tudo ou então gere-os no micro na sala de aula e mostre-os ao professor, durante a aula.

2. Descreva o que faz o código a seguir

```
<!DOCTYPE html> <html> <head> <style>
a:link, a:visited {     display: block;
 font-weight: bold;     color: #ffffff;
 background-color: #98bf21;      width: 120px;
 text-align: center;      padding: 4px;
 text-decoration: none;  }
a:hover, a:active {    background-color: #7A991A;  }
</style> </head> <body>
<a href="default.asp" target="_blank">This is a link</a>
</body> </html>
```

3. Descreva o que faz o código a seguir

```
<!DOCTYPE html> <html> <head> <style>
table, td, th { border: 1px solid green; }
th { background-color: green; color: white; }
</style> </head> <body> <table>  <tr>
<th>Firstname</th>  <th>Lastname</th> <th>Savings</th> </tr>
<tr><td>Peter</td><td>Griffin</td><td>$100</td></tr>
<tr><td>Lois</td><td>Griffin</td><td>$150</td></tr>
<tr><td>Cleveland</td><td>Brown</td><td>$250</td></tr>
</table> </body> </html>
```

.

106-76106 - ga/ a

U Positivo - UTFPR - PUCPr – 03/05/2024 - 17:27:26.0
Prof Dr P Kantek (pkantek@gmail.com)
Tutoriais HTML/CSS/JS VIVXh52a, V: 1.05
76113 NICOLE DA SILVA DE AZEVEDO
24FRO106 - 11 entregar ate 23/maio/24 _____ / / _____

CSS 2

Para aplicar estilos a links, pode-se usar qualquer propriedade CSS (color, font-family, background etc). Além disso, os links podem ser estilizados dependendo do estado em que estão. Existem 4 estados

a:link	link ainda não visitado
a:visited	link já visitado
a:hover	quando o mouse passa sobre o link
a:active	link no momento em que é clicado

Veja no exemplo:

```
a:link { color: #FF0000; }
a:visited { color: #00FF00; }
a:hover { color: #FF00FF; }
a:active { color: #0000FF; }
```

Exercício 1

Aproveite os arquivos brasil.html e brasil.css e garanta que o html tenha pelo menos 5 links. Aplique no arquivo brasil.css os links acima descritos com as cores aqua, violet, sienna e chartreuse pela ordem. Acrescente nos 4 o amarelo como cor de fundo (`background-color:yellow`) Veja o funcionamento disso tudo num browser.

Para aplicar estilos a listas, pode-se usar a propriedade `list-style-type:` que pode ser circle, square, upper-roman, lower-alpha. Pode-se usar também `list-style-image` que indica qual imagem usar como o marcador de itens em listas não ordenadas. Eis o formato:

```
ul { list-style-image: url('sqpurple.gif');}
```

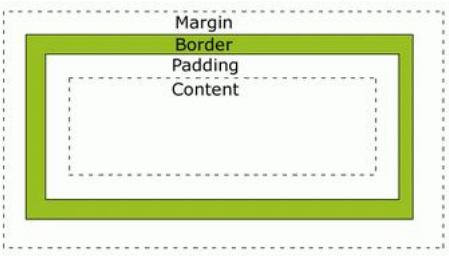
Para as tabelas, as propriedades são: `border:`, `border-collapse:`, `width:`, `height:`, `text-align:`, `vertical-align:`, `padding:`, `color:` e `background-color:`. Veja um exemplo de algumas dessas opções:

```
<style>
table, td, th { border: 1px solid green; }
th { background-color: green; color: white;}
</style>
...
<table> <tr> <th>Nome</th><th>Cidade</th>
<th>Contribuição</th> </tr>
<tr> <td>Pedro</td> <td>Curitiba</td>
<td>100</td>   </tr>
<tr> <td>Maria</td> <td>Joinville</td>
<td>300</td>   </tr>  </table>
```

Exercício 2

Defina arquivos tabela.html e tabela.css e escreva o exemplo acima. Sobre ele, aplique pelo menos 3 propriedades de tabelas diferentes e veja em cada caso o resultado em um browser.

Agora vamos estudar o modelo caixa do CSS (*CSS box model*). Todos os elementos HTML podem ser considerados como uma caixa. Este conceito é muito importante quando se pensa em layout. Pense no modelo da caixa como uma cebola quadrada composta de 4 camadas. De fora para dentro tem-se: margin, border, padding e finalmente o conteúdo que é o elemento HTML. Veja o desenho a seguir para entender o conceito



Eis um exemplo disso tudo junto

```
div { width: 300px;  padding: 25px;
   border: 25px solid navy; margin: 25px;}
```

Note que quando você define largura e altura de alguma coisa, essas medidas se referem à área de conteúdo. Para saber o tamanho total do elementos as outras medidas (se existentes) devem ser somadas. Por exemplo, o elemento abaixo ocupa 350px:

```
div { width: 320px;  padding: 10px;
    border: 5px solid gray; margin: 0;}
```

As propriedades da borda são: `border-style:` com os valores none, dotted, dashed, solid, double, groove, ridge, inset e outset. Todas as demais propriedades só atuam se a propriedade border-style for assinalada.

Depois vem `border-width:`, `border-color:` etc. Para definir a borda de maneira abreviada, a sequência é `border-width:`, `border-style:` (obrigatória), `border-color:`.

A seguir existem diversas propriedades para tratar o outline, elas são muito parecidas com as propriedades da borda e ficam como dever de casa para serem pesquisadas.

A margem clareia completamente o entorno de um elemento (fora da borda). A margem não tem cor de background e sempre é completamente transparente. As margens superior, direita, inferior e esquerda podem ser estabelecidas separadamente, ou pode-se usar a maneira abreviada e fazer tudo junto. Os valores possíveis de `margin:` são: auto (o browser calcula), tamanho (medida em px, pt, cm, ... - default é 0px), % medida em percentagem do tamanho do elemento contido e inherit quando a margem é herdada do elemento pai. Note que em tamanho pode-se usar valores negativos para sobrepujar algum conteúdo. As margens podem ser especificadas como `margin-top:`, `margin-bottom:`,`margin-right:` e `margin-left:` ou se preferir escrever apenas `margin:` e depois 4, 3 2 ou 1 valor de tamanho. A regra é: 3 tamanhos → top,laterais,bottom; 2 tamanhos → cima-baixo, laterais, e 1 tamanho → os 4 iguais.

O padding limpa o entorno do elemento (dentro da borda). Ele é afetado pela cor de fundo do elemento. De maneira muito parecida à margem, ele pode ser estabelecido com 4 propriedades ou de maneira abreviada com uma única propriedade.

Os elementos HTML também podem ter suas dimensões estabelecidas pelas propriedades: `height:`, `max-height:`, `min-height:`, `width:`, `max-width:` e `min-width:`. Veja um exemplo:

```
p { max-width: 20%;
    background-color: yellow;  }
```

A propriedade `display:` determina se e como um elemento é mostrado e a propriedade `visibility` especifica se um elemento deve ser visível ou escondido. Um elemento pode ser apagado fazendo `display: none` ou `visibility:hidden`. Este último apaga o elemento mas preserva o espaço ocupado por ele (em branco). Ou seja o layout permanece inalterado. Já `display: none` apaga o elemento e consome com o espaço ocupado por ele. O lay-out é refeito.

Exercício 3

Escreva um html chamado image.html e um css chamado image.css. Fale sobre o litoral paranaense. Apresente 3 fotos de mesma dimensão. Use 3 classes diferentes. No CSS alterne as propriedades vistas acima. Veja o resultado no browser.

Relembrando elementos em bloco (como `<h1>`, `<p>`, `<li>` e `<div>` por exemplo) são aqueles que forçam quebra de linha antes e depois dele. Já os elementos inline não quebram linha nem antes nem depois. Para transformar um elemento inline em bloco e vice-versa, usa-se a propriedade `display:`. Para inline o valor é `inline` e para bloco o valor é `block`. Veja no exemplo como transformar itens de lista em elementos inline (todos os itens serão mostrados na mesma linha).

```
li { display: inline; }
```

As propriedades de posicionamento permitem posicionar um elemento. Também permitem por um na frente do outro e especificam o que acontece quando o conteúdo de um elemento é muito grande. A primeira propriedade é `position:`. Ela tem 4 valores:

static é o valor default. O elemento é posicionado de acordo com o fluxo normal de preenchimento do browser. O elemento não é afetado pelas propriedades `top`, `bottom`, `left` e `right`.

fixed posição fixa em relação à janela do browser. Ele não é movido mesmo que a janela seja *scrolled*. Veja um exemplo
`p.fixo { position: fixed; top: 30px; right: 5px;}`
Neste caso o elemento é removido do fluxo normal. Os outros elementos são posicionados como se o elemento fixo não existisse.

relative modifica a posição em relação à sua posição normal. Aqui por exemplo, os cabeçalhos h2, serão trazidos para fora da janela:
`h2 { position: relative; left: -20px;}`

absolute posiciona relativo ao primeiro elemento pai que tenha outra posição que não static. Se não existir, será o `<html>`.

Quando elementos são posicionados fora do fluxo normal eles podem sobrepujar outros elementos. A propriedade `z-index:` indica qual a ordem de apresentação. (podem ser atribuídos valores positivos ou negativos). A regra é que um número maior sempre vai aparecer na frente de um número menor. Veja o exemplo:

```
<style>
img { position: absolute; left: 0px;
    top: 0px; z-index: -2; }
</style> </head> <body>
<h1>Um cabeçalho</h1>
<img src="qualquer.gif" width="100" height="140">
</body>
```

Exercício 4

Copie o exemplo acima, usando uma figura qualquer. Varie a propriedade `z-index` e veja o que acontece.

☞ Para você fazer

1. Imprima os arquivos pedidos (brasil htlm e css, tabela htlm e css e image html e css) grampeie-os nesta folha e devolva tudo ou então gere-os no micro na sala de aula e mostre-os ao professor, durante a aula.

2. Descreva o que faz o código a seguir

```
<!DOCTYPE html> <html> <head> <style>
div {background-color: lightblue;
 width: 200px; padding: 25px;
 border: 25px solid navy; }
</style> </head> <body>
<div>Lorem ipsum dolor sit
amet, consectetur adipiscing
elit, sed.</div>
</body> </html>
```

3. Descreva o que faz o código a seguir

```
<!DOCTYPE html> <html> <head> <style>
.floating-box {float: left; width: 150px;
 height: 75px; margin: 10px; border: 3px solid #8AC007;}
.after-box {clear: left; border: 3px solid red;}
</style> </head> <body>
<h2>The Old Way - using float</h2>
<div class="floating-box">Floating box</div>
<div class="floating-box">Floating box</div>
<div class="floating-box">Floating box</div>
<div class="after-box">Another box, after...</div>
</body> </html>
```

.

106-76113 - ga/ a

U Positivo - UTFPR - PUCPr – 03/05/2024 - 17:27:26.0
Prof Dr P Kantek (pkantek@gmail.com)
Tutoriais HTML/CSS/JS VIVXh52a, V: 1.05
76120 PAULO JOSE FREIRE CORREA
24FRO106 - 12 entregar ate 23/maio/24 _____ / / _____

CSS 2

Para aplicar estilos a links, pode-se usar qualquer propriedade CSS (color, font-family, background etc). Além disso, os links podem ser estilizados dependendo do estado em que estão. Existem 4 estados

a:link	link ainda não visitado
a:visited	link já visitado
a:hover	quando o mouse passa sobre o link
a:active	link no momento em que é clicado

Veja no exemplo:

```
a:link { color: #FF0000; }
a:visited { color: #00FF00; }
a:hover { color: #FF00FF; }
a:active { color: #0000FF; }
```

Exercício 1

Aproveite os arquivos brasil.html e brasil.css e garanta que o html tenha pelo menos 5 links. Aplique no arquivo brasil.css os links acima descritos com as cores aqua, violet, sienna e chartreuse pela ordem. Acrescente nos 4 o amarelo como cor de fundo (`background-color:yellow`) Veja o funcionamento disso tudo num browser.

Para aplicar estilos a listas, pode-se usar a propriedade `list-style-type:` que pode ser circle, square, upper-roman, lower-alpha. Pode-se usar também `list-style-image` que indica qual imagem usar como o marcador de itens em listas não ordenadas. Eis o formato:

```
ul { list-style-image: url('sqpurple.gif');}
```

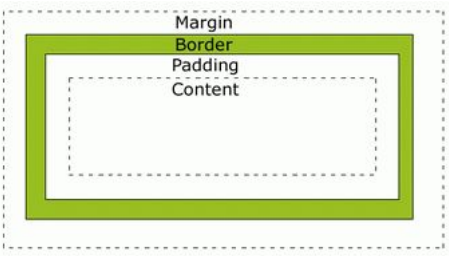
Para as tabelas, as propriedades são: `border:`, `border-collapse:`, `width:`, `height:`, `text-align:`, `vertical-align:`, `padding:`, `color:` e `background-color:`. Veja um exemplo de algumas dessas opções:

```
<style>
table, td, th { border: 1px solid green; }
th { background-color: green; color: white;}
</style>
...
<table> <tr> <th>Nome</th><th>Cidade</th>
<th>Contribuição</th> </tr>
<tr> <td>Pedro</td> <td>Curitiba</td>
<td>100</td>   </tr>
<tr> <td>Maria</td> <td>Joinville</td>
<td>300</td>   </tr>  </table>
```

Exercício 2

Defina arquivos tabela.html e tabela.css e escreva o exemplo acima. Sobre ele, aplique pelo menos 3 propriedades de tabelas diferentes e veja em cada caso o resultado em um browser.

Agora vamos estudar o modelo caixa do CSS (*CSS box model*). Todos os elementos HTML podem ser considerados como uma caixa. Este conceito é muito importante quando se pensa em layout. Pense no modelo da caixa como uma cebola quadrada composta de 4 camadas. De fora para dentro tem-se: margin, border, padding e finalmente o conteúdo que é o elemento HTML. Veja o desenho a seguir para entender o conceito



Eis um exemplo disso tudo junto

```
div { width: 300px;  padding: 25px;
   border: 25px solid navy; margin: 25px;}
```

Note que quando você define largura e altura de alguma coisa, essas medidas se referem à área de conteúdo. Para saber o tamanho total do elementos as outras medidas (se existentes) devem ser somadas. Por exemplo, o elemento abaixo ocupa 350px:

```
div { width: 320px;  padding: 10px;
    border: 5px solid gray; margin: 0;}
```

As propriedades da borda são: `border-style:` com os valores none, dotted, dashed, solid, double, groove, ridge, inset e outset. Todas as demais propriedades só atuam se a propriedade border-style for assinalada.

Depois vem `border-width:`, `border-color:` etc. Para definir a borda de maneira abreviada, a sequência é `border-width:`, `border-style:` (obrigatória), `border-color:`.

A seguir existem diversas propriedades para tratar o outline, elas são muito parecidas com as propriedades da borda e ficam como dever de casa para serem pesquisadas.

A margem clareia completamente o entorno de um elemento (fora da borda). A margem não tem cor de background e sempre é completamente transparente. As margens superior, direita, inferior e esquerda podem ser estabelecidas separadamente, ou pode-se usar a maneira abreviada e fazer tudo junto. Os valores possíveis de `margin:` são: auto (o browser calcula), tamanho (medida em px, pt, cm, ... - default é 0px), % medida em percentagem do tamanho do elemento contido e inherit quando a margem é herdada do elemento pai. Note que em tamanho pode-se usar valores negativos para sobrepujar algum conteúdo. As margens podem ser especificadas como `margin-top:`, `margin-bottom:`,`margin-right:` e `margin-left:` ou se preferir escrever apenas `margin:` e depois 4, 3 2 ou 1 valor de tamanho. A regra é: 3 tamanhos → top,laterais,bottom; 2 tamanhos → cima-baixo, laterais, e 1 tamanho → os 4 iguais.

O padding limpa o entorno do elemento (dentro da borda). Ele é afetado pela cor de fundo do elemento. De maneira muito parecida à margem, ele pode ser estabelecido com 4 propriedades ou de maneira abreviada com uma única propriedade.

Os elementos HTML também podem ter suas dimensões estabelecidas pelas propriedades: `height:`, `max-height:`, `min-height:`, `width:`, `max-width:` e `min-width:`. Veja um exemplo:

```
p { max-width: 20%;
    background-color: yellow;  }
```

A propriedade `display:` determina se e como um elemento é mostrado e a propriedade `visibility` especifica se um elemento deve ser visível ou escondido. Um elemento pode ser apagado fazendo `display: none` ou `visibility:hidden`. Este último apaga o elemento mas preserva o espaço ocupado por ele (em branco). Ou seja o layout permanece inalterado. Já `display: none` apaga o elemento e consome com o espaço ocupado por ele. O lay-out é refeito.

Exercício 3

Escreva um html chamado image.html e um css chamado image.css. Fale sobre o litoral paranaense. Apresente 3 fotos de mesma dimensão. Use 3 classes diferentes. No CSS alterne as propriedades vistas acima. Veja o resultado no browser.

Relembrando elementos em bloco (como `<h1>`, `<p>`, `<li>` e `<div>` por exemplo) são aqueles que forçam quebra de linha antes e depois dele. Já os elementos inline não quebram linha nem antes nem depois. Para transformar um elemento inline em bloco e vice-versa, usa-se a propriedade `display:`. Para inline o valor é `inline` e para bloco o valor é `block`. Veja no exemplo como transformar itens de lista em elementos inline (todos os itens serão mostrados na mesma linha).

```
li { display: inline; }
```

As propriedades de posicionamento permitem posicionar um elemento. Também permitem por um na frente do outro e especificam o que acontece quando o conteúdo de um elemento é muito grande. A primeira propriedade é `position:`. Ela tem 4 valores:

static é o valor default. O elemento é posicionado de acordo com o fluxo normal de preenchimento do browser. O elemento não é afetado pelas propriedades `top`, `bottom`, `left` e `right`.

fixed posição fixa em relação à janela do browser. Ele não é movido mesmo que a janela seja *scrolled*. Veja um exemplo
`p.fixo { position: fixed; top: 30px; right: 5px;}`
Neste caso o elemento é removido do fluxo normal. Os outros elementos são posicionados como se o elemento fixo não existisse.

relative modifica a posição em relação à sua posição normal. Aqui por exemplo, os cabeçalhos h2, serão trazidos para fora da janela:
`h2 { position: relative; left: -20px;}`

absolute posiciona relativo ao primeiro elemento pai que tenha outra posição que não static. Se não existir, será o `<html>`.

Quando elementos são posicionados fora do fluxo normal eles podem sobrepujar outros elementos. A propriedade `z-index:` indica qual a ordem de apresentação. (podem ser atribuídos valores positivos ou negativos). A regra é que um número maior sempre vai aparecer na frente de um número menor. Veja o exemplo:

```
<style>
img { position: absolute; left: 0px;
    top: 0px; z-index: -2; }
</style> </head> <body>
<h1>Um cabeçalho</h1>
<img src="qualquer.gif" width="100" height="140">
</body>
```

Exercício 4

Copie o exemplo acima, usando uma figura qualquer. Varie a propriedade `z-index` e veja o que acontece.

☞ Para você fazer

1. Imprima os arquivos pedidos (brasil htlm e css, tabela htlm e css e image html e css) grampeie-os nesta folha e devolva tudo ou então gere-os no micro na sala de aula e mostre-os ao professor, durante a aula.

2. Descreva o que faz o código a seguir

```
<!DOCTYPE html> <html> <head> <style>
.floating-box {float: left; width: 150px;
 height: 75px; margin: 10px; border: 3px solid #8AC007;}
.after-box {clear: left; border: 3px solid red;}
</style> </head> <body>
<h2>The Old Way - using float</h2>
<div class="floating-box">Floating box</div>
<div class="floating-box">Floating box</div>
<div class="floating-box">Floating box</div>
<div class="after-box">Another box, after...</div>
</body> </html>
```

3. Descreva o que faz o código a seguir

```
<!DOCTYPE html> <html> <head> <style>
div {background-color: lightblue;width: 200px;}
</style> </head> <body>
<div>Lorem ipsum dolor
adipiscing elit, sed
ut labore et dolore. </div>
</body> </html>
```

.

106-76120 - ga/ a

U Positivo - UTFPR - PUCPr – 03/05/2024 - 17:27:26.0
Prof Dr P Kantek (pkantek@gmail.com)
Tutoriais HTML/CSS/JS VIVXh52a, V: 1.05
76137 PEDRO HENRIQUE PIEKARSKI DE 24FRO106 - 13 entregar ate 23/maio/24 _____ / / _____

CSS 2

Para aplicar estilos a links, pode-se usar qualquer propriedade CSS (color, font-family, background etc). Além disso, os links podem ser estilizados dependendo do estado em que estão. Existem 4 estados

a:link	link ainda não visitado
a:visited	link já visitado
a:hover	quando o mouse passa sobre o link
a:active	link no momento em que é clicado

Veja no exemplo:

```
a:link { color: #FF0000; }
a:visited { color: #00FF00; }
a:hover { color: #FF00FF; }
a:active { color: #0000FF; }
```

Exercício 1

Aproveite os arquivos brasil.html e brasil.css e garanta que o html tenha pelo menos 5 links. Aplique no arquivo brasil.css os links acima descritos com as cores aqua, violet, sienna e chartreuse pela ordem. Acrescente nos 4 o amarelo como cor de fundo (`background-color:yellow`) Veja o funcionamento disso tudo num browser.

Para aplicar estilos a listas, pode-se usar a propriedade `list-style-type:` que pode ser circle, square, upper-roman, lower-alpha. Pode-se usar também `list-style-image` que indica qual imagem usar como o marcador de itens em listas não ordenadas. Eis o formato:

```
ul { list-style-image: url('sqpurple.gif');}
```

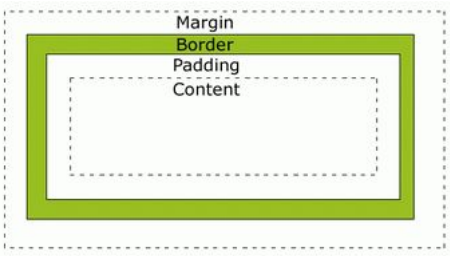
Para as tabelas, as propriedades são: `border:`, `border-collapse:`, `width:`, `height:`, `text-align:`, `vertical-align:`, `padding:`, `color:` e `background-color:`. Veja um exemplo de algumas dessas opções:

```
<style>
table, td, th { border: 1px solid green; }
th { background-color: green; color: white;}
</style>
...
<table> <tr> <th>Nome</th><th>Cidade</th>
<th>Contribuição</th> </tr>
<tr> <td>Pedro</td> <td>Curitiba</td>
<td>100</td>   </tr>
<tr> <td>Maria</td> <td>Joinville</td>
<td>300</td>   </tr>  </table>
```

Exercício 2

Defina arquivos tabela.html e tabela.css e escreva o exemplo acima. Sobre ele, aplique pelo menos 3 propriedades de tabelas diferentes e veja em cada caso o resultado em um browser.

Agora vamos estudar o modelo caixa do CSS (*CSS box model*). Todos os elementos HTML podem ser considerados como uma caixa. Este conceito é muito importante quando se pensa em layout. Pense no modelo da caixa como uma cebola quadrada composta de 4 camadas. De fora para dentro tem-se: margin, border, padding e finalmente o conteúdo que é o elemento HTML. Veja o desenho a seguir para entender o conceito



Eis um exemplo disso tudo junto

```
div { width: 300px;  padding: 25px;
   border: 25px solid navy; margin: 25px;}
```

Note que quando você define largura e altura de alguma coisa, essas medidas se referem à área de conteúdo. Para saber o tamanho total do elementos as outras medidas (se existentes) devem ser somadas. Por exemplo, o elemento abaixo ocupa 350px:

```
div { width: 320px;  padding: 10px;
    border: 5px solid gray; margin: 0;}
```

As propriedades da borda são: `border-style:` com os valores none, dotted, dashed, solid, double, groove, ridge, inset e outset. Todas as demais propriedades só atuam se a propriedade border-style for assinalada.

Depois vem `border-width:`, `border-color:` etc. Para definir a borda de maneira abreviada, a sequência é `border-width:`, `border-style:` (obrigatória), `border-color:`.

A seguir existem diversas propriedades para tratar o outline, elas são muito parecidas com as propriedades da borda e ficam como dever de casa para serem pesquisadas.

A margem clareia completamente o entorno de um elemento (fora da borda). A margem não tem cor de background e sempre é completamente transparente. As margens superior, direita, inferior e esquerda podem ser estabelecidas separadamente, ou pode-se usar a maneira abreviada e fazer tudo junto. Os valores possíveis de `margin:` são: auto (o browser calcula), tamanho (medida em px, pt, cm, ... - default é 0px), % medida em percentagem do tamanho do elemento contido e inherit quando a margem é herdada do elemento pai. Note que em tamanho pode-se usar valores negativos para sobrepujar algum conteúdo. As margens podem ser especificadas como `margin-top:`, `margin-bottom:`,`margin-right:` e `margin-left:` ou se preferir escrever apenas `margin:` e depois 4, 3 2 ou 1 valor de tamanho. A regra é: 3 tamanhos → top,laterais,bottom; 2 tamanhos → cima-baixo, laterais, e 1 tamanho → os 4 iguais.

O padding limpa o entorno do elemento (dentro da borda). Ele é afetado pela cor de fundo do elemento. De maneira muito parecida à margem, ele pode ser estabelecido com 4 propriedades ou de maneira abreviada com uma única propriedade.

Os elementos HTML também podem ter suas dimensões estabelecidas pelas propriedades: `height:`, `max-height:`, `min-height:`, `width:`, `max-width:` e `min-width:`. Veja um exemplo:

```
p { max-width: 20%;
    background-color: yellow;  }
```

A propriedade `display:` determina se e como um elemento é mostrado e a propriedade `visibility` especifica se um elemento deve ser visível ou escondido. Um elemento pode ser apagado fazendo `display: none` ou `visibility:hidden`. Este último apaga o elemento mas preserva o espaço ocupado por ele (em branco). Ou seja o layout permanece inalterado. Já `display: none` apaga o elemento e consome com o espaço ocupado por ele. O lay-out é refeito.

Exercício 3

Escreva um html chamado image.html e um css chamado image.css. Fale sobre o litoral paranaense. Apresente 3 fotos de mesma dimensão. Use 3 classes diferentes. No CSS alterne as propriedades vistas acima. Veja o resultado no browser.

Relembrando elementos em bloco (como `<h1>`, `<p>`, `<li>` e `<div>` por exemplo) são aqueles que forçam quebra de linha antes e depois dele. Já os elementos inline não quebram linha nem antes nem depois. Para transformar um elemento inline em bloco e vice-versa, usa-se a propriedade `display:`. Para inline o valor é `inline` e para bloco o valor é `block`. Veja no exemplo como transformar itens de lista em elementos inline (todos os itens serão mostrados na mesma linha).

```
li { display: inline; }
```

As propriedades de posicionamento permitem posicionar um elemento. Também permitem por um na frente do outro e especificam o que acontece quando o conteúdo de um elemento é muito grande. A primeira propriedade é `position:`. Ela tem 4 valores:

static é o valor default. O elemento é posicionado de acordo com o fluxo normal de preenchimento do browser. O elemento não é afetado pelas propriedades `top`, `bottom`, `left` e `right`.

fixed posição fixa em relação à janela do browser. Ele não é movido mesmo que a janela seja *scrolled*. Veja um exemplo
`p.fixo { position: fixed; top: 30px; right: 5px;}`
Neste caso o elemento é removido do fluxo normal. Os outros elementos são posicionados como se o elemento fixo não existisse.

relative modifica a posição em relação à sua posição normal. Aqui por exemplo, os cabeçalhos h2, serão trazidos para fora da janela:
`h2 { position: relative; left: -20px;}`

absolute posiciona relativo ao primeiro elemento pai que tenha outra posição que não static. Se não existir, será o `<html>`.

Quando elementos são posicionados fora do fluxo normal eles podem sobrepujar outros elementos. A propriedade `z-index:` indica qual a ordem de apresentação. (podem ser atribuídos valores positivos ou negativos). A regra é que um número maior sempre vai aparecer na frente de um número menor. Veja o exemplo:

```
<style>
img { position: absolute; left: 0px;
    top: 0px; z-index: -2; }
</style> </head> <body>
<h1>Um cabeçalho</h1>
<img src="qualquer.gif" width="100" height="140">
</body>
```

Exercício 4

Copie o exemplo acima, usando uma figura qualquer. Varie a propriedade `z-index` e veja o que acontece.

☞ Para você fazer

1. Imprima os arquivos pedidos (brasil htlm e css, tabela htlm e css e image html e css) grampeie-os nesta folha e devolva tudo ou então gere-os no micro na sala de aula e mostre-os ao professor, durante a aula.

2. Descreva o que faz o código a seguir

```
<!DOCTYPE html> <html> <head> <style>
div {background-color: lightblue;width: 200px;}
</style> </head> <body>
<div>Lorem ipsum dolor
adipiscing elit, sed
ut labore et dolore. </div>
</body> </html>
```

3. Descreva o que faz o código a seguir

```
<!DOCTYPE html> <html> <head> <style>
a.five:link {color:#ff0000;text-decoration:none;}
a.five:visited {color:#0000ff;text-decoration:none;}
a.five:hover {text-decoration:underline;}
</style> </head> <body>
<p>Mouse over the links and watch them change layout:</p>
<p><b><a class="five" href="default.asp"
target="_blank">This link changes text-decoration</a></b></p>
</body> </html>
```

.

106-76137 - ga/ a

U Positivo - UTFPR - PUCPr – 03/05/2024 - 17:27:26.0
Prof Dr P Kantek (pkantek@gmail.com)
Tutoriais HTML/CSS/JS VIVXh52a, V: 1.05
76144 RAUL KNAPKI CUNHA 24FRO106 - 14
entregar ate 23/maio/24 _____ / _____ / _____

CSS 2

Para aplicar estilos a links, pode-se usar qualquer propriedade CSS (color, font-family, background etc). Além disso, os links podem ser estilizados dependendo do estado em que estão. Existem 4 estados

a:link	link ainda não visitado
a:visited	link já visitado
a:hover	quando o mouse passa sobre o link
a:active	link no momento em que é clicado

Veja no exemplo:

```
a:link { color: #FF0000; }
a:visited { color: #00FF00; }
a:hover { color: #FF00FF; }
a:active { color: #0000FF; }
```

Exercício 1

Aproveite os arquivos brasil.html e brasil.css e garanta que o html tenha pelo menos 5 links. Aplique no arquivo brasil.css os links acima descritos com as cores aqua, violet, sienna e chartreuse pela ordem. Acrescente nos 4 o amarelo como cor de fundo (`background-color:yellow`) Veja o funcionamento disso tudo num browser.

Para aplicar estilos a listas, pode-se usar a propriedade `list-style-type:` que pode ser circle, square, upper-roman, lower-alpha. Pode-se usar também `list-style-image` que indica qual imagem usar como o marcador de itens em listas não ordenadas. Eis o formato:

```
ul { list-style-image: url('sqpurple.gif');}
```

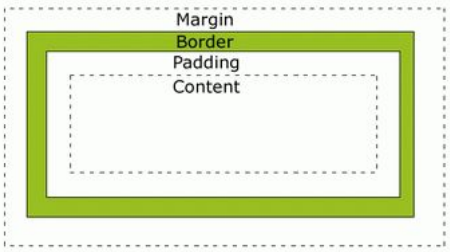
Para as tabelas, as propriedades são: `border:`, `border-collapse:`, `width:`, `height:`, `text-align:`, `vertical-align:`, `padding:`, `color:` e `background-color:`. Veja um exemplo de algumas dessas opções:

```
<style>
table, td, th { border: 1px solid green; }
th { background-color: green; color: white;}
</style>
...
<table> <tr> <th>Nome</th><th>Cidade</th>
<th>Contribuição</th> </tr>
<tr> <td>Pedro</td> <td>Curitiba</td>
<td>100</td>   </tr>
<tr> <td>Maria</td> <td>Joinville</td>
<td>300</td>   </tr>  </table>
```

Exercício 2

Defina arquivos tabela.html e tabela.css e escreva o exemplo acima. Sobre ele, aplique pelo menos 3 propriedades de tabelas diferentes e veja em cada caso o resultado em um browser.

Agora vamos estudar o modelo caixa do CSS (*CSS box model*). Todos os elementos HTML podem ser considerados como uma caixa. Este conceito é muito importante quando se pensa em layout. Pense no modelo da caixa como uma cebola quadrada composta de 4 camadas. De fora para dentro tem-se: margin, border, padding e finalmente o conteúdo que é o elemento HTML. Veja o desenho a seguir para entender o conceito



Eis um exemplo disso tudo junto

```
div { width: 300px;  padding: 25px;
   border: 25px solid navy; margin: 25px;}
```

Note que quando você define largura e altura de alguma coisa, essas medidas se referem à área de conteúdo. Para saber o tamanho total do elementos as outras medidas (se existentes) devem ser somadas. Por exemplo, o elemento abaixo ocupa 350px:

```
div { width: 320px;  padding: 10px;
    border: 5px solid gray; margin: 0;}
```

As propriedades da borda são: `border-style:` com os valores none, dotted, dashed, solid, double, groove, ridge, inset e outset. Todas as demais propriedades só atuam se a propriedade border-style for assinalada.

Depois vem `border-width:`, `border-color:` etc. Para definir a borda de maneira abreviada, a sequência é `border-width:`, `border-style:` (obrigatória), `border-color:`.

A seguir existem diversas propriedades para tratar o outline, elas são muito parecidas com as propriedades da borda e ficam como dever de casa para serem pesquisadas.

A margem clareia completamente o entorno de um elemento (fora da borda). A margem não tem cor de background e sempre é completamente transparente. As margens superior, direita, inferior e esquerda podem ser estabelecidas separadamente, ou pode-se usar a maneira abreviada e fazer tudo junto. Os valores possíveis de `margin:` são: auto (o browser calcula), tamanho (medida em px, pt, cm, ... - default é 0px), % medida em percentagem do tamanho do elemento contido e inherit quando a margem é herdada do elemento pai. Note que em tamanho pode-se usar valores negativos para sobrepujar algum conteúdo. As margens podem ser especificadas como `margin-top:`, `margin-bottom:`,`margin-right:` e `margin-left:` ou se preferir escrever apenas `margin:` e depois 4, 3 2 ou 1 valor de tamanho. A regra é: 3 tamanhos → top,laterais,bottom; 2 tamanhos → cima-baixo, laterais, e 1 tamanho → os 4 iguais.

O padding limpa o entorno do elemento (dentro da borda). Ele é afetado pela cor de fundo do elemento. De maneira muito parecida à margem, ele pode ser estabelecido com 4 propriedades ou de maneira abreviada com uma única propriedade.

Os elementos HTML também podem ter suas dimensões estabelecidas pelas propriedades: `height:`, `max-height:`, `min-height:`, `width:`, `max-width:` e `min-width:`. Veja um exemplo:

```
p { max-width: 20%;
    background-color: yellow;  }
```

A propriedade `display:` determina se e como um elemento é mostrado e a propriedade `visibility` especifica se um elemento deve ser visível ou escondido. Um elemento pode ser apagado fazendo `display: none` ou `visibility:hidden`. Este último apaga o elemento mas preserva o espaço ocupado por ele (em branco). Ou seja o layout permanece inalterado. Já `display: none` apaga o elemento e consome com o espaço ocupado por ele. O lay-out é refeito.

Exercício 3

Escreva um html chamado image.html e um css chamado image.css. Fale sobre o litoral paranaense. Apresente 3 fotos de mesma dimensão. Use 3 classes diferentes. No CSS alterne as propriedades vistas acima. Veja o resultado no browser.

Relembrando elementos em bloco (como `<h1>`, `<p>`, `<li>` e `<div>` por exemplo) são aqueles que forçam quebra de linha antes e depois dele. Já os elementos inline não quebram linha nem antes nem depois. Para transformar um elemento inline em bloco e vice-versa, usa-se a propriedade `display:`. Para inline o valor é `inline` e para bloco o valor é `block`. Veja no exemplo como transformar itens de lista em elementos inline (todos os itens serão mostrados na mesma linha).

```
li { display: inline; }
```

As propriedades de posicionamento permitem posicionar um elemento. Também permitem por um na frente do outro e especificam o que acontece quando o conteúdo de um elemento é muito grande. A primeira propriedade é `position:`. Ela tem 4 valores:

static é o valor default. O elemento é posicionado de acordo com o fluxo normal de preenchimento do browser. O elemento não é afetado pelas propriedades `top`, `bottom`, `left` e `right`.

fixed posição fixa em relação à janela do browser. Ele não é movido mesmo que a janela seja *scrolled*. Veja um exemplo
`p.fixo { position: fixed;`
`top: 30px; right: 5px;}`
Neste caso o elemento é removido do fluxo normal. Os outros elementos são posicionados como se o elemento fixo não existisse.

relative modifica a posição em relação à sua posição normal. Aqui por exemplo, os cabeçalhos h2, serão trazidos para fora da janela:
`h2 { position: relative; left: -20px;}`

absolute posiciona relativo ao primeiro elemento pai que tenha outra posição que não static. Se não existir, será o `<html>`.

Quando elementos são posicionados fora do fluxo normal eles podem sobrepujar outros elementos. A propriedade `z-index:` indica qual a ordem de apresentação. (podem ser atribuídos valores positivos ou negativos). A regra é que um número maior sempre vai aparecer na frente de um número menor. Veja o exemplo:

```
<style>
img { position: absolute; left: 0px;
    top: 0px; z-index: -2; }
</style> </head> <body>
<h1>Um cabeçalho</h1>
<img src="qualquer.gif" width="100" height="140">
</body>
```

Exercício 4

Copie o exemplo acima, usando uma figura qualquer. Varie a propriedade `z-index` e veja o que acontece.

☞ Para você fazer

1. Imprima os arquivos pedidos (brasil htlm e css, tabela htlm e css e image html e css) grampeie-os nesta folha e devolva tudo ou então gere-os no micro na sala de aula e mostre-os ao professor, durante a aula.

2. Descreva o que faz o código a seguir

```
<!DOCTYPE html> <html> <head> <style>
div {background-color: lightblue; width: 200px;
 padding: 25px; border: 25px solid navy;
 margin: 25px;}
</style> </head> <body>
<div>Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed.</div>
</body> </html>
```

3. Descreva o que faz o código a seguir

```
<!DOCTYPE html> <html> <head> <style>
table, td, th { border: 1px solid green; }
th { background-color: green; color: white; }
</style> </head> <body> <table>  <tr>
<th>Firstname</th>  <th>Lastname</th> <th>Savings</th> </tr>
<tr><td>Peter</td><td>Griffin</td><td>$100</td></tr>
<tr><td>Lois</td><td>Griffin</td><td>$150</td></tr>
<tr><td>Cleveland</td><td>Brown</td><td>$250</td></tr>
</table> </body> </html>
```

.

106-76144 - ga/ a

U Positivo - UTFPR - PUCPr – 03/05/2024 - 17:27:26.0
Prof Dr P Kantek (pkantek@gmail.com)
Tutoriais HTML/CSS/JS VIVXh52a, V: 1.05
76168 VICTOR BRONDANI DOS SANTOS
24FRO106 - 15 entregar ate 23/maio/24 _____ / / _____

CSS 2

Para aplicar estilos a links, pode-se usar qualquer propriedade CSS (color, font-family, background etc). Além disso, os links podem ser estilizados dependendo do estado em que estão. Existem 4 estados

a:link	link ainda não visitado
a:visited	link já visitado
a:hover	quando o mouse passa sobre o link
a:active	link no momento em que é clicado

Veja no exemplo:

```
a:link { color: #FF0000; }
a:visited { color: #00FF00; }
a:hover { color: #FF00FF; }
a:active { color: #0000FF; }
```

Exercício 1

Aproveite os arquivos brasil.html e brasil.css e garanta que o html tenha pelo menos 5 links. Aplique no arquivo brasil.css os links acima descritos com as cores aqua, violet, sienna e chartreuse pela ordem. Acrescente nos 4 o amarelo como cor de fundo (`background-color:yellow`) Veja o funcionamento disso tudo num browser.

Para aplicar estilos a listas, pode-se usar a propriedade `list-style-type:` que pode ser circle, square, upper-roman, lower-alpha. Pode-se usar também `list-style-image` que indica qual imagem usar como o marcador de itens em listas não ordenadas. Eis o formato:

```
ul { list-style-image: url('sqpurple.gif');}
```

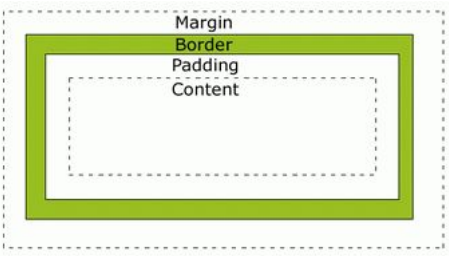
Para as tabelas, as propriedades são: `border:`, `border-collapse:`, `width:`, `height:`, `text-align:`, `vertical-align:`, `padding:`, `color:` e `background-color:`. Veja um exemplo de algumas dessas opções:

```
<style>
table, td, th { border: 1px solid green; }
th { background-color: green; color: white;}
</style>
...
<table> <tr> <th>Nome</th><th>Cidade</th>
<th>Contribuição</th> </tr>
<tr> <td>Pedro</td> <td>Curitiba</td>
<td>100</td>   </tr>
<tr> <td>Maria</td> <td>Joinville</td>
<td>300</td>   </tr>  </table>
```

Exercício 2

Defina arquivos tabela.html e tabela.css e escreva o exemplo acima. Sobre ele, aplique pelo menos 3 propriedades de tabelas diferentes e veja em cada caso o resultado em um browser.

Agora vamos estudar o modelo caixa do CSS (*CSS box model*). Todos os elementos HTML podem ser considerados como uma caixa. Este conceito é muito importante quando se pensa em layout. Pense no modelo da caixa como uma cebola quadrada composta de 4 camadas. De fora para dentro tem-se: margin, border, padding e finalmente o conteúdo que é o elemento HTML. Veja o desenho a seguir para entender o conceito



Eis um exemplo disso tudo junto

```
div { width: 300px;  padding: 25px;
   border: 25px solid navy; margin: 25px;}
```

Note que quando você define largura e altura de alguma coisa, essas medidas se referem à área de conteúdo. Para saber o tamanho total do elementos as outras medidas (se existentes) devem ser somadas. Por exemplo, o elemento abaixo ocupa 350px:

```
div { width: 320px;  padding: 10px;
    border: 5px solid gray; margin: 0;}
```

As propriedades da borda são: `border-style:` com os valores none, dotted, dashed, solid, double, groove, ridge, inset e outset. Todas as demais propriedades só atuam se a propriedade border-style for assinalada.

Depois vem `border-width:`, `border-color:` etc. Para definir a borda de maneira abreviada, a sequência é `border-width:`, `border-style:` (obrigatória), `border-color:`.

A seguir existem diversas propriedades para tratar o outline, elas são muito parecidas com as propriedades da borda e ficam como dever de casa para serem pesquisadas.

A margem clareia completamente o entorno de um elemento (fora da borda). A margem não tem cor de background e sempre é completamente transparente. As margens superior, direita, inferior e esquerda podem ser estabelecidas separadamente, ou pode-se usar a maneira abreviada e fazer tudo junto. Os valores possíveis de `margin:` são: auto (o browser calcula), tamanho (medida em px, pt, cm, ... - default é 0px), % medida em percentagem do tamanho do elemento contido e inherit quando a margem é herdada do elemento pai. Note que em tamanho pode-se usar valores negativos para sobrepujar algum conteúdo. As margens podem ser especificadas como `margin-top:`, `margin-bottom:`,`margin-right:` e `margin-left:` ou se preferir escrever apenas `margin:` e depois 4, 3 2 ou 1 valor de tamanho. A regra é: 3 tamanhos → top,laterais,bottom; 2 tamanhos → cima-baixo, laterais, e 1 tamanho → os 4 iguais.

O padding limpa o entorno do elemento (dentro da borda). Ele é afetado pela cor de fundo do elemento. De maneira muito parecida à margem, ele pode ser estabelecido com 4 propriedades ou de maneira abreviada com uma única propriedade.

Os elementos HTML também podem ter suas dimensões estabelecidas pelas propriedades: `height:`, `max-height:`, `min-height:`, `width:`, `max-width:` e `min-width:`. Veja um exemplo:

```
p { max-width: 20%;
    background-color: yellow;  }
```

A propriedade `display:` determina se e como um elemento é mostrado e a propriedade `visibility` especifica se um elemento deve ser visível ou escondido. Um elemento pode ser apagado fazendo `display: none` ou `visibility:hidden`. Este último apaga o elemento mas preserva o espaço ocupado por ele (em branco). Ou seja o layout permanece inalterado. Já `display: none` apaga o elemento e consome com o espaço ocupado por ele. O lay-out é refeito.

Exercício 3

Escreva um html chamado image.html e um css chamado image.css. Fale sobre o litoral paranaense. Apresente 3 fotos de mesma dimensão. Use 3 classes diferentes. No CSS alterne as propriedades vistas acima. Veja o resultado no browser.

Relembrando elementos em bloco (como `<h1>`, `<p>`, `<li>` e `<div>` por exemplo) são aqueles que forçam quebra de linha antes e depois dele. Já os elementos inline não quebram linha nem antes nem depois. Para transformar um elemento inline em bloco e vice-versa, usa-se a propriedade `display:`. Para inline o valor é `inline` e para bloco o valor é `block`. Veja no exemplo como transformar itens de lista em elementos inline (todos os itens serão mostrados na mesma linha).

```
li { display: inline; }
```

As propriedades de posicionamento permitem posicionar um elemento. Também permitem por um na frente do outro e especificam o que acontece quando o conteúdo de um elemento é muito grande. A primeira propriedade é `position:`. Ela tem 4 valores:

static é o valor default. O elemento é posicionado de acordo com o fluxo normal de preenchimento do browser. O elemento não é afetado pelas propriedades `top`, `bottom`, `left` e `right`.

fixed posição fixa em relação à janela do browser. Ele não é movido mesmo que a janela seja *scrolled*. Veja um exemplo
`p.fixo { position: fixed;`
`top: 30px; right: 5px;}`
Neste caso o elemento é removido do fluxo normal. Os outros elementos são posicionados como se o elemento fixo não existisse.

relative modifica a posição em relação à sua posição normal. Aqui por exemplo, os cabeçalhos h2, serão trazidos para fora da janela:
`h2 { position: relative; left: -20px;}`

absolute posiciona relativo ao primeiro elemento pai que tenha outra posição que não static. Se não existir, será o `<html>`.

Quando elementos são posicionados fora do fluxo normal eles podem sobrepujar outros elementos. A propriedade `z-index:` indica qual a ordem de apresentação. (podem ser atribuídos valores positivos ou negativos). A regra é que um número maior sempre vai aparecer na frente de um número menor. Veja o exemplo:

```
<style>
img { position: absolute; left: 0px;
    top: 0px; z-index: -2; }
</style> </head> <body>
<h1>Um cabeçalho</h1>
<img src="qualquer.gif" width="100" height="140">
</body>
```

Exercício 4

Copie o exemplo acima, usando uma figura qualquer. Varie a propriedade `z-index` e veja o que acontece.

☞ Para você fazer

1. Imprima os arquivos pedidos (brasil htlm e css, tabela htlm e css e image html e css) grampeie-os nesta folha e devolva tudo ou então gere-os no micro na sala de aula e mostre-os ao professor, durante a aula.

2. Descreva o que faz o código a seguir

```
<!DOCTYPE html> <html> <head> <style>
div {background-color: lightblue;width: 200px;}
</style> </head> <body>
<div>Lorem ipsum dolor
adipiscing elit, sed
ut labore et dolore. </div>
</body> </html>
```

3. Descreva o que faz o código a seguir

```
<!DOCTYPE html> <html> <head> <style>
table, td, th { border: 1px solid green; }
th { background-color: green; color: white; }
</style> </head> <body> <table>  <tr>
<th>Firstname</th>  <th>Lastname</th> <th>Savings</th> </tr>
<tr><td>Peter</td><td>Griffin</td><td>$100</td></tr>
<tr><td>Lois</td><td>Griffin</td><td>$150</td></tr>
<tr><td>Cleveland</td><td>Brown</td><td>$250</td></tr>
</table> </body> </html>
```

.

106-76168 - ga/ a

U Positivo - UTFPR - PUCPr – 03/05/2024 - 17:27:26.0
Prof Dr P Kantek (pkantek@gmail.com)
Tutoriais HTML/CSS/JS VIVXh52a, V: 1.05
76175 VICTOR HUGO DOS SANTOS DE CA
24FRO106 - 16 entregar ate 23/maio/24 _____ / / _____

CSS 2

Para aplicar estilos a links, pode-se usar qualquer propriedade CSS (color, font-family, background etc). Além disso, os links podem ser estilizados dependendo do estado em que estão. Existem 4 estados

a:link	link ainda não visitado
a:visited	link já visitado
a:hover	quando o mouse passa sobre o link
a:active	link no momento em que é clicado

Veja no exemplo:

```
a:link { color: #FF0000; }
a:visited { color: #00FF00; }
a:hover { color: #FF00FF; }
a:active { color: #0000FF; }
```

Exercício 1

Aproveite os arquivos brasil.html e brasil.css e garanta que o html tenha pelo menos 5 links. Aplique no arquivo brasil.css os links acima descritos com as cores aqua, violet, sienna e chartreuse pela ordem. Acrescente nos 4 o amarelo como cor de fundo (`background-color:yellow`) Veja o funcionamento disso tudo num browser.

Para aplicar estilos a listas, pode-se usar a propriedade `list-style-type:` que pode ser circle, square, upper-roman, lower-alpha. Pode-se usar também `list-style-image` que indica qual imagem usar como o marcador de itens em listas não ordenadas. Eis o formato:

```
ul { list-style-image: url('sqpurple.gif');}
```

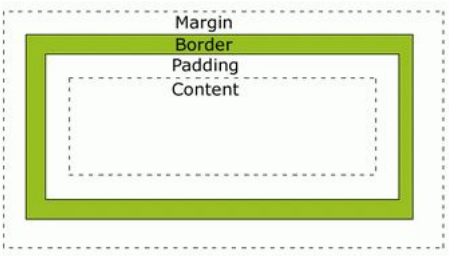
Para as tabelas, as propriedades são: `border:`, `border-collapse:`, `width:`, `height:`, `text-align:`, `vertical-align:`, `padding:`, `color:` e `background-color:`. Veja um exemplo de algumas dessas opções:

```
<style>
table, td, th { border: 1px solid green; }
th { background-color: green; color: white;}
</style>
...
<table> <tr> <th>Nome</th><th>Cidade</th>
<th>Contribuição</th> </tr>
<tr> <td>Pedro</td> <td>Curitiba</td>
<td>100</td>   </tr>
<tr> <td>Maria</td> <td>Joinville</td>
<td>300</td>   </tr>  </table>
```

Exercício 2

Defina arquivos tabela.html e tabela.css e escreva o exemplo acima. Sobre ele, aplique pelo menos 3 propriedades de tabelas diferentes e veja em cada caso o resultado em um browser.

Agora vamos estudar o modelo caixa do CSS (*CSS box model*). Todos os elementos HTML podem ser considerados como uma caixa. Este conceito é muito importante quando se pensa em layout. Pense no modelo da caixa como uma cebola quadrada composta de 4 camadas. De fora para dentro tem-se: margin, border, padding e finalmente o conteúdo que é o elemento HTML. Veja o desenho a seguir para entender o conceito



Eis um exemplo disso tudo junto

```
div { width: 300px;  padding: 25px;
   border: 25px solid navy; margin: 25px;}
```

Note que quando você define largura e altura de alguma coisa, essas medidas se referem à área de conteúdo. Para saber o tamanho total do elementos as outras medidas (se existentes) devem ser somadas. Por exemplo, o elemento abaixo ocupa 350px:

```
div { width: 320px;  padding: 10px;
    border: 5px solid gray; margin: 0;}
```

As propriedades da borda são: `border-style:` com os valores none, dotted, dashed, solid, double, groove, ridge, inset e outset. Todas as demais propriedades só atuam se a propriedade border-style for assinalada.

Depois vem `border-width:`, `border-color:` etc. Para definir a borda de maneira abreviada, a sequência é `border-width:`, `border-style:` (obrigatória), `border-color:`.

A seguir existem diversas propriedades para tratar o outline, elas são muito parecidas com as propriedades da borda e ficam como dever de casa para serem pesquisadas.

A margem clareia completamente o entorno de um elemento (fora da borda). A margem não tem cor de background e sempre é completamente transparente. As margens superior, direita, inferior e esquerda podem ser estabelecidas separadamente, ou pode-se usar a maneira abreviada e fazer tudo junto. Os valores possíveis de `margin:` são: auto (o browser calcula), tamanho (medida em px, pt, cm, ... - default é 0px), % medida em percentagem do tamanho do elemento contido e inherit quando a margem é herdada do elemento pai. Note que em tamanho pode-se usar valores negativos para sobrepujar algum conteúdo. As margens podem ser especificadas como `margin-top:`, `margin-bottom:`,`margin-right:` e `margin-left:` ou se preferir escrever apenas `margin:` e depois 4, 3 2 ou 1 valor de tamanho. A regra é: 3 tamanhos → top,laterais,bottom; 2 tamanhos → cima-baixo, laterais, e 1 tamanho → os 4 iguais.

O padding limpa o entorno do elemento (dentro da borda). Ele é afetado pela cor de fundo do elemento. De maneira muito parecida à margem, ele pode ser estabelecido com 4 propriedades ou de maneira abreviada com uma única propriedade.

Os elementos HTML também podem ter suas dimensões estabelecidas pelas propriedades: `height:`, `max-height:`, `min-height:`, `width:`, `max-width:` e `min-width:`. Veja um exemplo:

```
p { max-width: 20%;
    background-color: yellow;  }
```

A propriedade `display:` determina se e como um elemento é mostrado e a propriedade `visibility` especifica se um elemento deve ser visível ou escondido. Um elemento pode ser apagado fazendo `display: none` ou `visibility:hidden`. Este último apaga o elemento mas preserva o espaço ocupado por ele (em branco). Ou seja o layout permanece inalterado. Já `display: none` apaga o elemento e consome com o espaço ocupado por ele. O lay-out é refeito.

Exercício 3

Escreva um html chamado image.html e um css chamado image.css. Fale sobre o litoral paranaense. Apresente 3 fotos de mesma dimensão. Use 3 classes diferentes. No CSS alterne as propriedades vistas acima. Veja o resultado no browser.

Relembrando elementos em bloco (como `<h1>`, `<p>`, `<li>` e `<div>` por exemplo) são aqueles que forçam quebra de linha antes e depois dele. Já os elementos inline não quebram linha nem antes nem depois. Para transformar um elemento inline em bloco e vice-versa, usa-se a propriedade `display:`. Para inline o valor é `inline` e para bloco o valor é `block`. Veja no exemplo como transformar itens de lista em elementos inline (todos os itens serão mostrados na mesma linha).

```
li { display: inline; }
```

As propriedades de posicionamento permitem posicionar um elemento. Também permitem por um na frente do outro e especificam o que acontece quando o conteúdo de um elemento é muito grande. A primeira propriedade é `position:`. Ela tem 4 valores:

static é o valor default. O elemento é posicionado de acordo com o fluxo normal de preenchimento do browser. O elemento não é afetado pelas propriedades `top`, `bottom`, `left` e `right`.

fixed posição fixa em relação à janela do browser. Ele não é movido mesmo que a janela seja *scrolled*. Veja um exemplo
`p.fixo { position: fixed; top: 30px; right: 5px;}`
Neste caso o elemento é removido do fluxo normal. Os outros elementos são posicionados como se o elemento fixo não existisse.

relative modifica a posição em relação à sua posição normal. Aqui por exemplo, os cabeçalhos h2, serão trazidos para fora da janela:
`h2 { position: relative; left: -20px;}`

absolute posiciona relativo ao primeiro elemento pai que tenha outra posição que não static. Se não existir, será o `<html>`.

Quando elementos são posicionados fora do fluxo normal eles podem sobrepujar outros elementos. A propriedade `z-index:` indica qual a ordem de apresentação. (podem ser atribuídos valores positivos ou negativos). A regra é que um número maior sempre vai aparecer na frente de um número menor. Veja o exemplo:

```
<style>
img { position: absolute; left: 0px;
    top: 0px; z-index: -2; }
</style> </head> <body>
<h1>Um cabeçalho</h1>
<img src="qualquer.gif" width="100" height="140">
</body>
```

Exercício 4

Copie o exemplo acima, usando uma figura qualquer. Varie a propriedade `z-index` e veja o que acontece.

☞ Para você fazer

1. Imprima os arquivos pedidos (brasil htlm e css, tabela htlm e css e image html e css) grampeie-os nesta folha e devolva tudo ou então gere-os no micro na sala de aula e mostre-os ao professor, durante a aula.

2. Descreva o que faz o código a seguir

```
<!DOCTYPE html> <html> <head> <style>
table, td, th {border: 1px solid black; }
td {height: 50px;  vertical-align: bottom; }
</style> </head> <body> <table>  <tr>
<th>Firstname</th> <th>Lastname</th> <th>Savings</th>
</tr> <tr> <td>Peter</td> <td>Griffin</td> <td>$100</td>
</tr> <tr> <td>Lois</td> <td>Griffin</td> <td>$150</td>
</tr> </table> </body> </html>
```

3. Descreva o que faz o código a seguir

```
<!DOCTYPE html> <html> <head> <style>
div {background-color: lightblue;width: 200px;}
</style> </head> <body>
<div>Lorem ipsum dolor
adipiscing elit, sed
ut labore et dolore. </div>
</body> </html>
```

.

106-76175 - ga/ a

U Positivo - UTFPR - PUCPr – 03/05/2024 - 17:27:26.0
Prof Dr P Kantek (pkantek@gmail.com)
Tutoriais HTML/CSS/JS VIVXh52a, V: 1.05
76182 VITOR EDUARDO DE GOES FONTES
24FRO106 - 17 entregar ate 23/maio/24 _____ / / _____

CSS 2

Para aplicar estilos a links, pode-se usar qualquer propriedade CSS (color, font-family, background etc). Além disso, os links podem ser estilizados dependendo do estado em que estão. Existem 4 estados

a:link	link ainda não visitado
a:visited	link já visitado
a:hover	quando o mouse passa sobre o link
a:active	link no momento em que é clicado

Veja no exemplo:

```
a:link { color: #FF0000; }
a:visited { color: #00FF00; }
a:hover { color: #FF00FF; }
a:active { color: #0000FF; }
```

Exercício 1

Aproveite os arquivos brasil.html e brasil.css e garanta que o html tenha pelo menos 5 links. Aplique no arquivo brasil.css os links acima descritos com as cores aqua, violet, sienna e chartreuse pela ordem. Acrescente nos 4 o amarelo como cor de fundo (`background-color:yellow`) Veja o funcionamento disso tudo num browser.

Para aplicar estilos a listas, pode-se usar a propriedade `list-style-type:` que pode ser circle, square, upper-roman, lower-alpha. Pode-se usar também `list-style-image` que indica qual imagem usar como o marcador de itens em listas não ordenadas. Eis o formato:

```
ul { list-style-image: url('sqpurple.gif');}
```

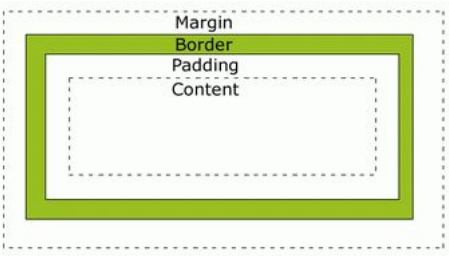
Para as tabelas, as propriedades são: `border:`, `border-collapse:`, `width:`, `height:`, `text-align:`, `vertical-align:`, `padding:`, `color:` e `background-color:`. Veja um exemplo de algumas dessas opções:

```
<style>
table, td, th { border: 1px solid green; }
th { background-color: green; color: white;}
</style>
...
<table> <tr> <th>Nome</th><th>Cidade</th>
<th>Contribuição</th> </tr>
<tr> <td>Pedro</td> <td>Curitiba</td>
<td>100</td>   </tr>
<tr> <td>Maria</td> <td>Joinville</td>
<td>300</td>   </tr>  </table>
```

Exercício 2

Defina arquivos tabela.html e tabela.css e escreva o exemplo acima. Sobre ele, aplique pelo menos 3 propriedades de tabelas diferentes e veja em cada caso o resultado em um browser.

Agora vamos estudar o modelo caixa do CSS (*CSS box model*). Todos os elementos HTML podem ser considerados como uma caixa. Este conceito é muito importante quando se pensa em layout. Pense no modelo da caixa como uma cebola quadrada composta de 4 camadas. De fora para dentro tem-se: margin, border, padding e finalmente o conteúdo que é o elemento HTML. Veja o desenho a seguir para entender o conceito



Eis um exemplo disso tudo junto

```
div { width: 300px;  padding: 25px;
   border: 25px solid navy; margin: 25px;}
```

Note que quando você define largura e altura de alguma coisa, essas medidas se referem à área de conteúdo. Para saber o tamanho total do elementos as outras medidas (se existentes) devem ser somadas. Por exemplo, o elemento abaixo ocupa 350px:

```
div { width: 320px;  padding: 10px;
    border: 5px solid gray; margin: 0;}
```

As propriedades da borda são: `border-style:` com os valores none, dotted, dashed, solid, double, groove, ridge, inset e outset. Todas as demais propriedades só atuam se a propriedade border-style for assinalada.

Depois vem `border-width:`, `border-color:` etc. Para definir a borda de maneira abreviada, a sequência é `border-width:`, `border-style:` (obrigatória), `border-color:`.

A seguir existem diversas propriedades para tratar o outline, elas são muito parecidas com as propriedades da borda e ficam como dever de casa para serem pesquisadas.

A margem clareia completamente o entorno de um elemento (fora da borda). A margem não tem cor de background e sempre é completamente transparente. As margens superior, direita, inferior e esquerda podem ser estabelecidas separadamente, ou pode-se usar a maneira abreviada e fazer tudo junto. Os valores possíveis de `margin:` são: auto (o browser calcula), tamanho (medida em px, pt, cm, ... - default é 0px), % medida em percentagem do tamanho do elemento contido e inherit quando a margem é herdada do elemento pai. Note que em tamanho pode-se usar valores negativos para sobrepujar algum conteúdo. As margens podem ser especificadas como `margin-top:`, `margin-bottom:`,`margin-right:` e `margin-left:` ou se preferir escrever apenas `margin:` e depois 4, 3 2 ou 1 valor de tamanho. A regra é: 3 tamanhos → top,laterais,bottom; 2 tamanhos → cima-baixo, laterais, e 1 tamanho → os 4 iguais.

O padding limpa o entorno do elemento (dentro da borda). Ele é afetado pela cor de fundo do elemento. De maneira muito parecida à margem, ele pode ser estabelecido com 4 propriedades ou de maneira abreviada com uma única propriedade.

Os elementos HTML também podem ter suas dimensões estabelecidas pelas propriedades: `height:`, `max-height:`, `min-height:`, `width:`, `max-width:` e `min-width:`. Veja um exemplo:

```
p { max-width: 20%;
    background-color: yellow;  }
```

A propriedade `display:` determina se e como um elemento é mostrado e a propriedade `visibility` especifica se um elemento deve ser visível ou escondido. Um elemento pode ser apagado fazendo `display: none` ou `visibility:hidden`. Este último apaga o elemento mas preserva o espaço ocupado por ele (em branco). Ou seja o layout permanece inalterado. Já `display: none` apaga o elemento e consome com o espaço ocupado por ele. O lay-out é refeito.

Exercício 3

Escreva um html chamado image.html e um css chamado image.css. Fale sobre o litoral paranaense. Apresente 3 fotos de mesma dimensão. Use 3 classes diferentes. No CSS alterne as propriedades vistas acima. Veja o resultado no browser.

Relembrando elementos em bloco (como `<h1>`, `<p>`, `<li>` e `<div>` por exemplo) são aqueles que forçam quebra de linha antes e depois dele. Já os elementos inline não quebram linha nem antes nem depois. Para transformar um elemento inline em bloco e vice-versa, usa-se a propriedade `display:`. Para inline o valor é `inline` e para bloco o valor é `block`. Veja no exemplo como transformar itens de lista em elementos inline (todos os itens serão mostrados na mesma linha).

```
li { display: inline; }
```

As propriedades de posicionamento permitem posicionar um elemento. Também permitem por um na frente do outro e especificam o que acontece quando o conteúdo de um elemento é muito grande. A primeira propriedade é `position:`. Ela tem 4 valores:

static é o valor default. O elemento é posicionado de acordo com o fluxo normal de preenchimento do browser. O elemento não é afetado pelas propriedades `top`, `bottom`, `left` e `right`.

fixed posição fixa em relação à janela do browser. Ele não é movido mesmo que a janela seja *scrolled*. Veja um exemplo
`p.fixo { position: fixed; top: 30px; right: 5px;}`
Neste caso o elemento é removido do fluxo normal. Os outros elementos são posicionados como se o elemento fixo não existisse.

relative modifica a posição em relação à sua posição normal. Aqui por exemplo, os cabeçalhos h2, serão trazidos para fora da janela:
`h2 { position: relative; left: -20px;}`

absolute posiciona relativo ao primeiro elemento pai que tenha outra posição que não static. Se não existir, será o `<html>`.

Quando elementos são posicionados fora do fluxo normal eles podem sobrepujar outros elementos. A propriedade `z-index:` indica qual a ordem de apresentação. (podem ser atribuídos valores positivos ou negativos). A regra é que um número maior sempre vai aparecer na frente de um número menor. Veja o exemplo:

```
<style>
img { position: absolute; left: 0px;
    top: 0px; z-index: -2; }
</style> </head> <body>
<h1>Um cabeçalho</h1>
<img src="qualquer.gif" width="100" height="140">
</body>
```

Exercício 4

Copie o exemplo acima, usando uma figura qualquer. Varie a propriedade `z-index` e veja o que acontece.

☞ Para você fazer

1. Imprima os arquivos pedidos (brasil htlm e css, tabela htlm e css e image html e css) grampeie-os nesta folha e devolva tudo ou então gere-os no micro na sala de aula e mostre-os ao professor, durante a aula.

2. Descreva o que faz o código a seguir

```
<!DOCTYPE html> <html> <head> <style>
a.five:link {color:#ff0000;text-decoration:none;}
a.five:visited {color:#0000ff;text-decoration:none;}
a.five:hover {text-decoration:underline;}
</style> </head> <body>
<p>Mouse over the links and watch them change layout:</p>
<p><b><a class="five" href="default.asp"
target="_blank">This link changes text-decoration</a></b></p>
</body> </html>
```

3. Descreva o que faz o código a seguir

```
<!DOCTYPE html> <html> <head> <style>
div {background-color: lightblue;
 width: 200px; padding: 25px;
 border: 25px solid navy; }
</style> </head> <body>
<div>Lorem ipsum dolor sit
amet, consectetur adipiscing
elit, sed.</div>
</body> </html>
```

.

106-76182 - ga/ a