

MÓDULO IX

Introdução ao CSS3 e Estilização Básica

MÓDULO IX

HTML ou Body: como definir a largura e a altura como o tamanho total da página

Quando falamos de altura e largura da página, você sabe o que você deve configurar no elemento **HTML**? E no elemento **body**?

Você simplesmente joga os estilos nos elementos e espera que dê certo?

Se faz isso, saiba que não está sozinho.

As respostas para essas perguntas não são intuitivas.

Não é incomum ver propriedades do **CSS** aplicadas tanto aos elemento **HTML** quanto ao elemento **body** dessa forma:

```
html, body {  
    min-height: 100%;
```

```
}
```

MÓDULO IX

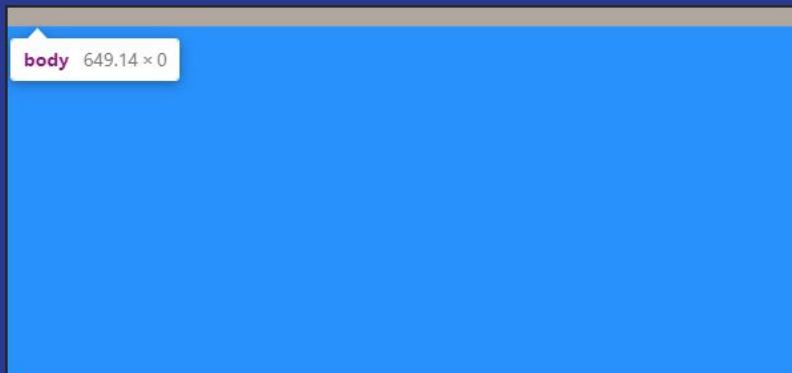
Isso importa?

Sim, importa.

Definir **min-height** como 100% nos dois elementos não permite que o elemento **body** preencha a página como você deveria esperar. Se você verificar os valores de estilo calculados nas ferramentas de desenvolvedor, o elemento **body** tem uma altura de zero.

Enquanto isso, o elemento **HTML** tem uma altura igual à parte visível da página no navegador.

Veja a imagem abaixo, retirada das Ferramentas do Desenvolvedor do Chrome:



O elemento **body** tem uma margem padrão de **8px**, indicada pela barra na parte superior. O valor da altura é o.

MÓDULO IX

Por que isso acontece?

Usar uma porcentagem como o valor para o tamanho requer que o elemento faça referência a um elemento pai para ter a base dessa porcentagem.

O elemento **HTML** faz referência à **viewport**, que tem um valor de altura igual à altura visível da viewport. Porém, definimos apenas uma altura mínima (**min-height**) no elemento **HTML**...

NÃO um valor para a propriedade altura (**height**).

Assim, o elemento **body** não tem o valor de altura do elemento pai para fazer referência quando precisa decidir a que os **100%** estão se referindo.

MÓDULO IX

O problema também pode estar oculto

Se você começou sem conteúdo suficiente para preencher o **body** da página, você pode não ter notado esse problema.

Para tornar tudo mais difícil de perceber, se você definiu uma **background-color** nos dois elementos ou apenas em um deles, a **viewport** está totalmente daquela cor. Isso dá a impressão de que o elemento **body** tem a altura da **viewport**.

Mas não tem. Ele ainda tem altura zero.

MÓDULO IX

A imagem abaixo é retirada de uma página com o seguinte CSS:

```
html, body {  
    min-height: 100%;  
}
```

```
body { background-color: dodgerblue; }
```

Herança inversa?

É uma virada de jogo estranha, mas o elemento HTML recebe a background-color do elemento body se você não definir uma background-color separada para o elemento html.

MÓDULO IX

Como, então, definir uma altura ideal para uma página inteira responsiva?

Por anos, a resposta foi a seguinte:

```
html {  
  height: 100%;  
}  
body {  
  min-height: 100%;  
}
```

Isso permite que o elemento **HTML** faça referência à **viewport** pai e tenha um valor de altura (**height**) igual a **100%** do valor da **viewport**.

MÓDULO IX

Como o elemento **HTML** recebendo um valor para **height**, o valor de **min-height** atribuído ao elemento **body** dá a ele uma altura inicial que corresponde ao elemento **HTML**.

Isso também permite que o **body** seja até maior se o conteúdo tiver uma altura maior do que a da página visível.

O único efeito colateral é o fato de que o elemento **HTML** não consegue crescer para além da **viewport** visível. No entanto, permitir que o elemento **body** seja maior que o elemento **HTML** vem sendo considerado algo aceitável.

MÓDULO IX

A solução moderna é mais simples

```
body { min-height: 100vh; }
```

Este exemplo usa a unidade **vh** (a altura da **viewport**) para permitir que o **body** defina um valor de altura mínima com base na altura total da **viewport**.

Da mesma forma que ocorre com o **background-color** discutido anteriormente, se não definirmos um valor de altura (**height**) para o elemento **HTML**, ele assumirá o mesmo valor de **height** dado ao elemento **body**.

Portanto, essa solução evita o **overflow** do elemento **HTML** presente na situação anterior.

Os dois elementos crescem junto com seu conteúdo!

O uso das unidades **vh** causou alguns problemas nos dispositivos móveis no passado, mas parece que o Chrome e o Safari agora estão consistentes com relação às unidades da **viewport**.

MÓDULO IX

A altura da página pode gerar uma barra de rolagem horizontal

Em outra série estranha de eventos, sua altura da página pode ativar uma barra de rolagem horizontal no seu navegador.

Quando o conteúdo da sua página ficar maior do que a altura da **viewport**, a barra de rolagem vertical à direita é ativada. Isso pode fazer com que sua página também tenha, instantaneamente, uma barra de rolagem horizontal.

MÓDULO IX

Como consertar isso?

Talvez você durma melhor sabendo que isso começa com uma definição da largura da página. Esse problema surge quando qualquer elemento - não apenas os elementos **HTML** e **body** - estiverem definidos com **100 vw** (unidades de largura da **viewport**).

As unidades da **viewport** não respondem pelos aproximadamente **10 pixels** que a barra de rolagem vertical toma para si.

Assim, quando a barra de rolagem vertical é ativada, você também recebe uma barra de rolagem horizontal.

MÓDULO IX

Como definir a largura total da página

Não definir uma largura (**width**) nos elementos **HTML** e **body** deixará o padrão como sendo a largura total da tela. Se você definir um valor de **width** que não seja auto, considere utilizar uma redefinição do **CSS** primeiro.

Lembre-se: por padrão, o elemento **body** tem **8px** de margem em todos os lados.

Uma redefinição do **CSS** remove isso. Do contrário, definir **width** como **100%** antes de remover as margens fará com que o elemento **body** tenha um **overflow**. Esta é a redefinição de **CSS** que eu uso:

```
* {  
  margin: 0;  
  padding: 0;  
  box-sizing: border-box;  
}
```

MÓDULO IX

Como definir uma largura de sua preferência

Embora nem sempre seja necessário definir uma largura (**width**), isso é o que eu normalmente faço.

Pode ser apenas questão de hábito.

Se você definir a largura como **100%** no elemento **body**, terá uma largura inteira de página.

Isso é essencialmente equivalente a não definir um valor para **width** e permitir que seja o padrão.

Se quiser usar o elemento **body** como um contêiner menor e deixar que o elemento **HTML** preencha a página, você pode definir um valor máximo de largura (**max-width**) para o **body**.

MÓDULO IX

Aqui vemos um exemplo:

```
html { background-color: #000; }  
body {  
    min-height: 100vh;  
    max-width: 400px;  
    background-color: papayawhip;  
    margin: 0 auto;  
}
```

MÓDULO IX

Conclusão

Se nenhum valor de **height** tiver sido fornecido para o elemento **HTML**, definir a altura (**height**) e/ou a altura mínima (**min-height**) do elemento **body** como **100%** resultará em nada de altura (antes de você adicionar conteúdo).

Porém, se nenhum valor de **width** tiver sido fornecido para o elemento **HTML**, definir a largura (**width**) do elemento **body** como **100%** resulta em uma largura total da página. Isso pode ser contraintuitivo e confuso.

Para ter uma altura total de página responsiva, defina **min-height** no elemento **body** como **100vh**.

Se você definir uma largura de página, prefira **100%** a **100 vw** para evitar a surpresa com as barras de rolagem horizontais.

MÓDULO IX

Como criar uma lista em HTML? (OL, UL e DL)

Ao criar seu site ou página web com **HTML**, vários tipos de elementos podem ser adicionados, como **imagens, vídeos, textos** e também listas.

Você pode usar uma **lista HTML** para agrupar e exibir itens relacionados.

Por exemplo, você pode criar uma lista com um passo a passo para executar alguma tarefa, ou uma lista de ingredientes para fazer um bolo.

MÓDULO IX

Hoje temos no HTML as listas não ordenadas, ordenadas e de descrição.
Confira esses três tipos de listas renderizadas no navegador:

- Item
- Item
- Item
- Item

1. First item
2. Second item
3. Third item
4. Fourth item

Coffee
- black hot drink
Milk
- white cold drink

MÓDULO IX

Quais os tipos de listas do HTML?

Vamos usar aqui uma lista não ordenada, pois não queremos colocar os itens em sequência e nem colocar descrição para cada um, apenas dizer seus nomes.

- Não ordenadas
- Ordenadas
- Descrição

MÓDULO IX

Como criar uma lista não ordenada no HTML

A tag **HTML ** é utilizada para criar uma lista não ordenada, ou seja, com elementos que não precisam aparecer em sequência.

Para criar cada elemento utilize a tag .

```
<ul>  
  <li>Não ordenadas</li>  
  <li>Ordenadas</li>  
  <li>Descrição</li>  
</ul>
```

MÓDULO IX

Como criar uma lista ordenada no HTML

A tag **HTML ** é utilizada para criar uma lista ordenada, ou seja, com elementos que precisam aparecer em sequência.

Para criar cada elemento utilize a tag ****.

```
<ol>  
  <li>Item 1</li>  
  <li>Item 2</li>  
  <li>Item 3</li>  
</ol>
```

MÓDULO IX

Como criar uma lista de Descrição no HTML <dl>

A tag **HTML <dl>** é utilizada para criar uma lista com elementos que não precisam aparecer em sequência, mas que precisam de uma descrição.

Para criar cada elemento utilize a tag **<dt>**, e para cada descrição a tag **<dd>**.

```
<dl>  
  <dt>Coffee</dt>  
  <dd>- black hot drink</dd>  
  <dt>Milk</dt>  
  <dd>- white cold drink</dd>  
</dl>
```

MÓDULO IX

Tags de lista do HTML

Tag	Descrição
	Define uma lista não ordenada
	Define uma lista ordenada
	Define um item de lista ordenada ou não ordenada
<dl>	Define uma lista de descrição
<dt>	Define um termo da lista de descrição
<dd>	Descreve um termo da lista de descrição

MÓDULO IX

Como formatar uma lista HTML com CSS

Uma lista HTML pode ser formatada visualmente de diversas formas com regras CSS.

Não é o objetivo deste post mostrar essas formas, mas vamos conferir um exemplo básico de como trocar o ícone da lista não ordenada para quadrado:

```
<style>  
  li {  
    list-style: square;  
  }  
</style>
```

MÓDULO IX

Margens, Bordas e Box Model

Uma página é feita de blocos. Estes blocos são "**empilhados**" de cima para baixo, de acordo com a ordem do código: o que está no início do código fica em cima e os elementos subsequentes vão se acomodando em baixo uns dos outros. Através de **CSS** podemos mudar esta ordem, o que veremos na unidade "Trabalhando com **Layout**". Agora, iremos estudar as principais propriedades que podemos aplicar a estes "**blocos**" que compõem a página.

Cada elemento pode ter uma margem interna (***padding***), margem externa (***margin***) e uma borda (***border***). A forma como as margens e bordas se comportam junto com os elementos constitui o que é chamado de **box model**.

A visualização das margens e bordas pode ser facilitada utilizando extensões de navegador como Firebug e Web Developer Toolbar.

MÓDULO IX

Margens

Vamos trabalhar com uma div com fundo colorido para exemplificar:

```
div.margem {  
    background: #900;  
    color: #FFF;  
}
```

[Visualizar Exemplo](#)

MÓDULO IX

Onde está a **div**? Sem nenhum conteúdo ou **margens/bordas**, nada aparece.
Vamos aplicar uma margem interna:

```
div.margem2 {  
  background: #900;  
  color: #FFF;  
  padding: 10px;  
}
```

[Visualizar Exemplo](#)

MÓDULO IX

A declaração **padding: 10px** coloca margens internas de **10** pixels em cada um dos lados da **div**.

Veja no exemplo abaixo a diferença com conteúdo e como a margem interna é aplicada:

[Visualizar Exemplo](#)

MÓDULO IX

Perceba como as **divs** estão "coladas" uma na outra.

Vamos aplicar `margin: 10px;` apenas na de baixo:

```
div.margem3 {  
  background: #900;  
  color: #FFF;  
  padding: 10px;  
  margin: 10px;  
}
```

[Visualizar Exemplo](#)

MÓDULO IX

Margens externas foram aplicadas a todos os lados da **div**, assim como as internas. Caso queiramos inserir margens em apenas alguns dos lados, podemos utilizar as propriedades **padding-top**, **padding-right**, **padding-bottom** e **padding-left**; para as margens externas, **margin-top**, **margin-right**, **margin-bottom** e **margin-left**.

Exemplo:

```
div.margem4 {  
    background: #900;  
    color: #FFF;  
    padding-left: 10px;  
    padding-right: 10px;  
    margin-bottom: 10px;  
}
```

[Visualizar Exemplo](#)

MÓDULO IX

Ao invés de digitar uma propriedade para cada lado, podemos utilizar a forma abreviada, ou ***shorthand***, das propriedades **margin** e **padding**.

Funciona da seguinte forma:

- Para aplicar um valor para todos os lados, apenas digitamos um valor, como **padding: 10px;**.
- Para aplicar um valor para cima/baixo e um para esquerda/direita, escrevemos dois valores: **padding: 5px 10px;**. O primeiro será aplicado para cima/baixo e o segundo para esquerda/direita.
- Para aplicar um valor para cima, um para esquerda/direita e um para baixo, escrevemos três valores: **margin: 5px 10px 20px.** O primeiro será aplicado para cima, o segundo para esquerda/direita e o terceiro para baixo;

MÓDULO IX

- Para aplicar um valor para cada lado, utilizamos a ordem cima/direita/baixo/esquerda (sentido horário partindo de cima), desta forma: **margin: 5px 10px 15px 30px**. O primeiro valor será aplicado para cima, o segundo para direita, o terceiro para baixo e o quarto para esquerda.

Podemos também utilizar outras unidades além de **pixels**, como %, em (unidade relativa ao tamanho da fonte no elemento), **pt** (pontos, mais utilizado para impressos), **cm** (idem), entre outras.

MÓDULO IX

Bordas

Com a propriedade border, podemos adicionar bordas ao redor de qualquer elemento.

Primeiro, vamos conhecer os componentes desta propriedade.

No exemplo abaixo, aplicamos uma borda no topo de um parágrafo:

```
p.borda1 {  
    border-top-width: 3px;  
    border-top-style: solid;  
    border-top-color: #000;  
}
```

MÓDULO IX

Podemos resumir as três declarações acima em apenas uma (***shorthand***), como o exemplo abaixo:

```
p.borda1 {  
  border-top: 3px solid #000;  
}
```

[Visualizar Exemplo](#)

MÓDULO IX

O mesmo vale para **border-right**, **border-bottom** e **border-left** e suas sub-propriedades, como **border-left-color**, **border-bottom-style** ou **border-right-width**.

Para controlar as bordas em um elemento inteiro, podemos utilizar as propriedades **border-width**, **border-style** e **border-color**:

```
p.borda2 {  
    /* a ordem dos lados é a mesma que com margens */  
    border-width: 2px 5px 3px 6px;  
    border-style: dashed dotted;  
    border-color: #000 #090 #900 #009;  
    padding: 20px;  
}
```

[Visualizar Exemplo](#)

MÓDULO IX

Finalmente, podemos utilizar a propriedade border para definir as bordas uniformemente no elemento:

```
p.borda3 {  
  border: 1px solid #000;  
}
```

[Visualizar Exemplo](#)

Com a propriedade border, primeiro especificamos a sua espessura.

Neste caso, utilizei **pixels**, mas você pode utilizar qualquer outra unidade ou palavras-chave como **thin** (fina), **medium** (média) e **thick** (grossa). Depois, escolhemos um tipo de borda.

Os principais tipos são: **solid** (sólida), **dashed** (tracejada) e **dotted**.

Há outros, como **double**, **inset** e **outset**.

MÓDULO IX

Box Model

Por padrão, um elemento como **p** ou **div** irá ocupar todo o espaço horizontal disponível.

Podemos dizer, em propriedades **CSS**, que temos **width: auto; height: auto;**, ou seja, que a largura e a altura são determinadas automaticamente, sendo que a largura por padrão será **100%** na maioria dos casos e a altura será determinada pela quantidade de conteúdo dentro do elemento.

Um bloco como um **p** pode ter até quatro áreas distintas na sua composição.

Da mais externa para a mais interna: margem externa, borda, margem interna e conteúdo.

[Visualizar Exemplo](#)

MÓDULO IX

Perceba que, ao aplicar margens e bordas no parágrafo, elas diminuíram a área de conteúdo do parágrafo. Ou seja, a largura total do parágrafo continua a mesma enquanto as margens e bordas **subtraíram a área útil do elemento**.

Isto ocorreu porque por padrão a largura e a altura estão sendo calculadas automaticamente.

Vamos agora definir uma largura de **300 pixels** e uma altura de **150 pixels** para o parágrafo, além de uma cor de fundo para facilitar a visualização:

```
p.boxmodel1 {  
  width: 300px;  
  height: 150px;  
  background: #900;  
  color: #FFF;  
}
```

[Visualizar Exemplo](#)

MÓDULO IX

O que acontece se colocarmos margens internas e bordas?

```
p.boxmodel2 {  
  width: 300px;  
  height: 150px;  
  background: #900;  
  color: #FFF;  
  padding: 20px;  
  border: 5px solid #CCC;  
}
```

[Visualizar Exemplo](#)

MÓDULO IX

Compare este exemplo com o anterior. As margens e bordas **adicionaram à largura e altura do elemento**. Esta é uma noção muito importante: quando trabalhamos com dimensões automaticamente definidas, as margens e bordas irão se adaptar às dimensões do elemento; se utilizamos largura e altura definidas, margens e bordas irão adicionar a estas dimensões.

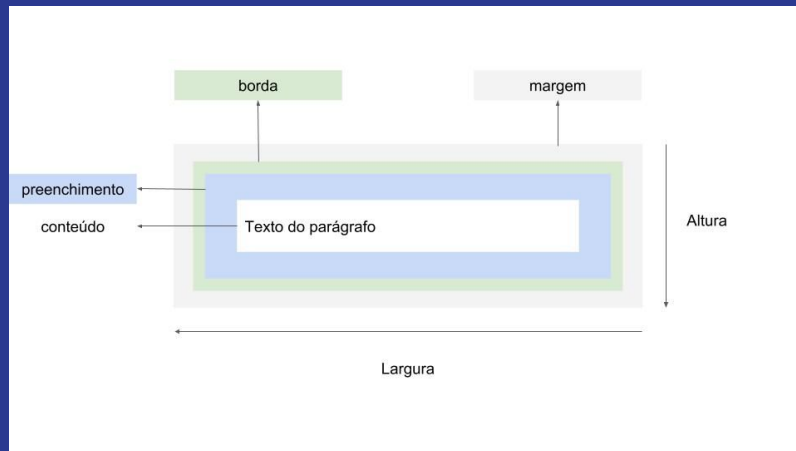
Veja os parágrafos lado a lado para melhor comparação:

[Visualizar Exemplo](#)

MÓDULO IX

Box Model

O motor de renderização do navegador representa cada elemento como uma caixa retangular, de acordo com o padrão definido pelo **CSS** conhecido como **box** (caixa) **model**. Dessa forma, o conteúdo do elemento é uma das quatro partes que compõem o **box**, sendo as demais o seu preenchimento, **borda** e **margem**.



MÓDULO IX

Na maioria dos casos, o que vemos é apenas o conteúdo do elemento, geralmente um **texto**, **imagem**, **vídeo**, etc. Esse conteúdo é o que consideramos as suas dimensões, **altura** e **largura**.

Entretanto, é o conteúdo somado a **margem**, **borda** e preenchimento do elemento que determinam o espaço que o mesmo ocupa na tela do navegador. Por essa razão entender o **box model** é fundamental para a criação de layouts consistentes.

Esta seção trata do controle das áreas que compõem um **box: margin, padding, width e height**.

MÓDULO IX

Width e Height

As propriedades **width** e **height** permitem controlar a **altura** e **largura**, como os nomes sugerem, da área de conteúdo de um elemento. Abaixo vemos um exemplo no qual um parágrafo recebe **16 pixels** de **altura** e **20** de **largura**:

```
div {  
    height: 16px; width: 20px; display: inline-block;  
}
```

MÓDULO IX

Algumas vezes pode ser necessário ter maior controle sobre as dimensões de um elemento, especificando o quanto ele poderá crescer ou se encolher na página. Fazemos isso com as propriedades **min-width**, **max-width**, **min-height** e **max-height**, como no exemplo abaixo:

```
div {  
  
    width: 100px; min-width: 80px;  
  
}
```

A forma como o valor dessas propriedades afetam o elemento depende do valor da propriedade **box-sizing**. Essa propriedade define como a **altura** e **largura** de um elemento são calculadas.

MÓDULO IX

O valor padrão de **box-sizing** é **content-box**, que define que a **largura** e **altura** de um elemento serão definidas pelos valores atribuídos para as propriedades **height** e **width**:

```
div {  
    height: 16px; width: 20px; padding:  
    16px; box-sizing: content-box;  
}
```

No caso acima, as dimensões finais do elemento serão **48 pixels** de altura (**height** de **16px** + **padding** de $2 * 16px = 32px$) e **52 pixels** de largura (**width** de **20px** + **padding** de $2 * 16px = 32px$) pois os valores das propriedades (**height** e **padding**) e (**width** e **padding**) serão somados quando o motor de renderização do navegador os apresentar.

MÓDULO IX

O segundo valor possível para essa propriedade é **border-box**, que define que a **altura** e **largura** de um elemento deve levar em consideração também a sua borda e preenchimento.

```
div { height: 16px; width: 20px; padding:  
16px; box-sizing: border-box; }
```

MÓDULO IX

Neste caso, uma vez que definimos o **padding** do elemento como **16 pixels**, a altura final do elemento será de **32 pixels**, considerando que este valor corresponde a área de preenchimento superior e inferior (**16 + 16**). Sendo a altura, **height**, de **16 pixels**, ela será considerada dentro dos **32 pixels** já calculados a partir da área de preenchimento. Isso acontece pois no **box-sizing** o elemento tende a se encolher para se ajustar ao espaço correspondente às suas dimensões. Uma vez que a altura, **height**, do elemento supere os **32 pixels** definidos pelo **padding**, por exemplo ao receber **33 pixels**, a área de conteúdo do elemento passará a ter **1 pixel** e a altura final do elemento passará a ser um pixel maior (**33 - 32 = 1**). O mesmo se aplica a largura, **width**.

MÓDULO IX

Padding

A propriedade **padding** controla a área de preenchimento de um elemento, que envolve o seu conteúdo. Ela pode receber como valor uma lista de dimensões, como no exemplo abaixo:

```
div { padding: 16px 10px 20px 30px; }
```

Neste caso, **16 pixels** corresponde a área superior, **top**, **10** pixels a direita, **20** a área inferior e **30** a esquerda, sendo a distribuição feita como os ponteiros de um relógio.

MÓDULO IX

Podemos ainda escrever essas propriedades de forma curta, sendo isso útil em alguns casos nos quais desejamos atribuir o mesmo valor para a parte superior e inferior de uma elemento, tal qual para sua área de preenchimento esquerda e direita:

```
div { padding: 16px 10px; }
```

No exemplo acima, a altura da área de preenchimento do elemento terá **32 pixels (16 + 16)**, enquanto sua largura será de **20 pixels**.

MÓDULO IX

Existem outras formas de escrever os valores dessa propriedade, por exemplo utilizando três valores:

```
div { padding: 16px 10px 20px; }
```

Assim, definimos os valores para as áreas superior, esquerda e direita ao mesmo tempo, e inferior. Contudo, alguns navegadores podem ter dificuldades em ler essas sintaxe e, portanto, devemos evitá-la.

Vale lembrar que o valor **border-box**, da propriedade **box-sizing**, influência na forma como **padding** funciona, neste caso empurrando a área de conteúdo do elemento para dentro.

MÓDULO IX

Margin

A propriedade **margin** controla a área de margem de um elemento, que influencia no espaçamento entre esse e os demais elementos ao seu redor. Ela pode receber como valor uma lista de dimensões, como no exemplo abaixo:

```
div { margin: 16px 10px 20px 30px; }
```

Neste caso, **16 pixels** corresponde a área superior, top, **10 pixels** a direita, **20** a área inferior e **30** a esquerda, sendo a distribuição feita como os ponteiros de um relógio.

MÓDULO IX

Podemos ainda escrever esses valores de forma curta, sendo isso útil em alguns casos nos quais desejamos atribuir o mesmo valor para a parte superior e inferior de uma elemento, tal qual para sua área de margem esquerda e direita:

```
div { margin: 16px 10px; }
```

No exemplo acima, o elemento terá uma distância de **32 pixels** (**16 + 16**) daqueles acima e abaixo dele, enquanto os lados da área de margem criarão uma distância de **20 pixels**.

MÓDULO IX

Existem outras formas de escrever os valores dessa propriedade, por exemplo utilizando três valores:

```
div { margin: 16px 10px 20px; }
```

Assim, definimos os valores para as áreas superior, esquerda e direita ao mesmo tempo, e inferior. Contudo, alguns navegadores podem ter dificuldades em ler essas sintaxe e, portanto, devemos evitá-la.

MÓDULO IX

Border

A propriedade **border** define o estilo da borda de um elemento. Ela pode receber uma lista de valores de diferentes tipos, como apresentamos a seguir:

```
div { border: 1px solid #eee; }
```

No exemplo acima, o primeiro valor define a largura da borda, que neste caso será de **1 pixel**. O segundo valor informa o estilo da borda. Por fim podemos atribuir a borda uma cor.

MÓDULO IX

Para determinar a largura da borda podemos utilizar a propriedade **border-width**. Ao utilizar essa propriedade, podemos definir individualmente as bordas superior, esquerda, inferior e direita de um elemento:

```
div { border-width: 1px 2px 1px 2px; }
```

Neste caso, atribuímos 1 pixel para a parte superior/inferior e 2 pixels para a borda esquerda/direita do elemento.

MÓDULO IX

Podemos ainda utilizar palavras chave para definir o valor dessa propriedade, sendo elas **thin**, **medium** e **thick**.

```
div { border-width: thin; }
```

Essa propriedade pode receber ainda o valor **unset**, que cancela os valores anteriormente aplicados.

MÓDULO IX

Com a propriedade **border-style** determinamos como a borda do elemento será apresentada.

Ela pode ser sólida com o valor **solid**, pontilhada com o valor **dotted** ou mesmo dupla, se aplicado o valor **dashed**.

```
div { border-style: dotted solid; }
```

Nesse exemplo, o elemento receberá uma borda superior/inferior pontilhada e esquerda/direita sólida.

Podemos ainda utilizar o valor **none**, que cancela qualquer outro anteriormente aplicado.

MÓDULO IX

EXERCÍCIOS...

MÓDULO IX

FIM...